Broadening the Scope of Security Usability from the Individual to the Organizational: Participation and Interaction for Effective, Efficient, and Agile Authorization

Steffen Bartsch

A thesis submitted to fulfill the requirements for the degree of
"Doktor der Ingenieurwissenschaften"
– Dr.-Ing. –

Fachbereich 3
(Mathematik und Informatik)
Universität Bremen

Tag des Kolloquiums: 29. Juni 2012

Gutachter/Reviewers
Prof. Dr.-Ing. Carsten Bormann (Universität Bremen)
Prof. M. Angela Sasse (University College London)

Abstract

Restrictions and permissions in information systems – *Authorization* – can cause problems for those interacting with the systems. Often, the problems materialize as an interference with the primary tasks, for example, when restrictions prevent the efficient completing of work and cause frustration. Problems are not only caused by restrictive permissions, though, but also by permissive ones, for example, from permissions that remain assigned. In this case, the security measure becomes ineffective. Conversely, its effectiveness can also be impacted when staff is forced to circumvent the measure to complete work – typically sharing passwords among each other.

This is the perspective of functional staff and the organization. There are further perspectives involved in the administration and development of the authorization measure. For instance, functional staff need to interact with policy makers who decide on the granting of additional permissions, and policy makers, in turn, interact with policy authors who actually implement changes. If the procedures of the interactions or the activities themselves incur high effort, the authorization measure will be inefficient. Similarly, developers implement the technical authorization mechanisms, and need to interact with other stakeholders to take their problems into account to arrive at usable mechanisms.

To unravel this entanglement of problems and their interrelation, this thesis analyzes the diverse contexts in which authorization occurs, limits the scope to organizational environments, and systematically examines the problems that surround the different perspectives on authorization in organizational settings, deriving requirements and open research questions. Based on prior research and original research in secure agile development, eight principles to address the authorization problems are identified and explored through practical artifacts.

The Authorization Principles aim to foster the *participation and interaction* among involved and affected stakeholders, including *reducing the burden* and making the abstract aspects of authorization understandable by *increasing the concreteness*. Moreover, the mitigations explicitly aim to *integrate approaches from diverse disciplines*, going beyond the currently predominant technical approaches, for example, by applying socio-organizational approaches. Particularly, the *behavior of individuals* in their social setting should be accounted for, and the *security awareness and expertise* of the involved individuals should be increased. To better cope with the dynamics surrounding authorization, it is also suggested to *design the measures for dynamics* and generally aim to *tailor for the context* regarding procedures (formality, centralization) and tools (flexibility, individuals' expertise).

Applying the principles in the practical artifacts and drawing on the respective empirical evaluations, the principles show to be useful in improving authorization measures, even though the degree of usefulness strongly depends on the context of use. The thesis concludes by proposing to apply the principles and its main theme – broadening security usability to the *organizational* – to other areas of information security: fostering the *participation* between and integration of perspectives on the security measure, and opening information security research further to *interdisciplinarity*.

Zusammenfassung

Probleme mit Berechtigungen und Restriktionen in Informationssystemen – der *Autorisierung* – sind weit verbreitet. Häufig treten sie als Störung in den Arbeitsabläufen zu Tage; beispielsweise, wenn eine Aufgabe wegen fehlender Berechtigungen nicht effizient bearbeitet werden kann und diese Frustration hervorrufen. Probleme werden aber nicht nur durch restriktive Berechtigungen verursacht, sondern auch durch zu weitgehende, zum Beispiel, wenn nicht mehr benötigte Berechtigungen zugewiesen bleiben. In diesem Fall leidet die *Effektivität* der Sicherheitsmaßnahme. Die Effektivität kann allerdings auch beeinträchtigt werden, wenn restriktive Berechtigungen die Anwender zwingen, die Autorisierung für ihre Aufgaben zu umgehen, und, als eine Möglichkeit, Passwörter weiterzugeben.

Die geschilderten Probleme sind allerdings nur die der Organisation und der Anwender. Weitere Perspektiven sind die Verwaltung und die Entwicklung der Sicherheitsmaßnahme. Beispielsweise müssen Anwender sich an Verantwortliche wenden, wenn sie zusätzliche Berechtigungen benötigen. Diese beauftragen dann technische Administratoren mit der Umsetzung der Änderungen. Wenn diese Abläufe, Interaktionen und Aktivitäten einen hohen Aufwand verursachen, leidet die Effizienz der gesamten Maßnahme. In ähnlicher Weise müssen sich Entwickler mit Anwendern, Administratoren und Entscheidern austauschen, um deren Probleme in die Entwicklung von benutzbaren Mechanismen einzubeziehen.

Um dieses Knäuel von Problemen und deren Abhängigkeiten zu entwirren, analysiert diese Arbeit die Kontexte, in denen Autorisierung eingesetzt wird. Konkret beschränkt sie sich auf Organisationen als Umgebung und analysiert systematisch die Probleme, die dort im Zusammenhang mit der Autorisierung auftreten. Daraus werden Anforderungen an *benutzbare* Autorisierung und damit verbundene offene Forschungsfragen abgeleitet. Basierend auf bestehenden Ansätzen aus der benutzbaren Sicherheit und der sicheren Agilen Software-Entwicklung werden acht Prinzipien als Lösungsansätze identifiziert, die durch die Entwicklung von konkreten Artefakten untersucht werden.

Die Autorisierungsprinzipien zielen darauf ab, die *Partizipation* der verschiedenen Perspektiven und deren Austausch untereinander zu fördern. Dazu wird insbesondere die *Belastung für den Einzelnen* verringert und das *Abstrakte zur leichteren Verständlichkeit konkretisiert*. Außerdem werden explizit Ansätze aus *verschiedenen Disziplinen* integriert, beispielsweise soziologische und organisatorische Herangehensweisen als Ergänzung der momentan vorherrschenden technischen. Das betrifft zum einen die Berücksichtigung des *Verhaltens von Menschen* in ihrer sozialen Umgebung, zum anderen das generelle *Fördern des Sicherheitsbewusstseins und der -expertise*. Um besser der Dynamik der Umgebung zu entsprechen, wird die Autorisierung als Maßnahme außerdem explizit auf die *Dynamik ausgerichtet* und strebt generell eine *Angemessenheit entsprechend der Umgebung* in Bezug auf Prozesse (Formalität, Zentralisation) und Werkzeuge (Flexibilität, Erfahrung) an.

Das Anwenden dieser Prinzipien in den praktischen Artefakten und die damit verbundenen empirischen Auswertungen zeigen, dass die Prinzipien generell nützlich sind, um Autorisierung benutzbarer zu gestalten. Allerdings hängt ihre Nützlichkeit im Einzelnen stark von der konkreten Umgebung der Autorisierung ab. Diese Arbeit schließt, in dem sie darlegt wie die Prinzipien und ihr übergeordneter Ansatz – die Ausdehnung der benutzbaren Sicherheit auf das Organisatorische – auf weitere Bereiche der Informationssicherheit übertragen werden kann: Durch das *Fördern der Partizipation* und des Austausches zwischen den Perspektiven auf Sicherheitsmaßnahmen und durch eine größere Offenheit der Forschungsgemeinschaft für *Interdisziplinarität*.

Acknowledgments

I would like to thank my supervisor, Prof. Dr.-Ing. Carsten Bormann for triggering the research work, bringing up the lack of flexibility of information systems in early discussions and reassuring my research direction over the entire process. While working on the research, I enjoyed the discussions on technical and organizational aspects of security with my colleagues in the Computer Networks research group and at TZI's Concerted Activity 'Software Quality and Information Security' (SQuIS), particularly the cooperation with researchers from the Software Engineering group. I am particularly grateful for the support, advice, and encouragement of Dr. Karsten Sohr, and the fruitful discussions on project proposals and the thematic orientation of SQuIS that also steered this research.

I am grateful that the Computer Networks group and SQuIS enabled my research visit with Prof. Angela Sasse's group at the University College London (UCL), which broadened my view on security-usability research – and the scope of the dissertation. I much benefited from the discussions with Angela and her research fellows and students, and from observing their empirical research.

I would also like to thank my friends who again and again helped me to avoid a complete immersion in the research work. Finally, I am much indebted to my parents whose support and relaxed upbringing, among others, allowed me to enjoy a relaxed education and thus enabled this work in the first place.

If the customs and practices of the ancestors will have no authority, life would be dangerous, treacherous, and unsafe.

Cicero: In Verrem, 70 BCE (after Takács, 2009)

Contents

1. Introduction

Authority extends beyond [coercive] powers

Joseph Raz: Authority (1990, p. 15)

Authorization protects resources in information systems[1] as a security measure by restricting permissions. Authorization also denotes the act of granting authority to individuals or other actors to allow them to interact with the system. Even though authorization has been an integral part of protecting information and systems since early in the computing age[2], this measure continues to trouble individuals in their daily interaction with systems. One typical problem is that if an individual lacks a permission to carry out a task, authorization inevitably interferes with the task and causes frustration (Whalen et al., 2006).

Even before the advent of computer systems, problems with the granting of authority have affected societies. In ancient Rome, the authority of Emperors was based upon a complex system of public opinion (Takács, 2009). Since Cicero, Emperors were seen as "the public figure who upheld all that was honorable, virtuous, and worthy of imitation" (p. xx). Their *natural authority* allowed them to rule. When they failed to uphold the virtues and their constructed narratives collapsed, the political system struggled[3]. Modern political philosophers, such as Raz (1990), consider state authority a paradox: People find it "deeply disturbing... that one person should have the right to rule another" (p. 3). One argument for legitimizing state authority is the need of the community to be ruled and that complying is in the best interest of the members of the community (p. 5), particularly with respect to its coordination. The problems surrounding authorization in information systems involve authority of finer granularity and thus different types of decisions. However, we will see in the course of this dissertation that the underlying conflict of the adequacy of authority and the legitimacy of restrictions play an important role in building effective and efficient authorization measures.

The problem with the adequacy of restrictions points to conflicts between the security measure and the individual's primary task. One cause are the technical mechanisms in information systems when they enforce a rigidness of rules that is unknown from non-technical areas. Before specifically focusing on authorization, we need to consider the broader scope: Problems with security measures have been known for more than a century; in the context of military cryptography, Kerckhoffs (1883) incorporated them in his sixth principle:

> "Finally, regarding the circumstances in which such system is applied, it must be easy to use and must neither require stress of mind nor the knowledge of a long series of rules."

Kerckhoffs emphasizes the circumstances of application. One important difference between Kerckhoffs' military context and business or leisure activities relates to the priorities that are assigned to

[1] "Information systems" is a heavily overloaded term. In this thesis, it denotes computer-based information systems, that is, networks of human actors and technical artifacts (Jessup and Valacich, 2008, p. 12). To instead refer to the research discipline, it is capitalized or abbreviated "IS".

[2] Multi-user systems emerged in the 1960s and required the protection of systems from users and of users from other users (Graham, 1968).

[3] One example is Nero's "performative act" as a ruler (Takács, 2009, p. xx).

the securing of information and systems. While security breaches can critically harm businesses and privacy incidents can seriously impact individuals, the primary goal of organizations and individuals when interacting with a system is not security, but the original task. The effect of security having a secondary role is furthered by the primary goals being more tangible and appearing more concrete than the risks from security incidents (West, 2008).

In authorization, the underlying tension occurs between three distinct security goals. *Availability* should ensure that information and systems can be read or interacted with if necessary. In contrast, *confidentiality* should guarantee that information is only retrieved by those that should, and *integrity* that information is only modified as wanted. Configuring permissive restrictions will make the availability of resources more likely for those that need them. At the same time, permissive restrictions could enable unwanted access or modification. The practical relevance of this tension could be observed in the disclosure of sensitive United States military and diplomatic documents by WikiLeaks in 2010. Following the September 11 attacks, the failure to discover the attack plans was attributed to the restrictions on information sharing between government agencies. The National Commission on Terrorist Attacks Upon the United States (2004) stated in the “9/11 Report” that it is no longer the case that the “risk of inadvertent disclosure outweighs the benefit of wider sharing.” This realization led to the expansion of access to government databases to foster the sharing of information between government agencies. The resulting higher availability of information is seen as one factor that enabled the disclosure of the sensitive documents (BBC News, 2010).

To resolve the conflict between the security goals, this dissertation will argue, we need to improve the usability of the security measure: A second factor in the WikiLeaks disclosure was the lack of usability of the security measures in place so that the measures were not applied effectively. As Kerckhoffs (1883) emphasizes, a security measure must fit the context of use (Sasse, 2011; Sasse et al., 2001). However, since Kerckhoffs’ time, the application of security technology, and of authorization in information system in particular, has changed significantly: The complexity of the context of use increased dramatically. The context is rarely as predetermined as in Kerckhoffs’ military environment, but rather highly diverse and dynamic. Individuals who interact with systems, and have to cope with and configure restrictions, have become significantly more diverse with the spreading of networked systems in the 1990s and the popularity of the Web. Instead of highly-specialized professionals that worked with and configured computer systems exclusively for a long time, a large proportion of the work force and society is now impacted by and configures restrictions, whether for the shared folder at work or Social Network privacy settings at home. “Networked” environments (Castells, 2000) and the trend to “interwoven” computing (Pallas, 2009) lead to contexts that are difficult to specify or predict. The dynamics can particularly be observed for organizations (Truex et al., 1999) and require the restrictions to be frequently adapted to keep up with the organizational changes (Sinclair et al., 2008).

The complexity and dynamics of the contexts are the primary reason why it is difficult to comprehensively enforce authorization by technical means. It is difficult to model adequate restrictions in algorithmic terms. In terms of organizational economics, trying to comprehensively restrict access through technical means will result in high coordination costs of adapting the restrictions or in the loss of staff productivity if the measures interfere with work (Pallas, 2009). Considering the complexities and the risks associated with information and systems due to the reliance of organizations and society on networked systems for business and critical infrastructure, what is the way forward to achieve effective and efficient authorization measures? Can we find inspiration from how authority is addressed in other fields?

1.1. Widening the scope

In case of *political* authority, many modern societies only allow the state to employ coercive powers to a limited extend to prevent negative collateral effects and to preserve the coercive powers' efficacy, that is, their being obeyed more often than not (Hart, 1994, p. 103). Concerning authorization, the coercive power of laws can serve as an analogy to the technical mechanisms that interfere with primary tasks. If coercion is used excessively or inappropriately by governments, citizens will circumvent the laws and their enforcement. Hart (1994) notes that "Laws may be condemned as morally bad simply because they require men to do particular actions which morality forbids individuals to do." (p. 168) Hall (1971) gives the example of civil disobedience to rigorous abortion laws in the U.S., citing a physician:

> "where blatant injustice does exist, as is the case with the present abortion laws, I believe it is the duty of all…to actively help the people who are the present victims of such unjust laws." (p. 104)

For authorization, we will see in Chapter 5 that functional staff are likewise creative in mitigating authorization problems that interfere with their work and find ways around technical restrictions, sometimes in dangerous or inefficient ways, for example, when sharing passwords with colleagues. It is as challenging to rely solely on technical means for authorization in information systems as it is to use pure coercion for the state.

A common approach by democratic governments is to coerce citizens only for matters for which the citizens generally agree that coercion is legitimate. Locke (1728) argues that "nothing [is] able to put [man] into subjection to any earthly power, but only his own consent." (p. 223)[4] In authorization measures, this can be likened to the acceptability of the measures and depends on the perceived adequacy of the imposed restrictions. However, until the late 1990s, research on authorization primarily focused on the technical means of the models and mechanisms (Siponen and Oinas-Kukkonen, 2007), and neglected that humans need to interact with the systems and need to configure the restrictions. This is part of the challenges in information security research: To achieve the re-orientation from the often purely technical to the socio-technical (Dhillon and Backhouse, 2001). Beginning with Zurko and Simon (1996), the academic community started to consider socio-technical factors for authorization, first in laboratory experiments on the usability of the configuration tools (Zurko et al., 1999) and on the comprehensibility of authorization concepts (Brostoff et al., 2005) and configurations (Rode et al., 2006). One recurring theme is how difficult the abstract representation of restrictions is (Blackwell et al., 2008). Since the organizational practice is more complex than the isolated editing of restrictions and, for instance, also involves coordination, a small number of studies have recently been published on how authoring of restrictions is hindered in practice (Whalen et al., 2006; Bauer et al., 2009; Smetters and Good, 2009).

However, focusing on formulating adequate restrictions can be likened to only address the writing of adequate laws and neglect the legislative process. Instead, further socio-technical factors need to be considered. For example, to adequately modify restrictions, administrators in organizations need to be in a position to judge which permissions are needed and which would be dangerous. For Social Networks, Acquisti and Grossklags (2005) show how users do not decide rationally on the restrictions, but trade short-term benefits for long-term privacy effects. As West (2008) argues,

[4]Weber (1968) is more precise here and differentiates between Power (*Macht:* the probability that will is carried out despite resistance) and Domination (*Herrschaft:* the probability of obedience without coercion): "[E]very genuine form of domination implies a voluntary compliance, that is, an interest…in obedience" (p. 212). Since domination is more efficient than directly enforcing power, those with power "employ" legitimacy – for example, every highly privileged group develops a myth of its superiority (p. 953), as in the case of the Emperors of Rome.

decision-making is difficult in information security since primary tasks often appear more concrete and tangible and thus receive more attention than the seemingly abstract risks.

Selectively focusing only on the managers deciding on and administrators configuring restrictions still results in an incomplete picture, though. Firstly, this only considers centralized management of authorization and neglects the common case of delegated decision-making, for instance, when individuals decide on who is allowed to read a specific document in the organization. Secondly, and more fundamentally, this ignores advances in organizational culture of the last century, when the traditional model of top-down management gave gradually way to a "Human Resource Model" in which all members of an organization "contribute to the limits of their ability" (Miles et al., 1978). For adequate restrictions in authorization, we need to leverage this potential. Organizationally, this may be achieved by integrating authorization into the workflow of functional staff (Whalen et al., 2006). However, an additional problem is that "different groups have different ways of knowing" (Brown et al., 1993). For effective integration of different groups, Brown et al. (1993) thus argue for adequate measures, aiming for "informed participation", since "to participate fully you must be fully informed, and to be fully informed, you must participate."

Returning to the aspect of the adequacy of restrictions, we have to consider that its perception is relative and is also affected by motivational factors, such as security awareness. We not only need to take the organization's perspective into account (the optimal security/efficiency trade-off), but also regard the motivation and behavior of staff (Albrechtsen, 2008; Sasse et al., 2001). When restrictions are adequate from the organization's perspective but unacceptable for staff, one option may be to explain the rationale behind restrictions to create understanding. In this way, staff may internalize the externalities of the negative consequences for the organization (Cooter, 2006) and might thus be more likely to accept interferences.

However, it might not be in the best interest of organizations to make employees always comply with technical measures, since business goals at stake may overrule the risks in individual instances. For instance, it is too costly to have staff unable to work because they do not have a specific permission after changing departments. For political authority, Thomas Jefferson even stated that "To lose our country by scrupulous adherence to written law, would be to lose the law itself,. . . thus absurdly sacrificing the end to the means" (Hall, 1971). Hart (1994) uses the term *Rule skepticism* to describe how jurisdiction solves these issues:

> "In fact all [legal] systems, in different ways, compromise between two social needs: the need for certain rules which can, over great areas of conduct, safely be applied by private individuals to themselves without fresh official guidance or weighing up of social issues, and the need to leave open, for later settlement by an informed, official choice, issues which can only be properly appreciated and settled when they arise in a concrete case"

Transferring this paradigm to organizational authorization, we need to consider the full range of how the risks from a misuse of information system can be reduced, not just technical means (Latour, 1991). In sociology, the Circuit of Power model by Clegg (1989) describes how direct commands, social norms, and formal rules or technical means interact to achieve overall authority. In organizations, formal and informal rules can complement technical mechanisms to loosen the reliance on technical means and on the adequacy of the encoded restrictions (Pallas, 2009). Whalen et al. (2006) recommend that technical measures support existing social controls, resulting in more permissive restrictions or more flexible authorization measures.

Increased flexibility is one example of how the overall usability of authorization depends on the developer to design and implement adequate mechanisms in information systems. Further critical aspects include the expressiveness or comprehensibility of the restrictions, which can cause errors

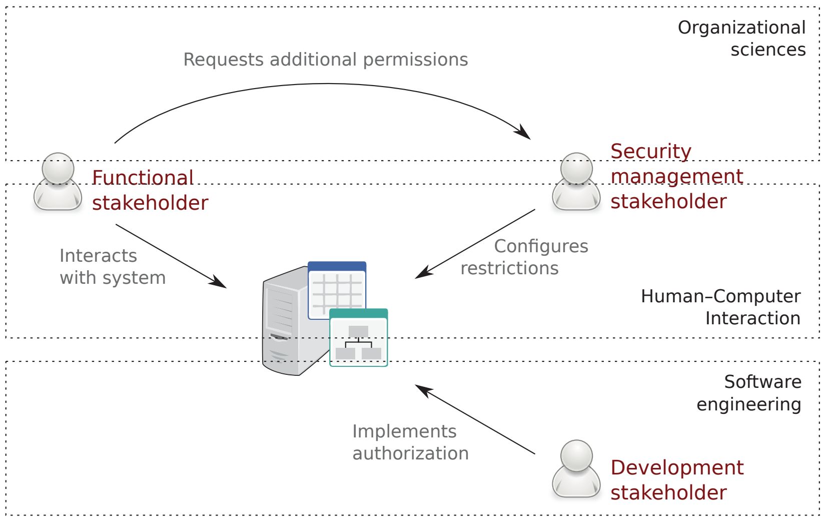


Figure 1.1.: Perspectives in authorization and a selection of relevant research disciplines

or inefficiencies in their management (Bauer et al., 2009). This is in analogy to the interrelations between laws as the restrictions that depend on adequate means for enforcement, for example, technically in case of road traffic safety: speed traps and central registers for road traffic offenses. Similarly, developers implementing authorization in information systems need to closely interact with decision makers on how restrictions are enforced in the system (He and Antón, 2009).

We touched upon the perspectives of users being restricted by, administrators configuring, and developers implementing authorization in the course of this section. The respective perspectives of these *stakeholders* can be sketched as follows (cf. Figure 1.1):

- *Functional stakeholders* rely on the information systems, the supported processes, and contained data to complete their tasks – both in direct interaction with the systems and indirectly as managers supervising the use of the systems,
- *Security management stakeholders* make the decisions of what restrictions to enforce (management) and configure the restrictions in the systems (administration),
- *Development stakeholders* decide upon and implement authorization in information systems, and integrate information systems in organizational infrastructures.

Stakeholders of each of these perspectives undertake specific activities, can face respective problems surrounding authorization, and take decisions that may cause problems for other stakeholders with other perspectives, as shown in the examples in Table 1.1. In this section and in the table, we can observe how the problems of these perspectives interrelate: For example, the functional user might not be able to complete a task because of the restrictions that security managers implemented accidentally due to the incomprehensible restrictions. Hence, we may need to integrate the perspectives to achieve an adequate measure. We also saw how technical, organizational, and socio-technical approaches need to be combined for effective and efficient authorization measures (Figure 1.1 shows three obviously relevant disciplines). However, there apparently is a bias towards technical research in prior research on authorization (Siponen and Oinas-Kukkonen, 2007; Dhillon and Backhouse, 2001), preventing the research community from solving the problems more fundamentally. This thesis pursues to counter the effects and integrate the perspectives and research areas to improve the effectiveness and efficiency of authorization.

Perspective	Typical activity	Challenges	Impacts
Functional	Carry out work tasks	Hindered by restrictions, circumvention of system	Reduced productivity, ineffective security, respectively
Security management	Formulate restrictions, configure authorization	Incomprehensible restriction configuration	Inadequate restrictions
Development	Implement authorization in system	Unknown expertise of security managers	Incomprehensible restriction configuration

Table 1.1.: Examples of impacts on authorization usability

1.2. Integrating perspectives and research areas

From the discussion of the problems surrounding the usability of authorization and potential mitigations, the following two leading **research questions** emerge:

1. *What authorization problems do stakeholders face in information systems and how do their problems and perspectives interrelate?*

2. *What methods, procedures, and technologies are required to address the authorization problems and achieve usable authorization measures in information systems, and what fields of research do we need to draw upon?*

The core approach of this thesis is to integrate the different perspectives and research areas for the analysis of the problems and their mitigation. More specifically, to guide the research, we can formulate hypotheses on how authorization problems manifest themselves (primarily covered in Chapter 5) and how they can be addressed (Chapters 7 to 10):

Problem manifestation

H 1 *Stakeholders face usability problems surrounding authorization from diverse and interrelated perspectives on authorization*

The example problems outlined in Table 1.1 indicate that stakeholders have different perspectives on authorization and that the usability issues of the perspectives interrelate. This can be observed for the impact of the security management perspective, inadequate restrictions, and its recurrence as an issue for functional stakeholders. It is thus necessary to particularly analyze what perspectives exist and how the respective problems interrelate to improve the usability of authorization measures.

H 2 *The interaction of stakeholders between perspectives regarding authorization suffers from problems surrounding authorization*

Another observation from the above examples is that the problems interfere with the integration of the different perspectives concerning authorization. The challenge of inadequate restrictions for the functional stakeholders in Table 1.1 is a case in point: Here, enabling the functional stakeholders to enter the security management realm of configuring restrictions may reduce friction of communication between perspectives. However, to foster the interaction, we need to solve problems with authorization in the first place – for example, through more comprehensible configuration tools.

Problem mitigation

H 3 *Technical restrictions must be combined with non-technical approaches to improve the effectiveness of authorization*

Technical means are only part of the puzzle to achieve effective authorization. For example, when restrictions are perceived as inadequate, the effectiveness of the authorization measure can be impaired. This can be observed in Table 1.1 for the functional staff, who circumvent the authorization and thus reduce the overall security. One approach is to integrate technical mechanisms with further organizational and social measures, including to establish rules of conduct and to increase motivation for compliance through the awareness of the concrete risks.

H 4 *Authorization measures that are designed to handle dynamic environments improve the adequacy of restrictions*

As discussed in the beginning of this chapter, the context of authorization measures, such as organizational structures, business processes, and information systems, constantly change. In the example of the challenges for the functional stakeholder, changing work tasks would require the restrictions to be adapted to reflect these changes to remain adequate and prevent interferences with work tasks. Authorization measures need to explicitly account for the dynamics – for example, through the addition of non-technical rules to the technical restrictions – to adequately handle the dynamics.

H 5 *Involving affected stakeholders in the decision-making and authoring of restrictions improves the adequacy of the restrictions*

Functional staff often have the most precise information about what permissions their work tasks require. Selectively relying on managers or administrators may thus prevent comprehensive decisions and reduce the adequacy of the restrictions. Instead, we need to take the perspectives and goals of the organization, and of the affected and deciding stakeholders into account. We should strive to integrate all stakeholders with "informed participation" (Brown et al., 1993), for example, through concrete decision-support, comprehensible authoring tools, and light-weight procedures for changes.

1.3. Drawing on security usability and software engineering

The hypotheses name the central problems to address in this thesis and how we will approach them. While the approaches require us to draw from several research areas, we will particularly focus on two successful areas that address related problems in an attempt to identify principles to apply. The first area is *security usability*, that is, employing methods and techniques from human–computer interaction (HCI) research to improve the effectiveness, efficiency, and satisfaction of stakeholders interacting with security measures (Chapter 2). Security usability has incorporated numerous research approaches since its inception as a research strand, beginning with the application of usability engineering techniques to identify shortcomings of security designs and mitigations (Zurko and Simon, 1996; Adams and Sasse, 1999). Since then, the security usability community has striven to better understand mental models of security measures and to develop better mechanisms (Whitten, 2004; Brostoff et al., 2005). Researchers recently began to focus on the behavior of individuals and their motivation through economic impacts (Beautement et al., 2008; Herley, 2010). From security usability, models of individuals' behavior, motivation, and stakeholder interaction are promising starting points for the research in this thesis.

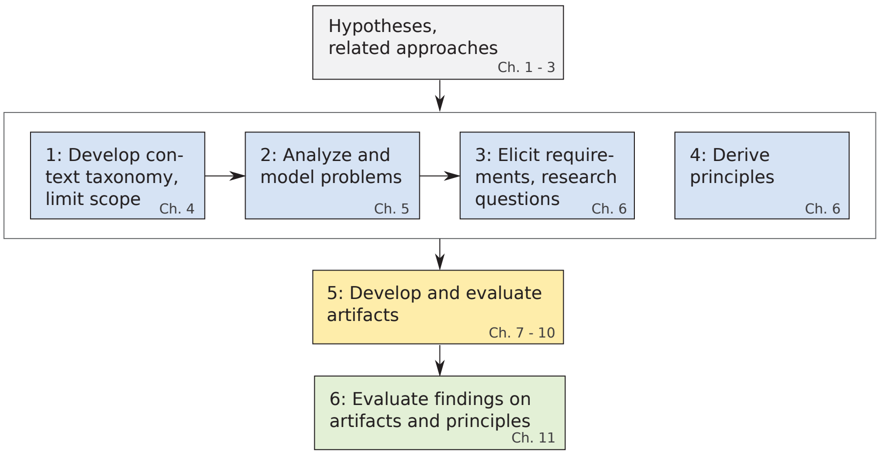

Hypotheses, related approaches
Ch. 1 - 3
1: Develop context taxonomy, limit scope
Ch. 4
2: Analyze and model problems
Ch. 5
3: Elicit requirements, research questions
Ch. 6
4: Derive principles
Ch. 6
5: Develop and evaluate artifacts
Ch. 7 - 10
6: Evaluate findings on artifacts and principles
Ch. 11

Figure 1.2.: Methodology applied in this dissertation

Among other approaches, the hypotheses refer to the interaction of stakeholders and the adaption to changing contexts. Software engineering has to address similar problems in the course of crafting adequate systems, so that it should be useful to explore the approaches in software engineering (Chapter 3). One aspect is how security requirements are integrated into the plan-driven software development process (Section 3.2). Since we are interested in stakeholder integration and in handling the dynamics of the context, we particularly consider agile development (Section 3.3). Agile development methods closely integrate stakeholders from the customer and the different roles in software development. Further key characteristics of agile methods are the empirical approach and the iterative and evolutionary nature to adapt to changing environments. However, agile development was not originally intended for security-critical development and there can be conflicts between agile methods and secure software engineering. Since authorization is security-critical, we discuss how agile methods need to be tailored for (Section 3.4).

1.4. Integrative information security research

The methodology applied in this thesis is depicted in Figure 1.2 and consists of three primary parts: structuring and scoping the problem domain (primarily analytical), analyzing the problems (empirical), and exploring mitigating *principles* through the development of artifacts (practical/empirical):

1.4.1. Structuring and scoping the problem domain

1. *Develop a context taxonomy and scope horizontally:* We systematically define the horizontal range of contexts in a taxonomy and limit the scope to a specific domain by applying the taxonomy.

The hypotheses state that it is necessary to take an integrative approach on the broad range of problems surrounding authorization, both for the perspectives on authorization (e.g. functional, security management, development: H1, H5) and for the research areas (e.g. socio-technical, organizational, software engineering: H3). However, authorization is relevant in a variety of contexts. As sketched in Figure 1.3, we can align the contexts *horizontally*, referring to the breadth of the field. Problems of the distinct perspectives are shown on the *vertical* axis. Since it is unrealistic to cover the entire field, it is common in research to divide the field and select a part to cover. Often, the scoping occurs

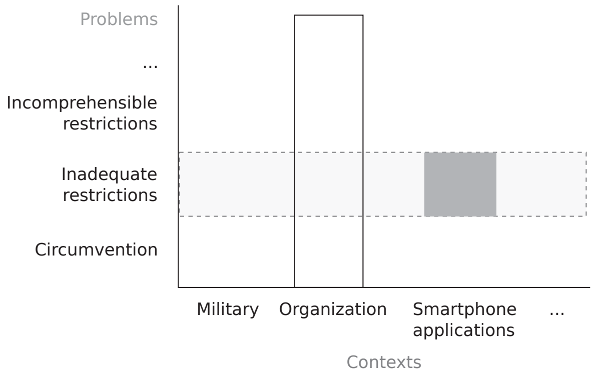


Figure 1.3.: Vertical and horizontal scoping within the field of authorization

along the horizontal axis for a specific problem, as shown for "inadequate restrictions", or one problem in a specific context ("inadequate restrictions" for "smartphone applications"). In our case, these approaches would prevent an integrative approach to analyze the problems of different perspectives on authorization. Accordingly, the approach in this dissertation is to limit the scope vertically, as shown for the context "organization", to allow for effective integrative research.

Following the review of related background material, we develop a *taxonomy* of authorization contexts, and particularly examine organizations, that is, "systems of purposive activity of a specified kind" (Weber, 1947), and limit the thesis' scope to organizational contexts (Step 1). The rationale is that many particularly challenging problems in authorization – the interaction of stakeholders and related organizational issues – are particularly pronounced in organizations.

1.4.2. Analyzing the problems surrounding authorization

2. *Analyze and model problems:* Within the defined scope, we empirically study the problems thoroughly and identify structures within the identified problems and those in literature, modeling perspectives on the security measure and the interrelation between problems.

For a clearer picture of the existing problems with authorization in organizations, we empirically study the problems in a large organization. Combined with problems identified in literature on authorization, we derive a *model of the problems* surrounding authorization in organizations.

3. *Elicit requirements and detailed research questions:* Based on the problem model, we derive requirements for a usable measure and enrich these with existing approaches to identify shortcomings, formulated as open research questions.

The model of the authorization problems allows us to formulate *requirements* for usable authorization measures and, together with existing approaches from literature, *open research questions*.

4. *Derive principles to address the problems:* Approaches from related fields allow us to distill mitigating principles to apply to the problems.

While the hypotheses guide the research effort, the ultimate goal is to develop and evaluate a theory of how to approach the authorization problems. Towards this goal, we draw on the reviewed approaches from security usability and secure agile development to derive *Authorization Principles*.

1.4.3. Exploring mitigations in organizational contexts

5. *Develop and evaluate artifacts:* Applying the principles, we design, develop, and evaluate artifacts that address selected open research questions.

Based on the principles and the analysis of authorization problems, we target selected open research questions and explore mitigations to the problems by developing and evaluating six artifacts as practical contributions:

- An authorization framework to support the implementation of authorization in applications (Chapter 7),
- An associated policy model to involve functional stakeholders in the authoring and decision-making of restrictions (Chapter 7),
- A change support tool that suggests the abstract changes of the formalized restrictions to allow the functional stakeholders to make more concrete changes and better integrate them in the authoring (Chapter 7),
- The implementation and evaluation of more flexible authorization, policy override, in an organizational information system (Chapter 8),
- A process model to evaluate and design adequate and effective authorization processes (Chapter 9),
- A prototype and procedures to support the decisions on what permission to assign (Chapter 10).

6. *Evaluate the findings on the artifacts and principles:* We discuss the implications from the findings on the artifacts and consider the usefulness of the principles, the validity of the hypotheses, and the gained insights on open research questions.

The dissertation concludes with a discussion of the research question and hypotheses. The discussion focuses on the artifacts and, particularly, to what extent the Authorization Principles succeed in addressing the open research questions, and the transferability of the results to other authorization contexts.

1.5. Research contributions

The key research theme of this dissertation is emphasizing the need in information security research to take an integrative approach and rather to focus on a smaller range of contexts than reducing the problem scope to one perspective on a system or one research discipline. The thesis argues that the integrative approach complements more focused research and is necessary to fundamentally solve usability problems in information security.

Methodological contributions This thesis proposes and follows a methodology for integrative research in information security to arrive at comprehensive mitigations for usability problems. The integrative character of the methodology manifests itself on two levels: integrating research strands and research disciplines, and fostering the participation and interaction of involved and affected individuals. Specifically, we apply both approaches in three ways: *theoretically* by deriving and evaluating principles that integrate research and people, *empirically* by analyzing usability problems and evaluating artifacts from multiple perspectives, and *practically* by integrating multiple research strands in developing artifacts that foster the integration and participation of individuals.

Theoretical contributions The primary theoretical contribution of this dissertation are the *Authorization Principles* to address authorization problems in organizations (Section 6.4). The principles bring together several research strands – including research from information security, software engineering, and organizational and socio-technical research. In addition, two analytical frameworks support and structure the research in this dissertation: a taxonomy of authorization contexts (Section 4.1) and its application to organizational contexts (Section 4.2), and a holistic model of the problems surrounding authorization – considering the perspectives of the affected stakeholders, their interrelation, and the interrelation of the problems (Section 5.3).

Empirical contributions Empirical work primarily serves in this dissertation to develop hypotheses on what problems exist in practice – for an analysis of authorization problems in a large organization (Section 5.1) – and to evaluate the application of the Authorization Principles in the artifacts (see below) and derive insights on the open research problems:

- Explore how different representations of authorization restrictions and modes of management support changing restrictions (Section 7.3),
- Study the effects of the proposed authorization framework in agile development (Section 7.5),
- Examine the application of policy override in a medium-sized enterprise (Chapter 8),
- Evaluate the use of the authorization process model to describe processes in organizations (Chapter 9),
- Explore the participatory collection of factors to support authorization decisions (Chapter 10).

In addition, we explore – as part of the background work – how agile practitioners approach security (Appendix C).

Practical contributions The hypotheses on approaches to address usability problems and the derived principles are evaluated by developing and exploring artifacts. This thesis makes four primary practical contributions in six artifacts: the authorization framework with an accessible policy and change support (Chapter 7), the implementation of policy override (Chapter 8), the integrative authorization process model (Chapter 9), and a decision-support prototype for authorization decisions (Chapter 10).

1.6. Publications

A number of publications resulted from the research for this dissertation:

- Bartsch and Sasse (2012). Guiding decisions on authorization policies: A participatory approach to decision support. *27th ACM Symposium On Applied Computing (SAC 2012)*
- Bartsch (2012). Policy override in practice: Model, evaluation, and decision support. In journal *Security and Communication Networks* (Wiley, in print)
- Bartsch (2011c). Exploring twisted paths: Analyzing authorization processes in organizations. *5th International Conference on Network and System Security (NSS 2011)*
- Bartsch (2011b). Practitioners' perspectives on security in agile development. *6th International Workshop on Frontiers in Availability, Reliability and Security* at ARES 2011

- Bartsch (2011a). An authorization enforcement usability case study. *ESSoS 2011*
- Bartsch (2010b). A calculus for the qualitative risk assessment of policy override authorization. Received Best Paper Award at the *3rd International Conference on Security of Information and Networks (SIN 2010)*
- Bartsch (2010a). Supporting authorization policy modification in agile development of Web applications. *Fourth International Workshop on Secure Software Engineering (SecSE 2010)*
- Bartsch, Sohr, and Bormann (2009). Supporting Agile Development of Authorization Rules for SME Applications. *3rd International Workshop on Trusted Collaboration (TrustCol 2008)*
- Bartsch and Bormann (2008). Berechtigungsmodellierung im Geschäftsprozessmanagement von KMU. *D.A.CH. Security 2008*

Part I.

Security in human-computer interactions and software engineering

2. Security usability

First they built the road,
then they built the town…

Arcade Fire: Wasted Hours (The Suburbs, 2010)

As discussed briefly in the introduction, if authorization causes problems – for example, interfering with work or requiring too much effort – functional stakeholders may not the accept the measure and, ultimately, circumvent it. The field of *Security in Human–Computer Interaction* (HCISec) or *Security Usability* addresses problems of this kind. The security goals are often defined as the "CIA triangle" of protecting (Confidentiality, Integrity) and enabling access to information (Availability) (Gollmann, 2011). The trade-off between confidentiality, integrity, and availability indicates that the tension does not lie between security and usability as it is still often proclaimed. Instead, the usability of security lies on middle ground in the continuum between confidentiality, integrity, and availability. Security usability optimizes security measures within this continuum and takes on two perspectives: First, how security measures influence the effectiveness and efficiency of the human–computer interaction, and the satisfaction of affected stakeholders (e.g. how much does security interfere with work); second, how the usability influences the overall effectiveness of the security measures (e.g. does security force stakeholders to behave in an insecure manner). Both sides often interact: If a security measure causes excessive extra work, it might not be employed as intended, thus reducing the overall security effectiveness.

Take, as one example of ineffective security among countless, the results from a classic paper on security usability: "Why Johnny can't encrypt" (Whitten and Tygar, 1999) covers the reasons why individuals cannot use one measure to prevent adversaries from eavesdropping on their emails (encryption) and to assure the receiver that they are the actual source of the unaltered email (digital signature). Specifically, Whitten and Tygar (1999) explored the problems for users to employ public key cryptography for their email correspondence. Among other issues, they demonstrated the burden for the user: the difficulties to understand key validity and trust concepts; and user interface problems, such as information overload. Moreover, the study also showed the ineffectiveness of the measure; participants, for example, accidentally sent their private key in plain text instead of the public one. Evidently, the security measure, including the software and support, was not appropriate for the study participants – they were just not *able* to use it.

However, the ability is only part of the problem: As Adams and Sasse (1999) showed for password usage, the *motivation* plays a crucial role. When the user is convinced of the need for secure behavior, the user is more likely to accept the additional burden to a certain extent. We can thus observe two interrelated effects: Usability problems cause mistakes that lead to the breakdown of the security protection due to the lack of ability, and the burden prevents the effective usage of the measure if there is insufficient motivation.

How, then, do users react if they are *forced* to use tedious, complicated, and error-prone security mechanisms, for example, in organizational settings? How do users balance the conflicting interests of themselves and the organization, in the email case, secure communication against productivity? How do environmental factors, such as time pressure and management pressure for productivity, affect the decision? How can we prevent these kinds of conflicts in the first place? Another example of this conflict is the forging of fax signatures in organizations. Since fax signatures are simple to forge, it is common practice for secretaries to imitate their superior's signature if necessary (Odlyzko, 2003). They thus decide to solve the conflict in favor of productivity and circumvent the security measure of guaranteeing the authenticity of the documents. This kind of "rule bending" is essential to efficient organizations. While it is still possible to alleviate potential losses of productivity caused by the originally intended procedure in this case, a more secure mechanism could close this "hole". A subsequent (more secure) mechanism might result in even more difficult conflicts between dangerous circumventions and degraded productivity. Instead, measures – not only the technical mechanism, but also organizational policy and procedures – need to carefully account for the context of use and, in this case, offer the necessary degree of flexibility.

In addition to the functional perspective on security usability – completing primary tasks –, there are further perspectives, particularly in organizations: Administrators need to configure the numerous security-relevant systems – for instance, comprehend and modify firewall rules or authorization policies. In software development, developers are tasked with implementing complex security mechanisms to fulfill security requirements; developers risk to introduce critical defects when not entirely understanding the underlying security models, protocols, or programming interfaces. Accordingly, security-usability considerations need to encompass not only functional users, but also related perspectives, for example, security administrators and software developers.

This chapter provides an overview over the broad field of security usability, its problems and mitigations. It starts out by describing the field of HCI and software usability. We then discuss the range of security usability problems before exploring the respective mitigations, both generally and focused on authorization. Moreover, we derive principles from the mitigations to apply to our original problems surrounding authorization.

2.1. Human computer interaction and software usability

Since as early as the 19th century, people have explicitly addressed the interaction of humans and machines. The analyses started off from the characteristics of work environment as *ergonomics* (from Latin "ergon", "work"). Ergonomics in the modern, broader sense of humans interacting with technology also outside of work was introduced after the Second World War, often called *human factors*. Since the spreading of computers first at work, later as part of daily life, human factors has come to also include the study of how humans interact with computers in the field of Human–Computer Interaction (HCI) (Sears and Jacko, 2008).

HCI's original goal was to bring cognitive science into software development (Carroll, 2003), comparable to how ergonomics introduced scientific knowledge on humans into work environment. HCI is a very broad field and incorporates research from a range disciplines of computer, management and social sciences. HCI often requires (over-)simplifications to be of use for practitioners. Examples are the cognitive modeling as keystroke counting, checklist-based approaches, and guidelines of which many are grounded more in common sense than in research (Carroll, 2003). The practitioner's perspective of applied HCI is often called *usability engineering*.

To approach usability in practice, its characteristics are often broken down into multiple concepts that are partly overlapping. In usability engineering, these concepts are also used as usability quality attributes to assess the usability of software systems and, in part, as high-level principles of how to achieve usability. Practitioners and standardization bodies have arrived at different, but similar ways of differentiating aspects of usability as shown in Table 2.1. The usability engineering practitioners' definitions of Nielsen (1997) are both meant to work as quality attributes and principles. The list of

Nielsen (1997)	Dix et al. (2004)	Shneiderman and Plaisant (2005)	ISO/IEC 9126-1 (2001)	ISO 9241-11:1998 (1998)	ISO 9241-110:2006 (2006)
			Operability	Effectiveness	Suitability for the task, controllability
Learnability, memorability	Learnability (predictability, familiarity, generalizability, consistency, . . .)	Time to learn, retention over time	Understandability, learnability		Suitability for learning, self descriptiveness, conformity with user expectations
Efficiency	Flexibility (substitutivity, customizability, . . .)	Speed of performance		Efficiency	Suitability for individualization
Errors	Robustness (observability, recoverability, . . .)	Rate of errors			Error tolerance
Satisfaction		Subjective satisfaction	Attractiveness	Satisfaction	

Table 2.1.: Usability concepts, principles, and attributes

Shneiderman and Plaisant (2005) can be mapped to Nielsen's, but Shneiderman stresses that their attributes are more directly measurable. In contrast, the list of Dix et al. (2004) – "Principles to Support Usability" – aims to provide design support.

For the standards body perspective, ISO has two distinct approaches on usability. ISO/IEC 9126-1 (2001) defines a general software product quality model and measurable quality attributes that include usability as one of five major quality factors. The usability quality attribute only covers part of the breadth of usability, with, for example, efficiency and robustness being classified under other top-level quality attributes. Since usability even more than other quality aspects depends on the context of use, usability attributes cannot be considered inherent product attributes only. To account for the user perspective of product quality, ISO 9126 separately defines the "quality of use". The ISO 9241 series takes a different approach, aiming to provide guidance on achieving usability, including design processes and measurement. Part 11 defines usability concepts on a very abstract level (ISO 9241-11:1998, 1998). The principles that allow practitioners to achieve the ISO 9241-11 concepts are then described in the more concrete context of dialog design in part 110 (ISO 9241-110:2006, 2006).

Since the aim of this thesis is to improve the state of usability in broader areas than dialog interfaces, we will apply the ISO 9241-11 approach of generic top-level goals to prevent a overly narrow focus, employing the following attribute definitions, which are closely related to those in the standard. We consider the artifact broadly and include not only the technical mechanisms, but also the organizational measure, such as related procedures:

Effectiveness The ability to complete a task entirely and accurately with an artifact,

Efficiency The relation of effort necessary to achieve the effective usage of the artifact, that is, completing the task entirely and accurately,

Satisfaction The joy of the user experience to interact with the artifact.

Of the several underlying principles that can be applied to achieve those usability goals, a number of principles are particularly important with respect to security usability:

Comprehensibility Individuals' understanding of an artifact and the underlying model to take adequate decisions with respect to possible consequences when interaction is required (cf. "understandability" in ISO/IEC 9126-1, 2001).

Learnability The ability of individuals to acquire the expertise to efficiently interact with an artifact, explicitly through training or implicitly through active use (cf. ISO/IEC 9126-1, 2001).

Flexibility A decreased rigidness of an artifact to allow for higher overall efficiency, for example, by changing the normal behavior in exceptional situations (cf. Dix et al., 2004).

Robustness The tolerance of an artifact to erroneous actions or decisions. Making mistakes is an inherent aspect of human behavior, particularly under stress and pressure. An artifact should enable ways to remedy errors retrospectively (cf. Dix et al., 2004).

Maintainability The ability to efficiently operate an artifact, including, for example, adapting it to new requirements and changes in the environment (cf. ISO/IEC 9126-1, 2001).

2.1.1. Research disciplines in HCI

One approach of this thesis is to foster the integration of research disciplines to derive more holistic solutions to authorization problems. Thus, it is useful to explore the numerous research fields grouped under HCI. The three major research areas of HCI are computer sciences, human factors and ergonomics, and management sciences. On the technical side of HCI, researchers from the computer sciences develop the models, mechanisms, and algorithms that both require and improve the interaction. Computer scientists, for instance, strive to reduce the complexity of the interaction, and provide for flexible and robust interaction, employing diverse disciplines, such as artificial intelligence and information visualization.

On the human factors' side, fundamental research draws from the understanding of human behavior and mental processes (Salvendy, 2006; Sears and Jacko, 2008). Accordingly, psychology and sociology influence HCI most and provide a wealth of research methods. Psychology itself has many facets. For HCI, particularly the cognitive psychology with the modeling of mental processes is of high importance. Cognitive analysis, for example, provides insights into how information displayed on a screen will be noticed and influences the decision-making process. Over time, various mental models have been proposed for HCI, based on theories, analogies, and representations (Payne, 2003). According to the model of Wickens and Carswell (2006), information processing can be studied to begin with on the level of the selective or focused attention and visual search. On a second level, the action selection is performed, primarily affected by the choice that is given as well as the modality of input and the feedback afterwards. Problem solving models include intuitive judgment (Kahneman et al., 1982), the rule-based as well as complex problem solving. In "Human Error", Reason (1990) studied the different ways in which humans commit errors in the decision-making process. There are approaches for the quantification of the likelihood of human error, the errors' relation to the context, and models for error prediction (Sharit, 2006). Cognitive psychology is also the basis for predictive theories, such as the GOMS model, that provide a simplification of cognitive processes to estimate user performance (Card et al., 1983).

In contrast to the focus on the individual in psychology, sociology studies the interaction among individuals. The Social Learning Theory of Bandura (1986) describes the multitude of effects from interactions and observation, and how norms and values are built. Latour (1991) refers to technology as "society made durable", considering technology a building block of society. Accordingly, technology is seen as a means of exercising power and influences behavior alongside direct interaction, social norms and values, and formal rules (Clegg, 1989).

Anthropology takes an even broader approach than sociology in that it focuses on cultural and ethnic aspects (Räsänen and Nyce, 2006). Important research methods, such as ethnography, originate from anthropology, enabling researchers to study socio-technical effects and the social context of use in the actual environments. When employing ethnography for HCI, the "situatedness" of action can be analyzed, for example, to improve the understanding and representation of the user requirements (Anderson, 1997).

Since a large proportion of the use of information systems takes place in organizational contexts, management sciences also form an important aspect of HCI. In these contexts, processes may be established that support the usage of information systems (Zhang et al., 2006), for example, through training and awareness programs. Zhang and Li (2004) describe a framework of HCI issues in organizations.

2.1.2. Guidance and methods for usability engineering

The aforementioned research fields offer a wide range of empirical methods for HCI research. However, software engineers who need to craft usable systems resort to more pragmatic guidance and methods. *Guidelines* describe approaches to solve specific challenges and are ideally backed by empirical evidence. Shneiderman and Plaisant (2005) give an overview of well-known guidelines, for example, for navigation and data entry. Even more comprehensive is the overview published by the U.S. Department of Health and Human Services (HHS), which includes the supporting evidence. Tufte has also published guidelines on information visualization (Tufte, 2000, 2006, 2007). Critics complain that guidelines are too specific and therefore incomplete and difficult to apply to usability problems. If only based on common sense and not backed by profound evidence, guidelines may be biased and sometimes wrong. While often meant to apply broadly, guidelines are also provided for operating systems to achieve a consistent user experience throughout the platform (Apple Inc., 2009). On a more focused level, companies also produce guideline documents for their internal or external development as part of their brand development.

Principles offer more general advice on how to solve usability problems. Usability principles are also often used to define the problem domain, for example, by Nielsen (1997), Shneiderman and Plaisant (2005), or Dix et al. (2004). Hansen (1971) already described usability principles, stating that developers should know the user, minimize the need for memorization, optimize operations and engineer for errors. Shneiderman and Plaisant (2005) have a more detailed list, partly overlapping with Hansen's, encompassing for example the need of consistency, informative feedback, the reversal of actions, and keeping the users in control of the interface. The ISO 9241-110 standard defines similar principles explicitly for dialog interactions (ISO 9241-110:2006, 2006). Thimbleby (2010) takes a different approach with his principles for interaction programming, emphasizing, for example, similarity to previous versions, simple designs, and that devices should behave like and change the real world.

While the guidelines and principles are very much motivated from pragmatic needs, *theories* on usable systems are closer to the HCI foundations. Descriptive theories support the collaboration and training of designers, explanatory theories allow them to reason about why certain designs work better than others. Prescriptive theories suggest approaches and predictive theories allow one to estimate performances beforehand (Shneiderman and Plaisant, 2005). An example of a predictive theory is the GOMS (goals, operators, methods, and selection rules) theory that predicts the performance of users in a specific design by counting mental and physical steps in solving a task (Card et al., 1983).

Apart from guidance, *methods* have been developed that guide not the resulting design, but the design process. For example, participatory design focuses on the user involvement in software development. Ethnography can more thoroughly inform the design process on problems by identifying unarticulated problems (Sommerville et al., 1993; Viller and Sommerville, 1999). One variant is Contextual Design (Beyer and Holtzblatt, 1998), which is primarily based on contextual inquiry: interviews at workplaces and observing user behavior. Its principles include the focus on the context as a concrete and ongoing experience, and the interpretation and discussion of observances. While participatory methods deliver more accurate information on user needs, they can lead to conflicting requirements and, thus, suboptimal compromises (Ives and Olson, 1984). To solve these problems, the Persona approach has been developed and is widely employed in practice (Cooper, 1999; Cooper et al., 2007). Personas define a limited number of fictional people precisely to focus the development and support feature debates. Cooper et al. (2007) describe a method to develop a Persona that includes identifying behavioral variables (activities, attitudes, aptitudes, motivations, skills) and behavioral patterns based on rich empirical source-material such as interviews.

While Personas support the elicitation of user requirements, task analysis can be applied to accurately model the context in that a product will be used. Task analysis answers the questions, who carries out tasks (individuals, groups, technical artifacts or agents), which tasks occur (cognitive or observable actions) and why the tasks need to be fulfilled (purpose of the tasks, products, satisfaction) (Hollnagel, 2006). As an alternative to task-directed design, goal-directed design has been suggested by Cooper (1999). Goal-directed design starts off from Personas and their goals instead of analyzing the complex real-world setting that may lead astray from implementing the most essential features.

2.1.3. HCI research methods

The guidance for usability engineering described in the previous section has only a limited robustness with respect to varying contexts of use. As a result, the usability of designs is often explicitly evaluated as part of the development process. Similarly, to define guidance and validate research, HCI relies on empirical studies. The goals of usability evaluations and research methods are similar, but differ in their methods and in the concreteness of the approach that is evaluated. Usability evaluation has a more pragmatic scope, focused on identifying major problems in specific designs. Conversely, HCI research must provide reproducible validations and thus employs more formal evaluation methods (Cairns and Cox, 2008).

Practical usability evaluations are conducted at different points in the development lifecycle, often iteratively accompanying development. A simple form of usability evaluation is *usability inspection* (Holzinger, 2005): domain or user interface experts review a design, a prototype, or a running application for potential usability issues. Specific methods are heuristic evaluations, based on usability heuristics (Shneiderman and Plaisant, 2005), and cognitive walk-through, focusing on the cognitive load of users. Results from usability inspections are rather qualitative, generally difficult to compare, and depend on the expertise of the participating experts.

More direct *usability tests* with users attempt to quantify the combined performance of the system and user (Nielsen, 1997). Measurable factors include the success rate, the total time spent, the error frequency, and the number of help incidents. Apart from quantitative evaluations, which need to be brought into perspective against reference products, evaluations can focus on discovering problems (Lewis, 2006). Problem discovery tests typically employ thinking-aloud (Holzinger, 2005), resulting in qualitative information on the reasons behind problems.

In HCI research, more formal methods are required. *Controlled experiments*, largely drawn from psychology, help to evaluate interfaces and styles of interaction to understand the cognitive activities (Cairns and Cox, 2008). Important design decisions for controlled experiments include whether they are conducted “within” or “between” subjects and whether they take place in real environments (“in vivo”) or in constructed ones (“in vitro”). For more realistic results than from laboratory experiments, field observations are also employed (Holzinger, 2005). *Surveys* are less resource-intensive and allow evaluators to capture the opinions from larger numbers of participants. Surveys may be conducted through questionnaires, implicitly through user action logging, or in interviews and focus-group discussions (Cairns and Cox, 2008).

Similar to usability guidance, many evaluation approaches are not backed by sound empirical evidence and can result in weak research quality (Lewis, 2006). Often, the evaluations degenerate into existence proofs with researchers identifying one way in which the new approach is superior to the established ones. Instead, research benefits to a larger extent if general insights, such as recommendations are derived from rich empirical studies. For practitioners, Greenberg and Buxton (2008) note that usability evaluations may be harmful by causing premature decisions from early evaluations. They also argue that usability evaluations often do not adequately take the usefulness, the user experience, and the breadth of the later context-of-use into account. Particularly, interrelations between culture and the technology is difficult to estimate in usability evaluations (Greenberg and Buxton, 2008), since the interactions are complex and only occur once the artifact is deployed.

2.2. Security usability problems

While Saltzer and Schroeder (1975) formulated the classical requirements for security usability in the 1970s, only the general prevalence of threats and the need of security for lay people since the 1990s led to increasing research on usability in security. An obvious question at this point is whether there are differences in problems and methods from general software usability. The hypothesis is that the security-usability problems are more critical. General usability issues primarily degrade the performance of the socio-technical systems and impact the satisfaction of the users. In contrast, a lack of security usability can render the security architecture ineffective (Church et al., 2008). The "barn door property" exemplifies this effect: Once the horse has left the barn, closing the door cannot undo the damage. Applying the metaphor to security usability, once critical data is disclosed or lost, there often is no (simple) way of reversal (Whitten, 2004).

If people are overly burdened by security mechanisms, security-critical mistakes occur (Cranor, 2008; Sharit, 2006; Zurko, 2005; Reason, 1990). A similar effect is caused by a habituation of acknowledging alerts or warnings. One example can be found in the Windows Universal Access Control (UAC). Similar to the Unix sudo command, the Windows UAC allows applications to temporarily gain elevated privileges. However, UAC password prompts appear frequently, unnerving users and, thus, causing them to habitually allow access (DeVaan, 2009). A more deliberate form in which security usability affects security is the circumvention of security measures, for example, when users share passwords (Church et al., 2008). Circumvention is often caused by an unacceptable effort for the effective use of the measure in relation to the perceived risk from bypassing the measure.

Jøsang et al. (2007) enumerate three ways in which users may be vulnerable from security usability:

- The user may be *unable to understand* what actions are required because the user is unable to draw the necessary conclusions,
- The user may have *insufficient knowledge* and is thus unable to choose the *correct* action,
- The user may be *overburdened mentally or physically* to derive the necessary conclusion or taking the action, including high numbers of repetitions.

Below, we discuss a number of reasons for security-usability failures. First, there are several psychological characteristics of individuals that influence the perception of security and risk, and thus the effectiveness of security measures and its usability. Second, sociological effects of the interactions between individuals similarly affect the behavior of individuals concerning security. Third, as a result of the inherent complexity and abstractness of many security mechanisms, the cognitive burden for users from security measures can be higher than for general software usability. Additional challenges stem from the broad range of contexts of use. Security technology is often reused in different contexts than originally envisioned (Zurko, 2005), resulting in further usability problems. Moreover, the security requirements vary widely between users and even for individual users (Xia and Brustoloni, 2005).

2.2.1. Psychological factors: Rationality in decision-making

Already Saltzer's early security usability requirements draw from human characteristics by formulating the psychological acceptability of security measures (Saltzer and Schroeder, 1975; Bishop, 2005). More generally, cognitive psychology can explain the decisions regarding security (West, 2008). Since Kahneman and Tversky's experiments in the 1970s, the "irrationality" of human behavior is accepted (Kahneman et al., 1982; Gilovich et al., 2002). They showed that humans are not acting rationally in the sense of a *homo economicus* who consciously weighs the outcomes of alternative options (Simon, 1979). Instead, people are heavily biased in their intuitive judgment, for example, by valuing losses higher than gains (Prosperity Theory) or inadvertently factoring in prior knowledge, even if unrelated. The main reason is that intuitive decision-making is highly optimized through heuristics, such as, from prior knowledge.

Based on these psychological effects, we can describe the behavior of individuals that at times seems irrational. West (2008), for example, discusses how safety is more abstract as a reward than the primary goal, often a functional task, particularly without proper awareness of the associated risks. In security usability literature, this effect has been termed "secondary goal problem" (Whitten, 2004; Sasse et al., 2001) – an effect well-known in ergonomics (cf. Wickens et al., 1983). For example, when sending an email, the users' goal is to communicate, not to encrypt and sign the email. The integrity and confidentiality of the email is only seen as the secondary goal after the one to communicate at all. DeWitt and Kuljis (2006) show in a case study that users compromise security to complete their work. In many environments, only a limited amount of additional effort for security is acceptable. Beautement et al. (2008) proposes the Compliance Budget to estimate when individuals will accept security mechanisms and at what point the "budget" of acceptable security effort is depleted and individuals will stop complying. In a similar line of thought, Herley (2010) argues that individuals take a very economic approach when they circumvent security mechanisms and ignore security advice. However, the users' framing of the decisions is bounded (Simon, 1979): The users ignore or are unaware of the externalities from their actions, resulting in decisions that seem "irrational" to experts.

The psychological challenges can also be considered from an explicit attack perspective: Nohlberg (2009) describes various forms of deception that humans are prone to, ranging from a natural obedience to authority and reactions to scarcity over exploiting commitments and striving for consistency to simple social proof.

From the software development perspective, the secondary-goal factor may also affect the requirements elicitation for security (Tondel et al., 2008). On a high level, abstract non-functional security requirements compete for development resources with functional requirements that provide more immediate and more directly measurable added value. In addition, security requirements may also conflict with other non-functional requirements such as performance.

2.2.2. The social influence on individuals' behavior

Apart from the psychological effects, the behavior and decision-making of individuals is influenced by peers, society and culture (Bandura, 1986; Dontamsetti and Narayanan, 2009). Effects may stem from social norms and values; for example, the locking of desktop screens for breaks is unconsciously linked to a lack of trust towards colleagues. The reason is that the social context changes from earlier, for example, non-technical contexts, but the social norms are not following suit even though they might not be adequate for the changed situation (Church et al., 2008). Bandura (1986) describes with the Social Learning Theory how, for example, observational learning, self-regulation and reflection, and the flow of information in social networks influences human behavior. The Circuits of Power (Clegg, 1989) formulate three interacting levels on which power is enforced in society and through technology, and thus influences behavior: The "episodic" circuit refers to actual experiences, such as direct commands from superiors, "social integration" encompasses the social norms and values, and "system integration" the technology and formalized rules. Latour (1991) considers "technology. . . society made durable", thus mechanisms and rules as an extension of societal norms and values, and applies Actor-Network Theory to a hotel-key example to show how technology influences behavior. Inglesant and Sasse (2011) employ this approach to discuss the compliance of employees with security policies.

Risk perception is similarly influenced by societal structures. Adams (1995) argues that it is difficult to force people to specific levels of risk mitigation, since they are required to take risks in everyday life, for example in traffic, and unconsciously accept residual risks. According to Adams, risk is socially and culturally constructed: people adjust to risks similar to a "risk thermostat" – with improved security measures, people become more daring and might even neglect common sense.

2.2.3. Complexity and the incomprehensibility of the abstract

Security technology has developed through many iterations of increasing complexity to fulfill security and other non-functional requirements – despite Saltzer's demand of as simple security designs as possible in the 1970s (Saltzer and Schroeder, 1975). This effect can be observed in authorization models, arriving at role-based models for maintainability at the cost of an abstract and less comprehensible model than, for example, access control lists. Similarly, simple pre-shared key schemes for transport encryption developed into more maintainable, but more complex public-key-based systems. As a result, many users cannot comprehend today's security configuration, as shown in a large survey among users of above-average "IT literacy" (Furnell et al., 2006). Blackwell et al. (2008) generally argue that forcing computational thinking on humans can reduce the overall quality of the result, since important non-computational confutations are lost. Similarly, the abstract of security mechanisms encumbers comprehensibility.

In order to cope with this complexity the approach traditionally has been to hide the complexity from the users, creating a gap between using security mechanisms and understanding them (Zurko, 2005). In normal operation, users usually are not required to understand the background of the employed security technology. However, in exceptional situations, for example, in case of attacks, security mechanisms require user intervention. Not understanding the background may then cause the user to take wrong and potentially dangerous decisions. A prominent example is Web browsing over TLS-secured connections. In normal operation, the complexity of the public-key infrastructure is hidden from the user. In case of defective server configurations or attacks, warning dialogs then require users to make difficult decisions. This example also demonstrates the difficulty of providing useful feedback related to security technology, an important usability principle (Whitten, 2004).

Achieving usable security is further complicated by the fact that security usability acts on multiple

	Field study	Laboratory	Analytical	∑
Authoring	10	10	9	29
Making	7	1	1	9
Functional	2	0	2	4
Developing	1	3	0	4
	12	10	11	33

Table 2.2.: Perspectives covered by literature on authorization problems

layers. The aforementioned factors focused on the functional user of interacting with and configuring security mechanism for functional tasks. Centralized security administration in organizations may partly relieve functional users from the configuration burden. Here, the burden shifts to security administrators to configure the mechanisms, depending on usable configuration models and supporting tools. A study on how administrators use security tools identified the lack of knowledge, insufficient awareness, and poor strategies as reasons for vulnerabilities (Furnell and Bolakis, 2004). Administration usability problems also occur for firewall configuration, for example, due to the low-level configuration languages that require translations from high-level policies and encompass large number of rules with hidden interactions (Geng et al., 2005).

On a third layer, the security mechanisms need to be implemented as part of the software development. Developers often rely on security APIs and need to be careful to prevent implementation mistakes that might render the security mechanism ineffective. Compared to the research on problems of functional users and administrators, there has been little work on security usability in software development.

2.3. Problems surrounding authorization

The focus of this thesis is authorization. Accordingly, we narrow our focus from the general problems previously identified in security usability to those in the area of authorization. Specifically, we review 31 publications that describe problems with authorization (cf. Table D.1 in Appendix D). Going back to the rough sketch of perspectives in authorization in Section 1.1, we can structure prior literature according to the perspectives covered. Additionally, we differentiate here for the security management between those taking the decisions (*policy making*) and those modifying the policy (*policy authoring*). While numerous publications cover several perspectives as being affected by problems, policy authoring is most commonly addressed as shown in Table 2.2. In contrast, only a small proportion of publications touch the problems of functional staff and developers and few problems are explicitly stated for these perspectives. The argumentation of the publications is based on field studies (subjective or objective data from practice), laboratory experiments (data collected in controlled environments), and analytical considerations. The context of most of the publications is organizational, with selective publications on authorization in Grid, for firewalls, social networking, and the authorization of applications.

2.3.1. Policy authoring

To a large extent, studies in the area of authorization usability focus on the usability of policy authoring interfaces. Zurko and Simon (1996) formulated the following requirements for a usable policy authoring interface from their early study on usable security: (1) integrated management of different policy aspects, such as rules and groups, (2) convenient interface even for complex rules, (3) consistency checks, (4) high-level overview of the consequences of rules, and (5) a display of the rules that apply to a specific object. Whalen et al. (2006) report in their study on authorization in organizations that users still find it difficult to manage policies. Typical problems are the high effort that needs to be invested (Bertino et al., 2008; Gallaher et al., 2002), the complexity of rules from the high quantity (Al-Shaer and Hamed, 2003) and from the requirement complexity (Smetters and Good, 2009). Problems in authoring of policies can lead to misconfiguration as the result of configuration errors (Smetters and Good, 2009; Wool, 2004).

One cause of problems with the authoring of policies can be found in the characteristics of the authorization models that determine what types of rules can be formulated. Bauer et al. (2009) found in a field study on organizational policy-authoring that the model expressiveness is an important factor. Coarse controls may, for example, lead to the resource owners circumventing the controls to achieve their goals, as Ahern et al. (2007) showed for social networking authorization. Flexibility (Bauer et al., 2008a) and the handling of exceptions (Sikkel and Stiemerling, 1998; Bauer et al., 2009) can be necessary to efficiently model restrictions. Moreover, Brostoff et al. (2005) show that the comprehensibility of the model is important in policy authoring – for example, to allow authors to understand the role concepts, default rules, and different kinds of assignments (cf. also Inglesant et al., 2008).

Even more broadly discussed in literature is the comprehensibility of the policy itself (Brostoff et al., 2005). Policy authors also need to understand the consequences of their changes (Rode et al., 2006). In some cases, there is a lack of transparency of the currently enforced policy, caused, for instance, by a misleading presentation of the policy (Yee, 2005) or a missing visualization of the effective permissions that are actually enforced (Maxion and Reeder, 2005). The transparency aspect is particularly important for interrelated rules, policy conflicts, and their resolution (Reeder et al., 2008, 2011). When the policy is authored as a textual representation, the rule syntax may cause problems (Reeder et al., 2007), as can inconsistent policy files (Herzog and Shahmehri, 2006). If tool support is provided, the tool's usability can affect policy authors in numerous ways – for example, when the tool uses inconsistent terms (Reeder et al., 2007), or when it lacks the information needed to formulate the policy (Maxion and Reeder, 2005).

2.3.2. Problems for the functional, policy-making, and development perspectives

Less pronounced in literature than the usability problems for policy authors are those for functional users, policy makers, and developers. For functional users, who need to interact with information systems to complete their work tasks, Whalen et al. (2006) found that authorization often interferes with their work. Johnson et al. (2009) argues that strict restrictions can have adverse effects and force users to circumvent the authorization measures, for example, by sharing their password with people who need access to additional resources.

Policy makers take the decisions on changes to the policy and need to prevent overly restrictive or permissive permissions. The field study of Bauer et al. (2009) showed that it can be difficult for multiple policy makers and authors to interact. They also found that problems with the state of the documentation can impact the decision-making. Sinclair et al. (2008) reported that the complexity of the risk assessment can be high, due to the dynamics in organizations, the number of roles, and the heterogeneous applications involved.

The literature is very thin on what problems developers have with the integration of authorization in systems. Zurko and Simon (1996) recognized that it can be difficult for developers to implement the authorization controls in applications. Herzog and Shahmehri (2006) found that not only policy authors are affected by problematic authorization frameworks, but also the developers who need to integrate the enforcement. In a study of enforcement integration in the Linux kernel, Jaeger et al. (2004) showed that in multiple instances, the enforcement was not correctly placed.

Study	Study design	Environment	Focus/scope	Findings	Recommendations
Sikkel and Stiemerling (1998)	Subjective: interviews	Private and public organizations	How users specify policies	Policies are stated as grants/denials, refined by exceptions, e.g. scopes	Provide expressive model: object grouping, scope, delegation
Whalen et al. (2006)	Subjective: survey, interviews	Medium-sized research laboratory	Individuals' problems from mechanisms	Users manage policies, but struggle with it; authorization interferes with primary tasks	Integrate with workflows, support social controls, visualize policies, and simplify management
Bauer et al. (2009)	Subjective: interviews	Diverse organizations	Challenges for policy professionals	Problems from stakeholder interactions and inadequate models	Support communication of authors, improve authorization models
Smetters and Good (2009)	Objective: historical policy data	Medium-sized corporation	Usage of authorization and models	Complex, rarely changed policies, management errors	Simpler models, patterns, better management tools

Table 2.3.: Prior studies on authorization in organizations

2.3.3. Challenges in organizations

Since the primary focus of this dissertation is organizational authorization, it is useful to consider how existing literature covers this area. A small number of studies have analyzed how challenges in authorization actually materialize in organizations, summarized in Table 2.3. Three of the four identified studies exclusively focus on the challenges in policy authoring. Sikkel and Stiemerling (1998), Bauer et al. (2009), and Smetters and Good (2009) examine how the expressiveness of authorization models affect policy management and suggest model improvements. In addition, Bauer also analyzes the interactions of policy authors with different roles. In contrast, Whalen not only explores the problems of managing authorization, but also how authorization interferes with the primary tasks of functional users.

These studies still primarily focus on authorization usability issues on an individual level, primarily with respect to policy authoring. The consequences of the identified challenges remain implicit. Since most problems rarely become visible for management and then only anecdotally as individual cases, the authorization issues are still largely ignored in organizations. One reason is that the scale and the actual impacts on organizational goals are unknown. To make a case for fundamentally addressing the problems that functional users are faced with every day, the full breadth of authorization challenges needs to be explored with their complex interrelations and, particularly, their impact on organizational goals. This will be pursued in this thesis.

2.4. Addressing security usability problems

While early endeavors in security usability focused on what goes wrong, for example, in the case of "Why Johnny can't encrypt" (Whitten and Tygar, 1999) and the analysis of problems with passwords (Adams et al., 1997), the research gradually shifted to also include approaches to solve the problems. Researchers created numerous guideline documents and proposed principles on how to develop usable security mechanisms (Nurse et al., 2011): Garfinkel (2005) developed guidelines on interface design for security mechanisms; Yee (2005) focused on authorization and suggests principles such as finding "the most comfortable way to do tasks with the least granting of authority". Whitten (2004), on the other hand, proposes methodologies and principles, including building suitable security metaphors, incrementally increasing complexity, and the principle of learning "well in advance".

Many security usability guidelines and methodologies have a narrow scope and are based on weak empirical evidence. Little empirically-backed prescriptive theories have been developed for security usability that allow developers to derive concrete approaches. In the following, we group existing approaches under three themes: Reducing the burden of users interacting with security mechanisms, making the abstract mechanisms and models understandable, and integrating the perspectives of functional users, security management, and development.

2.4.1. Reducing the burden

One approach to improve the security usability is to reduce the burden that is placed on the user. The rationale is that a high cognitive effort imposed by a security mechanism will lead to errors and deter users from employing the mechanism. There are various ways to reduce the burden: First, the balance between the security requirements and the usability impacts can be shifted towards usability by applying effective risk management. Advocates of "good-enough security," including Tognazzini (2005) and Sandhu (2003), argue for an appropriate balance between the business needs and security measures. Garfinkel (2005) similarly formulates a security design principle of "good security now" to prevent users from finding their own, possibly even less secure, approach.

In another approach to reduce the users' burden, Zurko (2005) demands that security-related decisions should be minimized. This can be achieved, for example, by incorporating existing user behavior in security models and thus prevent extra burden in addition to the usual tasks (Zurko, 2005). One example of this approach is implicit granting of authorization to files. Following an email analogy, Johnson et al. (2009) propose a mechanism in which access is granted by sending a document to a user. The general approach in reducing explicit security decisions is to make security technology transparent for the user and thus hide its complexity to reduce the cognitive burden. This approach is in line with the oft-demanded security that "just works". Accordingly, Smetters and Grinter (2002) see implicit security as one of the building blocks in useful and usable systems. In their design proposal, security-related decisions are derived from user actions related to the primary goal. In the domain of application authorization, Yee (2005) differentiates *security by admonition* from *security by designation*. Security by admonition is the conventional approach, in which static rules constrain an application and require the prior configuration. In security by designation, the user has to acknowledge each widening of the constraints as part of the work tasks, possibly implicitly as in the aforementioned email analogy.

The latter approach aims to reduce the configuration effort. The user will often need to have a deep understanding of the complex security model to make educated decisions when configuring security mechanisms. An increasingly common approach is to provide decision support, often based on risk assessments to allow the user to adequately take the various factors into account (Beresnevichiene et al., 2010; Parkin et al., 2010). Circumventing the need of explicit configuration, adaptive systems respond to user actions and configure themselves accordingly (Ackerman and Mainwaring, 2005). One of Garfinkel's design principles is to develop standardized security policies that may be reused in different application contexts and lower the threshold for understanding the configuration of new products (Garfinkel, 2005).

The opposite approach to reducing the end-user configuration burden is the adaptability and tailorability of security systems. These systems allow users modify an application's behavior at the surface or at deeper levels (Ackerman and Mainwaring, 2005), taking inspiration from *end-user development* (EUD) (Lieberman, 2006) or "meta-design" (Fischer, 1999). The opposing principles of reducing the configuration burden and the adaptability or tailorability expose the trade-off between the learning effort and the achievable efficiency at expert level. While the novice performance will suffer from the up-front learning effort for understanding the configuration mechanisms, experts will profit in the long run from very high productivity. In the end, security measures need to fit the context

(Sasse, 2011), including the experience of the involved individuals.

One approach to handle the challenge of diverse types of users is to cluster users according to their specific goals and knowledge level to offer appropriate interfaces or adequately interpret the users' actions (Ackerman and Mainwaring, 2005). Following this path, Whitten's design pattern of "safe staging" offers adequate abstraction levels of the user interface for different user groups (Whitten, 2004). Whitten argues that this approach helps to bridge the gap between novices and experts by incrementally increasing complexity with increasing expertise.

2.4.2. Making the abstract understandable

Understanding is important on two levels for security usability: to adequately interact with the technical mechanism and for the security risks. First, understanding the security mechanism improves the ability of users to make adequate decisions if interaction is required. Second, motivation guides behavior and is thus an important factor for security to be effective. Understanding and being aware of the risks from avoiding the necessary effort will often increase the motivation to behave more securely.

Whitten (2004) argues in favor of creating understanding of the technology instead of hiding security mechanisms as proposed in the previous section. Invisible security can be dangerous since the users are missing the crucial information of whether they are protected and adequate security is in place (Zurko, 2005). The choice between explaining and hiding the complexity of security technology depends on the context. Relevant aspects are, for instance, the user expertise level, the motivation to learn and the likeliness of exceptional situations that would require user interaction. Whitten (2004) developed narrowly scoped rules on the application behavior that is required for the usability of invisible security: If security mechanisms fail, either the user should be disallowed to continue or the lack of protection must be made visible to the user. Accordingly, Hertzum et al. (2007) identify three alternatives for online-banking security: the instruction of users, the automation of tasks, and the understanding of the security technology. In instruction and automation, users complete tasks without understanding how the security mechanisms work.

There are numerous ways of how to explain security to users, including online documentation, context-sensitive help, and wizards and assistants (Herzog and Shahmehri, 2007). Explanations primarily occur when security technology fails or warnings are displayed because of suspected attacks. To prevent the conflicts between the primary tasks and the effort necessary to understand the problems, Whitten (2004) proposes the principle to learn "well in advance". A related approach is "mastery learning" through automatic training, similar to tutors, in order to expand the users' mental model on security tasks (Ackerman and Mainwaring, 2005).

According to Zurko (2005), people also understand physical security, at least on an abstract level, and this should be reflected in information security. For example, without the knowledge of the technical details of door locks, people can decide to lock the door or keep it unlocked. However, it is difficult to directly apply behavior or knowledge from the physical world as metaphors for virtual mechanisms (Wash, 2010). Complex, abstract, intangible risks and mechanisms are more difficult to understand for humans than concrete and real mechanisms (West, 2008; Blackwell et al., 2008). Thus, we need to aim for concreteness to make security mechanisms and risks comprehensible. Wash (2010) explores the mental models that people have of security to allow designers to continue from existing knowledge. Whitten (2004) proposes a development methodology to derive appropriate visual metaphors for security technology. Hardee et al. (2006) recommend in a study of computer risk perception to more explicitly formulate monetary and property losses in warnings. Dontamsetti and Narayanan (2009) suggest a conscious competence model that relates users' awareness to their knowledge levels to design appropriate training and awareness campaigns. Roper et al. (2005)

distinguish between the approaches of training, education, awareness, and motivation to improve people's behavior with respect to security. Stanton et al. (2005) propose a taxonomy of user security behaviors along the dimensions of expertise and intention and conclude from their model that improving awareness is often the least expensive way of improving security. Beautement et al. (2008) recommend adjusting the organizational culture for a higher focus on security risks. This can also lessen the management pressure for productivity that implicitly reduces the value of secure behavior and counteract the individuals' drive towards trading security for higher work efficiency (Zipf, 1949; Rasmussen, 1997).

2.4.3. Integrating perspectives for participative management and development

The approaches discussed in the previous sections demonstrate that security usability requires both, adequate management of defining the needs and configurations, and adequate development on the technical side. In case of management, the general goals are to improve expertise, awareness and motivation among users as well as establish rules for various stakeholders. Depending on the specific context, management may take a variety of shapes. For example, in the context of society, knowledge and awareness can be primarily influenced through broad awareness campaigns or in education. Rules are typically established in legislation, such as in privacy regulations. Explicit security management is more common in organizational contexts where there is more direct control over the individuals.

To arrive at adequate decisions based on the security and functional needs, security management requires intensive communication between the affected stakeholders (Jaferian et al., 2008). Only close interaction can lead to the necessary balance between conflicting objectives, such as between productivity and security (Besnard and Arief, 2004). Albrechtsen (2007) suggests a user-involving approach to security management, since it can be useful to exploit the domain knowledge of functional users (Church, 2008). Flechais and Sasse (2009) show in a model of the stakeholder interaction how functional, management, and development perspectives contribute their knowledge and experience to arrive at appropriate security measures.

Accordingly, interaction not only needs to be fostered between functional user and security management, but also with the developers. Developers often need to find trade-offs for each individual security-usability problem and context. Karat (1989) described an iterative security-usability process already at the end of the 1980s. A further example of a development process with a focus on security usability is the *Appropriate and Effective Guidance for Information Security* (AEGIS) (Flechais et al., 2003, 2007). AEGIS integrates security usability aspects in the requirement, design, and implementation phase of software development processes and particularly emphasizes the integration of the different stakeholders.

2.4.4. Specific approaches to the problems in authorization

Authorization has been the subject of a large number of research publications that cover approaches in theory and practice. In this section, we examine selected relevant approaches. The selection is aimed to provide a representative coverage of the field even though only a small proportion of the large body of publications on authorization can be discussed. A list of the considered approaches, categorized by the basis of the respective argument is given in Table E.1 in Appendix E[1].

In the examined literature, the coverage of authorization problems is very heterogeneous, both in quantity and in quality. For a quantitative indication of the distribution of approaches, each is

[1]The table is also structured already according to and contains the authorization requirements to be introduced in Chapter 6 to prevent repetition.

Category	Description	Example
Concrete approach	A very concrete concept or implementation of an approach to address authorization problems	Tool prototype implementation
Method	A methodological approach to solve a problem	Policy lifecycle model
Principle	A proved approach claimed to apply for a broad range of problems and contexts	"Make authority transparent"
Recommendation	Identified ways of addressing problems, often identified from results of field studies	"Integrate authorization in workflows"
General approach	An approach from a related problem domain, not necessarily already applied to authorization in particular	Task analysis
Security best practice	Often formulated in best-practice documents in standardization	"Training and awareness campaigns"
Requirement	Reference to more concrete requirements that provide additional approaches to address the problems (cf. Section 6)	–

Table 2.4.: Categories of approaches

assigned to one of the categories listed in Table 2.4. Of the 128 identified approaches, 112 directly apply to authorization (leaving out *security best practices* and *general approaches*). 67 of these approaches are *concrete approaches* or *methods*. Of the concrete approaches and methods, 72% primarily address development. Even though these quantities are not based on a comprehensive literature review, the numbers indicate a pronounced focus on development aspects and, particularly, on the usability of policies and tools. This matches the bias in the prior research on the problems with authorization towards authoring as seen in Section 2.3. The hypothesis of this thesis is that to effectively solve the problems with authorization in organizations, we have to broaden this narrow focus, considering non-technical aspects and the interrelation between stakeholders with different perspectives on authorization (cf. Section 1.2).

2.5. Methods in research on security usability

Whereas the HCI research offers a broad range of methods to analyze and address usability problems (cf. Section 2.1.3), *security usability* research has often relied on a limited set of research approaches (cf. Table 2.5)[2]. Analytical research is common, in which researchers reason on observed problems to define, for example, design guidelines (e.g. Yee, 2005). Equally common are laboratory experiments, such as the well-known "Johnny" study (Whitten and Tygar, 1999). While based on more reliable evidence than analytical reasoning, laboratory studies on security usability are problematic regarding validity (Sotirakopoulos et al., 2011), and only offer a limited amount of insight if constructed as quantitative studies. The research questions in surveys are often broader and the number of participants in the study may be larger. In-depth case studies, often based on interviews with practitioners, can produce much richer results than surveys, potentially revealing underlying problems – for example, in the study on password use by Adams et al. (1997). If adequate data is available, objective empirical evidence may be even more reliable and can reduce biases of participants and researchers potentially present in subjective data (cf. Ericsson and Simon, 1993). However, only a small number of studies on security usability has been based on objective evidence.

One problem with the particular focus on analytical and laboratory research (usability evaluations)

[2]Since the overview of research and the applied methods in Table D.1 focuses on authorization, it shows a skewed picture regarding the methods.

Method/evidence	Example
Analytical	
Anecdotal evidence	Identifying general problems in security management (Blakley, 1996)
Analytical reasoning	Eliciting design guidelines (Yee, 2005)
Design activity	Designing a tool for policy management (Vaniea et al., 2008a)
Subjective empirical	
Laboratory experiments	Evaluation of a tool for email security (Whitten and Tygar, 1999)
Survey	Comprehensbility of end-user security (Furnell et al., 2006)
Field/case study	Interviews on password use (Adams et al., 1997)
Objective empirical	
Usage data	Evaluating usage data of social network use (Ahern et al., 2007)
Artifacts	Evaluating historical policy data (Smetters and Good, 2009)

Table 2.5.: Methods in research on security usability

is that the results are often limited to insights on the interaction of the individual with security measures. This psychological perspective misses the sociology component in security usability, particularly the interaction of stakeholders in organizations and in society (power relations, organizational communication). One of the themes of this dissertation thus is to broaden the research approach and particularly consider the interaction between different stakeholders for more comprehensive approaches to usable security.

2.6. Beyond the individual: The organizational and security economics

In this chapter, we reviewed numerous approaches to identify problems with the usability of security and to solve the problems. The approaches typically focused on the individual user – for example reducing the burden, addressing complexity, increasing awareness, and appropriate management and development. These individual approaches seem to be insufficient to solve the problems. The narrow perspective on the individual misses the broader picture of the context of use and fails to account for the behavior of the individuals in the societal setting. This particularly becomes apparent when security technology is designed with only the ability of users in mind, either for the expert or for the "dumb user" (Sasse, 2011). This approach neglects the actual needs and primary tasks of users, and that users often have good reasons to choose insecure behavior (Herley, 2010). A current stream of research in security usability attempts to broaden and deepen the perspective and takes an economic approach to model and analyze behavior – and particularly considers the context of use.

Economic security usability models focus on the analysis of individuals' behavior and study their cost/benefit trade-offs. For safety risks, Rasmussen (1997) explains the organizational dynamics from users (drive for least effort), management (productivity) and safety campaigns (safety). He observes that under these influences, user behavior tends to be unsafe. In information security, Besnard and Arief (2004) apply similar approaches, discussing cognitive efforts and benefits of individuals. Beautement et al. (2008) take a broader approach with their "Compliance Budget" and particularly focus on the perceived costs and benefits and how these explain (non-)compliance with security policies. Focusing on the perception, their model includes further external factors, apart from the cognitive efforts, that motivate compliance, such as the culture of the organization. In addition to the economic behavior, we need to take into account that individuals do not behave fully rational as a "Homo economicus" (Simon, 1979), but rather base their decisions on incomplete knowledge (Baddeley, 2011) and are affected by cognitive biases and heuristics (West, 2008).

2.6.1. Perceived effort of secure behavior

For individuals interacting with information systems to complete tasks, behaving in a secure way and complying with security policies often creates additional effort when compared to a system without security mechanisms. For example, when using encrypted USB sticks, users cannot directly access the files, but need to start a dedicated program instead (Beautement et al., 2008). Beautement et al. (2008) lists consequences of increased physical and cognitive load, embarrassment, missed business opportunities, and the general "hassle factor" as typical additional costs for the individual. While many of these efforts are actual, it is necessary to stress that depending on the context, the perception of the effort differs. For instance, having to request a password reset shortly before a deadline will be perceived as higher hindrance than in a relaxed situation. Beautement et al. (2008) further argues with the Compliance Budget that every effort required for compliance reduces the acceptability of later efforts.

There are two primary means available to adjust the perceived effort (Beautement et al., 2008). Directly, the design of the measure can be improved to reduce the cognitive load, optimize the procedures involved, or reduce the friction with the primary tasks of an affected individual. Indirectly, training and education can both reduce the actual effort by optimizing the usage of a mechanism and reduce the perception of the effort by reducing the perception of the obstacles from mechanisms.

2.6.2. Perceived benefit from secure behavior

When individuals decide to use security measures effectively and behave securely, this may incur individual costs as laid out in the previous section. However, individuals also benefit from secure behavior. Beautement et al. (2008) give the examples of avoiding consequences of a security breach and of sanctions as benefits. These factors mostly concern the personal security of the individual, preventing disciplinary measures in case of breaking formal rules or coworkers' contempt when breaking informal rules. Personal security also encompasses dangers to the personal data of individuals, for instance, when they share passwords that also give access to the data on the personal salary in HR self-service. Benefits may also be directly awarded as personal gains, for example, when secure behavior is part of the personal performance objectives. More indirectly, benefits occur from internalized organizational security goals, when secure behavior reduces the risks to "their data" and secure behavior is abstractly seen as "doing the right thing" (cf. Inglesant and Sasse, 2011).

Both personal and organizational risks have the problem of the abstract nature and the subjective perception of risks. Generally, objective risks are difficult to define and social constructivists even deny any objectivity in risk, since risk is always based on social norms, values, and risk framing (Arnoldi, 2009). Statman (2010) found that risk tolerance is associated with culture and that tolerance is relatively low for individualistic and egalitarian countries. Borge (2001) lists common risk perception pitfalls, including overconfidence and optimism, faulty hindsight and pattern seeking, and inertia that prevents adequate reactions to risks. Similarly, the risk thermostat of Adams (1999) attempts to explain why people react irrationally with respect to risks, predicting that people are prepared to take higher risks the safer they feel.

Organizations can increase the perceived benefit in order to increase the likelihood of secure behavior. Beautement et al. (2008) argues, for example, that the Compliance Budget can be increased, that is, individuals will continue longer to comply with the rules. A sociological model of influencing individuals' behavior is given by Clegg (1989) in his "Circuits of power" (cf. Section 2.2.2 and Inglesant and Sasse, 2011) with its three interacting layers: the episodic layer of direct experience, the social layer of informal social rules, and the domination layer of mechanisms and formal rules. The episodic layer primarily acts through direct experience, for example, from commands from superiors. On the social layer, education and awareness can similarly increase the perceived benefit when organizational goals are internalized as motivation (cf. the understanding of risks as motivation in Section 2.4.2). On the dominance layer of the circuits framework, clear formal rules, such as, organizational security policies, combined with technical measures, such as technical restrictions, and organizational measures of effective sanctions can also increase the perceived benefit from secure behavior.

2.6.3. Organizational security

Since the behavioral models focus on the prediction of user behavior, they lack the ability to comprehensively support decisions on the design of security measures in organizations. To support design decisions, we need to incorporate the goals of the organization, as, for example, proposed by Parkin et al. (2010) in a prototype on password policies. Specifically, we need to go beyond the individual's trade-offs that govern the behavior and take on a *systemic perspective*: how to achieve effective and efficient security in the organizational or societal context.

In economic terms, the behavioral models are important since they let us consider the externalities of security behavior, that is, how the behavior of individuals affects others. Externalities can be positive, for example, when the behavior of one individual also reduces others' risks, or negative, when the behavior exposes others to risks (Pallas, 2009, Section 4.1.1). However, for organizations, the goal is not to achieve absolute security of every individual behaving perfectly, but adequate and efficient security. Organizations aim to expend the least effort necessary to achieve effective security according to the security needs. Pallas (2009) formulates this in economic terms: Organizations strive for the optimum payoff from information security. Although not accurately measurable, "the optimal level of security is reached where the marginal costs of an additional unit of security equal its marginal benefit". He differentiates between direct costs of implementing technology as well as the coordination, motivation, and supporting measures, such as awareness campaigns. Indirect costs incur from averse effects of information security, such as productivity losses from the time spent on security measures. To derive the optimal level of security, these costs need to be contrasted with the benefits. Direct benefits result from the reduction of risks from additional security measures, indirect benefits from further value of the measures, for example, from certifications that are the prerequisite for loans or contracts (Pallas, 2009).

2.6.4. An economical and organizational approach

This thesis aims to broaden the focus beyond the individual, who is the primary focus of prior research on authorization. In contrast, we will take both an economical perspective on the individual's behavior and an organizational approach to consider the context of use. Specifically, we will start out from the *organization's goals* – for example, the level of security and productivity. Accordingly, we will transfer the economic behavior models to an organizational perspective and model their factors as the *perceived effort of* and *perceived benefit from secure behavior* of individuals, similar to the "perceived costs and benefit" of Beautement et al. (2008).

2.7. Applying security usability to authorization

This chapter started out by reviewing HCI and software usability, listing principles, such as comprehensibility and flexibility, as well as usability engineering and research methods. To achieve usable authorization in systems, these principles, guidance, and methods are necessary to understand problems and improve measures. Similarly, the problems raised and the approaches presented on security usability in this chapter directly apply to the usability of authorization. We need to be aware of the security risks from usability failures, psychological and socio-technical factors, the complexity of security, and how security interacts with the primary tasks. Authorization usability can also make use of the broad range of approaches to security-usability problems and we can derive the following principles for improving the usability of authorization:

- *Reducing the burden:* A number of burdens are related to authorization measures, depending on the individual's perspective on the measure. A functional user, for instance, needs to request changes to the restrictions if they are inadequate, a supervisor may need to decide whether the request is legitimate, and an administrator needs to implement the change. Depending on the perspective and context, this also encompasses the application of the HCI principles of flexibility and maintainability (Section 2.1).

- *Making the abstract understandable:* The risks and benefits of restrictions as well as the technical concepts in authorization can be highly abstract. Making these abstract concepts understandable or turning them into more concrete problems may help with the usability of authorization. Related HCI principles are the comprehensibility and learnability of the artifact.

We have seen that the separate consideration the individual behavior often falls short of addressing the problems effectively and efficiently. According to the sketch of affected perspectives in the introduction (Figure 1.1), prior research focused primarily on the HCI part, but neglected the interrelation with organizational sciences and software engineering. As a result, typical approaches in security usability (and particularly authorization) primarily focus on the *ability* of users and neglect the *motivational* side. However, even if generally able to behave in a secure manner and even if the extra effort is rather small, any effort will be perceived as a hassle if the rationale for this secondary task is unclear. In addition, we can observe an ongoing drive for productivity for functional stakeholders, increasing stress levels and limiting the ability of stakeholders to efficiently use the security measures. As part of this drive, security-relevant decisions are increasingly delegated from security experts to lower functional management. Thus, it is the responsibility of the organization to bring the affected stakeholders into the position to balance secure behavior and the push for productivity, enabling motivational forces to cause secure behavior. For this, the behavior of users need to be understood in the context of use of a system and organization, taking a broad range of interrelated effects into account, ideally applying HCI, but also organizational sciences. As discussed in Section 2.6, we need to comprehensively model the user interaction and behavior:

- *Integrating perspectives: Adequate security management and participatory development:* Authorization usability can also be impacted by how well the stakeholders with different perspectives on the measure can interact. This may be, for example, supported by the HCI principles of robustness and comprehensibility.

- *Applying a systemic model of individuals' behavior and take the specific context into account:* In authorization, the security needs of organizations relate, for example, to how critical data and processes are that are governed by restrictions. Adapting authorization restrictions to share a document among employees may incur a higher perceived effort than simply sharing the password with the coworker. Perceiving the proper use of the authorization measures as beneficial can help to reduce these types of circumventions. We need to analyze problems and support decisions on approaches to the problems based on systemic models of individuals' behavior and organizational needs. Such holistic models can improve the design of authorization measures when decision makers explicitly take the specific context – and its dynamics – into account.

3. From plan-driven to agile security

kein Ding, kein Ich, keine Form, kein Grundsatz sind sicher, alles ist einer unsichtbaren, aber niemals ruhenden Wandlung begriffen. . . und die Gegenwart ist nichts als eine Hypothese, über die man noch nicht hinausgekommen ist.[1]

R. Musil: Der Mann ohne Eigenschaften (1930)

In authorization in information systems, permissions and restrictions are technically codified. The challenge is to arrive at adequate restrictions that reflect the security requirements and the functional requirements of the stakeholders[2]. In most environments, we can expect the context to change, for example, due to changes in the organizational structures or individual task assignments. These changes, in turn, require adapting the restrictions. The process of defining, assuring, and adapting the restrictions as part of the technical artifact can be likened to the software development process, specifically, to the process of securely developing software. In secure software development, the challenge lies likewise in balancing security measures in the system against their impact on non-functional requirements, such as performance and usability. Building upon the analogy between software development and authorization, this chapter analyzes how secure software development achieves adequate measures and how we can transfer approaches to authorization.

Software development in general has undergone several paradigm shifts since its inception, particularly regarding security. The traditional approach to software security was to rapidly fix vulnerabilities when discovered. Two developments have significantly reduced the viability of this approach: First, the likelihood of attacks has increased dramatically since vulnerabilities are exploited at significantly faster due to the systems increased interconnectedness, for example, through "computer worms"[3]. Second, the potential impact of these attacks has increased to the point that "real" assets (e.g. critical infrastructure) are threatened by "virtual" attacks[4]. Together, these factors have increased the risk induced from a lack of software security and ultimately have led to a paradigm shift towards more comprehensive approaches to security in the software development lifecycle (SDL). Not only the reliability of the software has been addressed, but also the reliability of the development processes (Gollmann, 2011). As a result, stricter development processes increasingly regulated software development.

In a parallel development, the agile development community successfully spread the idea of deregulating the software development process, also in an attempt to improve software quality. Despite the similar aim, the two approaches – a more regulated development model for reliable software development and the agile principles – are in stark contrast. Agile development is claimed to have

[1]"no thing, no self, no form, no principle, is safe, everything is undergoing an invisible but ceaseless transformation. . . and the present is nothing but a hypothesis that has not yet been surmounted." (Musil, 1995, p. 269)

[2]Compare the discussion of the conflicting security goals confidentiality, integrity, and availability in Chapter 2.

[3]The Morris worm from 1988 was one of the earliest reported computer worm (Spafford, 1988).

[4]A widely publicized case in point is the malware "Stuxnet" that targets SCADA systems, apparently primarily uranium enrichment facilities (Albright et al., 2010).

several shortcomings related to secure software development. For example, heavy-weight security methods are not applicable in short incremental iterations of agile development in the same way as in plan-driven processes. Also, the agile development model focuses on functional requirements and does not provide well for non-functional requirements, such as high-level quality and security objectives. However, agile approaches have also advantages that make them particularly useful for achieving adequate security measures, including the close customer integration.

This conflict makes it interesting to contrast agile security with the traditional plan-driven approaches in our search for approaches to problems surrounding authorization. In this chapter, we will briefly review secure software development methods in plan-driven development, and agile development. Subsequently, we will explore the primarily theoretical literature on security in agile development. We close the chapter with a discussion of how to transfer the plan-driven and agile approaches to authorization.

3.1. Security in software development

Software security vulnerabilities are those problems of software systems that can be exploited to degrade the system's security and are caused by software defects (Software Engineering Institute, 2002). To classify the software defects, we may refer to categories of general software faults. However, there is a high variance in the proposed categories of software faults and no established set of factors (Ploski et al., 2007). Since the early categorization of defects by Endres (1975) as a mix of triggers (user error) and affected artifacts (documentation error), a number of different approaches for systematic categorization have been proposed. Schneidewind and Hoffmann (1979) grouped defects on the highest level primarily by (development) activity (e.g. design, coding, debugging, clerical errors). Chillarege et al. (1992) rather differentiate on a lower level: by defect type (e.g. function, assignment, interface) and the defect trigger (e.g. boundary condition, bug fix, user code, timing). Leszak et al. (2000) considers, among others, the development phase of detection and origin of the defects, but also the human root causes (e.g. lack of system knowledge, individual mistake, change coordination, lack of process knowledge).

For our purposes of secure software, one important category is the *development activity* from which a defect originates: In requirements engineering, security requirements may be neglected or risks underestimated. In the design phase, software architects may overlook deficiencies of the security architecture, for example, caused by false assumptions about the security of an underlying platform. In implementation, developers can introduce defects at the source code level, for example, through missing validations of input data. In assurance, defects can slip through the tests because of an insufficient test coverage. In operation, configuration defects can facilitate misuses of an otherwise secure system. Lastly, activities before and accompanying the process may result in problems, for example, when security training is skipped in the process.

Secondly, the *type of defect* as their human root cause (cf. Leszak et al., 2000) is important for our perspective. Following the human error model of Reason (1990), we can broadly categorize human root causes into lapses and conceptual errors: *Unintentional errors* are introduced because the action, such as a source code change, was not as intended, often a "lapse" resulting in a syntactic error ("Individual mistake" in Leszak et al., 2000). *Intentional errors* are caused by conscious actions, for example, conceptual defects from not understanding the context thoroughly, often a semantic error (e.g. "Domain knowledge" or "Process knowledge" in Leszak et al., 2000).

The goal of security in software development now is to minimize the number of defects of the different kinds in the development activities. How the defects are actually addressed depends very much on the development process. In the following, we will briefly discuss approaches in the traditional plan-driven processes and those in agile development.

3.2. Plan-driven software security

As elaborated in Appendix A, plan-driven software development differentiates several development phases, such as requirements engineering, implementation, and assurance. Security SDL thus either enrich the process itself, such as establishing risk-management process accompanying the SDL, or modify the individual development phases with security practices – for example, by adding threat modeling activities to the requirements engineering phase (cf. Section A.3).

For these approaches to be successful, plan-driven software development heavily relies on formalized processes to achieve secure software through a high level of control of the development activities. Each step is specifically defined, often in waterfall-oriented procedures. Organizational measures are implemented to assign roles and responsibilities, and to establish accompanying security training. Security requirements are elicited early in a development project through structured procedures that include extensive threat and risk analysis and the turning of non-functional security requirements into implementable functional requirements. Similarly, a security architecture is derived early from the security requirements, relying on security design principles and architectural risk analysis. Elaborate and heavy-weight assurance practices offer quality control before release and encompass dynamic and static analysis as well as internal or external reviews and audits. In operation, well-defined procedures on incident management and security documentation is employed to uphold the system security.

Overall, the plan-driven approach is based on control by extensive planning and process formalization, relying on the early, complete, and precise specification of the context-of-use and a static development environment. However, this approach is problematic when plans need to be adapted and when the environment changes, as is to be expected in today's development environment.

3.3. Agile development

Agile development follows an opposing approach from the formalization and control in plan-driven development. Instead, agile development formalizes processes only where necessary and emphasizes informal and intensive interaction to craft systems with high business-value. Agile development refers to a set of values shared by related development methods, such as SCRUM and XP. The Agile Manifesto (Beck et al., 2001) lists the overarching values in abstract terms:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

In agile development, responsiveness is emphasized over the reliability of standardized development processes. The process is more likened to learning than to the application of prior knowledge. Nerur and Balijepally (2007) describe agile methods as "generative" instead of "adaptive" learning, applying double-loop learning. Also, the "command and control" approach of plan-driven models is exchanged for a more democratic model to profit from the tacit knowledge of the individuals in the team (cf. Miles et al., 1978). More specifically than the Agile Manifesto, Highsmith (2002) states that the principles of agile development are to "deliver something useful," "rely on people," "encourage collaboration," "technical excellence," "do the simplest thing possible," and "be adaptable."

As shown in Appendix B.2, popular methods in the agile ecosystem have a broad range of characteristics. More liberal methods, such as Crystal Clear and Scrum, formalize the development process to a lesser degree than more heavy-weight methods, such as Feature-Driven Development. Two distinct philosophies can be contrasted here: The liberal methods with an optimistic view that development teams are able to tailor the process to fit their particular development environment in the most efficient way. Conversely, the pessimists rather specify the process in detail to prevent failures from adaptation or problems in large or high-reliability projects.

A number of negative points have been raised since agile development spread in practice (Dybå and Dingsøyr, 2008). One argument is the lack of empirical backing for the claims made by proponents of agile development for their methods (McBreen, 2002). In particular, with the distinct agile methods and the adaption for the individual development environment, it is difficult to draw general conclusions from the existing evidence (Keefer, 2003). Moreover, it has been argued that pre-project release planning in market-driven development can be difficult, particularly for release-content planning when the market pull needs to be balanced against the technology push (Dzamashvili-Fogelström et al., 2010). It is also often stated that the customer is heavily burdened by the constant developer requests, often at an unsustainable level (Martin et al., 2004, 2009). Moreover, it is put forth that agile methods are rather applicable for projects with small teams since these do not require as well-defined communication (Cohen et al., 2004b). Lastly, opponents fear that suboptimal architectures result from the lack of focus on design activities (McBreen, 2002).

We draw a thorough picture of the agile ecosystem in Appendix B to discuss the existing evidence, popular development methods, and their practices. The close look at the breadth of agile development allows us to derive two useful insights: First, despite numerous studies, the performance of agile development in comparison to plan-driven approaches remains unclear. While the self-perception of developers in surveys is rather positive, experimental studies show mixed results, and are also sparse and often only apply to a limited scope. More specific studies show that the motivation may increase, and learning and communication may improve. Generally, we can assume that universal findings for the diverse development contexts are unrealistic: The appropriateness of agile methods depends on too many factors. Second, general findings are complicated by the breadth of agile methods and practices that practitioners can choose from, particularly related to the respective extent to which they conform to the agile values. For security, it follows from the two insights that, first, agile development can occur in many flavors and the development method needs to be chosen and adapted to fit the specific development environment. Second, positive effects on quality, motivation, communication, and learning in agile development are promising. To extend on these insights, we will further discuss the relation between security and agile development in the remainder of this chapter.

3.4. Security in agile development

In literature, there is a discussion on whether agile development methods and the underlying principles are appropriate to develop secure software. One reason is that the agile development proponents did explicitly not target high-risk software development. Kent Beck rather states in his XP book that XP in itself is not suitable for high-reliability requirements (Beck and Andres, 2004). However, security is not only relevant for high-reliability projects, but affects most software that is being developed.

The main issue with agile development concerning security is that the team-emphasizing, dynamic and tacit-knowledge-driven methods conflict with the assurance activities as demanded by traditional secure software development methods. However, there are indications that agile development improves quality, as discussed in Section B.1. Moreover, plan-driven development also poses challenges to secure software development that might be less critical in agile development: Early planning of security requirements may conflict with the changing requirements in practice, which agile development is better prepared for. Also, to address the challenges to security in agile development, various enhancements have been proposed to agile methods.

In the following, we discuss the security challenges that have been reported to impact agile development and explore the positive effects of agile development, attempting to contrast the advantages to the challenges. In a third step, we examine the numerous proposed security enhancements for agile methods.

3.4.1. Security challenges

Most challenges in achieving secure software development with agile methods result from difficulties in implementing security-related practices that were originally targeted at plan-driven software development. In several publications, authors have studied how compatible the agile development methods are with security practices, primarily analytically. Wäyrynen et al. (2004) analyzed how well software development certifications for SSE-CMM and Common Criteria (CC) match with XP. Although they found overlaps between the programs' requirements and stock XP practices, 5 of 11 SSE-CMM process areas were not covered by XP and the semi-formal and formal design and verification requirements for CC EAL above level 4 were not present in XP. More generally, Beznosov and Kruchten (2004) studied mismatches between XP and secure development practices, identifying 12 mismatches out of 26 practices. Similarly, Keramati and Mirian-Hosseinabadi (2008) evaluated the compatibility of security activities with agile development. Mellado et al. (2006) compared security requirements engineering methods for their agility value on a high level. We discuss the findings in the following, grouped by the agile characteristics that they originate from:

Lifecycle model One of the topics that are most often cited to pose challenges to the traditional secure development methods in agile development is the agile lifecycle model. Instead of one pass of the typical development activities, the activities are repeated in short intervals and not in a fixed sequence. From the perspective of plan-driven secure software development, the difficulty is where to place security practices in an agile process. For the auditing of agile processes, the dynamic nature of the lifecycle can be difficult to handle with the traditional measures because the process to audit against is not as clearly defined (Poppendieck, 2002). Also, in the short iteration time frames, repeating security engineering activities as in the plan-driven model for each pass is not feasible. The heavy-weight nature of the security activities results in difficulties to fit the activities into the short iteration cycles of agile development. Even more problematic are the prohibitive costs that would be caused when involving third parties for assurance for each iteration (Beznosov and Kruchten, 2004). Independent reviews would also cause months delay, conflicting with the goal of frequent deliveries (Goertzel et al., 2006).

Agile development is tailored to react to changing requirements as part of the development model. The continuous changes to requirements and, subsequently, to the design pose challenges to the traditional security practices that assume constant environments. New vulnerabilities or changing threat models would require backtracking or iterative processes and result in changes to all intermediate deliverables (De Win et al., 2009). The concept of up-front activities in plan-driven security processes, such as threat modeling, conflicts with the emergent requirements in agile development (Davis, 2005). However, security is also difficult to retrofit (Chivers et al., 2005; Wäyrynen et al., 2004). In particular, refactoring and major design changes late in the lifecycle clash with traditional assurance approaches (Beznosov and Kruchten, 2004; Goertzel et al., 2006). As the software artifact is continuously changing, assurance is difficult to conduct in parallel to development activities, since there is no freeze period in the lifecycle (Goertzel et al., 2007; Beznosov and Kruchten, 2004).

Non-functional requirements, such as security requirements, also are a general challenge in agile development (Goertzel et al., 2007). One problem is that they are difficult to turn into automatic tests. In agile development, refactoring is important to constantly adapt to changing plans and refactoring relies heavily on testing to ensure that invasive changes do not cause regressions and, in this case, vulnerabilities. Due to the incremental planning, functional requirements that offer more immediate value-add often receive higher priority, and non-functional requirements are ignored in the beginning. As a result, security and quality aspects may be postponed until late in the lifecycle (Cao and Ramesh, 2008). Generally, it has been observed that activities are difficult to implement if they involve non-functional artifacts, such as documentation (Goertzel et al., 2007).

Communication A related aspect that is often claimed to be problematic is the agile approach to communication and documentation. Agile development emphasizes the building of rich tacit knowledge through informal communication instead of formalized documentation. In contrast, security engineering traditionally requires security documentation and specifications for traceability. For example, the traceability of requirements (Poppendieck, 2002) allows auditors to trace functionality back to the requirements and identify the team members responsible for specific decisions and their implementation. Agile development, instead, emphasizes face-to-face communication, avoiding the creation of many documents that are typically produced as part of the communication between stakeholders and developers in plan-driven development models (Goertzel et al., 2006; Beznosov and Kruchten, 2004). Moreover, overly verbose documentation may also be counter-productive in agile development since the system's requirements and design are often moving targets, causing a mismatch of documentation and implementation to an even larger extent than in plan-driven development. Instead, agile methods often rely on implicit documentation to provide additional communication media, for example through source code and test cases.

Trust in the team and individuals Instead of sophisticated command-and-control measures to control the developers, agile development places more trust in and responsibility on the development team. The implicit trust in individual developers and their benevolence conflicts with the skeptic approach of traditional security processes, which fear malicious developer activities (Goertzel et al., 2006). Similarly, the objectivity of separate roles is missing in test-driven development when the same developer conducts the quality assurance who also implemented the functionality (Goertzel et al., 2006). Moreover, agile methods employ cross-functional teams, which often bring evaluators and developers close together and further reduce the objectivity from the separation of developers from evaluators (Beznosov and Kruchten, 2004). It is also argued that close customer participation cause a lack of distance between customers and developers, limiting the ability of customers to judge the resulting product objectively (Goertzel et al., 2006). In addition, specialist expertise of security managers may be missing without an explicit security expert role (Goertzel et al., 2007, 2006). Following the same line of argument, pair programming only improves security if enough team members are security aware (Wäyrynen et al., 2004).

3.4.2. Inherent positive effects on security

While the agile development community originally did not target improved security of the software, they did aim to produce superior system quality. This goal was primarily motivated by the increased business value for the customer and by the increased developer satisfaction from producing high-quality software. According to Cockburn (2001), the entire agile development process focuses on software quality. As discussed in Section B.1, there are indications of positive effects of agile methods and practices on the quality of the developed software, primarily from case studies and developer surveys. Quality and security are closely related: According to an analysis of vulnerability reports by CERT, software defects cause 90% of software vulnerabilities (Software Engineering Institute, 2002). Wang and Wang (2003), from a more theoretical perspective, found tight relations between quality attributes and security objectives. The positive effects of agile development on security, thus, foremost relate to the prevention of security-critical defects through improved software quality. Other general effects of agile development indirectly influence the software security, for example, through developer motivation and commitment.

There has been little empirical research up to now on the effects of agile development on software security. The existing studies treat this subject analytically: In an analysis of XP for its compatibility to SSE-CMM and the Common Criteria, Wäyrynen et al. (2004) found, for example, that XP is well aligned without modifications with 4 of 11 SSE-CMM process areas. They also reasoned that, if test cases are considered documentation, XP could fulfill Common Criteria certifications for EAL levels 1 and 2. In the following, we more generally discuss the positive effects of agile practices on software security, based on a literature review of security in agile development and on empirical findings on general quality attributes (cf. Section B.1):

Lifecycle model Agile development's ability to respond to changes in the course of the SDL is primarily due to the iterative and incremental lifecycle model. This approach can have positive effects on reliability since it has been long known in the software development community that prototyping and running code improves requirement engineering and reduces design errors (Boehm et al., 1984), a potential source of security vulnerabilities. Incremental and iterative development models can also positively affect developer motivation, for example, reducing "crunch time" and overtime, and improving the quality of life (Robinson and Sharp, 2004; Mann and Maurer, 2005). Higher motivation and satisfaction may improve the quality (cf. Section B.1).

The agile lifecycle model also causes development projects to be better prepared for the inevitable changes. As with most requirements, it is questionable whether security requirements may entirely be formulated up-front as assumed in plan-driven security processes. For example, security requirements need to take operational practice into account (Poppendieck, 2002). An emphasis on refactoring may result in a more maintainable code-base with less complexity, and higher quality and security, since simpler designs are, among others, easier to review for security (Wäyrynen et al., 2004). Another consequence of the lifecycle model is the high importance of automated tests (cf. Section B.3), which may improve software quality (Hamlet and Maybee, 2001), and support finding defects early (Berinato, 2002; Huo et al., 2004). Test-driven development has shown to improve software quality in several studies (George and Williams, 2003, 2004; Meszaros et al., 2003).

The openness to change also encompasses the tailoring of the development process to improve it and accommodate the specific environment. Having an adequate process is important since it is often stated that security lies in the processes (Goertzel et al., 2008). Agile developers perceive reflective process improvement to be highly efficient (Talby et al., 2006).

Communication Agile development fosters the communication between stakeholders, reducing defects by improving the common understanding of the requirements and by preventing misunderstandings. Firstly, agile development improves the internal communication between developers (cf. Section B.1) Studies indicate that communication-rich teams also increase the developer motivation

(Beecham et al., 2007) as team members encourage each other (Robinson and Sharp, 2004). Informal communication leads, on the other hand, to less documentation as the medium of communication, and an increase of implicit documentation, such as source code or test cases. Implicit documentation may reduce the number of potentially obsolete documents, which else cause defects from misunderstandings and other typical problems found in implementation that is based on specification documents (Hoffmann, 2008).

The increased internal communication can also lead to improved learning, sharing of knowledge, and spreading of security awareness. The sharing of knowledge has been indicated to increase in agile development on the level of tacit knowledge (Bahli and Zeid, 2005). Also, pair programming increases communication and explaining (Robinson and Sharp, 2005), and improves learning (Tessem, 2003).

Secondly, communication with customers improves through agile development (cf. Section B.1). It is well-known in software development that good designs require deep domain knowledge (Curtis et al., 1988). Improved communication can improve the requirements definitions (Dagnino et al., 2004) and reduce defect rates (Korkala et al., 2006). There are also secondary effects from improved external communication: More direct communication helps the developers to understand the functional users' needs, resulting in improved usability that can prevent security-critical usage errors (McInerney and Maurer, 2005; Hwong et al., 2004). Customer feedback also motivates developers (Robinson and Sharp, 2004) and improves the developer confidence (Hanssen and Fægri, 2006).

Trust in team and individuals In the agile methods, the principle of highly valuing the individual can be found in the responsibility that is given to teams as a whole, commonly reflected in the team structures and empowerment to self-organization (Siakas and Siakas, 2007). Self-organization and responsibility may be found in reflections and retrospectives (Hazzan and Tomayko, 2003) and can lead to higher motivation and satisfaction from the trust in developers and empowerment on the engineering level (Robinson and Sharp, 2004; Karlström and Runeson, 2005; Beecham et al., 2007). In an experiment, job satisfaction improved when the team could modify development methods (Acuña et al., 2009).

Agile teams are usually formed with cross-functionally, emphasizing the team effort with high involvement of all team members and shallow hierarchies. As a result, the collaboration between team members with different functional roles is improved and defects from misunderstandings can be reduced. In agile development, developers are often more involved in quality assurance, for example through test-driven development (Dubinsky and Hazzan, 2006). Quality assurance by developers may improve the awareness of quality measures (Cohen et al., 2004a) and allows the measures to be applied earlier and more effectively.

Agile development further emphasizes the team effort through close developer collaboration, found in its most intensive form in pair programming. Since multiple team members work together, pair programming can be seen as implicit security reviews, both for design and implementation (Wäyrynen et al., 2004; Beznosov and Kruchten, 2004). Even without pair programming, design decisions are often discussed between team members, supported by the culture of close collaboration. A further wide-spread practice in agile development is common code ownership, which also result in implicit, though non-systematic, source code reviews. Generally, review activities have shown to increase productivity, quality, and project stability (Fagan, 1986).

How inherent effects address security problems The agile methods take the opposite approach of the plan-driven focus on command and control and detailed planning: Agile development relies on the commitment and creativity of individuals working closely together, and the openness to changes at various stages. The empirical foundation of the claims of improved quality from agile development remains weak, but there are indications that the defect rate is reduced in appropriate environments. Similarly, there are several indications that agile development improves the motivation and satisfaction, which may improve software quality indirectly through increased commitment and more efficient employment of developer talent (Baddoo et al., 2006). Given that agile development can realize those effects in a specific environment, agile methods may achieve improvements of reliability.

Table 3.1 contrasts the security challenges from the previous section with positive effects discussed in this section. As shown in the table, most drawbacks affect the process reliability and the missing assurance that activities are completed as necessary and the software, thus, adheres to security standards. Related to the assurance aspect is the difficulty of traditional approaches to software security to cope with the trust that is placed in agile developers. In the following section, we discuss proposed enhancements of agile methods that address the challenges.

3.4.3. Security practices in agile development

As discussed in the previous sections, agile methods may lack proper means to address requirements of secure software development, particularly with respect to the process reliability, requirements traceability, and the trust in individuals and teams. Within the published work on security in agile development, numerous enhancements have been proposed to address those shortcomings. There are two main categories of enhancements: Either they modify the method in its entirety, or add individual security practices. The enhancements need to be simple, have pragmatic and complete guidance on how to apply, and adapt to the frequently changing requirements (Siponen et al., 2005).

Goertzel et al. (2006) analyzed the implications of agile principles on security and formulated conditions under which problems may be alleviated: In 6 of the 14 examined areas, customer and developer security awareness and commitment would help. The remaining two problematic areas would benefit from additional security documentation and high security standards on every delivered software artifact.

Multiple analyses exist of how security practices can be applied in agile development and how well the practices fit into the agile methodology. Beznosov and Kruchten (2004) found that of 26 studied security practices, 12 are either independent of the development process or may be conducted (semi-)automatically to fit agile methods. Keramati and Mirian-Hosseinabadi (2008) examined the compatibility of security practices according to multiple metrics, including simplicity and customer interaction. Since a number of enhancements apply to multiple challenges, the enhancements from literature are categorized according to the development activities in the following:

Process-spanning practices Very broad approaches to introduce security enhancements to agile development modify the method, such as the Microsoft SDL for Agile Development (Sullivan, 2008). Based on the Microsoft SDL (cf. Section A.2), the plan-driven activities are divided into “one time,” “some sprint,” and “every sprint” activities. Developers conduct one-time activities at the beginning of the project, for example, configuring tools properly and creating a baseline threat model. Every-sprint activities include, for example, updating the threat model. Most activities are in one of three buckets of some-sprint activities: security verification, design review, and response planning. The developers choose one activity from each bucket for every sprint in order to limit the security overhead.

Other suggestions on enhancements of agile processes focus on process-wide practices. Dedicated security training (Ge et al., 2007) and systematic knowledge transfer that is achieved by rotating security experts through programming pairs (Wäyrynen et al., 2004) increase security awareness

	Challenges	Positive effects	Enhancements
Lifecycle model			
Reflective process improvement	Difficult to assure changing process	Continuous improvement of development process	–
Openness to changes	Up-front activities conflict with emergent requirements	Improved requirements	High-level architecture up-front
	Security is difficult to retrofit	Fits risk-driven development	Security refactoring, security testing
	Major requirements and design changes late in the lifecycle	Less complex designs and systems	High-level architecture up-front, iterative security design
	Assurance cannot be conducted in parallel to development	Dynamic quality assurance early-on Improved product quality	Late security sprint, (semi-)automatic testing and static analysis
Incremental development	Independent assurance not viable with short iterations and frequent deliveries	Improved motivation and satisfaction Frequent deliveries improve quality	Security and release sprints
	Non-functional requirements ignored early-on		Early security sprint, abuse cases, security requirements methods, expert
	Difficulties with non-functional artifacts		Security review meetings, sprint
Communication			
Customer participation	Less objectivity because of close developer/-customer relation	Less defects from improved requirements Improved team motivation	–
Implicit documentation	Missing requirements traceability	Less obsolete documentation	Explicit connections between security requirements and user stories
	Lack of security documentation		Security review meetings, security sprint, tests cases as documentation
Trust in team			
Team responsibility	Implicit trust in developers	Increased motivation	–
Emphasized team effort	Reduced assurance objectivity	Improved learning, motivation, awareness, transparency, common understanding	–
	Lack of specialist expertise on team		Training, rotating security expert

Table 3.1.: Contrasting security challenges with advantages and enhancements of agile methods

and expertise. Security knowledge may also be spread through an institutionalized security-review meeting for every sprint to address potential security implications of the user stories in the sprint backlog (Kongsli, 2006).

To more intensively work on security aspects, development teams can insert security design, documentation, and testing sprints early in the lifecycle and before release (Beznosov and Kruchten, 2004). Addressing the challenges from frequently changing architectures, a high-level security architecture and security principles may be defined up-front before the first iteration (Ge et al., 2007). When developers conduct system hardening and secure deployment from early iterations on, operations may be more secure at release time (Kongsli, 2006). Particularly for documentation issues, practitioners can also turn to more documentation-intensive agile methods, such as FDD and DSDM. Another approach is to employ automated test suits as documentation (Wäyrynen et al., 2004).

Requirements engineering For security requirements engineering in agile development, it is often proposed to employ modified user stories to better reflect security threats. Abuse(r), misuse(r) cases, and stories have already been proposed for plan-driven secure development (McDermott and Fox, 1999). For agile development, abuser stories have the advantage of matching the already established user story concept to describe cross-cutting development tasks (Kongsli, 2006; Boström et al., 2006). In several studies, abuser stories were empirically tested and showed promising results (Boström et al., 2006; Heikka and Siponen, 2006; Mellado et al., 2006).

To identify further security aspects in the requirements, methods may be applied on the sprint backlog to systematically identify security requirements (Aydal et al., 2006). If necessary, a security engineer on the team may help with assessing security risks and security-related user stories (Wäyrynen et al., 2004). Instead of conducting threat modeling at each modification of requirements, developers may limit the number of threat-modeling points in the process to prevent overly frequent re-evaluations of threats (De Win et al., 2009). Lastly, it is often important to clearly mark the dependencies between high-level security requirements and the related implementation activities for security assurance and requirements traceability. Developers can achieve the traceability through explicitly noting the connections between high-level security/safety requirements and user stories in the product and sprint backlog (Poppendieck, 2002).

Design and implementation The main issues with the design and implementation in agile development from the perspective of plan-driven secure software development are the dynamics of the architecture and functionality. In contrast, Aydal et al. (2006) report on a case study of incrementally introducing security through refactoring with good results. Similarly, Chivers et al. (2005) compared different approaches of arriving at a security architecture over multiple iterations. In their study, a top-down and up-front approach resulted in an overly complex architecture when compared to iterative design. More broadly, Beznosov and Kruchten (2004) suggest to implement security practices that are independent of the development model, such as secure design principles, coding standards, and change management to cover SSE-CMM process areas.

Assurance To formally test and review software in agile development, a prevalent challenge is that there is no phase in which the product is frozen. Instead, development is typically ongoing. In this environment, it is important to (semi-)automatically apply requirements, security, and penetration testing, and static analysis (Beznosov and Kruchten, 2004; Kongsli, 2006). Erdogan et al. (2010) describe a testing method based on misuse cases in the product or sprint backlog, a highly testable architecture, automatic code review, and penetration testing during sprints.

The efficacy of security enhancements The above-outlined security enhancements cover a broad range of practices. Table 3.1 relates the enhancing practices to the security challenges from the previous sections for an indication of which challenges are addressed. The enhancements foremost target lifecycle and communication aspects, addressing 10 of the 14 identified challenges. The challenges that are not addressed fall into two categories:

- Challenges that stem from the *increased trust in individuals*, including the lack of distance between developers, evaluators, and customers, lack of independence in testing, and implicit trust in developers' benevolence. These challenges are difficult to address, since trust is a very essential agile value. In projects where the individual cannot be relied upon, agile development might not be the adequate development model.

- Challenges surrounding the *process assurances*: Continuous process improvement is an integral part of how high quality is achieved with agile methods. Certification and process assurance programs might need to be adapted to better apply to non-plan-driven models and acknowledge positive effects from well-suiting processes.

While the coverage of the challenges by positive effects of agile development and by enhancements appears broad, the analyzed literature is of rather theoretical nature and the problems and approaches are context-dependent. Thus, empirical studies in a variety of contexts are necessary to further explore the findings that are given here. Accordingly, we address the practitioner's perspective of security in the next section through interviews with agile developers.

3.5. Agile security in practice

In the previous section, we showed numerous problems surrounding security in agile development and respective approaches as present in literature. However, the analyses primarily target security-sensitive contexts. Since our goal is to understand how to improve authorization usability in organizational contexts, we should not exclusively focus on highly sensitive contexts. Instead, we need to understand the interrelation of security and agility particularly for projects with baseline security requirements. For these cases, rather than asking whether agile methods are adequate, we need to focus on how to improve the security in typical agile contexts, since the method is often a given.

The second problem with the analyses in the previous section is that they are primarily theoretical. For a good grounding of how agility and security can be combined, we need to expand on these analyses with insights from practice. Accordingly, we conducted a study on the practitioners' perspectives on security in agile development (cf. Appendix C). The findings indicate heterogeneous problems and approaches to the challenges in practice. Several problems, such as the problematic customer awareness, have not been considered in the prior theoretical research. As expected, several of the problems theorized in prior research are not applicable in non-critical environments and were not mentioned by practitioners. More interestingly, we found a range of approaches of practitioners that go beyond those in prior research and particularly leverage the strengths of agile development ("close customer integration", "spreading awareness and expertise").

From the results of the study, we derived a number recommendations on how to approach agile security, which may be relevant for agile authorization: First, to institutionalize the *customer involvement*; second, to foster developer *security awareness and expertise*; third, to continuously *improve the process* for security; and, fourth, to promote *implicit documentation*.

3.6. Contrasting plan-driven and agile authorization

In this chapter, we compared the rigid, plan-driven paradigms of traditional secure software engineering with agile approaches to security. On a general software development level, Nerur et al. (2005) contrasted key concepts of plan-driven and agile methods. Table 3.2 expands their general observations in two ways: by adding examples of plan-driven and agile approaches to secure software engineering from this chapter and by extrapolating how these approaches apply to the development and management of authorization policies[5]. The primary difference on the level of secure software development is that agile methods are particularly suitable for dynamic and complex contexts that may require changes to the development process as well as to the security requirements, design, and implementation. The agile approach closely integrates customers and trades controllability and traceability for increased motivation, productivity, and quality.

As elaborated in the introduction of this chapter, in order to derive insights on authorization, we can consider the development of an authorization policy as a development activity analogous to that of a software system. In plan-driven authorization, the authorization policy would primarily be developed in a single-pass with an upfront requirements and risk analysis, and subsequent design, authoring, and assurance activities. If requirements change, change requests would need to be issued and pursued. A mechanistic context (cf. Table 3.2) is assumed to allow the full top-down specification of a policy, for example, based on interviewed stakeholders. Formal, centralized change procedures define clear responsibilities and offer together with explicit documentation of requests the traceability of changes.

For agile authorization, we can turn to the approaches of the agile practitioners regarding authorization (Appendix C). They stated that policies need to be frequently adapted for functional changes and from the experience in production. The practitioners also strive for simplicity of policies first, then adapting as needed. Customers are involved primarily through natural language discussions and formal documents, with good experience with implicit documents.

Extending this line of thought (cf. Table 3.2), agile authorization would assume unpredictable, continuous changes of the organizational contexts and the policies. To provide for this environment, the following principles can be derived from agile methods:

- *Close participation of different perspectives, including functional stakeholders, for decisions and configuration:* The tacit knowledge of the stakeholders from different perspectives can provide valuable input (cf. Miles et al., 1978). They should thus closely interact in activities related to authorization, including in the decisions on, authoring of, and assurance of changes to the policy.

- *Adaptable, light-weight, informal, and decentralized measure:* Policy management needs to adapt to the specific environment. For instance, the surrounding organization – from the request of a functional stakeholder over the decision on its appropriateness to the actual change – would tend to be light-weight, informal, and decentralized to facilitate rapid changes with low overhead – if adequate.

- *Security awareness and expertise for a broad range of stakeholders:* Integration of stakeholders and the decentralization of decisions and authoring requires homogeneous awareness and expertise for the participating stakeholders.

For the dynamics of changing organizational contexts, the agile paradigm seems to be a viable approach. However, the loss of control and traceability of changes may be inadequate for many

[5] For a definition of the term "authorization policy" in this thesis refer to Chapter 4

	Plan-driven	Agile
Organizational form/structure	Mechanistic (bureaucratic with high formalization), fully specifiable and predictable	Organic (flexible and participative), changing context
Security	Well-defined, constant context (risks, requirements); upfront security architecture	Adapting the process, simple, incremental security design, close customer integration
Authorization	Predictable, stable permission requirements from rigid organizational structures and tasks	Continuously changing permission requirements, unpredictable upfront, striving for simplicity
Process model	Lifecycle model (Waterfall, Spiral, . . .)	Empirical, evolutionary-delivery
Security	Predefined and plan-driven process	Adapting the process to fit the context, iterative/incremental security design
Authorization	Primarily upfront policy authoring; stable, heavy-weight/slow policy change procedure	Evolving policy; rapid, light-weight, adaptable policy change procedure
Management style and roles	Command-and-control, process-centric, individual roles – favors specialization	Leadership-and-collaboration, people-centric, self-organizing teams
Security	Well-defined procedures and processes, security expert in team	Homogeneous awareness/expertise, security review meetings, holistic development
Authorization	Top-down elicitation of permissions; centralized, formalized process to request policy changes	Bottom-up; decentralized, informal process with functional stakeholders deciding on changes
Knowledge management	Explicit	Tacit
Security	Security documentation as interface between phases, e.g. security requirements	Developer spreading expertise, close customer integration, implicit documentation
Authorization	Additional policy documentation	Policy as documentation
Communication	Formal	Informal
Security	Security document-based, traceable requirements	Implicit code reviews, security review meetings, close customer integration
Authorization	Formalized, based on request forms	Informal requests
Customer role	Important	Critical
Security	Interviewed for security requirements, late assurance	Close customer integration, concrete requirements and threats
Authorization	Functional stakeholder request, review policy changes	Functional stakeholders participate in requesting, authoring, and reviewing changes
Assurance strategy	Planning, control; late assurance	Continuous, rapid feedback on artifacts, continuous testing
Security	Heavy-weight manual and automatic static and dynamic analyses, primarily late in the lifecycle	Close customer integration, non-functional requirements as done-definition, automatic testing
Authorization	On-request and regular policy reviews	Participation of functional stakeholders for policy changes

Table 3.2.: Plan-driven and agile approaches to secure software development and authorization

organizations. For the remainder of this thesis, a number of interesting research problems arise from this observation: While the results of the study on security in agile development indicated that agile security can be adequate for systems development, it could be challenging to carry over this approach to policy changes. Can the risk of intentional and unintentional misconfiguration of authorization be mitigated or accepted? Does the benefit of rapid changes and, thus, more adequate policies offset these additional risks? Will approaches similarly depend on the specific organizational context?

Part II.

The usability of authorization

4. Authorization in organizational information systems

[L]aw is best understood as a 'branch' of justice and that its congruence with the principles of morality rather than its incorporation of orders and threats is of its 'essence'

H.L.A. Hart: The Concept of Law (1994, p. 7)

Based on a cursory examination of the problems surrounding authorization in information systems, we formulated two research questions and six hypotheses on the problems and alleviations to address them in the introduction. Among others, the hypotheses stated that the flexibility of authorization, and the integration of different perspectives on authorization and research disciplines would be crucial to alleviate the problems. Accordingly, we identified principles on how to address the authorization problems from the fields of human-computer interaction and plan-driven and agile development in the first part. Following the methodology laid out in the introduction, we at this point need to more thoroughly examine the problems before applying the principles to authorization: We structure the problem domain, set the scope for this thesis, analyze the problems and existing approaches in that scope, and derive the requirements and open research questions to address.

So, what is *authorization*? To answer this question, it is useful to consider the broader use of the related term "authority". The political philosopher Raz (1990) notes three broader uses of the term: First, a person can be an authority in the sense of being a reliable source of information. More relevant for our context are the further two meanings: the sense of having the power to do something ("de-facto" authority) and of actually being meant or allowed to do something ("legitimate" authority, from being legitimized). Raz' second two meanings can be transferred to authorization in information systems, for instance, when users are *technically* able to access sensitive data ("de facto"), but know that they should not because they lack the *legitimacy*. Accordingly, we will differentiate the two forms:

- *Technical authorization:* Grant the authority in a way to technically (*de facto*) enable someone or something – the principal – to interact in a specific way with a system,
- *Legitimate authorization:* Grant the authority in a way to legitimate a principal to do something; for example, explicitly and organizationally through the delegation of a task, or implicitly through informal rules that may have been established in a social context.

At this point, we also need to clarify the relation of technical authorization to further terms in the area of information security, specifically *access control* and *authentication*. According to Lampson et al. (1992), authentication and authorization are part of access control. Access control is a two-step process: *Authentication* establishes the identity of the principal that requests to interact with the information system ("Who is the principal?"). *Authorization* establishes whether the principal is technically allowed to (has the authority to) do an activity in the information system ("Is the principal allowed to do the activity?"). In this dissertation, we will be chiefly concerned with *authorization* and assume the identity to be securely established.

What, then, constitutes authorization in information systems? Since this dissertation aims for a broad and integrative approach to authorization, we consider the *authorization measure* in an information system to consist of more than the technical mechanism. Authorization measures encompass:

- *Authorization mechanism:* The technical control to enact restrictions in an information system,
- *Definition of restrictions:* The technical and non-technical formulation of the restrictions to be enforced in the information system,
- *Organizational measures:* The management aspects that surround the mechanism and the restrictions, including the integration and assurance of the technical controls, the modification of the restrictions, and further supporting processes, such as monitoring compliance and enforcing disciplinary measures.

The definition of restrictions is often referred to with the heavily overloaded term "policy". The McMillan Dictionary defines policies generally as "a set of plans or actions agreed on by a government, political party, business, or other group"[1]. For information system security, Baskerville and Siponen (2002) explore the different perspectives on policy and distinguish between the management and the technical perspectives: *Management policies* limit activities and require tasks on an organizational level, for instance, through rules that forbid the sharing of passwords. The NIST Handbook on computer security (NIST, 1995) has similar management-level definition of "policy," differentiating the scope of policies between program, issue-specific, and system-specific policies. *Technical policies* define restrictions that the system enforces architecturally, for example, limiting the permissions of a user within the system. In this thesis, management-level policies will be termed *security policy*, technical policies *authorization policies* (or simply "policy"). To formulate the informal restrictions as technical policies, the authorization mechanism employs the policy as a model of the restrictions[2] – implementing the *authorization model* (e.g. allowing for roles and permissions).

Starting off from these definitions, this part of the dissertation aims to clarify what constitutes authorization in information systems, with a focus on organizations, to identify the problems surrounding this measure, and to compare the problems to existing alleviations to formulate detailed open research questions. In this chapter, we are concerned with the structuring of the problem domain, and explore the breadth of contexts and how we can systematically describe the contexts. We further discuss the perspectives that stakeholders have on authorization in organizations to lay a solid foundation for the examination of the specific problems.

4.1. Authorization contexts

When approaching authorization, we can observe a broad range of distinct contexts and designs of authorization systems. How can we, for instance, compare policies for smartphone applications that are formulated by the developers and acknowledged by the device users with permissions assigned centrally for organizational shared-folders? While these authorization contexts are very different and distinct approaches might be necessary to address usability problems in those cases, we can also identify common problems to solve. For example, the conflict of availability of resources in

[1] `http://www.macmillandictionary.com/dictionary/british/policy`

[2] For one definition of the term "model" we can refer to Minsky (1965): "To an observer B, an Object A* is a model of an object A to the extent that B can use A* to answer questions that interests him about A."

contrast to confidentiality and integrity recurs. Similarly, decisions on the policy need to be derived from complex factors and abstract consequences, requiring, for instance, to make the authorization model and concepts comprehensible for decision makers and policy authors. Identifying the range of authorization contexts will enable us to scope specific approaches and argue whether findings can be transfered to further contexts.

Existing definitions of authorization often remain on a technical level: subjects, objects, and the authorization model. Even when assessing authorization policies for the performance of different models, categories for policies are limited to these technical aspect instead of considering the authorization contexts (cf. e.g. Komlenovic et al., 2011). Moreover, the existing definitions also often selectively focus on human subjects. However, to comprehensively define the context of authorization as needed for an integrative approach to authorization, we are less interested in the technical detail of the mechanism, but need to include the socio-organizational aspects of authorization. In this section, we will thus develop a *taxonomy of authorization contexts* to structure the problem domain and contexts for which we expect to design authorization measures. While systematically developed, the taxonomy remains a hypothesis, primarily validated by its further use within this thesis: Apart from improving our understanding of the domain, the taxonomy will also allow us to define the scope of this thesis, estimate priorities of requirements with respect to context characteristics, and examine the applicability of the later developed artifacts.

In order to define criteria for the taxonomy, we follow a two-step approach. First, we will identify likely categories of contexts by examining textbook definitions of authorization and authorization models. Textbook definitions are typically technical and rather abstract. Those models hide the concrete context that we are particularly interested in, but since these abstractions were originally derived from reality, we need to read the technical definitions carefully to find aspects beyond the technical definition. For a more complete picture, the second step is to analyze contexts in authorization research to complete the criteria and identify the respective dimensions.

4.1.1. Context criteria in textbooks

Numerous similar definitions of the primary parts of authorization exist. Anderson (2008) describes *principals* (persons, processes, machines) that have access to *resources*. Gollmann (2011) agrees with this definition, and adds that "subject" is a technical representation of the principal and that the elementary access operations are "observe" and "alter." According to Benantar (2006), the principal is the system representation of a "user", a human being. According to Gollmann (2011), the *resource owner* "is in charge of setting security policies" (p. 73). We generalize this concept and define the resource owner to rather have significant interests in protecting the resource, without a necessary legal ownership or the ability to set permissions.

For the implementation of authorization, Gollmann (2011) distinguishes different *paradigms*: in *discretionary authorization*, the resource owner defines the policy, often in "identity-based access control" (IBAC) schemes. According to his definition, the opposite is *mandatory authorization* with the policy given by the system. The latter strictly separates policy authoring from the principals and thus offers a more restrictive approach, providing for higher *security needs* (Department of Defense, 1985). Often, Role-based Access Control (RBAC) is seen as a separate paradigm, since permissions are only indirectly assigned to principals through roles (Sandhu et al., 1996). Through its indirection, RBAC can handle contexts with higher *complexity* more efficiently (Sandhu and Samarati, 1994).

A further concept is that of a central point of *enforcement* of the authorization decision, often termed "reference monitor" (Department of Defense, 1985) or "enforcement point" (Moses, 2005). In his "security dimensions", Gollmann (2011) distinguishes the *scope* of authorization decisions through the placement of the controls, both on a "man–machine" scale (human, application, . . . , ker-

	Description	Identified dimensions
Contingency factors		
Resource owner	Entity with interests in protecting the resource (not necessarily the legal owner), e.g. an organization	Content, entity owner; individual, organizational owner
Security needs	The owner's security objectives and their criticality, e.g. from the risk of an unauthorized disclosure	Confidentiality, integrity, availability
Resource	Entity to be protected by the authorization measure, e.g. files, services, systems	Active asset, passive asset, container
Principal	Entity who requires the resource's availability and operate on or access it, e.g. employees, Web services	Human, non-human
Security expertise	Expertise and awareness that can be expected from potential human actors in the system, e.g. the ability to estimate consequences of a policy change	Resource owner, principal, policy maker, policy author
Complexity	The complexity of the context, affecting the measure's design, e.g. the number of resources	Quantity, distribution, heterogeneousness, dynamics
Design parameters		
Paradigm	The general approach to authorization	Discretionary, mandatory authorization
Scope	The scope that the authorization measure covers	Personal, inter-personal, organizational, inter-org.
Management model	The approach to how changes to the policy are made	Roles, centralization, formalization
Enforcement model	The socio-organizational or technical means that enforces the authorization restrictions	Socio-organizational, technical

Table 4.1.: Authorization context criteria

nel, hardware), and whether the controls are centralized or decentralized. More organizational is the mode of administration of the policy. Sandhu and Samarati (1994) discuss examples of *management models*, including centralized, hierarchical, cooperative, ownership, and decentralized. For instance, discretionary and decentralized models require *security expertise* on behalf of the owners (National Computer Science Center, 1987).

From these definitions and descriptions of authorization measures, we derive a taxonomy of authorization contexts, given in Table 4.1. The criteria are grouped into two areas: the *contingency factors* that govern the design of the authorization measure and the *design parameters* chosen for the measure. The two terms are borrowed from the criteria used by Mintzberg (1980) to describe organizational configurations, similarly multi-faceted artifacts as authorization measures. While the contingency factors should be more stable than the design of authorization measures, they are by no means set in stone. For example, removing data from systems can reduce security needs, security training programs can increase security expertise, and a more permissive policy can reduce the complexities when setting individual permissions.

The taxonomy is primarily concerned with the characteristics of the context. Accordingly, a detailed technical description of the authorization mechanism, such as the technical enforcement mechanism and the policy model, is out of scope. Indirectly, technical aspects are present as influencing factors for the complexity characteristic, though.

4.1.2. Contexts in authorization research

With the broad range of contexts in which authorization is employed, it is challenging to create a comprehensive overview of the research literature to validate the above-given criteria and examine

Database	Proceedings series	Examined	Rationale	Total	Applicable	
IEEE	–	25 most relevant of 3426	Broad overview of technical authorization papers	25	24	96%
IEEE	–	25 most cited of 3426	Important technical papers on authorization	25	14	56%
IEEE	POLICY (2002 – 2010)	25 most relevant of 109	Technical venue, close to authorization policies	25	25	100%
IEEE	ACSAC (2006 – 2009)	25 most relevant of 32	Recent publications at an applied security conference	25	18	72%
ACM	CCS (1993 – 2010)	25 most relevant of 253	Important, broad information security venue	25	23	92%
ACM	CHI (1994 – 2011)	20 most relevant of 27	Primary venue for human-factors, highly applied	20	8	40%
				145	112	77%

Table 4.2.: Sampling of authorization research publications

the breadth of the contexts. While research is not representative for the real-world problems, publications often focus on specific areas and problems to motivate their contributions. This approach poses a threat of missing contexts that the research community neglects. However, we can expect researchers to be particularly interested in focusing on previously-ignored contexts if these exhibit particular characteristics that are sufficiently different to motivate new research findings.

Since numerous publications relate to authorization (3,400 in the IEEE Xplore digital library, 3,800 for the ACM Digital Library), we take six samples from the two technical digital libraries, conducting searches for “authorization”, partly limited to specific proceedings series. The individual samplings are shown in Table 4.2 with the respective rationale, covering 145 publications in total. Publications are categorized as applicable if they cover authorization to a significant extent, which was the case for 77% of the studied publications. In the remaining cases, the publications primarily only touched authorization, for instance, as a motivation or future work. To arrive at a large proportion of applicable publications, five samplings were conducted with the sort order set to “by relevance”. For the sixth sampling, the “by citation count” sort order was applied to also include publications deemed as particularly important by the research community. As expected, this resulted in a lower proportion of applicability (56%). Even lower was the proportion for the very applied human-factors-focused conference CHI (40%), where authorization was primarily only mentioned in passing.

The applicable publications describe a broad range of application areas for authorization, ranging from Trusted Computing-based operating-system mechanisms to distributed grid systems and from discussions of purely theoretical authorization models to specific applications, such as health-care information systems. The publications mention the different context criteria to varying extents as shown in Table 4.3. While most at least mentioned the intended principals and the technical enforcement of the authorization mechanism (93% and 90% of applicable publications, respectively), it was rather uncommon to discuss the management (34%), complexity (42%), or resource owners (54%) of the measure. Also, resources were only in half of the cases more specifically defined than as abstract “objects.” This should not come as a surprise since organizational aspects of management, complexity, and resource owners are not necessary to motivate many technical aspects of authorization contexts for technology-focused publications. Still, the publications provide a broad range of contexts for analysis of the criteria in the following. To support the discussion of the individual criteria in the following, Tables 4.4 and 4.5 respectively show the contingency factors and design parameters of exemplary contexts derived from the publications.

Resource owner	60	54%
Resource	53	48%
Principal	103	93%
Complexity	47	42%
Authorization scope	66	59%
Management	38	34%
Enforcement	100	90%

Table 4.3.: Mentions of selected criteria, quantity and as proportion of applicable publications

Example	Resource owner	Security needs	Resource	Principal	Expertise	Complexity, dynamics
Rented office building	Organization (entity, organizational)	o/++ (depends on values)	Offices (container)	Employees (human)	+ (concrete measure)	Offices, employees; turnover, changes
Private housing	Tenants (content, organizational)	–/o (depends on values)	Home valuables (passive)	Tenants (human)	+ (concrete measure)	Tenants
File system	File creator (content, individual)	Data sensitivity, system criticality	Files (container)	System users (both)		Files, users, ACL entries
Smartphone applications	Device owner (entity, individual)	+ (banking, payment, privacy, service costs)	Personal data, calls (passive), messaging (active)	Installed applications (non-human)	–	Applications, permissions
Shared folders	Organization (entity, organizational)	Business risks	Folders (container)	Employees (human)	+ (trained administrators)	Employees, folders; organizational dynamics
Shared folders (decentralized)	Organization (entity, organizational)	Business risks	Folders (container)	Employees (human)	–	Employees, folders; organizational dynamics
Scientific Grid	Individual system owner (entity, organizational)	Experiments, computations	Computational resources (active)	Employees in virtual organization (human)	+ (trained administrators)	Data entities, organizations, principals
Web site privacy (visitor)	Web site visitors (content, individual)	Collected data (personal, payment data)	Personal information, activity log, payment data (passive)	Operator and partner employees (human)	– (visitor), + (policy author)	Data types, partners
Web site privacy (operator)	Web site operator (entity, organizational)	Collected data (personal, payment data)	Personal information, activity log, payment data (passive)	Operator and partner employees (human)	+	Data types, partners
User-generated content	Web site user (content, individual)	Individual privacy needs	Photos, messages, activities, connections (passive)	Profile visitors (human)	– (difficult to foresee consequences)	Data items, contacts

Table 4.4.: Contingency factors of exemplary authorization contexts

Examples	Paradigm	Scope	Management	Enforcement
Rented office building	Mandatory	Organizational	Centralized; CSO (maker), admin (author)	Property owner, system manufacturer (Physical)
Private housing	Discretionary	Inter-personal	Decentralized; tenant (maker/author)	Property owner, lock producer (Physical)
File system	Discretionary	Inter-personal	Decentralized; resource owner (maker/author)	Operating system (OS)
Smartphone applications	Mandatory	Personal	Centralized, developer (author), device owner (grants policy)	Smartphone OS manufacturer (OS)
Shared folders	Mandatory	Organizational	Centralized; CISO (maker), admin (author)	System manufacturer (System)
Shared folders (decentralized)	Discretionary	Organizational	Decentralized; individual employee (maker/author)	System manufacturer (System)
Scientific Grid	Discretionary	Inter-organizational	Decentralized; individual organization (maker/author)	Grid system operator (Distributed system)
Web site privacy (visitor)	Discretionary	Inter-organizational	Visitor (accepts policy), operator (author)	Web site operator (System)
Web site privacy (operator)	Mandatory	Inter-organizational	Centralized; operator (maker/author)	System manufacturer (System)
User-generated content	Discretionary	Inter-personal	Decentralized; user (maker/author)	Web site operator (System)

Table 4.5.: Design parameters of exemplary authorization contexts

4.1.3. Contingency factors

Contingency factors define the parameters and premises for an authorization measure and reflect the specific environment.

Resource ownership

Gollmann (2011) defines the owner as the one "who is in charge of setting security policies." While in many contexts the person taking the decision – the policy maker[3] – has significant interests in the protection of the resource (resource owner), these may be distinct: In the decentralized shared-folders case (cf. Table 4.4), we take the perspective of the organization as the resource owner with interests in protecting the folders, but the individual employee makes the policy decisions. Accordingly, we define the resource owner as an entity in the authorization context whose security needs are protected through the authorization measure. In the examined publications, owners can be grouped along two dimensions. One dimension is their relation to the resource:

- *Content owners* are actors who are the owner of the content to be protected, for instance, through explicitly or implicitly originally creating content or having the legal or de-facto ownership. Examples from Table 4.4 are Web site users creating content, such as messages in forums, and Web site visitors leaving log trails.

- *Entity owners* control the device or system that is to be protected due to their legal or de-facto ownership. This type of ownership can be observed in the examples for smartphone device owners, owners of individual systems in Grids, or Web site operators. Further entity owners mentioned in the publications are ISPs, network operators, and router owners.

[3]We further discuss the role of the policy maker as part of the management model.

The second dimension refers to the type of owner:

- *Individual owners* are people who protect data for their individual good, for example, smartphone device owners.
- *Organizational owners* are organizational entities that group interests in the protection of resources, typically entire organizations or departments in enterprises. In contrast to the individual owners, organizational owners need to establish power structures to enforce the resource's protection for the organization's good.

Often, several stakeholders have interests in the protection of resources. Take, for example, the two Web-site privacy cases in Table 4.4: One perspective is that of the visitor who has privacy concerns; simultaneously, the Web-site operator also acts as an owner and ideally protects the collected visitor data for its own interests and to comply with regulatory obligations. Thus, we can identify two distinct authorization contexts in this case, depending on the chosen perspective.

Security needs and the resources to protect

A second characterizing criteria of the authorization context are the security needs of the resource owner and, particularly, the resources that the resource owner needs to protect (Gollmann, 2011). In the authorization examples from research, the publications mention in half of the cases more specific resources than general "objects". The resources in the examined publications fall in the following categories:

- *Passive assets:* Content in information systems to be protected from unwanted disclosure (confidentiality) or modification (integrity), including personal information and photos. At the same time, they need to be accessible by those allowed to when needed (availability). In physical contexts, passive assets can also be physical assets, such as valuables in a flat.
- *Active assets:* Active functionality offered to principals. For instance, active assets may enable SMS messaging in smartphones or offer computational services in Grids. For active assets, the protection focuses on the availability of the asset (accepting and completing tasks) and its reliability (working as expected, including the integrity of the asset).
- *Containers:* Not necessarily of value themselves but guard active and passive assets. Typical examples of authorization containers are offices (having valuable assets inside) or folders (with files inside), but also network infrastructure and databases. Choosing containers as resource in an authorization context limits the granularity of the authorization, since decisions then apply to each container in its entirety. Here, the protection needs of the confidentiality, availability, and integrity concerns the assets inside the container.

Similarly to the resource owner, the resource to be protected depends on the perspective on the authorization system. Identifying a container as resource to be protected often is a simplification and operationalization of the authorization requirements, closer to the authorization architecture and implementation. This can be observed in Table 4.4 for the rented office building and private housing examples. In the office case, the offices are declared to be protected, which actually are the containers of valuable assets, in the private case, the resources are the valuables themselves.

Apart from the type of resource, explicit formulation of security needs and their criticality is sparse in the studied authorization literature. We find considerations of security needs only for very specific authorization contexts, such as health-care information systems, or very applied research on the formulation of policies.

Principals

Principals are those entities in authorization contexts that request to interact with the resource and have (or lack) the authority to do so (Gollmann, 2011). Principals can be "users or machines", and are often structured into groups, roles, or similar concepts (Abadi et al., 1993). In the examined publications, three fundamental categories of principals emerged:

- *Human principals:* Direct representations of people interacting with an information system, for example, as an employee in an organization who is represented as a user in an organizational information system. Human actors can be threats to the resources for both, intentional (malicious external attacker, insider) or unintentional (accidental, human errors) reasons.
- *Non-human principals:* Programmed agents that conduct tasks. Typical examples are applications installed on smartphones that interact with smartphone resources on behalf of the device user or Web services cooperating to complete a task. Further non-human actors in the authorization literature include devices (e.g. in networks) and system components. Here, the intentional and unintentional threats are caused by the assigned tasks, programming, or modification, accidentally or maliciously threatening the resource.
- *Undetermined:* Varied degree of indirection between human actor and principal. For example, in operating systems, system users can directly represent a human user or, more indirectly, processes.

Security expertise

The examined publications offer little detail on the security expertise of the relevant stakeholders, including resource owners, policy makers, and policy authors. This is probably due to the context-specificness of this aspect. However, comprehending authorization measures is necessary for their success (Brostoff et al., 2005), and we can expect a broad range of individuals – from "passive consumers" to "domain designers" (Fischer, 1999). We thus can expect that the security expertise of stakeholders – relative to how they interact with the system – will influence the authorization measure design. For example, with trained security administrators as policy authors, it is more likely that a complex authorization system will be successful than for average smartphone users.

Complexity and dynamics

Complexity characteristics govern the way in that authorization can be implemented effectively and efficiently. For instance, it will be inefficient to set the permissions individually for each principal in contexts with numerous principals. As shown in Table 4.6, the studied publications mentioned four main *dimensions* of complexity (quantity, distribution, heterogeneousness, dynamics), each applying to different *artifacts* in the authorization context. The mentioned dimensions of complexity are:

- *Quantity:* Primarily increases the complexity by requiring a higher number of management activities, such as setting permissions, and making it more difficult for policy makers and implementers to have an overview of the entire policy.
- *Distribution:* Defines the degree of distribution of artifacts in the authorization context. Authorization contexts can be technically distributed, for instance, with distributed systems that each require authorization policies, but also on an organizational level, with delegated, local policy management. A higher distribution may, for example, lead to increased efforts of policy management if the policy must be adapted at different points.

	Quantity	Distribution	Heterogeneousness	Dynamics
Security needs	Requirements	Distributed security needs	Types of requirements	Changing processes, functionality; dynamic decisions (delegation)
Resources	Applications, services	–	Types of artifacts	Changing services
Principals	Users, components	Distributed components	–	Changing users, applications
Authorization policy	Permission entries (granularity)	Conflicting policies	Types of roles and restrictions	Policy adaption
Enforcement	Applications, services, enforcement points	Distributed systems and services	Types of systems, authorization models	Dynamic system configuration, changing systems

Table 4.6.: Complexity dimensions and respective examples from examined publications

- *Heterogeneousness:* Extends the complexity from the degree of distribution in that the distributed parts of the measure differ. This may, again, both apply to the technical implementation (different types of authorization models in the systems) or organizational aspects (different types of authorization requirements). One result of heterogeneous authorization contexts is the necessity to generalize the authorization models or manage separate policies.
- *Dynamics:* Refers to the changes that affect an authorization context and need to be reflected in policy changes. Changes can occur for the resources (additional services to protect), security needs (changing environments, new threats), or from changes in the enforcing systems. High dynamics will increase the number of management activities and, together with a high quantity, may render individual authorization approaches inefficient.

The dimensions of complexities apply to criteria of the authorization context (security needs, resources, principals) and of the measure design (authorization policy, enforcement). The factors of the context and the measure design are interrelated regarding the complexity. For instance, the number of individual resources depends both on the context (how many entities need to be protected) and the measure design (how are the resources operationalized in the authorization measure).

4.1.4. Design parameters

While the contingency factors draw the field of play for an authorization measure, the design parameters represent how the measure is actually implemented for the context.

Authorization scope

The authorization scope represents the extent that an individual authorization context covers. In simple cases, the scope is limited to one system on which only one individual is active (cf. the smartphone example in Table 4.5). More complex contexts may govern multiple individuals or organizations. The following types of authorization scopes are present in the examined literature:

- *Personal:* Limits the applicability of authorization to the environment of individuals. Among others, this is the case for Personal Area Networks (PAN) that connect an individual's devices, and for application permissions on smartphones.

- *Inter-personal:* Includes multiple individuals in one context, who, for example, exchange files in peer-to-peer file sharing or work on a shared system and mutually or unilaterally assign permissions.

- *Organizational:* Covers groups of people who cooperate for a shared goal in one authorization context. This is a very common scope, named in half of the publications that mention an authorization scope, and encompasses, for instance, IT systems in organizations, such as shared folders between employees. In contrast to the inter-personal scope, the stakeholders in the organizational scope are governed by a shared organizational body.

- *Inter-organizational:* When organizations cooperate outside of a common organizational body for the cooperation, the authorization scope is inter-organizational. Typical examples are customer organizations that access their order status through their supplier's systems or organizations that cooperate as Virtual Organizations (VO) in Grid systems.

The authorization scope affects authorization systems primarily in the way authorization and policy decisions are organized. While only an individual is involved in authorization decisions and their consequences in personal scopes, in the other cases, restrictions affect other individuals or organizations. Moreover, in the personal and inter-personal case, no coordination between stakeholders is necessary to arrive at the policy decision of whether a permission is granted. Conversely, organizations often require the interaction between several stakeholders to arrive at a policy decision. We analyze the relationship between the authorization scope and the resource owner for the authorization paradigm below.

Enforcement

There are two perspectives on the enforcement as part of the design of the authorization measure. On a general level, we come back to the differentiation between legitimate and technical authorization:

- *Socio-organizational means:* An integral part of societies and organizations are power structures that rely on means, such as ethics, social norms, laws, and deterrence to enforce authorization (Clegg, 1989; Lessig, 1998). This includes the offloading of the responsibility to enforce authorization to another actor, such as an external service provider, as custodian.

- *Technical mechanism:* In information systems, technical mechanisms are often integrated to enforce the protection. Mechanisms typically rely on an authorization model to formalize the security needs in authorization policies that are then enforced by controls in the systems.

The technical mechanisms were the focus of the examined literature. While not the primary interest for the purpose of the taxonomy in this thesis, it is nevertheless interesting to consider the occurrences of enforcement architectures. A widespread technical concept of enforcement is the *reference monitor* that "mediates all accesses to objects by subjects" (Department of Defense, 1985). Gollmann (2011) notes two characteristics for the placement of reference monitors:

- *Platform:* hardware, hypervisor/kernel, OS, service layer (DB, middleware, Web browser), applications,

- *Placement relative to the program:* external (kernel), as interpreter (surrounds program), or in-line (part of program).

The examined publications mention six distinct types of enforcement: physical security (e.g. keys to offices), operating system (file or application permissions), middleware (databases), individual information systems (single-system applications), distributed information systems (systems of multiple components on multiple systems), and network (protecting the access to a network). The type of authorization enforcement impacts several of the technical complexity aspects, such as the applicable authorization models, the granularity of permissions, and the provisioning of policies to the individual systems.

Management model

As part of the organizational authorization architecture, organizational measures to manage the authorization policy need to be established. The number of stakeholders that must interact depends on the type of the authorization scope. Two primary roles can be identified for the management of authorization policies (Bauer et al., 2009). The first is the role of the *policy maker* who takes decisions on policy changes and thereby grants or restricts permissions. Secondly, the *policy author* formalizes the decision as an authorization policy, for instance, by setting a permission in an access control list.

In many of the studied publications, the resource owner holds both roles, for example, when a file owner decides to grant permissions to another system user and also changes the file's permission setting (cf. examples in Table 4.5). In organizational contexts, the separation of the policy maker from the author is more common since the decisions are often taken by higher level managers and implemented by system administrators (cf. shared folder example). In these contexts, we may also find that the policy maker is separate from the resource owner, when the organization as the resource owner has passed on the authority to make policy decisions to individual employees, as in the decentralized shared-folder example.

An opposite management approach for the making and authoring of the policy can be observed for the smartphone application authorization and the visitor perspective of the Web-site privacy examples. In these cases, the application developer or the Web site operator, respectively, authors a policy that the device user or visitor accepts as the act of decision-making.

In addition to the differentiation of making and authoring policies, the management can also be categorized according to the degrees of *centralization* and *formalization*. According to the continuum of centralization (Malone, 2004), policy decisions can either be taken hierarchically and centrally or delegated to local decision makers. An example for centralized management is the shared-folder case, where the decisions on access to individual folders is taken centrally for the organization. In contrast, in the Grid case, decentralized management is shown with each member organization deciding on permission for their systems. The examined publications also mention shared models, where part of the decisions are taken centrally and more fine-grained, for example, delegations, occur locally. Generally, the degree of centralization can be measured from the proportion of organizational coverage of individual policy makers with respect to the entire authorization scope.

The formalization similarly depends on the structures surrounding the measure. While not mentioned as often as the centralization of the management model, it is nevertheless a significant aspect of the measure design and refers to the formality of requesting changes and the associated processes.

Authorization paradigms: Discretionary and mandatory authorization

When examining the authorization contexts present in literature, particularly regarding the role of the policy maker and its interrelation with the authorization scope as well as the centralization of the management model, two distinct authorization paradigms emerge as key characteristic of the authorization context. In the first case, the policy makers in the system take *discretionary* decisions

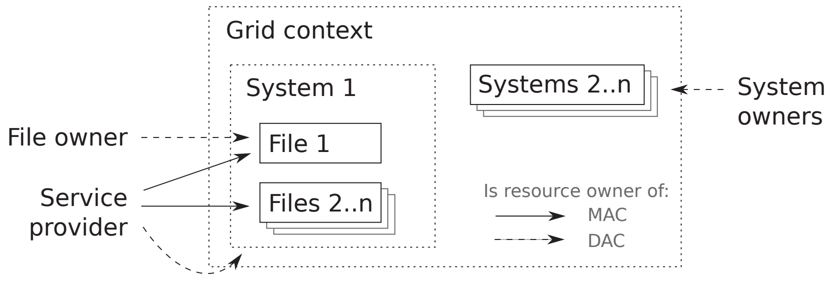


Figure 4.1.: Interrelation of discretionary and mandatory authorization contexts

on parts of the overall authorization policy. One example of discretionary authorization can be found in the examples in Table 4.5 for decentralized shared folders, where employees set the permissions as policy makers on individual resources. In contrast, *mandatory* authorization contexts have policy makers that define overall policies for the entire authorization scope. This is the case for centralized shared-folder example, where the CISO sets overarching policies. The primary difference is, thus, whether the policy maker governs the entire authorization scope or only subsets of the scope.

Both terms, “discretionary” and “mandatory”, have been historically defined in authorization to refer to categories of authorization models. However, the model that is understood as Discretionary Access Control (DAC, National Computer Science Center, 1987) is more appropriately referred to as IBAC (Gollmann, 2011). In Mandatory Access Control (MAC, Department of Defense, 1985), the policy is supposed to be set by the system (Gollmann, 2011). However, this assumption is imprecise, since even then, an actor decides on the system-enforced policy. Thus, to distinguish the authorization paradigm from the model, we will refer to mandatory and discretionary authorization instead of “access control”.

The two authorization paradigms are reflected in the policy management model, either delegated to local decision-makers, such as individual employees, or centralized in hierarchical organization. As noted for the management model above, authorization architectures occur on a continuum of centralization. For example, centralized procedures for granular permissions can be combined with local management of fine-grained permissions. Generally, for any discretionary context, one may define a mandatory context by taking on a higher-level perspective and establishing additional mandatory restrictions that, for instance, impose upper limits on the ones assigned decentrally. In the example depicted in Figure 4.1, file owners (policy makers in a discretionary context) govern the permissions of visitors to the content of their Web sites. At the same time, the owner of the shared-hosting service (“service provider”) represents the policy maker for the mandatory context of the system and limits the system resources generally available to the file owners. Considering the authorization context of a Grid of services, in which multiple system owners contribute services, would see the service provider as a discretionary policy maker with respect to restricting the access to her service as part of the Grid.

4.1.5. A taxonomy of authorization contexts

The above-discussed criteria together with the respective types and dimensions that we identified in authorization research (cf. Table 4.1) represent a taxonomy of authorization contexts. The contingency factors form the parameters for the authorization measure to fit into, including who is interested in protecting resources and to whom the resources should be made available. The second part, the design parameters, describe how the measure is actually designed, organizationally through the scope and management, and technically through the enforcement of authorization. A fundamental design decision that interrelates with several of the other criteria is the authorization paradigm to be applied, whether to allow individual policy-makers in the authorization system to decide on policy changes (discretionary) or whether this authority is centralized for the entire authorization scope (mandatory). Irrespective of the paradigm, the authorization scope, the security needs of the resource owner, and the authority of the policy maker relate organizationally as follows:

$$\textit{Authorization scope} \supseteq \textit{Resource owner needs} \supseteq \textit{Policy maker authority}$$

In mandatory authorization, the policy-maker authority will be congruent with the authorization scope. In discretionary authorization, the policy-maker authority will only cover part of the scope.

We can apply the taxonomy to define the scopes of application of individual approaches that address usability problems, and, in the context of this dissertation, specify the scope of the thesis.

4.2. Characteristics of organizational authorization contexts

> "An 'organization' is a system of purposive activity of a specified kind. A 'corporate organization' is an associative social relationship characterized by an administrative staff devoted to such continuous purposive activity." (Weber, 1947)

For the purpose of this thesis, Max Weber's "Corporate organization" should suffice as the definition of organizations. When discussing organizational authorization contexts, the primary aspects are the posture of the organization itself and of its information systems. Regarding the former, the *Contingency Theory* states that contingency factors define the most suitable configuration of an organization (Mintzberg, 1980). Similarly, authorization measures in organizational information systems need to adapt to the organizational configuration for effectiveness and efficiency.

To situate the characteristics of authorization contexts in those of organizations, we need to refer to theories of organizations. Of the several available theories, the *Structural Configurations* (Mintzberg, 1980) lend themselves well: Mintzberg (1980) describes organizations through a typology of five "pure" configurations, primarily distinguished through the organizational part (Strategic Apex, Middle Line, Operating Core, Technostructure, support staff) that dominates the organization[4]. For example, the Simple Structure is characterized by the "pull" of the Strategic Apex for centralization, and the absence of an "elaborated" organization. In the Machine Bureaucracy, the domination of the Technostructure leads to high standardization of work. The Structural Configurations are particularly useful since their definitions cover a broad range of characteristics at a rich level of detail (McPhee and Poole, 2000), including coordination mechanisms, design parameters, and contingency factors.

We analytically relate the organizational contingency factors and design parameters to those from the taxonomy of authorization contexts from the previous section (cf. Table 4.7). For example, the authorization complexity derives, among others, from the factors "Dynamics" and "Job specialization." The relation in Table 4.8 provides a hypothesis of the breadth and typical characteristics of organizational authorization, further guiding the research in this thesis. Moreover, we can employ the characteristics for a well-founded limitation of the scope of this thesis and evaluate the authorization taxonomy through its application.

[4]Mintzberg later modified his configurations (Mintzberg, 1989), adding two configurations and renaming others. The added "ideological" configuration bases its coordination on the corporate culture, the political configuration on each individuals' political power within the organization. For the authorization contexts, both will be similar to Adhocracies, even though with more pronounced dynamics. For brevity, we will thus stay with the more compact earlier list of five configurations.

	Mintzberg's design parameters	Mintzberg's contingency factors
Contingency factors		
Resource owner	Decentralization, Unit grouping	Power focus
Security needs		
Resource		Technical system
Principal		
Security expertise	Training, Indoctrination, Decentralization, Job specialization	
Complexity	Decentralization, Bureaucratic/organic, Unit size and grouping, Job specialization, Behavioral formalization	Dynamics, Complexity, Size, Technical system
Design parameters		
Authorization paradigm	Decentralization	
Authorization scope	Unit size and grouping, Decentralization, Job specialization	
Management model	Decentralization, Formalization of behavior, Bureaucratic/organic, Coordination mechanism	Power focus
Enforcement	Coordination mechanism	

Table 4.7.: Relation of characteristics of organizations to authorization contexts

	Simple Structure	Machine Bureaucracy	Professional Bureaucracy	Divisionalized Form	Adhocracy
Contingency factors					
Resource owner	Entire organization	Mostly entire organization	Individual decentralized units	Divisions	Selective groups
Security needs	Diverse	Diverse	Diverse	Diverse	Diverse
Resource	Unstructured	Structured	Structured	Structured	Unstructured
Principal	Diverse	Diverse	Diverse	Diverse	Diverse
Security expertise	Low training, high centralization	Low training, high vertical centralization	High training, low centralization	Medium training, limited decentralization	High training, selective decentralization
Complexity	High dynamics	High quantity, high heterogeneousness	High distribution, high heterogeneousness, medium dynamics	High quantity, selective distribution, medium heterogeneousness	Selective distribution, high heterogeneousness, high dynamics
Design parameters					
Authorization paradigm	Mandatory	Mandatory	Discretionary	Mandatory	Discretionary
Authorization scope	Large	Large	Medium	Divisions	Selective smaller scopes
Management model	Centralized	Centralized	Decentralized	Centralized (for devisions)	Decentralized
–Policy maker location	Strategic Apex	Technostructure	Operating Core	Middle management	Supporting staff
–Roles	Separated if dedicated support staff	Separated	Combined	Separated	Combined
–Formalization	Informal	Formal	Informal	Formal	Informal
Enforcement	More socio-organizational	More technical	More technical	More technical	More socio-organizational

Table 4.8.: Characteristics of authorization contexts in organizations

The second primary aspect of organizational authorization contexts are the information systems. In this respect, a context depends, among others, on the criticality of the contained data and the served business processes, and the implemented authorization model (ISO/IEC 27002:2005, 2005). The use of organizational information systems was, for instance, classified by Vaidya and Seetharaman (2005) according to the scope and sophistication of use. Other dimensions include the system flexibility (Knoll and Jarvenpaa, 1994).

4.2.1. Organizational contingency factors

Organizational resource owners

In organizations, the entity that is interested in protecting resources in information systems is a unit of the organization, such as the entire organization or a department. The resource owner is thus of organizational type as entity and/or content owner. Applying the organizational design parameters of Mintzberg (1980), a context's resource owner depends primarily on the organization's unit grouping and size, and its vertical and horizontal decentralization. In Simple Structures the resource owner will likely be the entire organization, while organizations of the Divisionalized Form will tend to have separate owners per division and Adhocracies will have individual groups in the organization as owners.

While external parties may be interested in the protection of resources – for example, customers trusting the organization with protecting the confidentiality of their data (organization as entity owner) – we often can operationalize external resource owners by assuming additional legal constraints, such as service-level agreements, or similar motivations to internalize the third-party security needs (externalities).

Security needs from risk tolerance and regulatory obligations

The security needs in organizations cover the full range of criticality. The needs depend on the organizational context and on the processes served from and data contained in the specific system. The general security needs are governed by the organization's tolerance for risks, the criticality of information confidentiality and availability for its goals, and the obligations for regulatory compliance (ISO/IEC 27002:2005, 2005; Pallas, 2009). Regulations may be market-specific, such as the Sarbanes–Oxley Act and HIPAA, or depend on the type of data, such as privacy regulations (Hermann, 2007).

Organizational resources

We can expect to find the full range of resources described for general authorization contexts in organizations. Active assets can, for example, involve data collection functionality in applications, passive assets can be present as documents, and the respective containers can cover entire applications or folders. An important factor is the structuredness of information in organizational decision-making (Gorry and Morton, 1989). In organizations with primarily knowledge-based decisions, the information will have less structure than in those with highly formalized or automated decisions. Regarding the Structural Configurations, the technical-system regulation and the structuredness of the organization are the primary influencing factors, making it likely that we will find rather unstructured resources in Simple Structures and Adhocracies (Mintzberg, 1980).

Individual members of organizations as principals

Of the types of principals that need to access the protected resources, we will limit the organizational authorization contexts to contexts with human principals. The rationale is that we can expect the organizational dynamics, processes, and decision-making to be particularly pronounced for human principals. In contrast, decisions on the authorization of non-human principals, such as components, will primarily be limited to system changes, such as the integration of components. Contexts with non-human principals will thus generally lack the inherent organizational aspects of organizational authorization and are sufficiently different to warrant a separate consideration.

For human principals in organizational authorization contexts, the purpose of the resources determines the principals who require their availability (Gorry and Morton, 1989). For management information systems, Gorry and Morton (1989) distinguish the purposes of operational control, management control, and strategic planning. Mintzberg (1980) defines organizational parts from which principals may primarily originate, including the strategic apex, the middle line, and the operating core.

Security expertise of the individuals

The security expertise of policy makers, policy authors, and principals in the organizational authorization contexts can be expected to vary widely and depends on several factors, including the organizational security-culture. Schlienger and Teufel (2003) structure security culture in three layers of "artifacts", "values", and "basic beliefs". The security culture typically consists of awareness, responsibility, ethics, and security management (OECD, 2002). These factors are influenced by the organizational design parameters of training and indoctrination (Mintzberg, 1980). For policy making and authoring, the security expertise of the responsible stakeholders will also be influenced by the degree of centralization, job specialization, and the behavior formalization (Mintzberg, 1980).

Complexity of the organization and its information systems

The complexity of authorization contexts depends both on the complexity of the organization and of the organizational information systems. According to the organizational complexity, the context complexity will vary along the four dimensions:

- *Quantity:* Influenced primarily by the contingency factor of organizational size, such as number of employees, and the design parameter of unit sizes, which affects the number of employees and resources to represent in authorization for individual systems (Mintzberg, 1980). A high organizational quantity can be expected for the Structural Configuration of the Technostructure-dominated Machine Bureaucracy and the middle management-dominated Divisionalized Form.

- *Distribution:* Depends on the design parameters of unit grouping (e.g. number of different departments with separate systems) and the vertical decentralization, that is, for example, the autonomy of departments in setting authorization policies (Mintzberg, 1980).

- *Heterogeneousness:* In addition to the distribution factors, which may also lead to heterogeneous systems, Mintzberg (1980) names the design parameters of job specialization, which encompasses the division of labor and thus the differences in system and authorization requirements within organizations.

- *Dynamics:* Truex et al. (1999) argue that organizations are increasingly "emergent", that is, in a continuous state of change to culture, meaning, relationships, and decision processes. Authorization dynamics will be primarily affected by changes to task assignments and structure.

Mintzberg (1980) already names the contingency factor of dynamic environments as reason for adaption and, as design parameters, a low behavior formalization with unstructured work and organic structures. For the Simple Structure, based on direct supervision, and the Adhocracy with a high influence by support staff, the dynamics should be particularly pronounced.

Apart from the influence of the organizational complexity, the information systems to be protected by authorization also contribute to authorization complexity. The contingency factors of technical system regulation and complexity affect the authorization complexity (Mintzberg, 1980). The heterogeneousness dimension of authorization complexity is affected by the sophistication of use from the differences in structuredness, as observed, for example, in the contrast between simple file sharing and complex decision-making based on knowledge data (Vaidya and Seetharaman, 2005).

Knoll and Jarvenpaa (1994) propose three flexibility dimensions of IT systems, of which the "Flexibility in Functionality" to operate well in different environments impacts the heterogeneousness of authorization. The other two dimensions of Knoll and Jarvenpaa (1994), "Flexibility in Use" (react to changes in goals) and in "Modification" (adaptable to process changes), will affect the dynamics of authorization. More generally, Truex et al. (1999) states that in emergent organizations, information systems follow suit or even drive the change. Traditional approaches of lengthy analysis phases, abstract requirements, and complete specifications are replaced by "always analysis", requirement negotiation, incomplete specifications, continuous redevelopment, and adaptability of information system development.

4.2.2. Design parameters of authorization in organizations

Authorization scope

In organizations, multiple individuals work together for a purpose. Since the individuals need to interact for authorization, the type of authorization scope will thus be *organizational* and the scope will cover the entire organization or organizational units. Lacking the interaction, *personal* and *interpersonal* types of authorization scopes will be ruled out by definition. Organizational contexts will always require a minimum of coordination between stakeholders to arrive at an authorization policy, even if this is only an implicit delegation of the authority to a policy maker. Mintzberg (1980) defined a number of design parameters of organizations that affect the choice of the scope for authorization, including the job specialization, the unit grouping, and the vertical and horizontal decentralization. Larger scopes will accordingly rather be found in configurations of Simple Structure and Machine Bureaucracy.

To reduce the complexity, we exclude *inter-organizational* scopes, given that these introduce additional problems of identity management and trust, which are not core authorization problems. With the trend towards market structures in organizations (Pallas, 2009), organizational authorization problems will resemble inter-organizational scopes due to their loose coupling, but the organization will represent a central point of coordination. Nevertheless, external stakeholders, such as customers, can act as principals in the organizational authorization contexts, when their access is considered under the authority of the original organization.

The authorization paradigm and management models

The characteristics of an organization will determine to a large extent the authorization paradigm, and the formalization and centralization of the authorization management. Pallas (2009) economically analyzed the formalization and centralization of information security in organizations and describes how organizations can rely on either hierarchical (centralized) or market (delegation) forms of coordination for information security. Effects such as opportunism and information asymmetries will cause different coordination and motivation costs. According to Malone's *Decentralization Continuum*, organizations can position between centralized hierarchies (military), loose hierarchies (research), democracies, and markets (Malone, 2004, p. 6). The degree of centralization also affects the authorization paradigm, more centralized organizations tending to mandatory, decentralized to discretionary authorization.[5]

One of the contingency factors defined by Mintzberg (1980) that influence the centralization is the need for power in specific parts of the organization, such as by the chief executive. Of the design parameters, the vertical and horizontal decentralization has the most direct impact on centralization (Mintzberg, 1980). Since authorization management incurs higher overhead when centralized, increased behavior and work formalization allows for higher centralization of policy management. More bureaucratic organizations will typically exhibit less dynamic structures and thus reduce the necessity of policy changes, making more centralized schemes less of a burden. Similarly, the primary coordination mechanisms of the organization can influence the policy management centralization, with Mutual Adjustments favoring more decentralized approaches and the Standardization of Work Processes more centralized ones. These effects are also reflected in the Structural Configurations, with more centralized management to be expected when the Strategic Apex (Simple Structure), middle management (Divisionalized Form), or Technostructure (Machine Bureaucracy) dominates and less for the domination by the Operating Core in Professional Bureaucracies (Mintzberg, 1980).

The roles of the policy maker and author in organizational authorization contexts depend on the centralization of the management, with higher centralization resulting in the policy maker to be found higher in the organizational hierarchy. Considering again the Structural Configurations, we most likely find the policy maker in the Strategic Apex for Simple Structures, in the middle management for Divisionalized Forms, in the Technostructure for Machine Bureaucracies, and in the Operating Core for Professional Bureaucracies. In Adhocracies, supporting staff that should be responsible primarily for the policy authoring may in practice also take the policy-maker role. For more decentralized cases, we will more often find the maker and author role combined in one person, while more centralization will favor a vertical specialization of these roles, requiring only the effort of decision-making and not the technical expertise of authoring from the policy maker.

The formalization of policy management depends on similar parameters of the organization. Organizations with higher formalization of behavior, such as Technostructure-dominated Machine Bureaucracies, will tend to also formalize the authorization management. Moreover, higher degrees of centralization and the separation of policy maker and author roles will also require more formalized approaches to coordinate the changes of policies.

Enforcement

As noted in the introduction of this chapter, the technical authorization is only one part of the overall approach to authorization, and thus to power and control in organizations. To consider the technical together with the legitimate authorization, we need to examine several layers of social and technical controls. Different perspectives on authority and control are given in Table 4.9. From the sociological perspective, the Circuit of Power (Clegg, 1989) is a useful framework that distinguishes *episodic power relations* (e.g. direct command) from those of *social integration* (e.g. norms) and *system integration* (e.g. technological measures and formal rules). Overlapping, but from a legal perspective, Lessig (1998) introduces four "modalities" that regulate the behavior of individuals in his framework:

[5]It is interesting to observe how technology and information systems enabled organizations to be decentralized and agile in the first place, but the security mechanisms introduced to protect the very same systems best suited centralized and bureaucratic structures.

	Clegg (1989)	Lessig (1998)	Pallas (2009)	Mintzberg (1980)
Discipline	Sociology	Jurisprudence	Org. inf. security	Organization design
Framework	Frameworks of Power	Modalities	Meta-measures	Coord. mechanisms
Enforcement	Episodic power relation	–	–	Direct Supervision
	Social integration	Norms	Informal rules	Mutual Adjustment
	System integration	Laws	Formal rules	Standardization of
		Market, architecture	Architectural means	work, skills, outputs

Table 4.9.: Enforcement of authority in organizations from different perspectives

laws, norms, market, and architecture. Laws are enforced centrally by the state, norms by the social communities, markets limit individuals through monetary constraints, and architecture through "features of the world". We find similar categories by Pallas (2009), defining informal and formal rules, and architectural means as "meta-measures" to enforce organizational information security.

Enforcement approaches are combined in organizations, reflecting the organizational context. Mintzberg (1980) defines primary coordination mechanisms for his Structural Configurations that may give indications of which form of authorization enforcement will dominate in an organization. Simple Structures, for example, use Direct Supervision, while control in Adhocracies is based on Mutual Adjustment, tending to Clegg's episodic power relations and social integration, respectively. Other configurations will more broadly rely on architectural means, that is, technical mechanisms.

The technical aspect of authorization measures occur on various levels in organizational information systems. The primary level will reflect the information security era (Pallas, 2009): *Isolated* systems were primarily protected physically, *mainframes* through technical approaches on the centralized system, and *distributed* systems through managerial means to achieve enforcement on distributed applications.

4.2.3. Organizational authorization as the scope of this thesis

The examination of the breadth of organizational authorization contexts allowed us to predict to a certain extent what characteristics to expect in authorization measures of organizations. A summary of typical contexts in Structural Configurations (Mintzberg, 1980) is given in Table 4.8. While the security needs and types of principals are diverse for organizations, we can derive characteristics for other criteria, particularly the resource owner, the complexity, and the management model.

Focusing on organizational authorization is useful to reduce the effort of vertically integrating the research in this dissertation to cover a broad range of problems surrounding and perspectives on authorization. More specifically, it allows us to limit the scope to human principals and organizational authorization scopes. This also allows us to emphasize the particularly challenging problems of authorization. In particular, we find a broad range of socio-technical perspectives in organizational contexts: the technical perspective of developing and integrating information systems, the organizational perspective of managing restrictions, and the socio-organizational perspective of interactions between the individuals. We will analyze how well the insights from organizational authorization can be transfered to further contexts, such as authorization for personal devices, as part of the concluding discussions of this thesis. Otherwise, we will primarily focus on organizational authorization for the remainder of this dissertation.

4.3. Personas for Authorization Problems

The authorization context taxonomy refers to a number of criteria that relate to human actors in authorization. Resource owners can be the individuals authoring content or owning devices, principals can be individuals accessing protected resources, and, as part of the policy management, policy makers and policy authors are responsible for deciding on and formalizing authorization restrictions. We discussed in the previous section analytically who these actors may be in organizational contexts. For a more concrete picture of who is impacted in what ways by organizational authorization, we need to examine actual authorization contexts in practice.

In the Authorization Problems Study, we examined how authorization affects employees at a large organization, a European, multi-national company. The organization operates systems and maintains information at several levels of criticality and sensitivity, involving, among others, critical operations and market regulation as well as the sensitive personal data of employees and customers. In a study on security compliance, 118 semi-structured in-depth interviews were conducted with employees from management and staff in two countries between January and September 2010[6]. This study thus represents a case study according to the categorization in Section 2.5. The interviewees were recruited via the company newsletter, inviting volunteers to take part in an "IT Security Research Study" on their experience with the security policy for a gift voucher. From the about 400 responses within two days, participants were primarily selected on a "first come, first served" basis, with additional participants from later responses added for gender balance and breadth of work environments. The interviews lasted approximately 45 minutes, 40 being conducted via telephone and 78 face-to-face. The interview questions covered the interviewee background, experiences with the security policy, and how it affects the primary tasks.

Independently of the interviews, the author who is unrelated to the company coded the interview transcripts for tasks and challenges related to authorization. A common approach in HCI would be to identify the roles of stakeholders with respect to authorization in an organization. However, roles have the drawback of abstracting detail that is necessary to understand the behavior of people. As an alternative to roles, usability engineers employ the Persona methodology to preserve concrete characteristics, such as motivations and activities, of typical users (cf. Section 2.1.2 and Cooper et al., 2007). Faily and Flechais (2011) successfully employed Personas in security engineering.

We followed the approach of Cooper et al. (2007) and identified behavioral variables, such as attitudes, motivations, and activities, in the codes of the interview transcripts. We found 11 categories of behavioral variables with a total of 53 variables. From common combinations of behavior variable assignments, we derived behavioral patterns and formed five Personas, listed in Table 4.10. We then assigned personal authorization challenges to the individual Personas. Since Personas are originally a design methodology, it is difficult to validate them outside of design endeavors. Moreover, since the Persona methodology requires common patterns to shine through, the results will rather represent common Personas than a comprehensive list of how people interact with authorization. In addition, the interviews and sampling were not particularly aimed for comprehensive coverage of interaction with authorization. Resulting from a study at a single organization, the Personas also cannot be claimed to be representational without further validation. However, the breadth of the study should provide a good initial hypothesis of Personas illustrating authorization problems ("Authorization Personas") and allow for a vivid picture of how employees are typically affected by authorization. It is important to note that the Personas do not represent actual individuals from the study, but rather fictional constructions to exemplify the problems surrounding authorization.

[6]The general security compliance in the study was analyzed by Inglesant and Sasse (2011).

Persona/-Role	Motivation	Activities	Problems
Amber Functional staff	Personal, organizational, society risks; productivity	Share/access data, request changes, circumvent measure	Restrictive policies, degraded productivity, change lead-time and effort, unclear/ineffective/inefficient procedures, non-transparent decisions and policies
Emily Technically-informed staff	Recognition, (see Amber)	Develop, solve problems, make decisions	Non-transparent policies, coarse-grained permissions, lack of usability and functionality, unclear permissions, lack of high-level policy, missing expertise, emotional costs of decisions, informal procedures
Brandon Personal assistant	(see Amber)	Make decisions	Non-transparent policies
Lauren Functional manager	Personal risks and gains, organizational/society risks, productivity	Motivate compliance, make/delegate decisions, review policy	Retained permissions, non-transparent policy, inadequate model, lack of usability, decision complexity, required expertise, inefficient procedure, inefficient/ineffective reviews
Nicole Administrator, developer	Personal and organizational and society risks, risk awareness	Administer/develop applications, make decisions, support requests	Lack of high-level policy, cumbersome permissions, conflict of authority

Table 4.10.: Personas related to authorization measures

4.3.1. Functional staff

Amber is a business analyst in a technical department of the organization. She uses a number of company IT systems, for example, to share documents with co-workers and access data for analyses. Her main motivation concerning the use of the systems are her personal productivity, but also the general organizational efficiency and effectiveness, in that the necessary tasks are completed. Authorization measures affect her most directly through the operational aspects of *restrictive policies* that hinder her work or reduce her productivity, for instance, when she cannot access a document that was sent as a link to her. In some cases she is forced to circumvent the authorization measures and, for example, use the password of a co-worker to access data in the system. However, she feels uneasy about not complying with the organization's security policy that forbids the sharing of passwords.

Amber cares about the security of the sensitive data that she is handling, both for the risks that the organization faces, for example, from a disclosure, but also because of the consequences to her personally (concerning her employment) or the consequences for the society in general from the disclosure of critical data. Due to these considerations, she tries to comply with the security policies and requests changes to the existing permissions, but the requests can require a *high effort* and *take a long time* to become active, both operational authorization issues. For some systems, it is not clear to her how to request changes or it is known that not all requests succeed. Motivated to keep their documents secure, her team decided to also protect documents in the system with passwords, in addition to the system's authorization, since it is not always transparent who has access.

Emily also works as a business analyst, but has more technical experience. She tries to alleviate the problems with authorization that originate, for example, from too granular permissions or missing functionality in their systems. Because of her interests, she was tasked to develop the SharePoint site for her team. As a result, she also manages the permissions to the site, but *lacks clear guidance* in the form of a high-level policy to whom she should grant what permissions. She generally does not feel she has the *necessary expertise* to make these decisions and it is sometimes *unclear* to her which *permissions* need to be assigned and how to do this correctly. There are only *informal procedures* for handling permission requests and she sometimes feels a *high emotional pressure* to grant permissions, even though she is unsure whether the permissions are appropriate.

Brandon is the personal assistant to a manager, who has delegated the decisions on who should receive permissions to him. He has similar attitudes towards information security as Amber, and is particularly affected by the *non-transparency of the current policy*, since this makes it difficult to limit the authorized employees to those with a legitimate need.

4.3.2. Functional management

Lauren manages 40 employees in the financial department and feels responsible for their compliance to the security policy. She motivates her staff to comply by reminding them of why the data is sensitive, and the consequences of non-compliance, including sanctions. She is motivated by both consideration for her own employment (she values job security and also is rewarded for meeting compliance related performance objectives) and an awareness of the risks to the organization caused by non-compliance with regulations. At the same time, she also cares about the productivity of her staff.

As part of her role, Lauren needs to make decisions about authorization policies and review existing permissions on a regular basis. In this function, she is affected by operational authorization issues, such as the *inefficiency of the procedures*, when, for example, the procedure requires the signature of more senior personnel or from other departments. Generally, she sees the policy-authoring aspects of decision-making as a burden, since the *risk-assessment is complex* and *requires security expertise*. Moreover, it *requires significant technical expertise* to set the permissions due to a *lack of usability* of the management tool. The authorization model sometimes does not allow the precise setting of permissions or they can only be set in inefficient ways, for instance, requiring her to set numerous permissions for each individual of her staff. Consequently, she delegates some of the decisions to her subordinates. For reviewing existing policies, the lack of transparency of the current policies causes a high effort and even limits her ability to review. Since there are no automatic procedures for role changes, employees will also in many cases *retain their permissions*. When delegating functional tasks, Lauren is also affected by the lack of delegation options in the authorization mechanism, forcing her to sometimes share her password.

4.3.3. Technical staff

Nicole administrates and develops applications as part of the information system department. Because of her detailed knowledge of the systems, she is often consulted on how policy changes can be achieved in the system, for example, which permissions are necessary, and whether the changes are appropriate. In effect, she takes the decision in many cases, although she is not aware of all relevant high-level policies and is sometimes caught in *conflicts of authority*, for example, between departments when one is more security or business-focused than the other. Nicole is also affected by cumbersome permissions that make the policy management difficult and inefficient, for example, when permissions need to be set in a number of separate applications to allow an activity.

4.4. Perspectives on authorization

Even though not representational or comprehensive, the Authorization Personas in the previous section demonstrate the breadth of perspectives on authorization in organizations. To structure these

Mintzberg (Machine Bureaucracy) *Organizational part*	**Authorization Personas** *Organizational role*	**NIST Handbook** *Security mgmt. role*	**Varadharajan** *Authorization role*	**Authorization perspectives** *Perspective*
Operating core	Functional staff	Users	End user	Functional
Supporting staff	Administrator	Technology provider	Administrator	Policy authoring
Middle management	Functional mgmt.	Functional manager	Policy setter	Policy making
Technostructure	–	Computer security management	–	Security tactics
Strategic Apex	–	Senior management	–	Security strategy
Supporting staff	Developer	Technology provider	Developer	Development

Table 4.11.: Roles and perspectives in authorization

perspectives systematically, we relate organizational parts of Mintzberg (1980) to the roles of the Personas, security management (NIST, 1995), and prior work on authorization (Varadharajan et al., 1998) in Table 4.11. Prior definitions of authorization roles are typically focused on the technical aspects of principals, policy makers, and policy authors as discussed for the authorization contexts in Section 4.1 and by Varadharajan et al. (1998). These are also the most common roles and we similarly find those as the Authorization Personas. However, these roles primarily cover those individuals affected by authorization and neglect the further stakeholders who may also heavily influence authorization measures from a higher level of security management, as described, for example, by NIST (1995). As a second problem with roles, we observed for the Personas that it is difficult to adequately capture the different forms of interaction of individuals with authorization in rigid roles: In many cases, there is no clear distinction between the responsibilities. Instead, we define the *perspectives* of stakeholders on authorization (right-most column in Table 4.11), additionally including the security management aspects of designing procedures and of how decisions are taken. A sketch of how the perspectives interrelate is shown in Figure 4.2, which extends the first sketch from the introduction (Figure 1.1) with the additionally elicited perspectives. Specifically, we identified the following perspectives:

Functional perspective Staff with the functional perspective have to cope with authorization restrictions in the information systems that they interact with to complete their functional task. As in the case of the Persona Amber, functional staff are affected primarily by authorization restrictions, which can interfere with the productive use of the systems. Functional staff generally strive to optimize their efficiency and may thus be averse to any external factors that impact their efficiency or the established processes. When interferences with the primary tasks occur, functional staff may find other ways to complete their work, circumventing authorization restrictions where possible. In these cases, they may also support the policy making and authoring perspectives in setting adequate policies by proposing required permissions.

Policy-authoring perspective The restrictions in information systems need to be configured in a formalized way in technical policies to be interpreted by the system. In structured organizations, such as Machine Bureaucracies (Mintzberg, 1980), we often find the authoring perspective to be held by security administrators from the supporting staff or technology provider, similar to the Persona of Nicole. In these cases, the authoring of the policy is separated from the decisions on what restrictions to implement. Conversely, depending on the degree of centralization and the technical complexity of managing the policies, technically informed staff, such as Emily, take on this perspective as well.

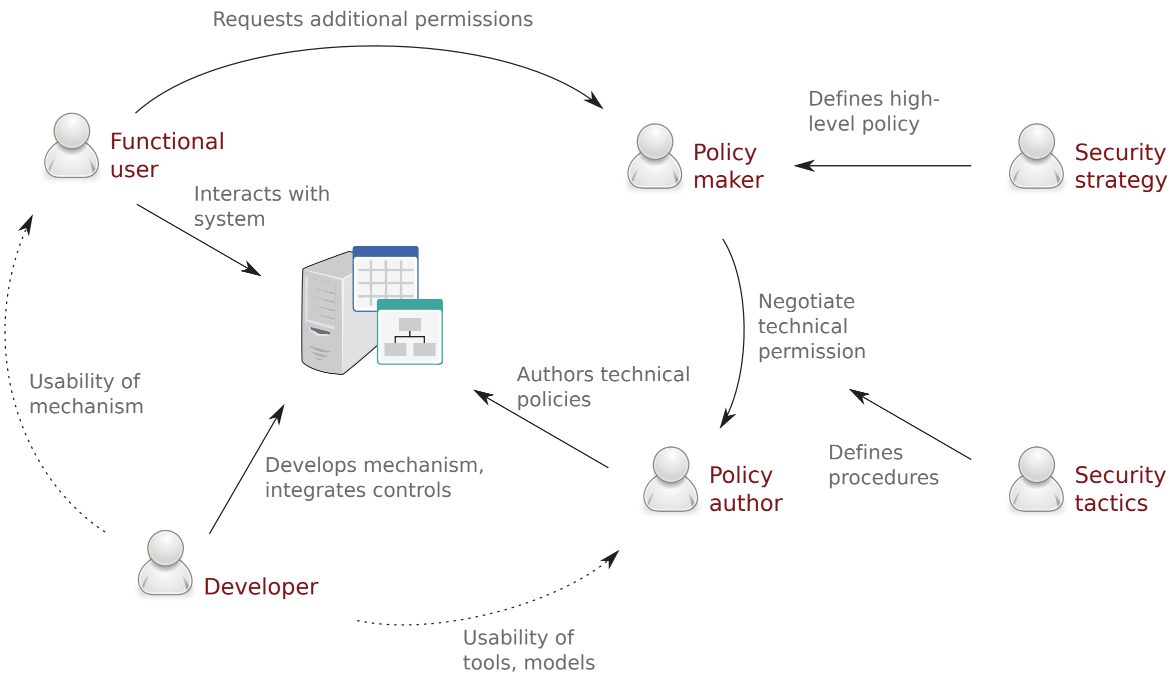

Requests additional permissions
Functional user
Interacts with system
Policy maker
Defines high-level policy
Security strategy
Negotiate technical permission
Usability of mechanism
Authors technical policies
Develops mechanism, integrates controls
Policy author
Defines procedures
Security tactics
Developer
Usability of tools, models

Figure 4.2.: A sketch of the interrelation between the Authorization Perspectives

Policy-making perspective The restrictions to implement need to be decided on before their formalization as technical policy by policy authors. These decisions can be taken at different points in an organization. In structured organizations, we will often find the decision to be made by a functional manager from middle management, such as the Persona Lauren. In other cases, the decision makers can also be personal assistants (Brandon) or technically-informed staff (Emily).

Security-tactics perspective Depending on the organization, formal processes may need to be established to organize the changes of the policy and the coordination between functional staff, policy makers, and authors as discussed for the management model in Section 4.2.2. The security-tactics perspective is responsible for defining the necessary procedures. In the Structural Configurations (Mintzberg, 1980), this is a typical task of the Technostructure. The NIST Handbook (NIST, 1995) assigns this perspective to the "computer security management."

Security-strategy perspective In order for policy makers to arrive at adequate decisions on what permissions to grant to whom, strategists need to define the parameters of these decisions, for example, in high-level policies. The NIST Handbook (NIST, 1995) calls for senior management to take a leading role by defining these parameters together with support staff who provide risk management and planning.

Development perspective On the systems development side, developers need to consider authorization when they perform development activities to implement and maintain the systems. The authorization-specific tasks depend on the development context. For custom-developed software, software developers need to take authorization into account in all the development activities to reliably achieve the enforcement in the system. For commercial off-the-shelf (COTS) software products, authorization development tasks in organizations are more focused on the integration into the existing infrastructure, for example, interfacing with the identity management system and the organization-wide policy provisioning.

4.5. Conclusion

As we saw in our analysis of authorization contexts in research, the contexts can be diverse, both in the characteristics of the contexts (resources, security needs) and in the architectural means to protect resources. Since our aim is to integrate different perspectives on authorization to effectively alleviate existing problems, we limit the scope of authorization by focusing specifically on organizational contexts, benefiting from the knowledge on organizational structures from decades of organizational research. We employed the Structural Configurations of Mintzberg (1980) to form a hypothesis of what authorization contexts to expect in what organization and to demonstrate that the authorization taxonomy is useful. We found that organizational characteristics very much affect authorization contexts: Only the security needs and the type of principals cannot be directly related to characteristics of the configurations.

Organizational authorization contexts are particularly interesting with respect to the interrelation between the diverse stakeholders as demonstrated by the Authorization Personas. To systematically foster the interaction between these individuals and integrate their tasks and responsibilities, we defined Authorization Perspectives to capture their individual points of view and structure further research. The perspectives are not independent of the taxonomy, but can be related: Stakeholders with the functional perspective will be principals of the authorization measure, policy authors and makers are part of the management model. The perspectives security tactics and strategy, and development are outside the scope of the taxonomy; stakeholders with these perspectives rather design the measure organizationally and technically, respectively.

Having more clearly defined the problem domain through the Authorization Taxonomy and the Authorization Perspectives, we can now explore in detail what problems affect the stakeholders with respect to authorization.

5. Problems in organizational authorization

I'm sorry, Dave. I'm afraid I can't do that.

Kubrick: *2001: A Space Odyssey* (1968)

In the previous chapter, we could observe the diversity of perspectives on authorization. Functional staff need to cope with restrictions as part of their daily work, policy authors formalize policies, policy makers take decisions on policy changes, and developers integrate authorization into information systems. As we already saw for the Authorization Personas, problems can occur in a variety of ways in these activities surrounding authorization and impact the effectiveness and efficiency of the authorization measure, the *authorization usability*.

Particularly the authoring of policies has been in the focus of research on the authorization usability (cf. Section 2.3). Researchers identified difficulties both in laboratory experiments (Zurko and Simon, 1996; Brostoff et al., 2005) and in organizational practice (Bauer et al., 2009; Smetters and Good, 2009; Whalen et al., 2006). However, the problems with authorization cannot be reduced to only the usability of management tools and the authoring of policies. Models of security economics emphasize the impact on security from usability and its interaction with productivity (cf. Chapter 2): For example, when the request of a policy change is perceived as too much effort, employees may rather share their password as a cheaper way of solving the problem. To understand the problems with authorization thoroughly, we cannot selectively focus on one perspective or the policy-authoring task, but must more broadly consider how authorization affects the different perspectives, their interaction, and how the problems interrelate.

In this chapter, we will analyze problems surrounding authorization with an integrative approach. We present the results of an in-depth study on authorization problems in a large organization. Based on the findings, we derive approaches to address the problems and a holistic model of the authorization problems. We close this chapter with a reflection on how the results from this chapter relate to the initial research questions and hypotheses on authorization problems.

5.1. Authorization problems in a large organization

The demonstrated selectiveness of the prior work with its focus on individual problems and policy authoring calls for a broader analysis of authorization problems, particularly in organizations where several individuals are affected by the authorization measure. Moreover, we find only few publications that explicitly discuss the causes and effects of the problems with authorization. One example is the study by Ahern et al. (2007) on social networking, in which they found that coarse controls cause users to circumvent the measure. Wool (2004) argued that more complex firewall rule sets lead to a higher number of errors in the policy. To effectively alleviate problems surrounding authorization, we need to address both challenges: broadly cover the different perspectives on authorization, and examine how the problems interrelate and where the problems originate. This extension of prior research may be compared to how the "Galilean-Newtonian revolution" (Finkenthal, 2001, p. 1) overcame the flat reporting of problems to rather examine the interrelation of causes and effects in a teleological approach (p. 44).

To derive insights about the authorization problems of the different perspectives, their interrelation, and their impact in practice, we conducted the Authorization Problems Study, based on interviews on the security compliance of employees at a large organization – a European, multi-national company. The organization operates systems and maintains information at several levels of criticality and sensitivity, involving, among others, market regulation, and the sensitive personal data of employees and customers.

5.1.1. Study design

Scholars of social sciences have spent decades on the discussion of the adequacy of different research methods to deliver reliable results (Bryman, 1988; Seale et al., 2007). Like for studying authorization in practice, key characteristics of social sciences are the context-dependent, diverse, and complex nature of research. We need to address two points here: first, whether to employ a quantitative or qualitative approach, and, second, what specific method to employ. Regarding the former aspect, Bryman (1988) argues that the distinction between quantity and quality is a purely technical one – two different ends of the spectrum of data collection strategies. He sees quantity to rather confirm existing theory, offer "hard, reliable" data, and having a general scope. Conversely, quality is more suited for emergent theory, offers "rich, deep" data for a more specific scope (p. 94). However, the generality of quantity is often "exaggerated" (p. 101) and it often "fails to give appropriate recognition to... entities which may not be directly observable" (p. 17). Quality allows us to better describe and contextualize findings (p. 61), often used as "reconnaissance" and "initial exploration" (p. 95); the point of qualitative research is not generalizability, but "the cogency of theoretical reasoning" (p. 123). This matches well with the goals of the Authorization Problems Study, which explores a breadth of problems and their interrelation not previously targeted.

The second major study-design aspect is the high-level method. Flyvbjerg (2007) strongly argues for case-study research, despite the conventional wisdom that "a case study cannot provide reliable information about the broader class" (p. 398). However, he argues that "formal generalization is overvalued as source of scientific progress" (p. 395). In complex disciplines with diverse contexts, "there does not... exist predictive theory... [only] concrete context-dependent knowledge" (p. 392), making case studies well-suited. We also need to be careful not to consider case studies "a sample of one," since they include a range of people (Bryman, 1988, p. 90). Moreover, context-dependency is vital, since – as in our study – it is often "more important to clarify the deeper causes... and its consequences... than how frequently [the phenomena] occur" (Flyvbjerg, 2007, p. 395). Since this corresponds with the goals of the Authorization Problems Study and since it is generally difficult to conduct studies in information security with large samples (cf. Kotulic and Clark, 2004), we decided to conduct an in-depth case study (cf. Section 2.5).

The data collection method and sampling for the study was described already for the Authorization Personas in Section 4.3. As detailed there, 118 semi-structured in-depth interviews were conducted for a security-compliance study with 118 employees from management and staff for the study, covering the interviewee background, experiences with the security policy, and how the policy affects the primary tasks.

According to Yin (2009), the quality of study design for case studies can be tested through four properties: *Construct validity* concerns that the phenomena studied are measured in an appropriate way. In this study, we apply a Grounded-Theory approach, which is argued for below. *Internal validity* ensures that causality claims are valid. In analysis, we build chains of causality and quantify their occurrence in a variety of contexts to ensure that causalities are valid. *External validity* relates to the above-discussed generalizability of the case study. Generalizability can analytically be shown to a certain extent through the adequate selection of the case under study (Yin, 2009, p. 43). We, accordingly, chose a company that offers a broad range of different authorization contexts and document the characteristics of the contexts to show for which contexts we expect the results to be generalizable. Further generalizability needs to be shown in follow-up studies. Lastly, *reliability* ensures that the study can be repeated in the same environment to come to the same results. We assure this property in this study by carefully documenting the data collection and analysis strategy.

Since the sampling process induced a potential self-selection bias, the quantity of mentions cannot be considered representative. However, the main goal of the study is not to elicit exact frequencies, but rather to derive a thorough description of the problems and their interrelation. The diverse backgrounds of the participants should allow us to formulate well-grounded hypotheses on the causes of usability problems in authorization for the remainder of this dissertation and future research.

Analysis

To analyze the interview transcripts, we applied a Grounded-Theory approach (Adams et al., 2008; Glaser and Strauss, 1967) because of its strength in systematically identifying, categorizing, and relating the concepts brought up by the participants of the study – particularly for broad descriptions of effects of technology and its generalization (p. 153). We focused on the authorization-related segments in the interview transcripts, coding for authorization usability problems and their causes and effects. For internal consistency of the coding, the author, who is unrelated to the organization conducted the coding. Almost two-thirds of the interviewees (75 of 118) mentioned authorization problems in one of the organizational information systems, including, amongst others, file sharing, administrative systems, and restrictions to Web access. Since authorization segments are sparsely distributed over the interviews, we coded in a two-stage process. In the first pass, we assigned broad categories of problems, for example, "policy change issue" to this quote:

> *"they may need temporary access... and a lot of the IS setup takes so long that a lot of these are workarounds to solve a temporary problem... problems tend to bounce around... for quite a long while"*

In the second pass, we then applied open and axial coding to the identified quotes for finer granularity. We established relationships between the codes through causal coding: We assigned three types of codes to quotes: the issue (the specific problem, "Change lead time" in the above example) as well as causes ("Multi-level procedure") and effects of the issue ("Social circumvention: Password sharing"). Employing our analysis tool, described below, the coding allowed us to generate causal diagrams of the authorization problems.

We were particularly interested in relating the problems to organizational goals. According to Schermerhorn et al. (2008), organizational goals fall into a three categories: *societal goals* (how to contribute to society), *output goals* (who is to benefit from the organization, e.g. shareholders, employees, customers), and *system goals* that relate to the survival of an organization. Authorization is most directly affected by system goals, including growth, harmony, prestige, productivity, profitability, and innovation. Particular priorities of the system goals depend on societal and output goals, but also vary among departments (Lawrence and Lorsch, 1969). When projecting these system goals to authorization measures, we arrive at the following underlying organizational goals that are affected:

- *Effectiveness of the security measure:* The degree to which the authorization measure increases the overall security as intended (profitability, harmony, prestige),
- *Efficiency of the security measure:* The effort expended by employees in operation to achieve effective security (profitability, productivity),

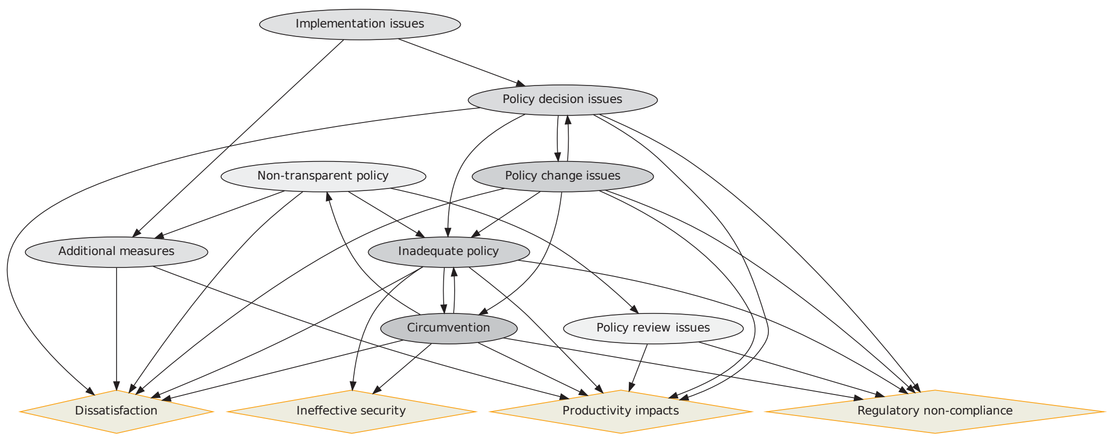


Figure 5.1.: Interrelation of categorized authorization problems

- *Regulatory compliance:* The compliance of the organization with laws and regulatory obligations, for example, from market regulation (survival, profitability),
- *Functional effectiveness:* The ability of employees to complete their primary tasks despite authorization restrictions (productivity, innovation),
- *Functional efficiency:* The effort expended by employees to complete functional tasks, particularly additional efforts caused by authorization measures (productivity, profitability),
- *Satisfaction:* Effects on the motivation of employees, such as frustration, from authorization measures (harmony).

Analysis tool

We coded 540 quotes in the interview transcripts, associating problems with the system context as well as with causes and effects as tuples or triples in a spreadsheet. We explored the data with an analysis tool that derives relationships from the coded quotes and generates diagrams using the Graphviz tool suite[1]. An example of the diagrams is shown in Figure 5.1: causal edges connect causes with problems, until reaching impacts on organizational goals at the bottom. The example quote from above results in the edge from “Policy change issue” to “Circumvention” in the diagram.

Since our coding is significantly more detailed, we implemented three levels of abstraction. At the most detailed level, all identified problems are shown with their *causal* (“Policy change issues” *cause* “Circumvention”) and *is-a* relationships (“Password sharing” *is a* “Social circumvention”). In a more abstract representation, all is-a relationships are flattened and the edges of the detailed level lifted to their parent nodes. The most abstract form is the one shown in Figure 5.1 and only displays problem categories. The different levels of abstraction allow us to both draw high-level conclusions and analyze the interrelations in detail.

The darkness of the shade of the nodes in the diagrams refers to the number of mentions in the study. Moreover, the diagrams also convey meaning through their structure, the node connectedness, and their relative position. For example, the central location and high interconnectedness of the

[1]http://www.graphviz.org/

problem “Inadequate policy” in Figure 5.1 points to its relative importance. Problems situated closer to the top of diagrams more indirectly impact organizational goals, but have, as a consequence, a higher knock-on effect (e.g. “Implementation issues” in Figure 5.1).

The tool further supported us by filtering the diagrams in two ways: First, by system context, allowing us to analyze individual authorization contexts. Second, limiting the diagrams to root-cause/ultimate-impact graphs, only showing those causes and effects that directly or indirectly relate to given problems.

5.1.2. Causes and effects

The authorization problems raised in the interviews allow us to derive general conclusions on authorization problems, their causes and effects. The causal diagram in Figure 5.1 depicts the interrelation of problems at a high level. Beginning at the impacts on organizational goals in the bottom of the diagram and following the causal links backwards, we describe the most severe and frequently mentioned problems in the following:

Problematic extremes in the continuum between restrictiveness and over-entitlements

As stated for the functional staff persona Amber in Section 4.3, the primary direct impact of authorization on primary tasks results from missing permissions due to restrictive policies (40 mentions), often seen as frustrating and affecting productivity, particularly when accessing the Web (26):

> *“all forums are blocked which is a bit of a pain… you are looking for sort of technical information… and you’ll find an old forum on it and you can’t view it so you kind of get ground to a halt”*

In contrast, over-entitlements (16) affect the organizational security when users have more permissions than necessary for their work. Interviewees named a number of causes for restrictive policies and over-entitlements. The most important ones are related to the policy-change procedures as well as to the decision-making for policy changes, discussed below. A further reason is the lack of transparency of policies (13). This problem leads, for example, to over-entitlement when policy makers cannot keep track of who has permissions on folders so that previously required permissions remain assigned (retained permissions: 4).

Restrictiveness and over-entitlements are the two ends on the continuum between confidentiality and integrity, and availability of information that we mentioned in the beginning of Chapter 2. Both extremes show to be problematic with respect to productivity, effective security, and stakeholder satisfaction.

Requesting policy changes

To correct restrictive policies and, less frequently, over-entitlements, functional staff, such as Amber, request changes to the policy as part of the authorization operation. The most frequently mentioned problems are the required effort to request changes (15) and the change lead-time (14), that is, the duration from requesting a change until its enactment in the system:

> *“if someone… needs to get access… immediately because it is job critical, then they will use that password in the meantime while they are waiting for theirs to come through.”*

The result from those problems is that the requester is forced to circumvent the authorization measure. The perception of the lead time and required effort also deters the functional stakeholder from requesting a permission in the first place, for example, when convenient circumvention is possible or the permission is only required temporarily. Similar to these issues are problems of unclear or ineffective procedures (13):

> *“accesses were challenging at the time... knowing who you go to, ask for what and how you know that that’s what you want... Shared areas were... problematic in identifying where the data was, who needed to approve the access to it”*

In these cases, the procedures are unknown or known not to help, thus further increasing the perceived effort due to the need to discover the procedure, or reducing the perceived effectiveness of pursuing a change of policy.

Making policy changes

The second perspective on problems with the change procedure is from policy making and authoring, from those deciding on and implementing the changes to the policies. Several of our identified personas are involved in these activities, including functional managers (Lauren), technically-informed functional staff (Emily), personal assistants (Brandon), and administrators/developers (Nicole). Here, one issue is the informality of procedures (3), leading, for example, to non-authoritative decisions (13):

> *“The responsibility in my group was just given to people that were the most computer-savvy at the time.”*

The primary challenge for policy makers is the lack of a high-level policy (5) that defines which permissions should be granted to whom. Determined to take appropriate decisions, policy makers find it difficult to properly evaluate requests without this kind of guidance:

> *“I don’t know about any policy on who should get access to my SharePoint site. It’s just based on need.”*

A common consequence is that many decisions are taken without a comprehensive consideration of the consequences of the decisions (17):

> *“did somebody actually sit there and think ‘Do you need this access?’... I get a person come and says ‘Hey, somebody told me I need this, can I have it? Give me this form and I can give it to you [signed].’ ”*

In this way, decisions are sometimes overly business- or security-driven, leading to over-entitlement and restrictive policies, respectively. In other cases, decisions are solely based on formalities, for example, neglecting to consider whether the access is actually necessary as long as the formalities, such as a specific training, are fulfilled by the requesting employee.

Related to these issues is the problem of conflicts of authority (2), for example, in the following example in which different departments have differing standards:

> *“one of the owners of the... shared drive, I’m one of the others, he was allowing all these other people, saying ‘I need to put so and so on.’ Well, I said ‘Do they have the [certification]? ... You’ve given them access to all that information.’ ”*

In other cases, particularly when decisions are decentralized, the emotional costs of denials (2) can be high and might even lead to policy makers taking inappropriate decisions:

Workaround/circumvention	Example	Effects
Utilize tech. loopholes (6)	Rename attachment extension	Functional inefficiencies
Increase redundancy (3)	Copy a document to a place which can be accessed	Inefficient, non-transparent policy
Multiple accounts (2)	Switch between accounts for different permissions	Undermines identity scheme, inefficient
Spare accounts (1)	Teams e.g. prepare a number of accounts for new temps	Potential missing traceability of activities
Make doc. public (1)	Move document to a public folder to allow access	Undermines policy enforcement
Share through alt. media (12)	Send document by email or physically on CD, rather than changing the policy	Loss of control over data; potentially increased risk; redundancy of data
Use private device (6)	Access information not available from work devices from smartphone or home PC	Risks from data on devices not governed by organization's security policy
Use external system (2)	Post documents on an organization-external system to grant access to externals	Risks from documents outside of the organization's security realm
Share passwords (21)	Share password instead of waiting for permissions to be changed	Lack of traceability/audibility; breaking security policy
Coworkers as proxy (3)	Turn to coworkers for a task due to lacking permissions	Potentially inefficient
Share logged-in account (1)	Have coworkers complete tasks at a logged-in computer	Lack of traceability/audibility

Table 5.1.: Workarounds, technical and social circumventions of authorization measures

> *"there have been a couple of people that have been 'Well, I'm not doing anything with it,'... 'Why are you so difficult'*

Another challenge arises from the implementation of authorization in information systems. In systems with inadequate authorization models (11), such as only offering coarse-grained restrictions, it is difficult to enforce the appropriate restrictions.

> *"So all the things people are working on, everyone has access to... that's the granularity that's given... because of the logistics associated with managing that sort of access."*

In some cases, the lack of usability of policy authoring also leads to high effort when employing finer-grained restrictions, thus preventing precise restrictions.

Circumventing authorization measures

The interviewees frequently reported that authorization problems, primarily restrictive policies and the perceived effort for policy changes, lead to the circumvention of the system, for example, through sharing passwords with coworkers:

> *"Sometimes people don't have access to information that they need to do their job and therefore the passwords are shared within teams. And I flagged that before, but it does happen, because it can take so long, months, to get something through. So it would be, 'Use somebody else's account.' "*

Overall, interviewees describe a high level of compliance in the organization and, for example, report that they generally feel uneasy when not complying with the rules. Considering the general tendency to comply, the high number of mentions of circumventions related to authorization (58) is interesting. The identified circumventions differ widely in severity with respect to their impact on the organizational goals. Circumventions range from sending documents by email instead of changing

	Microsoft SharePoint	Shared folders
Contingency factors		
Resource owner	Team, department	Organization
Security needs	diverse	diverse
Resource	Site (container)	Folder (container)
Principal	Employee	Employee
Security expertise		
—Principal	diverse	diverse
—Policy author	– (untrained staff)	+ (dedicated technical staff)
—Policy maker	diverse (often non-authoritative)	diverse (line manager, app. owner)
Complexity	High quantity, medium dynamics	High quantity, low dynamics
Design parameters		
Paradigm	Discretionary	Mandatory
Scope	Organization	Organization
Management model	Informal, decentralized: email to assigned person (policy maker and author)	Formal, centralized: Forms, approval from line manager and policy maker
Enforcement	Technical: SharePoint	Technical: folder access

Table 5.2.: Authorization characteristics of SharePoint and Shared folders

the policy to the sharing of passwords to grant access. We grouped the circumventions into the following categories:

- *Workarounds (13):* Using technical means within the system, for example, using multiple accounts,
- *Technical circumvention (20):* Using technical means outside of the system, such as sending documents on physical media,
- *Social circumvention (25):* Employing social means to work around authorization measures, such as sharing passwords.

We summarized the types of circumventions with examples and potential effects in Table 5.1 and found that circumventions impact five of the six organizational goals, including the productivity (security and functional efficiency), security effectiveness, regulatory compliance, and employee satisfaction.

5.1.3. Authorization paradigms

In the previous section, we studied general causes and effects of authorization problems. In practice, employees of the organization are affected by problems in a broad range of systems that inhibit distinct characteristics with respect to the authorization context. To analyze how these characteristics affect the problems, we selected two of the organization's contexts – "Shared folders" and Microsoft SharePoint – for a focused, comparative analysis. These systems are similarly broadly employed for sharing information and were mentioned frequently, but have opposing characteristics. Applying the previously developed taxonomy (cf. Section 4.1), we find that the systems represent different authorization paradigms and management models. As shown in Table 5.2, they particularly differ in the degree of formalization and centralization of the change procedures, and in the decision-making.

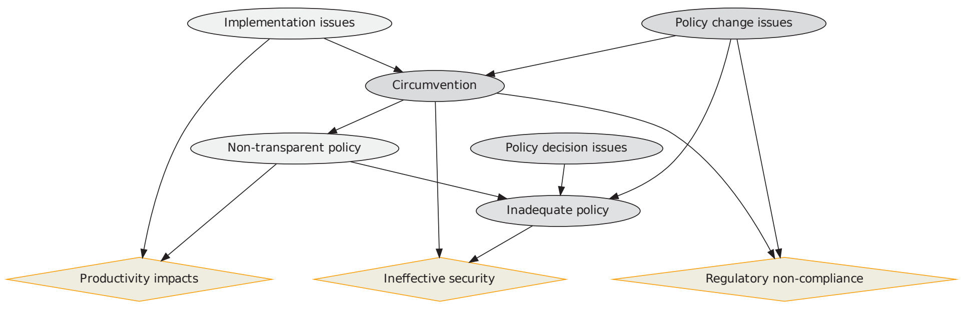


Figure 5.2.: Categorized problems with SharePoint sites

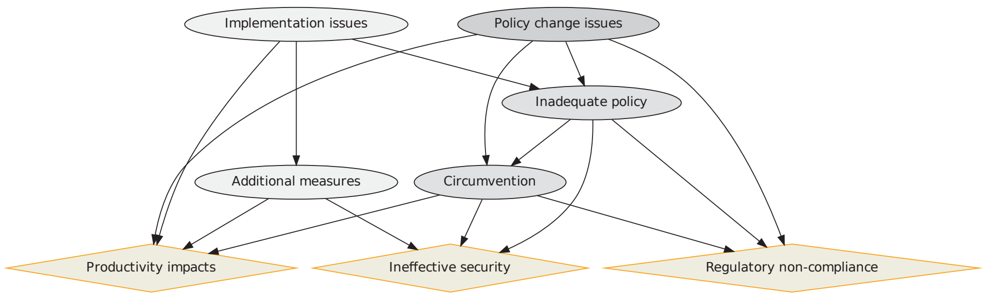


Figure 5.3.: Categorized problems with Shared folders

Microsoft SharePoint

Microsoft SharePoint is used in the organization to share documents within teams and departments as well as organization-wide. Departments and teams develop and operate local SharePoint sites. This includes the management of the authorization policy that grants access to the SharePoint-hosted resources on an employee level. Staff can request access to resources by clicking on a dedicated link when access is denied and thus send an email to the resource-specific policy maker, who then decides on the appropriateness of the request.

A high-level visualization of the interrelation of authorization problems for this system is shown in Figure 5.2. The effectiveness of security and regulatory compliance is impacted most, primarily due to permissive policies and circumventions of the authorization measures in the system. For example, staff use email for sharing documents instead of adapting the policy. The reason often are inadequate policies that result from problems with the change procedures and decisions, which are detailed in Table 5.3. Implementation problems, such as usability problems with the policy management, and the lack of transparency of the policies – preventing effective policy reviews – affect organizational goals as well, but to a lesser degree.

Shared folders

Shared folders represent the traditional way of sharing documents in the organization and are meant to be replaced by the above-discussed SharePoint infrastructure in the long run. Users access a shared-network drive from their desktop through the file explorer. Authorization is primarily en-

	SharePoint		Shared folders	
Problem	Dec.	Proc.	Dec.	Proc.
Decentralized decisions	2			
Required/present expertise	5			
Lack of high-level policy	3			
Non-comprehensive decision	3			
Business-driven decisions	1			
Coarse-grained restrictions	2		1	
Lack of usability	2			
Change lead-time		2		7
Required change effort				4
Ineffective change procedures		2		2
Availability of authority		2		1
Informal procedure		1		1
Inefficient procedure				5
Unclear procedure		1		4
Conflicts of authority				1
Non-authoritative decision		5		1
Effects				
Loss of traceability		2		
Over-entitlement	5		8	
Restrictive policy		1		6
Circumvention		8		13
Inefficient security		2		5
Functional efficiency		3		9
	23	29	9	59

Table 5.3.: Comparing mentions of problems with decisions and procedures

forced on a folder level. In contrast to the SharePoint procedures, folder permissions are granted to employees through a centralized process. They complete forms that are approved by their line manager and the policy maker for the resource (cf. Table 5.2).

As depicted in Figure 5.3, the policy change-procedures ("Policy change issues") were mentioned in the interviews most frequently (filled in darkest shade) as problems that impact, directly or indirectly, the effectiveness of security, productivity, and regulatory compliance. The policy change problems are listed in more detail in Table 5.3, notably including the high perceived change effort and change lead-time.

Comparing the paradigms

While apparently similar in terms of structures in the diagrams, there are a number of differences when examining the problems with change procedures and policy decisions. Specifically, procedure issues appear to be more prevalent for Shared folders (cf. Table 5.3, column "Proc."). For instance, the change lead-time and the required change effort is most relevant for Shared folders. Similarly, there is less mention of circumvention and productivity impacts from hindering work or inefficient procedures for SharePoint sites.

While the SharePoint procedure has fewer negative impacts overall, there can be more severe problems with the procedure if, for example, the responsible person does not respond:

> *"Sometimes you don't get a response for months and you don't know who to chase. At least with the Shared folders [process] you've got the request number and you can ring up about it."*

Overall, the sum of mentions of procedure issues (shown in the bottom row of the table) is lower for SharePoint.

A second area of effects of the paradigms are the decisions on policy changes. In this case, the sum of mentions in the table indicates that decision problems are more frequent for SharePoint than for Shared folders. Primarily, there is a concern in the case of SharePoint that due to the local decisions and the informal nature of the procedure, the decisions might not in all cases be adequate and the local policy administrators may lack security expertise (cf. Table 5.2):

> *"A user who is not trained properly can actually give access to everything easily through a couple of clicks in SharePoint; I think it is quite easy to not give access to the right areas... because it has quite a confusing way of giving permissions"*

In contrast, there is only a small number of mentions of decision problems for Shared folders.

The significance of the authorization paradigm

Overall, our analysis indicates that the design of the authorization measure (cf. Table 5.2), particularly the chosen paradigm, can have considerable effects on organizational goals. According to the mentions by study participants, it can be more efficient to have decisions taken and enacted locally (SharePoint) than in a centralized procedure (Shared folders): change lead-times are lower, there are less circumventions, and thus less impact on staff productivity and the effective security. Conversely, problems with the local changes primarily arise from overly informal procedures and a lack of expertise in the decision-making. These findings are in line with the theory by Pallas (2009), who predicts the trade-off between hierarchical and market forms of coordination for information security in organizations. Particularly, we see the effects of information asymmetries in the lack of expertise of local policy makers and the high hierarchical coordination costs for centralized decisions.

5.2. How to address the problems in organizations

The Authorization Problems Study showed problems and their interrelation, and allowed us to compare centralized/structured with local/informal procedures and decision-making. Despite the lack of representativeness, we can derive a number of recommendations on the design of authorization measures from the rich picture from the study:

5.2.1. Guide and monitor circumventions

Participants of our study reported numerous types of circumventions that impacted the effective security, their productivity, and their satisfaction. We can address the negative effects of circumventions in the following ways:

- *Foster acceptable circumventions:* Participants of our study reported that they would have to wait for policies to be changed to complete their task if they did not circumvent the authorization measure. Circumventions are thus not necessarily the inferior option, particularly when considering the impact on productivity. However, the participants also stated that they feel uncomfortable when they need to break a security policy in the process. To improve on this situation and since architectural means cannot entirely prevent circumventions, organizations should rather foster *acceptable* circumventions, guiding what circumventions are reasonable and in what situations. Formal rules through security policies and informal rules through common understanding in teams must complement architectural measures, as emphasized, for example, by Pallas (2009) and Whalen et al. (2006).

- *Monitor circumventions and the use of additional controls:* Our findings on the effects of authorization problems show that circumventions are a good indicator of underlying problems and should be monitored closely to identify and solve problems in existing policies and procedures. Similarly, additional authorization measures, such as passwords on shared documents, can indicate that the available authorization models, policy authoring tools, or change procedures are inadequate.

5.2.2. Establish adequate procedures

We found many cases in which the procedures for policy changes caused authorization problems. This extends the prior work on supporting the communication between stakeholders in policy management (cf. Section 2.3.3).

- *Define and communicate procedures:* Procedure ambiguity and informality affected the change operations, the interactions between the perspectives, and organizational productivity. For instance, participants did not know how to request permission changes. Clearly communicating the procedures to functional stakeholders will also lower the threshold for requesting changes.

- *Reduce the (perceived) change lead-time and change effort:* Circumventions were often caused by the duration for the changes to be enacted and the effort to initiate changes. Applying economic models to security usability (cf. Section 2.6) indicates that we need to reduce the costs of compliance. We thus expect that a reduction of the change lead-time and effort, and their perception may result in a reduction of circumventions.

- *Adjust the degree of centralization and formalization:* Our observations on the different authorization paradigms show that decentralized and informal procedures can be beneficial. This is also supported by economic models of information security management (Pallas, 2009). However, the adequate procedures depend on the context, the involved stakeholders must have baseline security expertise for adequate decisions (see below), and the procedures may need to remain traceable, depending on the environment.

5.2.3. Support policy decisions

Our findings indicate that many authorization problems originate in the decision-making part of policy management:

- *Provide high-level policies and guidance on authorization decisions:* Policy makers stated that they lacked guidance on what decisions were appropriate. One way to provide support is through high-level policies on how to decide on requests. These policies need to be adequate for the specific context, actionable, and comprehensible. A related option to support policy makers is to offer dedicated tools that help in the decision-making process. A comprehensible overview of decision factors may increase the awareness for the full breadth of factors. This follows the similar suggestions on decision support, such as for password policies (Parkin et al., 2010) and security investments (Beresnevichiene et al., 2010).

- *Increase the expertise and awareness of policy makers:* Participants further stated that decisions were often biased – for example, overly followed formalities, or the business or security perspectives. Additional expertise and awareness of concrete consequences from decisions, both of risks from grants and of functional impacts from denials (e.g. through participation of staff), may improve the appropriateness of the decisions (cf. West, 2008).

- *Improve authorization models and management tools:* Participants described how they are impacted by problems with the authorization model and the lack of usability of policy-management tools. These findings support prior work on policy editing and problems in practice (Sections 2.3.1 and 2.3.3) that emphasize the necessity of appropriate editing tools and authorization models for effective measures.

5.3. A holistic model of authorization problems

The Authorization Problems Study shows that solving the problems requires addressing several interrelated aspects. We first consolidate the identified problems with those from literature. Then, we derive a problem model that abstracts from the individual problems to reveal their structure and to clarify their interrelation – enabling a holistic approach to authorization usability.

5.3.1. Consolidating problems from literature and our study

Prior literature focused primarily on problems related to policy authoring and did not systematically study the relation of authorization problems with their respective causes. Nevertheless, the identified problems in prior literature cover a broad range. In Table 5.4, we consolidate the problems in literature and those from the Authorization Problems Study in this chapter according to the structure used in the study: Since our focus lies on solving the problems, we group problems by the perspective that is primarily responsible for the problem and can most directly address it, the *originating perspective*. Another interesting characteristic of the problems is which perspective they *affect*. For problems that are only present in literature, we analytically assigned originating and affecting perspectives to

		Affecting persp.							Originating	Liter-	Mentions		Disci-
		D	S	T	M	A	F	O	perspective	ature	in study	Context	plines
	Development												
1	Model expressiveness				x	x	x		D S M	F	35	O S	Se
2	Comprehensibility of model					x			D	F L	–	G O A	St
3	Comprehensibility of policy				x	x			D	F L A	–	G O Fs Fp	St
4	Modification usability					x			D A	L A	2	O Fs Fp A	St
5	Non-transparent mechanism				x	x	x		D M	F L A	13	O	St
6	Controls integration problems	x						x	D	F L	–	O K	Se
	Security strategy												
7	Insufficient decision guidance				x	x	x		S	–	9	O	Ds
8	Decision effort and complexity				x				S T D	F A	2	O	Ds St
9	Policy review usability				x			x	S T D	–	9	O	Ds O St
	Security tactics												
10	Change lead-time						x		T	–	14	O	O
11	Change request effort						x		T	–	15	O	O
12	Unclear, lacking, ineffective proc.						x		T	–	13	O	O
13	Procedure circumvention							x	T	–	2	O	O
14	Ineffective procedure execution				x	x		x	T	–	16	O	O
15	Inefficient procedure execution				x				T	F A	8	O Fw	O
	Policy making and authoring												
16	Non-comprehensive decision						x	x	M S	–	17	O	O
17	Policy complexity					x			M D	F A	–	O Fw	O St
18	Over-entitlement							x	M A	F A	16	O Fw	O St
19	Restrictive policy						x	x	M A	F	52	O	O St
	Functional management												
20	Circumvention							x	F	A	58	O	O St
	Organization												
21	Dissatisfaction							x	F M T S D	–	17	O	St
22	Ineffective security							x	F A M T S	–	27	O	O St
23	Reduced productivity							x	F M T S	–	40	O	O St
24	Regulatory non-compliance							x	F A M T S	–	7	O	O St

Table 5.4.: Consolidated authorization problems from literature and our study.
Perspectives: D: development, S: strategy, T: tactics, M: policy making, A: policy authoring, F: functional, O: organization.
Literature argumentation: F: field study, L: laboratory experiment, A: analytical.
Contexts: O: organizational, S: social networks, G: grid, A: application authorization, Fs: file sharing, Fp: file permissions, K: kernel, Fw: firewall.
Disciplines: Se: software engineering, St: socio-technical, Ds: decision science, O: (socio-)organizational.

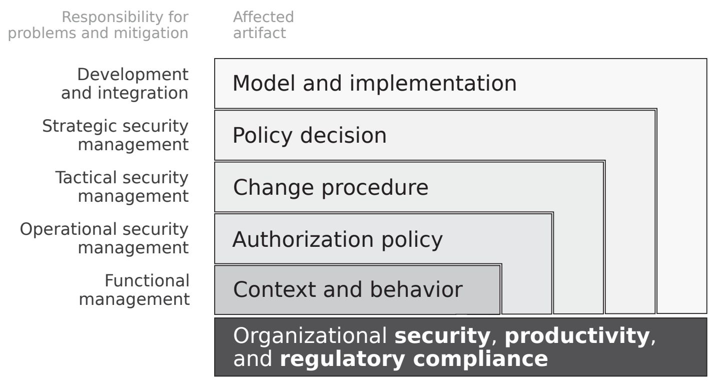

Responsibility for problems and mitigation
Affected artifact
Development and integration
Model and implementation
Strategic security management
Policy decision
Tactical security management
Change procedure
Operational security management
Authorization policy
Functional management
Context and behavior
Organizational security, productivity, and regulatory compliance

Figure 5.4.: Layered interrelations between the problems with authorization artifacts

enable a comparison with the problems from the study. For the problems from our study, we derived both from the identified interrelations of the problems.

Most prior literature concerns the model and tool usability, and related problems caused by the development perspective. Problems originating from policy making and authoring are present to a lesser extent. Only a small proportion covers the causes for problems with policy decisions (strategic perspective), procedures (tactical), and how the daily use of authorization mechanisms influence the organization (functional). Table 5.4 displays the type of evidence that the publications base their arguments on (column "Literature"). While, generally, most publications in the field of authorization are analytical, the table shows that most problems, if reported previously, were identified empirically in field studies or laboratory experiments.

As indicated by the column "Mentions in study" in Table 5.4, most of the consolidated problems were also described by our participants. In addition, our study covers a broader scope, including, for example, problems originating from the change procedures. Due to the broad spread of participants in the organization and no specific focus on administrative or development staff, our study emphasized problems affecting the functional perspective. As a result, the study does not cover three problems in prior literature that are caused by development and no problems *affect* strategy or tactics.

5.3.2. Identifying problem layers and perspectives

In Table 5.4, the problems surrounding authorization are structured according to their originating perspective. Examining the relation between the originating and the affected perspective, we can observe a layer structure: Beginning at the top of the table, the development perspective, problems from upper perspectives primarily affect those from lower ones and ultimately the organizational goals. To visualize the layers and their relationships, we can structure the problems by the affected artifact as depicted in Figure 5.4. Table 5.5 displays examples from our study of how the layers interact. The problem layers correspond well with the authorization perspectives as identified in Section 4.4. Only since the problems originating from the policy-making and authoring perspectives are closely interrelated, these are consolidated in the policy layer, attributed to the operational security management. A special case are the organizational problems: These problems affect the organization directly and originate from a range of higher layers so that they were rather grouped by their effect instead of by their origin.

Interestingly, within our study and the authorization literature, the upper three layers are seldom affected by problems. This may result from the focus of the study and the literature. While our study had a broader scope than the literature, it emphasized the functional perspective on problems and

Artifact	Responsibility	Activity	Example problem	Primarily affected	Example approach
Model, implementation	Development	Develop, integrate	Unusable model or interface	Policy making, authoring	Improve model or tool usability
Policy decision	Security strategy	Make high-level policy	Missing high-level policy	Policy making	Provide high-level policy
Change procedure	Security tactics	Design process	High change lead-time, inefficient procedure	Functional, policy authoring, making	Establish light-weight procedure
Authorization policy	Policy making, authoring	Change policy	Restrictive policy, permissive policy	Functional, organization	Grant adequate permissions
Context and behavior	Functional	Influence behavior	Reduced productivity, circumvention	Organization	Increase risk awareness

Table 5.5.: Artifacts in authorization and how problems with them interact

did not particularly target the development, strategy, and tactics perspectives. To properly address problems originating from these perspectives, we will need to further investigate the respective causes in future research. The practical contributions of this dissertation (cf. Section 6.4) in part serve this purpose.

5.3.3. The case for a holistic approach

As Table 5.4 indicates, prior work on authorization problems primarily focused on how policy authoring is affected and primarily covers problems originating from the development perspective (cf. Section 2.3). From the interrelations between the given layers, we can expect that such a selective approach can only solve part of the problem. There are indirect effects of higher layers that will reduce the effectiveness of approaches on individual layers. One of the examples given in Table 5.5 is the missing guidance for decisions on the *Policy decision* layer that impacts the adequacy of policy changes. If this problem exists, focusing selectively on the usability of the configuration interface might prove futile. Instead, we need to broaden the analysis and mitigation of problems surrounding organizational authorization and need to address problems of all affected perspectives by targeting the originating perspectives.

5.4. How the findings relate to research questions and hypotheses

We now have a clearer picture of the problems surrounding authorization in organizations. While it is difficult to claim comprehensiveness of problems in a field as diverse as authorization, the analysis of authorization problems in this chapter was based both on an extensive literature review and on the in-depth Authorization Problems Study in an organization with a variety of authorization contexts. This allowed us to enumerate a broad range of problems in organizational authorization and their interrelations, even though, as indicated above, further research will be useful on the problems that affect security strategy, tactics, and development. Despite this, the thorough analysis of problems can serve as a strong hypothesis of the problems to expect in authorization and results in a solid framework to structure authorization problems. Employing these results, we can now relate the findings to the research questions and hypotheses from the introduction of this dissertation (cf. Section 1.2):

5.4.1. Problem manifestation

Concerning the first research question, which asks for the problems of stakeholders, the stakeholders' perspectives and their interrelation, we first identified the perspectives in organizational authorization (cf. Section 4.4). Our model of the perspectives was useful for structuring the authorization problems from literature and our study, both regarding the affected and the originating perspectives of problems, thus supporting the perspective model. Further addressing the question, we thoroughly analyzed the problems surrounding authorization in the previous sections; we showed their interrelation in the holistic problem model and argued for a holistic approach to the problems. Generally, we can state that there is a close interrelation between the problems of the individual stakeholder and the those of the organization. Accordingly, the organization needs to take on the challenge to solve the stakeholders' problems in order to arrive at effective security and high productivity. The identified perspectives and the shown interrelation of the respective problems supports the hypothesis H1, which states that stakeholders face authorization problems from multiple, interrelated perspectives.

We can also observe how problems interfere with the interrelations between the perspectives, supporting H2. Specific examples from our study are:

- The comprehensibility of the authorization model and policy as well as the usability of modification tools and languages increase the minimal necessary security expertise of potential policy authors and can, for instance, prevent policy makers from also authoring policies,
- The lack of transparency of the policy makes it more difficult for policy makers to achieve adequate policies and prevents efficient communication with policy authors. For functional staff, not knowing their current permissions impedes precise requests for changes from policy makers,
- High change-request effort and lead-time, as well as unclear, lacking, and ineffective procedure obstructs the communication between functional staff and policy makers for policy changes,
- Ineffective, inefficient procedure execution interferes with the interrelation between functional staff, policy makers, and authors when working on enacting policy changes.

In particular, we saw two extremes of problems with authorization: In centralized procedures, problems were caused by high lead-times due to bureaucratic structures. At the other end of the spectrum, technology enables the delegation of management tasks to non-experts through decentralization and reduces those problems. However, this approach likewise fails to deliver effective security: The tasked stakeholders neither have the time nor the expertise to take adequate decisions on permission requests.

5.4.2. Problem mitigation

As part of our study on the problems in authorization, we only touched the potential mitigations as targeted by the second research question in this chapter. However, Table 5.4 shows examples of how the discovered problems surrounding authorization relate to research disciplines: We identified socio-technical problems from several perspectives that require *socio-technical studies* to enable effective and efficient interaction of the affected stakeholders with the systems. For several problems, socio-technical approaches need to be combined with *organizational sciences* to adequately integrate the interaction with the systems in effective and acceptable processes and procedures. Problems from the strategics perspective require *decision science* to enable and support decisions based on the risks, benefits, and the involved stakeholders. Lastly, *software engineering* can support the modeling of authorization and the integration of authorization in applications.

Moreover, the rich feedback of study participants led to a number of recommendations that we can relate to the hypotheses on the problem mitigation. Specifically, we saw how we can address the problems on a functional level (circumventions), tactical (procedures), strategic (decisions), and development (authorization model and management tools). The recommendations cover all the hypotheses on problem mitigation. For instance, the recommendations on acceptable circumventions combine technical with non-technical means (H3): architectural means, and formal and informal rules, respectively. Explicitly taking the dynamics into account (H4) is at the core of the recommendations on reducing the change lead-times and effort, and on appropriate centralization and formalization. The integration of perspectives (H5) is, for example, fostered by clearly defining and communicating the procedures, and by striving for usable editing tools and authorization models. The recommendations thus indicate that the hypotheses on problem mitigation are adequate. To more thoroughly explore the mitigations and test the hypotheses, we now need to take a more systematic look at the requirements that follow from the identified problems and at the mitigations of the problems in the following chapter.

6. Requirements and approaches

The rest was not perfect.

Sir Francis Bacon: New Atlantis (1626)

In the previous chapter, we took an in-depth look at the problems that can occur when implementing authorization measures in organizational information systems. This enabled us to address the first overarching research question from the introduction on the problems with authorization, the affected stakeholders, their respective perspectives, and the interrelation of problems and perspectives. Continuing from the identified problems, we will now extend on the recommendations from the Authorization Problems Study and systematically approach the second research question: How can the problems be solved in practice? We will take three steps towards answering this question in this chapter. First, we derive requirements for usable authorization measures from the previously identified problems. Second, we discuss how well the existing approaches on solving problems from literature fulfill these requirements and identify shortcomings. Third, we formulate detailed open research questions that address the shortcomings of the existing approaches.

In the introduction, we posed three hypotheses on how authorization problems might be approached (cf. Section 1.2). To explore approaches to the open research questions and test the hypotheses, a number of practical contributions will follow this chapter. These will be based on eight Authorization Principles to alleviate the problems that we consolidate from generalized approaches to security usability (Chapter 2) and security in agile development (Chapter 3). At the end of this chapter, the practical contributions will be introduced and related to the open research questions and Authorization Principles.

6.1. Authorization usability requirements

From the usability problems surrounding authorization that we identified in the previous chapter based on existing literature and the Authorization Problems Study in a large organization, we derive requirements for usable authorization measures. To illustrate that the requirements are well grounded in the problems, the problems in Table 5.4 addressed by a requirement are referenced for each requirement. First, the relation shows that all problems are covered by requirements: The requirements are complete with respect to the problems. Second, all requirements relate to identified problems, indicating the relevance of the requirements.

The requirements are grouped by the authorization perspective from which the requirement can be best addressed (cf. the problem model in Section 5.3). One example is the requirement of policy and tool usability (Req 13), which affects policy authors and makers, but can be best addressed in development and is thus assigned to the development perspective. A special case are the organizational requirements, which are rather abstract and can be addressed from several perspectives, and are thus separated from the more concrete requirements into a distinct group.

Similar to the interrelation of the problems as shown in the previous chapter, the requirements depend upon each other. The interrelation is particularly pronounced between perspectives. An example is the requirement on decisions on policy changes (Req 6), which, for instance, depends on adequate decision guidance (Req 11). As a consequence, addressing a more abstract requirement, such as decision-making, can also require addressing a more specific one, such as decision guidance.

6.1.1. Organizational requirements

Req 1 *Effective and efficient authorization measure*
Organizations as resource owners strive to achieve both, effective authorization, that is, the actual protection of resources as needed, and an efficient measure, that is only expending the minimal necessary efforts. Typical problems with the effectiveness of authorization stem from over-entitlements as well as from the loss of traceability and control from circumvention of the measures and problematic procedures. The efficiency is affected by the amount of work necessary for stakeholders to manage the measure, including the maintenance of the policy and the overhead from procedures and policy reviews. – *Problems 22, 23*

Req 2 *Minimal interference from the authorization measure with functional productivity*
Authorization measures can incur additional efforts for individuals when pursuing functional tasks. These are typically caused by interferences with the tasks from restrictions, requests of changes, or circumventions of the restrictions. – *Problem 23*

Req 3 *Conformance of the authorization measure with regulatory requirements*
A host of regulations affect organizations and depend, for example, on the specific industry, such as finances and health care (e.g. HIPAA), the form of organization (e.g. publicly listed companies: Sarbanes–Oxley Act), and the data to be protected (e.g. privacy legislation). For organizations, authorization can be a key measure to achieve compliance with the regulations. In these cases, the regulations need to be upheld by the policy, and by the decisions on and procedures for changes to the policy. – *Problem 24*

Req 4 *Satisfaction of stakeholders affected by and interacting with the authorization measure*
While employee satisfaction is only a secondary goal after, for example, shareholder value for many organizations, the satisfaction is still a requirement for authorization measures in many contexts. The requirement primarily addresses the inconvenience that is caused by authorization measures, both for functional staff hindered in their work, and for policy makers and authors in deciding on and configuring restrictions. – *Problem 21*

6.1.2. Functional – Context and behavior

Req 5 *Acceptable circumventions of the authorization measure*
For the overall effectiveness of the authorization measure, organizations may need to prevent circumventions of the measure, such as the sharing of passwords. Conversely, for the overall efficiency of the functional work, an organization may also foster acceptable circumventions. Both can be achieved, for example, through improved procedures, increased awareness for the involved risks, formal/informal rules and social controls, and the monitoring of circumventions. – *Problem 20*

6.1.3. Policy making and authoring – Authorization policy

Req 6 *Precise and efficient decisions on policy changes*
Precise decisions on restrictions increase the effectiveness of the measure (reduce over-entitlements) and the efficiency of functional tasks (reduce overly restrictive rules). It is also necessary to limit the effort expended on making the decisions to reduce the overall overhead of the authorization measure. Potential approaches are to reduce the complexity of decisions or to increase the concreteness of the considered risk and benefit factors. *– Problems 16, 18, 19*

Req 7 *Precise and efficient policy reviews and compliance monitoring*
Many organizations also task policy makers with regularly reviewing policies to ensure that the entitlements remain adequate. Policy makers may also need to decide in retrospect whether activities of functional users are compliant with the organizational rules as part of the monitoring of activities. In both cases, the decisions need to be made precisely and efficiently for an overall effective and efficient measure. *– Problems 9, 18*

Req 8 *Precise and efficient authoring of policy changes*
The decisions on policy changes need to be formalized in technical authorization policies so that they can be interpreted by the information systems. Imprecise changes can endanger the effectiveness of the measure, for instance, through mistakes in the editing of the policy. A high effort for changes reduces the measure's overall efficiency. *– Problems 17, 18, 19*

6.1.4. Strategics and tactics – Policy decision and change procedure

Req 9 *Clear, effective, and efficient procedures for requesting changes to the policy*
One problem for functional staff when encountering an inadequate permission is how to request the change of the permission. Process designers need to assure the awareness for the respective procedures and design them in a way that requests result in changes if adequate. The effort to trigger and the time to complete the procedure also needs to be adequate. *– Problems 10, 11*

Req 10 *Adequate, effective, and efficient processes for policy changes*
Policy changes often involve multiple stakeholders, from the requesting person over policy makers to the policy authors who implement the change. The processes for coordinating the changes need to have adequate degrees of centralization and formalization to achieve the necessary traceability and reliability of procedures at an acceptable overhead for policy makers and authors. *– Problems 12, 13, 14, 15*

Req 11 *Strategic guidance for precise and efficient decisions*
Policy makers need to be brought into the position to make precise and efficient decisions on policy changes and on the appropriate usage of permissions. Depending on the context, policy makers may require implicit or explicit guidance on what constitutes adequate usage of the system. Organizations can, for instance, provide high-level policies to support policy decisions that consider the risks and benefits of entitlements – but also the decision effort. *– Problems 7, 8, 9*

6.1.5. Development – Implementation

Req 12 *Adequately expressive and comprehensible authorization model*
The authorization model defines the form of the entitlements and restrictions in a policy. Problems can occur when the expressiveness prevents the formulation of the necessary restrictions or when the available constructs result in overly complex policies, reducing the effectiveness or efficiency of the measure, respectively. Similarly, incomprehensible models may cause mistakes by policy authors or require overly high management effort. *– Problems 1, 2*

Req 13 *Comprehensible policy, and effective and efficient policy modification*
Policy authors and makers need to understand a policy to review it or to implement changes correctly. Depending on the context, policy authors and makers need an adequate representation, and effective and efficient tools that increase the comprehensibility of the policy, and improve the precision and efficiency of changes. *– Problems 3, 4*

Req 14 *Transparent, effective, and efficient authorization mechanism*
The implemented authorization mechanism in an application needs to support the authorization model and the procedures for policy changes. For example, the flexibility of the model may require additional interaction with functional staff to delegate permissions. Missing transparency of policy decisions may limit the usefulness of this kind of model functionality. *– Problem 5*

Req 15 *Precise and efficient integration of authorization controls*
As part of systems engineering, authorization controls need to be integrated in systems to reliably enforce the policy and thus enable an effective measure. For an efficient measure, the effort for the integration needs to be acceptable. *– Problem 6*

6.2. Requirement priorities

The authorization usability requirements cover a broad range of aspects. When designing authorization measures, the individual requirements need to be prioritized according to the specific authorization context. In the following, we will analyze what priorities can be expected for the requirements in a variety organizational and non-organizational contexts.

6.2.1. Organizational contexts

For the priorities in organizations, we can apply the context characteristics described in Section 4.1 to give an estimation of how the requirement priorities are affected by context characteristics as shown in Table 6.1. For each of the characteristics, it is indicated whether a high value for the characteristic increases the requirement priority (+) or reduces it (–). Table 6.2 additionally relates the requirements to organizational configurations as defined by Mintzberg (1980), estimating increased (+), neutral (o), and reduced (–) priorities. Specifically, the context characteristics are:

- The *security needs* of the authorization context, encompassing both the risks from missing protection and the need for the availability of the protected resources. The security needs are particularly important for requirements that influence the authorization effectiveness, such as an effective measure (Req 1),

	Security needs	Complexity	Dynamics	Expertise
Req 1 *Authorization measure*	+		+	
Req 2 *Functional productivity*	+	+	+	
Req 3 *Regulatory requirements*	+			
Req 4 *Individual satisfaction*				
Req 5 *Circumventions*	+		+	–
Req 6 *Policy decisions*	+	+	+	
Req 7 *Reviews and monitoring*	+	+		
Req 8 *Policy changes*	+	+	+	
Req 9 *Change requests*		+	+	
Req 10 *Change procedures*	+		+	
Req 11 *Decision guidance*	+	+	+	–
Req 12 *Model usability*		+	+	–
Req 13 *Policy and tool usability*		+	+	–
Req 14 *Mechanism usability*			+	
Req 15 *Controls integration*	+	+		

Table 6.1.: Requirement priorities by authorization context characteristics

- The *complexity* of the authorization context, including quantities and heterogeneousness of permissions, important for requirements related to the making of decisions and changes,
- The *dynamics* of the authorization context, which cause higher number of changes and thus affect many requirements since most include efficiency aspects of policy management,
- The organization's security culture and the level of *security expertise* and awareness of the involved stakeholders, influencing, for instance, how likely circumventions are (Req 5) and how much decision guidance is necessary (Req 11). Whether requirements are affected depends on the level of expertise in the roles of decision makers, functional, and technical staff.

When examining how the organizational configurations affect the requirement priorities (Table 6.2), we can observe that only the requirement on functional work is broadly important (four of five configurations) in organizations. The priorities of the other requirements vary between configurations. For the configurations, the Professional Bureaucracy and the Adhocracy stand out with a high number of important requirements. The former because of its dynamics, the latter because it is highly heterogeneous.

6.2.2. Non-organizational contexts

While this dissertation focuses on organizational authorization, it is useful to compare how the requirements relate to non-organizational contexts. Table 6.3 shows which requirements are particularly important for the example contexts from Section 4.1. We only leave out the file system example due to its generality and the organizational sharing examples as these are covered by the previous section.

The physical access contexts both are rather static and simple so that most requirements are generally less critical. In contrast, the smartphone and the Web 2.0 contexts demonstrate how many of the requirements can be important for very different contexts. No pattern emerges of individual requirements generally being of higher criticality, again indicating the high dependence of the priorities on the specific environment.

	Simple Structure	Machine Bureaucracy	Professional Bureaucracy	Divisionalized Form	Adhocracy
Req 1 *Authorization measure*	+ (efficiency for dynamics)				+ (efficiency for dynamics)
Req 2 *Functional productivity*	+ (interference from dynamics)	+ (interference from heterogeneousness)	+ (interference from heterogeneousness)		+ (interference from dynamics)
Req 3 *Regulatory requirements*					
Req 4 *Individual satisfaction*		+ (powerful Technostructure)	+ (powerful functional staff)		+ (staff-dependent)
Req 5 *Circumventions*	+ (dynamic)	+ (heterogeneous)	– (high functional expertise)	– (static)	+ (dynamic)
Req 6 *Policy decisions*	o (dynamic, but little complex)	+ (heterogeneous)	+ (heterogeneous)	o (quantity, but static)	+ (dynamic, heterogeneous)
Req 7 *Reviews and monitoring*				+ (high quantity)	
Req 8 *Policy changes*	+ (dynamic)	+ (heterogeneous)	+ (heterogeneous)	o (quantity, but static)	+ (dynamic, heterogeneous)
Req 9 *Change requests*	o (dynamic, but informal)	+ (heterogeneous)	+ (heterogeneous)	– (static)	+ (dynamic, heterogeneous)
Req 10 *Change procedures*	o (dynamic, but informal)	o (formal, but static)	– (informal)	o (formal, but static)	o (dynamic, but informal)
Req 11 *Decision guidance*	– (expertise through centralization)	o (heterogeneous, expertise in Technostructure)	+ (heterogeneous)	o (quantity, but static)	o (heterogeneous, but expertise)
Req 12 *Model usability*	o (dynamic, expertise through centralization)	o (heterogeneous, expertise in Technostructure)	+ (heterogeneous)	+ (high quantity)	+ (dynamic, heterogeneous, expertise)
Req 13 *Policy and tool usability*	o (dynamic, expertise through centralization)	o (heterogeneous, expertise in Technostructure)	+ (heterogeneous)	+ (high quantity)	+ (dynamic, heterogeneous, expertise)
Req 14 *Mechanism usability*	+ (dynamic)				+ (dynamic)
Req 15 *Controls integration*		+ (heterogeneous)	+ (heterogeneous)		+ (heterogeneous)

Table 6.2.: Requirement priorities for organizational configurations (for the characteristics of the configurations see Mintzberg, 1980)

	Physical office	Physical private	Android smartphone	Web privacy (visitor)	Web privacy (operator)	Web 2.0
Req 1 *Authorization measure*	o (often rather static office allocation)	- (typically low values, infrequent changes)	+ (diverse asset values, changes in apps)	- (diverse security needs, low dynamics)	- (diverse security needs, low dynamics)	+ (dynamics of content creation)
Req 2 *Functional productivity*	o (static, low complexity, diverse needs, high availability)	- (low values, static, simple)	o (diverse asset values, changes in apps, low complexity)	- (diverse security needs, low dynamics, low complexity)	- (diverse security needs, low dynamics, low complexity)	+ (high availability, high dynamics)
Req 3 *Regulatory requirements*	o (diverse)	-	-	+	+	
Req 4 *Individual satisfaction*			+ (selling point for app market)	-	-	+ (selling point of platform)
Req 5 *Circumventions*	- (rather controllable, high relative expertise)	- (high relative expertise)	- (n/a)	- (n/a)	- (n/a)	+ (other ways of sharing)
Req 6 *Policy decisions*	- (simple decisions)	- (simple decisions)	+ (difficult decisions)			+ (difficult to predict consequences)
Req 7 *Reviews and monitoring*	- (n/a)	- (n/a)	+ (difficult to monitor, review)			+ (difficult to monitor, review)
Req 8 *Policy changes*	- (simple controls)	- (simple controls)	- (simple controls)			- (simple controls for individual items)
Req 9 *Change requests*	- (infrequent changes)	- (infrequent changes)	- (n/a)	+ (complex: consent from all existing users)	+ (contract negotiations)	- (simple requests)
Req 10 *Change procedures*	- (infrequent)	- (infrequent)	- (integrated)	+	+	- (integrated)
Req 11 *Decision guidance*	- (simple decisions)	- (simple decisions)	+ (complex factors: consequences)			+ (complex factors: consequences)
Req 12 *Model usability*	- (simple requirements)	- (simple requirements)	+ (comprehensibility: func. users)	+ (comprehensibility: func. users)		+ (comprehensibility: func. users)
Req 13 *Policy and tool usability*	- (simple requirements)	- (simple requirements)	+ (comprehensibility: func. users)	+ (comprehensibility: func. users)		+ (comprehensibility: func. users)
Req 14 *Mechanism usability*	- (simple requirements)	- (simple requirements)		- (simple requirements)		
Req 15 *Controls integration*	- (simple requirements)	- (simple requirements)	- (largely framework-based)			

Table 6.3.: Requirement priorities for non-organizational contexts

6.3. Gaps in literature and the open research questions

Based on the existing approaches (cf. Section 2.4.4 and Appendix E), we examine the approaches for their coverage of the problems and requirements, and identify shortcomings, for which we formulate open research questions. Depending on the depth of prior work for a requirement, the questions vary in concreteness, either aiming to further explore the current practice, or to advance specific approaches through more thorough research.

When discussing approaches, the interdependence between requirements becomes apparent. The more abstract requirements, such as those from the organizational perspective, are generally addressed more directly through approaches related to more concrete requirements. To prevent repetitions of the approaches, we will thus rather relate approaches only to the most concrete requirement that the approaches apply to. The dependencies on more concrete requirements are also noted among the approaches listed in Table E.1.

Since the requirements and the analysis of approaches cannot be comprehensive due to the large number of authorization-related publications, the open research questions cannot be claimed to be representative or comprehensive. While a broad coverage of literature has been achieved by systematically reviewing literature on the individual requirements, there remains the threat that the areas covered by practical contributions in this thesis have received more attention than others. More generally, it is epistemologically difficult to reach a comprehensive coverage of the open questions, since every in-depth study of a research question will result in discovering further questions (cf. Popper, 1935/2002) or will even lead to a shift in the research paradigm with new questions (cf. Kuhn, 1970). The detailed research questions can thus only represent a snapshot based on the state of the field at the time of writing. However, since the open research questions have been systematically derived from the literature review and the requirements, the questions offer solid guidance for further research.

6.3.1. Organizational perspective

Req 1 Effective and efficient authorization measure Little prior work generally aims at how to achieve adequate authorization, that is, at effectively protecting resources while keeping the overall effort low. Most existing literature is more specific and approaches can be found, for instance, for the functional perspective and the adequacy of circumventions (Req 5), but also for policy authoring and making on making changes (Req 8) and decisions (Req 6). However, the specificness of the concrete approaches often neglect the overarching problems in organizational authorization and how to address them at an organizational level, specifically:

Q 1.1 *What type of organization requires what authorization measure and what are the influencing factors?*

Moreover, more practically:

Q 1.2 *How can we support the design of adequate authorization measures?*

Req 2 Minimal interference from the authorization measure with functional productivity There are few publications on the disruptions of functional work from authorization. While Whalen et al. (2006) formulate the recommendation from their empirical study to integrate authorization-related activities into the workflows and Johnson et al. (2009) propose the principle of minimal friction, there are little concrete approaches on the organizational level. We can find those for the more concrete requirements on which this requirement depends, for instance, on integrating functional staff into decision-making (Req 6) and policy authoring (Req 8). For a more complete picture of the nature and impact of disruptions, we need further empirical work:

Q 2.1 *How disruptive are authorization measures with what characteristics and in what contexts, and how can we reduce the disruptions or their impact?*

Req 3 Conformance of the authorization measure with regulatory requirements While authorization is an important measure for many organizations to comply with high-level regulatory requirements, little work specifically addresses the relation between regulations and authorization. The requirements on effective authorization (Req 1), and, particularly, compliant procedures (Req 10) and policy decisions (Req 6) provide general approaches on how to honor regulations as part of authorization. However, it should be useful to also specifically address the relation of regulations with authorization, both in decisions and procedure design:

Q 3.1 *How can regulatory requirements be enforced in authorization and employed in decision guidance?*

Q 3.2 *How do procedures need to be structured to comply with regulations and how do regulations need to be structured to allow procedures to comply with them?*

Req 4 Satisfaction of stakeholders affected by and interacting with the authorization measure Similar to most of the organization-level requirements, we also find little work directed at studying or improving the satisfaction of stakeholders with respect to authorization. We can expect that increased efficiency, particularly of functional work (Req 2) and circumventions (Req 5) for the functional perspective, and an efficient overall measure (Req 1) will positively influence satisfaction. For a more specific exploration of the causes of dissatisfaction and its effects, we need to explore the problem further:

Q 4.1 *In what ways does the satisfaction of functional staff interrelate with functional disruptions, security awareness and expertise, and organizational culture?*

Q 4.2 *How is satisfaction impacted for the authoring and making of policies?*

Q 4.3 *How does satisfaction influence the effectiveness of the measure?*

6.3.2. Functional perspective

Req 5 Acceptable circumventions of the authorization measure There are little concrete approaches directly addressing the problems with the circumvention of authorization. Apart from reducing disruptions (Req 2), the Authorization Problems Study and best practices recommend increasing the risk awareness, providing guidance on circumventions, and monitoring and penalizing non-compliance. To address the problems with circumventions, we need a more comprehensive understanding under what conditions they occur:

Q 5.1 *Under what conditions will individuals resort to circumventions?*

There are a number of options to improve the adequacy of circumventions, including awareness campaigns, security policies, and decision guidance, but we need to study their efficacy:

Q 5.2 *How must mitigations to undesirable circumventions be designed to be effective and efficient?*

6.3.3. Policy making and authoring

Req 6 Precise and efficient decisions on policy changes The most concrete prior work on policy decision-making can be found for the systematic elicitation of authorization requirements through methods (Neumann and Strembeck, 2002; He and Antón, 2009) and requirement analysis (Koch and Parisi-Presicce, 2003), but apart from the approach by Takabi and Joshi (2010), these are primarily applicable in one-off policy creation. More specifically aimed at the practical decisions, Whalen et al. (2006) recommend to employ less restrictive controls and support social controls. The results of the Authorization Problems Study further call for decision guidance (Req 11) and expressive models (Req 12). Church (2008) advocates an increased integration of functional staff in decisions, supported by approaches of Rode et al. (2006) and Karp and Stiegler (2010), and the usability of the mechanism implementation (Req 14). However, to more fundamentally address problems in making policy decisions, we need to better understand the decisions in practice:

Q 6.1 *How do policy decisions take place in practice and what information is employed?*

Req 7 Precise and efficient policy reviews and compliance monitoring Reviews of policies are a long-standing best practice in information security management, described, for example, in ISO/IEC 27002:2005 (2005). The specific approaches on reviews for authorization and its practice have not received as much attention. Sinclair et al. (2008) report how self-reviews of functional staff worked well in a financial institution. On the technical side, we need to consider the usability of review tool support (Req 13). In addition, we must also study the organizational side of authorization reviews:

Q 7.1 *How must review procedures be designed to be effective and efficient?*

Q 7.2 *How can different perspectives cooperate for effective and efficient reviews?*

Monitoring functional staff to discover non-compliant behavior is equally established in security management (ISO/IEC 27002:2005, 2005) and has been discussed for authorization as part of the enforcement (Sandhu and Samarati, 1996; Røstad and Edsberg, 2006), involving misuse and anomaly detection that needs to be implemented as part of the authorization controls (Req 15). Two aspects particularly warrant further research: the use of anomaly detection and the way to employ results for policy improvements:

Q 7.3 *How can IDS/SIEM paradigms be applied to authorization, for instance, in risk-based auditing?*

Q 7.4 *How can problems with the authorization measure be extracted from activity logs?*

Req 8 Precise and efficient authoring of policy changes While most of the prior work on making policy changes has occurred on a technical level, a small number of publications also directly addresses the question how changes can be made effectively and efficiently. One important approach is to increase the concreteness of the policy and changes, in line with Blackwell et al. (2008). They generally argue that forcing computational thinking on individuals can reduce the overall quality of the result since important non-computational confutations are lost. Approaches to more concreteness have been presented by Inglesant et al. (2008), using examples, and by Johnson et al. (2009), employing an email-sending metaphor for granting permissions. The principle of "security by designation" by Yee (2005) has similar aims.

A second approach is to reduce the complexity of policies as recommended by Wool (2004), relying on an expressive model (Req 12) and flexible or permissive decisions (Req 6). More generally, comprehensible policies and usable editing and supporting tools (Req 13) should be useful. The latter may also help to integrate functional staff into security management (Church, 2008) and support the interaction between policy making and authoring (Req 10). While the technical aspects are well covered and continue to receive attention, we still need to extend the research on the integrative aspects of policy authoring:

Q 8.1 *How do policy authors interact with policy makers and functional staff for implementing changes?*

6.3.4. Security tactics and strategy

Req 9 Clear, effective, and efficient procedures for requesting changes to the policy Similar to the requirement on functional work (Req 2), there is only little prior work on how functional staff requests changes to policies. The Authorization Problems Study resulted the recommendations on a reduced change effort, procedure awareness, and procedure adequacy. Given the lack of substantial work in this area, we need to further explore the problems surrounding change requests and how the conditions for change requests can be improved:

Q 9.1 *How do requests for changes to the policy take place in practice and what are the obstacles for functional staff?*

Q 9.2 *How can we foster adequate requests for changes from functional staff?*

Req 10 Adequate, effective, and efficient processes for policy changes Processes and procedures for the coordination of policy changes have been covered by process models on several levels (Dai and Alves-Foss, 2002), including management-focused (Rees et al., 2003; OGC, 2007c) and technical models (Kern et al., 2002; Mönkeberg and Rakete, 2000). Similarly concrete are the best practices on change management (ISSEA, 2003; Joeris, 1997) and validation plans (ISSEA, 2003). These approaches are based primarily on analytical work and selectively focus on specific perspectives. More generally and arguing empirically, Sinclair et al. (2008) recommends process-based role management, that is, integrating the role management in organizational processes. Our study similarly indicated the need for adequate procedures and for the awareness of these. To provide concrete guidance on how to design change procedures, we need to address additional questions on practical problems with authorization processes:

Q 10.1 *How formalized and centralized should procedures be in what context?*

Q 10.2 *How can we uphold the procedures and prevent procedure circumventions?*

Organizations often have already established related processes, such as Information Security Management (ISMS) and incident/issue management processes, that partly overlap with authorization management:

Q 10.3 *How can the authorization procedures be integrated with related organizational processes?*

Req 11 Strategic guidance for precise and efficient decisions One approach to guide decisions on policy changes is from existing organizational structures and functional processes, for example, through role mining (Bertino et al., 2008), task mining (Kern et al., 2002), or the use of organizational structures in role-based authorization (Moffett, 1998; Crook et al., 2003). Apart from these very technical approaches, decision guidance can also be based on the security needs and risk assessment as best practices suggest (ISSEA, 2003; ISO/IEC 27002:2005, 2005), or on creating awareness for the consequences of decisions, proposed for online social networks (Ahern et al., 2007). Ahern et al. (2007) and our study also recommend to provide concrete decision-support, based on prior decisions and risk assessment, respectively. Nissanke and Khayat (2004), for example, implemented a risk-based policy analysis, Molloy et al. (2008) a market-based decision-support, and Zhao and Johnson (2008) a game-theoretical model for guidance. However, the existing approaches do not address more fundamental questions that are necessary for employing decision guidance in practice:

Q 11.1 *What kind of guidance for authorization-related decisions is effective and efficient?*

For effective support of decisions, irrespective of whether, for example, tool-based or as high-level policy, we must base the guidance on the risks and benefits of permissions:

Q 11.2 *How can the information that is required for decision guidance be elicited?*

6.3.5. Development

Requirements in the development realm are well covered with technical approaches. However, many of the implemented approaches do not address the actual needs in practice and lack empirical backing regarding the elicitation of the problems and the validation of approaches in practice.

Req 12 Adequately expressive and comprehensible authorization model Regarding the effects of the authorization model on the overall authorization usability, we primarily find approaches in three categories. First, Blakley (1996) recommends to reduce the cognitive effort and improve the learnability and maintainability of authorization models. More concretely, Smetters and Good (2009) derive the recommendation to use simpler models and policy patterns from their field study. Reeder et al. (2008, 2011) have particularly aimed at the conflict resolution in the authorization model in laboratory experiments. However, we also need to study other complexity aspects of models that influence cognitive effort, for instance, inheritance structures in role-based models:

Q 12.1 *What are the complexity characteristics of authorization models and how do the characteristics influence policy making and authoring?*

A second area of approaches cover the model expressiveness, including recommendations on providing fine-grained controls (Sikkel and Stiemerling, 1998; Ahern et al., 2007), the handling of unexpected situations (Sikkel and Stiemerling, 1998; Bauer et al., 2009), and advanced concepts, such as authorization based on presence, logging and notifications on access, and asking for permission (Bauer et al., 2008a; Kim et al., 2010). For the integration of non-experts into the policy authoring, we need to extend these lines of research to derive insights on how the expressiveness of models interacts with complexity and usability:

Q 12.2 *How do the factors of the expressiveness of models, the complexity, and policy authoring efforts interrelate?*

The third aspect is a special case of model expressiveness, the flexibility of the model. There have been several analytical and technical publications on reducing the rigidness of policies, including delegation (Ahn et al., 2003; Johnson et al., 2009), risk-based (Cheng et al., 2007), and audit-based authorization (Cederquist et al., 2007). Field studies on flexible authorization are scarce, although policy override has been studied for health care environments (Denley and Smith, 1999; Ferreira et al., 2009). We need to extend this work and more generally understand how flexibility affects practice in different contexts:

Q 12.3 *How effective are flexible models in practice regarding security, exceptional situations, and the reduction of disruptions?*

Req 13 Comprehensible policy, and effective and efficient policy modification Of the large amount of literature on development-related authorization requirements, most are related to the usability of policies and supporting tools. One broad area are policy-editing tools, with recommendations, for instance, on simpler management tools (Whalen et al., 2006) and laboratory experiments, such as on the transition between authoring phases (Vaniea et al., 2008b). In this area, many approaches provide for the presentation and visualization of the policy (Maxion and Reeder, 2005) and consequences (Rode et al., 2006), or provide editing support, such as through assistance (Cao and Iverson, 2006).

A second category are supporting tools, for example, for role mining (Kuhlmann et al., 2003), policy analysis (Fisler et al., 2005), and assurance tools (Schaad et al., 2006). The third category relates to the policy language usability with, for instance, laboratory studies on the use of controlled natural language (Inglesant et al., 2008). For all three categories, the approaches are primarily of analytical and technical nature or are confined to laboratory experiments. For improved integration of the different perspectives on authorization, we, in addition, need to study the use of these tools in practice and particularly the interrelation between stakeholders:

Q 13.1 *How effective are editing support and other supporting tools in integrating functional, policy making, and authoring perspectives?*

Req 14 Transparent, effective, and efficient authorization mechanism Approaches on the usability of the enforcement and decisioning mechanism in authorization measures primarily focus on transparency aspects. Whalen et al. (2006) suggests based on her field study to make authorization decisions visible to improve the understanding of the actual configuration. Similarly, Johnson et al. (2009) proposes the principle of transparency for authorization and Yee (2005) to make authority transparent. In implementations, approaches either explain decisions (Kapadia et al., 2004; Becker and Nanz, 2008) or suggest responsible policy makers to turn to (Bauer et al., 2008b).

A second area of mechanism usability is the support of interactive decisions (Bauer et al., 2008b) and delegation functionality of the model through delegation assistance (Brucker et al., 2009) or reactive delegation (Bauer et al., 2008b). However, most of these approaches are analytical works only and lack the validation in practice. Moreover, the focus on implementing transparency and delegation in mechanisms results in a lack of approaches on how to handle other model functionality, in particular, further variants of flexibility, and on how to involve functional staff through adequate mechanisms:

Q 14.1 *How can the mechanism leverage the flexibility of models, foster change requests, and integrate functional staff with policy makers and authors?*

Another aspect of the mechanism implementation that has not been researched thoroughly is how resource owners build trust in the mechanism that the mechanisms actually enforce policies as defined, for instance, in third-party software products:

Q 14.2 *How can a resource owner assure the reliability of authorization mechanisms?*

Req 15 Precise and efficient integration of authorization controls A broad range of different approaches to the integration of controls have been proposed and used in practice, including OS-based (Harrison et al., 1976), language-based (Goguen and Meseguer, 1982), aspect-oriented programming (Kiczales et al., 1997), and API-based controls (Gong and Ellison, 2003). Beznosov et al. (1999) formulated the principle of decoupling authorization from business logic and the separation of the policy from the enforcement. To support the integration of controls, Jaeger et al. (2004) suggested a static analysis tool that particularly targets authorization enforcement and Priebe et al. (2004) proposed security patterns for controls integration. From the management perspective, best practices on the maintenance and review of controls (ISSEA, 2003) and technical vulnerability management (ISO/IEC 27002:2005, 2005) have been established. Lately, He and Antón (2009) showed how policy specification can be included in the software development lifecycle. However, most of the existing techniques and approaches are only based on analytical works, so that more empirical work is necessary to thoroughly understand the problems and the efficacy of mitigations, such as usable enforcement APIs and analysis tools, in practice:

Q 15.1 *How can the integration of authorization measures be supported in practical systems development?*

Extending the work by He and Antón (2009) who integrated policy definition in the systems development, we need to further study the interaction of different perspectives on authorization, for instance, how requests for policy changes by functional staff can affect the controls integration:

Q 15.2 *How can the different perspectives interact for more effective and efficient controls integration?*

6.4. Principles to address the problems

The open research question in the previous section indicate that substantial additional work is necessary to more fundamentally address problems in authorization. Hypothesis H5 states in the introduction of this dissertation that we can improve authorization by fostering the interactions and the participation between perspectives. This approach is reflected in several of the formulated research questions, for instance, on changing the policy (Q8.1), on reviews (Q7.2), and on policy/tool (Q13.1) and mechanism usability (Q14.1). We need to explore the integrative approach in practical artifacts to test the hypothesis.

While the integrative approach is the main theme of this thesis, we also formulated two further hypotheses on problem mitigation and identified a number of useful principles from security usability (Section 2.7) and agile security (Section 3.6). Consolidating the hypotheses and principles, we arrive at eight partly interrelated *Authorization Principles* to address the problems. We relate the principles to the recommendations that we derived from the feedback of the participants in the Authorization Problems Study (cf. Section 5.2). The principles are particularly aimed at authorization in organizations. In the conclusions, we will analyze how well they apply to other authorization contexts and other security measures.

P 1 *Fostering the interaction between and participation of stakeholders from different perspectives on authorization through the integration of the perspectives – H5, security usability, agile security*
In security usability, problems have been addressed by increasing the participation of the perspectives on security measures (cf. Section 2.4.3). Similarly, in secure agile development, the customer is closely integrated (cf. Section C.3). We generalize these approaches as fostering the interaction and participation[1]. One goal is to overcome the "symmetry of ignorance" between perspectives through explicit "knowledge integration" (Fischer, 1999). Secondly, we strive for "informed participation," which requires adequate "entry points" to the measure for each perspective and the measure to support the communication (Brown et al., 1993) as a "boundary object" (Star and Griesemer, 1989).

P 2 *Reducing the burden on individuals from authorization measures – Security usability*
In many areas of security usability, problems have been addressed by reducing the overall burden of affected individuals (cf. Section 2.4.1). This, for example, includes more usable configuration tools and support for security decisions. This principle also supports the principle P1 by reducing the obstacles of participation, for example, when providing decision guidance as proposed in the recommendations.

P 3 *Making the abstract aspects of authorization understandable by increasing the concreteness – Security usability*
It has been accepted in security usability that we need to create understanding in many areas of security to improve the usability (cf. Section 2.4.2). This can, for example, be achieved through training or adequate metaphors. This principle supports principle P1, because understanding the concepts is often a prerequisite for "informed" participation (cf. recommendations on expertise).

P 4 *Combining the technical and non-technical in multidisciplinary approaches – H3*
HCI has employed a broad range of non-technical research approaches to solve the problems that occur when people interact with information systems. Since research in authorization is currently still dominated by technical approaches (e.g. mechanisms and models), this principle states that we need to broaden the approach, as reflected by the range of disciplines in the recommendations.

P 5 *Taking the behavior of individuals in their social setting into account – Security usability*
Current approaches to security usability attempt to predict the individual's behavior (e.g. behavioral economics). We need to draw on cognitive psychology (e.g. heuristics and bounded rationality; Kahneman et al., 1982; Simon, 1979), but also need to consider the subjective (Pace, 2004) and social/collective experience (Razavi and Iverson, 2006). For example, we can foster informal rules to motivate acceptable circumventions (cf. recommendation on circumvention).

P 6 *Fostering the security awareness and expertise of stakeholders from different perspectives on authorization – Agile security*
In secure agile development, we found that it is important that the stakeholders in the process are aware of security risks (cf. Section C.3). Similarly, all stakeholders involved in authorization may be required to take important decisions, such as, whether to respect authorization decisions. Thus, we propose to increase the awareness and expertise of the different perspectives for an overall improved measure and, for example, provide decision guidance as proposed as part of the recommendations.

[1]An approach also known from management sciences to improve decisions, compare Miles et al. (1978).

	P1 Interaction and participation	P2 Reduced burden	P3 Concreteness	P4 Multi-disciplinary	P5 Individuals' behavior	P6 Security awareness	P7 Design for dynamics	P8 Tailor for context
Authorization framework	o	x		o			x	
—Accessible policy	x	o		o				o
—Change support	x	x	x	o				o
Policy override	x	o		o	o		x	
INDUSE	x			o			x	x
Decision support	x	x	x	o	x	x	x	x

Table 6.4.: Authorization Principles applied in the practical contributions (x: primary, o: secondary)

P 7 *Designing authorization measures for dynamic environments – H4, agile security*
Authorization contexts can be dynamic (Sinclair et al., 2008) so that we need to assume subsequent changes to permissions. Agile development accepts changing requirements as a given and optimizes the development process for that. Similarly, we should design authorization mechanisms, processes, and tools in a way to accommodate changes (cf. e.g. the recommendation on circumventions).

P 8 *Tailoring the authorization measure for the context – Agile security, security usability*
Agile methods adapt to the specific environment to achieve effective and efficient security, and usable security technology takes the context into account. Authorization measures similarly need to be tailored for the context – for example, regarding the organizational overhead, formality, and centralization (cf. recommendation on procedures), or adequate tools for the involved stakeholders.

To explore these principles and the open research questions, the following practical contributions are made in this thesis:

1. Improved authorization development (Chapter 7): An *authorization framework* that aims to address problems with authorization in three ways: (1) improve the implementation and subsequent modification of authorization enforcement in software development, (2) integrate functional, policy-making, and policy-authoring perspectives through an *accessible policy*, and (3) provide *change support* to make the policy changes more concrete.
2. More flexible authorization (Chapter 8): A *policy override* model and implementation that allows functional staff to extend their permissions temporarily within well-defined bounds.
3. Improved authorization procedures (Chapter 9): The *Integrative Authorization Development Model for Usability* (INDUSE) supports the design and evaluation of adequate processes and procedures related to authorization and explicitly emphasizes the interrelation between the perspectives on authorization.
4. Decision guidance (Chapter 10): A *decision-support* prototype and method collect, evaluate, and present the factors required to make well-founded decisions on policy changes.

The aim of the selected practical approaches is not to comprehensively solve the remaining problems with authorization entirely. Instead, the contributions explore particularly interesting problems and the usefulness of the Authorization Principles. Table 6.4 shows which of the principles are applied in which of the contributions. The principles are broadly, but heterogeneously covered. While

	Q1.1 *Adequate measure*	Q1.2 *Design support*	Q2.1 *Disruptions*	Q3.1 *Regulation in decisions*	Q3.2 *Regulation in procedures*	Q4.1 *Functional satisfaction*	Q4.2 *Management satisfaction*	Q4.3 *Satisfaction effects*	Q5.1 *Circumvention thresholds*	Q5.2 *Circumvention guidance*	Q6.1 *Decisions in practice*	Q7.1 *Effective reviews*	Q7.2 *Review interaction*	Q7.3 *Monitoring support*	Q7.4 *Monitoring consequences*	Q8.1 *Change interaction*
Authorization framework																
—Accessible policy													x			x
—Change support																o
Policy override	o		o			o	o			x					x	
INDUSE	x	x			o						x	x	x			x
Decision support	o			o		o	o			o						o

	Q9.1 *Change request problems*	Q9.2 *Foster change requests*	Q10.1 *Process design*	Q10.2 *Effective processes*	Q10.3 *Process integration*	Q11.1 *Appropriate guidance*	Q11.2 *Guidance data*	Q12.1 *Model complexity*	Q12.2 *Model expressiveness*	Q12.3 *Model flexibility*	Q13.1 *Policy interaction*	Q14.1 *Mechanism interaction*	Q14.2 *Mechanism reliability*	Q15.1 *Controls integration*	Q15.2 *Controls interaction*
Authorization framework														x	x
—Accessible policy								x	o						
—Change support											x				
Policy override										x		x			
INDUSE	x		x		x							o			x
Decision support	o	o		o		x	x								

Table 6.5.: Coverage of research questions by the artifacts (x: primary, o: secondary)

the principles of interaction and participation, reduced burden, and designing for dynamics are applied in several contributions, the principle of awareness and concreteness are more specific and covered to a lesser extent.

Table 6.5 relates the artifacts in the practical contributions to the open research questions that the contributions apply. The matrix shows a heterogeneous coverage of questions, with primary coverage for questions from policy authoring/making, strategics/tactics, and development. Secondary coverage is spread over all perspectives, although four questions lack any coverage, including questions on circumventions (Q5.1), monitoring (Q7.3), and mechanism reliability (Q14.2), and should thus be addressed in future research.

Part III.

Exploring the integrative approach to authorization usability

7. Integrating authoring, making, and development: An agile authorization framework

"It is possible," says the gatekeeper,
"but not now."

Franz Kafka: Before the Law (1915)

When authorization mechanisms are integrated into information systems to satisfy security needs, the development of the mechanism and the surrounding tools affects the effectiveness and efficiency of the entire measure (cf. the interrelation of problems in Chapter 5). It is important that the policy can be continuously adapted to improve its precision and account for the changing needs. The policy makers depend on an adequate authorization model to formulate the restrictions to be enforced, requiring expressiveness and flexibility to fulfill the security needs (Req 12, Q12.2, Q12.3). Reviews will be necessary periodically to assure that granted permissions correspond to the current needs. When policy makers suffer from a lack of transparency or comprehensibility of the policy, it is difficult to take correcting decisions (Req 14, Q14.1, Req 13).

Similarly, policy authors who technically formulate the restrictions must understand the policy before making changes. Particularly for dynamic organizations without formal processes, it is necessary to negotiate the adequate formulation of the restrictions between policy makers and authors. In these contexts, it may be of help to employ the policy as the basis for the discussions (Q8.1). When actually conducting the changes, the efficiency of the change can be impacted by complex or incomprehensible authorization models (Req 12, Q12.1).

System developers work on a lower level but are not independent of the policy making and authoring. They integrate the controls in the system to enforce the policy when applications are developed or integrated in existing authorization infrastructure (Req 15, Q15.1). The developers' tasks interrelate with those of policy makers and authors (Q15.2), since, for instance, when the necessary control is missing in the application control flow, restrictions in the policy will not be enforced. Concerning the enforcement, the primary problem for developers is the reliable and efficient integration of the enforcement controls, otherwise rendering the mechanism ineffective, as, for example, found by Jaeger et al. (2004) in Linux kernel modules, or caused by unanticipated control flow for architectural enforcement measures (Sohr and Berger, 2010). This can be particularly challenging with continuous changes to the system or restrictions.

To address these problems, the authorization framework `declarative_authorization` aims to integrate the perspectives of the policy making, authoring, and development to arrive at mechanisms that enable an efficient and effective authorization measure (P1). The framework provides the technical basis for the stakeholders' interaction: For policy makers, an accessible policy and a supporting tool for policy changes should enable a greater participation in policy authoring and reduce the policy makers' and authors' burden (P2). The supporting tool particularly increases the concreteness of policy changes, leading from concrete change goals to respective change operations (P3). Moreover, the increased participation and improved comprehensibility should enable more adequate change procedures and the decentralization of policy changes, if appropriate for the authorization context (P8).

When integrating authorization controls in systems, developers benefit from the interaction with policy makers and authors (P1). The close integration of the policy specification with development should lead to more effective and efficient implementation of enforcement controls (He and Antón, 2009), since developers can better react to changing requirements. Developers further benefit from an adequate framework through reduced effort for the integration of enforcement (P2). The multi-disciplinary approach of software engineering and the socio-technical should result in an effective and efficient controls API (P4). The API is particularly designed to support dynamics in software development and cope with system and policy changes through a consequent separation of concerns between the policy and the enforcement (P7).

Overall, the authorization framework strives to support stakeholders from the development, policy-authoring, and policy-making perspectives. This chapter[1] describes the three primary approaches that the framework takes. The chapter first describes the authorization-policy side, including the underlying model, the policy language, and the tools that increase its comprehensibility. Further supporting the policy making and authoring, the change-support tool is subsequently presented. We discuss the specific strengths and weaknesses of textual and graphical representations and of change support based on formative evaluations with policy makers and authors. The third main section covers the integration of enforcement controls and the evaluation of the framework's ability to support this task.

7.1. Towards a comprehensible authorization policy and model

Policy makers decide on what restrictions to enact in a system and policy authors technically formulate those restrictions in policies. A usable, that is, effective and efficient authorization model can support these tasks, increase the comprehensibility, reduce the overall burden, and improve the interaction of stakeholders. For this reason, the `declarative_authorization` framework aims to provide the following contributions, detailed in the following:

- A simple, yet maintainable authorization model that increases the effectiveness and efficiency of policy changes,
- A readable policy language that reduces the thresholds for working with the policies and supports the participation of functional users,
- A graphical representation of the authorization policy to offer a more abstract, but in some contexts more appropriate representation of the policy,
- A testing framework to continuously assure that the policy conforms to the original intentions.

7.1.1. Authorization model

The authorization model governs the structures and the expressiveness that are available to formulate restrictions to enforce by the authorization mechanism (cf. Chapter 4). While not as thoroughly researched as editing tools for authorization policies, authorization models have still received a fair amount of attention in security-usability research. For models of privacy policies, Reeder et al. (2007) identified five usability challenges: Policy authors need to be able to efficiently group objects,

[1]Parts of this chapter are based on publications in the proceedings on CollaborateCom 2008 (Bartsch et al., 2009, on the authorization framework), ARES 2010 (Bartsch, 2010a, change support), and ESSoS 2011 (Bartsch, 2011a, enforcement study), which have been extensively revised for inclusion here.

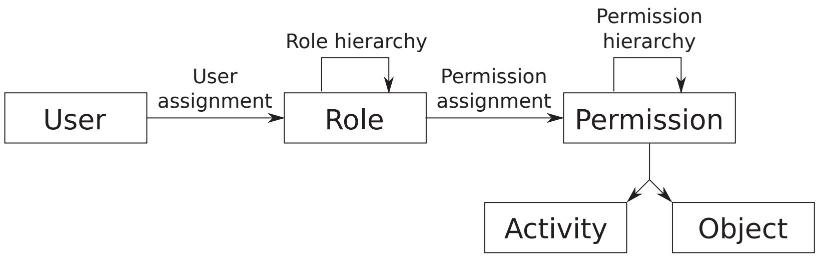


Figure 7.1.: Role-based access control model

use consistent terminology, understand the default rules, uphold the policy structure, and solve rule conflicts. In case of role-based policies, Brostoff et al. (2005) found that the primary challenges for policy authors lie in understanding the policy structure and the overall authorization paradigm. Inglesant et al. (2008) describe an iterative process of policy development that is necessary for users to understand the model.

Further research studied what constructs policy makers actually require (Sikkel and Stiemerling, 1998). For instance, for physical security Bauer et al. (2008a), identified policy operators, such as logging and notifications of access. For home security, Kim et al. (2010) additionally found "asking for permissions" and presence as authorization concepts. Generally, the model expressiveness must meet the demands of the specific context, often requiring fine-grained controls (Ahern et al., 2007) and flexibility (Bauer et al., 2008a). Model expressiveness can be in conflict with the comprehensibility when increasing complexity, and thus needs to balanced for the specific context. Smetters and Good (2009) argued, for instance, for simpler, pattern-based models. Reeder et al. (2011) focused on conflict resolution techniques in authorization models and found in their study that participants understood specificity-based resolution best.

The authorization model of `declarative_authorization` continues this research strand and aims for simple, yet maintainable and flexible policies. The model is similar to the well-known and wide-spread Role-based Access Control model (RBAC; Ferraiolo and Kuhn, 1992), shown in a basic variant in Figure 7.1. From left to right, the RBAC model assigns each user to a number of roles, which, for instance, represent job functions in organizations. Roles, in turn, group permissions that define the ways in which users of the role (e.g. with the specific position) are allowed to interact with the system. Permissions consist of a specific activity that a user may carry out and a specific object to which the permission applies. A role hierarchy allows a role to include permissions of a child role. Similarly, permissions may inherit from child permissions.

The `declarative_authorization` model is based on the RBAC model and is depicted in Figure 7.2. Instead of composing permissions of object and activity, the framework combines object types (contexts) and activities with constraints to increase the maintainability, for example, "allow access to all conferences that the user is signed up for." Permissions on individual objects are realized through context authorization constraints. Authorization constraints are a common concept for RBAC models to increase the policy flexibility (Bertino et al., 1999). In case of `declarative_authorization`, constraints act upon a specific context object that the authorization should be decided for. For example, in a conference management application, when deciding whether a user is allowed to access a specific conference, a constraint could check that the conference is already published.

An example policy for a simple conference management Web application that conforms to the authorization model, but for brevity forgoes the user assignments, is shown in Figure 7.3. Three roles are displayed in ellipses, a *Guest* role that all not-logged-in users are assigned to and should

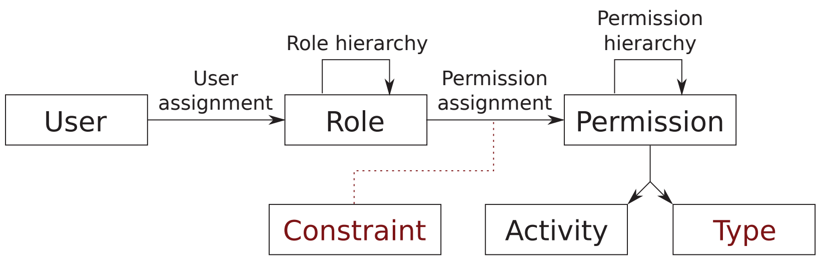


Figure 7.2.: `declarative_authorization` authorization model

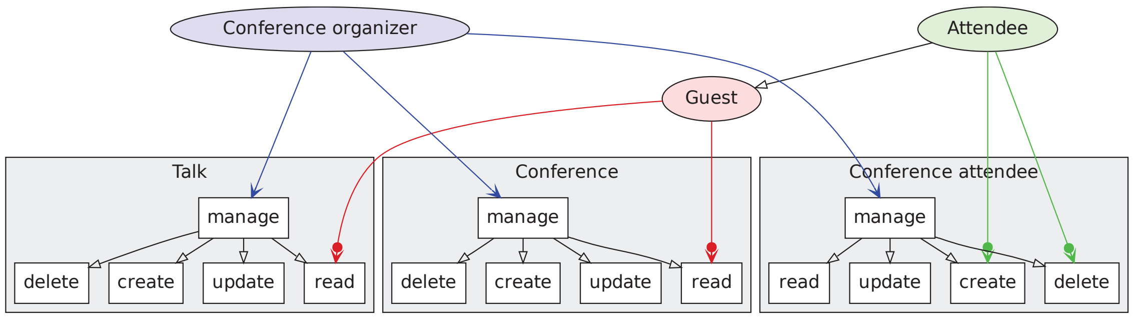


Figure 7.3.: Example authorization policy

only allow limited viewing of conference content, an *Attendee* role for users that would like to attend conferences and a *Conference organizer* role for administrative staff. Through role inheritance, the attendees have the same permissions as guest users, indicated by the edge between Attendee and Guest. The larger rectangles represent object types that privileges should apply to. Activities that act on objects of the different types are shown inside the type rectangles and constitute in association with the type the permissions. Permissions are assigned to roles as shown in edges between roles and permissions. For example, a guest user may read conferences, while conference organizers may manage conferences. There also is inheritance among permissions, indicated through edges between permissions, so that the "manage" permissions include "create", "read", "update", and "delete".

7.1.2. Policy language

The general policy structure as defined by the authorization model informs the representation of the policies. General goals for policy representations are to provide for the documentation of restrictions and support a common understanding as part of the communications of the policy-making and authoring stakeholders (Bauer et al., 2009). Languages for authorization policy specification are usually defined as domain-specific languages (DSL; Consel and Marlet, 1998), either in textual or graphical forms. For textual policies, which we focus on in this section, Inglesant et al. (2008) proposed "controlled natural language" to reduce the distance between the author and the policy definition (cf. Pane et al., 2001). For the usability of an authorization language, the distance between the mental model of humans and the syntax is important. Adequate representations depend on the context – for example, less detailed and complex, but more comprehensible representations of authorization policies can lower the threshold, support the communication about, and the development of policies. Generally, policy development should offer the appropriate level of detail to each of the stakeholders (Dai and Alves-Foss, 2002).

The `declarative_authorization` policy language follows these aims to make policies compre-

Listing 7.1: Example authorization rules

```
1  authorization do
2    role :guest do
3      has_permission_on :conferences, :to => :read do
4        if_attribute :published => true
5      end
6      has_permission_on :talks, :to => :read do
7        if_permitted_to :read, :conference
8      end
9    end
10
11   role :attendee do
12     includes :guest
13     has_permission_on :conference_attendees, :to => :create do
14       if_permitted_to :read, :conference
15     end
16     has_permission_on :conference_attendees, :to => :delete do
17       if_attribute :attendee => is { user }
18     end
19   end
20
21   role :conference_organizer do
22     has_permission_on :conferences, :to => :manage
23     has_permission_on :conference_attendees, :to => :manage
24     has_permission_on :talks, :to => :manage
25   end
26 end
27
28 privileges do
29   privilege :manage, :includes => [:create, :read, :update, :delete]
30 end
```

hensible and support discussions on policies while enabling the formulation of policies that adhere to the framework's authorization model. The authorization policy is formulated from the perspective of the roles, to which the allowed privileges (activities) on contexts (object types) are assigned. The policy language – while based on the syntax of the Ruby programming language (Flanagan and Matsumoto, 2008) – is aimed to be a human-readable and intuitively comprehensible textual DSL. Similar to a controlled natural language (cf. Inglesant et al., 2008), the language's syntax is derived from natural language and can be read in form of sentences, for example: "Role *conference organizer* has permissions on *conferences* to *manage*" in the following listing:

```
role :conference_organizer do
  has_permission_on :conferences, :to => :manage
end
```

The more comprehensive example in Listing 7.1 shows the authorization rules for the diagram in the previous section (Figure 7.3). Two types of authorization constraints are present in the example: `if_attribute` constraints restrict permissions through conditions on the attributes of a specific object. One example in line 4 restricts guests to only access those conferences that are marked "published." Similarly, attendees should only be able to unregister *themselves* from conferences, thus the rule in line 17 only allows attendees to delete a *Conference attendee* object (i.e. withdraw their attendance) if the object is associated through the attribute *attendee* to the current user. The attribute conditions may be nested, so that, for example, it is possible to allow attendees to unregister all users from their company:

```
has_permission_on :conference_attendees, :to => :delete do
  if_attribute :attendee => {:company => is { user.company } }
end
```

The expression in curly brackets is evaluated at the time of the authorization decision so that dynamic contextual information, such as the current user's company in this example, can be compared to attributes. To formulate the conditions on attributes, the policy language offers a number of operators apart from the equality operator `is` in this example. Other operators are derived from set theory, such as `contains` and `intersects_with`, or compare values, such as `greater_than`.

The second form of constraints are `if_permitted_to` rules that impose restrictions based on the current user's permissions regarding associated objects. In Listing 7.1, guests may only read talks if they are also allowed to read the associated conference (line 7). The attribute reference may be nested similar to the attribute rules.

The authorization constraints are formulated in DSL statements instead of direct Ruby expressions to more flexibly employ the constraints. The constraint semantics allow not only to decide specific authorization requests, but also to derive the obligations under which a user is permitted to access an object type. This ability is used to generate database query conditions – for example, only query for conferences that the current user may read.

Symbols beginning with colons (`:guest`), the block delimiters `do`/`end`, and hash associations (`=>`) are visible indications that the policy language is based on Ruby syntax. There are three reasons for using Ruby syntax. First, Ruby's metaprogramming features facilitate the implementation of a simple and readable DSL. Second, the robust Ruby parser can be used to interpret the policy definition and generate rich error messages. Third, the `declarative_authorization` framework itself is implemented in Ruby, facilitating the integration of the Ruby DSL parsing. However, since the policy is internally transformed into a tree structure, applications based on other platforms could equally well employ the policy language, for example, through appropriate Ruby bindings.

7.1.3. Graphical representation

To further improve the comprehensibility of the authorization policy, the framework can generate a graphical representation of the policy. Such a diagram (shown for the example policy in Figure 7.3) reduces the cognitive effort for policy authors to understand the interrelations of roles and permissions. The representation is more abstract than the textual policy and, thus, conveys a reduced amount of information – for example, it only indicates the presence of authorization constraints through dots on role–permission assignments, but not the actual constraints. On the other hand, this abstraction allows policy authors to focus on more essential aspects of the policy. In the example, the inheritance of guest permissions by attendees and its consequences are clearly visible. An important second advantage of the graphical representation is that it may lower the threshold for non-technical users to discuss the policy.

Despite the abstraction from the complete policy, larger policies may result in overloaded diagrams with excessive numbers of nodes and edges. To counter overloaded diagrams, the graphical representation allows to refine the diagram generation. First, users can "drill down" and limit the diagram to specific roles and object types. Second, the user may choose to display the effective permissions of the roles instead of only directly assigned ones, adding inherited permissions to the diagram. Third, the user can toggle whether all permissions should be displayed with the full permission hierarchy or only those that are assigned.

```
# setup specific objects: an_attendee, a_talk
the_user an_attendee do
    should_be_allowed_to      :read,   :conferences
    should_not_be_allowed_to  :update, :conferences
    should_be_allowed_to      :read,   a_talk
    should_not_be_allowed_to  :update, a_talk
end
```

Figure 7.4.: Example of an authorization policy test case

7.1.4. Policy testing

The complexity of the policies can make it difficult to grasp the multitude of side effects that an individual change has. For authorization policies, formal analysis of access control policies can guarantee that properties of the policy remain intact (Dougherty et al., 2006; Li and Tripunitara, 2006), including the identification of conflicts and suboptimal structures. Further approaches for property validation are model checking – for example, based on the analysis of UML models (Schaad et al., 2006; Sohr et al., 2008). In model checking, all possible states are checked for whether certain properties always hold.

In software engineering, a well-known assurance approach is the dynamic analysis (cf. Section A.3). Instead of static analysis, which is solely based on the program text, dynamic analyses examine executed artifacts, for example, through automated test cases. Carrying over testing from software engineering to authorization-policy development can have similar positive effects of detecting unintended side effects of modifications (Hwang et al., 2008). The `declarative_authorization` framework offers a testing framework based on the Ruby programming language to enable readable test cases. A set of example policy tests are shown in Figure 7.4. For a specific user, *positive* ("should be allowed") and *negative* ("should not be allowed") permission tests are specified. The test cases either apply to general object types, for example, all `:conferences`, or specific objects, such as a specific talk.

The readability of the test cases is an important aspect to tightly integrate the functional perspective in the development of authorization policies. In contrast to the authorization policy, which contains abstract rules, tests focus on specific cases. While abstract rules in policies can be difficult to understand for non-technical stakeholders (cf. Blackwell et al., 2008), these example-like test cases may be closer to the mental model of functional stakeholders. The test cases may thus also serve as additional documentation and basis for discussions, complementing the authorization policy.

7.2. Change support for concrete policy-authoring

One problem with the management of policies can be that the rules and structures are rather abstract, depending on the authorization model. For a role-based policy, permissions cannot be directly assigned, but require the policy author to transfer the concrete change goals ("Person X needs to be able to do Y") into changes to the abstract user–role and role–permission assignments. Generally, tool support for the creation of policies has been studied extensively in prior work. Zurko et al. (1999) developed a policy editor for policies similar to RBAC based on usability testing and user-centered design, which allowed novice users to produce meaningful results in under one hour. Herzog and Shahmehri (2006) studied the usability of the Java policy tool and criticized that help on semantics was missing and that the tool promotes lax policies. Reeder et al. (2008) found that the formulation of specific and effective permissions in "expandable grids" vastly improves the visualization and

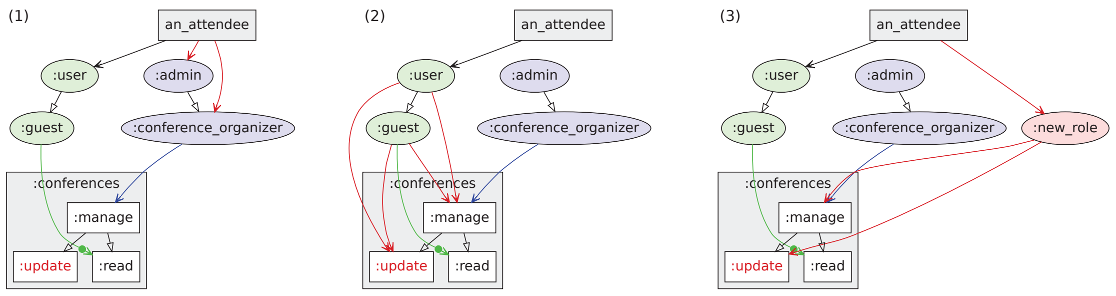


Figure 7.5.: Change options for granting *Modify* (“update”) to the attendee (“an_attendee”)

authoring of policies when compared to abstract rule formulation.

However, typical editing tools do not achieve an increased concreteness. Instead, the concreteness usually remains on the level of the model. When the models are concrete, policy authors need to directly assign permissions to users. The result is a high number of rules that is difficult to maintain with the continuous changes to policies caused by organizational changes. Therefore, it is not only necessary to improve the user interface on a superficial level, but also to address the underlying complexities. While originally introduced to improve the policy’s maintainability, RBAC, as a typical authorization model, inhibits a complexity that, conversely, often leads to deteriorating policies (Bertino et al., 2008). Moreover, the abstractions of RBAC reduce the policy’s comprehensibility. As pointed out earlier, Blackwell et al. (2008) generally argue that forcing computational thinking on stakeholders can reduce the overall quality of the result, since important non-computational confutations are lost. Similarly, the abstract nature of the RBAC model encumbers effective participation (P1).

To overcome the abstractness of typical authorization models, Cao and Iverson (2006) suggested the intentional access management (IAM) as a design principle that supports the policy decisions by deriving changes from the stakeholders’ expressed intentions. The IAM model is based on the idea that the intentions of stakeholders are collected and evaluated by a “consensus model” to remove inconsistencies and conflicts. From the consensus model, policies for different authorization models, such as ACL and RBAC, are derived. The IAM prototype implementation targets WebDAV authorization and is only geared for comparatively simple ACLs. The IAM algorithm follows a simple flow chart of seven questions to derive the policy changes, such as “Does group X3 exist such that X3 is denied privilege Y to object Z and I have the privilege to modify group X3?”. Consequently, IAM cannot present change suggestions that require to take multiple intentions into account, such as, “Grant privilege X to user Y, but not to user Z”.

The Change Assistant – developed in parallel to IAM – extends the ideas of IAM with more general change goals. In contrast to IAM, the Change Assistant focuses on helping a policy author to choose from the multitude of options to modify a policy for given goals. The following example policy for a simple conference-management Web-application illustrates the range of options. In the system, a user with an *Attendee* role should receive permissions to modify conferences. The role *Attendee* includes the role *Guest*, the privilege *Manage* includes *Modify*. As depicted in Figure 7.5, the following options are available to reach the desired modification:

1. Assign an existing role to the user that has the manage or modify privilege (*Conference organizer*, *Administrator*),

2. Add privileges *Modify* or *Manage* to roles *Attendee* or *Guest*,

3. Add privileges *Modify* or *Manage* to a new role for users with conference-modification permissions and assign the role to the user.

Even in this simple example with only four roles, there are already eight possible modifications of the policy. Security experts may intuitively rule out some options in this simple example, but might need support in more complex cases to choose the best-suited option. When working closely together with functional staff, even explaining an existing policy can be difficult. Thus, finding and keeping up with the wealth of modification options will overwhelm functional staff, who prefer simple and more specific but often unmaintainable access control lists.

Irrespective of the individual's perspective, the effort for policy modifications is significant (Gallaher et al., 2002) and building the role model is stated to be the most expensive part of introducing RBAC (Bertino et al., 2008). One option to reduce the effort of policy modifications are supporting tools. According to Kern et al. (2002), the entire role lifecycle needs to be supported with change management tools. The purpose of the Change Assistant is to support policy changes comprehensively and concretely. This section describes the algorithm and tool that suggest change options for specific change goals and support stakeholders to choose between the change options, taking the side effects into account. An artificial-intelligence algorithm is employed to generate the possible sequences of policy modifications to reach a given goal. The modification goal is specified by test cases for authorization rules (cf. Section 7.1.4). To support domain as well as security experts in choosing the most suitable modification for the specific goal, a prototype GUI facilitates the usage of the Change Assistant.

7.2.1. The Change Assistant algorithm

To suggest possible modifications to achieve a goal and to show their consequences, the Change Assistant algorithm offers change support. In practice, there is a limited number of atomic modification operations that may be applied to the authorization rules. Kern et al. (2002) define role management as creating new roles, users, or permissions as well as the connection/disconnection of users and roles or roles and permissions. In the Change Assistant algorithm, the following operations are employed. For removing permissions:

- Remove role assignments from users,
- Remove permissions from roles,
- Modify role or privilege hierarchies.

For extending permissions:

- Assign roles to users,
- Assign permissions to roles,
- Modify role or privilege hierarchies,
- Introduce new roles.

The overall goal of the modifications is specified in test cases, described in Section 7.1.4, of users' allowed and disallowed actions. The general idea of the Change Assistant is to combine the change operations into sequences of modifications so that the sequences fulfill all the given test cases.

$$\begin{aligned}
\mathit{Candidate} &: \mathit{AuthRules} \times \mathcal{P}(\mathit{User}) \times \mathit{Modifications}\\
\mathit{AuthRules} &: \mathcal{P}(\mathit{Role} \times \mathit{Privilege} \times \mathit{Type})\\
\mathit{Role} &: \text{Set of all system roles}\\
\mathit{Privilege} &: \{\mathsf{create}, \mathsf{update}, \ldots\}\\
\mathit{Type} &: \text{Set of all system object types}\\
\mathit{User} &: \text{Set of all system users}\\
\mathit{Modifications} &: \mathit{SpecificOperation}^*\\
\mathit{SpecificOperation} &: \mathit{AbstractOperation} \times \mathit{Parameters}\\
\mathit{AbstractOperation} &: \text{Set of abstract operations (Sect. 7.2.1)}\\
\mathit{Parameters} &: \text{Parameters of abstract operations (Sect. 7.2.1)}\\
\mathit{TestCase} &: \{\mathsf{allow}, \mathsf{prohibit}\} \times \mathit{User} \times \mathit{Privilege} \times \mathit{Type}
\end{aligned}$$

Figure 7.6.: Types used in the Change Assistant algorithm

The Change Assistant algorithm is based on the well-known best-first search algorithm A* and planning algorithms from AI (Russell and Norvig, 2003). The main algorithm for the Change Assistant is shown in Figure 7.7, the types in the algorithm are given in Figure 7.6. The algorithm works as follows: Beginning with an initial *candidate* derived from the original state, the given *test cases* are solved by generating new candidates with modifications. Each modification is derived in a two-step process: First, the algorithm chooses general modification options, *abstract operations*, that match the first failed test. Then, each abstract operation is expanded into *specific operations* (*generate_specifics_for*): From an abstract operation (e.g. "Remove a Permission from a Role") specific operations are derived that solve the failing test (e.g. "Remove Permission *Read* from Role *Organizer*"). The abstract operations' expansions are listed below. Apart from atomic abstract operations, compound operations are employed that combine multiple operations to guarantee solving one test in each step. Compound operations significantly reduce the number of branches in the decision tree.

Specific operations are applied to previous candidates to arrive at new candidates. The validity of each new candidate is tested according to three properties:

- The candidate must not reverse any of the previous operations,
- The candidate must not contain any operation that the end-user has prohibited (cf. following section),
- The candidate is rejected if it is a superset of an already known valid solution, that is, the candidate only adds further, unnecessary operations.

If the new candidate has no more failing tests, it is added to the list of valid solutions. Otherwise, it is added to the list of candidates for further modifications in subsequent steps. A heuristic function is used to sort the remaining candidates (Russell and Norvig, 2003), so that the most promising candidates are considered first. The heuristic function is based on the remaining failing tests and the invasiveness of earlier changes, defined for a given candidate c as:

$$f(c) = |\text{failing\_tests}(c)| + \sum_{i=1}^{|\text{steps}(c)|} \text{step\_weight}(c, i)$$

The weight of a step (invasiveness) is assigned individually to the abstract operation.

```
Input: candidates = [[current_auth_rules, users, $\emptyset$]],
  non-empty(test_cases)
Output: non-empty(solutions)
  while non-empty(candidates) and steps < max_steps do
    candidate ← remove-first(candidates)
    if is_positive(last_failing_test(candidate)) then
      abstract_operations ← positive_abstract_operations
    else
      abstract_operations ← negative_abstract_operations
    end if
    for abstract_operation in abstract_operations do
      specific_operations ← generate_specifics_for(candidate, abstract_operation)
      for specific_operation in specific_operations do
        if specific_operation is prohibited by user or reversal of previous operation
            or superset of known solution then
          next
        end if
        new_cand ← apply(candidate, specific_operation)
        check(new_cand, test_cases)
        if empty(failing_tests(new_cand)) then
          append(solutions, new_cand)
        else
          append(candidates, new_cand)
        end if
      end for
    end for
    candidates ← sort_by_heuristic(candidates)
  end while
```

Figure 7.7.: Change Assistant algorithm

The runtime effort of the Change Assistant algorithm depends on the number of test cases. Each test case results in one branching of the decision tree. Negative tests have a static branching factor of two with only two different operations available. For positive tests, the branching factor depends on the existing authorization rules: the number of roles as well as the inheritance levels in privileges and roles. To guarantee the termination of the algorithm, a maximum count of considered candidates is enforced. With the best candidates being considered first, it is sufficient to search only part of the solution space.

The current implementation of the Change Assistant algorithm is aimed at the framework's authorization model. Since the authorization model is similar to the standard RBAC model, the proposed algorithm should be applicable with minor modifications to other RBAC models.

Abstract operations

As described above, test cases are either positive ("should be allowed") or negative ("should not be allowed"). For both types of test cases, there are abstract operations that have the according effect for solving the failing test. From abstract operations, specific operations are derived, defining the specific modifications to apply to the authorization rules. First, abstract operations and derived specific operations for positive tests are described, followed by those for negative tests. For each abstract operation, the parameters are formally described that define the specific modifications. The following functions are used in the descriptions:

$$\begin{aligned}
SelfOrDescendants_R &: \mathcal{P}(Role) \to \mathcal{P}(Role) \\
SelfOrAncestors_R &: \mathcal{P}(Role) \to \mathcal{P}(Role) \\
SelfOrDescendants_P &: \mathcal{P}(Privilege) \to \mathcal{P}(Privilege) \\
SelfOrAncestors_P &: \mathcal{P}(Privilege) \to \mathcal{P}(Privilege) \\
AssignedRoles &: User \to \mathcal{P}(Role) \\
RolesSatisfy &: Privilege \to \mathcal{P}(Role) \\
RolesPrivileges &: \mathcal{P}(Role) \to \mathcal{P}(Privilege)
\end{aligned}$$

Operations to solve positive tests

Assign Privilege to Role This operation needs to consider which privileges are to be assigned to which roles. All the roles assigned to the user in the current test case and descendant[2] roles may be used. As privilege, the privilege in the test case and its ancestors are considered. Put formally:

$$\begin{aligned}
Parameters_{APR} &: User \times Privilege \to \mathcal{P}(Role) \times \mathcal{P}(Privilege) \\
Parameters_{APR}(\mathsf{user}, \mathsf{privilege}) &= SelfOrDescendants_R(AssignedRoles(\mathsf{user})) \times \\
&\quad SelfOrAncestors_P(\mathsf{privilege})
\end{aligned}$$

Assign Role to User The operation selects existing roles and assigns one to the user in the test case. Only roles are considered that include the test case's privilege or its ancestors. In addition, all

[2]Ancestors include the role or privilege, descendants are included. For instance, the role *Guest* is often a descendant of *User*, *User* an ancestor of *Guest*.

ancestor roles can be employed.

$$\begin{aligned}
\mathit{Parameters}_{ARU} &: \mathit{User} \times \mathit{Privilege} \to \mathcal{P}(\mathit{Role})\\
\mathit{Parameters}_{ARU}(\mathsf{user},\mathsf{privilege}) &= \mathit{SelfOrAncestors}_R(\mathit{RolesSatisfy}(\mathsf{privilege}))
\end{aligned}$$

Create Role and Assign Role to User This is a compound operation that creates a role, assigns a privilege to the role and then assigns the new role to the user in the test case. The operation has to select the privileges to assign to the new role. The privilege is chosen from the one in the test case and all ancestor privileges.

$$\begin{aligned}
\mathit{Parameters}_{CRARU} &: \mathit{User} \times \mathit{Privilege} \to \mathcal{P}(\mathit{Privilege})\\
\mathit{Parameters}_{CRARU}(\mathsf{user},\mathsf{privilege}) &= \mathit{SelfOrAncestors}_P(\mathsf{privilege})
\end{aligned}$$

Assign Privilege to Role and Assign Role to User For this compound operation, one of the present roles is assigned to the user in the test case. To solve the failing test, the operation also assigns a privilege to the role. As privileges, the privilege in the test case and its ancestors may be used. All roles are considered that are not yet assigned to the user and that together with their descendants do not satisfy the privilege.

$$\begin{aligned}
\mathit{Parameters}_{APRARU} &: \mathit{User} \times \mathit{Privilege} \to \mathcal{P}(\mathit{Role}) \times \mathcal{P}(\mathit{Privilege})\\
\mathit{Parameters}_{APRARU}(\mathsf{user},\mathsf{privilege}) &= (\mathit{Roles} - \mathit{AssignedRoles}(\mathsf{user}) - \mathit{RolesSatisfy}(\mathsf{privilege})) \times\\
&\quad \mathit{SelfOrAncestors}_P(\mathsf{privilege})
\end{aligned}$$

Operations to solve negative tests

Remove Privilege from Role This operation identifies the privilege to remove and the role to remove it from to solve the failing test. It searches all roles of the user in the test case and their descendants for the test's privilege or one of its ancestors. If the user has the privilege through multiple roles or privileges, this operation removes all of these.

$$\begin{aligned}
\mathit{Parameters}_{RPR} &: \mathit{User} \times \mathit{Privilege} \to \mathcal{P}(\mathit{Role}) \times \mathcal{P}(\mathit{Privilege})\\
\mathit{Parameters}_{RPR}(\mathsf{user},\mathsf{privilege}) &= (\mathit{AssignedRoles}(\mathsf{user}) \cap \mathit{RolesSatisfy}(\mathsf{privilege})) \times\\
&\quad (\mathit{SelfOrAncestors}_P(\mathsf{privilege}) \cap\\
&\quad\quad \mathit{RolesPrivileges}(\mathit{AssignedRoles}(\mathsf{user}) \cap \mathit{RolesSatisfy}(\mathsf{privilege})))
\end{aligned}$$

Remove Role from User The failing test is solved by removing the role(s) from the test's user that satisfy the privilege of the test. The operation identifies all roles of the user that have, directly or through a descendant, the privilege or an ancestor of the privilege assigned.

$$\begin{aligned}
\mathit{Parameters}_{RRU} &: \mathit{User} \times \mathit{Privilege} \to \mathcal{P}(\mathit{Role})\\
\mathit{Parameters}_{RRU}(\mathsf{user},\mathsf{privilege}) &= \mathit{AssignedRoles}(\mathsf{user}) \cap \mathit{RolesSatisfy}(\mathsf{privilege})
\end{aligned}$$

Examples

To demonstrate how the Change Assistant algorithm works, we come back to the example above. The user *Attendee 1* with the role *Attendee* should receive the permissions to modify conferences. This goal is formulated as a policy test ("Attendee can modify conferences" – cf. Section 7.1.4). As

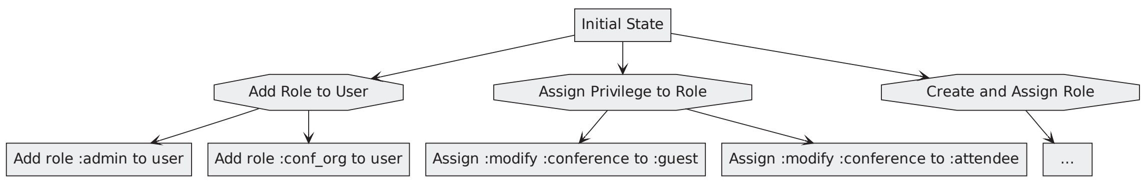

Initial State
Add Role to User
Assign Privilege to Role
Create and Assign Role
Add role :admin to user
Add role :conf_org to user
Assign :modify :conference to :guest
Assign :modify :conference to :attendee
...

Figure 7.8.: Simple Change Assistant example

depicted in Figure 7.8, the algorithm starts out with the original policy. This is the initial candidate. When this candidate is checked against the goal test, the test fails. To solve this failure, the abstract operations for positive test cases are considered. In Figure 7.8, abstract operations are shown as octagons.

Each of the chosen abstract operations is instantiated for the failing tests into specific operations, shown in Figure 7.8 in the lower row. Specific operations are validated as described in the previous section. Each valid specific operation is then applied to the initial candidate to create new candidates. The new candidates are checked against the policy tests. With all of the test cases succeeding on the generated candidates, no further operations are necessary and the identified approaches may be presented to the user.

In a more complex example, the intended policy change may be indicated through several test cases. While there might be one attendee who should be able to modify conferences, maybe others should not. In this case, adding the role *Conference Organizer* to Attendee 1 still solves all policy test cases in one step. However, adding the permission to the attendee role would, for example, require removing this role from the other users. Thus, for each candidate with remaining failing test cases, suitable abstract and specific operations are generated and applied as described above. In this way, the algorithm creates valid solutions with multiple modification steps.

7.2.2. The Change Assistant user interface

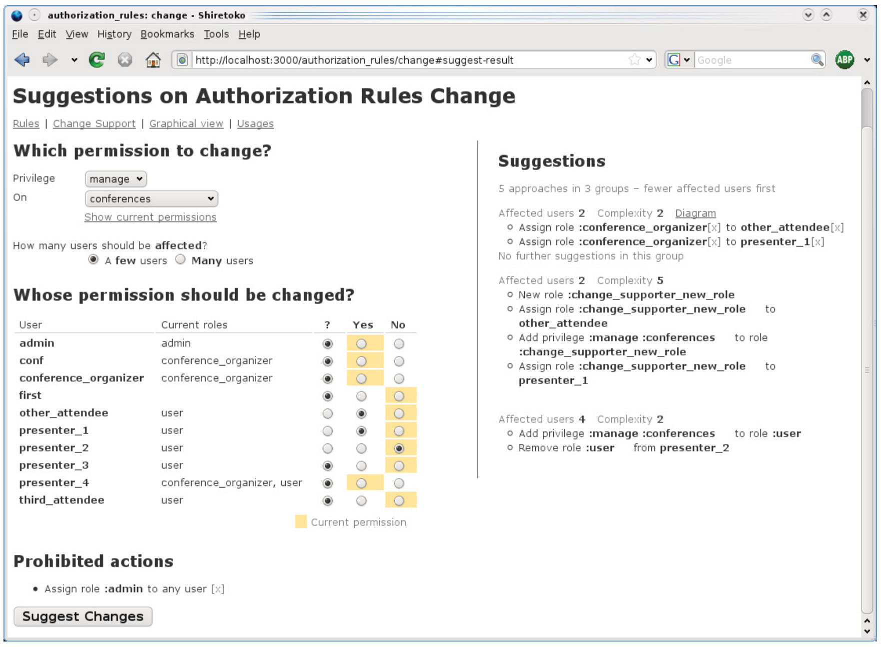
The above-described Change Assistant algorithm takes existing authorization rules and goal tests that specify the intended changes as inputs and derives several possible suggestions. To formulate the conditions as test cases and evaluate the resulting suggestions, functional stakeholders need additional support that a graphical user interface (GUI) provides.

We integrated the Change Assistant GUI into the authorization framework. The Change Assistant user interface adheres to the principles of *Programming by Example* (Myers et al., 2000; Lieberman, 2001). The GUI is integrated into the Web application as part of the authorization management backend. In Figure 7.9, the GUI is depicted as part of a demo application for managing conferences[3]. The GUI works in four steps:

1. Choosing the permission that should be changed, such as “creating conferences” in the demo application. Although the Change Assistant algorithm supports the mixing of different permissions in the test cases, the GUI currently limits the tests to one permission to improve the comprehensibility of the interface.

2. Answering basic questions to improve the result by taking the intentions of the stakeholders into account. Currently, the stakeholders are queried whether they aim to affect only a few or many system users. Choosing one of the two options causes a higher weight on the affected user count in the evaluation of the solution.

[3] `declarative_authorization` Demo Application at `http://github.com/stffn/decl_auth_demo_app`


authorization_rules: change - Shiretoko
File Edit View History Bookmarks Tools Help
http://localhost:3000/authorization_rules/change#suggest-result
Google

Suggestions on Authorization Rules Change

Rules | Change Support | Graphical view | Usages

Which permission to change?

Privilege: manage
On: conferences
Show current permissions

How many users should be **affected**?
[✓ A **few** users] **Many** users

Whose permission should be changed?

User	Current roles	?	Yes	No
admin	admin	●	○ 🟨	○
conf	conference_organizer	●	○ 🟨	○
conference_organizer	conference_organizer	●	○ 🟨	○
first		●	○	○ 🟨
other_attendee	user	○	●	○ 🟨
presenter_1	user	○	●	○ 🟨
presenter_2	user	○	○	● 🟨
presenter_3	user	●	○	○ 🟨
presenter_4	conference_organizer, user	●	○ 🟨	○
third_attendee	user	●	○	○ 🟨

🟨 Current permission

Prohibited actions

- Assign role **:admin** to any user [x]

Suggest Changes

Suggestions

5 approaches in 3 groups – fewer affected users first

Affected users **2** Complexity **2** Diagram
- Assign role **:conference_organizer**[x] to **other_attendee**[x]
- Assign role **:conference_organizer**[x] to **presenter_1**[x]

No further suggestions in this group

Affected users **2** Complexity **5**
- New role **:change_supporter_new_role**
- Assign role **:change_supporter_new_role** to **other_attendee**
- Add privilege **:manage :conferences** to role **:change_supporter_new_role**
- Assign role **:change_supporter_new_role** to **presenter_1**

Affected users **4** Complexity **2**
- Add privilege **:manage :conferences** to role **:user**
- Remove role **:user** from **presenter_2**

Figure 7.9.: Change Assistant GUI

3. Defining the test cases by deciding which system users should or should not have the above-chosen permission. Yellow markers show for the users whether they currently have the permission. Typically, stakeholders define permission tests for several users, both positive and negative, to improve the precision of the algorithm suggestions. It also helps to indicate for which users the permissions should remain unchanged.

4. Requesting, reviewing and refining the resulting suggestions. The suggestions are listed, ordered by the complexity of the changes and the number of affected users. Similar results are grouped if the same abstract operations are used. A graphical representation may be requested for each of the suggestions to better understand consequences of the changes. In addition, a list of effective changes can be shown. To improve the results, specific operations may be prohibited, removing all suggestions that make use of the operation. The prohibitions may either be very specific (“Don’t assign role Admin to User A”) or partly defined (“Don’t assign role Admin”). The stakeholders may repeat this step in a drill-down process until the suggestion count is decreased to a reasonable number and they arrive at a decision.

7.3. A comparison of the modes of policy management

The previous sections introduced three modes of policy management: A textual and a graphical representation, and a tool that supports policy changes based on concrete change goals. These artifacts lend themselves to explore the weaknesses and strengths of the different modes of policy management, and how the modes affect the interaction between stakeholders. Moreover, the modes are based on one authorization model so that we can also examine what problems surface when authors interact with the model. We thus primarily target the following three research questions in this section:

- How well can authors interact with policies through the different modes of management and what strategies do they apply? (Q13.1)

- How well do the modes support the interaction between policy maker and author? (Q8.1, Q7.2)
- How does the comprehensibility of the authorization model influence the policy management? (Q12.1)

7.3.1. Study design

Considering the richness of the research questions, similar argumentation for qualitative research as for the Authorization Problems Study (cf. Section 5.1.1) applies here. More specifically, we need to qualitatively evaluate the artifacts to "look more closely at how things work (or do not work). This can later lead to improved understanding of causal mechanisms" (Kelly, 2007, p. 470), instead of only "does it work?" (p. 468). Accordingly, we conducted a formative evaluation with a small number of carefully sampled participants who interacted with the artifacts to complete tasks. This study thus falls into the category of subjective, empirical laboratory studies (cf. Section 2.5).

Experiment procedure In the introductory phase, to prevent biases, it was explained to the participants that the experiment did not aim to identify the best management mode, but elicit problems with the different modes of managing policies. The participants answered semi-structured questions on their context, including their organization and position, and what authorization contexts they interact with. They were then introduced to the role-based authorization model through a diagram of an unrelated policy and to the overall scenario (described below) through a process diagram.

In the main part of the experiment, the participants were walked through one example problem and completed one task on their own for each management mode (Cairns and Cox, 2008). They were asked to think out loud while solving their task and report any problems that they encounter. Additional in-situ questions were posed to clarify their approach, explore problems, or obtain their opinion. The experiment ended with closing questions on the satisfaction with the management modes and on whether the modes would be adequate in the participant's context.

Scenarios The overall scenario was that of a small manufacturing company with 20 employees and the procurement process in the resource-planning tool of the company. The existing process was presented to the participants as a process diagram of six steps, beginning with creating a purchase order and ending with approving the payment. The exercises were each presented as concrete problems – for example, that a specific user complains about the time that orders take because of the delay at the procurement. The task was then twofold: First, identify the current permission situation ("Who holds the permission currently?"); second, how could the policy be changed to address the problems ("How could the permission be granted to the requesting person?").

The textual representation of the policy was presented to the participants as a syntax-highlighted printout (cf. Section 7.1.2) that contained German comments and a printout of a table showing the user–role assignments. For the graphical (cf. Section 7.1.3) and change-support mode (cf. Section 7.2), participants interacted with a laptop. The example policy had seven roles with a total of 41 permission statements on role–permission assignments and role inheritance.

The experiment design of using one policy for the entire experiment could result in a learning effect that simplifies each subsequent task. However, different policies would have caused overly high cognitive effort for readjustments and could have resulted in confusion. Also, it is realistic for policy authors to become familiar with a policy, even though this impedes a direct comparison of the performance with the different modes. We countered learning effects by carefully formulating the tasks to affect different parts of the policy.

Code	Position	Organization	Authorization context
A1	Administrator	Home care, 50 employees	Primarily office staff (10 users)
A2	Administrator	University department	Building access control (1000), research group systems (20)
A3	Research scientist	Personal/family context	Home systems (4)
M1	Managing stakeholder	ISV, process visualization, 10 employees	Access to office documents, source code repository (10)
M2	Managing director	Technical quality assurance, 50 employees	Business system (50)

Table 7.1.: Participant sampling for policy-change study

Participant sampling Flyvbjerg (2007) argues that "when the objective is to achieve the greatest possible amount of information on a given problem..., a representative case or random sample may...not be the richest in information" (p. 395). In this case, "information-oriented selection" is more suitable than "random selection" (p. 396). To achieve a precise "information-oriented" sampling, participants were recruited from the personal environment of the researcher. According to Nielsen and Landauer (1993), a rather small number of carefully selected participants suffice in in-depth usability studies to discover the problems. We thus recruited a small number of practitioners from a variety of backgrounds (cf. Table 7.1), where they either act as policy maker or policy author. While the number of participants cannot provide comprehensive or representative results, the careful sampling allows us to elicit useful, exploratory findings on the interaction with the different modes of policy management. Moreover, our recruitment strategy can result in a bias of the participants. However, the study design should counter this threat through a rich qualitative interaction between interviewer and participant. Another threat is the slight technical bias in the sampling, so that additional problems may be expected with policy makers with less technical affinity.

Analysis The evaluation sessions were audio-recorded and transcribed. In a Grounded-Theory approach (Adams et al., 2008; Glaser and Strauss, 1967, cf. Section 5.1.1), the transcripts were then openly coded for problems (encountered or stated obstacles), opinions (emotions or valuations), and observations (insights from observing participant behavior), resulting in 171 raw codes. The raw codes were consolidated in axial coding by code category (e.g. "cognitive load") and detailed code (e.g. "text scanning") and associated with the management mode. While the transcripts were in German, coding and consolidation was conducted in English and quotes in the following were translated into English.

7.3.2. Textual: High cognitive load when the structure is unknown

Cognitive load and incomprehensible policy

The participants experienced a high cognitive load while completing the exercise on the textual representation of the policy. One problem was that they needed to "scan" the text to, for example, find occurrences of permissions, particularly, since the structure of the policy was unknown:

> *M1: "If I know the text, it is something different...I first need to find out about the structure"*

Without knowing the structure, the participants were lacking a clear overview of the policy and, thus, had problems to elicit the alternative change-options and weigh them because of the perceived high number of options. When rather “simple” tasks require overly high cognitive effort, it impacted the satisfaction:

> *A2: “I wasn’t aware that they inherit from [role], that makes sense, but I need to read the hierarchy out of this, that’s annoying”*

The cognitive load also affected the comprehension of the policy. In case of the hierarchy, it occurred several times that participants overlooked an inheritance, as shown in the previous quote. Similarly critical was the perception of the consequences from the lack of overview:

> *A1: “Here, I am afraid: I would notice it two, three times, but then the error occurs that I overlook one [permission]”*

This also led to issues with the identification of the consequences (M1). Moreover, participants had problems with the syntax and semantics of the statements in the policy. For example, M2 did not notice the difference between system identifiers of roles and their descriptive string. For semantics, most problems related to the lack of connection between the system representation and reality, for example, for permissions:

> *M2: “OK, one would need to have a look what actually is meant with ‘create’ and ‘approve’… accepting a delivery can have to do with several things… what is it that one does when one accepts a delivery, you’d check the delivery note… ”*

Diverse problem-solving strategies

We also observed a number of problems in the problem-solving approaches. Common problems were that participants (1) resorted to confabulation (arriving at conclusions independent of the facts), (2) made changes without actually understanding them, and (3) employed “satisficing” (choosing an alternative without comprehensively evaluating the consequences, cf. March et al., 1958):

> *M1: “Following logics, if [the manager] is at the top, then ‘employee’ has to be the lowest level, so everyone in-between should also have the ‘employee’ role”*

The cause for the satisficing and confabulation could be the aforementioned high cognitive effort and the abstractness of the policy:

> *M2: “One would tend to assign everything to the employee role, because that’s quick and simple, then you don’t think much about the consequences, because the roles are not as visible”*

Apart from these problematic problem-solving strategies, participants reported a number of further approaches to lower the effort of policy management, enumerated in Table 7.2. The diversity of strategies highlights how context-dependent the interaction with a textual policy is.

Context dependency

Despite the problems, participants saw the textual representation as useful given the right context-of-use, that is, conversely, they saw it as inadequate for “lay users”:

> *A1: “Only* we *are able to do this… you cannot show this to anyone else, they panic”*

But adequate for experts:

Mechanistically following a simple pattern	*"Then I scan [the text], search for 'deliveries', then I match the verb... I do this purely mechanistic"* (A2)
Scanning the comments	*"I looked at the comments, that's the simplest"* (A3)
Creating and employing rich additional media	*"If I would work more often with this, I'd first draw myself an inheritance tree"* (A2)
Transferring the problem to reality for concreteness	*"One needs to think how this is achieved in the warehouse, maybe with a red cross [on the crate], I just imagine such a dusty warehouse"* (A1)
Relying on documentation to understand the semantics	*"Reading the manual; for the Wikis that we employ here, I regularly forget how one formulates the ACL"* (A2)
Relying on expertise of the policy, information system, or domain	*"Then, I know already what [the requester] means if he says he should be able to access this room or that wiki, then I know what domain this pertains... and there is no question [what permission is meant]... I know the system because I do this all the time, so I know this automatically already"* (A2)

Table 7.2.: Problem-solving strategies for textual policies

> *A3: "Using the text, one may proceed more strategically, there is less danger to overlook something, because it is all listed there... but I also do a lot with config files"*

And for frequent changes:

> *A2: "If I do this three times a day, it would suffice to edit the text file. That may be quicker, I can see on the first look what I need to see"*

7.3.3. Graphical: The benefits of the bigger picture

When using the graphical representation of the policy, participants still partly found it difficult to comprehend the policy, for example, concerning the semantics of the policy. While there were no problems with understanding the hierarchy, it was still difficult in some cases to have an overview over consequences from changes. This primarily occurred when participants used the option of zooming to only show individual roles.

The problem-solving approaches were different for the graphical representation. In contrast to the cognitive load to gain an overview from the text, participants were delighted that it was more pleasant, simpler, and quicker to use the diagram and gain an overview:

> *M1: "[In the diagram, it] is all clear, simpler; if I know the text by heart, have in front of me all the time, that would also be fairly quick, but in this way, I see it on the first look, who has got which permissions and what happens if I move one"*

Having the entire policy in one picture showed to be useful, for example, when A2 needed to refer to the "bigger picture" before being sure that he did not overlook something. The diagram was even seen to be a necessary complement to the text:

> *A1: "[For the text], I would need to do something manually in parallel, I would draw such a diagram"*

Moreover, the bigger picture created a sense of security (A1: "I would not want to zoom, I feel more secure [with the overview]"). In contrast to the textual form, participants did not resort to confabulation. It still was difficult to identify all change options, though. However, the graphical representation was perceived to be more accessible:

> *M2: "For the decision maker it is more real, not as virtual, more tangible, because there are also names of the users with the roles, that makes it more concrete"*

Generally, as already indicated in the above quote of M1, the adequacy of the graphical representation really depends on the context of use, for example, the expertise of the policy author and the frequency of changes:

> *A2: "Currently, since the system is new, [the diagram] is a supporting tool. If I do this three times a day, it would suffice to edit the text file. That may be quicker, I can see on the first look what I need to see"*

Participants also stated doubts on the viability of the graphical approach in other contexts, considering its scalability:

> *A2: "If that are 120 [employees], it does not work anymore... and with one additional role, this would get confusing again; if I added 10 new roles, it would not work anymore"*

7.3.4. Change support: Supporting, but abstract

When employing the change-support tool, the participants followed a different strategy. The difficulty here is not to identify options – these are provided by the system, including alternatives that the participants did not come up earlier (M1: "Ah, a new role, that is another idea"). Similarly, consequences of changes are easier to elicit:

> *M1: "This is great, because it showed me [the options] comparatively quickly and surely, I did not need to think myself, I could not overlook consequences. I could live with missing alternatives, but missing consequences would be risky"*

Thus, participants could focus on the weighing of the alternatives:

> *A1: "But then they can create suppliers, that would not be good, would it? So rather a new role, but that could get annoying, because that would create a flood; and then everyone likes to have something... It would be best to give the individual employee the permission, but then again you get quarrels"*

The general strategy is to "play" with the tool, for example, through excluding those alternatives that are irrelevant, reducing the suggestions to those that need to be assessed (cf. Section 7.2.2). Playing with the tool and the output is perceived to support both the policy author and the interaction with the policy maker:

> *M1: "If someone needs to decide on this, then he will ask first for the consequences... not approximate [consequences, but clear]... For discussions, this is the only reasonable thing, I could print this out and hand this over. And I can play with this and try out what the consequences are"*

However, A1 noted that he does not believe non-technical policy makers could "cope" with the output from the tool.

The preference for the textual or graphical representation of the change consequences corresponded to the opinion on the textual and graphical management mode. While A3 prefers a textual listing "to then see which permissions he is granted overall," M2 benefits more from a graphical view:

M2: “That she may actually do everything then, that is not as clear [in the textual listing as in the graphical view], that it is very much”

Participants categorized the change-support tool as “helpful” (A2), even fascinated from the technology (A1). Interestingly, there are conflicting opinions on the feeling of security that the tool provides. On the one hand, M1, as quoted above, would use the output to offload responsibility when interacting with the policy maker. Similarly, A1 feels secure because of the mechanism:

A1: “Here, I can be sure that the objects were checked already by the machine, I can be very sure. While with the text file... ”

Conversely, the abstraction from the actual policy created the need to consult the textual policy (A1, A2) and even reduced the feeling of security for other participants:

A2: “We don’t see the roles that people have indirectly; we don’t understand why the person has the permission... For an administrator... it is important to have as much information as possible – one doesn’t like surprises, such as: ‘Oh, why is he able to do that’ ”

Further critical points addressed the technical feasibility of the tool, concerning the scalability as well as the number and relevance of the generated alternatives:

A2: “It surprises me that the two should get different roles... With more users you’d have millions of [suggestions]... Could become unusable because there are too many options that are irrelevant”

7.3.5. Discussion

Management modes have fundamental strengths and weaknesses

Each policy-management mode showed specific weaknesses, particularly with respect to the comprehensibility of the policy and consequences of changes. While these were particularly pronounced for the textual representation, we found problems with the policy semantics for all modes. Other problems depend on the focus of the individual mode; for instance, some participants had difficulties in identifying and weighing alternative change-options for the textual and graphical representations, but not for the change support.

The specific weaknesses and strengths of the approaches also resulted in different problem-solving strategies: The textual representation already required a high cognitive effort to elicit alternatives, leading to satisficing and confabulation. The graphical representation provided the bigger picture of the policy, which helped to induce a feeling of security when considering the entire policy. It was also seen as more concrete than the textual policy where roles and users are separated. In contrast, the change-support tool allowed the participants to focus on weighing alternatives, but it also abstracted detailed information and required trust in the algorithm.

Independent of the management mode, technical challenges occurred due to the specific authorization model that limited the model expressiveness and, for example, did not allow to directly assign permissions to users. For the graphical mode and the change-support tool, participants also stated doubts about the scalability of the approaches with larger numbers of users, roles, and permissions. Despite these doubts, the participants tended to be more satisfied with these modes than with the text policy. However, these opinions seem to correspond strongly with the participants’ backgrounds and preferences.

Validity of the findings

The findings are based on subjective empirical data, thus relying on honest feedback by participants. There is a controversy on the value of rich, qualitative empirical evidence (Adams et al., 2008). However, it is difficult to design objective experiments that are representative without overly limiting the scope or only resulting in binary findings. Thus, subjective empirical research is more suitable in the exploratory phase that policy usability is in. The subjective approach was also chosen for its richness that allowed to explore the reasons for problems and derive recommendations.

Further threats originate from the participant recruitment strategy: For a precise sampling of the participant backgrounds, the participants were recruited from the researcher's personal environment. General, personal connections can lead to a sampling bias, since participants may make statements in a way to please the researcher. However, the rich qualitative setting allowed to detect dishonest statements. Moreover, this effect was further countered by a clear statement at the beginning that the study was not to establish the best way of policy management, but to contrast strengths and weaknesses of the different modes.

Despite the recruitment strategy, while providing a broad range of backgrounds, the small sample prevents a comprehensive coverage of the variety of authorization contexts in practice. For example, the policy makers in the study had a rather high technical affinity. We can assume even more problems with the syntax primitives of the textual policy for less technical affinity. However, the diversity covered already allow for useful findings without the claim of comprehensiveness or representativeness. Related threats may be caused by choosing one specific authorization model (role-based) as the technical background of the study. The findings may not apply to simpler models. However, role-based authorization is wide-spread and very relevant in practice.

Lastly, the specific experiment procedure poses a threat to the validity. A bias may be introduced into the reported problems from the sequence of management modes: Participants could have become used to the tasks and the policy. Quantitative results (purposely not employed in the findings) on problems and the general cognitive load from the tasks may have been influenced by this effect. We counteracted this effect with independent scenarios, although they were all based on a single policy to prevent confusion from changing policy or excessive cognitive load from switching context. Also, due to the limited number of participants, a between-subject study design was not an option. However, we focused on rich feedback and largely ignored quantitative results so that the effects should be contained.

Overall, despite these threats and a lack of comprehensiveness and representativeness, the richness of the findings should offer good hypotheses for further research at the current state of research on policy management. Follow-up studies with further improved artifacts should comprise a larger number of participants from a broader range of contexts and a more robust experiment design, for example, with changing sequences of modes or a between-subject design for more comparative results.

Recommendations on the design of policy management

Considering the reported problems and further comments and opinions by the participants, we can derive general recommendations on how to design policy management:

- *Support policy decision:* As reported by participants of the study, an often important aspect of making changes is to weigh the different options of achieving a change goal. When identifying alternative change options and their consequences incurred a high cognitive load or the options were too abstract and could not be systematically compared, assessing the risks showed to be difficult (P2, P3). Moreover, participants noted that they would benefit as policy authors from a support of the interaction with policy makers by the change tool (P1).

- *Take the context into account:* Despite the small sample, the study demonstrated how different the authorization contexts of policy authors are and how the differences affect the requirements and preferences regarding policy management. While rich, graphical representations were appreciated for some contexts, textual ones were seen as more appropriate for others (P2). Mentioned or observed factors that influence the appropriateness include the policy-author preference and expertise (domain, policy, change mode), the policy complexity (change frequency, quantity of users, roles, permissions), and the complexity of the authorization model (e.g. role inheritance). The design of the policy management tool, particularly its mode, must reflect the context (cf. the authorization taxonomy for context characteristics in Section 4.1; P8). The modes may complement each other, allowing the policy author to choose the adequate tool depending, for example, on the task, expertise, and feeling of security.

- *Support the problem-solving strategies:* Depending on the management mode and the participant background, participants employed a variety of problem-solving approaches. Extending the previous recommendation, we also need to support the specific strategies of policy authors and makers (P2, P8). For example, in unknown textual policies, the study showed that it is important to support the text scanning and the building of a mental model of the policy structure.

- *Improve the user experience:* The study revealed several ways in which the modes influence the experience and satisfaction of policy authors: Annoyances were, for instance, caused by inconsistencies of terms, missing information, or when apparently simple tasks, such as understanding the inheritance structure, required excessive cognitive effort. Conversely, the satisfaction was improved from the feeling of being in control, and when tasks could be completed quickly with little effort (P2) – potentially reducing the threshold for making changes (P5). Participants also responded positively when a tool was intuitive, when it fascinated them, or when they could “play” with it.

7.4. Effective and efficient integration of enforcement controls

The previous sections focused on how the authorization framework can support policy makers and authors in managing the policy. From the development perspective, a framework can support the integration of enforcement controls. Developers can improve the software quality through increased reliability of authorization – employing, for example, a usable API for the controls. The main focus is the enforcement effectiveness through the precise implementation of enforcement, and the enforcement efficiency, from a low effort (Req 15, Q15.1). The main challenges in authorization enforcement are the correct placement of enforcement statements. Errors in placement were, for example, found in Linux kernel modules (Jaeger et al., 2004). Similarly, measures to enforce authorization as part of the architecture may be circumvented by accident through unanticipated control flow (Sohr and Berger, 2010).

Another challenge lies in changing authorization requirements. With inappropriate enforcement models, the developer needs to modify enforcement statements for changes to the permissions. For the enforcement modification, the developer has to identify the relevant enforcement points in the program code by inferring the control flow paths and has to understand the parts in the code where the enforcement is to be changed. Frequently, information on why a certain state is reached is missing in development (Ko et al., 2007). The API design also affects the enforcement comprehensibility, improving the understanding of the consequences of enforcement statements (Clarke, 2004; Stylos et al., 2006). It is particularly difficult to achieve adequate authorization enforcement in software evolution, which often requires program comprehension for further development (von Mayrhauser and Vans, 1995).

Therefore, enforcement was ideally implemented once and could be left unmodified even when authorization requirements change (Beznosov et al., 1999). Vice versa, the authorization policy would not need to be modified even with functional changes as long as the authorization requirements remain unchanged. *Loose coupling* of authorization policy and enforcement is needed, following the principle of the separation of concerns (De Win et al., 2003). Systematic authorization enforcement – for example, in the form of reference monitors – were the first step in this direction (Anderson, 1972). Current authorization enforcement mechanisms come in a variety of approaches, often applied in combinations (Pandey and Hashii, 1999). Operating system-based enforcement relies on operating system mechanisms that check permissions on the level of files or processes (Harrison et al., 1976). In runtime system-based mechanisms, the running program is encapsulated and every call to a protected component is first checked against a reference monitor; an example is the Java security model (Gong and Ellison, 2003). For language-based approaches, a compiler generates additional enforcement code for the specified authorization policy (Goguen and Meseguer, 1982). Similarly, aspect-oriented programming (AOP) can be employed to enforce authorization at the join points (Kiczales et al., 1997). A related approach is the transformation of program bytecode to enforce program behavior, for example in Java programs to secure hosts against potentially malicious mobile code (Pandey and Hashii, 1999). In these approaches, the program code is regarded as potentially hostile, requiring external supervision. In contrast, many authorization decisions are taken in program code for external users, permitting a cooperative approach of program code and authorization. In these API-based approaches, enforcement hooks are placed into the source code – for example, as `checkPermission` calls in Java (Gong and Ellison, 2003) or as security hooks in the Linux kernel (Jaeger et al., 2004).

The problem with these approaches is that they are often very complex and difficult to comprehend in the program code, require a high effort (e.g. AOP), or do not separate the concerns. This is where the enforcement mechanism of `declarative_authorization` starts off from, offering the following characteristics:

- Low-threshold authorization enforcement to motivate the implementation of enforcement early in the development and reduce the effort for precise enforcement. One approach is "convention over configuration" that prevents redundancy in enforcement statements by leveraging the information on the context,
- Separation of concerns for authorization to reduce the enforcement maintenance effort and authorization-related defects. This principle works in two directions: Functional changes should not require unnecessary policy changes and policy changes should not require enforcement changes,
- Code as documentation through comprehensible policy and enforcement code, which provide more current information than secondary documentation.

`declarative_authorization` targets Web applications, so that the enforcement is structured by the model–view–controller (MVC) approach, which is typical for Web application platforms. The framework provides for enforcement on each of those layers. The enforcement points are shown in the schematic diagram in Figure 7.10. The controller layer receives client HTTP requests and implements the business logic. Here, the primary enforcement occurs: The controller coordinates the

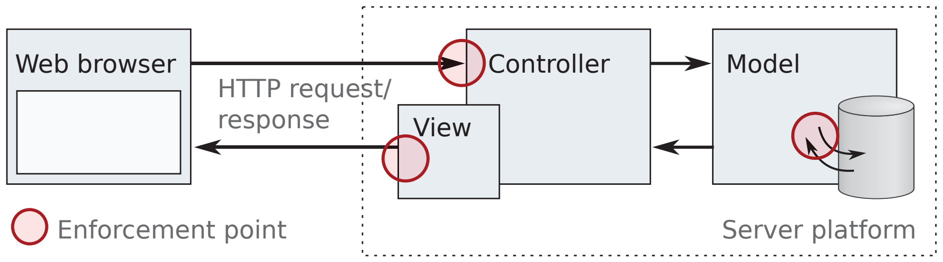


Figure 7.10.: Enforcement points in `declarative_authorization`

requests and often relies on the model layer that provides access to the application data. The model interfaces with databases, implementing an object-relational mapping. This is a second enforcement point, for example, limiting the retrieved data according to the permissions of the requesting user. Data from the model is passed back to the controller, which then generates the views to be sent back as results to the client. The authorization policy can also be enforced in views to hide restricted elements, such as links to restricted functionality.

To support the effectiveness and efficiency of authorization in software development, the ease of its integration in the application is important. Imposing a reduced overhead for enforcement allows the authorization infrastructure to be integrated early-on and reduces the potential defects from authorization that is implemented as an afterthought. The main principle that is followed on the different enforcement layers is reducing redundancy and adhering to the principle of convention over configuration. Simple and concise enforcement declarations take advantage from best-practice conventions, reducing explicit configuration that can be derived from the context of the enforcement statement. For more complex cases, detailed configuration options provide the necessary flexibility. In the following, we discuss the respective enforcement controls on each layer.

7.4.1. Controller

In MVC architectures, controllers are responsible for the business logic and the coordination of incoming requests and are thus the natural place for the first line of defense. In case of HTTP requests, URIs are routed to the controller actions, such as “list” (generates a list of items) or “show” (showing one item’s details). The enforcement can be imposed as a control for each of the actions and is then enforced prior to the execution of the business logic. `declarative_authorization` provides filters that are established with the `filter_access_to` statement. General enforcement for all actions set up with one statement as shown in the following example:

```
class ConferencesController
  filter_access_to :all
  def index
    # ...
  end
end
```

When the “index” action in the example’s `ConferencesController` is called by a request to display a listing of conferences, the authorization is enforced before the control is passed on to the “index” method. The authorization framework derives the necessary permission “index conferences” from the controller name and the requested action. If the permission has not been directly or indirectly assigned to any of the current user’s roles, the request will be denied. By default, authorization constraints (e.g. “may only read published conferences”) are not considered since context object cannot always be derived from the request to check the constraints against. The “index” action in the example is one of these cases. In case of "show" actions that directly target one object to be displayed, authorization constraints may be checked as well.

A common implementation pattern for Rails controllers are resource-oriented actions that correspond to the REST HTTP methods and implement seven actions mapped to the CRUD (Create, Read, Update, Delete) activities. The methods that require authorization-constraint checks can be deduced for the conventional methods. Accordingly, the authorization framework provides the `filter_resource_access` statement as an alternative that sets up the necessary filter configuration and object loading for the individual REST actions.

Even with only one-line enforcement statement on the controller layer, developers may still introduce defects – for example, through missing individual actions. To provide a simple overview of the established filters, `declarative_authorization` offers an analysis of the controller-level authorization enforcement and displays the coverage for each of the controller actions in a textual representation, differentiating between enforcement with and without authorization constraints.

7.4.2. Model

The model level of the MVC architecture provides the link to the application's data. Here, a second line of defense can be established with additional authorization enforcement – in two ways: First, programmatically restrict the results of database retrievals to those objects that the current user may access. In the conference-controller example in the previous section, this type of model enforcement can be employed to limit the conferences that are shown in the listing:

```
class ConferencesController
  filter_access_to :all
  def index
    # @conferences = Conference.find(:all, :conditions => ...)
    @conferences = Conference.with_permissions_to(:read).all
  end
end
```

Without the framework-provided enforcement `with_permissions_to` that imposes limitations on the database query, the conditions would need to be explicitly stated for each role, as demonstrated in the commented line. Instead, for a guest user and the policy given in Listing 7.1, the conditions would thus be set to only retrieve published conferences. While similar query rewriting has been described already as early database-specific security mechanism (Stonebraker and Rubinstein, 1976), we in this case reuse the rules on multiple layers. The important effect for developers is that they do not need to adapt the enforcement statement to changing policies, such as new roles or additional constraints, in multiple places.

The programmatical restrictions on database queries are most relevant for reading operations. For create, update, and delete actions that typically operate on individual objects, the framework provides the statement `using_access_control` on the model layer. The statement activates permission checks on those actions, so that, for instance, a conference may only be saved if the current user has the permission to update conferences. In practice, this approach may be too restrictive. There are cases in which users should not be able to edit objects directly, such as conferences, but the application should nevertheless be able to update the object for administrative tasks in their requests, for example, updating a timestamp of the last access.

7.4.3. View

The majority of authorization is enforced on the controller and model layer. Nevertheless, the view may contain data that should only be displayed to users with certain permissions. In addition, the user interface often needs to be adapted according to the current user's restrictions – for example, only showing menu items of actions that the user is allowed to access. View enforcement restricts which part of view templates and thus which data is rendered. `declarative_authorization` provides the `permitted_to?` statements for this, which either operate on general object types, such as "read conferences", or specific objects, such as "edit this conference".

7.5. Enforcement usability in practice

To assess the impact of authorization on software development and the usability of the enforcement API, we studied the development of a custom Web application for a medium-sized enterprise (50 employees) from the automotive-supplier industries. The application supports the company's business processes – among others, the collection and reporting of quality-assurance results. In particular, the aim was to improve the understanding of how the loose coupling of policy and enforcement affects real-world software development in a case study. The approach is an in-depth analysis of authorization through an examination of a project's source-code repository. At the time of examination, development had been ongoing for 2.5 years with small teams that employed agile development practices. The application had been in full productive use for 1.5 years.

The case study has the following research goals: First, what is the actual impact of authorization on agile software development? The hypothesis is that agile development causes frequent changes of functional requirements, including authorization requirements – that is, requirements related to permissions in the system. Thus, a substantial part of the software development should affect authorization (Hypothesis H_E1). Second, how does a loose-coupling authorization framework help in authorization development? When policy and functional changes are fully coupled, all policy changes would affect functional code and all functional changes to enforcement would affect the policy. Conversely, with perfect loose-coupling, we can expect that changes only affect either the authorization policy or the functionality (H_E2). Third, which properties of authorization frameworks positively or negatively affect software development? Despite loose coupling, we can still expect that there are authorization-enforcement modifications for authorization requirement changes and vice versa (H_E3).

7.5.1. Study design

Similar to the Authorization Problems Study (cf. Section 5.1.1), we here also need to conduct an in-depth analysis for meaningful results and are constrained in the number of available organizations as cases. Accordingly, we conducted a case study on one development project. For the study, we analyzed the version-control-system commits to the development branch of the Subversion repository (Collins-Sussman et al., 2004) over almost 1.5 years – beginning with the introduction of the `declarative_authorization` plugin in the project, which superseded a very simple authorization mechanism. We target the efficiency aspect of developer usability and, thus, would need to study the effort that developers spend on different tasks. Since development effort is difficult to assess directly, the analysis is based on commit counts and the relation between the commits that touch authorization development aspects. The rationale is that a developer generally risks to introduce defects and spends effort if a commit affects authorization, impacting the effectiveness and efficiency, respectively. We focused on two distinct aspects, policy changes and changes to authorization enforcement, and manually coded commits according to the reasons of modifications. Since a commit may contain several changes, all coding was non-exclusive.

Firstly, we determined the total number of commits on the development branch. For policy

Functionality	Changes to the policy due to changes to the functionality of the application, e.g. the adding of a new function
Authorization bugfixes	Authorization policy modification to fix problems caused by the policy, e.g. when a user is supposed to be able to access a function but is not
Authorization refactoring	Changes to authorization policy without affecting the semantics of the policy, e.g. for cosmetic or readability reasons
Authorization requirements	Modifications related to changed permissions without further application functionality changes, e.g. when a role receives an additional permission

Table 7.3.: Categories of reasons for authorization policy changes

Functionality	Functional changes to the application that affect enforcement controls in the source code, e.g. when a new function is added and it needs to be protected with enforcement controls
General refactoring	General application refactoring that affects enforcement statements in the code, but does not change the functionality of the application
Authorization bugfixes	Changes that affect enforcement code and are caused by problems with permissions of users, e.g. when a user is supposed to be able to access a function but is not
Authorization refactoring	Refactoring of application code that is due to changes in the authorization structures, e.g. a change in the role structure
Authorization requirements	Changes in enforcement that result from changes of permissions, e.g. when a user should be able to complete additional tasks

Table 7.4.: Categories of reasons for authorization enforcement changes

changes, we analyzed commits to the authorization configuration file and coded the commits into categories (Table 7.3) using the commit log entries, inspection of the version differences, and developer knowledge of underlying reasons for the changes.

For enforcement, we automatically analyzed the differences between versions and considered a commit enforcement-related if the changeset contained a change line with an enforcement statement. All enforcement commits were manually verified and categorized according to Table 7.4.

7.5.2. Analysis

The analysis of the commit quantities are shown in Table 7.5. For each commit type, a line lists the quantity and, if applicable, the ratio against the total commit quantity and against the enforcement or policy-related commits. Commits were not exclusively coded, so that the ratios do not sum up to 100%. Accordingly the commit counts cannot be aggregated directly, so that the table additionally contains aggregations of the raw categories:

- *Requirements-related:* Commits to the policy that are motivated by changes to functionality or authorization requirements,
- *Non-functional:* Commits that change the policy to modify the application behavior without involving changes to the functionality of the application (authorization bugfixes or authorization requirements),
- *Authorization-only:* Commits that change enforcement controls and are motivated by changes to authorization (authorization bugfixes, refactoring, or requirements),

	# commits	of total commits	of enforcement/ policy commits
Total commits	610		
Total authorization-related	137	22.5 %	
Policy-related	58	9.5 %	
Functionality	17	2.8 %	29.3 %
Authorization bugfixes	8	1.3 %	13.8 %
Authorization refactoring	15	2.5 %	25.9 %
Authorization requirements	33	5.4 %	56.9 %
Requirements-related	44	7.2 %	75.9 %
Non-functional	39	6.4 %	67.2 %
Enforcement-related	116	19.0 %	
Functionality	52	8.5 %	44.8 %
General refactoring	35	5.7 %	30.2 %
Authorization bugfixes	11	1.8 %	9.5 %
Authorization refactoring	20	3.3 %	17.2 %
Authorization requirements	9	1.5 %	7.8 %
Authorization-only	37	6.1 %	31.9 %
Non-authorization	83	13.6 %	71.6 %

Table 7.5.: Quantities and proportions of the commit reasons

- *Non-authorization:* Commits that affect enforcement controls and are caused by changes to functionality (functionality or general refactoring).

Generally, almost a quarter (23%) of all commits were related to authorization, almost 10% to the authorization policy. From the quantitative results in Table 7.5, we can derive several more specific findings:

- *Changes to the requirements caused changes of the policy:* 7% of all commits (76% of all policy-related commits) affected the policy and resulted from changed requirements (authorization and functional).

- *Changes to the application behavior often only required to modify the policy:* In two thirds of the commits relating to the policy, authorization behavior was changed without functionality changes.

- *Changes to permissions more often affected the policy than enforcement:* There were 33 commits caused by authorization-requirement changes on the policy compared to only 9 for enforcement. Only 8% of the enforcement commits were related to authorization-requirement changes without further functionality changes.

- *Most enforcement changes were necessary as part of the functional development:* 72% of all enforcement-related changes were for general refactoring or functional changes ("Non-authorization"), thus could not be replaced by only changing the policy even for perfect loose coupling.

- *Functional changes were more likely to affect the enforcement than the policy:* While 9% of all commits changed enforcement because of functionality changes, functionality changes caused policy changes only in 3% of all commits.

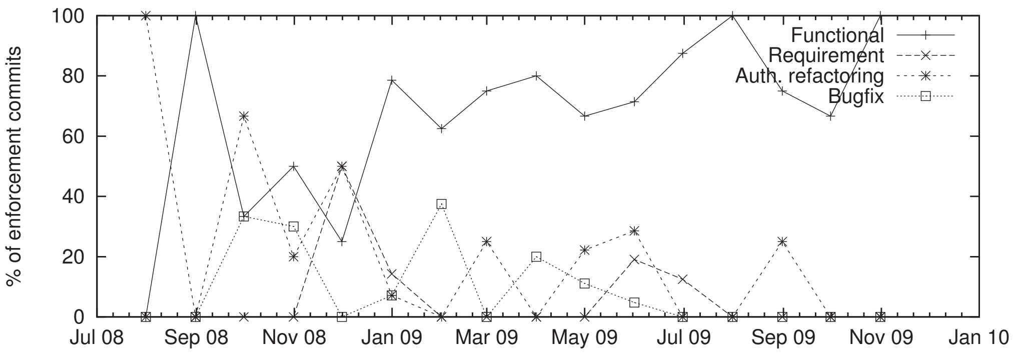


Figure 7.11.: Distribution of reasons for enforcement commits over time

Although, in perfect loose-coupling, no authorization-related modifications of enforcement should be necessary (H_E2), one third (32%) of the enforcement changes related to authorization bugfixing, refactoring and requirement changes (“authorization-only”). We analyzed the reasons of authorization enforcement changes further by examining the distribution of commits over time:

- *Authorization-related enforcement commits primarily occurred in the transitional period:* The monthly commits relative to the total number of enforcement commits are shown in Figure 7.11. In the diagram, the most striking development is that authorization-related enforcement commits were decreasing after the first five months. This indicates that there was a transitional period after the introduction of the framework, in which most of the authorization refactoring occurred. Later, the authorization enforcement stabilized and less refactoring was necessary.

- *Authorization-related enforcement commits were in some cases required to address authorization problems:* A closer examination of the remaining enforcement changes indicates that when authorization problems needed to be fixed, originally missing or wrong enforcement controls needed to be added or modified. In particular, the authorization framework allows to place either very specific controls, aimed, for example, at a concrete object, or general checks for a permission without a relation to an object. Thus, if a more concrete authorization check became necessary, more general controls needed to be modified.

7.5.3. Discussion

The findings are for the most part in line with the expectations. A considerable part of the development was related to authorization as expected from agile development, which causes frequent changes to functional and authorization requirements over time (H_E1). Functionality-related authorization changes affected enforcement to a higher degree than policy. Similarly, policy modifications are more often caused by authorization requirement changes. This indicates a loose coupling between policy and enforcement that improves the developer usability by reducing effort and preventing errors (H_E2). We still found authorization-requirement commits that affected enforcement and vice versa – an indication that the loose coupling is not perfect. The primary reasons were a large amount of transitional work in the first months after the framework introduction. To a lesser extent, enforcement changes were also necessary to fulfill changing authorization requirements (H_E3).

Since we conducted a single-project case study, there are several threats to the study validity. For the scope validity (external validity, cf. Section 5.1.1), the results are mainly applicable to custom-developed business Web applications that involve a non-trivial amount of authorization, which prevents the definition of the entire authorization requirements in advance. The specific development process is also important, since the agile methods in this project caused continuously changing functional and authorization requirements. As most plan-driven development processes also have changing requirements (Lehman, 1980), the effects should be similar in other process models. The specific development context could also pose a threat to the validity since the development was conducted in the university realm. However, the development was very focused on the product owner's requirements and was conducted by skilled developers of at least two years of experience with the development framework. Lastly, the specific programming language, Web development platform and authorization framework may have affected the results. We analyzed whether changes to the framework resulted in additional refactoring but the API was rather stable and only one commit was caused by the refactoring of the application code due to a non-backward compatible change to the framework. There was no authorization refactoring from newly introduced framework functionality. Still, future research should include studying other authorization frameworks and platforms.

Other threats to validity are posed by the study author's involvement in the development (reliability). While it is common in the social sciences to conduct empirical work with active involvement of the study authors, for instance in action research (Cairns and Cox, 2008), it is less common in computer science. Still, the involvement should not have significantly influenced the results as the analysis was fully conducted post mortem and, more importantly, the analysis was only decided after the studied period was over. Thus, the author should not have been biased in development. Moreover, the developer involvement in the study was inevitable as the coding of commits was only possible with knowledge of the system and the course of the development.

Lastly, threats to the construct and internal validity originate from the methodology of analyzing commit quantities instead of the actual development effort as indicated in the methodology section. Generally, the metric of commit quantity is no quantitative measure of the work effort, but gives an abstract approximation through the frequencies of occurrences. If authorization was touched in a development step, a developer needed to invest cognitive effort. To rule out issues from large and infrequent in contrast to small and frequent commits, we also analyzed the number of enforcement changes per commit. While means on the change quantities are problematic because of our non-exclusive coding, the categories of enforcement changes showed similar patterns of commits with high (up to 40 changes, six commits with more than 20 changes) and small numbers of changes. The large ones were mostly merge commits, encompassing several changes in one commit. On the other hand, the combination of changes into single commits can be expected to have occurred randomly and should thus not have impacted the validity of the ratios that were the foremost source for the findings.

Despite the threats to the validity, the richness of the analyzed data on the development process should offer a good hypothesis as the basis for future research on the integration of enforcement controls.

7.6. Implementation and real-world use

The `declarative_authorization` framework is implemented as a plugin for the Web development platform Ruby on Rails (Ruby et al., 2011). Making use of Ruby's metaprogramming features, the framework integrates on the controller and model layers into the Rails infrastructure. When integrated as a Ruby gem in a Rails project, developers need to provide the current user and the user's roles through a predefined interface to the authorization framework.

The `declarative_authorization` Rails plugin has been published in 2008 as open source under the liberal MIT license that does not restrict the usage of the framework. Since then, the framework has developed into a moderately successful open source project as the second most-used authorization plugin in the Rails community with, at the time of writing in November 2011, over 95,000 downloads from the central Ruby extension repository rubygems.org[4]. The framework is hosted on the Git-based source code hosting platform Github[5] and has more than 1000 “watchers” on Github – developers who are interested in updates and news on the project. More than 20 external developers have already contributed bug fixes and extensions through patches. The Github issue tracker provides management of bug reports and feature requests. A Google-News-hosted mailing list[6] offers public support for usage-related questions.

7.7. Conclusion

The authorization framework `declarative_authorization` aims to improve authorization usability through increased participation of and interaction between perspectives in authorization. Firstly, it supports policy making by integrating functional staff into the process, providing a comprehensible authorization model and policy language, and supporting tools, including change support. We explored the strengths and weaknesses of different modes of policy management and the model in formative evaluations with practitioners, contrasting textual and graphical representations and the change-support tool:

Q13.1 *How effective are editing support and other supporting tools in integrating functional, policy making, and authoring perspectives?* The study indicated how difficult and inefficient it can be to directly interact with textual policies in certain contexts – for instance, when policy authors only seldom need to change the policy (P8). It was shown how additional modes of policy management can improve the policy comprehensibility and reduce the cognitive load for identifying change options and their consequences (P2). Adequate supporting tools may not only increase the authors’ feeling of security and the precision of consequences (concreteness, P3), but also allow them to focus on the main task – the communication and negotiation of changes –, and to offload responsibility to the tool to a certain extent (P1). As formulated in the recommendations (cf. Section 7.3.5), supporting tools need to account for the specific context, support the policy author’s and maker’s problem-solving strategy, and improve the user experience to successfully support the editing and the interaction between stakeholders.

In future work, we should extend the research on the semantics of permissions. For change support, the cognitive gap between the system permissions and the author’s or maker’s mental model of the application needs to be bridged. One approach could be to couple the change support more tightly to the application for easier selection of the desired permissions that are to be modified (cf. Karp and Stiegler, 2010). Explaining denials of authorization decisions could offer another starting point (Q14.1: Becker and Nanz, 2008; Bonatti et al., 2006).

A rather technical challenge that needs to be further explored is the scalability of the graphical representation and the change-support algorithm. For example, to improve the relevance of change-support suggestions, the risks and benefits of the suggestions could be estimated (Q11.1, Q11.2, cf. Chapter 10). Finally, the evaluation of the modes of policy management needs to be extended by studying their applicability in a broader range of contexts and their impact in practical settings on actual change requests in organizations.

[4]`http://rubygems.org`
[5]`http://github.com/stffn/declarative_authorization`
[6]`http://groups.google.com/group/declarative_authorization`

Q8.1 *How do policy authors interact with policy makers and functional staff for implementing changes?* / Q7.2 *How can different perspectives cooperate for effective and efficient reviews?* Adequate representations and supporting tools showed to be useful to make the policy more comprehensible and easier to modify when the representations fit the context (P8). While reviews were not the focus of the studies, we can expect that the problem-solving approach will differ between conducting reviews and enacting changes – but also need to be adequately supported according to the context. The findings indicate that it can be dangerous or inefficient if basic tasks, such as establishing the inheritance hierarchy, require an excessive amount of cognitive effort. However, even though adequate means may lower the threshold of participation, the findings also indicated that integrating with policy makers can be generally problematic in some contexts due to low technical affinity. There are also general problems with policy comprehensibility – for instance, due to the policy semantics – that threaten the effective management and integration of perspectives (P2, P1).

Q12.1 *What are the complexity characteristics of authorization models and how do the characteristics influence policy making and authoring?* Only a few of the reported problems related to the authorization model. For instance, participants had problems to understand the concept of the role inheritance and of the distinction between user–role and role–permission assignment. However, the problems with the comprehensibility of the policy and the consequences of changes were caused to a larger extent by the semantics of the permissions and the inadequacies of the management modes (P2). Since the conducted empirical work only touched upon the aspects of the model and language usability, these aspects need a further thorough evaluation in future work to identify the specific factors that impact the policy comprehensibility (Q12.1) and how this relates to the model's expressiveness (Q12.2).

The second usability perspective that the framework targets is the development perspective. The framework was designed to support efficient modifications of the policy and of program code with enforcement controls. It implements a separation of concerns so that authorization requirement changes only require a limited amount of enforcement changes – and functional changes only require minimal policy changes. The in-depth case study of software development with the authorization framework in Section 7.5 resulted in a number of findings:

Q15.1 *How can the integration of authorization measures be supported in practical systems development?* The separation of concerns between the policy and the enforcement reduced the necessary changes to the program code from policy changes, thus reducing the overall implementation effort and better suiting the dynamic environment (P2, P7). However, even a framework that implements a loose coupling between policy and enforcement can incur enforcement changes for reasons that are purely related to authorization – for example, in transitional periods or when the enforcement statements need to become more specific. Research is needed to identify ways of further reducing the dependencies between policy and enforcement and thus the authorization logic in program code.

Q15.2 *How can the different perspectives interact for more effective and efficient controls integration?* Through the loose coupling of policy and enforcement, the developers were less dependent on early and stable requirements, both in case of the policy and the system functionality. Thus, the system and the policy could evolve over time with intensive interaction based on experiences with the actual system (P1). In future research, we can further study this relation between the interaction of developers with policy makers and the development work.

8. The flexibility of social controls: Policy Override

The first was never to accept anything as true that I did not clearly know to be such; that is to say, carefully to avoid precipitancy and prejudice, and to comprise nothing more in my judgement than what was presented to my mind so clearly and distinctly as to exclude all ground of doubt.

Descartes: Discourse on the Method (1637)[1]

In *The Use of Knowledge in Society*, Hayek (1945) argues that in economies, the information for making choices is present locally on the spot and not centrally where the plans usually are made[2]. Similarly, many decisions about authorization restrictions are made far from the functional staff's workplaces where those decisions impact authorization usability from the functional perspective, when, for example, they interfere with the primary tasks in an organization (cf. Section 4.4).

Even if the functional staff participate in the authoring of authorization policies, staff typically only take common processes into account and cannot anticipate exceptions to the rule or future organizational changes. In practice, every once in a while, users need privileges that are not assigned at the time to complete the work at hand (cf. Authorization Problems Study, Chapter 5). According to Sinclair et al. (2008), reasons are the dynamic organizational processes and the organizational complexities that cause frequently changing authorization requirements. Depending on the procedures for policy changes and organizational security policies, inadequate permissions bring functional staff into uncomfortable situations of deciding whether to comply with the security policy (Beautement et al., 2008). According to the Authorization Problems Study, complying with the security policies leaves them with the following options:

- Follow the organizational procedures to have the privileges extended – while appropriate for longer-term changes in organizational processes, the lead time for the permission change may be too long for time-critical tasks, the effort for the change process may be overly high for one-time requirements, and secondary security risks arise from the accumulation of over-entitlements (Sinclair et al., 2008),
- Turn to coworkers with the necessary privileges who then conduct the task in their name – inefficient, since identifying permitted users can be time-consuming and the identified user needs to switch context to validate and complete the task,
- Ignore the task, for example, due to the level of frustration with long-running change procedures – thereby possibly causing a severe impact on the organization's business goals.

[1]Descartes (1850, p. 61)

[2]This chapter is based on two publications: the policy-override model was first proposed at SIN 2010 (Bartsch, 2010b), the evaluation in practice appeared in an earlier form in Wiley's *Security and Communication Networks* (Bartsch, 2012).

As Beautement et al. (2008) argue with the Compliance Budget model, the decision of whether to comply is, among others, governed by the security motivation and previous effort to comply. Accordingly, the consistent compliance in this kind of cases will require additional motivational measures, such as awareness campaigns, or result in less compliant personnel in the long run. For authorization usability issues, the affected stakeholders typically have several options of non-compliance that have drawbacks from the organization's point of view. Two frequent examples of the several that we observed in the Authorization Problems Study are the following:

- Socially circumvent the security measure and request the credentials of a permitted user to impersonate that user in the system – a practice that undermines the traceability of activities, may affect the organization's compliance to regulations, and can have uncontrollable consequences,

- Evade the restrictions by technically circumventing the information system, for example, using email rather than a document management system to share information – often inefficient, since this causes redundant copies of documents and incurring additional risks from the loss of control of the data.

In terms of the authorization requirements, the aforementioned problems relate to the overall security effectiveness and efficiency (Req 1), the measure's interference with functional work (Req 2), and the impacts on employee satisfaction (Req 4). To address problems with circumventions, we also need to adequately design the measure (Q5.2, Req 5). Can we achieve more flexible authorization that matches the flexibility of social controls?

One approach is to reconsider the authorization paradigm and take both the *least-privilege* principle (Saltzer and Schroeder, 1975) as known from conventional authorization schemes as well as the *most-tolerable privilege* into account. To these ends, Sikkel and Stiemerling (1998) describe a physical approach that allows users to cope with exceptional situations by placing an enveloped super-user password in a safe. Similarly, flexibility of authorization in information systems can also be increased by not entirely defining restrictions *a priori*. Jones and Rastogi (2004) name four forms of controls (corrective, deterrent, detective and preventive), which may be used to support the practice of informal delegation with formal bounds and safeguards. For example, a speeding-ticket analogy can be employed to reduce the improper use of authorization (Zhao and Johnson, 2009). Povey (2000) calls these flexible approaches *optimistic security*.

According to Stevens and Wulf (2002), authorization approaches can be categorized as the traditional *ex ante*, defined in advance, as well as *ex post*, decided after the activity, and *uno tempore*, deciding just-in-time. Ex-post authorization can be implemented through system activity auditing (Cederquist et al., 2007; Jajodia et al., 1995; Røstad and Edsberg, 2006). The prohibitively high auditing effort prevents a wide-spread usage of audit-only authorization. However, in combination with explicit authorization *override*, the optimistic authorization approach may be viable. The override models establish a soft boundary at the normal least-privilege privilege extent and a hard boundary at the most-tolerable privilege extent, thus implementing both the need-to-know and the must-not-know principle (Badger, 1990; Rissanen et al., 2004; Cheng et al., 2007). While specific models exist – at times called Policy Override, Authorization Escalation, or Break-glass mechanisms – little research has been conducted on how they can be successfully implemented in organizational practice.

This chapter addresses the challenges how to employ policy override in practice: The chapter describes a model and an implementation of policy override for the `declarative_authorization` framework (cf. Chapter 7). Both are evaluated in the context of an SME Web application for insights on the effects of policy override. The primary interest lies in how useful the flexibility of the model is in practice (Q12.3, Req 12). Moreover, we discuss how the mechanism can support the authorization measure, for example, by bridging the gap between functional staff and policy makers (Q14.1, Req 14), and how policy makers can be supported in monitoring the use of policy override (Req 7).

8.1. Prior work on policy override

8.1.1. Earlier practical application

Policy override is an authorization mechanism that is already in use in health care applications. Accordingly, most literature on practical experience with override and its implementation is from health care environments. Denley and Smith (1999) report on the experience from implementing override as addition to other authorization mechanisms in clinical information system. Medical personnel not previously connected to a patient is presented with a warning that an override action is possible but will be logged. They report about 50 override actions a day, mostly from less computerized staff. Røstad and Edsberg (2006) analyzed the audit logs of a clinical information system's policy override. The system has predefined reasons for override and the user may access the document for set periods. The study states a very high usage frequency of override mechanisms with records of over 50% of patients in contact in the study timespan being accessed in override mode. The high count of override accesses suggests that no appropriate auditing is possible. There also is a U.S. HIPAA (Health Insurance Portability and Accountability Act) document on the usage of policy override in health care (HIPAA, 2009).

Longstaff et al. (2000) describe a complex UML electronic health record model as an enhancement of UK's Healthcare Model with $RBAC_3$-styled authorization as defined by Sandhu et al. (1996). For the case of patients not able to communicate or not willing to cooperate in a risky treatment, policy override is available to access confidential information with logging and automatic notification of clinical governance and the patients' general practitioner. Two different types of override are described: specific override, which allows access only to the parts that the user would have been able to read in his role, if the user had a connection to the patient. The second type is general override, permitting access to all information in a patient's electronic record.

Outside of healthcare institutions, Stevens and Wulf (2002) report on an authorization case study at a steel mill that analyzes inter-organizational cooperation and competition from a CSCW perspective. From the results of the study, they implemented an authorization scheme that allows to authorize access manually or in retrospect, but do not evaluate its application.

Thus, the small body of literature on the practical application of policy override is limited to one specific domain: health care. Since policy override entails additional risks for organizations, it is vital for the organizations to weigh the risks against potential gains. This consideration is rather straight-forward for hospital health-care information systems with the high potential gain of saving life. For other environments, the decision is not as clear-cut because organizations need to balance a possibly increased insider threat against the reduced interferences from authorization with functional work. This chapter thus aims to provide insights on when and how policy override is appropriate and beneficial.

8.1.2. Existing models

Further literature on policy override discusses specific models and mechanisms. Badger (1990) proposed a formalism for the recovery of a system from the state of override in a mandatory access control system. The mechanism allows to relax constraints of the policy and reports in detail which parts of the policy was violated while in the state of override. Povey (2000) introduces policy override as *Optimistic Security* and formulates requirements. One of those requirements is a mechanism for reverting illegitimate actions in retrospect. His approach is a formal database model of transactions based on the Clark-Wilson integrity model for guaranteeing system integrity. Jajodia et al. (1995) suggest a formal model for auditing transactions in databases.

Ferreira et al. (2009) propose an obligation-based policy override approach. Brucker and Petritsch (2009) model policy override for XACML with SecureUML, including a low and high emergency mode. Cheng et al. (2007) analyze the economic aspects of access governance to propose an adaptive, risk-based authorization mechanism. They use risk quantification for risk/benefit analysis with "risk credit lines." Based on Multi Layer Security (MLS), the Fuzzy MLS system has hard and soft authorization boundaries to enact the appropriate amount of authorization. Instead of quantifying risk, Zhao and Johnson (2009) employ a game-theoretical approach to model incentives in policy override authorization. They discuss information governance and a speeding ticket analogy to enforce the proper employment of override possibilities by users. The details of the game theory model are described in Zhao and Johnson (2008).

Johnson et al. (2009) take on an economic perspective as well and compare the effects of strict and centrally-managed file access control with the failures of centrally-planned economies. They give a comprehensive overview of possible consequences of strict policies and argue that some control over authorization should be left with the individual document owner. They defined the requirements ownership, freedom of delegation, transparency, dependability, and minimization of friction. As a natural user interface, Johnson et al. designed the delegation by the email sending metaphor, but have problems when implementing it in the Windows shared folder domain. Their approach differs in that they simplify delegation, which is sufficient for file sharing as there is at least one person involved who already possesses read permissions. However, authorization override is more global in its application scope and is not limited to file sharing.

Rissanen et al. (2004) also propose adding override to authorization mechanisms and offer an algorithm to identify users responsible for auditing policy overrides. Lastly, Cederquist et al. (2007) describe a formal audit-based authorization mechanism that requires justifications from users. The domain of their approach is similar to Johnson et al.'s in that users directly interact with each other. Cederquist et al. show the framework's viability through an example from a consultancy firm.

The models and mechanisms in the referenced works are rather complex and tend to involve additional factors, such as the quantification of risks. However, to apply policy override in practice, the complexity may be hindering and make it even more difficult to estimate the consequences of policy override. Thus, we aim to reduce the complexity of the model in this chapter and only minimally extend the well-known RBAC model.

8.2. A simple policy-override model and implementation

When creating an application's authorization policy, policy makers may conduct interviews as well as task and process analysis to identify functional needs. Figure 8.1 shows an abstract example of a role's requirements lightly shaded on the left side. Depending on the effort and the analysis approach, the resulting authorization requirements may or may not be accurate. In the figure, the actual needs are shown with a dotted line.

The next step for policy makers is to formalize permissions from the authorization requirements. In most cases, the permissions will not exactly match the requirements, since authorization models are typically coarse when compared to the identified requirements due to the abstractions in the model. The default permissions are shown as white area in Figure F.2. In conventional authorization models, the individual has to cope with these permissions. As stated in the introduction, for any missing permissions in daily routine or exceptional situations the individual needs to request changes

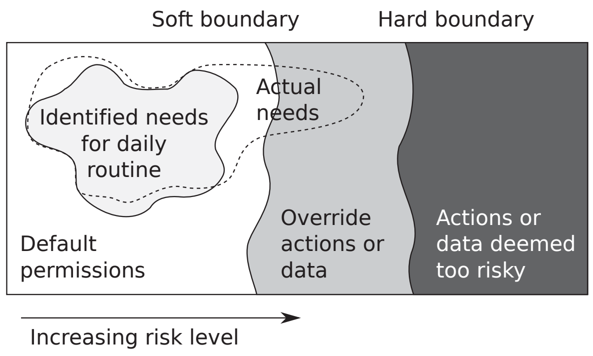


Figure 8.1.: Abstract example of a role's hard and soft boundary

or circumvent authorization.

When using an application that supports policy override, the functional staff may have the additional option of overriding denied permissions. This integrates functional staff into the policy making of their individual permissions (P1). As proposed by Cheng et al. (2007), we introduce a *soft boundary* as an addition to the hard boundary in conventional authorization models. As shown in Figure 8.1, the soft boundary separates the default permissions from those that can be gained by overriding the policy. The overriding of the policy offers the flexibility necessary for dynamic environments (P7).

Policy override reduces the burden of the policy makers and authors to adapt the default permissions for exceptional cases (P2). It also fosters "rational" staff behavior and offers a convenient alternative to circumvention (P5). Any activity in override may be subject to more scrutiny than standard activities, for example, through logging and auditing. The logging as organizational and further social controls should limit the use of override when not actually necessary, combining technical and non-technical approaches (P4).

The *hard boundary* marks the limit of privileges gainable through policy override. Anything beyond the hard boundary is thought to be of too high risk to allow the specific role to have access to it. Note that it may be appropriate to disallow any override for certain roles so that the hard and soft boundary are identical.

8.2.1. Authorization model

To implement policy override, the authorization model needs to be adapted. The authorization model used here is based on a conventional RBAC model with hierarchies, RBAC$_1$, after Sandhu et al. (1996), which is comprised of the following elements:

- *User*, representing the principal requesting access; *Role*, grouping specific functions of users; *Permission*, allowing access to objects; and the *Session* of a user acting in a system
- Assignments of permissions to roles: $PA \subseteq Permission \times Role$
- Assignments of users to roles: $UA \subseteq User \times Role$
- Role hierarchy, that is, roles inheriting permissions from ancestor roles, as a partial order $RH \subseteq Role \times Role$, with role dominance, denoted by the symbol $\geq$
- Junior roles, that is, all roles transitively inheriting from a given role:

$$JuniorRoles : (Role, Session) \to \mathcal{P}(Role), \text{ with } (r,s) \mapsto \{r' | r \geq r'\}$$

- The session's user: $SessionUser : Session \rightarrow User$
- The session's activated roles: $SessionRoles : Session \rightarrow \mathcal{P}(Role)$, with

$$SessionRoles(s) \subseteq \{r \mid \exists r'[r \in JuniorRoles(r', s) \land (SessionUser(s), r') \in UA]\}$$

- The permissions of the user in the current session: $SessionPermissions : Session \rightarrow \mathcal{P}(Permission)$, with

$$s \mapsto \{p \mid (p, r) \in PA \ \land \ r \in \cup_{r' \in SessionRoles(s)} JuniorRoles(r', s)\}$$

To implement policy override, an additional role relation is introduced to define to which roles a user may extend his privileges. This *Override Roles* relation is then used to modify the available roles that can be activated when the session is in override mode:

- Override roles: $OR \subseteq Role \times Role$ is a relation that defines to which role the holder of the first role may extend her privilege in case of override
- $IsOverrideMode : Session \rightarrow bool$ is a predicate that indicates whether a user has activated the override mode on a specific session
- *JuniorRoles* is redefined to return roles in the override hierarchy if the override mode is active for the current session:

$$(r, s) \mapsto \begin{cases} \{r' \mid r \geq r' \lor \exists r''[(r'', r') \in OR \land r \geq r'']\}, & \text{if } IsOverrideMode(s) \\ \{r' \mid r \geq r'\}, & \text{else} \end{cases}$$

The policy override modifications do not affect authorization constraints, which are enforced as usual and are for brevity not included in the definition.

8.2.2. Implementation in the authorization framework

To implement the policy override model in `declarative_authorization`, introduced in the previous chapter, two steps are necessary. First, the authorization model and policy language need to support the override options. Second, the decisioning needs to take the override options into account. For the model modification, the original authorization model, shown in Figure 7.2, was only slightly extended. As depicted in Figure 8.2, override roles may now be assigned to the roles. Users that are assigned to a role can thus override to any override role of the originally assigned role, including the role inheritance. The override option is not transitive so that, when the policy is represented as a graph, only one override edge can be on any path from the user to a permissions.

In the `declarative_authorization` policy language, the keyword `overridable_to` is added to the available statements in the roles block. This statement represents the override edge between the role that is to be extended and the override role. In the example in Listing 8.1, conference organizers can override to the permissions of the administrator role, which allows them to modify system users. This might, for example, be useful to allow the quick modification of speakers in urgent cases.

In decisioning, the main modification is to have an override status for the session, which is honored in each decision. In the override case, overridable roles of directly assigned roles as well as of inherited roles are added to the available roles and are handled as if assigned to the current user.

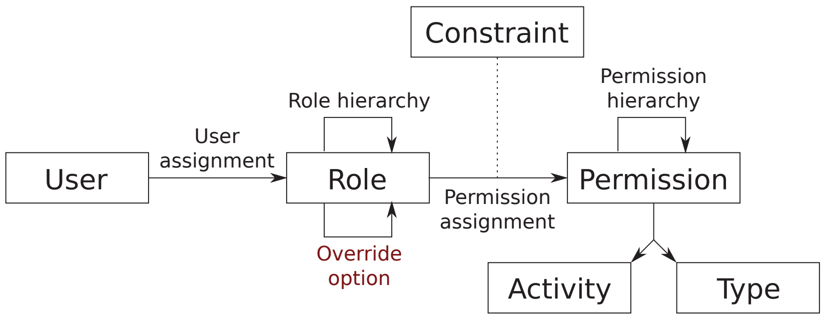


Figure 8.2.: `declarative_authorization` model, extended for policy override

Listing 8.1: Policy override example in authorization rules

```
authorization do
  role :administrator do
    has_permission_on :users, :to => :manage
  end

  role :conference_organizer do
    overridable_to :administrator
    has_permission_on :conferences, :to => :manage
    has_permission_on :conference_attendees, :to => :manage
    has_permission_on :talks, :to => :manage
  end
end
```

8.3. Policy override in practice

In case of innovative security measures, such as optimistic security and policy override authorization, practical experience with the technology in actual organizational contexts is crucial to judge the approaches' appropriateness and guide further research. However, the existing publications on the experience with policy override mechanisms and auditing is still limited to only a few domains (Røstad and Edsberg, 2006), discussed in Section 8.1. The aim of the evaluation in this section thus is twofold. Firstly, the evaluation should enhance the understanding of employing policy override in business Web applications of small and medium enterprises. Secondly, the evaluation should show the viability of the above-described policy override model. To pursue both aims, we implemented policy override through an extension to `declarative_authorization` in the business Web application described in Section 7.5. Apart from the changes in the decisioning and the policy, three application-specific implementation tasks had to be completed:

- *Implementation of an override activation mechanism:* We opted to place a well-visible button with the German label “Ausnahme-Modus” (“Exception mode”) in the top navigation area of the Web application. When in override mode, the background of the header bar is colored red to signal the exceptional state.

- *Implementation of an auditing mechanism:* We extended the existing application log viewer to display the activities in override and, thus, support the task of auditing override sessions. An additional filter option allows the supervisor to limit the displayed actions to those in override mode.

- *Signaling the override sessions to the supervisor for review:* The task management infrastructure in the application displays a symbol for pending tasks in the top area of the Web application. We add a task for each override case to users with a specific role in the branch of the user who entered the override mode.

8.3.1. Study design

For similar reasons as for the Authorization Problems Study (cf. Section 5.1.1), we conduct this study as case study: First, only one organization was available as a study object; second, an in-depth analysis of the phenomena is more important than the generalization of the results. We combined quantitative and qualitative research for the practical evaluation of the model and implementation (cf. Section 5.1.1). First, we quantitatively analyzed the application log that records the activities in the Web application and specifically marks the override activities (objective empirical data, cf. Section 2.5). The analyzed log spans the calendar year 2010. The override options were implemented at different times for different roles but all well before the start of 2010. Its implementation was communicated to functional users together with other functionality changes. From this source, we derive quantitative data on the affected users and actions together with the proportion in policy override mode. Since each HTTP request causes a raw action entry, the action quantities may be skewed from the different counts of HTTP requests that depend on the individual task in the application. As a consequence, we group the raw actions into activities by user and day, providing an alternative representation of how frequently override is used. The quantitative results should provide a good overview of the usage of policy override. Since policy override should only be necessary for temporary authorization inadequacies or exceptional situations, the hypothesis is that individual users employ policy override sparsely only ($H_{PO}1$).

The second source is qualitative: Five affected stakeholders were interviewed on their experience, both users employing override and a superior responsible for acknowledging the use of policy override (subjective, empirical case study, cf. Section 2.5). The sampling is based on the quantitative results and includes the different kinds of users as identified by their assigned roles and override usage profiles. The short, semi-structured interviews of about 15 minutes were analyzed from audio recordings and notes for each interviewee. The interviews with functional users focused on (1) the reasons for employing policy override and (2) the consequences that they drew from using it. The hypotheses are that users employ override for temporary authorization inadequacies and exceptional situations ($H_{PO}2$) and that users strive to have authorization inadequacies changed and exceptional situations better handled in the application if they recur ($H_{PO}3$).

For the superior, the questions involve (1) the practice of acknowledging override situations and (2) consequences from the superior's perspective. For this perspective, the hypotheses are that the superiors can adequately judge whether employing override is reasonable ($H_{PO}4$) and that they will try and reduce the number of override occurrences by discussions with the users, changes of the business process, or modifications of the authorization policy or functionality of the application ($H_{PO}5$).

To ensure the validity of the results, we need to consider the quality of the study design (Section 5.1.1, Yin, 2009). Regarding the construct and internal validity, we chose to analyze both objective quantitative and additional qualitative interview data to ensure the causality. For the external validity, the interviewees, whereas small were chosen to represent the different kinds of override users in the case study. The setting of the case study was specific to business Web applications and the domain of SMEs so that the generalizability is primarily limited to this domain. Lastly, the reliability was ensured by evaluating objective data in a well-described way. Thus, even if the generalizability of the study is limited, the rich exploratory results can inform the development of

	Total	By users with over-ride option	In override mode	Of those w/override option	2nd half with over-ride option	2nd half in override	Of those w/override option
Act. users	46	26	9	34.6%	22	8	36.4%
Activities	2713	2145	150	7.0%	1030	149	14.5%
Actions	534735	488373	47830	9.8%	249483	47787	19.2%

Table 8.1.: Quantitative analysis of activity and action log

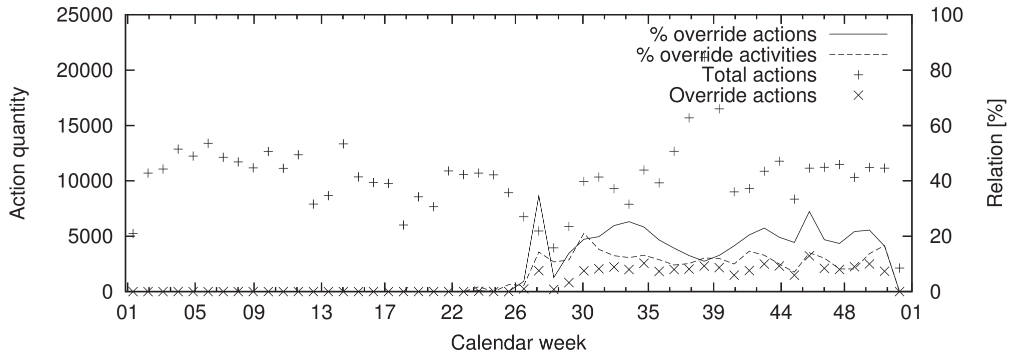


Figure 8.3.: Override use in comparison to normal use over time

policy-override implementations.

8.3.2. Quantitative analysis

The quantitative analysis should provide an answer to the questions who used policy override how often and when. Table 8.1 shows the quantities of usage of users, activities and actions. Of the 46 total active users in the period, 26 had the option of using override, of whom one third actually used override. Surprisingly, of the activities and actions by users with the option to enact override, 7% and 10% were conducted in override mode, respectively. The proportions roughly double when only considering the second half of 2010. In contrast to $H_{PO}1$, these results indicate that override was used not only in exceptional cases but as a convenient option to complete the daily tasks.

The results of concentrated use in the second half require further analysis. In Figure 8.3, the weekly use of override is compared to normal use with lines for proportions and points for action quantities. The chart shows that, indeed, all significant override activities occurred in the second half of the year, although the total usage of the system remained similar to the first half. Moreover, from week 26 onwards, the proportion of override use is, apart from a few spikes, constantly hovering around 20% of actions.

The last approach to the quantitative analysis is a closer look at the distribution among the users who employed the override mode, as shown in Table 8.2. Secretary F and Operator A can be excluded from the analysis since the action counts indicate that they entered it accidentally or only tested the override mode. From the other users, we can make out one project manager (B) and four secretaries with significant proportions of override activity. Another secretary and project manager sporadically used policy override.

	Total		Override			
Pseudonym	activities	actions	activities		actions	
Project Manager A*	75	3633	1	1.3%	43	1.2%
Project Manager B	115	1645	5	4.3%	60	3.6%
Secretary A*	237	77495	2	0.8%	12	0.0%
Secretary B*	246	99509	22	8.9%	6133	6.2%
Secretary C	34	12050	5	14.7%	573	4.8%
Secretary D*	246	54358	70	28.5%	26008	47.8%
Secretary E	134	45161	43	32.1%	14994	33.2%
Secretary F	23	6260	1	4.3%	2	0.0%
Operator A	2	6	1	50.0%	5	83.3%

Table 8.2.: Override activities and actions by user

8.3.3. Qualitative analysis

For an analysis of the reasons behind the surprisingly intensive usage of policy override by individual users and how the superiors judged the override usage, five stakeholders were interviewed. Four of the interviewees were override users from different branches, marked with an asterisk in Table 8.2, and one was a superior, responsible for acknowledging the override use in one branch.

Secretary D stated that her override usage began with a short-term vacation replacement on short notice for colleagues at a different branch. Normally, she would not have the necessary permissions for the tasks at the other branch, so she used the override mode. Although the vacation replacement ended, she continued to fill in for the other branch because of the high workload there. Since no appropriate role for the access to multiple branches existed, an alternative would have been for her to use multiple logins. Switching logins is quite a hassle, though, so she preferred the override mode for quickly completing tasks at different branches. Other exceptional situations for her to use the override mode are, for example, when the responsible project manager is not in the branch and certain tasks need to be completed quickly. Interestingly, she uses secondary credentials of project managers for these tasks despite the possibility of using override for those. She did not know that the override mode would also apply there and does "not have the time to experiment to find that out". She also stated that she "just used it and it's OK", fearing that any consequences from the recurring override usage would impact her productivity.

The reasons for Secretary B are similar, generally using the override mode "as fill-in" for other branches and to not "log out and in again" for cases for which she has a secondary login. While she did not remember to have explicitly discussed the override usage, she had, in fact, indicated to a superior that it would make sense to receive permissions to all branches. But policy override is "an alternative to that" so that she did not further pursue the request.

Secretary A's override usage override is closer to the original intention of policy override. As the quantitative analysis shows, she only used policy override in two occasions. She says that normally, she is "not missing any permissions" in daily work and only used policy override for convenience to create a new user for which she would else have needed to turn to her superior. From her perspective, many "normal" users should not have all permissions in override and "rummage wildly around the system or change things," though. For her, it is difficult to assess the risks. Project Manager A has a similar override usage pattern with only one activity in the studied period. In his case, the application would not allow task assignments to arbitrary users, but only to those with specific roles, and adding roles to users is only allowed to administrators. By using the override mode, he was able to assign the role. Subsequently, the functionality was changed to not require the additional role to be assigned. For him the positive side is that the override mode allows more flexibility without generally loosening the restrictions.

In an interview with one branch manager who was in charge of acknowledging the override activities, he could largely recall who used the override mode and for what reasons. He was aware that some functional users employed override already over a long time regularly and stated that with "so many actions, it is difficult to relate them and evaluate whether the actions are OK." This also pertained the difficulty of estimating the risks. While acknowledging works fine generally, it is difficult for him to review the specific actions that are taken in override mode. The review tool would need to better summarize the activities to allow effective reasoning. Moreover, he said that the original aim was to "look at the override usage and draw consequences from it" but with the large number of acknowledgments, he got accustomed and did not "pay attention to it anymore."

8.3.4. The consequences of inadequate organization

From the quantitative and qualitative analysis, we can observe two ways in which policy override was used in the studied application. One was for exceptional situations when, for example, the responsible user was not present and a task needed to be completed quickly, in line with the hypothesis $H_{PO}2$. For these cases, the main challenge for the users was to understand which activities they can conduct in override mode. In other cases, the application functionality was not prepared for changes in the workflow, so that the override mode helped until the functionality was changed. The second general way of employing override was the intensive and regular usage over a long period without modifications being pursued to remedy the situation, contradicting $H_{PO}2$, $H_{PO}3$ and $H_{PO}5$, but without malicious misuse. Superiors became accustomed to the acknowledgments. Functional users did not have any incentives to require changes either, even fearing that their productivity could be impacted by any changes. Also, no harm was seen in using override, although superiors invested considerable effort in acknowledging override activities. Acknowledging the specifics of override activities was difficult, partly contradicting $H_{PO}4$. Superiors need a good review tool that summarizes override activities for better evaluation. Similarly, the responsible person needs to estimate the risks and benefits from override.

Overall, the study shows a mixed result on the use of policy override. The mechanism certainly came in handy for functional staff to complete work in exceptional situations, thereby increasing productivity of the staff and resulting in less need for policy changes. However, because of the organizational inadequacies surrounding the mechanism, it was used far too often and in ways not originally foreseen – even though not maliciously –, potentially increasing the risk to the system to an unexpected level.

8.3.5. Recommendations on the implementation of policy override

From the results of the policy override evaluation, we recommend to pay attention to the following aspects when deploying policy override in applications:

- *Awareness for appropriate override usage and availability:* We, for example, saw in the study how staff used override excessively and, conversely, was unaware of override options. Accordingly, we need to improve users' awareness for when using policy override is appropriate to influence their behavior (P6, P5). Similarly, users need to understand which additional activities are available in override mode, for example, through visualization in the application.

- *Adequate tool support:* Policy makers, for example, had problems with the override-review tool. We need to provide an adequate tool that summarizes the actions taken in an override session in a comprehensible manner – enabling frequent use (P2, P7).

- *Organization for consequences:* The study showed how override can become the normal mode of operation. We, thus, need to make the provisions to guarantee that changes are enacted when override use is recurring. There are multiple options to achieve this goal, including the following:
  - Technically through indications on which override reasons recur and thus require consequences (P2),
  - Technically through further measures that raise the awareness of hidden costs incurred from repeated override use, such as an override budget that needs to be explicitly increased by superiors when used up (P5, P6),
  - Organizationally through procedures and responsibilities for reactions to recurrences (P8), or,
  - Through incentives for users to reduce the override usage (P5, cf. above).

  The consequences from recurring override usage can involve the modification of either the application functionality, the authorization policy, or the business processes.

- *Decision support:* Participants saw problems in estimating the benefit and risk from policy override. We need to provide support for the estimation of these consequences from policy override to support the decisions on the policies (P2)[3].

8.4. Conclusion

Authorization flexibility is one of the ways of increasing authorization usability for dynamic contexts (P7). The policy makers benefit from a reduced workload from unnecessary policy changes for exceptional cases and the functional staff are less often hindered in their daily tasks by overly strict policies. This chapter presents a simple policy-override model and its implementation within an authorization framework. We discussed the evaluation of one year of use of the model and implementation in the context of a business Web application at a medium-sized enterprise:

Q2.1 *How disruptive are authorization measures with what characteristics and in what contexts, and how can we reduce the disruptions or their impact?* Policy override resulted in less interference with the primary task for functional staff (Req 2). When responsible individuals were not available, but time-critical tasks needed to be completed, the override mode helped (P2, P7) – and allowed staff to participate in policy making by extending their permissions (P1). In this way, policy override increased the flexibility of authorization and reduced the gap to social control. However, we also observed that staff need to be aware of when override is available and appropriate. Accordingly, policy override cannot simplistically rely on social controls that are already established in an environment. The technical mediation appears to hamper the direct transferring of social controls from reality to the system. Instead, policy override requires adequate organizational measures in addition to the technical mechanism. We need to further explore these organizational measures in future research.

Q4.1 *In what ways does the satisfaction of functional staff interrelate with functional disruptions, security awareness and expertise, and organizational culture?* / Q4.2 *How is satisfaction impacted for the authoring and making of policies?* As a result of the reduced interference and the more convenient ways of completing work, policy override was generally seen positively

[3]We demonstrate one approach to policy-override decision-support in Appendix F.

by functional staff (Req 4, P5). Policy makers were less content with policy override, since the high number of override events required a considerable amount of effort. They missed the adequate tools that would enable efficient and effective control over override activities (Req 4). In future research, we should explore how the satisfaction of staff and policy makers changes from a usage pattern as intended and effective and efficient review tools, respectively.

Q1.1 *What type of organization requires what authorization measure and what are the influencing factors?* For the observed case – a medium-sized enterprise with only a limited amount of formal structure –, the study indicates a reduction of the overall effort expended for the authorization measure due to less need for policy changes (P2, P7). This points to override being a useful design aspect for these kinds of organizations. However, we also noticed the side effects from the unintended usage pattern, potentially reducing the effectiveness of the authorization measure (Req 1). Further research on the adequacy of policy override thus not only needs to cover a broader range of organizational contexts, but also how improvements are affected from a more intended usage pattern. We may then consider quantitative evaluation of the advantages as well.

More specifically, a number of insights can be derived on acceptable circumventions (Req 5), monitoring (Req 7), and model (Req 12) and mechanism usability (Req 14):

Q5.2 *How must mitigations to undesirable circumventions be designed to be effective and efficient?* Policy override was shown to be a viable way of reducing unintended and uncontrolled circumventions. Functional staff chose to use policy override rather than sharing passwords (P5). However, the study also demonstrated how without adequate organizational measures, such as procedures to react to override use and awareness building for when override is adequate, override will be used in unintended ways.

Q7.4 *How can problems with the authorization measure be extracted from activity logs?* As we observed in the study, the monitoring and review of policy-override activities was hindered in its effectiveness by a lack of usability of the necessary tools (P2). As stated in the recommendations on override in practice, an adequate tool needs to summarize activities in a comprehensible way for effectiveness and efficiency.

Q12.3 *How effective are flexible models in practice regarding security, exceptional situations, and the reduction of disruptions?* The study showed that the flexibility brought about a number of improvements for the authorization measure and its adequacy in case of changing business processes and exceptional situations. However, it also requires accompanying measures to ensure an overall gain in security effectiveness and efficiency.

Q14.1 *How can the mechanism leverage the flexibility of models, foster change requests, and integrate functional staff with policy makers and authors?* One problem for the use of policy override was for the functional staff to realize in which situations override may be of help (P1). This demonstrates that the mechanism needs to be appropriately integrated and provide the necessary information. Future research needs to address how to visualize the additional possibilities through override as part of the default user interface, for example, through explanations of denials (Kapadia et al., 2004; Bonatti et al., 2006; Becker and Nanz, 2008).

9. The twisted paths of obtaining permission: Integrative authorization processes

HAL: This mission is too important for me to allow you to jeopardize it.

Kubrick: *2001: A Space Odyssey* (1968)

The Authorization Problems Study in Chapter 5 demonstrated that the adequacy of an authorization measure depends to a large extent on the adequacy of the surrounding procedures for requesting and granting permissions: The dynamics of organizations require the frequent adaption of the policy (Whalen et al., 2006). We saw that if the lead time or the effort for policy changes is too high, functional staff might not request the changes, but rather fail to comply with the security policy (cf. Beautement et al., 2008). This problem also recurred in the study on policy override in Chapter 8, in which the lack of adequate processes created security risks due to missing reactions on the overuse of policy override, and due to functional staff not requesting the necessary changes to the policy. Similarly, Sinclair et al. (2008) described that entitlements often remain assigned despite organizational changes. Moreover, policy management can suffer from problems in the interaction of policy makers and authors (Bauer et al., 2009). Thus, adequate authorization procedures are crucial for the organizational productivity and security (Req 1).

However, we also observed in the Authorization Problems Study how diverse authorization contexts are, reflecting the characteristics of the organization and the information systems. Small, informally-run companies benefit from a small organizational overhead, while highly structured, large enterprises require reliable, traceable processes. The problem is thus what the adequate form of authorization measures is for a specific context (Q1.1) and how we can support its design (Q1.2). It is particularly important to arrive at the adequate degree of formalization and centralization of the authorization processes (Q10.1, Req 10). To keep the overhead at an acceptable level, it may be necessary to integrate authorization processes with further IT management processes (Q10.3). As indicated by the study on policy override, the threshold for functional staff needs to be low (Q9.1, Req 9). For regulated contexts, process designers must consider how to fulfill regulatory requirements through adequate processes (Q3.2, Req 3).

A very important aspect is the interaction of the involved stakeholders: One reason for the problems in the policy-override study was that there was insufficient interaction between functional staff, and policy makers and authors. Similarly, Bauer et al. (2009) found that it can be problematic that multiple stakeholders are involved and that the decisions on the policy are separated from the implementation (Q8.1, Req 8). Likewise, reviews need to be coordinated so that policy makers have the necessary information to judge on the necessity of permissions (Q7.1, Q7.2, Req 7). This also relates to the problem of how policy decisions are taken and by whom (Q6.1, Req 6).

The interaction is not only important between functional staff and policy makers and authors, but also with developers (Q15.2, Req 15). The evaluation of the controls integration for the authorization framework in Section 7.5 demonstrated that it is difficult to achieve a complete separation of concerns between enforcement and the policy, despite the framework's loose-coupling approach. Flechais

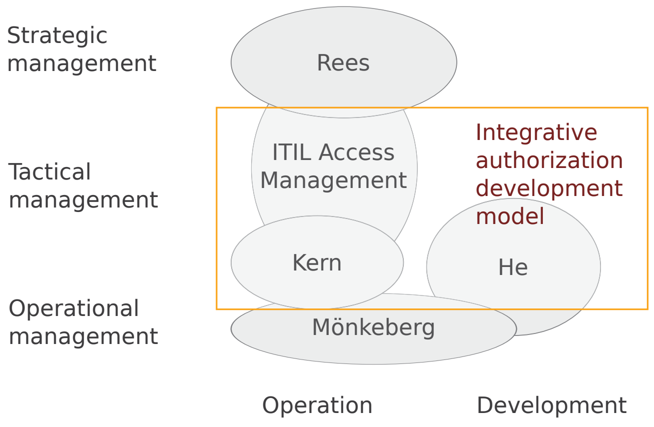


Figure 9.1.: Authorization development processes

and Sasse (2009) argue for security usability that all the different stakeholder perspectives need to contribute their tacit knowledge for a usable measure.

Despite these challenges regarding the interactions, the field of research on authorization is missing a broad and integrative perspective on the processes. Rather, the field is often selectively focused on developing policies or on the systems-engineering part of implementing authorization in systems. Systems-development aspects are often seen only as one step in the policy-development process, such as by Rees et al. (2003), instead of as a process in its own right. To improve the understanding of authorization processes and foster the interaction of stakeholders with different perspectives, the *Integrative Authorization Development Model for Usability* (INDUSE) is proposed in this chapter[1].

This chapter begins by briefly discussing existing authorization process models and how the field of organizational communication approaches processes. Subsequently, INDUSE is described through authorization activities, interrelations, and the derived process definitions. We also consider how INDUSE interacts with other IT management processes. Particularly, we study how well the model applies to authorization processes in practice for a broad range of organizational contexts.

9.1. Authorization process models

There are several existing authorization development processes that exclusively focus on the policy lifecycle and provide for specific aspects of the lifecycle at different management levels as depicted in Figure 9.1. According to Dai and Alves-Foss (2002), the policy lifecycle encompasses the establishment, maintenance, and analysis on the tactical level as well as the specification, refinement, and integration of policies on the operational level. Rees et al. (2003), instead, propose a framework for the security policy lifecycle on the strategic level closer to business management, implementing a Plan–Do–Check–Act cycle.

On a tactical level, but from an IT-operations perspective, the ITIL process "Access Management" includes receiving and validating change requests, granting permissions, and monitoring the operation (OGC, 2007c). Closer to operation is ISO/IEC 29146 "A framework for access management", which will focus on technical aspects of the policy management processes when completed (ISO/IEC JTC 1/SC 27 Secretariat, 2009). However, those approaches are too narrowly focused on

[1]This chapter is based on a publication in the proceedings of the *International Conference on Network and Systems Security* (NSS 2011: Bartsch, 2011c), which has been revised and extended for inclusion here.

the policy management to capture the full breadth of authorization procedures in practice and lack the interrelation with functional users and systems development. Kern et al. (2002) define a lifecycle for role-based authorization and consider the interrelation between stakeholders from administration and development, albeit, on a technical level. More practical and even closer to the technical level, Mönkeberg and Rakete (2000) describe a system and process for the management of roles for several parallel systems.

General security-management processes have broader approaches, but consider authorization only abstractly. ISO/IEC 27002:2005 (2005) provides guidance on achieving security goals in several areas, including access control and human-resources security[2]. In IT management, ITIL (OGC, 2007c) and CobiT (IT Governance Institute, 2007) define processes and activities that can support authorization management, for example, request fulfillment, incident/problem management, and configuration management.

On the development side, He and Antón (2009) suggested the integration of authorization policy design in the systems-development and requirements-engineering process, but neglect to provide for the authorization procedures as part of the operation of the system. As a more general process models for secure systems development, SSE–CMM only touches upon authorization as part of the controls to be implemented (ISSEA, 2003).

The discussed process models are either too abstract to adequately describe the actual interrelations in authorization procedures in practice, or they focus on specific areas or technologies and, thus, cannot support the analysis of the interrelation of different perspectives. Instead, an integrative approach is required, as proposed with INDUSE.

9.2. Organizational communication

The interactions between different actors to request and change permissions are primarily necessary due to the separation of different tasks in the process. Stakeholders with the functional perspective, in general terms, may not grant the permissions to themselves, but need to request them, leading to a number of steps until the permission is technically granted. We can distinguish two reasons for this: First, since authorization is a means of enforcing authority, the assignment of permission must obviously be separated from the use of the permission, often following the differentiation of work into “managerial” and “labour,” one dimension of differentiation by Weber (1947). Second, the process of changing permissions can be sufficiently complex that it requires specialist expertise. Complexity of work leads, according to Smith (1776), to the differentiation of labor for increased productivity. This differentiation requires coordination and integration of the tasks, leading, in turn, to organization and the definition of processes (Monge and Contractor, 2000; Weber, 1947).

In Chapter 4, we predicted how the degree of differentiation and further characteristics of organizational configurations influence the authorization measure. In Chapter 5, we then observed in the Authorization Problems Study the actual effects of the characteristics of authorization contexts: Among other, formalization led to high change lead-times and decentralization to informal processes. In organizational communication, Barnard (1938) considers informal organization as “the conditions under which formal organization may arise” (p. 116). Despite this need of informal interactions to base the more formal ones on, he also argues that “informal organization compels a certain amount of formal organization, and probably cannot persist or become extensive without the emergence of formal organization” (p. 117). Organizational communication research has pursued the importance of informal and its relation to formal interaction extensively, for example, mapping email communication in organizations to formal structures (cf. Monge and Contractor, 2000). To similarly analyze

[2]The interrelation of INDUSE with related IT and security management processes is discussed in Section 9.4.

	Problems		Problems		Problems
F.1	18, 19	A.1	8, 18, 19	D.1	1, 2
F.2	18, 19	A.2	17	D.2	1, 2, 3
F.3	18, 19, 20	A.3	18, 19	D.3	5, 6
		A.4	18, 19	D.4	6
		A.5	18, 19, 20	D.5	6

Table 9.1.: Problems addressed by INDUSE activities (cf. Table 5.4)

the formality and centralization of authorization processes, we will start by identifying the activities in processes surrounding authorization measures (the differentiation of work) with the associated roles before we explore actual processes in practice.

9.3. INDUSE: An integrative authorization process model

INDUSE is a descriptive model of authorization procedures in organizations that focuses on the interrelations between stakeholders and activities, fostering the participation of functional staff, policy makers and authors, and developers (cf. Principle P1). It is derived from the authorization problems identified in Chapter 5, and from literature on concrete authorization activities and high-level processes (cf. Sections 2.4.4 and 9.1).

INDUSE goes beyond the existing process models by offering both a broad, integrative perspective on the activities and the level of detail to effectively describe real-world authorization procedures. INDUSE takes a software-engineering perspective on authorization: The model considers the authorization policy an artifact that is adapted with the changing context, with policy defects impacting productivity and security (cf. Chapter 3). The model follows agile development models in that it emphasizes the integration of and communication between stakeholders, and the context-specific tailorability of the process, covering light-weight as well as rather formal procedures (cf. P8, Section 3.6; Highsmith, 2002). Through adequate procedures and improved interaction, the model particularly aims to suit dynamic environments (P7).

In Figure 9.2, the INDUSE activities are shown for the stakeholder perspectives, indicating the expected flows of information between perspectives as interrelations (cf. Tables 9.2, 9.3, and 9.4). The activities address the problems identified in Chapter 5 from the Authorization Problems Study and relevant literature. As shown in Table 9.1, the activities address problems originating in the development, policy authoring and making, and functional perspectives. Of these problems, only Problem 4 (usability of the modification tools) is out of scope of the activities, since these tools are considered to be rather static. The problems of the organization, strategy, and tactics perspectives are more generally covered by applying the INDUSE model (cf. Section 9.3.4).

The perspectives in Figure 9.2 are derived from the general perspectives on authorization (cf. Section 4.4), with policy making and authoring combined into an administrative perspective for brevity. Functional stakeholders primarily use information systems to complete work tasks. Administrative stakeholders decide on and author policies. Developers integrate authorization in information systems. Security tactics and strategy perspectives are outside the scope of INDUSE, but can employ the model to evaluate existing processes for potential improvements and comprehensiveness (cf. Section 9.3.4). The activities are assembled into processes, such as, for example, separate systems-development and policy-management processes, not necessarily adhering to a strict sequence of the activities.

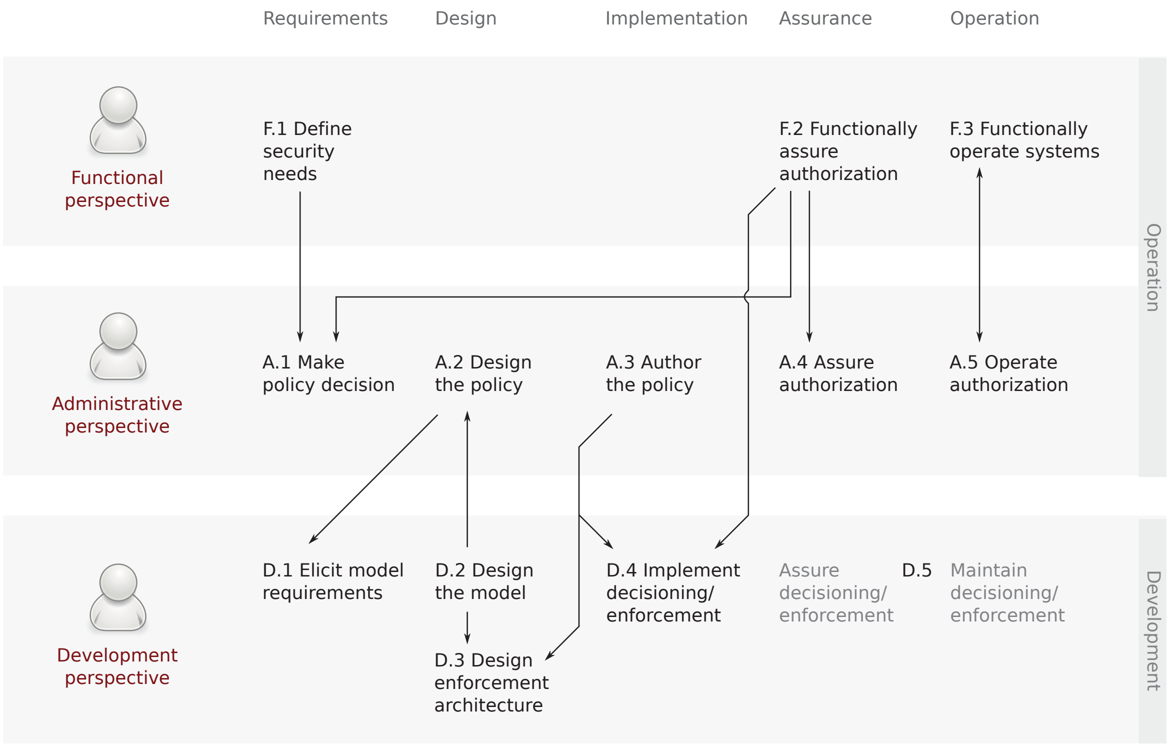

Requirements
Design
Implementation
Assurance
Operation
Functional perspective
Administrative perspective
Development perspective
F.1 Define security needs
F.2 Functionally assure authorization
F.3 Functionally operate systems
A.1 Make policy decision
A.2 Design the policy
A.3 Author the policy
A.4 Assure authorization
A.5 Operate authorization
D.1 Elicit model requirements
D.2 Design the model
D.3 Design enforcement architecture
D.4 Implement decisioning/ enforcement
Assure decisioning/ enforcement
D.5 Maintain decisioning/ enforcement
Operation
Development

Figure 9.2.: INDUSE activities and interrelations

F.1 → A.1	Task descriptions and security needs
F.2 → A.1	Policy decision problems
F.2 → A.4	Policy specification problems
F.2 → D.4	Defects related to decisioning and enforcement
F.3 → A.5	Indications of inadequate restrictions

Table 9.2.: Information flows from the functional activities

9.3.1. Functional activities

F.1. Define security needs

To enable well-founded decisions on what permissions to grant, functional stakeholders need to provide information on the security needs of resources, including who requires the availability of resources to carry out functional tasks and how critical the integrity and confidentiality of the resources are (ISSEA, 2003)[3]. Eliciting the needs can be supported through task analysis (Hollnagel, 2006) and should also take the system's purpose and the legal context into account (ISSEA, 2003). According to ISO/IEC 27002:2005 (2005), business requirements for authorization include the risks that information is facing, relevant legislation, contractual obligations, and general compliance requirements.

[3]Compare the definition of security needs as part of the Authorization Taxonomy in Section 4.1.

A.1 → A.2	Policy decisions
A.2 → A.3	Policy design
A.2 → D.1	Model expressiveness requirements for policy design
A.3 → A.2	Problems with policy design
A.3 → A.4	Policy specification
A.3 → D.3	Specific requirements on controls architecture
A.3 → D.4	Specific requirements on provisioning and controls
A.4 → A.1	Problems with decisions
A.4 → A.2	Policy design problems
A.4 → A.3	Policy specification problems
A.5 → A.4	General authorization problems
A.5 → F.3	Dissemination and disciplinary measures

Table 9.3.: Information flows from the administrative activities

F.2. Functionally assure authorization

Functional stakeholders can be integrated into the assurance (in contrast to administrative assurance in A.4) and assess whether policy changes are adequate concerning their primary tasks, functionally assuring changes (ISO/IEC 27002:2005, 2005). Typical means are acceptance tests (Boehm, 1988) and reviews of authorization policies, either as a sign-off step in the process or in regular self-reviews (cf. Sinclair et al., 2008). Information on discovered defects need to flow into administrative and development activities, as listed in Table 9.2.

F.3. Functionally operate systems

The goal of functional stakeholders is to complete their primary tasks. The functional operation enables them to cope with interference from authorization measures, for example, when they are unable to complete a task because of missing permissions, and provides efficient and effective ways to address the problems, for example, through change requests. In certain contexts, the efficiency losses due to inadequate authorization policies can be reduced by implementing more flexible authorization models. This includes delegation mechanisms that allow staff to grant the permissions to those usually not permitted (Ahn et al., 2003; Brucker et al., 2009). Policy override, as described in Chapter 8, goes one step further and enables staff to temporarily extend permissions on their own (Cederquist et al., 2007; Denley and Smith, 1999; Povey, 2000). In contrast to the acceptance tests in the previous activity, this activity targets the continuous challenge of functional stakeholders in practice.

9.3.2. Administrative activities

A.1. Make policy decisions

Policy makers need to make decisions on how the policy should govern the activities of users in systems, for instance, only allowing specific users to read sensitive data. The decisions may be taken on different levels of abstraction: high-level, business-focused ("enforce the need-to-know principle") as well as low-level decisions focused on specific permissions and role assignments ("HR users may access personnel data"). The decisions are based on the functional security needs and adhere to the strategics on how to take policy decisions. Policy makers often apply the *need-to-know* and *least-privilege* principles. Decisions can also be based on organizational structures in top-down approaches (Crook et al., 2003) and often also account for the risks involved (ISSEA, 2003;

ISO/IEC 27002:2005, 2005). One control to elicit the risks is the screening of employees (ISO/IEC 27002:2005, 2005, 8.1.2).

A.2. Design the policy

The design of the policy defines the overall approach to authoring the policy, including, for example, how the organizational structures should be mapped to the policy (Crook et al., 2003). For role-based authorization, the design can be derived in role mining from the users' tasks or permissions (Bertino et al., 2008) as part of a role engineering process (Neumann and Strembeck, 2002; Kern et al., 2002).

A.3. Author the policy

Adhering to the policy design, policy authors actually implement the policy and, for example, assign permissions to roles and roles to users. The policy specification occurs on multiple layers: Close to the strategic layer, policies consist of abstract business rules, while, for example, role authoring and role assignment result in more concrete policies. Ideally, policy changes should be traceable, for instance, through configuration management (ISSEA, 2003, BP 01.02).

A.4. Assure authorization

To verify that authorization has the intended effect, policy authors and makers can conduct assurance activities to guarantee, for example, that an implemented policy complies with higher-level policies and that the concrete changes correspond to the decisions. Assurance can be conducted on the policy decisions, the policy design, and specification. Policy makers can conduct policy reviews, as sign-off of changes or at regular intervals (ISO/IEC 27002:2005, 2005). Static or dynamic analysis can be conducted for implemented policies to assure specific characteristics. Validation and verification plans can define what artifacts should be assured by whom with what practices and at what points (ISSEA, 2003).

A.5. Operate authorization

The operation of authorization supports the technical authorization mechanism. Close to the mechanism, the monitoring of access violations, abnormal activities, and non-compliance with security policies can help to identify inadequate policies and other defects (ISO/IEC 27002:2005, 2005, 13.1.1). One consequence from incidents is to trigger assurance activities to improve existing policies and more generally learn from the incidents (ISO/IEC 27002:2005, 2005, 13.2.2).

On a management level, incidents can require disciplinary measures as deterrent and further containment activities (ISO/IEC 27002:2005, 2005, 8.2.3). Appropriate training and awareness campaigns can prevent incidents in the first place (ISO/IEC 27002:2005, 2005, 13.1.2). Roper et al. (2005) distinguish between the approaches of training, education, awareness, and motivation to influence the behavior with respect to security. For disciplinary and dissemination measures, interaction with functional staff (F.3) is necessary.

9.3.3. Systems development activities

D.1. Elicit authorization model requirements

Development stakeholders define the requirements that the authorization model needs to fulfill with respect to the targeted policy design: For instance, whether a role-based policy should be imple-

D.1 → D.2	Authorization model requirements
D.2 → A.2	Available policy constructs
D.2 → D.3	Decisions to be enforced for authorization model
D.2 → D.4	Authorization model
D.3 → D.4	Enforcement architecture
D.4 → D.5	Decisioning and enforcement implementation
D.5 → D.2	Authorization model defects
D.5 → D.3	Enforcement architecture defects
D.5 → D.4	Decisioning and enforcement defects

Table 9.4.: Information flows from the development activities

mented or how dynamic the policies are. This interrelates closely with the policy design activity (A.2), which informs the required authorization constructs used in the authorization policy.

D.2. Select/design authorization model

Development stakeholders have to choose an adequate authorization model that addresses the requirements from D.1. Existing authorization models include identity-based (e.g. ACL) and mandatory (e.g. BLP: Pernul, 1995), role-based (Ferraiolo and Kuhn, 1992), attribute-based (e.g. XACML), and a broad range of other approaches (Samarati and de Vimercati di Vimercati, 2001; Antoniou et al., 2007), as well as extensions, regarding, for example, delegation in RBAC (Ahn et al., 2003; Zhang et al., 2003).

D.3. Design enforcement architecture

For the enforcement architecture, developers design how the authorization decisions are enforced in the program control flow, for example, implementing reference monitors for complete mediation (Anderson, 1972; Saltzer and Schroeder, 1975). Enforcement approaches include runtime-system mechanisms, such as in Java (Gong and Ellison, 2003), aspect-oriented programming (AOP: Kiczales et al., 1997; Ancona et al., 1999; Ray et al., 2004), and enforcement hooks in the source code (Gong and Ellison, 2003; Jaeger et al., 2004). Typically, the architecture implements a separation of concerns for the authorization decision and enforcement (Beznosov et al., 1999; De Win et al., 2003).

D.4. Implement decisioning and enforcement

Based on the algorithms defined by the authorization model, the decisioning is implemented and derives whether to allow or deny an access request, based on a given policy, for example, whether specific data may be displayed to a user. For the enforcement of the decisions, developers often insert enforcement controls in the program control-flow (Gong and Ellison, 2003; Jaeger et al., 2004). When integrating a system in an existing authorization infrastructure, the central policies need to be distributed to the system and mapped to the respective authorization model (Mönkeberg and Rakete, 2000).

D.5. Assure and maintain decisioning and enforcement

The authorization-specific assurance and maintenance practices follow the typical secure systems development (cf. Section A) and are thus consolidated in INDUSE. Assurance should be based organizationally on a verification and validation plan (ISSEA, 2003) and technically on dynamic (testing) and static analysis (Ganapathy et al., 2006). Maintenance similarly requires secure systems development practices, for example, vulnerability management (ISO/IEC 27002:2005, 2005).

9.3.4. Strategic and tactical perspectives

INDUSE supports the design and evaluation of authorization processes (tactics), including the definition of how decisions are taken (strategy). The stakeholders from the tactical perspective compose processes from sequences of activities, assigning responsibilities (ISO/IEC 27002:2005, 2005, 6.1.3), as well as defining process inputs and outputs, and process performance indicators (OGC, 2007a). Process triggers act as the starting points for policy changes, for example, from organizational or from functional software changes.

Often, multiple interrelated processes need to be established in parallel. In small organizations with one systems development project, one operation (policy management) and one development process may suffice. For larger organizations with several levels of hierarchy, layered processes are more appropriate. For example, high-level policies at a strategic level, close to business goals can be managed in a process independent from the changes to roles that follow organizational changes to accommodate the different change frequencies. On the development side, one development process could be instantiated per software development or integration project.

9.4. Interrelation with IT management processes

Authorization processes are not executed in pure vacuum. Instead, organizations typically establish related processes for designing and maintaining IT systems (IT management) to control risks to information systems (Information Security Management Systems, ISMS), to manage changes to information systems (configuration or change management), and to develop the systems (secure systems development). To explore the interrelation with existing processes and the re-use of artifacts from related processes, we consider the touch points of INDUSE with process models from those areas.

The method applied to arrive at a mapping is to explore the major guideline or standard documents for each area, either relating each given procedure or activity to an INDUSE activity or noting it as not covered. There are two primary types of interrelation: A procedure or activity can either be a high-level definition of the INDUSE activity and trigger the latter, or an enabling method that produces an output to be employed in an INDUSE activity.

9.4.1. IT management

Two very broad frameworks for the IT management processes are ITIL and CobiT. The IT infrastructure library (ITIL, Version 3, OGC, 2007b) defines multiple layers of IT service management, including the Service Strategy, Design, and Operation (cf. Table 9.5). As part of the service definition, broad functions and processes can trigger the authorization activities that are suggested by INDUSE. For example, ITIL application management can result in systems development activities and service change management in activities on the operations side. Conversely, ITIL also provides enabling functions, processes, and activities that support authorization development activities. Examples are the request fulfillment, event/incident/problem management, and configuration management. Apart from the general IT management processes, ITIL also defines a specific authorization process with the Access Management process. Generally, ITIL provides broad support of the INDUSE activity Operate authorization (A.5), but only shallow and high-level support on the policy authoring (A.3), systems development, and the functional perspective. Most direct interrelations are on the Service

Transition and Service Operation level. Service Strategy, Service Design, and Continuous Service Improvement can support the definition and improvement of the overall INDUSE process.

The Control Objectives for Information and related Technologies (CobiT, IT Governance Institute, 2007) more directly focus on the development and maintenance of IT management processes. It is structured in four primary domains, Plan and Organize, Acquire and Implement, Deliver and Support, and Monitor and Evaluate. Similar to ITIL, CobiT defines processes, such as "Manage changes" and "Acquire and maintain application software", that will trigger INDUSE activities. Again, many of the INDUSE-related CobiT activities can serve as enabling processes to INDUSE activities. The interrelation is most pronounced for policy decisions (A.1) and operating authorization (A.5), while the interrelation for development is rather at a high level. Many high-level IT management processes of CobiT, mostly from "Plan and Organize", "Deliver and Support", and "Monitor and Evaluate", provide support for the process management and can be applied to INDUSE processes together with the CobiT process control objectives and maturity model.

9.4.2. Information security management

Information security management defines processes to continuously improve an organization's information security. On a high level, these management processes follow a *Plan–Do–Check–Act* (PDCA) cycle to guarantee the systematic and continuous planning, implementation, assurance, maintenance, and assessment of security measures. The ISO/IEC 27001:2005 (2005) standard "Information security management systems – Requirements" defines the requirements for Information Security Management Systems (ISMS), employed, for example, for the certification of ISMS processes. Most of the process management requirements in ISO 27001 are out of scope of INDUSE because of their high-level management perspective. INDUSE-based authorization processes integrate with a ISO 27001-compliant ISMS and complement the ISMS processes.

ISO/IEC 27001 is mostly concerned with the process management. Conversely, ISO/IEC 27002:2005 (2005) on the "Code of practice for information security management" is closer to practically enacting the security goals and thus to INDUSE, relating specifically to access control, risk assessment, security policy, human resources security, and the information systems lifecycle. ISO 27002 primarily offers enabling procedures. A projection of ISO 27002 practices to INDUSE activities is shown in Table 9.6. ISO 27002 overlaps with INDUSE on the security needs (F.1) and policy decisioning (A.1), specifying best practices on how to elicit high-level requirements, assess risks, and consider compliance. For the administrative perspective, support can be found for the authorization operation (A.5). In addition, there are practical recommendations on access control in systems development and general development aspects. Since ISO 27002 has a very broad scope – for example including physical security – much of the standard is out of scope for authorization development, since INDUSE is foremost targeted at authorization in information systems.

The ISF Standard of Good Practice (SGP, ISF, 2007) is similar to ISO 27002 in that it advises approaches of how to practically achieve adequate information security. The SGP document structures the practices into multiple perspectives, of which end-user environment, security management, critical business applications, computer installations, and systems development relate to INDUSE. In detail, there is similar overlap as with ISO 27002, although SGP has a broader scope in the area of systems development.

ISO/IEC 27005:2008 (2008) defines the requirements for a risk assessment process. In INDUSE, risk assessment is foremost applied as part of the policy-decisioning activity (A.1), for which ISO 27005 provides high-level support.

INDUSE activity	ITIL	CobiT
F.1 Security needs	Requesting access (SO/AM/E)	Information architecture (PO2/E); Ensure compliance with external requirements (ME3/H)
F.2 Assurance	Verification (SO/AM/E)	
F.3 Operation		
A.1 Policy decisions	Providing rights (SO/AM/E); Removing or restricting rights (SO/AM/E); Request fulfillment (SO/E); Change management (ST/H)	Assess and manage IT risks (PO9/E); Manage changes (AI6/H); Ensure systems security (DS5/E)
A.2 Policy design	Providing rights (SO/AM/E)	
A.3 Authoring	Technical Management (SO/E); IT Operations Management (SO/E); Configuration Management (ST/E)	Manage the configuration (DS9/E)
A.4 Assurance	Service validation and testing (ST/H)	Ensure systems security (DS5/E), compliance with external requirements (ME3/H)
A.5 Operation	Monitoring identity status (SO/AM/E); Logging and tracking access (SO/AM/E); Event/incident/problem management (SO/E); Service Desk (SO/E); Technical Management (SO/E)	Enable operation and use (AI4/E); Ensure systems security (DS5/E); Educate and train users (DS7/E); Manage service desk and incidents (DS8/E), problems (DS10/E)
D.1 Model requirements	Application Management (SO/H); Configuration Management (ST/E)	
D.2 Model design	Application Management (SO/H)	
D.3 Enforcement architecture	Application Management (SO/H); Configuration Management (ST/E)	Manage the configuration (DS9/E)
D.4 Implementation	Application Management (SO/H); Configuration Management (ST/E); Deployment Management (ST/H)	Acquire and maintain application software (AI2/H), technology infrastructure (AI3/H); Install and accredit solutions and changes (AI7/H); Manage the configuration (DS9/E)
D.5 Assurance & maintenance	Application Management (SO/H); Deployment Management (ST/H); Service validation and testing (ST/H)	Acquire and maintain application software (AI2/H); Enable operation and use (AI4/E); Install and accredit solutions and changes (AI7/H); Ensure systems security (DS5/E); Manage problems (DS10/E)

Table 9.5.: Interrelation of INDUSE activities with IT management processes
ITIL: ST: Service Transition, SO: Service Operation, AM: Access Management;
CobiT: PO: Plan and Organise, ME: Monitor and Evaluate, AI: Acquire and Implement, DS: Deliver and Support;
Type of interrelation: E: enabling, H: high-level process

INDUSE activity	ISO 27002	ISO 10007	ISO 21827/SSE-CMM
F.1 Security needs	Organization responsibilities (6.1.3); Responsibilities for asset management (7.1); Human resources responsibilities (8.1.1), screening (8.1.2); High-level, business requirements (11.1); Compliance (15.1)		Security needs (PA10); Responsibilities (BP01.01); Awareness (BP01.03)
F.2 Assurance	System acceptance (10.3.2)		Verification/validation (PA11)
F.3 Operation	Human resources: management (8.2.1), awareness (8.2.2)		Awareness (BP01.03)
A.1 Policy decisions	Risk assessment (4.); Risks from externals (6.2.1); Removal of permissions (8.3.3)	Identification (5.3); Change control (5.4)	Awareness (BP01.03); Impact (PA02), risk (PA03), threat (PA04); Security input (PA09)
A.2 Policy design			
A.3 Authoring	Privilege management (11.2.2)	Identification (5.3); Change control (5.4)	Security configuration management (BP01.02)
A.4 Assurance	Review of access rights (11.2.4); Auditing controls (15.3.1)	Change auditing (5.6)	Manage controls (BP01.04); Verification/validation (PA11); Assurance argument (PA06)
A.5 Operation	Disciplinary process (8.2.3); Incident management: reporting events (13.1.1) and weaknesses (13.1.2); Learning from incidents (13.2.2);		Security posture (PA08); Security input (PA09)
D.1 Model requirements	Systems security requirements (12.1)	Identification (5.3); Change control (5.4)	Security configuration management (BP01.02)
D.2 Model design			
D.3 Enforcement architecture			
D.4 Implementation	Asset labeling and handling (7.2.2); Application access restrictions (11.6.1); Access control to program source code (12.4.3); Change management (12.5.1)	Identification (5.3); Change control (5.4)	Security configuration management (BP01.02) Awareness (BP01.03)
D.5 Assurance & maintenance	Auditing controls (15.3.1); Technical vulnerability management (12.6)	Change auditing (5.6)	Manage controls (BP01.04); Verification/validation (PA11); Assurance argument (PA06)

Table 9.6.: Interrelation of INDUSE activities with security, configuration management, and development processes (with references to sections in the standards)

9.4.3. Configuration and change management

Configuration and change management generally affects the authorization processes in two ways. First, change management is an integral part of systems development and thus affects the authorization decisioning and enforcement development by triggering policy changes as a high-level process. Secondly, authorization policies and related documents need to be included in the change management to provide traceability of authorization policy-decisioning and authoring (enabling/supporting). Joeris (1997) presents a conceptual model of change management for software configuration management. In his model, change management may be process- (activity, change) or product- (product-state, object) centered. Configuration management is broader in scope, encompassing all products in an organization, but focuses on the integration aspects.

ISO 10007:2003 (2003) "Quality management systems – Guidelines for configuration management" describes configuration management processes at a high level. Table 9.6 shows how the ISO 10007 process activities can be mapped to the INDUSE activities. Configuration management affects INDUSE in the two ways indicated above: The authorization policy and program source code may be seen as a configuration artifact under configuration management together with the related documents, such as requirements. The primary aim is to provide increased reliability and accountability for changes applied to the artifacts. The configuration items need to be identified and placed under change control for all INDUSE activities that produce evolving artifacts. In assurance activities, change audits are conducted. The practices from ISO 10007 offer guidance on how to implement the related INDUSE activities from the change-management perspective (enabling/supporting).

9.4.4. Secure systems engineering

INDUSE significantly overlaps with systems development processes. General software engineering processes as described in Sections 3.3 and A.1 relate only at a high level to the INDUSE activities, though. Closer are secure systems engineering processes. The Systems Security Engineering Capability Maturity Model (SSE-CMM), published as the ISO/IEC 21827 standard (ISSEA, 2003), describes a broad secure-systems engineering model. SSE-CMM is foremost described through security engineering practices, called "Base Practices", from 22 process areas (PA). Additionally, "Generic Practices" define capability-related practices, such as "allocate resources", that follow the CMM approach and are orthogonal to the base practices.

A projection of SSE-CMM practices to INDUSE activities is shown in Table 9.6. SSE-CMM covers authorization development on a high level, but does not provide equivalent base practices for all INDUSE activities. For example, the functional operation activity (F.3) has no high-level equivalent, although awareness practices support the INDUSE activity. Detailed approaches can be found for requirements elicitation, policy decisioning, and assurance activities.

Since SSE-CMM covers a broader scope, several of SSE-CMM's Base Practices have no counterpart in INDUSE. PA07 and PA12 to PA22 apply to project management and the organization, out of scope of INDUSE. Similarly, the Generic Practices may be employed to assess the capability maturity. In general, SSE-CMM appears to complement INDUSE well with process management activities, particularly for organizational, project management, and process assessment aspects. Moreover, Base Practices on requirements engineering, risk assessment, and assurance can serve as implementation guidance.

Case	Systems	Type of development	Users	Roles
Large bank	200 banking systems	Custom and COTS	50000	n/a
Central uni. IT	E-learning, email, ...	Custom and COTS	37000	10
Hospital	Numerous medical systems	COTS	3500	900
Regional bank	Ca. 150 banking systems	Custom and COTS	3000	100
Food industry	Navision-based ERP	Individualization	1000	200
University admin.	SAP-based ERP	Individualization	200	800
Charity org.	Business Web application	Custom	150	20
Quality assurance	Business Web application	Custom	100	16

Table 9.7.: Authorization contexts

9.5. Authorization processes in practice

Since there is little prior research on authorization processes in organizations, we study in the following the processes relating to authorization. The primary purpose of the study is to systematically explore the processes in diverse environments and provide a basis for further research into authorization procedures. This includes studying (a) how policy changes occur in practice, and (b) how actors and activities interrelate. As secondary aims, the study evaluates the ability of INDUSE to describe diverse authorization processes and to support gap analyses by indicating potentials of improvement.

9.5.1. Study design

Research instruments

The primary source of the study are semi-structured interviews of between 30 and 90 minutes with stakeholders close to IT or organizational management (subjective empirical case studies, cf. Section 2.5). We chose a qualitative approach with interviews, since these are beneficial for exploratory work (Cairns and Cox, 2008, cf. Section 5.1.1). The interviews covered the authorization context (type of system, number of users), the policy-change procedures (change activities, responsibilities), and the interrelation with systems development (systems integration, enforcement implementation). As a secondary source, we reviewed process documents relating to authorization for two of the studied organizations.

Sampling

Organizations are generally reluctant to disclose information related to the sensitive subject of information security unless there is a trust relationship with the researching party, making broad empirical studies in this field difficult. Kotulic and Clark (2004) advise to focus on a small number of in-depth analyses with carefully selected organizations. In this study, we likewise focus on a small number of cases, applying a careful, “information-oriented” sampling (cf. Section 7.3.1) to cover a broad field of organizational contexts (cf. Section 4.2). We operationalized the authorization contexts into two sampling dimensions: Organizational complexity (number of users) and estimated risk-level (business risks, regulations). Eight organizations were individually contacted through personal contacts for this study (cf. Table 9.7), distributed according to the sampling dimensions as depicted in Figure 9.3.

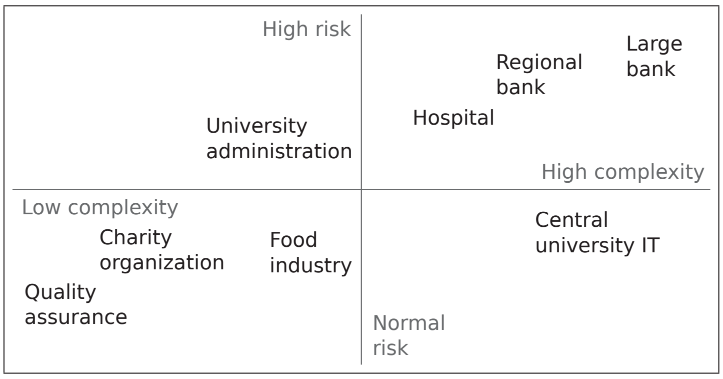


Figure 9.3.: Sampling of organizations in study

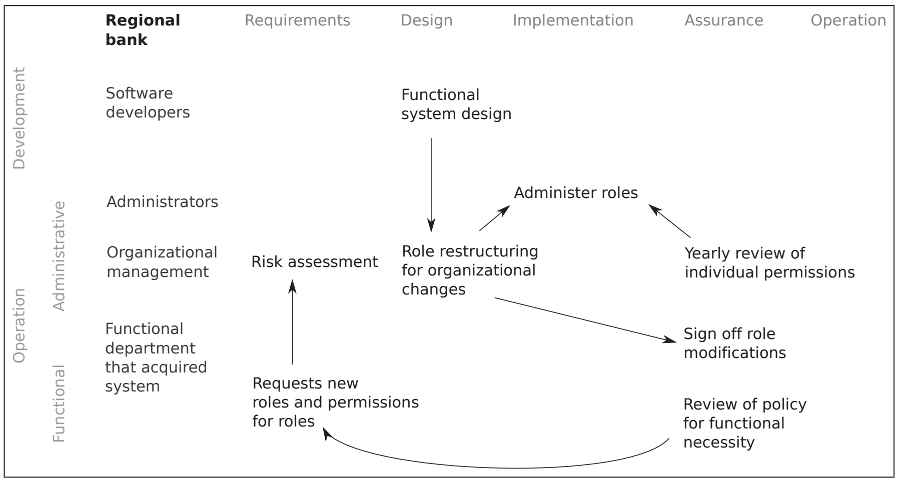

Regional bank
Requirements
Design
Implementation
Assurance
Operation
Development
Operation
Administrative
Functional
Software developers
Functional system design
Administrators
Administer roles
Organizational management
Risk assessment
Role restructuring for organizational changes
Yearly review of individual permissions
Functional department that acquired system
Sign off role modifications
Requests new roles and permissions for roles
Review of policy for functional necessity

Figure 9.4.: Example of procedure visualization based on INDUSE activity model

Analysis with INDUSE

The goal of the analysis is to understand the processes and interrelation of activities concerning authorization, both on the formal and informal level. From the detailed interview notes and the process documents (where available), we extracted data on the processes into a spreadsheet and graphically modeled the activities and interrelations of each case, as shown in one example for the regional bank in Figure 9.4. We systematically analyzed the processes in the following categories:

- *Process characteristics:* Process formality, triggers, and instantiation,
- *Perspectives and stakeholders:* Presence of perspectives, and the respective roles and tasks of stakeholders,
- *Activities and interrelations:* Coverage of INDUSE activities, interrelations between activities.

Study design quality

Following Yin (2009), the quality of the study design may be ensured through four tests (cf. Section 5.1.1). The *construct validity* is ensured in this study by employing the INDUSE framework for

Case	Systems dev. proc.	Operational process	Degree of formality	Process triggers
Large bank	Per-system	Central and per-system	Formal	Software, organizational
Central uni. IT	Local	Central and local	Informal	Software, organizational
Hospital	Per-system	Central and local	Formal	Organizational, review
Regional bank	Per-system	Central	Formal	Software, organizational, review
Food industry	Central	Central	Informal	Software, hindrances
University admin.	Central	Central	Formal	Software, organizational, hindrances
Charity org.	Central	Central	Informal	Software, hindrances
Quality assurance	Central (w/role modifications)	Central (role assignment)	Informal	Software, hindrances

Table 9.8.: Authorization process characteristics, cases ordered by number of users

analysis, which is derived from a broad set of literature on authorization. The *internal validity* is addressed by the use of rich semi-structured interviews, which allow us to conclude causalities from the descriptions of the interviewees. While the subjectivity of interviews may cause in places imprecise individual descriptions, the larger picture as a hypothesis for more comprehensive research should be unaffected. The *external validity* is affected by the limited number of cases in the study. However, the careful sampling should result in a good spread of cases to provide a solid description of current authorization processes, even though neither comprehensive nor representative. The generalizability is naturally limited to the covered range of cases, as described above. Lastly, the *reliability* of the study is ensured by carefully collecting and systematically analyzing the data based on the INDUSE framework. Thus, the study should be able to provide a rich hypothesis for future research on authorization procedures.

9.5.2. Findings on authorization procedures

Process characteristics

Depending on the organizational context, authorization processes are defined in varying degrees of formality, from implicit ("lived") processes to those laid out in process documents. As shown in Table 9.8, larger organizations in this study are more likely to have a formally defined process. Of the four largest contexts, only one, the central university IT, has an informal process, largely because of the focus on the identity-management and infrastructure part of authorization. Likewise, three of the four smallest contexts only have informal processes.

Four different triggers for policy changes can be observed in the organizations (cf. Table 9.8):

1. Functional software changes (7 cases) that, for example, require new roles for added functionality or require role modifications to preserve behavior,

2. Organizational changes (5), for example, when business processes are restructured or additional departments need to gain permissions to complete tasks,

3. Hindrances affecting functional stakeholders (4) in their primary tasks,

4. Policy reviews (2) that are regularly conducted to identify inadequate and out-of-date permission assignments.

Case	Functional	Administrative	Development
Large bank	Functional dep.: provides functional role concept, requests change, QA	Functional admin.: role concept; Central admin.: implements changes	Developers: integrate enforcement
Central uni. IT		Admin.: assigns role to function/person; Local admin.: manage role subtree	Developers: design role subtree
Hospital	Line manager: requests role change; key user: signs off; requester: QA	Role admin: coordinates, changes roles; Local admin: assigns role	Developers: integrate external systems
Regional bank	Functional dep: requests role changes, signs off role changes, reviews policy	IT: manages role; Org. management: assesses risk, restructures, reviews	Developers: modify system, request new roles
Food industry	Functional department: informal requirements	Admin: manages roles	Developers: integrate enforcement with functional changes
Uni. admin.	Line manager and key user: request change; func. user: reports defects	Admin: changes roles, validates requests, corrects roles after updates	Developers: implement system update
Charity org.	Functional dep: discusses policy change; func. user: QA, reports defects	Functional consultant: design, implement policy; conduct QA	Developers: implement authorization model and enforcement
Quality assurance	App. owner: requests user story, discuss auth. reqs; end user: report problems	Developer/admin: modifies roles; App. owner: assigns roles	Developers: implement authorization with functionality

Table 9.9.: Stakeholder roles and tasks by perspective

Organizations mostly establish separate processes for systems development and operational policy management (cf. Table 9.8). Only in the agile-development case of the quality-assurance company, the systems development process includes role modifications, although roles are assigned in an operational process. In the further three of four smallest cases, one central software development and one central policy management process are established. In larger cases, separate processes for role modifications, role assignments, and user creation are instantiated, in part running locally, such as in the hospital case. In cases with numerous systems, several systems development processes are established in parallel.

Perspectives and stakeholders

As shown in Table 9.9, in seven of the eight cases, all three INDUSE perspectives are present in the process descriptions. Only the central university IT lacks the functional perspective due to the focus on the infrastructure. Overall, the functional activities often relate to requesting authorization changes (5 cases), either directly or through discussions about functional requirements with development stakeholders. Other activities are quality assurance of changes (3), sign-off of requested changes (2) and reporting inadequate or defective restrictions (3). In two formal cases, technically-trained “key users” support policy changes and reviews. Only in the three smallest cases, functional staff are broadly explicitly incorporated in the process.

Stakeholders from the administrative perspective are either from the central IT departments or decentralized into local functional departments. In the case of the regional bank, organizational management staff is responsible for explicit requirement and design activities. In structured processes, such as the large bank and the hospital case, stakeholders from central IT and local administrators are both involved and interact in administrative activities, with more complex and critical activities centralized. For example, while role assignments are conducted locally in the hospital context by

	F.1	F.2	F.3	A.1	A.2	A.3	A.4	A.5	D.1	D.2	D.3	D.4	D.5	Interrelations
Large bank	x	x		x	x	x					o	x		F.1→A.1→A.2→A.3, A.2→D.4→A.3
Central uni. IT					o	x				o	x	x		*D.2→D.4, D.2→A.2, D.2→A.3*
Hospital	x	x		o		x	x					o	o	F.1→A.1→A.3, F.2→A.3, *D.4→A.3*
Regional bank	x	x		x	x	x	x			x				F.1→A.1→A.2→A.3, A.4→A.3, *D.2→A.2, F.2→F.1*
Food industry	x		x	o		x			o			o		*F.1→A.1→A.3, F.3→A.3, D.4→A.3*
Uni. admin.			o	x		x	x					o		A.1→A.4→A.3, *F.3→A.3, D.4→A.3*
Charity org.	o	x	o	o	x	x	x		o	o	o	x		*F.3→A.1, F.2→A.4→A.2, D.1→A.1*
Quality assurance	x		x	o	o	x				o	o	x		*F.1→A.1, F.3→A.3, F.3→A.1, D.2→A.2*

Table 9.10.: Activities covered by the studied processes (o: partly, x: mostly/fully; informal interrelations set in italics)

the associated role administrators, roles are only modified centrally. Explicitly assigning responsible stakeholders to roles is a common practice for formal contexts (2 of 4 cases).

Authorization-related software development activities are mostly embedded in the general software development activities. In three of the smaller cases, the administrative perspective and the development perspective are partly merged. This results either from in-house development (food industry) or from the policy management as part of the software development (quality assurance).

Activities and interrelations

Assessing which of the studied processes covers which INDUSE activity, as shown in Table 9.10, can offer insights into the process characteristics. Several processes lack or only partly cover the authorization activities in the areas of explicit decision-making (A.1; 3 full, 4 partly of 8), design (A.2; 3/2), and assurance (A.4; 4/0). None of the processes take the monitoring of authorization, disciplinary measures or awareness/training (A.5) into account. The analysis of the activities also indicate specific focuses of the processes. For example, the central university IT has only few functional and administrative activities due to their infrastructure focus, the hospital case few development activities, employing mainly COTS systems.

The formally defined and informal interrelations between activities, listed in Table 9.10, show how diverse the authorization procedures are in practice. In six processes, there are formal or informal flows from security needs (F.1) to the administrative activities. In the other cases, the processes seem to be more development or administration-focused. While interrelations are mentioned between systems development and administrative activities in all cases, for example, to reflect new or changed functionality in roles, eight of the nine mentions are informal ones. In the university administration case, the lack of a formal interrelation frequently causes problems of missing permission in operation after software updates.

9.5.3. INDUSE as a means to describe diverse processes

The study shows how broad the range of authorization processes is in actual organizations. We employed the INDUSE analysis method laid out in Section 9.5.1 to systematically describe and compare the processes. The *process characteristics* supported the categorization of processes (formality, instantiation) and indicated potential improvements (triggers). *Perspectives and stakeholders* showed how the processes are constructed in terms of centralization and how the perspectives are interrelated. *Activities and interrelations* provided details on the focus of the processes and the informal and formal interrelations, suggesting additional formal relations. The informal relations between development and operation show that the integrative modeling approach, including development, is indeed necessary.

While several of the identified interrelations are not directly suggested by INDUSE, most unexpected ones represent short cuts of INDUSE interrelations, for example, F.2→A.3 for the hospital case where INDUSE suggests F.2→A.4→A.3. Unexpected interrelations are present between functional stakeholders (F.2→F.1) for reconsidering functional needs after reviews and from software requirements to policy decisions (D.1→A.1) when functional feature discussions lead to authorization changes. The expected INDUSE interrelations should thus be considered as lower bounds at the current stage and need to be extended in further research.

While the study focused on authorization contexts in organizational environments today, INDUSE should also be applicable to future environments. One development in organizations is increasing its degree of distribution, requiring local decision-making to lower security-management costs (Pallas, 2009). As shown for the hospital case, where stakeholders from functional departments have administrative perspectives, this can be modeled well with INDUSE. Similarly, distributed services (cloud computing) can be modeled as individual systems as in the case of the large bank. Processes in federated authorization architectures can be represented as distributed decision-making and distributed systems development, and need to consider the provisioning and merging of policies for individual systems. Another development are privacy-enhancing technologies (PETs), which cause additional high-level requirements, for example, from contracts with customers, to be incorporated in the policy-decision and model-requirement activities. Future technologies to guarantee these contracts can be integrated as part of the development activities.

9.5.4. Designing processes for interaction and participation

From this study, the following recommendations on authorization process design can be derived:

- *Integrate functional staff:* Only half of the organizations in the study explicitly integrate functional users. The example of the charity organization indicates that it can be useful to motivate functional users to express their problems with authorization measures. In other cases, existing informal interactions could profit from a formalization to make them more reliable (cf. Section 9.2; Principles P1, P8). According to security usability models (Beautement et al., 2008), acting early to increase the measure acceptability will not only improve the productivity, but also increase compliance and thus overall security (P5).

- *Establish operative activities:* None of the studied organizations has operative measures for authorization (F.3, A.5) in place. However, monitoring authorization problems and behavior (e.g. circumventions; P5) will help to react quickly and to take consequences (P7), and increasing awareness will further improve acceptability and thus overall security (West, 2008, P6).

- *Integrate systems development:* Considering the full authorization lifecycle, including the systems development aspects (Hu et al., 2006; Kern et al., 2002), can reduce inadequacies of policies (P1). One negative example is the university administration case, where productivity of functional users is impacted after software updates because permissions are missing (P7). This could be addressed by a formalization of these interactions (P8).

- *Improve stakeholder interaction:* Interrelations are not only important between perspectives, but also within perspectives, for example, when local and central administrators cooperate. As seen in the central university IT case, it can be problematic if there is no defined channel for interactions (cf. Section 9.2; P1, P8; Bauer et al., 2009; Flechais and Sasse, 2009).

9.6. Conclusion

The process model INDUSE describes authorization processes and aims to foster the integration of different perspectives on authorization through adequate and flexible processes. The model is focused on structuring the key characteristics of the authorization processes, namely the activities, stakeholders, and interrelations. These process primitives can then be used to evaluate existing processes for comprehensiveness through control and data flow as shown in the evaluation for eight organizations from practice.

Q1.1 *What type of organization requires what authorization measure and what are the influencing factors?* The study reinforced the insight that adequate authorization processes are essential for adequate authorization measures (P8). For example, participants reported the interference with functional work when procedures are inadequate. The study showed that important factors include the organization's size, its configuration (formality, centralization), and the focus of the authorization measure (e.g. on systems integration). Further research is needed to explore the relationship between organizational factors and authorization-process design.

Q10.1 *How formalized and centralized should procedures be in what context?* The characteristics of the processes for the different contexts in the study also indicated relations between attributes of the organization and the formality and centralization of processes: Larger organizations have more formally defined processes and more complex processes and more dynamic smaller organizations rather informal processes (P7). The findings from this study may serve as a hypothesis for further examination of these relations. Moreover, organizational research should also aim to explain what the characteristics of effective and efficient procedures are in what contexts.

INDUSE was successfully used to describe the authorization processes in the studied organizations, indicating the usefulness of the model for further tasks:

Q1.2 *How can we support the design of adequate authorization measures?* The study showed that a process model, such as INDUSE, can be employed to describe a broad range of authorization processes to identify potentials for process improvement, indicating activities and interrelations that could be more explicitly modeled in the process (P1, P8).

Q3.2 *How do procedures need to be structured to comply with regulations and how do regulations need to be structured to allow procedures to comply with them?* While this chapter did not directly address this question, INDUSE may serve as the basis for defining procedures that comply to regulations. The regulations could state requirements on the activities and characteristics, such as the degree of formality, that the processes have to fulfill (P8).

Authorization processes often need to be integrated with other organizational processes. We examined how INDUSE relates to processes from IT, information security, and configuration management as well as secure systems development:

Q10.3 *How can the authorization procedures be integrated with related organizational processes?* INDUSE as one authorization process model relates primarily in two distinct ways to other processes: One form are activities that offer infrastructural support to authorization tasks, such as configuration management, or to specific challenges, such as risk assessment. The second form are higher-level processes that need to encompass authorization activities, such as in the case of information security management processes, and can use INDUSE to more concretely specify the procedures. Practitioners will profit in both ways: from consulting other process models for enabling procedures and the management of INDUSE processes, and from the integration of INDUSE processes in the broader IT process landscape of the organization.

From the studied processes in practice, we can derive further insights:

Q9.1 *How do requests for changes to the policy take place in practice and what are the obstacles for functional staff?* In the study, we saw a number of different ways for functional staff to request changes, for example, informal discussions or informally turning to a designated key user who then triggers a formal process. However, well-defined change procedures can only be found in half of the cases and we recommend to explicitly "integrate functional staff" and "establish operative activities" (F.3; P1). Further research is needed to identify the characteristics of effective and efficient change requests.

Q6.1 *How do policy decisions take place in practice and what information is employed?* The decision-making on policy changes differs among the studied organizations. Smaller and less formal organizations tend to have more dynamic and informal decision procedures in discussions (P7), while more formal contexts define key users and role owners who, in cooperation, decide on policy changes (P8).

Q8.1 *How do policy authors interact with policy makers and functional staff for implementing changes?* Depending on the decision-making procedures, policy authoring may occur ad hoc (directly by the policy maker) or in defined interactions between local and central administrators. One recommendation derived from the study is to make these stakeholder interactions explicit (P1, P8).

Q7.1 *How must review procedures be designed to be effective and efficient?* / Q7.2 *How can different perspectives cooperate for effective and efficient reviews?* Only very formal contexts had defined procedures for regular reviews or sign-off procedures as part of policy changes. While this may not be appropriate for all organizations, the study indicates that some can profit from explicitly integrating functional staff into the review process (P1, P8).

Q15.2 *How can the different perspectives interact for more effective and efficient controls integration?* While we can observe informal relations between operational and development perspectives for all studied organizations, these are in only one case explicit. Considering potential consequences of the missing feedback loops, we recommend to foster the interaction and make it explicit (P1, P7, P8).

10. Comprehensive decisions: Decision support

[S]hutting up the houses was perfectly insufficient [to prevent the spreading of the infection]. Indeed it seemed to have no manner of public good in it, equal or proportionable to the grievous burden that it was to the particular families that were so shut up.

D. Defoe: A Journal of the Plague Year (1722)

As the Authorization Problems Study showed, decisions on the adequacy of policy changes are at the core of achieving an effective and efficient authorization measure (cf. Section 2.3.3, Req 1). We saw primarily two consequences of inadequate policies, both impacting the measure's effectiveness by increasing the organization's risks: First, over-entitlements enabled the potential misuse of permissions; second, indirectly, restrictive policies caused disruptions of functional tasks and thus led to circumventions that entailed additional risks (Req 2, Req 5). To address these problems, we need to address the question of how to design adequate measures for specific contexts (Q1.1) and how to arrive at satisfactory approaches for functional (Q4.1) and administrative staff (Q4.2). One particular point that participants of the Authorization Problems Study raised was the problem of taking comprehensive decisions. Addressing this problem, how can we guide the decisions (Q11.1) and what information can we employ (Q11.2)?

Looking more generally at decisions in organizations, decision theory traditionally distinguishes between decisions under certainty (complete knowledge of decision consequences), under risk (accurate risk probabilities), and under uncertainty (without prediction of the outcomes; March et al., 1958, p. 137). Because of the unknowns in organizational authorization – including the future tasks and behavior of individuals – policy decisions will often fall into the uncertainty category. In practice, decisions are often not taken individually and consciously; Cohen et al. (1972), accordingly, describe decisions in "organized anarchies" through a "garbage can" model: In a dynamic environment, streams of choices, problems, and solutions can take on the state of garbage that needs to be processed to take decisions. This can lead to more actions being taken than thinking and, for example, require the interpretation of superiors' decisions by subordinates. The problems from non-comprehensive decisions (cf. Section 5.1) indicate that policy decision-making in organizations often takes on this form without structured procedures for adequate decisions. The benchmark for the decision-support measure is thus whether we can improve decisions in such environments.

The aim of this chapter[1] is to explore the problems with and approaches to support structured decision-making with decision-support measures, primarily addressing the effectiveness and efficiency of decision-making (Req 6). This not only involves the challenge of the guidance itself (Req 11), but requires us also to address the change requests and procedures that surround the decision (Q9.1, Q9.2, Q10.2), and how stakeholders interact for policy changes (Q8.1).

When designing decision procedures, we can apply either authoritative, consultative, or group decisions (Vroom, 2000). Since one important principle of this thesis is to improve the participation

[1]The main parts of this chapter appear similarly, but shortened in the security track at the ACM Symposium on Applied Computing (SEC@SAC: Bartsch and Sasse, 2012).

of functional staff in authorization management (P1), the artifact in this chapter aims to support consultative or group decisions. The integration of diverse perspectives on authorization is particularly useful when the information that is required for taking decisions is disseminated insufficiently between the decision maker and subordinates (cf. Vroom, 2000). We aim to support the participation of non-experts through concrete decisions (P3) and a reduction of the overall burden of decision-making while increasing the quality of decisions (P2). Depending on the importance and likelihood of commitment of subordinates (cf. Vroom, 2000), it may also be helpful to increase the security awareness among functional staff and decision makers as part of the decision process (P6), and, in extension, increase the motivation of compliance of the individuals (P5).

The approach in this chapter is to systematically develop a decision-support measure and employ the artifact in formative evaluations to derive insights on how to implement decision support. The measure in this chapter is an extension of the Policy Override Calculus from Appendix 8, which focused on the particularities of override policies and on deriving a policy in one pass. Particularly the latter showed to be problematic in consultations with decision makers due to the required effort. In contrast, the decision-support measure targets standard authorization models and is more process-oriented to support evolutionary policy changes.

10.1. The problem of policy decisions

Problems with decision-making have been intensively researched in cognitive psychology, beginning with the discovery by Kahneman and Tversky in the 1970s of the heuristics and biases that humans employ when making decisions (Gilovich et al., 2002), eventually leading to the awarding of the Nobel prize in economics in 2002. In information security, these effects have been shown for privacy decisions, for which Acquisti and Grossklags (2005) showed how individuals traded short-term benefits, for example, revealing personal data for discounts despite their general privacy concerns. Similarly, Ahern et al. (2007) described how social network users decided to share location information in practice despite earlier contrary statements.

The reason for this inconsistent behavior is the “bounded rationality” of humans when taking decisions, not carefully weighing the individual arguments of a security decision, but applying heuristics that, for instance, depend on the framing, whether a risk factor can be easily recalled from memory, or one of several other biases (Camp, 2009). For authorization decisions, the “satisficing” effect (March et al., 1958), which we also observed for policy editing (cf. Section 7.3), is particularly relevant, describing behavior of decision makers that choose the first option that seems adequate instead of examining the available options. For policy decisions, problems in decision-making have been related to the lack of awareness of the consequences of the decisions. For instance, Ahern et al. (2007) recommend to increase the awareness of social networking users on what aggregated location data may reveal. West (2008) similarly argues that security decisions are difficult because the risks are often abstract when compared to functional benefits from taking the risks. This has also been discussed for medical contexts where abstract rules on treatment (“compliance” with treatment rules) have shown to be problematic (Martins, 2005).

One could argue that these effects are not relevant to *organizational* decision-making, since the decision makers are trained in taking adequate decisions, evaluating the factors comprehensively and objectively. Instead, policy decisions are often taken ad hoc or well-defined processes are significantly modified. Examples can, for example, be found in the Authorization Personas in Section 4.3: Emily as technically-informed functional staff needs to take decisions without clear guidance or training. Brandon as the personal assistant takes the decisions on behalf of his manager. Nicole as technical staff effectively takes decisions that should rather be taken by the responsible functional managers. In the Authorization Problems Study in Chapter 5, we examined problems with the decisions in more detail and discovered three categories of problems:

- *Cognitive problems:* Cognitive problems involved the effort and complexity of decisions, caused by missing guidance and leading to non-comprehensive decisions. The consequence are biased business-, security-, or formality-driven decisions, thus indicating the above-described effects of bounded rationality on an organizational scale: Decisions are framed by the mindset of the decision maker so that technical staff tend to take security-driven decisions and functional managers rather business-driven ones.
- *Organizational problems:* Participants reported various types of organizational problems with ineffective or inefficient procedures, including conflicts of authority and emotional costs of denial. For organizational information security, further problems have been theorized based on the principal–agent theory, including externalities and information asymmetry (Pallas, 2009). Externalities are positive or negative effects of the decision on individuals not involved in the decision, for example, when insecure behavior of individuals endangers the entire organization. Information asymmetry refers to different levels of information for staff and managers, for instance, leading to problematic decisions by managers because they lack the knowledge of consequences for staff.
- *Socio-technical problems:* A third category of problems in the study related to socio-technical problems with the interaction with the respective tools, including non-transparent and incomprehensible policies, inadequate model expressiveness, and unusable mechanisms to review policies.

10.2. Approaches to decision guidance

One important goal of this chapter is to further explore the aforementioned problems and how they can be addressed. From literature, a number approaches lend themselves to address the problems:

Participation of functional staff Participatory processes have been shown to motivate staff, even though the effects depend on the context and particularly work for small groups with high identification (Wagner III. and Gooding, 1987). Participation may thus increase the acceptance of policy decisions, leading to higher satisfaction and a higher effectiveness of the authorization measure. Scully et al. (1995) demonstrate that participation improves the information dissemination and thus increases the adequacy of decisions and the performance of subordinates, particularly in cases of information asymmetries when information is unevenly distributed. This indicates that participation will improve decisions particularly in dynamic and complex contexts where information is not generally widely distributed at decision time.

Increased security awareness and expertise To make adequate decisions, the involved stakeholders need to be aware of the associated risks. Siponen (2001) argues that everyone should be security aware who interacts with information systems. He differentiates between descriptive and prescriptive awareness, that is, the general security expertise or "action-guiding commitment to the objectives of awareness", respectively. While prescriptive awareness is more useful for policy decisions, we need to restrain ourselves from pure indoctrination without the deeper understanding, in analogy to the effective passing on of morality (Siponen, 2001). In organizational behavior, a typical approach to awareness-building is reinforcement, that is, supporting appropriate and disciplining inappropriate behavior, and social learning, which encompasses the learning from close contact, imitation, and the understanding of concepts (Schermerhorn et al., 2008; Bandura, 1977). We can foster these effects with appropriate decision-guidance.

Comprehensible decision factors The comprehensibility of security is generally impacted by the way humans think about the abstract security measures. Humans also need to build mental models (Johnson-Laird, 1980) of how security measures work to understand the measures at a specific level of detail. Generally, it is useful to build a simplified, "instrumental" mental model of mechanisms to reduce the cognitive effort of learning to interact with them. Camp (2009) studied what mental model humans have of security, considering, for instance, a physical security model, through the analogy of locks and doors, but also a warfare model. As one would expect, Asgharpour et al. (2007) found that there are stark differences between the mental models of experts and lay users. Mental models have been employed to improve the comprehensibility and behavior in various fields. For instance, in the risk communication for medical drugs, patients were found to understand package inserts, but ignore the formulated advice (Jungermann et al., 1988). For lay users of security, Wash (2010) studied their folk models to selectively influence the models to achieve more secure behavior.

As stated in the previous section, one of the important problems in comprehending decision factors is the lack of concreteness of risk factors. Through adequate risk communication, we can increase the level of concreteness in order to make the information more understandable. For medical risk communications, for example, to enable well-founded choices of treatment options, Rothman and Kiviniemi (1999) distinguish three different goals of the communication: informing about risks, creating a personalized awareness of risks, and the persuasion of a change of behavior. In comparison to simply conveying risk probabilities, contextualizing risks is more successful in creating awareness and influence behavior. Cognitive psychology indicates that it is important that people are able to "simulate" or imagine the antecedents and consequences of risks (Kahneman and Tversky, 1982). For medical risks, consequences (symptoms) that are easier to picture increased the awareness, as do testimonials of affected individuals when there is an identification with those, and disturbing images, such as, from car accidents. To actively contextualize medical risks, physicians emphasize antecedents by letting patients review their personal behavior, for instance, in case of HIV prevention, or provide personalized information about the links between the personal behavior and health problems, correcting pessimistic and optimistic perceptions (Rothman and Kiviniemi, 1999). Risk communication has also been researched in sociology: Cannell and Otway (1988) argue that it is particularly important for risks that the "communication...take[s] into account the knowledge and experience of the audience it addresses." De Marchi (1991) found that people contextualize risks when discussing natural hazards.

For authorization policy decisions, we thus need to develop adequate and consistent mental models in the communication of decision factors and increase the concreteness of the factors by contextualizing them for the individual.

Guiding decisions Ahern et al. (2007) argue from their field study on privacy decisions in social networks that supporting decisions through guidance could be useful. For general policy decisions, effective guidance requires an adequate modeling of the involved risks from the changes. Several risk models for authorization have been proposed in literature. Han et al. (2009) developed an abstract risk model for evaluating delegation based on difference in ranks and the permission range of a delegated role. Molloy et al. (2008) instead employ market mechanisms that are based on an authorization risk-model to guide decisions on permissions. Their simulation shows that under their authorization model, individuals behave optimally for the organization. However, their model assumes measurable benefits and risks, and rational individuals. The decision-support approach by Beresnevichiene et al. (2010) for security investment choices targets authorization measures in their case study. They base the calculations for their case study on expert consultations, including estimations of user trustworthiness, the ability for unauthorized access, and mitigation effects. However, the risk models in literature are rather abstract and do not address the practical problem of eliciting the necessary information for the risk calculations in practice.

The required data to support decisions concerns risks and benefits from a policy decision. Risk factors for authorization primarily encompass human threat aspects, both from intentional activities, such as insider attacks (Pfleeger and Stolfo, 2009), and from accidental incidents, such as human errors (Stoneburner et al., 2002). Reason (1990) groups human errors as mistakes (planning), lapses (storage), and slips (execution). According to the Situation Awareness theory, factors such as workload, stress, and system complexity influence the human reliability (Bedny and Meister, 1999; Endsley, 1995).

10.3. Decision-support study

To explore how selected approaches to decision support help in improving the policy decision-making and what problems we are faced with in practice, we built a decision-support prototype based on the risk factors stated by employees of a large enterprise. We conducted formative evaluations with policy makers and functional staff from a variety of backgrounds to evaluate the practical viability of the decision-support tool. Of the aforementioned problems in decision-making, the prototype is primarily focused on the cognitive effort of providing the information necessary for the decision support and of making comprehensive decisions, and on the organizational challenges of integrating a supporting tool into organizational processes without incurring disproportionate overhead.

10.3.1. Elicitation of decision factors

The first part of the study pertains the elicitation and structuring of the factors that need to be considered in policy decisions. For this we again employed the data from the large, multi-national organization: 118 interviews of about 45 minutes each in two countries (cf. Section 4.3). One researcher who is not affiliated with the organization coded transcripts of these interviews on security compliance. The coding resulted in 172 quotes and 62 raw codes on risk and benefit factors that participants of the study considered relevant in a variety of contexts and systems, including in highly critical and privacy-relevant areas.

Using a Grounded Theory approach (Cairns and Cox, 2008; Glaser and Strauss, 1967), we related risk factors to arrive at five high-level decision-factor groups: "benefit", "high-level policy", "data sensitivity", "impact", and "threat". For example, the quote:

> *"... but it could do damage to the company, you know, the company share price and markets and so forth"*

was coded as "Impact: Organizational: Share value", thus part of the factor group "impact". We then built the decision-support prototype by creating six spreadsheets as decision artifacts: We assigned the factor groups to the spreadsheets to enable the efficient collection of information and formulated questions for each factor.

Pseud.	Role	Perspective	Position	Organization	Organizational conf.
M1	Manager	Technical	Program manager, leading dev. team of 26	Software industry, multi-national, 1000 emp.	Simple Structure, divisionalized
M2	Manager	Technical	Project leader of 6	Web agency, 15 emp.	Simple Structure
M3	Manager	Technical	Unit manager of 25	Development company for large bank, 300 emp.	Professional bureaucracy
S1	Staff	Non-technical	Quality assurance	Food industry, multinational, 2500 emp.	Divisionalized professional bureaucracy
S2	Staff	Non-technical	Customer relations	Regional utilities company, 2500 emp.	Machine Bureaucracy, divisionalized
S3	Staff	Technical	Technical quality assurance	Security-sensitive electronics, 1500 emp.	Professional bureaucracy

Table 10.1.: Study participant sampling

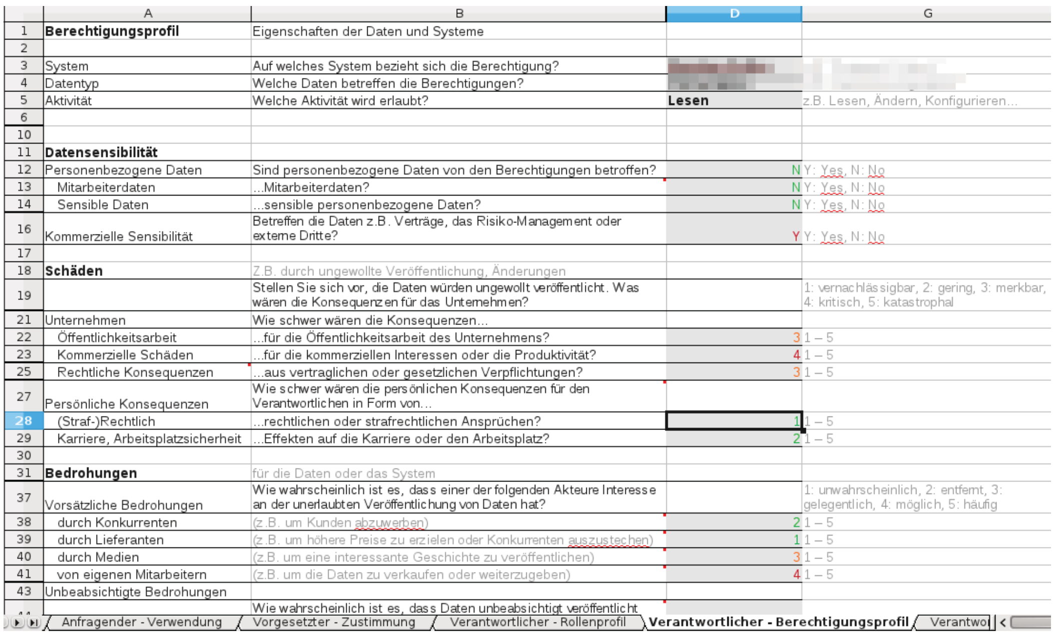
10.3.2. Formative evaluation with functional staff and decision makers

We conducted the second part of the study as a qualitative evaluation to arrive at rich descriptions of problems and causation (cf. Section 7.3.1; a subjective empirical laboratory study, cf. Section 2.5). This part consisted of formative evaluations based on the decision-support prototype with three functional staff and three policy makers from a variety of practical backgrounds. The main questions that we addressed in the evaluations were:

- In what context is a decision-support tool in what form appropriate?
- What data can be collected from whom? Who has the information with the necessary degree of precision?
- How comprehensible are the collected decision factors and how can we improve their comprehensibility?
- Which information are individuals willing to provide?

The participants were selected in an “information-oriented” sampling (cf. Section 7.3.1) to arrive at a broad sampling with respect to organizational type (size and Structural Configuration of Mintzberg (1980), cf. Section 4.2) and position as shown in Table 10.1. The evaluations were not geared to result in representative and comprehensive coverage of decision-support problems, but rather provide us with rich subjective data to inform further research. We chose to cover a limited number of participants thoroughly, since this should allow us to gain a broad overview of the problems to be expected with decision support (cf. Section 7.3.1 and Nielsen and Landauer, 1993).

The evaluations lasted about one hour each and were semi-structured: They first covered preliminary questions on the participant’s background and relation to authorization in practice. Then, the interviewer went with the participant through a participant-specific scenario, asking the participant to complete the questionnaires. While making the evaluations more difficult to compare, the individual scenarios allowed us to reduce the effect of participants having to understand the possibly remote scenario (cf. Wash, 2010). The participants were asked to “think out aloud” while completing the forms and the walk-through was enriched with in-situ prompts (Cairns and Cox, 2008). Following



	A	B	D	G
1	**Berechtigungsprofil**	Eigenschaften der Daten und Systeme		
2				
3	System	Auf welches System bezieht sich die Berechtigung?	[redacted]	[redacted]
4	Datentyp	Welche Daten betreffen die Berechtigungen?	[redacted]	[redacted]
5	Aktivität	Welche Aktivität wird erlaubt?	**Lesen**	z.B. Lesen, Ändern, Konfigurieren...
6				
10				
11	**Datensensibilität**			
12	Personenbezogene Daten	Sind personenbezogene Daten von den Berechtigungen betroffen?	N	Y: Yes, N: No
13	Mitarbeiterdaten	...Mitarbeiterdaten?	N	Y: Yes, N: No
14	Sensible Daten	...sensible personenbezogene Daten?	N	Y: Yes, N: No
16	Kommerzielle Sensibilität	Betreffen die Daten z.B. Verträge, das Risiko-Management oder externe Dritte?	Y	Y: Yes, N: No
17				
18	**Schäden**	Z.B. durch ungewollte Veröffentlichung, Änderungen		
19		Stellen Sie sich vor, die Daten würden ungewollt veröffentlicht. Was wären die Konsequenzen für das Unternehmen?		1: vernachlässigbar, 2: gering, 3: merkbar, 4: kritisch, 5: katastrophal
21	Unternehmen	Wie schwer wären die Konsequenzen...		
22	Öffentlichkeitsarbeit	...für die Öffentlichkeitsarbeit des Unternehmens?	3	1 – 5
23	Kommerzielle Schäden	...für die kommerziellen Interessen oder die Produktivität?	4	1 – 5
25	Rechtliche Konsequenzen	...aus vertraglichen oder gesetzlichen Verpflichtungen?	3	1 – 5
27	Persönliche Konsequenzen	Wie schwer wären die persönlichen Konsequenzen für den Verantwortlichen in Form von...		
28	(Straf-)Rechtlich	...rechtlichen oder strafrechtlichen Ansprüchen?	1	1 – 5
29	Karriere, Arbeitsplatzsicherheit	...Effekten auf die Karriere oder den Arbeitsplatz?	2	1 – 5
30				
31	**Bedrohungen**	für die Daten oder das System		
37	Vorsätzliche Bedrohungen	Wie wahrscheinlich ist es, dass einer der folgenden Akteure Interesse an der unerlaubten Veröffentlichung von Daten hat?		1: unwahrscheinlich, 2: entfernt, 3: gelegentlich, 4: möglich, 5: häufig
38	durch Konkurrenten	(z.B. um Kunden abzuwerben)	2	1 – 5
39	durch Lieferanten	(z.B. um höhere Preise zu erzielen oder Konkurrenten auszustechen)	1	1 – 5
40	durch Medien	(z.B. um eine interessante Geschichte zu veröffentlichen)	3	1 – 5
41	von eigenen Mitarbeitern	(z.B. um die Daten zu verkaufen oder weiterzugeben)	4	1 – 5
43	Unbeabsichtigte Bedrohungen			
		Wie wahrscheinlich ist es, dass Daten unbeabsichtigt veröffentlicht		

Anfragender - Verwendung
Vorgesetzter - Zustimmung
Verantwortlicher - Rollenprofil
Verantwortlicher - Berechtigungsprofil
Verantwo

Figure 10.1.: Screenshot of the decision-support prototype (permission profile)

the walk-through, additional a-posteriori questions were asked on the perception of the adequacy and usefulness for the participant's organizational context.

The evaluations were audio-recorded and transcribed. The analysis followed the principles of Grounded Theory (cf. Adams et al., 2008; Glaser and Strauss, 1967). The transcripts were coded for the context at the participant organization, the comments on and problems with the decision factors, and statements on the adequacy and usefulness of decision support. The open coding resulted in 114 raw codes that were consolidated as shown in the findings below. While all evaluations were conducted in German, the coding occurred in English and the quotes were translated for inclusion in the findings.

10.4. Decision-support prototype

The general idea of the decision-support prototype is to separate different categories of risk factors so that each group can be elicited individually when appropriate as part of the change procedure. The prototype consists of interrelated spreadsheets (artifacts) to collect the risk and benefit factors, and present aggregations of the factors to guide decisions. The permission-profile artifact is shown in Figure 10.1.

In line with the aforementioned approaches to decision support, the prototype aims to increase the participation of functional staff in the decision process by asking them to provide their estimation of risks and benefits (P1). The collection and presentation of risks and benefits is designed in a way to increase the awareness of the interacting stakeholders for the different factors involved (P6). The prototype relies on concrete, contextualized inputs for collection of data and outputs for evaluation to achieve comprehensible decision factors (P3). The prototype provides decision guidance, reducing the burden of decision makers (P2), even though the focus does not currently lie on a comprehensive risk model. The prototype is designed to fit in a variety of organizational measures (P8) that should

Artifact	Example role	Activity	Information	Source
Change request	Requester	Permission request	Purpose of, benefit from change, regulatory/qualification aspects, awareness of risks	Manual input
Usage environment	Requester	Permission request	Risks from the environment in which the requester uses the permission (e.g. "office" in contrast to "on the road")	Manual input, reusable
Change approval	Requester manager	Approve request	Summary of the benefits and risks of the request	Aggregation of request and usage environment
Permission profile	Resource owner	Verify request	Risks associated with granting a permission of a specific activity on a set of data	Manual input, reusable
Role profile	Resource owner	Verify request	Aggregate of the risks from role members and from permissions assigned to the role	Aggregation of data from previous requests
Change decision	Resource owner	Verify request	Summary of the benefits and risks from permission–role and user–role assignments	Aggregation of data from prior artifacts

Table 10.2.: Artifacts produced and used in the decision-support tool as part of the change procedure

be applicable in dynamic environments (P7).

10.4.1. Prototype design

The artifacts encompass questions to elicit the decision factors and aggregations of the factors. The questions aim for qualitative input, requiring either ratings (1–5 for a given scale, e.g. expected benefit), binary yes/no answers (e.g. having a specific awareness), or textual inputs (type of activity, informal benefit). Each question and output is accompanied by contextualizing clues, such as examples from the factor-elicitation study. The six artifacts of the tool are listed in Table 10.2.

While the decision-support tool does not prescribe a specific policy-change process, we can assume an example procedure for didactic reasons that the roles in Table 10.2 refer to:

1. A "requester" completes a request form for a concrete activity in a system on a specific set of data and chooses an adequate usage profile for the request;
2. The manager of the requester signs off the request, verifying the necessity of the request;
3. The "owner" of the resource decides on whether to enact the requested changes and how.

Change request

The change request is primarily thought to be completed by staff who require extended permissions, or on behalf of them. The artifact encompasses questions on the general task, the specific activity and data, benefit of providing the permission, and a self-assessment of security awareness. While the participants were generally comfortable answering the questions, we still noted interesting difficulties. For example, there was contradicting feedback on the concreteness of the benefit to be specified. A rather abstract question ("How high would you rate the benefit for the organization granting this permission?") can be difficult:

S1: "This is difficult to say [as one factor] since [activity] only makes out a small part of my work, but I need that transaction a lot if I do [activity]"

Conversely, a more concrete form (frequency, benefit per use) demonstrated the problem of specific benefits not fitting the given schema of frequency and time savings:

S3: "Productivity improvement would be from 0 to 100, because else I could not do it… there are also cases… [in which] having access improves the quality of work"

Another benefit factor that the prototype asked for were indirect risks, that is, risks that result from *not* granting the permission and staff needing to take more dangerous approaches to addressing the business needs (e.g. sharing passwords). We found problems caused by the concept being unfamiliar or not applicable in the domain. S2, for instance, first misunderstood the question and then states on further prompting:

"We don't share passwords among each other… that is not wanted… and if then something happens while I'm away, then I'm held accountable, so we won't do that"

S1 similarly has problems understanding the concept, but rather because it is common practice to share logins, only developing the awareness of potential problems in the course of the discussion:

"In our team, for example, when a person is coming back from parental leave… and the IT has not set the permissions correctly in time, she would have sat there three days not able to work… so I log in with my permissions on her PC… I don't think the company has problems with that, even though my name would appear next to something she saves"

Usage environment

The artifact on the usage environment should elicit risk factors that concern the context in which the permission would be used if granted. This includes, among others, the physical security of the environment, the satisfaction of staff, and stress in the specific environment. Generally, participants agreed to distinguish the contexts as it is common practice to take it into account for decisions, for example, distinguishing between desktop and laptop computers (e.g. S3 and M3).

Problems were seen in the added value of specific factors. For example, multiple participant raised doubts on whether the satisfaction has a significant impact on the threat of malicious activity (without referencing the participant):

"Well, the overall satisfaction in the company is not really high, but as I hear it people are all very loyal and rather shocked if someone takes information to another company"

A more practical problem with the collection of the data was that staff stated that they would not answer honestly, fearing that the data could be misused:

"I could answer the employee satisfaction… but I would not answer questions on satisfaction, unless it is anonymous"

Instead, it was suggested to reuse available data, for example, from employee surveys.

It was also interesting to observe how the physical-security factor was misunderstood or unclear depending on the framing of the participant. In case of organizations with a high emphasis on physical safety (e.g. utilities in case of S2), physical security is understood as the safety of the workplace (S2: "Is this about tripping hazards?").

Permission profile

The permission profile defines the risks associated with misusing a specific permission, such as “read source code from project X”. The factors of this artifact cover the general sensitivity of the data (e.g. personal data, commercial sensitivity), impact of misuse (e.g. from commercial or legal consequences), and the threat (the probability e.g. of a misuse or of accidental incidents). For higher precision of the risk estimation, we would need to elicit the impact and threat values for individual scenarios, and aggregate the risk scores. However, to reduce the burden of completing the form, we decided to approximate the overall risk from separate elicitations.

Whether staff was able to complete the form depended on the individual context. For instance, for the legal impact:

> *S3: “The data are also sometimes part of the contract... [the technical project manager] knows better about that than me”*

In other cases, the variation of the impact is high:

> *S1: “It depends on the kind of information... it might be already rounded... or aggregated... in other cases you don’t want to hand it out [at all]”*

Or, similarly, the potential attack scenarios are complex:

> *S1: “This data alone would not help, you need the actual product with the data... and I don’t know which way they need to go to get the product”*

Generally, there were doubts about the scalability of this kind of data collection:

> *M1: “What would be in the case that you had 20 different projects... Then, one would want to have a hierarchy, ‘general project’ and an ‘override’ for specific ones.”*

Role profile

The role profile is generated from aggregations of data on assigned permissions and contexts of assigned users. First, this artifact can inform decisions on whether it is appropriate to assign a role to a user, that is, whether the risks from the role are adequate for the benefit and the context of the request. Second, the artifact can support the decision of whether the users already assigned to a role make it adequate to assign an additional permission to the role. M1 noted that he would need more information on the users in a role. He also remarked potential problems of scalability:

> *“Many roles are like ‘developer’, so have a too broad spread, so you can only get sensible data in the role profile for focused roles.”*

Change approval and decision

Aggregating risks and benefits can support both an approval, for example, by the line manager of the requester, and the final decision of a policy maker. For the approval by a manager of the requester, the approval artifact aggregates information on the request benefits and context risks. For the actual decision, the policy maker may derive insights on the adequacy of different change options from the risks and benefits. The decision artifact, in addition to the approval data, aggregates the information on the risks from the requested permission, from the users assigned to a role and from permissions assigned to a role (see role profile). Problems stated by participants with respect to the aggregations of values primarily related to the transparency of the source of the aggregations (M2: “I haven’t realized where those numbers come from.”).

Request addressee	Approval	Decision	Reported by
Line manager	–	Line manager	M2, S1, S2
Line manager	Line manager (implicit by forward)	Application/resource owner	M1
Resource owner	–	Resource owner	S3
Administrator	Project manager	Unit manager	M3

Table 10.3.: Change request processes in the study

10.4.2. Change procedures with decision support

Organizations implement very distinct procedures for policy changes, ranging from informal direct communication to highly regulated form-based procedures (cf. the processes analyzed using INDUSE in Section 9.5). The decision-support prototype does not prescribe a specific degree of formalization or process model. Of the participants' authorization contexts, five had an informal and one had a formal procedure.

Although the forms of the decision-support prototype introduce extra effort to informal procedures, the participants did not categorically reject employing such a system and saw advantages even for small organizations:

> *M2: "The current size of our company would not justify this tool, but in larger companies where you cannot have the overview over all areas and the applications... In our case, it would be a nice way to structure the decisions"*

S3, for example, further remarked with respect to the change-request artifact:

> *"Actually, I already provide this kind of information – only, I... formulate it as an email"*

Apart from the additional effort that the tool introduces, we also need to examine whether the artifacts fit in the current processes. Thus: who is the *addressee* for the change request, is there an *approval* and by whom, and who *decides*. The participants of the study report four distinct models (cf. Table 10.3), which fit the artifacts: In each case, the requester would complete the change request and choose the usage environment. If there is an approval step, the person responsible would receive both original artifacts and the approval summary. Finally, the decision is taken, additionally considering the generated role profile, the permission profile, and the decision summary.

In addition to the actual procedure, participants also report rich discussions before the actual request to assign a permission that the tool needs to support:

> *S1: "That has been agreed upon with [other department] beforehand... we discuss with them how I can access that transaction... then my manager requests the permission from IT"*

10.4.3. Expected effort

One of the central requirements for authorization measures is its efficiency with respect to the management overhead (Req 1). Although the decision-support tool is aimed to be light-weight, a certain amount of cognitive effort is inevitable to collect input and evaluate output of a decision-support tool. This may not necessarily increase the overall effort, since non-supported processes incur efforts as well. In the following, we analytically compare the expected costs of decision-supported procedures to non-supported ones.

Authorization contexts can be diverse. For a broad perspective of decision-support costs, the analysis targets two distinct authorization contexts from hypothetical organizations:

	Original process			Decision-support process		
	Frequency per request	Effort per occurrence	Mean effort per request	Frequency per request	Effort per occurrence	Mean effort per request
Open form/tool	1	1	1	1	1	1
Establish permissions	1	15	15	1	15	15
Enter permissions	1	2	2	1	2	2
Explain/answer questions	1	10	10	1	20	20
Create usage profile	–	–	–	0.01	15	0.15
Send request	1	1	1	1	1	1
			29			39.15

Table 10.4.: Detailed analysis of activities for the request step in the formal context

- *Formal context:* The baseline process is formalized and involves form-based requests of permission changes as it can be found in many organizations (compare the Shared folder procedure in Section 5.1):
  1. Requester: Completes a form for a permission request, informally stating the motivations for the change,
  2. Requester manager: Checks and acknowledges request,
  3. Resource owner: Checks and acknowledges request.
- *Informal context:* This process is rather informal and only requires requesters to click on a link and provide a reason to request a change from the resource owner (compare the Sharepoint procedure in Section 5.1):
  1. Requester: Clicks a link, informally stating the motivations for the change,
  2. Resource owner: Receives email, checks, and acknowledges request.

Each of the steps in the original, unsupported processes involve several activities with respective cognitive efforts. The decision-supported processes in each of the baseline contexts encompass different activities or the activities will incur different efforts. Table 10.4 shows the activities for the request step in the formal context in detail, both for the unsupported and supported process. We apply a simplified model of effort for the individual activities, inspired by the GOMS model (Card et al., 1983). For each activity, we note the frequency relative to the requests, the effort at each occurrence, and the derived mean effort. For example, "Establish permissions" needs to be executed for each unsupported request (frequency: 1), has a relative effort of 15, resulting in a mean effort per request of 15. The sum of the mean efforts indicates the effort for the entire step in the process.

Table 10.5 summarizes the efforts per step for the formal and informal context and the respective unsupported and supported processes. For the formal context, the analysis indicates a reduction by 10% for the supported process. While the request has a notable higher effort, due to the necessity of entering more precise information on the request, both the approval and the decision step profit from the support. For the approval step, this is due to a reduced effort to understand the request (better structured data) and less need to query the background of the request. Both effects also reduce the effort for the decision step, offsetting the infrequent need to create permission profiles. Overall, in comparison to formalized processes we can expect similar or slightly reduced efforts from introducing decision support.

For the informal context, the overall effort per request for the informal context should be similar as well. While, again, the request requires higher effort, the reduction in decision-making makes up for that. We assume here a good integration into the application as for the informal process, thus not

		Formal context effort		Informal context effort	
Step	Role	Original	Supported	Original	Supported
Request	Staff	29	39	12	24
Approve	Manager	19	16	–	–
Decide	Resource owner	38	23	35	23
		86	78	47	47

Table 10.5.: Comparison of efforts with and without decision support in formal and informal contexts

requiring the requester to establish the necessary permissions. Still, the formulation of the request incurs the higher mental effort of answering decision-support questions.

Overall, the analytical results indicate that it should be possible to implement decision support as a light-weight measure with similar overhead as existing procedures. There could be negative effects from the increased request effort by the functional stakeholder who might instead be tempted to circumvent the measure. Apart from an empirical validation of these results, it also remains to be researched whether the potential increase in security awareness through the decision-support approach offsets these effects. Moreover, we need to explore whether the overall benefits for the organization through more traceable processes and more effective reviews and monitoring make up for this downside.

There are a number of potential optimizations that will lead to an even further reduction of the effort, including automatic data collection, for example, of security trainings and employee qualification, supporting profile creation with existing ones as templates, and a deep integration with administrative tools. Parts of the procedure could also be limited to permissions with a risk above a certain threshold. Moreover, existing risk information from organizational ISMS can be incorporated into the risk model and reduce the amount of data to be collected.

10.4.4. Security awareness

We asked the participants of the study as part of the introductory phase what kind of risks they are aware of regarding granted permissions. The participants either primarily stated unintentional risks, such as "stolen laptop" (M1), or intentional risks, such as "data manipulation" (M2, S2, S3), no one stated a comprehensive range of threats. Moreover, in the evaluations, the awareness of risks depended on the framing from the work practice as described, for example, for the indirect risks in the change-request artifact or the physical security for the usage-environment artifact. Despite the missing clarity, the respective participants (e.g. M2, S2) self-assessed no gain in awareness from the discussions after the evaluations. Conversely, M1 remarked:

> *"On our level, we only look at... the benefit, and ignore the risk... that's why this could be helpful"*

Thus we observed two phenomena: Participants who discussed new risks on the basis of the factors and presumably increased their awareness, and participants who see the tool as helpful to improve overall awareness. Still, future work is needed to explore whether completing a form is already helpful or whether it requires accompanying discussions.

10.5. Categories of problems

In the course of the evaluations of the decision-support prototype, several problems recurred that we categorize in the following to guide further research on the collection of decision factors. Generally, we need to distinguish problems related to the collection of information (the main focus of this work) from those related to the interpretation of aggregated data that support the actual decision. Of the problems enumerated below, only the comprehensibility of the factor and transparency were brought up for aggregations, with the latter exclusively for aggregations.

Availability of precise knowledge Depending on the perspective of the individual completing the form, certain information may be difficult to elicit. One reason in the study was that the data was not available to the specific role (see permission profile). In other cases, the participants lacked the comparison (e.g. for satisfaction in the usage-environment artifact). Third, problems can occur due to the complexity of the elicited data, as described for potential attack scenarios (see permission profile). Fourth, the precision of the data collection is also impacted from the status-quo bias (Gilovich et al., 2002): individuals relying overly on the current situation (S1: "So many have the permission, I don't see it as critical").

Acceptability of information request Problems also occurred when the request for the data is seen as sensitive. Apart from the usage-environment factor of satisfaction (see artifact), this could also be observed for the stress level in the team due to a general mistrust of how stress levels would be interpreted by superiors.

Adequate degree of abstraction The goal of the prototype design was to achieve a high level of concreteness. However, we observed both problems due to overly abstract and overly concrete factors. Too abstract factors primarily affected questions that covered an overly broad area (e.g. the benefit rating for the change request, see artifact). Conversely, the questions could also be too concrete when they excluded a specific kind of input that does not fit the given questions (see request artifact).

Comprehensibility of the question and the factor Questions are difficult to understand if they are too distant for the individual. For instance, S2 found it difficult to relate to the physical security (see usage environment). Thus, the contextualizing clues need to match the environment of the individual. Moreover, the factor behind the question can be difficult on a conceptual level: It was difficult to explain what the aim was of the question on indirect risks if participants cannot relate to those risks (see indirect risks for change request). Similarly, a participant had problems to distinguish impact and threat for the permission profile – two concepts that are difficult to separate for non-security experts.

For the aggregation of data, participants noted similar problems of relating to factors. For example, M1 advocated a clear separation of personal and organizational impact to simplify the interpretation of the aggregations.

Applicability of the factor One reason for the incomprehensibility of a factor can be that it is not applicable to the specific environment, as seen for the indirect risks. More specifically, M3 stated that indirect risks are not an issue in their environment since password sharing is prohibited and users go through the official channels instead. Also, a factor can be inapplicable if it is irrelevant in the context. M2, for example, stated that the amount of external contacts is irrelevant for her decision on granting permissions.

Transparency of aggregations For aggregations of factors, participants stated problems with the transparency of the presented information. For example, for the change-approval artifact, M2 was missing the indication of where the data came from (see artifact). M1 similarly requested for the role profile to receive richer information on its members (see artifact).

10.6. Discussion

General problems with decision support As part of the evaluations, participants questioned whether a risk- and benefit-based decision is generally realistic:

> *M3: "If someone should develop... and needs a permission for that, we cannot say 'not possible' "*

Further, participants argued that a certain level of trust is necessary in a work relationship:

> *M3: "If I had someone who I cannot trust, I should maybe rather let him go"*

Both arguments show how context-dependent decision support is.

Perceived usefulness Participants stated a number of positive effects from such a system:

- *Improves awareness:* Participants found the prototype to increase their security awareness (cf. Section 10.4.4).
- *Improves decision transparency:* The reasoning for decisions on granting permissions can be intransparent:

  > *M1: "What would be important, is a certain transparency in the decision... currently it's rather just 'no'... if some details were known, it would be more understandable"*

- *Improves request communication:* Participants remarked that a decision-support tool may improve the communication for changes to permissions:

  > *S2: "It would reduce misunderstandings if I could formulate through such a form what permissions I require and for what reason"*

Validity of the results The study relies on subjective data from individual employees of different contexts. Consequently, we cannot expect to achieve a precise and comprehensive picture of the individual contexts. Because of the broad range of contexts in practice, this study also cannot provide a representative result, despite the careful sampling to achieve a broad picture of the problems involved in collecting and aggregating data for decision support. Nevertheless, because of the rich data, the study findings should offer a good hypothesis for further research on this subject.

Recommendations From the elicited problems and the feedback by the participants, we can derive the following recommendations on the design of distributed data collection for decision support:

- *Adequately embed collection in procedures:* A separate collection of the decision factors incurs high effort in dynamic environments. Despite the problems found in the study, the results indicate that a distributed data-collection is possible and may reduce the effort (P2, P7). However, the procedures in practice vary with the organizational environments. Considering the required formality and centralization, a decision-support system must be adequately integrated for the specific environment (cf. Section 10.4.2; P8).

- *Contextualize factors:* Several of the problems surrounding the factors related to their comprehensibility. When participants misunderstood questions, they could often still answer them after further explanations framed the questions in a way that was more familiar. Questionnaires for collecting information thus need to be concrete, domain-specific, and tailorable to provide appropriate clues to those completing the form (P3, P8). This might be challenging for the diversity of large organizations.

- *Respect the sensitivity of factors:* We need to respect that certain data is very sensitive and cannot be directly collected for privacy reasons (P5). The sensitivity of factors needs to be elicited, and potential indirect sources and the necessity of the factors should be considered.

- *Offer flexibility in data collection:* For efficiency, the forms must be flexible to provide for varying security needs, for example, requiring a baseline evaluation of risks in uncritical cases, but more details for critical ones (cf. Section 10.4.1; P8, P2).

- *Reuse existing data:* Every question in a form will require cognitive effort and threatens its acceptability if we assume a limited motivational "budget" (Beautement et al., 2008). Moreover, additional information from further sources may validate manually provided information. As shown in this chapter, we can reuse data between requests (P2), but also need to leverage additional available information, such as employee surveys and application logs.

- *Support discussions:* A decision-support system should not only guide a streamlined procedure, but also take into account that decision processes often involve negotiations between stakeholders (cf. Section 10.4.2). Stakeholders discuss the needs, risks, and technical alternatives as part of the process and decision guidance should support this to enable its thorough use (P1, P5, P8).

10.7. Conclusion

This chapter examined the problems regarding decision support for authorization and evaluates a systematically developed, concrete prototype to derive insights on the practical viability of decision support for authorization. A formative evaluation explored particularly how well the numerous relevant benefit and risk factors can be collected in a distributed form to achieve the required efficiency. Participants stated insightful critique, both relating to problems with individual factors and the general concept, and positive feedback on the use of a decision-support system. We can derive the following insights on the design of decision guidance from the study:

Q11.1 *What kind of guidance for authorization-related decisions is effective and efficient?* Appropriate decision-support showed to be highly context-dependent (P8). For example, the general necessity of decision support depends on how difficult the decisions on granting permissions are and how distant policy makers are from the requesting staff. Moreover, a decision-support system needs to be embedded in the organization-specific procedures to offer sufficient efficiency for the frequent decisions (P7). The system also needs to reflect the actual decision-process, that is, support the negotiations between stakeholders if necessary.

Q11.2 *How can the information that is required for decision guidance be elicited?* There are two aspects to this questions: First, how can the factors be collected? The study in this chapter showed that it is important that the factors are comprehensible (through appropriate contextualization; P3) and respect the sensibility of certain factors. Second, how can the collection be implemented efficiently? As stated in the recommendations, we can implement the collection in a distributed manner as part of the policy change procedure – and integrate functional staff (P1). Flexible forms can adapt the required effort to the expected security needs and the reuse of existing data, both from earlier requests and from external sources, can reduce the effort for completing the forms (P2).

The study on decision support for authorization further gave insights on related questions:

Q4.1 *In what ways does the satisfaction of functional staff interrelate with functional disruptions, security awareness and expertise, and organizational culture?* / Q4.2 *How is satisfaction impacted for the authoring and making of policies?* The study participants mentioned how the denial or withdrawal of permissions affect individuals emotionally. It can thus be useful to increase the transparency of decisions (cf. Section 10.6) or even factor in this aspect as a benefit factor when evaluating whether to withdraw a permission.

Q5.2 *How must mitigations to undesirable circumventions be designed to be effective and efficient?* Decision support may help in two ways to guide functional staff to circumvent authorization only when adequate. First, it can improve the transparency of the decisions through more structured decisioning. Second, it can increase the awareness for the potential risks from modifications or disclosure of information through the participation in the decision-making (P6). Both ways may foster the overall acceptability of authorization denials and reduce the temptation to circumvent the authorization measure (P5).

Q9.1 *How do requests for changes to the policy take place in practice and what are the obstacles for functional staff?* / Q9.2 *How can we foster adequate requests for changes from functional staff?* Study participants mentioned misunderstandings of change requests as one problem in requesting changes. A structured decision procedure may reduce these problems. Together with the aforementioned increased transparency and awareness, this may result a higher overall acceptability of the change procedure and thus make it more likely that changes are actually requested.

Q8.1 *How do policy authors interact with policy makers and functional staff for implementing changes?* / Q10.2 *How can we uphold the procedures and prevent procedure circumventions?* The policy changes can take place informally through email or face-to-face, or formally through forms in an approval process. Particularly in the former case, we need to expect negotiations between functional departments and technical staff on how to grant permissions (cf. Section 10.4.2, P1). When honoring these needs, we may be able to reduce decisions that are taken outside of the defined process (P2).

Q1.1 *What type of organization requires what authorization measure and what are the influencing factors?* We observed a number of factors in the participants' contexts that influence the design of authorization measures and support earlier findings. The design of the procedure (who to address for changes, the degree of formalization, approvals) depends on organizational characteristics, such its configuration (Mintzberg, 1980). New factors that influence the design of the authorization measure are how the context defines the comprehensibility of decision factors and how clear the decisions on permissions are.

Future work on decision support for authorization needs to encompass thorough case studies of employing a decision-support system in practice in a number of authorization contexts to further explore its practical viability and its benefits. More specifically, we may study which factors are acceptable and completable by whom in what context, generating insights on Q11.2 *Guidance data*. Continuing the work on security awareness in this chapter, we may further examine the connection between participation and completing questionnaires, and gains in security awareness (e.g. related to Q5.2 *Circumvention guidance*). Further open questions that may be addressed relate to:

- *Regulation:* How can we systematically integrate regulatory obligations into guidance (Q3.1 *Regulation in decisions*)?
- *Additional uses of risk data:* The collected data on risks and benefits of permissions, usage contexts, and permission assignments can be considered a "risk policy." How can this information inform risk-based reviews (Q7.1 *Effective reviews*, Q7.2 *Review interaction*) and risk-based monitoring (Req 7 *Reviews and monitoring*, Q7.3 *Monitoring support*)?
- *Additional sources of data:* We need to integrate further sources of information on risks and benefits. How can we, for example, make use of results from compliance monitoring (Req 7 *Reviews and monitoring*), employee surveys, and related processes, such as ISMS (Q10.3 *Process integration*)?

11. Conclusion

And though their soil be not very fruitful, nor their air very wholesome, yet against the air they defend them with temperate diet, and so order and husband their ground with diligent travail, that in no country is greater increase and plenty of corn and cattle, nor men's bodies of longer life and subject or apt to fewer diseases.

Thomas More: Utopia (1516)

In the beginning of this dissertation, we briefly discussed how the political exercises authority. The central points were the acceptability of authority through legitimate laws and regulations ("rational authority": Weber, 1968) and its flexibility through specific judgments in court. We applied these approaches – acceptability and flexibility – to authorization in information systems, and formulated two research questions on the problems surrounding authorization and their mitigations, and five respective hypotheses. We explored approaches from security usability, and from secure and agile software engineering as the more concrete basis to address the problems – leading to the formulation of eight Authorization Principles. An analysis of authorization contexts allowed us to more clearly define the problem domain through a taxonomy and scope this dissertation to organizational authorization. We empirically studied the problems with authorization in practice to derive requirements and open research questions. To explore the open questions, we applied the principles in the development and evaluation of six artifacts that cover a broad range of authorization problems.

At this point, we are now in a position to draw overall conclusions on the usefulness and completeness of the principles. We will also discuss the methodology of this thesis, and the implications of the results for the design of authorization measures in organizational and other contexts. Considering research directions for authorization, we emphasize particularly ethics as an important area. The thesis concludes with a consideration of the implications of the results for the research in information security beyond authorization, and argues for participatory approaches and interdisciplinarity.

11.1. Discussion of the contributions

11.1.1. Critical evaluation of the artifacts

We explored six artifacts as practical contributions, grouped into four chapters, to test the validity of the principles and hypotheses, and address a subset of the identified detailed research questions. The selection of the artifacts aimed to cover a broad range of research questions (cf. Table 6.5 on p. 113 and the overview in Table 11.1), covering development efficiency, participation in policy changes, flexible authorization, and the adequacy of procedures and policy decisions.

One question is how universally applicable the artifacts are. To better understand their scoping, we relate characteristics according to the Authorization Taxonomy (cf. Section 4.1) to the artifacts in Table 11.1, both for the contingency factors (i.e. in what environment the artifact can be applied) and the design parameters (for what kind of authorization measure the artifact is useful). From the original intentions and the evaluation results for the individual artifacts, we can, for example, infer

	Framework	Policy model	Change support	Policy override	INDUSE	Decision support
Primary aim	Dev. efficiency	Policy participation	Policy participation	Flexibility	Process adequacy	Adequate decisions
Artifact type	Technical	Technical	Technical	Technical	Conceptual	Technical
Perspective	Development	Policy maker and author	Policy author	Functional staff	Security tactics	Security strategy, maker, functional
Empirical cxt.	SME Web app.	Diverse	Diverse	SME Web app.	Diverse	Diverse
Argumentation	Objective empirical	Subjective empirical	Subjective empirical	Objective/subjective empirical	Subjective empirical	Subjective empirical
Shortcomings	Remaining coupling of policy and enforcement	Adequacy is context-dependent	Abstraction, scalability, suggestions quantity	Lack of administrative control/tools, adequate processes, application integration	Descriptive model	Prototype only, scalability, context-dependent usefulness
Contingency factors						
Res. owner	Independent	Independent	Independent	Independent	*Organizational*	*Organizational*
Sec. needs	Low to high	*Low to medium*	*Low to medium*	*Medium*	Medium to high	*Medium*
Resource	*Structured*	*Structured*	*Structured*	*Structured*	Independent	*Structured*
Principal	*Structured* (roles)	*Structured* (roles)	*Human, structured*	*Human, structured*	Independent	*Human, structured*
Sec. expertise	*Medium to high*	*Medium to high*	Low to high	Low to high	Low to high	Low to high
Complexity	Dynamic system, policy; small to medium-sized, homogeneous system	Dynamic, low to medium heterogeneous, small to medium-sized policy	Dynamic, medium-sized policy	Dynamic policy and system, heterogeneous policy	Dynamic policy, system	Dynamic policy, medium heterogeneous system/policy
Design parameters						
Paradigm	Independent	Mandatory to discr.	Mandatory to discr.	*Mandatory/discr.*	Mandatory to discr.	Mandatory to discr.
Auth. scope	Organizational	Organizational	Organizational	Organizational	Organizational	Organizational
Management	Independent	Independent	Independent	Independent	Independent	Independent
Formalization	Independent	Independent	Independent	Independent	*Institutionalized procedures*	*Institutionalized procedures*
Enforcement	Mechanism	Mechanism	Mechanism	Social, mechanism	Social, mechanism	Social, mechanism

Table 11.1.: Summary of and applicable contexts for the developed artifacts

that four of the artifacts are independent of the specific kind of resource owner (they can be used to protect the security interests of individuals and organizations, content and entity owners) and two aimed at organizational owners, since the latter artifacts aim to support the interrelation of multiple stakeholders. Security needs are affected by what level of criticality the artifact is adequate for – for example, whether the minimally imposed overhead is acceptable for low security needs. Resources and principals need to be structured for five of the artifacts, for example, in roles. For the necessary security expertise of involved stakeholders, we find limitations such as the requirement of at least medium security expertise. Similarly, the adequate levels of complexity of the context may limit the contexts to those with policies of medium sizes. In case of the design parameters, the main limitation is that the authorization scope is generally expected to be of organizational type, since the artifacts assume a surrounding organization for which the policy is specified. Since the limitations of the artifacts' scopes were primarily necessary to arrive at concrete artifacts to be evaluated, we may extend those boundaries as part of future work to transfer the approaches to other authorization contexts.

We need to address two potential problems with the argumentative basis of the evaluation of the artifacts. First, the quantitative empirical arguments are derived from a very focused authorization context: a business Web application in a medium-sized enterprise. The reason is that the practical evaluation required a clear focus on a context for the case studies to generate rich and meaningful results (Flyvbjerg, 2007; Yin, 2009). Second, four contributions are based upon subjective empirical work. Generally, broad objective and quantitative studies may result in more profound evidence and thus in higher validity. However, the aim of the contributions was not to generally prove the superiority of specific approaches or claim comprehensiveness or representativeness, but to explore solutions to problems and derive rich insights for the application of principles and approaches. For this, HCI research generally applies qualitative methods for richer results (Adams et al., 2008; Glaser and Strauss, 1967)[1]. Regarding the credibility (construct and internal validity, reliability) and generalizability (external validity) of evaluation results, each artifact chapter describes the respective rationale of the study design and the scope validity. Generally, while most studies have a limited scope of generalizability, they, in sum, provide well-founded hypotheses. Overall, the exploratory approach allowed us to observe interesting shortcomings for the individual artifacts – also summarized in Table 11.1 – and derive well-founded recommendations (cf. Table G.2). Both represent useful points of departure for future research, which are discussed below, and for their implementation in practice.

11.1.2. Further empirical contributions

Apart from the evaluation of the artifacts, two further empirical contributions should be mentioned. One relates to the *secure development with agile methods* (Section C) and is based on interviews with ten agile practitioners. Despite this limited number of interviews, the study allowed us, in combination with a review of the analytical literature on the subject, to derive three principles for authorization measures in dynamic environments – with a particular emphasis on participation.

Second, we conducted the *Authorization Problems Study*, which was based on interviews with 120 individuals from a variety of backgrounds in a large corporation (Section 5.1). Together with the literature on problems with authorization in organizations, this study enabled us to derive a holistic model of those problems. Although the participants were recruited from a single company, the variety of contexts and the richness of the feedback resulted in a useful description of the problems and a profound basis for the problem model.

[1]See Section 5.1.1 for the rationale of conducting qualitative research

11.1.3. Supporting theoretical contributions

Apart from the main theoretical results (the Authorization Principles, discussed in Section 11.1.5), two further theoretical contributions supported and guided the research in this dissertation:

First, the *Authorization Taxonomy* describes and structures contexts of authorization measures (Section 4.1). The taxonomy is based on a systematic literature review, considering the authorization contexts in 145 publications. Using the criteria and their dimensions helped us to clearly describe organizational authorization contexts on the basis of the Structural Configurations (Mintzberg, 1980), and how organizational characteristics influence authorization measures (cf. Section 4.2). Moreover, the taxonomy allowed us to differentiate the priorities of the requirements with respect to authorization contexts (Section 6.2), to describe differences between the authorization contexts contrasted in the Authorization Problems Study (Section 5.1), and to evaluate the applicability of the practical contributions (Section 11.1.1). Thus, while not explicitly validated, the taxonomy showed to be useful in differentiating between contexts in the thesis, and improved the understanding of differences and applicability.

Second, from the thorough empirical analysis of the problems surrounding authorization in the Authorization Problems Study, we derived the *holistic problem model* (Section 5.3). The model is an abstraction of the problems in organizations, their interrelation, and the affected perspectives – thus, primarily aimed at providing a structured description (cf. Minsky, 1965). The model proved useful for structuring the requirements and the open research questions, and thus a substantial part of this thesis. However, this does not represent a validation of the model, since the requirements were based on the same empirical data that the model was originally derived from. On the other hand, all problems that we discovered in the course of the artifact evaluations fit this problem model, offering an indication of its usefulness.

11.1.4. Requirements and detailed research questions

From the problems surrounding authorization, we derived a set of requirements that authorization measures may need to fulfill – depending on the specific context (cf. Sections 6.1 and 6.2). Considering the existing approaches to these requirements, we arrived at a number of detailed open research questions to address in this thesis (cf. Section 6.3). We already gave a brief overview of how the developed artifacts cover the questions in Section 6.4 (Table 6.5). Having described and evaluated those artifacts, we now summarize the respective insights on the questions and requirements[2]. In the following, we will discuss the key points for each authorization perspective:

Organizational and functional perspective

As expected, authorization measures are multifaceted and the effectiveness, efficiency, and satisfaction depends both on the context (size, formality, risks) and on the design of the measure (mechanisms, procedures, accompanying measures). The artifacts and their evaluation indicate how the design needs to fit the context to reduce interferences with the functional work, and satisfied functional and administrative staff. We saw how mechanisms (e.g. policy override) in combination with adequate procedures (e.g. based on INDUSE) and further support (decision support) can lead to an improved authorization measure, depending on the context. Conversely, we also observed, in the case of policy override, how missing organizational measures can reduce the overall effectiveness of an authorization measure.

[2]The summary is primarily based on the results on the questions given in the conclusion sections of the artifact chapters and the respective argumentative basis given in Table 11.1.

While we also addressed how to improve overall satisfaction and how to reduce undesirable circumvention of authorization, the question of how satisfaction impacts effectiveness (Q4.3) and when staff will resort to circumvention (Q5.1) remains to be researched in future work. Also, further studies into a broader range of contexts could deepen the current exploratory findings and develop descriptive and prescriptive theories on authorization-measure design.

Policy making and authoring

Policy makers and authors have a very central role in achieving adequate authorization: The decisions they take and the changes they make to the policy will determine how frequently authorization interferes with functional work. We saw a variety of ways in which and by whom decisions are taken and changes are made, depending on the characteristics of the context. For instance, we are more likely to find informal decision-making in small and informal contexts, while responsibilities are in part assigned key users and role owners in more structured contexts. Nevertheless, we found that adequate (e.g. institutionalized) interaction between functional staff, policy makers, and authors is important and that we can support interaction through procedures, mechanisms, and tools. For example, change-support tools can support the policy author in interacting with policy makers and misunderstandings may be reduced through a decision-support system. Whereas we addressed most questions of these perspectives, we only touched on approaches for monitoring and policy reviews. Also, the approaches here could again profit from further, more thorough studies that target a broader range of contexts.

Security tactics and strategy

The perspectives of security tactics (processes) and strategy (decisions) are primarily supported in process design through the process model INDUSE, and in guiding decisions through the decision-support prototype. As indicated from both artifacts, change requests, procedures, and decisions are often approached on a rather informal level, but can benefit from an adequate degree of formalization – supported, for example, through a decision-guidance tool. Then, the more transparent and less arbitrary decisions may improve the change-request and procedure effectiveness. Further systematic case studies with broader scopes could clarify what type of processes and guidance is appropriate for what contexts.

Development

The practical contributions indicated that it is beneficial when developers strive for flexibility and comprehensibility of the policy, particularly to foster the interactions between functional staff, policy makers, and authors. For the models to actually achieve their goals (e.g. regarding flexibility), mechanism must be integrated in an adequate way. For example, we must signal to staff where policy override can be of use and help with the semantics of policies as indicated by the problems in policy management. We further need to study how the expressiveness and comprehensibility of authorization models interrelate. The interactions with developers are also important to enable dynamic changes to system functionality and policies. Here, an adequate controls framework can support the agility in development. One of the questions that we have not addressed in this thesis is how we can foster the trust in the reliability of the mechanism and controls. Generally, a more thorough examination from the development perspective would target a broader range of contexts, models, and mechanisms.

Concluding remarks

The authorization usability requirements were derived from a systematic elicitation of problems surrounding authorization. A broad literature review of existing approaches led to the formulation of the detailed research questions. Despite their systematic roots, the choice of the specific questions contains a subjective element. However, as shown in the conclusions of the individual artifacts and in the overview above, the questions worked well for providing a solid foundation for the research in this thesis, and for structuring the breadth of problems and the insights from the artifacts. Since several of the questions were not addressed or can be further studied more thoroughly, the detailed research question will also be useful to guide future research.

11.1.5. Usefulness and completeness of the Authorization Principles

The Authorization Principles (Section 6.4) are the main contribution of this thesis on how to solve problems surrounding authorization. The principles were derived from two sources: first, from security usability, for which we extended and generalized context-specific approaches from prior literature to apply to (organizational) authorization problems; second, from agile security, for which approaches either had similarly been mentioned in the relevant literature or were based on the results of the empirical study on practitioners' perspectives on security in agile development.

To assess the quality of the proposed principles, the set of principles can be considered a synthetic or prescriptive theory – providing qualitative (rather than quantitative) guidance, scoped to a specific domain (Dix, 2008). While, generally, theories need to explain and be verifiable (Popper, 1935/2002), generative theories, such as those common in HCI, are difficult to verify, since "the evaluation of generative artifacts is methodologically unsound" (Dix, 2008, p. 194): How can you show that the application of a theory – the principles, in this case – produces the desired outcome with numerous variables in design and development tasks? Dix (2008) argues to focus on the "justification" (p. 195) of the theory, that is, on the "audit trail" (ibid.) from the original assumptions to the derived theory.

Specifically, to evaluate – and justify – the usefulness of the Authorization Principles, we need to assess their *correctness* (Does adhering to them actually improve the usability of authorization measures?) and the *completeness* of the list (Does the list include all necessary aspects?). Further important properties of a theory are the simplicity[3] (Is the list as short as possible?) and its transferability (Can the list be applied to other problems in organizational authorization?) (Dix, 2008).

For the Authorization Principles, we can draw on two sources for their validation that are both independent of the original formulation of the principles: The application of principles in artifacts and the relation of the principles to the recommendations derived from empirical work. Considering the findings from the application of the principles is most useful for assessing the principles' correctness: Which principle did actually improve usability? A summary of the insights and shortcomings derived from applying the principles is given in Table G.1 in Appendix G, including the respective coverage and argumentative basis. In contrast, the second source (the recommendations, cf. Table G.2), is rather appropriate to assess the completeness of the principles: Are all the recommendations from the empirical work covered by the principles?

We will first discuss the correctness and transferability separately for each principle before considering the entire list's completeness and simplicity[4]:

[3]There are a number of arguments on why simpler theories are better. For example, Popper (1935/2002) argues that simpler theories are easier to falsify. Simplicity is also one important criterion of Occam's razor.

[4]This discussion only considers the scope of this thesis – organizational authorization. For a brief discussion of applying the principles beyond organizations, see Section G.1.

P1 Interaction and participation

Integration is the primary research paradigm of this thesis and it is applied on several layers. On the layer of practical approaches to authorization problems, integration is represented through the principle of interaction and participation. This principle is applied in each of the artifacts, fostering interactions between administrative staff and developers, among administrative staff, or the participation of functional staff. These interactions resulted in "knowledge integration" (Fischer, 1999), for example, when functional staff provided decision factors in the *decision-support study*. Moreover, we could also witness "informed participation" (Brown et al., 1993) with the authorization measure as the "boundary object" (Star and Griesemer, 1989) to support the interaction – for example, when policy makers and authors perceived the *change support* as a good basis to discuss policy-change alternatives.

In most cases, integration was seen positively; for example, participants of the study on *decision support* stated that transparency of decisions and the overall satisfaction could improve. Conversely, the study of *policy override* in practice resulted in an overuse of the participation means (policy override) due to a lack of accompanying organizational measures. Similarly, some participants in the studies were careful about its transferability to their specific context, particularly to integrate policy makers with a low technical affinity. Thus, assuming that participative measures are adequately encompassed with organizational measures and given an adequate context, the artifacts strongly indicate the usefulness of this principle.

P2 Reduced burden

Reducing the burden of individuals interacting with information systems has been a wide-spread approach for HCI and security usability. We applied the principle for all six artifacts, reducing functional staff's change-request effort, policy makers' decision effort, authors' change effort, and developers' implementation effort. We found that any approach to actually reduce the burden needs to fit the context – for example, for the different *modes of policy management* or through contextualization of the elicited factors in *decision support*. While the impact of the reduced burden varied so that the concrete approaches may not be universally applicable, there are strong indications that this principle is useful in improving authorization usability.

P3 Concreteness

We applied the principle of concreteness to improve comprehensibility in two ways: to reduce the cognitive effort of changing abstract policies and to reduce the abstractness of factors in policy decision-making. For policy management, while concrete change suggestions from the *change support* were useful, this at the same time also brought about new abstractions on the actual changes because the actual policy is rather distant in this case. For the collection of the factors for *decision support*, problems with the abstractness were prevalent and increased concreteness improved comprehensibility. Even though further studies need to extend our knowledge on how the concreteness is best achieved (e.g. contextualization), this principle shows to be a good candidate for authorization usability.

P4 Multidisciplinary

Each of the artifacts required technical and non-technical approaches. Most artifacts are technically informed from information security, employing models or algorithms relating to authorization. We relied on further particular technical disciplines for effective and efficient approaches, including software engineering for the integration of *enforcement controls* and artificial intelligence for *change support*. In case of technical artifacts, the adequate interaction of stakeholders – whether developers, administrative, or functional staff – with the mechanism depended on socio-technical approaches. Organizational, risk, and decision sciences provided the basis for the processes and the decision guidance surrounding authorization. The multidisciplinary approach was not only necessary to realize the artifacts, their evaluation even indicated the necessity of considering additional disciplines – for example, for policy override, for which organizational measures were lacking.

P5 Individuals' behavior

Considering the rationale of individuals' behavior guided the design of the *policy-override mechanism* (offering convenient, but controlled options for exceptional cases) and the participatory approach of the *decision-support prototype* (increasing acceptability of restrictions through participation). While this principle lacks a broader argumentative basis in this thesis, the policy-override study indicated how useful this approach is in two ways: First, it reduced the number of potential circumventions of authorization. Second, rational behavior also led to an over-use of the mechanism, indicating that policy override was perceived as an easy option even for continuous application and that, conversely, adequate long-term mitigation strategies (changes to the policy or to the application) were perceived as requiring too much effort. The statements of the participants of the decision-support study indicate that transparency and improved communication are a means to influence behavior. Even though only applied to two artifacts, ten recommendations relate to this principle, demonstrating its importance.

P6 Security awareness

Raising the general security awareness to influence the individuals' behavior was only directly applied and evaluated for the *decision-support prototype*. The study indicated that security awareness can be increased through participation in the decision-making process, but focused research is necessary for more solid conclusions. The importance of this principle additionally showed in the study on *policy override* in practice. Here, increased awareness for the drawbacks of policy override could have led to a more adequate use of the mechanism. Raising the awareness should thus be an important factor in influencing the behavior of functional and administrative staff. Overall, six recommendations relate to improving the security awareness.

P7 Design for dynamics

Several of the practical contributions, both technical artifacts (e.g. the enforcement framework and the policy-override mechanism) and non-technical (e.g. INDUSE) aimed to improve authorization in dynamic environments. The addressed dynamics related to the security needs, policies, or system functionality. This principle helped to reduce the functional disruptions (e.g. through *policy override*), and the administrative (*decision support*) and development overhead (*enforcement*), indicating the usefulness of this principle – given that the context allows for low-overhead approaches (cf. INDUSE). Moreover, five of the recommendations relate to this principle.

P8 Tailor for context

In four artifacts, we explored how the authorization measure or supporting tools have to be designed to fit the diverse contexts of authorization. Most generally, the INDUSE study showed the interdependence of authorization process design and the characteristics of the organization (centralization, formality). The evaluation of *policy management* demonstrated how the adequate mode depends on the stakeholders' backgrounds, and organizational and policy complexity. The *decision-support prototype* showed how important it is that the guidance fits the process and flexibly adapts to different security needs for efficiency. The empirical studies in this dissertation – both on individual artifacts and the Authorization Problems Study – further demonstrated the variety of contexts and their respective requirements. This principle thus should help remind designers to take the specific contexts into considerations – between as well as within organizations.

Completeness and simplicity of the principles

Concerning the correctness, P1 *Interaction and participation*, P4 *Multidisciplinary*, and P2 *Reduced burden* are best represented and show high impact. Conversely, others – particularly P6 *Security awareness*, P3 *Concreteness*, and P5 *Individuals' behavior* –, despite appearing useful, would benefit from further examination, both more thorough and broader in scope. When addressing the simplicity of the set of principles, we cannot compromise on the principles' approaches due to their usefulness, but might at most subsume less important ones under others. One candidate is P3 *Concreteness*, which was only applied to two artifacts, is only related to three recommendations, and is covered by P6 (concreteness is often necessary to increase awareness) and P2 (concreteness improves understanding and reduces learning effort). However, its importance for the usability of authorization appears to be high enough to justify its explicit naming. All other principles were either broadly successfully applied or relate to numerous recommendations and should thus be kept.

Regarding the completeness of the principles, Table G.2 shows that all recommendations are covered by at least one principle. Additional aspects in the recommendations listed in the table as not explicitly covered by the principles can still be related to the existing ones: The *satisfaction* of stakeholders, *observing* the authorization measure for potentials of improvement, and *intuitive user interface* are all implicitly addressed by or a prerequisite for P2 *Reduced burden*. Improving the *decision-making* can be achieved through P2 *Reduced burden* and P6 *Security awareness*. Deriving *consequences* from observations influences behavior (P5) and is necessary to reduce the burden (P2). Adequate *procedures* and *flexibility* are both more specific forms of P8 *Tailor for context*.

Overall, the validation shows how arbitrary the selection of the principles is. While we may reduce the list to a small number of core principles to increase its simplicity, we would lose the ability to emphasize important corollaries. Conversely, the more specific additional aspects, even though important and worth noting here, would increase the redundancy if added as principles. Overall, the selection of the principles – particularly the granularity of the list – will necessarily be subjective.

11.1.6. Hypotheses and main research questions

The first two of this dissertation's hypotheses target the manifestation of problems surrounding authorization. Based on supporting examples from the problem analysis, their validity was already discussed after the problem analysis in Section 5.4. The conclusion there is that, indeed, problems affect several interrelated perspectives on authorization (H1) and interfere with their interaction (H2). To complete the evaluation of the hypotheses, we will consider the second three hypotheses on problem mitigation in the following, before we address the main research questions from the introduction.

Hypotheses on problem mitigation

Two aspect are of interest with respect to the validity of the hypotheses on problem mitigation: First, and more importantly, whether the hypotheses are *correct* and are, accordingly, reflected in

Hypothesis	Corresponding principle	Related principles
H3: Technical/non-technical	P4 *Multidisciplinary*	Applied to P5, P6
H4: Dynamics of the environment	P7 *Design for dynamics*	Supported by P2, P5, P8
H5: Interaction and participation	P1 *Interaction and participation*	Supported by P2, P3, P6

Table 11.2.: The relation of the hypotheses to the Authorization Principles

the principles (summarized in Table 11.2); second, whether they are *complete* and cover all the principles. We examine both properties in the following:

Hypothesis H3, which calls for a combination of technical and non-technical approaches, largely corresponds to the principle of multidisciplinary approaches (P4). Other principles (P5 *Individuals' behavior*, P6 *Security awareness*) successfully combine non-technical and technical approaches; their broad application supports the hypothesis. Moreover, the policy-override study showed the consequences of overly relying on technical approaches: a reduced effectiveness of the measure.

Accounting for dynamics of the environment (H4) received a similar amount of coverage in the artifact development through principles: Apart from the corresponding principle (P7), the hypothesis is supported by further principles (P2 *Reduced burden*, P5 *Individuals' behavior*, P8 *Tailor for context*). The empirical results related to these principles offer a strong indication in support of the hypothesis. For example, the data on the enforcement implementation showed the consequences of a lack of provisions for dynamics: a high implementation effort.

The third hypothesis on problem mitigation concerns the integration of perspectives, the interaction of stakeholders, and their participation (H5). The empirical work on the application of the corresponding principle (P1) as well as of the supporting principles (P2 *Reduced burden*, P6 *Security awareness*, P3 *Concreteness*) support this hypothesis. The positive effect can be particularly strongly observed in the studies on decision and change support: reduced effort for policy changes due to improved interactions.

The existence of corresponding principles for each of the hypotheses indicates the *correctness* of the hypotheses. For their *completeness*, we need to consider whether all principles can be subsumed under the hypotheses, since we showed in the previous section that the principles are complete with respect to our empirical work. First, it is implausible that P2 *Reduced burden* and P3 *Concreteness* are *only* relevant regarding the dynamics and integration in authorization. Instead, they appear to also increase usability for infrequent changes by dedicated experts. Along the same lines, P6 *Security awareness* and P5 *Individuals' behavior* solve problems independent of the dynamics – for example, preventing circumvention in static environments. Lastly, it is also obvious that P8 *Tailor for context* not only helps in dynamic contexts, but can particularly improve effectiveness in cases where more reliable procedures may be employed. Thus, the hypotheses on problem mitigations are correct (are reflected in the principles) and served as a good point of departure for exploring mitigating principles – but cannot be considered complete.

Main research questions

We asked two main research questions in the introduction (Section 1.2):

1. *What authorization problems do stakeholders face in information systems and how do their problems and perspectives interrelate?*

   We discussed this question subsequent to the problem analysis (cf. Section 5.4), referring to the model of perspectives on authorization (Section 4.4) and the holistic model of authorization problems (Section 5.3). We identified a broad set of authorization problems for six perspectives

– functional staff, policy makers, policy authors, developers, strategics (decisions), and tactics (processes) – and the problems' interrelations, enumerated in Table 5.4 and summarized in the layered diagram in Figure 5.4.

2. *What methods, procedures, and technologies are required to address the authorization problems and achieve usable authorization measures in information systems, and what fields of research do we need to draw upon?*

   Concluding from the practical contributions and the related empirical work, we observed that with the broad range of authorization contexts, there is no "one grand solution" to the usability problems surrounding authorization. However, addressing the identified detailed open research questions and applying the Authorization Principles (including a multi-disciplinary research approach), authorization measures can become more effective and efficient, and increase the productivity of and satisfaction within organizations. Due to the variety of contexts, problems, and mitigations, this dissertation primarily offers guidance on addressing problems with authorization through principles, but cannot offer a prescriptive theory on the concrete measure design with respect to context characteristics.

Both research questions were posed broadly and make it difficult to arrive at a definitive answer. However, addressing the questions allowed us to explore a variety of problems and mitigations from a broad range of contexts. While the model of authorization problems and the Authorization Principles are rather comprehensive for the organizational scope, the mitigations through concrete artifacts could only be shown to be useful in solving specific problems for a limited scope.

One might generally criticize that the results are primarily qualitative: They rather address *how* we can improve usability, instead of – quantitatively – *how much* improvement can be achieved. However, it is common in HCI research to focus on qualitative research – particularly, if the problems are not clear at the beginning (Adams et al., 2008). Moreover, with the breadth of contexts, quantitative results would have forced us to focus on a smaller evaluation scope and to pose narrow research questions, reducing the richness, usefulness, and transferability of the results[5]. Thus, despite the qualitative nature of the results, the proposed principles and mitigations should provide an important contribution to improve authorization usability in practice and to guide further research.

11.1.7. Critical evaluation of the methodology

The methodology underlying this thesis is to thoroughly analyze the problem domain, to scope the treated area to a specific context, and to identify and structure the problems in the area – considering an extensive vertical scope of diverse perspectives and problems. Based on these problems, requirements and open research questions lead to the development of concrete artifacts to explore the open problems and mitigating principles. The two central aspects of this methodology are:

- *Vertical integration with a limited horizontal scope:* A broad range of problems along the vertical axis – that is, the range of problems from different perspectives (e.g. functional and administrative staff) – but a limited scope of distinct contexts to cover (e.g. only organizational authorization),

- *The exploration of problem mitigations through artifact development:* Approaching the problems by developing a selection of artifacts and evaluating their respective characteristics for insights on solving the problems.

[5]For further discussion of the qualitative research approach, refer to Section 5.1.1.

Both aspects have implications that should be interrogated here as a critical evaluation of the methodology. We formulate and refute three potential pieces of critique in the following:

> *"The vertical integration of problems reduces the analytical and empirical depth when compared to focused single-issue research approaches."*

It is a valid argument that further depth can be reached when focusing on individual problems. However, the hypothesis underlying this methodology is that usability in information security is suffering from the fragmentation that results from the ever-increasing specialization in research[6]. Highly effective and efficient solutions to problems from very focused research can be ineffective or inefficient when applied in practice. One example from authorization are editing tools: Even if they are perfectly usable for policy authors and would thus be considered effective and efficient, broadening the focus to include the actual procedures in organizations shows that authors need to discuss restrictions with policy makers so that such a tool may be suboptimal if it does not support the interaction.

The integrative approach in this thesis instead emphasizes the consideration of a broad range of perspectives on authorization measures. This focus naturally leads to a more exploratory, hypothesis-building approach than objective and representative empirical evidence. The goal is rather to understand and explain problems and potential mitigations than to universally prove the mitigations efficacy. This thesis' approach is particularly useful when the contexts are very diverse and mitigations need to be adequate for the context to be effective and efficient. Then, any representative empirical proof is difficult to acquire without imposing a very narrow scope, which offers little added value in practice[7].

> *"A broader horizontal scope of more diverse contexts with less vertical integration would offer insights from the overlap of problems and on the universality of mitigations."*

One could choose to integrate horizontally instead of vertically, considering only singular problems or perspectives on problems, but from more diverse contexts than possible with the vertical spread in this thesis' approach. While the insights from the horizontal breadth would certainly be valuable – particularly on a theoretical level to craft complete models of the problems and mitigations – this approach would not solve the problems from the fragmentation of the field. Moreover, the authorization problems are often related to the interaction of people. For example, functional staff need to interact with administrative staff to achieve adequate policies. Here, it is more fruitful to aim to solve the problems more fundamentally by vertically integrating the problems than horizontally. Furthermore, universality as to be gained from broad theoretical models is unlikely to have the necessary level of detail to further practical mitigations due to the above-argued context-specificness of mitigations. Lastly, the scope of authorization in organizations cannot actually be considered "narrow" as the broadness within this scope has shown; organizations are very diverse. Even if we forgo a number of additional characteristics of further contexts that could be exploited for optimization in those contexts, we see many of the challenging problems in the organizational scope: the necessity of complex coordination, complexity and dynamics, and the necessity of efficient and effective security. This aspect is further discussed below in Section G.1.

> *"Focusing on an individual artifact allows for more sophisticated technical results and a more thorough empirical validation."*

[6]Dogan and Pahre (1989) argue for social sciences that the increasing specialization and fragmentation requires researchers to form informal hybrids as interdisciplinary fields. We elaborate on interdisciplinarity in Section 11.4.4.

[7]Again, compare the discussion of the qualitative research approach in Section 5.1.1.

A narrower focus on one artifact would indeed allow for a more thorough exploration and validation. However, a higher validity of results would be primarily useful if we strove for universality. In contrast, as argued above, this is not the aim of this thesis and that approach would not result in the most useful results. Instead, this thesis' approach is to understand the problems more thoroughly and derive principles for addressing the problems in a more general way. This, in turn, requires a number of artifacts to cover the vertically integrated scope. For instance, this approach allowed us to both study how to design procedures surrounding authorization and how to integrate decision support in the processes.

Reviewing the critique and the arguments defending the chosen approach, we can find that the representativeness and validity of individual artifacts from this thesis may be threatened. However, the more global contribution – the exploration and understanding of problems and mitigations – is only affected to a limited extent by this critique, since we can describe and argue for promising mitigations for the problems.

The methodology presented here is thus suited for research areas that are in need of exploratory research and of an integrative approach. Particular indicators are the fragmentation of the research and a research bias towards individual subfields, when problems actually interrelate and affect the interaction of people from a variety of perspectives on the problems. A further indicator is the diversity of contexts in which the problems occur and for which the mitigations need to apply.

In comparison to the traditional, more focused methodology, this thesis' approach can be considered closer to practical application. More theoretically sound results that result from focused research are often primarily shown to be effective in isolation. This approach can lead to problems when the produced artifacts are confronted with the complexities of reality[8]. While both approaches – the focused and the broad one – are needed in research, the focused one dominates information security research. The apparent consequence is that research results are primarily of rather theoretical nature and are thus often unsuitable as short-term solutions to the pressing practical security problems.

11.2. Implications for authorization measure design

From a more practical perspective, the important question is how usable authorization is achieved in concrete organizations. For abstract guidance, we can refer to the Authorization Principles. More specific are the recommendations that we derived from evaluating the individual artifacts (cf. Table G.2). However, even though the full list of principles and recommendations is helpful (cf. Section 11.1.5), we can summarize them as follows – loosely corresponding to H5, H4, and H3, respectively:

- *Design authorization measures for participation (P1):* We need to source the tacit knowledge of the stakeholders from the diverse perspectives on authorization in organizations to achieve an effective and efficient measure. This thesis analyzes where interaction is beneficial (Section 5.3) and how we can solve problems through integration and participation (P1) – for example, between policy authoring and controls integration through an adequate enforcement framework. We address the motivation of individuals (P6, P5), reduce their burden (P2) – for example, by providing decision guidance for policy makers –, and improve the comprehensibility of policies and decisions (P3).

- *Account for the context (P8):* Organizations and the information systems that are protected by authorization are diverse (cf. Section 4.1). Many organizations and systems are subject to

[8]We further argue for a more holistic approach to research in Section 11.4.4.

	Security needs	Complexity	Dynamics	Expertise
P1 *Interaction and participation*			+	
P2 *Reduced burden*		+	+	−
P3 *Concreteness*				−
P4 *Multidisciplinary*				
P5 *Individuals' behavior*	+		+	−
P6 *Security awareness*	+			−
P7 *Design for dynamics*			+	
P8 *Tailor for context*	±	±	±	±

Table 11.3.: Correlation between context characteristics and Authorization Principles (+ positive, − negative correlation; ± either way)

change and require flexible and efficient measures; other organizations require a high level of confidence in changes to the permissions. In this thesis, we improve the flexibility (e.g. through policy override; P7), adapt processes to the context (INDUSE) and demonstrate how security needs and stakeholders' expertise govern decisions (decision support; P2, P5, P6).

- *Combine technical authorization mechanisms with non-technical means:* Technical mechanisms are not the only means of protection. Instead, organizations should combine the mechanisms with socio-organizational measures. If the security needs permit, and social controls and the individuals' awareness can be relied upon (P6), the flexibility of the measure can be increased (policy override) and with it the effective security (P5). Moreover, authorization measures also require adequate procedures for requesting and deciding on changes (P8, INDUSE). Drawing on the technical and the socio-organizational can improve the effectiveness and efficiency of, and the satisfaction with the authorization measure (P4).

To explore the relative importance of the principles for concrete organizational contexts, we can analytically correlate the principles to context characteristics (cf. Chapter 4) as shown in Table 11.3: We find that most principles positively correlate with the security needs, complexity, and dynamics of the context (e.g. higher security needs lead to an increased importance of the security awareness). Conversely, the security expertise is negatively correlated with several principles (higher expertise will reduce the importance of those principles). Lastly, P8 *Tailor for context* relates to all characteristics, independent of their manifestation since the authorization measure has to adapt to the characteristics in any case. Now, we can refer to characteristics of typical organizations, given in Table 4.8, to determine the relative impact of the principles with respect to concrete organizations – for example, particularly designing for dynamics (P7) in Simple Structures and Adhocracies.

11.3. Directions for future research on authorization

The results from this dissertation indicate a range of directions for further research. We find research challenges in extending the practical research on the artifacts in this thesis, by targeting those detailed research questions that are not comprehensively covered, and by further following the central theme of this dissertation: integrating research approaches and the perspectives of affected stakeholders.

11.3.1. Expanding on the artifacts

As indicated in the evaluation of the artifacts, one of the limitations of some of the practical contributions is the breadth and depth of the respective empirical work. Regarding breadth, it would, for instance, be useful to analyze the integration of controls with an agile authorization framework in further contexts (cf. Chapter 7), or implement and evaluate policy override in further contexts (cf. Chapter 8). These contexts might also extend beyond organizational authorization (cf. Section G.1). For depth, we could, for example, benefit from thorough case studies on designing and implementing adequate authorization-processes in practice (cf. Chapter 9), and from implementing and evaluating decision support in organizational practice (cf. Chapter 10). Both breadth and depth would further our understanding of authorization problems and mitigations in practice.

The conducted empirical evaluations also indicated avenues for practical work to improve the individual artifacts and generate further research results. In case of the authorization framework, interesting options include the optimization of the authorization API for efficiency – particularly the passing of context to the decisioning – to further decrease the threshold for the enforcement implementation. On a higher level, the policy-override study showed shortcomings of the override measure that would be worth rectifying in future work. This includes adequate processes to ensure consequences from policy override, improved administrative tools – particularly for efficient monitoring of override use – and improved application integration to highlight to functional staff where they can benefit from policy override.

On the organizational level, the process model INDUSE currently offers a descriptive model to support the modeling of authorization development processes. From more careful analysis of the interrelation of context factors and successful or problematic process designs, we could extend the process model and derive a prescriptive model from correlations. Reflecting the characteristics of the respective context, authorization-process building-blocks could then be proposed to improve the process. The building blocks could, amongst others, emphasize the stakeholder interaction, establish performance indicators for continuous process improvement, provide for compliance with regulatory obligations (Q3.2), and enable the integration with higher-level organizational processes such as ISMS (Q10.3). Ideally, the prescriptive model would be implemented as a supporting tool for the process design.

Concerning policy decisions, we could expand on the decision guidance through decision support. The current guidance model could be enriched with a systematic integration of regulatory and other high-level decision-factors (Q3.1). The data collection could be supported by making use of existing data that is explicitly or implicitly collected in organizations, for instance, from staff surveys or application-usage logs (Q11.2). Generally, it would be useful to develop the current prototype into a tool that could be employed and evaluated in an organizational setting. An important aspect to address would then be the tool's flexibility regarding the context, offering both extensive and compact data collection, depending on the involved risks. We need to develop a methodology to derive the context-specific decision support and decision factors. Moreover, we could further emphasize the integrative nature of the decision-support tool and position it as a platform for the discussion of policy changes, supporting the feedback loops that occur currently in implicit procedures when negotiating the necessary permissions (Q8.1).

11.3.2. Further addressing the open research questions

As indicated above in the evaluation of the requirements and open research questions, this thesis covers a broad range of questions, but not all are covered or covered comprehensively. For the covered questions, the conclusions of the individual artifact chapters include further research that could be undertaken for the requirements and questions. Apart from deepening and broadening existing empirical work and the extension of the existing artifacts, both described above, potential future research concerning the detailed questions is discussed below:

Organizational and functional From the organizational perspective, one important aspect is the design of effective and efficient authorization measures. Similar to the work on adequate authorization processes, further research on measure design could study the correlation of successful and problematic measures with context characteristics and design factors (Q1.1). From the correlation, we would ideally be able to develop a prescriptive model of authorization measure design (Q1.2). This effort complements (and needs to be coordinated with) further work on the process design, discussed above as extension to the work on INDUSE.

Further advances on the adequacy of authorization measures could be realized by systematically studying the way in which authorization measures cause disruptions, and then use the gained insights to formulate recommendations to reduce their impact (Q2.1). A related aspect is the connection between the satisfaction of individuals and the effectiveness of authorization measures (Q4.3). In both cases, we need to apply research methods from the social sciences and their models, for example, on power relationships and social learning (cf. Section 11.3.4), to gain insights on how to improve authorization measures for the affected stakeholders. Similar methods are necessary to more comprehensively address circumventions – both to estimate their likelihood and to prevent *undesired* ones – acknowledging the need for flexible handling of exceptional situations, but guide the circumventions (Q5.1, Q5.2).

Policy making and authoring While the authoring of policies is well covered by prior literature and in this thesis, we can further expand on how decisions on policy changes are made. The work on decision support focused on guiding the decisions and improving the related procedures. This leaves room for more thorough analysis of the decision process itself. Similar to the models on process and measure design, empirical work on successful and problematic forms of decision-making and correlated decision and context characteristics could lead to a prescriptive decision-model (Q6.1). The decision model could then be employed to recommend responsibilities and how the necessary information can be acquired.

Reviews of authorization policies were only touched upon in this thesis. The effective security would profit from more effective reviews. Interesting fields here concern the design of adequate review procedures and the integration of different perspectives in this task (Q7.1, Q7.2). Risk estimations from decision support could be reused for more efficient risk-based reviews. Cooperative reviews, including self-reviews by functional staff, could similarly improve the review efficiency – given adequate organizational and technical means, that is, procedures and tool support. In similar ways, we could address the problems surrounding monitoring, which also incurs a high effort. Adequate technical means could streamline the procedures, offer support based on the risk estimations, and offload the monitoring to broad ranges of stakeholders to efficiently identify problems with policies in daily application use (Q7.4). Moreover, we could carry over anomaly detection to authorization problems as known from network security (Q7.3).

Security tactics and strategy We already addressed a range of problems with the processes in authorization measures and the strategy of how policy decisions are made. In addition to prescriptive models on procedures and decision guidance, further research could employ sociology to focus on power structures and compliance – improving our understanding and the influencing of change requests (Q9.2) and the circumvention of procedures (Q10.2).

Development A number of further technical aspects can be addressed from the development perspective in future research. For example, the effectiveness of the flexibility of authorization mechanisms need to be further explored, particularly comparatively: contrasting the approaches of delegation, risk-based policy models, and policy override for different contexts (Q12.3). More generally, the interaction of the technical factor of expressiveness with the perceived complexity and the comprehensibility leaves room for further research: What factors influence the comprehensibility (e.g. its concreteness P3) and when should we resort to simpler models (Q12.1, Q12.2). Can modular authorization models be helpful to arrive at adequate models for specific contexts?

For effective integration of non-technical staff in the policy authoring, one identified barrier is the necessity to understand the application model, both to choose the required permission and to formulate authorization constraints. In both cases, improved mechanisms could take these problems into account and support the request of and decision on specific policy changes (Q14.1).

Further interesting challenges regarding the mechanism can be found in the assurance of individuals and organizations on the reliability of the controls, particularly for extensible systems, such as third-party applications on smartphones (Q14.2). Here, the user granting permissions to applications needs to trust that the system and the application adequately protect the personal data from other applications. Viable approaches could include usable static and dynamic analyses of the applications. A related problem, but affecting the developer instead of the functional and administrative staff, is integrating authorization controls reliably in the application program flow (Q15.1). Static or dynamic analysis methods of programs can help here as well. For instance, a control or data flow analysis may indicate missing authorization enforcement on specific paths.

Further tool integration A common theme of the future work are supporting tools, both for the artifacts and for further open questions, such as the measure design. Considering the thesis' theme of integration (P1), we could apply this approach on a practical level to these tools. Considering the developed artifacts, a comprehensive decision and change-support tool could offer risk-based change suggestions, based upon concrete change requests, and, at the same time, offer the traceability of policy changes and their rationale (Q13.1). Further, an INDUSE tool, as described above, could not only support the process design, but also generate the configuration for the integrated decision and change-support tool. The integrative approach does not have to stop at the developed artifacts, though. A common model for authorization usability tools would facilitate the integration of all activities of the authorization lifecycle. This could begin with the design of the measure, decision strategies, and processes, and continue with the operative part of requesting changes, making decisions, and authoring policy-changes. Related activities would include policy reviews and monitoring. The model could then also interact with other organizational processes, such as ISMS.

11.3.3. The ethics of authorization

One important aspect that needs to receive further attention in future research is how we can design ethical authorization measures. Whereas we touched upon the legitimacy of restrictions as one requirement to increase the satisfaction of the affected stakeholders and thus to increase the overall effectiveness of the measure, legitimacy is only one aspect of ethics in authorization. In the grander picture of ethics, we primarily need to consider authorization a means to exercise power since "technology is society made durable" (Latour, 1991). Power is exercised either technically through mechanisms in systems, socio-technically by combining social factors with the technical mechanisms, or socially by relying entirely on the social factors (cf. the socio-technical design of authorization enforcement in Section 4.2.2). Critical philosophers scrutinize the tacit aspects of exercising power and examine the "workshops where ideals are manufactured" to manipulate (Nietzsche, 1913/2003, p. 27) – and "criticize the workings of institutions that appear to be both neutral and independent... in such a manner that the political violence that has always exercised itself obscurely through them will be unmasked" (Foucault in Chomsky and Foucault, 2006, p. 41). In case of authorization and the

Authorization Principles put forward in this dissertation, we need to reconsider the persuasion and manipulation that organizations employ as the means to achieve effective security in this respect. Persuasion appears most openly in P5 *Individuals' behavior* where we attempt to predict and change the behavior by applying insights from cognitive psychology and sociology to the decision-making of individuals. One way to influence behavior is to make restrictions legitimate by increasing awareness for the security issues at stake and explaining the organization's rationale (internalization, P6), and by increasing participation (P1).

In the broader context of society, the ethics of persuasion and manipulation have been discussed in a number of fields. Very prominent is the case of advertising, for which practices such as exaggerated claims and explicit persuasion are considered to violate individuals' rights, to cause human addictions, and to be simply dishonest (Tsalikis and Fritzsche, 1989). More subtle is the situation in case of medical communication intervention, which should improve the health of people, but cause "concerns. . . related to persuasion, which can infringe on people's autonomy or privacy" (Guttman, 1997, p. 109), since "justifications embedded in health interventions tend to be paternalistic" (ibid.). In case of virtual reality, Brey (1999) argues that designers are responsible for encouraging ethical behavior of users and an ethical representation of the virtual world (e.g. concerning stereotypes and portrayed values), but need to avoid paternalism. Fogg (2003), writing on "persuasive technology", is more careful, draws on moral relativism, and only calls for "those who design, study, or use persuasive technologies. . . to become sensitive to the range of ethical issues involved," since there is "no single ethical system."

The second aspect of the ethics of authorization is what ends are accomplished with the measures. For this, we can turn to the organizational ethics of decision-making, where individuals as part of organizations need to be aware of the dangers of the rationalization of organizational goals – potentially leading to dehumanization (Clegg et al., 2007). Clegg et al. (2007) see "management's task. . . [as] one of enhancing and maintaining structures within which moral agents face, understand and act" responsibly.

What follows for the ethics of authorization? First, we need to interrogate the principles and recommendations for manipulation/persuasion and paternalism. One result could be to establish additional principles to ensure ethical authorization measures[9]. A *principle of ethical authorization measures* could demand that the technical, socio-technical, and social means of enforcing authorization are employed in an ethical way with respect to the specific cultural framing. For an organization, ethical authorization would not only serve as an end in itself, but may also support the overall organizational culture and allow stakeholders to take part in the organization more wholeheartedly (cf. Clegg et al., 2007). Since such a principle would be rather imprecise, it would need to be instantiated for specific cultural contexts. Nevertheless, moral actors would be certainly faced with a broad spectrum of difficult decisions with regard to what is considered ethical, both within a specific society and interculturally.

More concretely, a *principle of the transparency of the means* could help to resolve the ethical problems of paternalism and manipulation by respecting the dignity of the affected individuals and indicate how authority is exercised. On a technical level, transparency could mean to indicate additional activities and information not available to the current user – even though this kind of information can already pose a security threat in itself. From this principle, it also follows that we need to require consent to the logging of activities to respect the privacy of the users. On a social level, we would need to avoid covert and manipulative awareness campaigns or participation that should

[9]Since ethics are a common means of influencing behavior (cf. Clegg et al., 2007), the additional principles can be subsumed under P5 *Individuals' behavior*. Thus, they would not contradict the completeness of the existing set (cf. Section 11.1.5).

only shape behavior – for instance, by openly communicating the actual goals of the campaign or approach to the affected individuals.

11.3.4. Further integration of research disciplines

On a research level, we can integrate further research disciplines to address the most challenging problems surrounding authorization: an overall adequate measure (Req 1, Req 4, Req 2), the interaction of the stakeholders, particularly between perspectives, adequate circumventions (motivation to comply, Req 5), procedures, and decision guidance. We can benefit in these areas from promising approaches in other areas of information-security research and social sciences – some of which we already touched on in this thesis:

- *Information-security decision-making of individuals:* Individuals can be considered to act rationally, weighing the perceived costs and benefits of alternatives before deciding – for example, to comply with the rules (e.g. not sharing the password) or rather work more productively. Research on the security economics and the motivation of compliance of the individual can be found in the Compliance Budget (Beautement et al., 2008) and similar approaches on the economic reasoning of individuals (Herley, 2010). Extending the view of individual rationality, we may also take the cognitive heuristics and biases into account that influence individuals' intuitive decisions as modeled by the Prospect Theory (Kahneman et al., 1982).

- *Sociology of behavior and decisions:* Social interactions influence the decisions and the actions of individuals – for example, when social rules are formed and learned through observation, as described by the Social Learning Theory (Bandura, 1986). These effects influence the individuals from the different perspectives on authorization and can be employed to arrive at more secure behavior.

- *Sociology of power relations:* Sociologists consider different layers of power that influence the interaction of individuals with socio-technical artifacts. For example, the Actor-Network Theory allows us to analyze how technical artifacts and humans interact. Theories on power, such as the Circuits of Power (Clegg, 1989), for instance, distinguish between the "episodic" power relations (when individuals actually experience restrictions, e.g. when access is denied), social norms that influence decisions, and the technology or formal rules that govern the situation (e.g. forbidding the sharing of passwords, cf. Inglesant and Sasse, 2011). This offers another promising approach to understand and influence individuals' behavior.

- *New institutional economics on information-security coordination and motivation:* We can apply economic theories, such as the Agency Theory, to study how security-related decisions are taken in organizations, how the coordination occurs, and how individuals are motivated (Pallas, 2009). A promising approach could be to analyze the actual communication flows within organizations with respect to behavior and decision-making for information security – and how conflicting goals interfere with secure behavior.

- *Organizational communication for authorization procedures:* Interaction and coordination occur in organizations either in formalized or informal relations. The work of Barnard (1938) on the utilitarian connections between actors could be the point of departure for further research on the actual information flows in organizations regarding authorization decisions. Organizational communication provides research approaches on formal and informal networks in organizations (cf. Monge and Contractor, 2000), enabling the extension of the work on INDUSE.

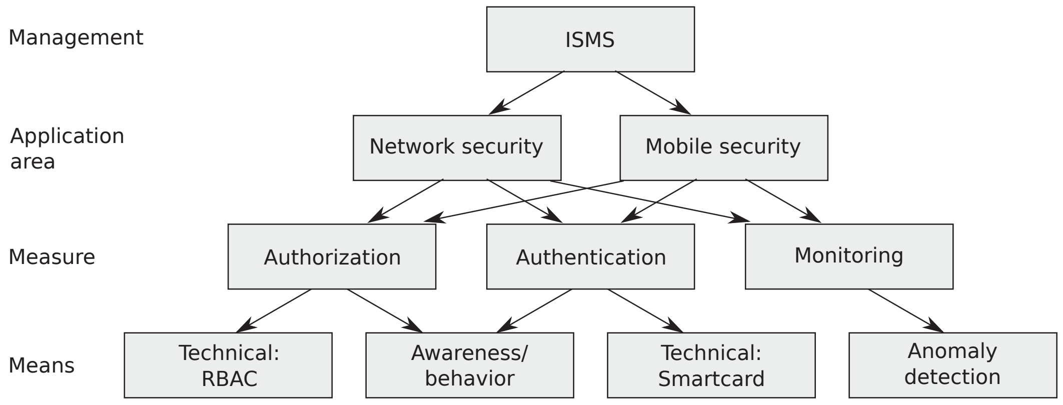


Figure 11.1.: Example of a hierarchy of information-security fields to be integrated

- *Insider-threat modeling:* Risk models are needed to guide information-security decisions. In case of authorization decisions, the risks particularly relate to the threat from malicious insiders who would abuse the granted permissions. Improved insider threat modeling, for example, based on criminological research, could improve the precision of decision guidance (Pfleeger and Stolfo, 2009).

- *Ethics in information security:* As discussed in the previous section, ethics are important on several levels for authorization measures and need further scrutiny: One area in policy management is the handling of (sensitive) personal data and the protection of the privacy of the involved individuals. For example, to support decisions on policy changes, we may need to analyze usage data or ask for sensitive information (cf. Chapter 10). Another area and more generally is applying psychological and sociological models and methods to influence individuals' behavior – and the dangers of paternalism and manipulation (cf. the previous section and the "disciplines" as discussed by Foucault, 1977).

11.4. Implications for information security research

11.4.1. Practical integration beyond authorization

On a practical level, we can extend the integrative approach in future research beyond authorization. Instead of limiting the integration of tools to authorization, we could integrate the management of the numerous interrelated aspects of information security. This should include the general information security management system (ISMS) as well as the management of specific areas (e.g. mobile security, network security), measures (authorization, security monitoring), and specific means to implement the measures (RBAC, anomaly detection). Roughly, we can consider these fields to relate to each other in a hierarchy (cf. Figure 11.1), with general security management at the top, above the management in specific areas, which, in turn, lies above the specific measures. There are overlaps between the activities of many of the fields, both on the same level (e.g. between measures, such as authentication and authorization) and between higher and lower layers. For example, decisions within specific measures, such as on the granting of access in authorization, are based on related factors as in other measures, for example, judging the appropriateness of activities in an application log as part of the monitoring measure. Similarly, these factors also need to be considered in higher-level security decisions, for instance, on whether to introduce a specific measure.

In a first step, the integration between fields requires common models of decision factors, such as the risks and contingencies, to enable the exchange between tools from different fields. Having established the integration between the fields, optimized processes would organize the collection of data at different levels of abstraction and their re-use between the fields to complement and validate the data. Further steps could include the coordinated decision-support between fields: decision factors collected at different levels would be employed to guide high-level decisions (e.g. on the introduction of specific measures), but would also result in guidance on the implementation of the concrete measure. The latter would then inform lower-level (technical), measure (design), and operational decision-support (permission granting). Analogously, monitoring of the effectiveness and the correct implementation could occur in detail at the specific measure, could propagate upwards to high-level analysis, and feed into improvement plans to complete the Plan–Do–Check–Act cycle (ISO/IEC 27001:2005, 2005).

11.4.2. A broader application of the principles

The practical integration of the perspectives is not the only way in which the theme of this dissertation can be extended to improve the effectiveness and efficiency of information security management. We can also transfer the Authorization Principles to other fields. In Table 11.4, we sketch analytically how a number of fields of information security relate to the hypotheses and principles from this dissertation. The considered fields include high-level (security management), area-specific (network security), and measure-specific (identification) management, and foundational fields (cryptography).

Taking a closer look at the hypotheses on interrelation of perspectives (H1, H5), we can identify perspectives for the fields that have a similar structure as the ones for authorization: Individuals that strive to complete their primary tasks and are affected by security (functional perspective), those organizing and deciding on measures (administrative), and those implementing and integrating security mechanisms in systems (development). Similarities can also be found for the usability problems between perspectives (H2), the multidisciplinary nature (H3), and dynamics (H4) of the measures. While the degree of impact on the measures will vary, the structural similarity warrants further examination and efforts to transfer the approaches from this thesis.

Considering the principles, the analysis in Table 11.4 further indicates that the principles apply in several of the selected fields. The applicability is most pronounced for the fields most dependent on management, less so for foundational fields, such as cryptography. While increasing awareness (P6) and reducing the individuals' burden (P2) are already broadly applied in several fields, others, such as the integration of perspectives (P1), increased concreteness (P3), and designing for dynamics (P7) are not widespread in any of the considered fields. It may thus be beneficial to further integrate perspectives as discussed in the following section. The second major approach is to further guide behavior of the different perspectives – implicitly by reducing the burden of adequate behavior (e.g. P2 *Reduced burden*) or explicitly by providing decision support (cf. decision-support prototype).

11.4.3. Fostering "informed participation"

The lack of integrative approaches may be one reason for the persisting attitude among information security practitioners that "users are the enemy" (Adams and Sasse, 1999), even though research has long shown this attitude to be counterproductive. The design of the measure – both of socio-organizational means and technical mechanisms – needs to fit the context (P8; Sasse, 2011) and thus the specific problems of the different perspectives. The first goal is to lessen the interference from security measures and improve the design process. As Fischer (1999) argued for cooperative design in the area of Computer-Supported Cooperative Work (CSCW), the integration of viewpoints not

	Overarching Security management	*Area* Software security	*Area* Network security	*Measure* Ident./authentication	*Foundations* Cryptography
H1 / H5 Perspectives					
Functional	Func. staff, administrator	Func. staff, administrator	Functional staff	Functional staff	Functional staff
Administrative	Policy maker, process designer	Policy maker, software architect	Network architects, policy makers, administrator	Administrator, policy maker	Administrator, policy maker
Development	Supporting tool developer	Developer	Component, tool developer	Application developer	Crypto designer, application developer
H2 Burden	Process comprehensibility, effectiveness, efficiency	Metrics comprehensibility, process problems	Configuration usability	Configuration usability, process effectiveness, efficiency	Implementation comprehensibility
H3 Multidisciplinary	Organizational, technical security	Software engineering, organizational, security	Network management, organizational, security	Organizational, security usability, software engineering	Cryptography, software engineering, security usability
H4 Dynamics	Organization, threats	Requirements, threats, mechanisms	Network topology, mechanisms	Users, mechanisms	Threats, use cases
P1 *Interaction and participation*	+ (func./admin. stakeholders)	+ (functional, security, developers)	o (staff, architects, policy makers…)	o (staff, policy maker)	o (dev.-centered design)
P2 *Reduced burden*	+ (continuous process)	+ (development effort)	+ (design tools)	**+ (mechanism usability)**	+ (for implementer, admin)
P3 *Concreteness*	+ (measure consequences)	o (mechanism implementation consequences)	o (design consequences)	o (configuration consequences)	o (crypt. design consequences)
P4 *Multidisciplinary*	o	o	o	**+ (cognitive psych., etc.)**	**o (math, security)**
P5 *Individuals' behavior*	o (compliance with policy)	+ (developer motivation)	+ (staff, admin behavior)	**+ (mechanism use)**	–
P6 *Security awareness*	**+ (motivation to comply)**	**+ (developer)**	+ (administrators)	**+ (staff: passwords risks)**	o (correct crypto use for devs.)
P7 *Design for dynamics*	+ (organization)	+ (requirements, threats)	+ (topology)	o (new users)	o (changes in uses, threats)
P8 *Tailor for context*	**+ (security needs, organization)**	+ (security needs)	o (security, performance needs)	+ (security needs, expertise)	o (security needs, performance)

Table 11.4.: Applying the hypotheses and principles to further areas of information security (+ strong, o to some extent; widespread adoption in bold)

only leads to conflicts, but also to "social creativity." Where knowledge cannot be held by one individual alone due to the complexity, it also cannot be expected to be simply "passed on", but instead needs to be rebuilt in collaborative knowledge construction: The process of "knowledge integration" overcomes the "symmetry of ignorance" of differing perspectives and "melds the information that is collaboratively constructed into the problem-solving context" (P1; Fischer, 1999).

Having understood the concrete problems of the different perspectives (P3), decision makers may be able to make transparent the rationale and why security-related effort cannot be further reduced (cf. Section 2.6; P5, P6). Increased integration of the diverse perspectives and, particularly, participation of functional staff could thus not only help to improve the measure, but also the change the mindset of the decision makers (P1). Even more important is the opportunity of increased democracy and empowerment through "informed participation". We observed this effect in the decision process for policy changes when study participants saw a chance to improve the communication of their benefit and increase the transparency of the decision. Even though information technologies "are not inherently democratic or empowering[10]", Brown et al. (1993) argue that informing and participating "brings democracy and technology into the same arena" – and may improve an organization's robustness to changes and disruptions[11].

Organizations need to recognize their primary responsibility for the effectiveness, efficiency, and satisfaction of security measures: This depends to a large extent on the organization's structure, interactions, and processes; adequate organizational decisions on technical and non-technical security measures are rather the result of optimal organization.

11.4.4. Integrative research: Interdisciplinarity

It is not only the mindset of practitioners that hamper effective information security. This thesis also argues that we need an integrative approach to the research as an addition to the focused traditional approach. While focused disciplinarian research is necessary to solve problems in detail and delivered respectable results, the growing fragmentation can be a limiting factor in achieving usable security measures in practice. Finkenthal (2001) argues that interdisciplinarity is necessary, since it "enables us to cope with complexity and prevents excessive divisions within our culture", and overcomes the "reductionist" approach of "disciplinarian thinking" (p. 9).

With regard to security usability, we can observe this kind of complexity in the actual implementation of security measures in practice. For example, we saw for decision support and policy changes how context-dependent the actual application of such technical means are. Influencing factors were, for instance, the formality and centralization of the organization, and the security expertise of the involved stakeholders. If we limit our scope to the technical perspective, we are likely to encounter (usability) problems from the complexity of combining the mechanism with organizational means into practical security measures. The "emergent properties" of multicomponent systems are also one of the core argument of Kline (1995) for interdisciplinarity. It helps us to understand the characteristics that combinations of components exhibit and that often differ from those of the individual components (p. 4). Conversely, interdisciplinarity is the "understanding [of] knowledge as a whole" (Finkenthal, 2001, p. 80). Similarly to the security practitioners' mindset, the mindset of researchers might also need to open to more integration, both to reduce fragmentation in the field and to integrate

[10] According to Brown et al. (1993), "deskilling technology in the workplace, for example, has shown that technology can at times be fundamentally disempowering."

[11] Organizational robustness is closely linked to the organizational culture and thus extends beyond information security or a superimposed "security culture" in an organization: Handling problems from information security interrelates with how an organization addresses problems from other areas – another argument for the integration of organizational and information security research.

the existing approaches from related disciplines (cf. Dhillon and Backhouse, 2001).

Even though interdisciplinarity was seen in social science as “old hat” by the late 1950s (Frank, 1988), we still find obstacles in the research communities when attempting to integrate disciplines. These may relate to the problems claimed to be caused by “weak” interdisciplinarity (Finkenthal, 2001, p. 10), that is, the mechanical application of research methods or models from foreign disciplines. The “nonsensical use of foreign concepts” can lead to “botched interdisciplinarity” (p. 87). Snow (1993) claims that there even is “a gulf of mutual incomprehension... hostility and dislike” (p. 4) between social and natural sciences. Two examples of problems surrounding the integration of research disciplines in security research can be found in the following quotes from anonymous reviews on a submission on the problems surrounding authorization in organizations (overlapping with Section 2.3.3) and the INDUSE paper (Bartsch, 2011c), both for mainline security conferences:

> *“The problems discussed in this paper, while of interest to ACSAC, do not beg technical resolution... While security is often termed a ‘human problem’, ACSAC’s mission [is] to pursue practical technical solutions, which effectively removes this work from further consideration.”*

> *“I am afraid that the purported contribution is rather tailored for the (enterprise) IS community than for a strictly security community, which makes the paper somehow inappropriate – i.e. out of scope – for the conference at stake”*

The reviews were for the Annual Computer Security Applications Conference (ACSAC) and for the Conference on Network and System Security (NSS), respectively. ACSAC rejected the submission, even though they call for “contributions in any aspect of applied security.” In contrast, NSS accepted the submitted paper, but the reluctance of the reviewer is revealing, since NSS calls for “research on all theoretical and practical aspects related to network and system security.” Even though only anecdotes, this viewpoint can be observed to be widespread in the information security research community: The “strictly security community” is only concerned with technical mechanisms, not the surrounding “soft” problems. Thimbleby (2004) argues for HCI that a general problem of interdisciplinarity is that researchers in new combinations of disciplines suffer from reviewers of the established disciplines who measure the merits of a paper or of a funding proposal by their respective discipline’s standards only, neglecting the added value of the combination of work, and thus reduce overall diversity by rejecting the idea. However, after 15 years of continuously growing security-usability research, one would expect that an integrative research approach would be a mainstay. In the end, information security research will need to further open up to the integration of research fields and disciplines to achieve more effective, efficient, and satisfactory security.

Bibliography

Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, September 1993. ISSN 0164-0925. doi: 10.1145/155183.155225.

Pekka Abrahamsson, Juhani Warsta, Mikko T. Siponen, and Jussi Ronkainen. New directions on agile methods: a comparative analysis. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 244–254, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1877-X.

Marc S. Ackerman and Scott D. Mainwaring. Privacy Issues and Human-Computer Interaction. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and usability: designing secure systems that people can use*. O'Reilly, 2005.

Alessandro Acquisti and Jens Grossklags. Privacy and rationality in individual decision making. *Security Privacy, IEEE*, 3(1):26–33, jan. 2005.

Silvia T. Acuña, Marta Gómez, and Natalia Juristo. How do personality, team processes and task characteristics relate to job satisfaction and software quality? *Information and Software Technology*, 51(3):627–639, 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.08.006.

Anne Adams and M. Angela Sasse. Users are not the enemy. *Commun. ACM*, 42:40–46, December 1999. ISSN 0001-0782. doi: 10.1145/322796.322806.

Anne Adams, M. Angela Sasse, and Peter Lunt. Making Passwords Secure and Usable. In *People and Computers XII, Proceedings of HCI '97*, pages 1–19. Springer, 1997.

Anne Adams, Peter Lunt, and Paul Cairns. A qualitative approach to HCI research. Cairns and Cox (2008).

John Adams. *Risk*. UCL Press, 1995.

John Adams. Cars, Cholera and Cows: the management of Risk and Uncertainty. *Policy Analysis*, (335), 1999.

Dakshi Agrawal, James Giles, Kang-Won Lee, and Jorge Lobo. Policy Ratification. In *POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 223–232, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2265-3.

Shane Ahern, Dean Eckles, Nathaniel S. Good, Simon King, Mor Naaman, and Rahul Nair. Over-exposed?: privacy patterns and considerations in online and mobile photo sharing. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 357–366, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240683.

Gail-Joon Ahn, Longhua Zhang, Dongwan Shin, and B. Chu. Authorization management for role-based collaboration. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4128–4134, Oct. 2003.

Ehab S. Al-Shaer and Hazem H. Hamed. Firewall Policy Advisor for anomaly discovery and rule editing. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, March 2003. doi: 10.1109/INM.2003.1194157.

Christopher Alberts, Audree Dorofee, James Stevens, and Carol Woody. *Introduction to the OCTAVE Approach*. CarnegieMellon, Pittsburgh, PA, USA, August 2003.

Eirik Albrechtsen. A qualitative study of users' view on information security. *Computers & Security*, 26(4):276–289, 2007. ISSN 0167-4048. doi: 10.1016/j.cose.2006.11.004.

Eirik Albrechtsen. *Friend or foe? Information security management of employees*. PhD thesis, NTNU, Trondheim, Norway, March 2008.

David Albright, Paul Brannan, and Christina Walrond. Did Stuxnet Take Out 1,000 Centrifuges at the Natanz Enrichment Plant? Technical report, Institute for Science and International Security, 2010. URL `http://isis-online.org/uploads/isis-reports/documents/stuxnet_FEP_22Dec2010.pdf`. Online, retrieved June 6, 2011.

Julia H. Allen, Sean Barnum, Robert J. Ellison, Gary McGraw, and Nancy R. Mead. *Software Security Engineering*. Addison-Wesley, Upper Saddle River, NJ, 2008.

Steven Alter and Susan A. Sherer. A General but Readily Adaptable Model of Information System Risk. *Communications of the AIS*, 14:1–28, 2004.

Massimo Ancona, Walter Cazzola, and Eduardo B. Fernández. Reflective Authorization Systems: Possibilities, Benefits, and Drawbacks. In Jan Vitek and Christian Damsgaard Jensen, editors, *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 1999. ISBN 3-540-66130-1.

Bob Anderson. Work, ethnography and system design. *The Encyclopedia of Microcomputers*, 20, 1997.

James P. Anderson. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Deputy for Command and Management Systems, L.G. Hanscom Field, Bedford, MA, October 1972.

Ross Anderson. *Security Engineering*. John Wiley & Sons, 2008.

Grigoris Antoniou, Matteo Baldoni, Piero A. Bonatti, Wolfgang Nejdl, and Daniel Olmedilla. Rule-Based Policy Specification. In T. Yu and S. Jajodia, editors, *Secure Data Management in Decentralized Systems*. Springer, 2007.

Apple Inc. Apple Human Interface Guidelines. Online, retrieved 15 Oct 2010, 2009. URL `http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGIntro/XHIGIntro.html`.

Jakob Arnoldi. *Risk*. Polity Press, Cambridge, 2009.

Farzaneh Asgharpour, Debin Liu, and L. Jean Camp. Mental Models of Computer Security Risks. In *WEIS '07: Workshop on the Economics of Information Security*, 2007.

Association of Certified Fraud Examiners (ACFE). Report to the Nation on Occupational Fraud & Abuse, 2006.

Audit Commission. *Opportunity Makes a Thief: an Analysis of Computer Abuse*. Audit Commission Publication, London, UK, 1994.

Jussi Auvinen, Rasmus Back, Jeanette Heidenberg, Piia Hirkman, and Luka Milovanov. Software Process Improvement with Agile Practices in a Large Telecom Company. In Jürgen Münch and Matias Vierimaa, editors, *PROFES*, volume 4034 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2006. ISBN 3-540-34682-1.

Emine G. Aydal, Richard F. Paige, Howard Chivers, and Phillip J. Brooke. Security Planning and Refactoring in Extreme Programming. In Pekka Abrahamsson, Michele Marchesi, and Giancarlo Succi, editors, *XP '06: 7th International Conference on Extreme Programming and Agile Processes in Software Engineering*, volume 4044 of *Lecture Notes in Computer Science*, pages 154–163. Springer, 2006. ISBN 3-540-35094-2.

Michelle Baddeley. Information Security: Lessons from Behavioural Economics. In *SHB '11: Security and Human Behavior Workshop*, 2011.

Nathan Baddoo, Tracy Hall, and Dorota Jagielska. Software developer motivation in a high maturity company: a case study. *Software Process: Improvement and Practice*, 11 (3):219–228, 2006.

Lee Badger. Providing a Flexible Security Override for Trusted Systems. In *CSFW*, pages 115–121, 1990.

Bouchaib Bahli and El Sayed Abou Zeid. The role of knowledge creation in adopting extreme programming model: an empirical study. In *Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on*, pages 75–87, 5-6 2005. doi: 10.1109/ITICT.2005.1609616.

Albert Bandura. *Social Foundations of Thought and Action*. Prentice Hall, Englewood Cliffs, NJ, 1986.

Alberta Bandura. *Social Learning Theory*. Prentice Hall, Englewood Cliffs, NJ, 1977.

Chester I. Barnard. *The Functions of the Executive*. Harvard University Press, Cambridge, MA, USA, 1938.

Steffen Bartsch. Supporting Authorization Policy Modification in Agile Development of Web Applications. In *SecSE '10: Fourth International Workshop on Secure Software Engineering*. IEEE Computer Society, 2010a. doi: 10.1109/ARES.2010.19.

Steffen Bartsch. A calculus for the qualitative risk assessment of policy override authorization. In *SIN '10: Proceedings of the 3rd international conference on Security of information and networks*, pages 62–70, New York, NY, USA, 2010b. ACM. ISBN 978-1-4503-0234-0. doi: 10.1145/1854099.1854115.

Steffen Bartsch. An Authorization Enforcement Usability Case Study. In *ESSoS 2011*. Springer, 2011a. doi: 10.1007/978-3-642-19125-1_16.

Steffen Bartsch. Practitioners' Perspectives on Security in Agile Development. In *FARES '11: 6th International Workshop on Frontiers in Availability, Reliability and Security*. IEEE Computer Society, 2011b. doi: 10.1109/ARES.2011.82.

Steffen Bartsch. Exploring Twisted Paths: Analyzing Authorization Processes in Organizations. In *NSS '11: Proceedings of the 5th International Conference on Network and System Security*. IEEE Computer Society, 2011c. doi: 10.1109/ICNSS.2011.6060003.

Steffen Bartsch. Policy Override in Practice: Model, Evaluation, and Decision Support. *Security and Communication Networks*, 2012. doi: 10.1002/sec.547. In print.

Steffen Bartsch and Carsten Bormann. Berechtigungsmodellierung im Geschäftsprozessmanagement von KMU. In Patrick Horster, editor, *D.A.CH. Security 2008*. syssec Verlag, June 2008.

Steffen Bartsch and M. Angela Sasse. Guiding Decisions on Authorization Policies: A Participatory Approach to Decision Support. In *SAC '12: 27th ACM Symposium On Applied Computing*. ACM, 2012.

Steffen Bartsch, Karsten Sohr, and Carsten Bormann. Supporting Agile Development of Authorization Rules for SME Applications. In *TrustCol '08: 3rd International Workshop on Trusted Collaboration*, pages 461–471, Berlin Heidelberg, 2009. Springer. doi: 10.1007/978-3-642-03354-4_35.

Richard Baskerville. Information systems security design methods: implications for information systems development. *ACM Comput. Surv.*, 25(4):375–414, 1993. ISSN 0360-0300. doi: 10.1145/162124.162127.

Richard Baskerville and Mikko T. Siponen. An information security meta-policy for emergent organizations. *Logistics Information Management*, 15(5/6), 2002.

Lujo Bauer, Lorrie Faith Cranor, Robert W. Reeder, Michael K. Reiter, and Kami Vaniea. A user study of policy creation in a flexible access-control system. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 543–552, New York, NY, USA, 2008a. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357143.

Lujo Bauer, Scott Garriss, and Michael K. Reiter. Detecting and resolving policy misconfigurations in access-control systems. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 185–194, New York, NY, USA, 2008b. ACM. ISBN 978-1-60558-129-3. doi: 10.1145/1377836.1377866.

Lujo Bauer, Lorrie Faith Cranor, Robert W. Reeder, Michael K. Reiter, and Kami Vaniea. Real life challenges in access-control management. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 899–908, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518838.

BBC News. Siprnet: Where the leaked cables came from. Online, retrieved 10 Jul 2011, 2010. URL `http://www.bbc.co.uk/news/world-us-canada-11863618`.

Adam Beautement, M. Angela Sasse, and Mike Wonham. The Compliance Budget: Managing Security Behaviour in Organisations. In *NSPW '08: Proceedings of the 2008 Workshop on New Security Paradigms*, pages 47–58. ACM, 2008. doi: 10.1145/1595676.1595684.

Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, first edition, 1999. ISBN 0201616416.

Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Boston, 2nd ed. edition, 2004. ISBN 0321278658.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development. Online, retrieved 10 Jul 2010, 2001. URL `http://agilemanifesto.org`.

Moritz Y. Becker and Sebastian Nanz. The Role of Abduction in Declarative Authorization Policies. In Paul Hudak and David Scott Warren, editors, *Symposium on Practical Aspects of Declarative Languages (PADL 2008)*, volume 4902 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2008. ISBN 978-3-540-77441-9.

Gregory Bedny and David Meister. Theory of Activity and Situation Awareness. *International Journal of Cognitive Ergonomics*, 1(3):63–72, 1999.

Sarah Beecham, Helen Sharp, Nathan Baddoo, Tracy Hall, and Hugh Robinson. Does the XP environment meet the motivational needs of the software developer? An empirical study. In *AGILE 2007*, 2007. doi: 10.1109/AGILE.2007.22.

Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. Motivation in Software Engineering: A systematic literature review. *Information and Software Technology*, 50(9-10):860–878, 2008. ISSN 0950-5849. doi: 10.1016/j.infsof.2007.09.004.

Andrew Begel and Nachiappan Nagappan. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 255–264, 20-21 2007. doi: 10.1109/ESEM.2007.12.

Messaoud Benantar. *Access Control Systems: Security, Identity Management and Trust Models*. Springer US, New York, NY, USA, 2006.

Yolanta Beresnevichiene, David Pym, and Simon Shiu. Decision support for systems security investment. In *Network Operations and Management Symposium Workshops (NOMS Wksps)*, pages 118–125, April 2010. doi: 10.1109/NOMSW.2010.5486590.

Scott Berinato. Finally, a Real Return on Security Spending. *CIO Magazine (Australia)*, 2002. URL `http://www.cio.com.au/index.php/id`.

Elisa Bertino, Elena Ferrari, and Vijay Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999. ISSN 1094-9224. doi: 10.1145/300830.300837.

Elisa Bertino, Seraphin Calo, Hong Chen, Ninghui Li, Tiancheng Li, Jorge Lobo, Ian Molloy, and Qihua Wang. Some Usability Considerations in Access Control Systems. In *USM '08: Workshop on Usable IT Security Management*, 2008.

Denis Besnard and Budi Arief. Computer security impaired by legitimate users. *Computers & Security*, 23(3):253–264, 2004. ISSN 0167-4048. doi: 10.1016/j.cose.2003.09.002.

Hugh Beyer and Karen Holtzblatt. *Contextual design – Defining Customer-Centered Systems*. Morgan Kaufmann, San Francisco, CA, USA, 1998.

Konstantin Beznosov and Philippe Kruchten. Towards agile security assurance. In *NSPW '04: Proceedings of the 2004 workshop on New security paradigms*, pages 47–54, New York, NY, USA, 2004. ACM.

Konstantin Beznosov, Yi Deng, Bob Blakley, and John Barkley. A Resource Access Decision Service for CORBA-Based Distributed Systems. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, page 310, Los Alamitos, CA, USA, 1999. IEEE Computer Society. doi: 10.1109/CSAC.1999.816041.

Matt Bishop. Psychological Acceptability Revisited. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and usability: designing secure systems that people can use*. O'Reilly, 2005.

Alan F. Blackwell, Luke Church, and Thomas Green. The Abstract is 'an Enemy': Alternative Perspectives to Computational Thinking. *PPIG*, 2008.

Bob Blakley. The Emperor's old armor. In *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*, pages 2–16, New York, NY, USA, 1996. ACM. ISBN 0-89791-944-0.

Barry W. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5):61–72, 1988.

Barry W. Boehm, Terence E. Gray, and Thomas Seewaldt. Prototyping vs. specifying: A multi-project experiment. In *ICSE '84: Proceedings of the 7th international conference on Software engineering*, pages 473–484, Piscataway, NJ, USA, 1984. IEEE Press. ISBN 0-8186-0528-6.

P. Bonatti, D. Olmedilla, and J. Peer. Advanced Policy Explanations on the Web. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 200–204, 2006.

Dan Borge. *The Book of Risk*. Wiley, 2001.

Gustav Boström, Jaana Wäyrynen, Marine Bodén, Konstantin Beznosov, and Philippe Kruchten. Extending XP practices to support security requirements engineering. In *SESS '06: Proceedings of the 2006 international workshop on Software engineering for secure systems*, pages 11–18, New York, NY, USA, 2006. ACM.

Christina Braz, Ahmed Seffah, and David M'Raihi. Designing a Trade-Off Between Usability and Security: A Metrics Based-Model. In Maria Cecilia Calani Baranauskas, Philippe A. Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa, editors, *INTERACT (2)*, volume 4663 of *Lecture Notes in Computer Science*, pages 114–126. Springer, 2007. ISBN 978-3-540-74799-4.

Philip Brey. The ethics of representation and action in virtual reality. *Ethics and Information Technology*, 1:5–14, 1999.

David W. Britton and Ian A. Brown. A Security Risk Measurement for the RAdAC Model. Master's thesis, Naval Postgraduate School Monterey, CA, March 2007.

Sacha Brostoff, Martina Angela Sasse, David W. Chadwick, James Cunningham, Uche M. Mbanaso, and Sassa Otenko. 'R-What?' Development of a role-based access control policy-writing tool for e-Scientists. *Softw., Pract. Exper.*, 35(9):835–856, 2005. doi: 10.1002/spe.691.

John Seely Brown, Paul Duguid, and Susan Haviland. Towards informed participation: Six scenarios in search of democracy in the electronic age. In *The Promise and Perils of Emerging Information Technologies*. The Aspen Institute, Queenstown, Maryland, USA, 1993.

Achim D. Brucker and Helmut Petritsch. Extending access control models with break-glass. In *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 197–206, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-537-6.

Achim D. Brucker, Helmut Petritsch, and Andreas Schaad. Delegation Assistance. In *POLICY '09: Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 84–91, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3742-9. doi: 10.1109/POLICY.2009.35.

Alan Bryman. *Quantity and quality in social research*. Unwin Hyman, London, UK, 1988.

Bundesamt für Sicherheit in der Informationstechnik (BSI). BSI-Standard 100-2: IT-Grundschutz-Vorgehensweise. Version 2.0, 2008.

Paul Cairns and Anna L. Cox. *Research methods for human-computer interaction*. Cambridge Univ. Press, Cambridge, 2008.

L. Jean Camp. Mental models of privacy and security. *IEEE Technology and Society Magazine*, 28(3), Fall 2009.

Philip L. Campbell and Jason E. Stamp. A Classification Scheme for Risk Assessment Methods. Technical Report SAND2004-4233, Sandia National Laboratories, 2004.

William Cannell and Harry Otway. Audience Perspectives in the Communication of Technological Risks. *Futures*, Oct 1988.

Lan Cao and Balasubramaniam Ramesh. Agile Requirements Engineering Practices: An Empirical Study. *Software, IEEE*, 25(1):60–67, jan.-feb. 2008. ISSN 0740-7459. doi: 10.1109/MS.2008.1.

Xiang Cao and Lee Iverson. Intentional access management: making access control usable for end-users. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 20–31. ACM, 2006.

Dawn Cappelli, Andrew Moore, Randall F. Trzeciak, and Timothy J. Shimeall. Common Sense Guide to Prevention and Detection of Insider Threats 3rd Edition – Version 3.1. Technical report, CarnegieMellon, January 2009.

Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983.

John M. Carroll, editor. *HCI models, theories, and frameworks: toward a multidisciplinary science*. Kaufmann, 2003.

Manuel Castells. *The rise of the network society*. The information age: economy, society and culture. Blackwell, Oxford, 2. ed edition, 2000. ISBN 0631221409.

Jan G. Cederquist, Ricardo Corin, Marnix A. C. Dekker, Sandro Etalle, Jerry I. den Hartog, and Gabriele Lenzini. Audit-based compliance control. *Int. J. Inf. Sec.*, 6(2-3): 133–151, 2007.

David Chadwick and M. Angela Sasse. The Virtuous Circle of Expressing Authorization Policies. In *SWPW '06: Proceedings of Second Semantic Web Policy Workshop*, 2006.

Pau-Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A. Karger, Grant M. Wagner, and Angela Schuett Reninger. Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control. In *S&P '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 222–230, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2848-1.

Brian Chess and Jacob West. *Secure Programming with Static Analysis*. Pearson Education, Boston, 2007.

Ram Chillarege, Inderpal S. Bhandari, Jarir K. Chaar, Michael J. Halliday, Diane S. Moebus, Bonnie K. Ray, and Man-Yuen Wong. Orthogonal defect classification-a concept for in-process measurements. *IEEE Transactions on Software Engineering*, 18(11):943–956, nov 1992. doi: 10.1109/32.177364.

Ramkumar Chinchani, Anusha Iyer, Bharat Jayaraman, and Shambhu J. Upadhyaya. Insecure Programming: How Culpable is a Language's Syntax? In *IEEE Systems, Man and Cybernetics Society Information Assurance Workshop*, pages 158–163, 2003.

Howard Chivers, Richard F. Paige, and Xiaocheng Ge. Agile Security Using an Incremental Security Architecture. In Hubert Baumeister, Michele Marchesi, and Mike Holcombe, editors, *XP '05: 6th International Conference on Extreme Programming and Agile Processes in Software Engineering*, volume 3556 of *Lecture Notes in Computer Science*, pages 57–65. Springer, 2005. ISBN 3-540-26277-6.

Noam Chomsky and Michel Foucault. *The Chomsky-Foucault debate: On human nature*. The New Press, New York, NY, USA, 2006.

Rahim Choudhary. A policy based architecture for NSA RAdAC model. In *IAW '05: Proceedings of the IEEE Information Assurance Workshop*, pages 294–301, June 2005.

Lawrence Chung and Julio Cesar Sampaio do Prado Leite. On Non-Functional Requirements in Software Engineering. In Alexander Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. K. Yu, editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 363–379. Springer, 2009. ISBN 978-3-642-02462-7.

Lawrence Chung and Brian A. Nixon. Dealing with non-functional requirements: three experimental studies of a process-oriented approach. In *ICSE '95: Proceedings of the 17th international conference on Software engineering*, pages 25–37, New York, NY, USA, 1995. ACM. ISBN 0-89791-708-1. doi: 10.1145/225014.225017.

Luke Church. End User Security: The democratisation of security usability. In *SHB '08: Security and Human Behavior Workshop*, 2008.

Luke Church, Matthew Kreeger, and Marcus Streets. Introducing Usability to the Common Criteria. In *ICCC 2008*, 2008.

Steven Clarke. Measuring API Usability. *Dr. Dobb's Journal*, May 2004.

Stewart Clegg, Martin Kornberger, and Carl Rhodes. Organizational ethics, decision making, undecidability. *The Sociological Review*, 55(2), 2007.

Stewart R. Clegg. *Frameworks of power*. SAGE, London, UK, 1989.

Alistair Cockburn. *Agile Software Development*. Addison-Wesley Professional, December 2001. ISBN 0201699699.

Alistair Cockburn. *Crystal clear a human-powered methodology for small teams*. Addison-Wesley Professional, 2004. ISBN 0201699478.

Cynthia F. Cohen, Stanley J. Birkin, Monica J. Garfield, and Harold W. Webb. Managing conflict in software testing. *Commun. ACM*, 47(1):76–81, 2004a.

David Cohen, Mikael Lindvall, and Patricia Costa. An introduction to agile methods. *Advances in Computers*, 62: 2–67, 2004b.

Michael D. Cohen, James G. March, and Johan P. Olsen. A Garbage Can Model of Organizational Choice. *Administrative Science Quarterly*, 17(1):pp. 1–25, 1972. ISSN 00018392.

Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly, Sebastopol, CA, USA, 2004.

Giulio Concas, Marco Di Francesco, Michele Marchesi, Roberta Quaresima, and Sandro Pinna. An Agile Development Process and Its Assessment Using Quantitative Object-Oriented Metrics. In Pekka Abrahamsson, Richard Baskerville, Kieran Conboy, Brian Fitzgerald, Lorraine Morgan, and Xiaofeng Wang, editors, *XP '08: 9th International Conference on Extreme Programming and Agile Processes in Software Engineering*, volume 9 of *Lecture Notes in Business Information Processing*, pages 83–93. Springer, 2008. ISBN 978-3-540-68254-7.

Charles Consel and Renaud Marlet. Architecture Software Using: A Methodology for Language Development. In Catuscia Palamidessi, Hugh Glaser, and Karl Meinke, editors, *PLILP/ALP*, volume 1490 of *Lecture Notes in Computer Science*, pages 170–194. Springer, 1998. ISBN 3-540-65012-1.

Alan Cooper. *The inmates are running the asylum*. SAMS, Indianapolis, Ind., 1999. ISBN 0672316498.

Alan Cooper, Robert Reimann, and Dave Cronin. *About face 3: The essentials of interaction design*. Wiley, 2007.

Robert D. Cooter. The Intrinsic Value of Obeying a Law: Economic Analysis of the Internal Viewpoint. *Fordham Law Review*, 75, 2006.

Lorrie Faith Cranor. A Framework for Reasoning About the Human in the Loop. In *UPSEC 08*, Pittsburgh, Pennsylvania, 2008.

Robert Crook, Darrel Ince, and Bashar Nuseibeh. Modelling access policies using roles in requirements engineering. *Information and Software Technology*, 45 (14):979–991, 2003. ISSN 0950-5849. doi: 10.1016/S0950-5849(03)00097-1. Eighth International Workshop on Requirements Engineering: Foundation for Software Quality.

Bill Curtis, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems. *Commun. ACM*, 31(11):1268–1287, 1988. ISSN 0001-0782. doi: 10.1145/50087.50089.

Aldo Dagnino, Karen Smiley, Hema Srikanth, Annie I. Antón, and Laurie A. Williams. Experiences in applying agile software development practices in new product development. In *IASTED Conf. on Software Engineering and Applications*, pages 501–506. ACTA Press, 2004.

Jie Dai and Jim Alves-Foss. Logic Based Authorization Policy Engineering. In *The 6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

Noopur Davis. Secure Software Development Life Cycle Processes: A Technology Scouting Report. Technical Report CMU/SEI-2005-TN-024, CarnegieMellon, December 2005.

Noopur Davis and Julia L. Mullaney. The Team Software Process (TSP) in Practice: A Summary of Recent Results. Technical Report CMU/SEI-2003-TR-014, CarnegieMellon, September 2003.

Noopur Davis, Watts Humphrey, Samuael T. Jr. Redwine, Gerlinde Zibulski, and Gary McGraw. Processes for producing secure software. *IEEE Security Privacy*, 2 (3):18–25, May-June 2004. ISSN 1540-7993. doi: 10.1109/MSP.2004.21.

Bruna De Marchi. The Seveso Directive: An Italian Pilot Study in Enabling Communication. *Risk Analysis*, 11 (2), 1991.

Bart De Win, Frank Piessens, Wouter Joosen, and Tine Verhanneman. On the importance of the separation-of-concerns principle in secure software engineering. In *ACSA Workshop on the Application of Engineering Principles to System Security Design*, 2003.

Bart De Win, Riccardo Scandariato, Koen Buyens, Johan Grégoire, and Wouter Joosen. On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, 51(7):1152–1171, 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.01.010.

Concettina Del Grosso, Giuliano Antoniol, Massimiliano Di Penta, Philippe Galinier, and Ettore Merlo. Improving network applications security: a new heuristic to generate stress testing data. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1037–1043, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8. doi: 10.1145/1068009.1068185.

Ian Denley and Simon Weston Smith. Privacy in clinical information systems in secondary care. *BMJ*, 318(7194): 1328–31, May 1999. ISSN 0959-8138.

Department of Defense. *Trusted Computer Systems Evaluation Criteria*. Department of Defense Standard. DoD, 1985. 5200.28-STD.

Rene Descartes. *Discourse on the Method of Rightly Conducting the Reason and Seeking Truth in the Sciences*. Sutherland and Knox, Edinburgh, 1850.

Jon DeVaan. Update on UAC. Online, retrieved 15 Oct 2010, 2009. URL `http://blogs.msdn.com/b/e7/archive/2009/02/05/update-on-uac.aspx`.

Premkumar T. Devanbu and Stuart Stubblebine. Software engineering for security: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 227–239, New York, NY, USA, 2000. ACM. ISBN 1-58113-253-0. doi: 10.1145/336512.336559.

Alexander J. DeWitt and Jasna Kuljis. Aligning usability and security: A usability study of Polaris. In Lorrie Faith Cranor, editor, *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 1–7. ACM, 2006. ISBN 1-59593-448-0. doi: 10.1145/1143120.1143122.

Gurpreet Dhillon and James Backhouse. Current directions in IS security research: towards socio-organizational perspectives. *Information Systems Journal*, 11:127–153, 2001.

Nguyen Ngoc Diep, Le Xuan Hung, Yonil Zhung, Sungyoung Lee, Young-Koo Lee, and Heejo Lee. Enforcing Access Control Using Risk Assessment. In *Universal Multiservice Networks, 2007. ECUMN '07. Fourth European Conference on*, pages 419–424, feb. 2007. doi: 10.1109/ECUMN.2007.19.

Nathan Dimmock, András Belokosztolszki, David Eyers, Jean Bacon, and Ken Moody. Using trust and risk in role-based access control policies. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 156–162, New York, NY, USA, 2004. ACM. ISBN 1-58113-872-5. doi: 10.1145/990036.990062.

Torgeir Dingsøyr, Tore Dybå, and Pekka Abrahamsson. A Preliminary Roadmap for Empirical Research on Agile Software Development. In *AGILE 2008*, pages 83–94, 4-8 2008. doi: 10.1109/Agile.2008.50.

Alan John Dix. Theoretical analysis and theory creation. Cairns and Cox (2008).

Alan John Dix, Janet Finlay, Gregory D. Abowd, and Russell Beale. *Human-computer interaction*. Pearson Prentice-Hall, Harlow [u.a.], 3. ed edition, 2004. ISBN 0130461091.

Mattei Dogan and Robert Pahre. Fragmentation and recombination of the social sciences. *Studies in Comparative International Development*, 24(2), 1989.

Mahi Dontamsetti and Anup Narayanan. Impact of the Human Element on Information Security. In Gupta and Sharman (2009).

Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Specifying and Reasoning About Dynamic Access-Control Policies. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 632–646. Springer, 2006. ISBN 3-540-37187-7.

Yael Dubinsky and Orit Hazzan. Using a role scheme to derive software project metrics. *Journal of Systems Architecture*, 52(11):693–699, 2006. doi: 10.1016/j.sysarc.2006.06.013.

Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859, 2008. ISSN 0950-5849.

Tore Dybå and Torgeir Dingsøyr. What Do We Know about Agile Software Development? *IEEE Softw.*, 26(5):6–9, 2009. ISSN 0740-7459. doi: 10.1109/MS.2009.145.

Nina Dzamashvili-Fogelström, Tony Gorschek, Mikael Svahnberg, and Peo Olsson. The impact of agile principles on market-driven software product development. *Journal of Software Maintenance*, 22(1):53–80, 2010.

Albert Endres. An analysis of errors and their causes in system programs. In *Proceedings of the international conference on Reliable software*, pages 327–336, New York, NY, USA, 1975. ACM. doi: 10.1145/800027.808455.

Mica R. Endsley. Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, March 1995. doi: 10.1518/001872095779049543.

Gencer Erdogan, Per Håkon Meland, and Derek Mathieson. Security Testing in Agile Web Application Development - A Case Study Using the EAST Methodology. In Will Aalst et al., editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 14–27. Springer, Berlin Heidelberg, 2010. ISBN 978-3-642-13054-0.

K. Anders Ericsson and Herbert A. Simon. *Protocol analysis: verbal reports as data*. MIT Press, Cambridge, MA, USA, 1993.

Michael E. Fagan. Advances in software inspections. *IEEE Trans. Softw. Eng.*, 12(7):744–751, 1986. ISSN 0098-5589.

Shamal Faily and Ivan Flechais. Persona Cases: A Technique for grounding Personas. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, Vancouver, BC, Canada, 2011. ACM.

David Ferraiolo and Richard Kuhn. Role-Based Access Controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

Ana Ferreira, David Chadwick, Pedro Farinha, Ricardo Correia, Gansen Zao, Rui Chilro, and Luis Antunes. How to securely break into RBAC: the BTG-RBAC model. In *ACSAC '09: Proceedings of the Annual Computer Security Applications Conference*, pages 23–31, 2009. doi: 10.1109/ACSAC.2009.12.

Michael Finkenthal. *Interdisciplinarity: Toward the Definition of a Metadiscipline?* Peter Lang, New York, NY, USA, 2001.

Donald G. Firesmith. Engineering Security Requirements. *Journal of Object Technology*, 2(1), 2003.

Gerhard Fischer. Symmetry of igorance, social creativity, and meta-design. In *C&C '99: Proceedings of the 3rd conference on Creativity & cognition*, pages 116–123, New York, NY, USA, 1999. ACM. ISBN 1-58113-078-3. doi: 10.1145/317561.317582.

Kathi Fisler, Shriram Krishnamurthi, Leo A. Meyerovich, and Michael Carl Tschantz. Verification and change-impact analysis of access-control policies. In Gruia-Catalin Roman, William G. Griswold, and Bashar Nuseibeh, editors, *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 196–205. ACM, 2005. doi: 10.1145/1062455.1062502.

Brian Fitzgerald, Gerard Hartnett, and Kieran Conboy. Customising agile methods to software practices at Intel Shannon. *Eur. J. Inf. Syst.*, 15(2):200–213, 2006. ISSN 0960-085X. doi: 10.1057/palgrave.ejis.3000605.

David Flanagan and Yukihiro Matsumoto. *The Ruby Programming Language*. O'Reilly, Sebastopol, CA, USA, 2008.

Ivan Flechais and M. Angela Sasse. Stakeholder Involvement, Motivation, Responsibility, Communication: How to Design Usable Security in e-Science. *International Journal of Human-Computer Studies*, 67(4), 2009.

Ivan Flechais, M. Angela Sasse, and Stephen Hailes. Bringing security home: a process for developing secure and usable systems. In Christian Hempelmann and Victor Raskin, editors, *NSPW '03: Proceedings of the 2003 workshop on New security paradigms*, pages 49–57. ACM, 2003. ISBN 1-58113-880-6.

Ivan Flechais, Cecilia Mascolo, and M. Angela Sasse. Integrating security and usability into the requirements and design process. *Int. J. Electron. Secur. Digit. Forensic*, 1(1):12–26, 2007. ISSN 1751-911X. doi: 10.1504/IJESDF.2007.013589.

Bent Flyvbjerg. Five Misunderstandings about Case-Study Research. In Seale et al. (2007).

B.J. Fogg. *Persuasive technology: using computers to change what we think and do*. Morgan Kaufmann, San Francisco, CA, USA, 2003.

Michel Foucault. *Discipline and punish: the birth of the prison*. Knopf Doubleday Publishing Group, 1977.

Roberta Frank. "Interdisciplinarity": The First Half Century. In E.G. Stanley and T.F. Hoad, editors, *Words*. D.S. Brewer, Woodbridge, Suffolk, UK, 1988.

B. Fraser. Site Security Handbook. RFC 2196 (Informational), September 1997. URL `http://www.ietf.org/rfc/rfc2196.txt`.

Steven Furnell and Stamatis Bolakis. Helping us to help ourselves: Assessing administrators' use of security analysis tools. *Network Security*, 2004(2):7–12, 2004. ISSN 1353-4858. doi: 10.1016/S1353-4858(04)00035-2.

Steven Furnell, Adila Jusoh, and Dimitris Katsabas. The challenges of understanding and using security: A survey of end-users. *Computers & Security*, 25(1):27–35, 2006.

Michael P. Gallaher, Alan C. O'Connor, and Brian Kropp. The Economic Impact of Role-Based Access Control. Planning Report 02-1 for the NIST, March 2002.

Vinod Ganapathy, Trent Jaeger, and Somesh Jha. Retrofitting Legacy Code for Authorization Policy Enforcement. In *S&P '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 214–229, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

Simson Garfinkel. *Design principles and patterns for computer systems that are simultaneously secure and usable*. PhD thesis, Massachusetts Institute of Technology, 2005.

Xiaocheng Ge, Richard F. Paige, Fiona Polack, and Phillip J. Brooke. Extreme Programming Security Practices. In Giulio Concas, Ernesto Damiani, Marco Scotto, and Giancarlo Succi, editors, *XP '07: 8th International Conference on Extreme Programming and Agile Processes in Software Engineering*, volume 4536 of *Lecture Notes in Computer Science*, pages 226–230. Springer, 2007. ISBN 978-3-540-73100-9.

Weiwei Geng, Scott Flinn, and John M. DeDourek. Usable Firewall Configuration. In *PST '05: Third Annual Conference on Privacy, Security and Trust*, 2005.

Boby George and Laurie A. Williams. An Initial Investigation of Test Driven Development in Industry. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 1135–1139. ACM, 2003.

Boby George and Laurie A. Williams. A structured experiment of test-driven development. *Information & Software Technology*, 46(5):337–342, 2004.

Thomas Gilovich, Dale W. Griffin, and Daniel Kahneman, editors. *Heuristics and biases: the psychology of intuitive judgement*. Cambridge University Press, Cambridge, 2002.

Barney G. Glaser and Anselm L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 1967.

Karen Mercedes Goertzel, Theodore Winograd, and Patrick Holley. *Security in the Software Lifecycle: Making Software Development Processes–and the Software Produced by Them–More Secure*. DHS, August 2006. Draft Version 1.2.

Karen Mercedes Goertzel, Theodore Winograd, Holly Lynne McKinley, Lyndon Oh, Michael Colon, Thomas McGibbon, Elaine Fedchak, and Robert Vienneau. Software security assurance: A state-of-art report. Technical report, Information Assurance Technology Analysis Center (IATAC), 2007.

Karen Mercedes Goertzel, Theodore Winograd, Holly Lynne McKinley, and Patrick Holley. Enhancing the Development Life Cycle to Produce Secure Software: A Reference Guidebook on Software Assurance. online, 2008. URL `https://www.thedacs.com/techs/enhanced_life_cycles`.

Mikhail I. Gofman, Ruiqi Luo, Ayla C. Solomon, Yingbin Zhang, Ping Yang, and Scott D. Stoller. RBAC-PAT: A Policy Analysis Tool for Role Based Access Control. In *TACAS '09: 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2009.

Joseph A. Goguen and José Meseguer. Security Policies and Security Models. In *S&P '82: Proceedings of the 1982 IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA, 1982. IEEE Computer Society. doi: 10.1109/SP.1982.10014.

Dieter Gollmann. *Computer Security*. Wiley, West Sussex, UK, 3. ed edition, 2011.

Li Gong and Gary Ellison. *Inside Java(TM) 2 Platform Security: Architecture, API Design, and Implementation*. Pearson Education, 2003. ISBN 0201787911.

G. Anthony Gorry and Michael S. Scott Morton. A Framework for Management Information Systems. *Sloan Management Review*, 1989.

Robert M. Graham. Protection in an information processing utility. *Commun. ACM*, 11:365–369, May 1968. ISSN 0001-0782. doi: 10.1145/363095.363146.

Saul Greenberg and Bill Buxton. Usability evaluation considered harmful (some of the time). In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 111–120, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357074.

Sergio B. Guarro. Principles and procedures of the LRAM approach to information systems risk anaylsis and management. *Comput. Secur.*, 6:493–504, December 1987. ISSN 0167-4048. doi: 10.1016/0167-4048(87)90030-7.

Manish Gupta and Raj Sharman, editors. *Social and Human Elements of Information Security: Emerging Trends and Countermeasures*. Information Science Reference, Hershey, 2009.

Nurit Guttman. Beyond Strategic Research: A Value-Centered Approach to Health Communication Intervention. *Communication Theory*, 7(2), May 1997.

Charles B. Haley, Robin C. Laney, and Bashar Nuseibeh. Deriving security requirements from crosscutting threat descriptions. In *AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development*, pages 112–121, New York, NY, USA, 2004. ACM. ISBN 1-58113-842-3. doi: 10.1145/976270.976285.

Anthony Hall and Roderick Chapman. Correctness by construction: developing a commercial secure system. *Software, IEEE*, 19(1):18–25, jan/feb 2002. ISSN 0740-7459. doi: 10.1109/52.976937.

Robert T. Hall. *The Morality of Civil Disobedience*. Harper & Row, New York, NY, USA, 1971.

D. Hamlet and J. Maybee. *The Engineering of Software: Technical Foundations for the Individual*. Addison-Wesley, 2001.

Weili Han, Qun Ni, and Hong Chen. Apply Measurable Risk to Strengthen Security of a Role-Based Delegation Supporting Workflow System. In *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY 2009)*, pages 45–52, july 2009. doi: 10.1109/POLICY.2009.26.

Wilfred J. Hansen. User engineering principles for interactive systems. In *AFIPS '71 (Fall): Proceedings of the November 16-18, 1971, fall joint computer conference*, pages 523–532, New York, NY, USA, 1971. ACM. doi: 10.1145/1479064.1479159.

Geir Kjetil Hanssen and Tor Erlend Fægri. Agile customer engagement: a longitudinal qualitative case study. In *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 164–173, New York, NY, USA, 2006. ACM. ISBN 1-59593-218-6. doi: 10.1145/1159733.1159759.

Jefferson B. Hardee, Ryan West, and Christopher B. Mayhorn. To download or not to download: an examination of computer security decision making. *interactions*, 13(3):32–37, 2006. ISSN 1072-5520. doi: 10.1145/1125864.1125887.

Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976. ISSN 0001-0782.

H.L.A. Hart. *The Concept of Law*. Clarendon Press, Oxford, UK, 2nd edition, 1994.

Friedrich A. Hayek. The Use of Knowledge in Society. *American Economic Review*, 35:519–530, September 1945. Reprinted in F.A. Hayek (ed.), Individualism and Economic Order. London: Routledge and Kegan Paul, 1949.

Orit Hazzan and James E. Tomayko. The Reflective Practitioner Perspective in eXtreme Programming. In Frank Maurer and Don Wells, editors, *XP/Agile Universe*, volume 2753 of *Lecture Notes in Computer Science*, pages 51–61. Springer, 2003. ISBN 3-540-40662-X.

Qingfeng He and Annie I. Antón. Requirements-based Access Control Analysis and Policy Specification (ReCAPS). *Information and Software Technology*, 51(6): 993–1009, 2009. ISSN 0950-5849.

Juhani Heikka and Mikko Siponen. Abuse Cases Revised: An Action Research Experience. In *PACIS 2006 Proceedings*, 2006.

Cormac Herley. So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users. In *NSPW '09: Proceedings of the 2009 workshop on New security paradigms*, 2010.

Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. Threat Modeling – Uncover Security Design Flaws Using The STRIDE Approach. Online, 2006. URL `http://msdn.microsoft.com/en-us/magazine/cc163519.aspx`. Retrieved 2010/02/03.

Debra S. Herrmann. *Complete Guide to Security and Privacy Metrics*. Auerbach Publications, Boston, MA, USA, 2007. ISBN 0849354021.

Morten Hertzum, Niels Jørgensen, and Mie Nørgaard. Usable Security and E-Banking: ease of use vis-a-vis security. *Australasian Journal of Information Systems*, 11(2), 2007. ISSN 1326-2238.

Almut Herzog and Nahid Shahmehri. A Usability Study of Security Policy Management. In *Security and Privacy in Dynamic Environments (SEC)*, volume 201, pages 296–306. Springer, 2006.

Almut Herzog and Nahid Shahmehri. User help techniques for usable security. In *CHIMIT '07: Proceedings of the 2007 symposium on Computer human interaction for the management of information technology*, page 11, New York, NY, USA, 2007. ACM. ISBN 1-59593-635-6. doi: 10.1145/1234772.1234787.

HHS. *Research-based Web Design and Usability Guidelines*. U.S. Department of Health and Human Services. URL `http://www.usability.gov/guidelines`. Accessed online 2011/05/23.

James A. Highsmith. *Agile software development ecosystems*. Addison-Wesley, Boston, MA, USA, 2002. ISBN 0-201-76043-6.

HIPAA. Break Glass Procedure: Granting Emergency Access to Critical ePHI Systems. Retrieved on Jan, 11 2009, 2009. URL `http://hipaa.yale.edu/security/sysadmin/breakglass.html`.

Andreas Höfer and Marc Philipp. An Empirical Study on the TDD Conformance of Novice and Expert Pair Programmers. In Pekka Abrahamsson, Michele Marchesi, and Frank Maurer, editors, *XP '09: 10th International Conference on Extreme Programming and Agile Processes in Software Engineering*, volume 31 of *Lecture Notes in Business Information Processing*, pages 33–42. Springer, 2009. ISBN 978-3-642-01852-7.

Lance J. Hoffman, Eric H. Michelman, and Don Clements. SECURATE – Security evaluation and analysis using fuzzy metrics. In *International Workshop on Managing Requirements Knowledge*, pages 531–540, Los Alamitos, CA, USA, 1978. IEEE Computer Society. doi: 10.1109/AFIPS.1978.169.

Dirk W. Hoffmann. *Software-Qualität*. Springer, Berlin / Heidelberg, 2008.

Erik Hollnagel. Task Analysis: Why, What, and How. In *Handbook of Human Factors and Ergonomics* Salvendy (2006).

Andreas Holzinger. Usability engineering methods for software developers. *Commun. ACM*, 48(1):71–74, 2005. ISSN 0001-0782. doi: 10.1145/1039539.1039541.

Michael Howard and David E. Leblanc. *Writing Secure Code*. Microsoft Press, Redmond, WA, USA, 2002. ISBN 0735617228.

Vincent C. Hu, David Ferraiolo, and D. Rick Kuhn. Assessment of Access Control Systems. Technical report, NIST, 2006. IR-7316.

Liang Huang and Mike Holcombe. Empirical investigation towards the effectiveness of Test First programming. *Information and Software Technology*, 51(1):182–194, 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.03.007.

Hanna Hulkko and Pekka Abrahamsson. A multiple case study on the impact of pair programming on product quality. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 495–504, 15-21 2005. doi: 10.1109/ICSE.2005.1553595.

Ming Huo, June Verner, Liming Zhu, and Muhammad Ali Babar. Software Quality and Agile Methods. *Computer Software and Applications Conference, Annual International*, 1:520–525, 2004. ISSN 0730-3157. doi: 10.1109/CMPSAC.2004.1342889.

JeeHyun Hwang, Evan Martin, Tao Xie, and Vincent C. Hu. Policy-Based Testing, 2008. Entry submitted for publication in the Encyclopedia of Software Engineering.

Baetrice Hwong, D. Laurance, Arnold Rudorfer, and Xiping Song. User-Centered Design and Agile Software Development Processes. In *Identifying 2004, Workshop Bridging Gaps Between HCI and Software Eng ineering and Design, and Boundary Objects to Bridge Them*, CHI workshop, 2004.

Sylvia Ilieva, Penko Ivanov, and Eliza Stefanova. Analyses of an Agile Methodology Implementation. In *EUROMICRO Conference*, Los Alamitos, CA, USA, 2004. IEEE Computer Society. doi: 10.1109/EURMIC.2004.1333387.

Philip Inglesant and M. Angela Sasse. Information Security as Organizational Power. In *STAST '11: Workshop on Socio-Technical Aspects in Security and Trust*. IEEE, 2011.

Philip Inglesant, M. Angela Sasse, David Chadwick, and Lei Lei Shi. Expressions of expertness: the virtuous circle of natural language for access control policy specification. In *SOUPS '08: Proceedings of the 4th symposium on Usable privacy and security*, pages 77–88, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-276-4. doi: 10.1145/1408664.1408675.

Cynthia Irvine and Timothy Levin. Quality of security service. In *NSPW '00: Proceedings of the 2000 workshop on New security paradigms*, pages 91–99, New York, NY, USA, 2000. ACM. ISBN 1-58113-260-3. doi: 10.1145/366173.366195.

ISF. *The Standard of Good Practice for Information Security*. Information Security Forum, 2007.

ISO 10007:2003. *Quality management systems – Guidelines for configuration management*. ISO, Geneva, Switzerland, 2003.

ISO 9241-110:2006. *Ergonomics of Human System Interaction – Part 110: Dialogue principles*. ISO, Geneva, Switzerland, 2006.

ISO 9241-11:1998. *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*. ISO, Geneva, Switzerland, 1998.

ISO/IEC 15408-1:2009. *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*. ISO, Geneva, Switzerland, 2009.

ISO/IEC 27001:2005. *Information technology – Security techniques – Information security management systems – Requirements*. ISO, Geneva, Switzerland, 2005.

ISO/IEC 27002:2005. *Information technology – Security techniques – Code of practice for information security management*. ISO, Geneva, Switzerland, 2005.

ISO/IEC 27005:2008. *Information technology – Security techniques – Information security risk management*. ISO, Geneva, Switzerland, 2008.

ISO/IEC 9126-1. *Software engineering – Product quality – Part 1: Quality model*. ISO, Geneva, Switzerland, 2001.

ISO/IEC JTC 1/SC 27 Secretariat. Standing Document 7 (SD7): Catalogue of ISO/IEC JTC 1/SC 27 Standards and Projects. Online, retrieved 23 Sep 2010, 2009. URL `http://www.jtc1sc27.din.de/sbe/SD7`.

ISSEA. The Systems Security Engineering Capability Maturity Model (SSE-CMM), Model Document, 2003. Version 3.0.

IT Governance Institute. *CobiT 4.1*. ITIG, Rolling Meadows, IL, 2007.

Blake Ives and Margrethe H. Olson. User Involvement and MIS Success: A Review of Research. *Management Science*, 30(5):586–603, 1984. ISSN 00251909.

Trent Jaeger, Antony Edwards, and Xiaolan Zhang. Consistency analysis of authorization hook placement in the Linux security modules framework. *ACM Trans. Inf. Syst. Secur.*, 7(2):175–205, 2004. ISSN 1094-9224. doi: 10.1145/996943.996944.

Pooya Jaferian, David Botta, Fahimeh Raja, Kirstie Hawkey, and Konstantin Beznosov. Guidelines for designing IT security management tools. In *CHIMiT '08: Proceedings of the 2nd ACM Symposium on Computer Human Interaction for Management of Information Technology*, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-355-6. doi: 10.1145/1477973.1477983.

Jeevan Jaisingh and Jackie Rees. Value at Risk: A Methodology for Information Security Risk Assessment. In *In Proceedings of the INFORMS Conference on Information Systems and Technology*, 2001.

Sushil Jajodia, Shashi K. Gadia, and Gautam Bhargava. Logical design of audit information in relational databases. In Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors, *Information Security: An Integrated Collection of Essays*. IEEE Computer Society Press, 1995.

Leonard M. Jessup and Joseph S. Valacich. *Information systems today: managing in the digital world*. Pearson Prentice Hall, 3. ed edition, 2008.

Gregor Joeris. Change Management Needs Integrated Process and Configuration Management. In Mehdi Jazayeri and Helmut Schauer, editors, *ESEC / SIGSOFT FSE*, volume 1301 of *Lecture Notes in Computer Science*, pages 125–141. Springer, 1997. ISBN 3-540-63531-9. doi: 10.1145/267895.267907.

Maritza Johnson, Steven Bellovin, Robert Reeder, and Stuart Schechter. Laissez-faire file sharing: access control designed for individuals at the endpoints. In *New Security Paradigms Workshop 2009*, pages 1–10, 2009. doi: 10.1145/1719030.1719032.

P.N. Johnson-Laird. Mental models in Cognitive science. *Cognitive Science*, 4(1):71–115, Jan 1980. doi: 10.1207/s15516709cog0401_4.

Russell L. Jones and Abhinav Rastogi. Secure Coding: Building Security into the Software Development Life Cycle. *Information Systems Security*, 13(5):29–39, 2004. doi: 10.1201/1086.

Audun Jøsang, Bander AlFayyadh, Tyrone Grandison, Mohammed AlZomai, and Judith McNamara. Security Usability Principles for Vulnerability Analysis and Risk Assessment. In *ACSAC '07: Proceedings of the 23rd Annual Computer Security Applications Conference*, pages 269–278, 2007.

James B.D. Joshi, Elisa Bertino, and Arif Ghafoor. An analysis of expressiveness and design issues for the generalized temporal role-based access control model. *IEEE Transactions on Dependable and Secure Computing*, 2 (2):157 – 175, april-june 2005. ISSN 1545-5971. doi: 10.1109/TDSC.2005.18.

Helmut Jungermann, Holger Schütz, and Manfred Thüring. Mental Models in Risk Assessment: Informing People About Drugs. *Risk Analysis*, 8(1):147–155, 1988. ISSN 1539-6924. doi: 10.1111/j.1539-6924.1988.tb01161.x.

Jan Jürjens. UMLsec: Extending UML for Secure Systems Development. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 412–425, London, UK, 2002. Springer-Verlag. ISBN 3-540-44254-5.

Daniel Kahneman and Amos Tversky. The simulation heuristic. In Kahneman et al. (1982).

Daniel Kahneman, Paul Slovic, and Amos Tversky, editors. *Judgment under uncertainty: heuristics and biases*. Cambridge University Press, Cambridge, MA, USA, 1982.

Apu Kapadia, Geetanjali Sampemane, and Roy H. Campbell. KNOW Why your access was denied: regulating feedback for usable security. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2004. ACM. ISBN 1-58113-961-6.

Bilge Karabacak and Ibrahim Sogukpinar. ISRAM: information security risk analysis method. *Computers & Security*, 24(2):147–159, 2005. ISSN 0167-4048.

Clare-Marie Karat. Iterative Usability Testing of a Security Application. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 33:273–277(5), 1989.

Daniel Karlström and Per Runeson. Combining agile methods with stage-gate project management. *IEEE Software*, 22(3):43–49, 2005. ISSN 0740-7459. doi: 10.1109/MS.2005.59.

Alan H. Karp and Marc Stiegler. Making policy decisions disappear into the user's workflow. In *CHI EA '10: Proceedings of the 28th of the international conference on Human factors in computing systems*, pages 3247–3252, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-930-5. doi: 10.1145/1753846.1753966.

Gerold Keefer. Extreme Programming Considered Harmful for Reliable Software Development 2.0. Online-Report, 2003.

Moira J. Kelly. Qualitative Evaluation Research. In Seale et al. (2007).

Udo Kelter, Marc Monecke, and Markus Schild. Do We Need 'Agile' Software Development Tools? In Mehmet Aksit, Mira Mezini, and Rainer Unland, editors, *NetObjectDays*, volume 2591 of *Lecture Notes in Computer Science*, pages 412–430. Springer, 2002. ISBN 3-540-00737-7.

Hossein Keramati and Seyed-Hassan Mirian-Hosseinabadi. Integrating software development security activities with agile methodologies. In *IEEE/ACS International Conference on Computer Systems and Applications, 2008. AICCSA 2008*, pages 749–754, 2008.

Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, Jan 1883.

Axel Kern, Martin Kuhlmann, Andreas Schaad, and Jonathan Moffett. Observations on the role life-cycle in the context of enterprise security management. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 43–51, New York, NY, USA, 2002. ACM. ISBN 1-58113-496-7.

Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In *ECOOP'97 — Object-Oriented Programming*, pages 220–242, 1997.

Georgia Killcrece. Incident Management. Online, retrieved 5 Aug 2010, 2008. URL `https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/incident/223-BSI.html`.

Tiffany Hyun-Jin Kim, Lujo Bauer, Newso James, Adrian Perrig, and Jesse Walker. Challenges in access right assignment for secure home networks. In *Proceedings of HotSec'10*, Berkeley, CA, USA, 2010. USENIX Association.

Stephen Jay Kline. *Conceptual foundations for multidisciplinary thinking*. Stanford University Press, Stanford, 1995.

Henrik Kniberg and Mattias Skarin. *Kanban and Scrum – making the most of both*. C4Media, 2010.

Kathleen Knoll and Sirkka L. Jarvenpaa. Information technology alignment or "fit" in highly turbulent environments: the concept of flexibility. In *SIGCPR '94: Proceedings of the 1994 computer personnel research conference on Reinventing IS*, pages 1–14, New York, NY, USA, 1994. ACM. ISBN 0-89791-652-2. doi: 10.1145/186281.186286.

Andrew J. Ko, Robert DeLine, and Gina Venolia. Information Needs in Collocated Software Development Teams. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 344–353, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2828-7. doi: 10.1109/ICSE.2007.45.

Manuel Koch and Francesco Parisi-Presicce. Formal access control analysis in the software development process. In *FMSE '03: Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, pages 67–76, New York, NY, USA, 2003. ACM. ISBN 1-58113-781-8. doi: 10.1145/1035429.1035437.

Marko Komlenovic, Mahesh Tripunitara, and Toufik Zitouni. An empirical assessment of approaches to distributed enforcement in role-based access control (RBAC). In *CODASPY '11: Proceedings of the first ACM conference on Data and application security and privacy*, pages 121–132, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0466-5. doi: 10.1145/1943513.1943530.

Vidar Kongsli. Towards agile security in web applications. In *OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 805–808, New York, NY, USA, 2006. ACM. ISBN 1-59593-491-X.

Mikko Korkala, Pekka Abrahamsson, and Pekka Kyllonen. A case study on the impact of customer communication on defects in agile software development. In *AGILE 2006*. IEEE Computer Society, 2006. doi: 10.1109/AGILE.2006.1.

Andrew G. Kotulic and Jan Guynes Clark. Why there aren't more information security research studies. *Information & Management*, 41(5):597–607, 2004. ISSN 0378-7206. doi: 10.1016/j.im.2003.08.001.

Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 179–186, New York, NY, USA, 2003. ACM. ISBN 1-58113-681-1.

Thomas S. Kuhn. *The Structure of Scientific Revolution*. University of Chicago Press, 2nd edition, 1970.

Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: theory and practice. *ACM Trans. Comput. Syst.*, 10: 265–310, November 1992. ISSN 0734-2071. doi: 10.1145/138873.138874.

Craig Larman and Victor R. Basili. Iterative and Incremental Development: A Brief History. *Computer*, 36(6):47–56, 2003. ISSN 0018-9162.

Bruno Latour. Technology is Society Made Durable. In John Law, editor, *A Sociology of Monsters Essays on Power Technology and Domination*. Routledge, London, UK, 1991.

Paul R. Lawrence and Jay W. Lorsch. *Organization and environment. Managing differentiation and integration*. Irwin, Homewood, 1969.

Lucas Layman, Laurie Williams, and Lynn Cunningham. Exploring Extreme Programming in Context: An Industrial Case Study. In *ADC '04: Proceedings of the Agile Development Conference*, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2248-3.

Lucas Layman, Laurie Williams, and Lynn Cunningham. Motivations and measurements in an agile case study. *Journal of Systems Architecture*, 52(11):654–667, 2006. ISSN 1383-7621. doi: 10.1016/j.sysarc.2006.06.009.

Meir M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.

Arjen K. Lenstra and Tim Voss. Information Security Risk Assessment, Aggregation, and Mitigation. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 391–401, Berlin Heidelberg, 2004. Springer. ISBN 3-540-22379-7.

Lawrence Lessig. The New Chicago School. *The Journal of Legal Studies*, 27(2), June 1998.

Marek Leszak, Dewayne E. Perry, and Dieter Stoll. A case study in root cause defect analysis. In *ICSE '00: Proceedings of the 22th International Conference on Software Engineering*, pages 428–437, 2000. doi: 10.1109/ICSE.2000.870433.

James R. Lewis. Usability Testing. In *Handbook of Human Factors and Ergonomics* Salvendy (2006).

Ninghui Li and Mahesh V. Tripunitara. Security analysis in role-based access control. *ACM Trans. Inf. Syst. Secur.*, 9(4):391–420, 2006. ISSN 1094-9224.

Henry Lieberman, editor. *Your wish is my command: programming by example*. Morgan Kaufmann, San Francisco, 2001. ISBN 1558606882.

Henry Lieberman. *End user development*. Springer, 2006.

Steve Lipner. The Trustworthy Computing Security Development Lifecycle. In *ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference*, pages 2–13, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2252-1. doi: 10.1109/CSAC.2004.41.

Lin Liu, Eric Yu, and John Mylopoulos. Security and privacy requirements analysis within a social setting. In *Requirements Engineering Conference*, pages 151–161, 8-12 2003.

John Locke. *Two Treatises of Government: In the Former, the false Principles and Foundations of Sir Robert Filmer, And his Followers, are Detected and Overthrown. The Latter, is an Essay Concerning the True Origin, Extent, and End of Civil Government*. A. Bettesworth, Pater-Noster-Row, 5th ed. edition, 1728.

Torsten Lodderstedt, David A. Basin, and Jürgen Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 426–441, London, UK, 2002. Springer-Verlag. ISBN 3-540-44254-5.

Jim J. Longstaff, Mike A. Lockyer, and Michael G. Thick. A model of accountability, confidentiality and override for healthcare and other applications. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 71–76, New York, NY, USA, 2000. ACM. ISBN 1-58113-259-X.

Francisco Macias, Mike Holcombe, and Marian Gheorghe. A Formal Experiment Comparing Extreme Programming with Traditional Software Construction. In *Mexican International Conference on Computer Science*, Los Alamitos, CA, USA, 2003. IEEE Computer Society. ISBN 0-7695-1915-6. doi: 10.1109/ENC.2003.1232877.

Adrian Mackenzie and Simon Monk. From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice. *Comput. Supported Coop. Work*, 13(1):91–117, 2004. ISSN 0925-9724. doi: 10.1023/B:COSU.0000014873.27735.10.

George Magklaras and Steven Furnell. Insider Threat Prediction Tool: Evaluating the probability of IT misuse. *Computers & Security*, 21(1):62–73, 2002.

Thomas W. Malone. *The future of work: how the new order of business will shape your organization, your management style, and your life*. Harvard Business Press, 2004.

Chris Mann and Frank Maurer. A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In *ADC '05: Proceedings of the Agile Development Conference*, pages 70–79, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2487-7. doi: 10.1109/ADC.2005.1.

Katiuscia Mannaro, Marco Melis, and Michele Marchesi. Empirical Analysis on the Satisfaction of IT Employees Comparing XP Practices with Other Software Development Methodologies. In *XP '04: 5th International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 166–174. Springer, 2004.

James G. March, Herbert Alexander Simon, and Harold Steere Guetzkow. *Organizations*. Wiley, New York, 1958. ISBN 0471567930.

Angela Martin, Robert Biddle, and James Noble. The XP customer role in practice: three studies. In *ADC '04: Agile Development Conference*, pages 42–54, 22-26 2004. doi: 10.1109/ADEVC.2004.23.

Angela Martin, Robert Biddle, and James Noble. The XP Customer Team: A Grounded Theory. In *AGILE 2009*, pages 57–64, 24-28 2009.

Evan Martin, Tao Xie, and Ting Yu. Defining and Measuring Policy Coverage in Testing Access Control Policies. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 139–158. Springer, 2006. ISBN 3-540-49496-0.

Evan Martin, JeeHyun Hwang, Tao Xie, and Vincent Hu. Assessing Quality of Policy Properties in Verification of Access Control Policies. In *ACSAC '08: Proceedings of the Annual Computer Security Applications Conference*, pages 163–172, dec. 2008. doi: 10.1109/ACSAC.2008.48.

David S. Martins. Compliance Rhetoric and the Impoverishment of Context. *Communication Theory*, 15(1):59–77, 2005. ISSN 1468-2885. doi: 10.1111/j.1468-2885.2005.tb00326.x.

Roy A. Maxion and Robert W. Reeder. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies*, 63(1-2):25–50, 2005. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2005.04.009.

Pete McBreen. *Questioning Extreme Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 0201844575.

John McDermott and Chris Fox. Using Abuse Case Models for Security Requirements Analysis. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, page 55, Washington, DC, USA, 1999. IEEE Computer Society.

Gary McGraw. *Software Security: Building Security In*. Addison-Wesley, 2006.

Paul McInerney and Frank Maurer. UCD in agile projects: dream team or odd couple? *interactions*, 12(6):19–23, 2005. ISSN 1072-5520. doi: 10.1145/1096554.1096556.

Robert D. McPhee and Marshall Scott Poole. Organizational Structures and Configurations. In Fredric M. Jablin and Linda L. Putnam, editors, *The New Handbook of Organizational Communication*. SAGE, 2000.

Nancy R. Mead and Ted Stehney. Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7, 2005. doi: 10.1145/1082983.1083214.

Daniel Mellado, Eduardo Fernández-Medina, and Mario Piattini. A Comparative Study of Proposals for Establishing Security Requirements for the Development of Secure Information Systems. In Marina L. Gavrilova et al., editors, *ICCSA (3)*, volume 3982 of *Lecture Notes in Computer Science*, pages 1044–1053. Springer, 2006. ISBN 3-540-34075-0.

Grigori Melnik and Frank Maurer. Perceptions of Agile Practices: A Student Survey. In *XP/Agile Universe '02: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods*, pages 241–250, London, UK, 2002. Springer-Verlag. ISBN 3-540-44024-0.

Grigori Melnik and Frank Maurer. A cross-program investigation of students' perceptions of agile methods. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 481–488, New York, NY, USA, 2005. ACM. ISBN 1-59593-963-2. doi: 10.1145/1062455.1062543.

Charles G. Menk. System Security Engineering Capability Maturity Model and Evaluations: Partners within the Assurance Framework. In *19th National Information Systems Security Conference*, 1996.

Gerard Meszaros, Shaun Smith, and Jennitta Andrea. The Test Automation Manifesto. In Frank Maurer and Don Wells, editors, *XP/Agile Universe*, volume 2753 of *Lecture Notes in Computer Science*, pages 73–81. Springer, 2003.

Peter Middleton. Lean Software Development: Two Case Studies. *Software Quality Control*, 9(4):241–252, 2001. ISSN 0963-9314. doi: 10.1023/A:1013754402981.

Raymond E. Miles, Charles C. Snow, Alan D. Meyer, and Henry J. Coleman. Organizational Strategy, Structure, and Process. *The Academy of Management Review*, 3(3), July 1978.

Harlan D. Mills, Michael Dyer, and Richard C. Linger. Cleanroom Software Engineering. *Software, IEEE*, 4(5): 19–25, sept. 1987. ISSN 0740-7459. doi: 10.1109/MS.1987.231413.

Marvin Minsky. Matter, Mind and Models. Technical Report MAC-M-230, Massachusetts Institute of Technology (MIT), 1965.

Henry Mintzberg. Structure in 5's: A Synthesis of the Research on Organization Design. *Management Science*, 26(3), March 1980.

Henry Mintzberg. *Mintzberg on management: inside our strange world of organizations*. Free Press, New York, NY, USA, 1989.

Subhas Chandra Misra, Vinod Kumar, and Uma Kumar. Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 82(11):1869–1890, 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2009.05.052.

Jonathan D. Moffett. Control principles and role hierarchies. In *Proceedings of the third ACM workshop on Role-based access control*, RBAC '98, pages 63–69, New York, NY, USA, 1998. ACM. ISBN 1-58113-113-5. doi: 10.1145/286884.286900.

Ian Molloy, Pau-Chen Cheng, and Pankaj Rohatgi. Trading in Risk: Using Markets to Improve Access Control. In *NSPW '08: Proceedings of the 2008 Workshop on New Security Paradigms*, pages 107–125, 2008. doi: 10.1145/1595676.1595694.

Peter R. Monge and Noshir S. Contractor. Emergence of Communication Networks. In Fredric M. Jablin and Linda L. Putnam, editors, *The New Handbook of Organizational Communication*. SAGE, 2000.

Axel Mönkeberg and René Rakete. Three for one: role-based access-control management in rapidly changing heterogeneous environments. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 83–88, New York, NY, USA, 2000. ACM. ISBN 1-58113-259-X. doi: 10.1145/344287.344306.

Andrew Moore, Dawn Cappelli, and Randall F. Trzeciak. The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures. Technical Report CMU/SEI-2008-TR-009, CarnegieMellon, May 2008.

Andrew P. Moore, Robert J. Ellison, and Richard C. Linger. Attack Modeling for Information Security and Survivability. Technical Report CMU/SEI-2001-TN-001, CarnegieMellon, March 2001.

Tim Moses. *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS, 2005. URL http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

Haralambos Mouratidis and Paolo Giorgini. Secure Tropos: a Security-Oriented Extension of the Tropos Methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.

Robert Musil. *The man without qualities*. Pan Macmillan, 1995.

Brad A. Myers, Richard McDaniel, and David Wolber. Programming by example: intelligence in demonstrational interfaces. *Commun. ACM*, 43(3):82–89, 2000. ISSN 0001-0782.

Nachiappan Nagappan, E. Michael Maximilien, Thirumalesh Bhat, and Laurie Williams. Realizing quality improvement through test driven development: results and experiences of four industrial teams. *Empirical Softw. Engg.*, 13(3):289–302, 2008. ISSN 1382-3256. doi: 10.1007/s10664-008-9062-z.

National Computer Science Center. *A Guide to Understanding Discretionary Access Control in Trusted Systems*. NCSC, 1987. NCSC-TG-003-87.

Sridhar Nerur and VenuGopal Balijepally. Theoretical reflections on agile development methodologies. *Commun. ACM*, 50(3):79–83, 2007. ISSN 0001-0782. doi: 10.1145/1226736.1226739.

Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Commun. ACM*, 48:72–78, May 2005. doi: 10.1145/1060710.1060712.

Gustaf Neumann and Mark Strembeck. A scenario-driven role engineering process for functional RBAC roles. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 33–42, New York, NY, USA, 2002. ACM. ISBN 1-58113-496-7. doi: 10.1145/507711.507717.

Ojelanki K. Ngwenyama. Developing end-users' systems development competence: An exploratory study. *Information & Management*, 25(6):291–302, 1993. ISSN 0378-7206.

Jakob Nielsen. *Usability engineering*. AP Professional, Boston [u.a.], 4th edition, 1997. ISBN 0125184069.

Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 206–213, New York, NY, USA, 1993. ACM. ISBN 0-89791-575-5. doi: 10.1145/169059.169166.

Friedrich Wilhelm Nietzsche. *The Genealogy of Morals*. Dover Publications, Mineola, NY, USA, 1913/2003.

Nimal Nissanke and Etienne J. Khayat. Risk Based Security Analysis of Permissions in RBAC. In Eduardo Fernández-Medina, Julio César Hernández Castro, and L. Javier García-Villalba, editors, *WOSIS*, pages 332–341. INSTICC Press, 2004. ISBN 972-8865-07-4.

NIST. FIPS 65: Guidelines for Automatic Data Processing Risk Analysis. Technical report, NIST, 1975.

NIST. FIPS 191: Guideline for The Analysis Local Area Network Security. Technical report, NIST, 1994. URL http://www.itl.nist.gov/fipspubs/fip191.htm.

NIST. Special Publication 800-12: An Introduction to Computer Security – The NIST Handbook. Technical report, National Institute of Standards and Technology, 1995. URL http://csrc.nist.gov/publications/nistpubs/800-12/800-12-html/index.html.

Marcus Nohlberg. Why Humans are the Weakest Link. In Gupta and Sharman (2009).

Jason R. C. Nurse, Sadie Creese, Michael Goldsmith, and Koen Lamberts. Guidelines for Usable Cybersecurity: Past and Present. In *CSS '11: The 3rd International Workshop on Cyberspace Safety and Security*. IEEE, 2011.

Andrew M. Odlyzko. Economics, Psychology, and Sociology of Security. In Rebecca N. Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 182–189. Springer, 2003. ISBN 3-540-40663-8.

OECD. Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security. Online, retrieved 23 Sep 2010, 2002. URL http://www.oecd.org/document/42/0,3343,en_2649_34255_15582250_1_1_1_1,00.html.

OGC. *ITILv3 – Continual Service Improvement*. TSO, London, UK, 2007a.

OGC. *The official introduction to the ITIL service lifecycle*. TSO, London, UK, 2007b.

OGC. *ITILv3 – Service Operation*. TSO, London, UK, 2007c.

OWASP. CLASP Project. Online, retrieved 5 Aug 2010. URL http://www.owasp.org/index.php/Category:OWASP_CLASP_Project.

Steven Pace. A grounded theory of the flow experiences of Web users. *International Journal of Human-Computer Studies*, 60(3):327–363, 2004. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2003.08.005.

Frank Pallas. *Information Security Inside Organizations – A Positive Model and Some Normative Arguments Based on New Institutional Economics*. PhD thesis, TU Berlin, 2009.

Steve R. Palmer and Mac Felsing. *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001. ISBN 0130676152.

Raju Pandey and Brant Hashii. Providing Fine-grained Access Control for Java Programs. In Rachid Guerraoui, editor, *ECOOP '99: Proceedings of 13th European Conference Object-Oriented Programming*, volume 1628 of *Lecture Notes in Computer Science*, pages 449–473. Springer, 1999. ISBN 3-540-66156-5.

John F. Pane, Chotirat Ann Ratanamahatana, and Brad A. Myers. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54(2): 237–264, 2001. ISSN 1071-5819.

Simon Parkin, Aad van Moorsel, Philip Inglesant, and M. Angela Sasse. A stealth approach to usable security: helping IT security managers to identify workable security solutions. In *NSPW '10: Proceedings of the 2010 workshop on New security paradigms*, pages 33–50, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0415-3. doi: 10.1145/1900546.1900553.

David Parsons, Hokyoung Ryu, and Ramesh Lal. The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects. In *Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda*. Springer, 2007.

Joshua J. Pauli and Dianxiang Xu. Misuse case-based design and analysis of secure software architecture. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 398–403 Vol. 2, 4-6 2005.

Stephen J. Payne. Users' Mental Models: The Very Ideas. In Carroll (2003).

Holger Peine. Rules of thumb for secure software engineering. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 702–703, New York, NY, USA, 2005. ACM. ISBN 1-59593-963-2. doi: 10.1145/1062455.1062626.

Thomas R. Peltier. *Information security risk analysis*. CRC press, Boca Raton, FL, 2005.

Günther Pernul. Information systems security: Scope, state-of-the-art, and evaluation of techniques. *International Journal of Information Management*, 15(3):165–180, 1995. ISSN 0268-4012. doi: 10.1016/0268-4012(95)00010-5.

Kai Petersen and Claes Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, 2010.

Shari Lawrence Pfleeger and Salvatore J. Stolfo. Addressing the Insider Threat. *IEEE Security and Privacy*, 7(6), 2009.

Minna Pikkarainen, Jukka Haikara, Outi Salo, Pekka Abrahamsson, and Jari Still. The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3):303–337, 2008.

Jan Ploski, Matthias Rohr, Peter Schwenkenberg, and Wilhelm Hasselbring. Research issues in software fault categorization. *SIGSOFT Softw. Eng. Notes*, 32, November 2007. ISSN 0163-5948. doi: 10.1145/1317471.1317478.

Mary Poppendieck. XP in a Safety-Critical Environment. *Cutter IT Journal*, September 2002. URL `http://www.poppendieck.com/safety.htm`.

Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 0321150783.

Karl Raimund Popper. *The logic of scientific discovery*. Routledge, 1935/2002.

Gerald V. Post and David J. Diltz. A stochastic dominance approach to risk analysis of computer systems. *MIS Q.*, 10:363–376, December 1986. ISSN 0276-7783. doi: 10.2307/249191.

Dean Povey. Optimistic security: a new access control paradigm. In *NSPW '99: Proceedings of the 1999 workshop on New security paradigms*, pages 40–45, New York, NY, USA, 2000. ACM. ISBN 1-58113-149-6.

Torsten Priebe, Eduardo B. Fernández, Jens Ingo Mehlau, and Günther Pernul. A Pattern System for Access Control. In Csilla Farkas and Pierangela Samarati, editors, *DBSec*, pages 235–249. Kluwer, 2004. ISBN 1-4020-8127-8.

Asif Qumer and Brian Henderson-Sellers. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology*, 50(4):280–295, 2008. ISSN 0950-5849. doi: 10.1016/j.infsof.2007.02.002.

Rex Kelly Rainer, Charles A. Snyder, and Houston H. Carr. Risk analysis for information technology. *J. Manage. Inf. Syst.*, 8(1):129–147, June 1991. ISSN 0742-1222.

Narayan Ramasubbu and Rajesh Krishna Balan. The impact of process choice in high maturity environments: An empirical analysis. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 529–539, 16-24 2009. doi: 10.1109/ICSE.2009.5070551.

Marisa Reddy Randazzo, Michelle Keeney, Eileen Kowalski, Dawn Cappelli, and Andrew Moore. Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector. Technical Report CMU/SEI-2004-TR-021, CarnegieMellon, June 2005.

Minna Räsänen and James M. Nyce. A new role for anthropology?: rewriting "context" and "analysis" in HCI research. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, pages 175–184, New York, NY, USA, 2006. ACM. ISBN 1-59593-325-5.

Jens Rasmussen. Risk management in a dynamic society: a modelling problem. *Safety Science*, 27(2-3):183–213, 1997. ISSN 0925-7535. doi: 10.1016/S0925-7535(97)00052-0.

Indrakshi Ray, Na Li, Robert B. France, and Dae-Kyoo Kim. Using uml to visualize role-based access control constraints. In Trent Jaeger and Elena Ferrari, editors, *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 115–124. ACM, 2004. ISBN 1-58113-872-5.

Joseph Raz, editor. *Authority*. Readings in Social and Political Theory. New York University Press, New York, NY, USA, 1990.

Maryam N. Razavi and Lee Iverson. A grounded theory of information sharing behavior in a personal learning space. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, New York, NY, USA, 2006. ACM. ISBN 1-59593-249-6. doi: 10.1145/1180875.1180946.

James Reason. *Human error*. Cambridge University Press, New York, 1990.

Robert W. Reeder, Clare-Marie Karat, John Karat, and Carolyn Brodie. Usability Challenges in Security and Privacy Policy-Authoring Interfaces. In Maria Cecilia Calani Baranauskas, Philippe A. Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa, editors, *INTERACT (2)*, volume 4663 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2007. ISBN 978-3-540-74799-4.

Robert W. Reeder, Lujo Bauer, Lorrie Faith Cranor, Michael K. Reiter, Kelli Bacon, Keisha How, and Heather Strong. Expandable grids for visualizing and authoring computer security policies. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1473–1482, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357285.

Robert W. Reeder, Lujo Bauer, Lorrie F. Cranor, Michael K. Reiter, and Kami Vaniea. More than skin deep: measuring effects of the underlying model on access-control system usability. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2065–2074, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979243.

Jackie Rees, Subhajyoti Bandyopadhyay, and Eugene H. Spafford. PFIRES: a policy framework for information security. *Commun. ACM*, 46(7):101–106, 2003. ISSN 0001-0782.

Erik Rissanen, Babak Sadighi Firozabadi, and Marek J. Sergot. Towards a Mechanism for Discretionary Overriding of Access Control. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols Workshop*, volume 3957 of *Lecture Notes in Computer Science*, pages 312–319. Springer, 2004. ISBN 3-540-40925-4.

Hugh Robinson and Helen Sharp. The Characteristics of XP Teams. In *XP '04: 5th International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 139–147. Springer, 2004.

Hugh Robinson and Helen Sharp. The Social Side of Technical Practices. In *XP '05: 6th International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 100–108. Springer, 2005.

Jennifer Rode, Carolina Johansson, Paul DiGioia, Roberto Silva Filho, Kari Nies, David H. Nguyen, Jie Ren, Paul Dourish, and David Redmiles. Seeing further: extending visualization as a basis for usable security. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 145–155, New York, NY, USA, 2006. ACM. ISBN 1-59593-448-0. doi: 10.1145/1143120.1143138.

Carl A. Roper, Joseph J. Grau, and Lynn F. Fischer. *Security Education, Awareness and Training: from Theory to Practice*. Butterworth-Heinemann, 2005.

Lillian Røstad and Ole Edsberg. A Study of Access Control Requirements for Healthcare Systems Based on Audit Trails from Access Logs. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference*, pages 175–186. IEEE Computer Society, 2006. ISBN 0-7695-2716-7.

Alexander J. Rothman and Marc T. Kiviniemi. Treating People With Information: an Analysis and Review of Approaches to Communicating Health Risk Information. *J Natl Cancer Inst Monogr*, (25), 1999.

Winston W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of IEEE WESCON*. IEEE, 1970.

Sam Ruby, Dave Thomas, and David Heinemeier Hansson. *Agile Web Development with Rails*. The Pragmatic Bookshelf, Dallas, TX, USA, 4th edition, 2011.

Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in Artificial Intelligence. Prentice Hall, Upper Saddle River, 2nd international ed. edition, 2003. ISBN 0130803022.

Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, Sep 1975. doi: 10.1109/PROC.1975.9939.

Gavriel Salvendy. *Handbook of human factors and ergonomics*. Wiley, 2006.

Pierangela Samarati and Sabrina de Vimercati di Vimercati. Access Control: Policies, Models, and Mechanisms. *Foundations of Security Analysis and Design*, pages 137–196, 2001.

Ravi Sandhu. Good-Enough Security: Toward a Pragmatic Business-Driven Discipline. *IEEE Internet Computing*, 7(1):66–68, 2003. ISSN 1089-7801. doi: 10.1109/MIC.2003.1167341.

Ravi Sandhu and Pierangela Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32 (9):40–48, sep 1994. ISSN 0163-6804. doi: 10.1109/35.312842.

Ravi Sandhu and Pierangela Samarati. Authentication, access control, and audit. *ACM Comput. Surv.*, 28:241–243, March 1996. ISSN 0360-0300. doi: 10.1145/234313.234412.

Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

M. Angela Sasse. Designing for Homer Simpson – d'oh! *Interfaces*, (86), 2011.

M. Angela Sasse, Sacha Brostoff, and Dirk Weirich. Transforming the 'Weakest Link' – a Human/Computer Interaction Approach to Usable and Effective Security. *BT Technology Journal*, 19:122–131, July 2001. ISSN 1358-3948. doi: 10.1023/A:1011902718709.

Andreas Schaad, Volkmar Lotz, and Karsten Sohr. A model-checking approach to analysing organisational controls in a loan origination process. In David F. Ferraiolo and Indrakshi Ray, editors, *SACMAT '06: Proceedings of the 11th ACM symposium on Access control models and technologies*, pages 139–149. ACM, 2006. ISBN 1-59593-353-0. doi: 10.1145/1133058.1133079.

John R. Schermerhorn, James G. Hunt, and Richard N. Osborn. *Organizational Behavior*. Wiley, 10th ed. edition, 2008.

Jürgen Schlegelmilch and Ulrike Steffens. Role mining with ORCA. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 168–176, New York, NY, USA, 2005. ACM. ISBN 1-59593-045-0.

T. Schlienger and S. Teufel. Analyzing information security culture: increased trust by an appropriate information security culture. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, sept. 2003. doi: 10.1109/DEXA.2003.1232055.

N.F. Schneidewind and Heinz-Michael Hoffmann. An Experiment in Software Error Data Collection and Analysis. *IEEE Transactions on Software Engineering*, 5:276–286, 1979. ISSN 0098-5589. doi: 10.1109/TSE.1979.234188.

E. Eugene Schultz. A framework for understanding and predicting insider attacks. *Computers & Security*, 21(6): 526–531, 2002. ISSN 0167-4048.

Markus Schumacher and Utz Roedig. Security Engineering with Patterns. In *PLoP*, 2001.

Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2001.

Judith A. Scully, Shelley A. Kirkpatrick, and Edwin A. Locke. Locus of Knowledge as a Determinant of the Effects of Participation on Performance, Affect, and Perceptions. *Organizational Behavior and Human Decision Processes*, 61(3):276–288, 1995. ISSN 0749-5978. doi: 10.1006/obhd.1995.1022.

Robert Seacord. *Secure coding in C and C++*. Addison-Wesley Professional, 2005. ISBN 9780768685923.

Clive Seale, Giampietro Gobo, Jaber F. Gubrium, and David Silverman, editors. *Qualitative Research Practice*. SAGE, London, UK, 2007.

Andrew Sears and Julie A. Jacko. *The human-computer interaction handbook: fundamentals, evolving technologies, and emerging applications*. Human factors and ergonomics. Erlbaum, New York, 2nd edition, 2008.

Joseph Sharit. Human Error. In *Handbook of Human Factors and Ergonomics* Salvendy (2006).

Helen Sharp and Hugh Robinson. Collaboration and coordination in mature eXtreme programming teams. *International Journal of Human-Computer Studies*, 66(7): 506–518, 2008. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2007.10.004.

Eric Shaw, Keven G. Ruby, and Jerrold M. Post. The Insider Threat to Information Systems – The Psychology of the Dangerous Insider. *Security Awareness Bulletin*, (2), 1998.

Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2005.

Adam Shostack. Experiences Threat Modeling at Microsoft. In Jon Whittle, Jan Jürgens, Bashar Nuseibeh, and Glen Dobson, editors, *Proceedings of the Workshop on Modeling Security (MODSEC08)*, 2008.

Kerstin V. Siakas and Errikos Siakas. The Agile Professional Culture: A Source of Agile Quality. *Softw. Process Improve. Pract.*, 12, 2007.

Klaas Sikkel and Oliver Stiemerling. User-Oriented Authorization in Collaborative Environments. In *COOP '98*, 1998.

Herbert A. Simon. From substantive to procedural rationality. In Frank Hahn and Martin Hollis, editors, *Philosophy and Economic Theory*. Oxford University Press, 1979.

Sara Sinclair, Sean W. Smith, Stephanie Trudeau, M. Eric Johnson, and Anthony Portera. Information Risk in Financial Institutions: Field Study and Research Roadmap. In *Proceedings for the 3rd International Workshop on Enterprise Applications and Services in the Finance Industry*, pages 165–180, 2008. doi: 10.1007/978-3-540-78550-7_11.

Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. *Requir. Eng.*, 10(1):34–44, 2005.

Maria Siniaalto and Pekka Abrahamsson. A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 275–284, 20-21 2007. doi: 10.1109/ESEM.2007.35.

Mikko T. Siponen. Five dimensions of information security awareness. *SIGCAS Comput. Soc.*, 31:24–29, June 2001. ISSN 0095-2737. doi: 10.1145/503345.503348.

Mikko T. Siponen and Harri Oinas-Kukkonen. A review of information security issues and respective research contributions. *SIGMIS Database*, 38:60–80, February 2007. ISSN 0095-0033. doi: 10.1145/1216218.1216224.

Mikko T. Siponen, Richard Baskerville, and Tapio Kuivalainen. Integrating Security into Agile Development Methods. In *HICSS*. IEEE Computer Society, 2005. ISBN 0-7695-2268-8. doi: 10.1109/HICSS.2005.329.

Diana K. Smetters and Nathan Good. How users use access control. In *SOUPS '09: Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-736-3. doi: 10.1145/1572532.1572552.

Diana K. Smetters and Rebecca E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *NSPW '02: Proceedings of the 2002 workshop on New security paradigms*, pages 82–89, New York, NY, USA, 2002. ACM.

Adam Smith. *An inquiry into the nature and causes of the wealth of nations*. W. Strahan and T. Cadell, London, UK, 1776.

Charles Percy Snow. *The two cultures*. Cambridge University Press, Cambridge, 1993.

Software Engineering Institute. Preventing Security-Related Defects. Online, retrieved 5 Aug 2010, June 2002. URL `http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature12q02.cfm`.

Karsten Sohr and Bernhard Berger. Idea: Towards Architecture-Centric Security Analysis of Software. In Fabio Massacci, Dan S. Wallach, and Nicola Zannone, editors, *ESSoS 2010*, volume 5965 of *Lecture Notes in Computer Science*, pages 70–78. Springer, 2010. ISBN 978-3-642-11746-6.

Karsten Sohr, Michael Drouineaud, Gail-Joon Ahn, and Martin Gogolla. Analyzing and Managing Role-Based Access Control Policies. *IEEE Trans. Knowl. Data Eng.*, 20(7):924–939, 2008.

Ian Sommerville, Tom Rodden, Pete Sawyer, Richard Bentley, and Michael Twidale. Integrating ethnography into the requirements engineering process. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 165–173, jan 1993. doi: 10.1109/ISRE.1993.324821.

Andreas Sotirakopoulos, Kirstie Hawkey, and Konstantin Beznosov. On the challenges in usable security lab studies: lessons learned from replicating a study on SSL warnings. In *SOUPS '11: Proceedings of the 7th Symposium on Usable Privacy and Security*, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0911-0. doi: 10.1145/2078827.2078831.

Eugene H. Spafford. The Internet Worm Program: An Analysis. Technical Report CSD-TR-823, Purdue University, 1988.

William Stallings and Lawrie Brown. *Computer security: principles and practice*. Pearson Prentice Hall, Upper Saddle River, NJ, 2008.

Jeffrey M. Stanton, Kathryn R. Stam, Paul Mastrangelo, and Jeffrey Jolton. Analysis of end user security behaviors. *Computers & Security*, 24(2):124–133, 2005. ISSN 0167-4048. doi: 10.1016/j.cose.2004.07.001.

Jennifer Stapleton. *DSDM: business focused development*. Pearson Education, 2nd ed. edition, 2003.

Susan Leigh Star and James R. Griesemer. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19 (3), August 1989. doi: 10.1177/030631289019003001.

Meir Statman. The Cultures of Risk Tolerance. Available at SSRN, 2010. URL `http://ssrn.com/abstract=1647086`.

Gunnar Stevens and Volker Wulf. A new dimension in access control: studying maintenance engineering across organizational boundaries. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 196–205, New York, NY, USA, 2002. ACM. ISBN 1-58113-560-2.

Oliver Stiemerling and Volker Wulf. Beyond 'Yes or No' - Extending Access Control in Groupware with Awareness and Negotiation. *Group Decision and Negotiation*, 9(3): 221–235, May 2000.

Michael Stonebraker and Peter Rubinstein. The INGRES protection system. In *Proceedings of the 1976 annual conference*, ACM '76, pages 80–84, New York, NY, USA, 1976. ACM. doi: 10.1145/800191.805536.

Gary Stoneburner, Alice Goguen, and Alexis Feringa. Risk Management Guide for Information Technology Systems – NIST Special Publication 800-30. Technical report, National Institute of Standards and Technology, 2002.

Gary Stoneburner, Clark Hayden, and Alexis Feringa. Engineering Principles for Information Technology Security (A Baseline for Achieving Security) – NIST Special Publication 800-27 Rev A. Technical report, National Institute of Standards and Technology, 2004.

Detmar W. Straub and Richard J. Welke. Coping with System Risk: Security Planning Models for Management Decision-Making. *MIS Quarterly*, 22(4):441–469, 1998.

Jeffrey Stylos, Steven Clarke, and Brad A. Myers. Comparing API Design Choices with Usability Studies: A Case Study and Future Directions. In *Proceedings of the 18th Workshop of the Psychology of Programming Interest Group*, 2006.

Bryan Sullivan. Streamline Security Practices For Agile Development, 2008. URL `http://msdn.microsoft.com/en-us/magazine/dd153756.aspx`.

Harald Svensson and Martin Höst. Introducing an Agile Process in a Software Maintenance and Evolution Organization. *Software Maintenance and Reengineering, European Conference on*, 0:256–264, 2005a. ISSN 1534-5351. doi: 10.1109/CSMR.2005.33.

Harald Svensson and Martin Höst. Views from an Organization on How Agile Development Affects Its Collaboration with a Software Development Team. In *Product Focused Software Process Improvement, 6th International Conference, PROFES 2005*, pages 487–501. Springer, 2005b.

Hassan Takabi and James B.D. Joshi. StateMiner: an efficient similarity-based approach for optimal mining of role hierarchy. In *SACMAT '10: Proceeding of the 15th ACM symposium on Access control models and technologies*, pages 55–64, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0049-0. doi: 10.1145/1809842.1809853.

Sarlota A. Takács. *The Construction of Authority in Ancient Rome and Byzantium – The Rhetoric of Empire*. Cambridge University Press, New York, NY, USA, 2009.

David Talby, Orit Hazzan, Yael Dubinsky, and Arie Keren. Reflections on reflection in agile software development. In *AGILE 2006*, pages 11 pp.–112, 23-28 2006. doi: 10.1109/AGILE.2006.45.

Andrew Tappenden, Patricia Beatty, and James Miller. Agile Security Testing of Web-Based Systems via HTTPUnit. In *AGILE 2005*, pages 29–38. IEEE Computer Society, 2005. ISBN 0-7695-2487-7.

Bjørnar Tessem. Experiences in learning XP practices: a qualitative study. In *XP '03: Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering*, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-40215-2.

The CRAMM Manager. CRAMM User Guide Issue 5.1. Technical report, Insight Consulting, 2005.

The National Commission on Terrorist Attacks Upon the United States. *9-11 Commission Report*. U.S. Government Printing Office, July 2004. URL `http://govinfo.library.unt.edu/911/report/index.htm`.

Harold Thimbleby. Supporting Diverse HCI research. In *Proceedings of BCS HCI Conference*, Bristol, 2004. Research Press International.

Harold Thimbleby. *Press On: Principles of Interaction Programming*. MIT Press, 2010.

Bruce Tognazzini. Design for Usability. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and usability: designing secure systems that people can use*. O'Reilly, 2005.

Inger Anne Tondel, Martin Gilje Jaatun, and Per Håkon Meland. Security Requirements for the Rest of Us: A Survey. *Software, IEEE*, 25(1):20–27, jan.-feb. 2008. ISSN 0740-7459. doi: 10.1109/MS.2008.19.

Duane P. Truex, Richard Baskerville, and Heinz Klein. Growing systems in emergent organizations. *Commun. ACM*, 42:117–123, August 1999. ISSN 0001-0782. doi: 10.1145/310930.310984.

John Tsalikis and David J. Fritzsche. Business Ethics: A Literature Review with a Focus on Marketing Ethics. *Journal of Business Ethics*, 8(9), September 1989.

Edward R. Tufte. *Visual explanations: images and quantities, evidence and narrative*. Graphics Press, Cheshire, Conn., 4. print., with revisions edition, 2000. ISBN 0961392126.

Edward R. Tufte. *Beautiful evidence*. Graphics Press, Cheshire, Conn., 2006. ISBN 0961392177.

Edward R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, Conn., 2. ed., 5. print. edition, 2007.

United States General Accounting Office (GAO). Information Security Risk Assessment: Practices of Leading Organizations. Technical Report AIMD-00-33, GAO, 1999.

Sanjiv D. Vaidya and Priya Seetharaman. Collaborative Technology Use in Organizations: A Typology. In *Proceedings of the Americas Conference on Information System (AMCIS)*, 2005.

Hans van Vliet. *Software Engineering: Principles and Practice*. Wiley, West Sussex, England, 2008.

Kenneth R. van Wyk and Gary McGraw. Bridging the Gap between Software Development and Information Security. *IEEE Security and Privacy*, 3(5):75–79, 2005. ISSN 1540-7993. doi: 10.1109/MSP.2005.118.

Kami Vaniea, Lorrie Faith Cranor, Qun Ni, and Elisa Bertino. Access Control Policy Analysis and Visualization Tools for Security Professionals. In *USM '08: Workshop on Usable IT Security Management*, 2008a.

Kami Vaniea, Clare-Marie Karat, Joshua B. Gross, John Karat, and Carolyn Brodie. Evaluating assistance of natural language policy authoring. In *SOUPS '08: Proceedings of the 4th symposium on Usable privacy and security*, pages 65–73, New York, NY, USA, 2008b. ACM. ISBN 978-1-60558-276-4. doi: 10.1145/1408664.1408674.

Vijay Varadharajan, Chris Crall, and Joe Pato. Issues in the design of secure authorization service for distributed applications. In *Global Telecommunications Conference (GLOBECOM 98)*. IEEE Computer Society, 1998. doi: 10.1109/GLOCOM.1998.776857.

Tine Verhanneman, Frank Piessens, Bart De Win, and Wouter Joosen. Requirements traceability to support evolution of access control. In *SESS '05: Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications*, pages 1–7, New York, NY, USA, 2005. ACM. ISBN 1-59593-114-7. doi: 10.1145/1083200.1083212.

John Viega. Building security requirements with CLASP. In *SESS '05: Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications*, pages 1–7, New York, NY, USA, 2005. ACM. ISBN 1-59593-114-7. doi: 10.1145/1083200.1083207.

Stephen Viller and Ian Sommerville. Social analysis in the requirements engineering process: from ethnography to method. In *Requirements Engineering, 1999. Proceedings. IEEE International Symposium on*, pages 6–13, 1999. doi: 10.1109/ISRE.1999.777980.

Anneliese von Mayrhauser and A. Marie Vans. Program comprehension during software maintenance and evolution. *Computer*, 28(8):44–55, aug. 1995. ISSN 0018-9162. doi: 10.1109/2.402076.

Victor H. Vroom. Leadership and the decision-making process. *Organizational Dynamics*, 28(4):82–94, 2000. ISSN 0090-2616. doi: 10.1016/S0090-2616(00)00003-6.

John A. Wagner III. and Richard Z. Gooding. Shared Influence and Organizational Behavior: A Meta-Analysis of Situational Variables Expected to Moderate Participation-Outcome Relationships. *The Academy of Management Journal*, 30(3), Sep 1987.

Huaiqing Wang and Chen Wang. Taxonomy of security considerations and software quality. *Commun. ACM*, 46(6): 75–78, 2003. ISSN 0001-0782. doi: 10.1145/777313.777315.

Rick Wash. Folk models of home computer security. In *SOUPS '10: Proceedings of the 6th Symposium on Usable Privacy and Security*, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0264-7. doi: 10.1145/1837110.1837125.

Jaana Wäyrynen, Marine Bodén, and Gustav Boström. Security Engineering and eXtreme Programming: An Impossible Marriage? In Carmen Zannier, Hakan Erdogmus, and Lowell Lindstrom, editors, *XP/Agile Universe*, volume 3134 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2004. ISBN 3-540-22839-X.

Max Weber. *The Theory of Social and Economic Organization*. Simon and Schuster, 1947.

Max Weber. *Economy and society: An outline of interpretive sociology*. Bedminster Press, New York, NY, USA, 1968.

Carol A. Wellington, Thomas Briggs, and C. Dudley Girard. Comparison of student experiences with plan-driven and agile methodologies. In *FIE '05: Proceedings 35th Annual Conference on Frontiers in Education*, 2005. doi: 10.1109/FIE.2005.1611951.

Ryan West. The psychology of security. *Commun. ACM*, 51:34–40, April 2008. ISSN 0001-0782. doi: 10.1145/1330311.1330320.

Tara Whalen, Diana K. Smetters, and Elizabeth F. Churchill. User experiences with sharing and access control. In *CHI '06 extended abstracts on Human factors in computing systems*, pages 1517–1522, New York, NY, USA, 2006. ACM. ISBN 1-59593-298-4. doi: 10.1145/1125451.1125729.

Whitehat Security. Which Web Programming Languages are Most Secure? Online, retrieved 5 Aug 2010, 2010. URL `http://www.whitehatsec.com/home/resource/stats.html`.

Alma Whitten. *Making Security Usable*. PhD thesis, CMU, 2004. CMU-CS-04-135.

Alma Whitten and J. D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *8th USENIX Security Symposium*, August 1999.

Christopher Wickens, Arthur Kramer, Linda Vanasse, and Emanuel Donchin. Performance of concurrent tasks: a psychophysiological analysis of the reciprocity of information-processing resources. *Science*, 221(4615): 1080–1082, Sep 1983. doi: 10.1126/science.6879207.

Christopher D. Wickens and C. Melody Carswell. Information Processing. In *Handbook of Human Factors and Ergonomics* Salvendy (2006).

Robert Willison. Understanding the perpetration of employee computer crime in the organisational context. *Information and Organization*, 16(4):304–324, 2006. ISSN 1471-7727.

Robert Willison and James Backhouse. Opportunities for computer crime: considering systems risk from a criminological perspective. *European Journal*, 15(4):403–414, 2006. doi: 10.1057/palgrave.ejis.3000592.

Brad Wood. An insider threat model for adversary simulation. In Robert H. Anderson, Thomas Bozek, Tom Longstaff, Wayne Meitzler, Michael Skroch, and Ken Van Wyk, editors, *Research on Mitigating the Insider Threat to Information Systems #2*, Santa Monica, CA, 2000. RAND.

Avishai Wool. A quantitative study of firewall configuration errors. *Computer*, 37(6):62–67, june 2004. ISSN 0018-9162. doi: 10.1109/MC.2004.2.

Haidong Xia and José Carlos Brustoloni. Hardening Web browsers against man-in-the-middle and eavesdropping attacks. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 489–498, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi: 10.1145/1060745.1060817.

George Yee. Recent Research in Secure Software. Technical Report NRC/ERB-1134, National Research Council of Canada, 2006.

Ka-Ping Yee. Guidelines and Strategies for Secure Interaction Design. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and usability: designing secure systems that people can use*. O'Reilly, 2005.

Robert K. Yin. *Case Study Research: Design and Methods*. SAGE, 4th edition, 2009.

Eric Yu and Luiz Marcio Cysneiros. Designing for Privacy and Other Competing Requirements. In *2nd Symposium on Requirements Engineering for Information Security (SREIS 02)*, Raleigh, North Carolina, October 2002.

Marvin V. Zelkowitz. Resource utilization during software development. *J. Syst. Softw.*, 8(4):331–336, 1988. ISSN 0164-1212. doi: 10.1016/0164-1212(88)90016-7.

Ping Zhang and Na Li. An assessment of human-computer interaction research in management information systems: topics and methods. *Computers in Human Behavior*, 20(2):125–147, 2004. ISSN 0747-5632. doi: 10.1016/j.chb.2003.10.011. The Compass of Human-Computer Interaction.

Ping Zhang, Ben Shneiderman, and Dennis F. Galletta. *Human-computer interaction and management information systems: foundations*. M. E. Sharpe, Inc., Armonk, NY, USA, 2006.

Xinwen Zhang, Sejong Oh, and Ravi Sandhu. PBDM: a flexible delegation model in RBAC. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 149–157, New York, NY, USA, 2003. ACM. ISBN 1-58113-681-1. doi: 10.1145/775412.775431.

Xia Zhao and M. Eric Johnson. Access flexibility with escalation and audit. In *WISE '08: Twentieth Workshop on Information Systems and Economics*, 2008.

Xia Zhao and M. Eric Johnson. The Value of Escalation and Incentives in Managing Information Access. In *Managing Information Risk and the Economics of Security*, pages 165–177. Springer-Verlag New York, Inc., 2009. doi: 10.1007/978-0-387-09762-6_8.

George Kingsley Zipf. *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley, Cambridge Mass., 1949.

Mary Ellen Zurko. User-Centered Security: Stepping Up to the Grand Challenge. In *ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, pages 187–202. IEEE Computer Society, 2005. ISBN 0-7695-2461-3. doi: 10.1109/CSAC.2005.60.

Mary Ellen Zurko and Richard T. Simon. User-centered security. In *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*, pages 27–33, New York, NY, USA, 1996. ACM. ISBN 0-89791-944-0. doi: 10.1145/304851.304859.

Mary Ellen Zurko, Rich Simon, and Tom Sanfilippo. A User-Centered, Modular Authorization Service Built on an RBAC Foundation. In *S&P '99: Proceedings of the 1999 IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA, 1999. IEEE Computer Society. doi: 10.1109/SECPRI.1999.766718.

List of Figures

List of Tables

A. Security in plan-driven software development

Most research regarding security in plan-driven software development has been published on secure software processes and models, followed by approaches on security requirements engineering (Yee, 2006). We will briefly discuss security in plan-driven development before turning to the achieving secure software in agile development in the following section. First, we introduce common plan-driven development models as a backdrop of what secure development has to apply to. Then, we discuss common secure development methods and security practices in plan-driven development.

A.1. Plan-driven software development

Traditionally, software development has been divided into a set of development activities that structure the development process. Although the process models in software development differ, most development models distinguish the following activities:

- *Requirements engineering:* In the requirements elicitation activity, engineers specify the behavior of the system in requirements. The requirements and their priorities are documented in the requirements-specification document. Typically, non-functional requirements, such as performance requirements (Chung and Leite, 2009), are distinguished from functional requirements that describe the functionality of the system.
- *Modeling and design:* Architects design a software architecture that fulfills the requirements. Typically, the platform is chosen, and software components and similar structural artifacts are defined at this stage.
- *Implementation:* Developers implement the program flow in source code that reflects the software architecture and fulfills the requirements specification.
- *Assurance:* The program is verified and validated against the requirements on whether it delivers the functionality and adheres to the non-functional requirements as expected. Assurance activities may include testing, reviews, and external auditing of the program code.
- *Integration and operation:* The program is configured, deployed, maintained, and operated in the production environment. The configuration is managed and defects are removed as part of maintenance.

In the *waterfall* development model, the development activities are passed through as consecutive phases (Royce, 1970). Thus, developers first define a precise plan of the user needs and requirements so that subsequent phases, such as design and implementation, can produce the intended results. Derivations from the successive execution of the development phases, for example, to refine the requirements, is seen as an unintended effect. At the end of each phase, a verification and a validation of the result is conducted, assuring that the result is as planned and the result is as intended by the customer, respectively. In contrast, empirical data from case-study projects implementing waterfall processes show, on average only half of the design work in the design phase and as much as 16% after the implementation phase (Zelkowitz, 1988). The *V-model* is an adaptation of the waterfall model and focuses to a larger extent on validation and verification (van Vliet, 2008). The core phases are similar to the waterfall model and include requirements engineering, global design, detailed design, and coding. The main difference is that every phase is linked to a corresponding assurance phase, aligned in a "V" shape.

When it became evident that waterfall and similar models did not deliver the required reliability, more formal development models were proposed. *Cleanroom software engineering* uses formal methods and statistical testing to improve the software quality (Mills et al., 1987). Since no automatic software verification was available at the time of the process development in the 1980s, the development team conducted manual verification of specification through team review.

On the other hand, the practice of developing sophisticated software architectures led to processes that were centered around the architectures. The *Rational Unified Process* (RUP) is a very elaborate and complex process, with heavy tool support including a broad application of UML modeling (van Vliet, 2008). RUP is similar to agile development in that the plan may be continually adapted and users participate through use cases. In contrast to agile principles, RUP is a very document-centric and well-defined processes. The RUP is structured in two dimensions. First, several workflows, such as business modeling, implementation, and deployment run in parallel. Second, each workflow passes through the phases inception, elaboration, construction, and transition with multiple iterations within the phases.

In the 1990s, the focus shifted from achieving reliability through implementing predefined processes to the management of the reliability of processes and development teams. To systematically improve development processes, process management models were developed, for example, the *Capability Maturity Models* (CMM) (van Vliet, 2008). The CMM define five levels of maturity of the process management in organizations. Beginning at the lowest level, each level introduces additional measures of process reliability. For level 2, for instance, the focus lies on the repeatability of processes, requiring that the organization conducts, amongst others, project planning and requirements management. In higher levels, it is then required to conduct quantitative process management and continuous process improvement.

A.2. Secure software development methods

As noted above, the interconnectedness and complexity increased the likelihood of vulnerabilities and attacks, and the increased importance of systems their severity. One reaction was to improve the software development methods and processes. A prominent example of mitigating these threats in the process was Microsoft's announcement of "Trustworthy Computing" in 2002, including the development of a security SDL. While high-reliability development methods had been in use for safety-critical systems, now, practices were deeply integrated into the processes that explicitly aimed at security issues, such as threat modeling. Within the range of security SDL, methods still differ in the degree of focus on reliability and defects reduction in contrast to addressing specific security challenges (Goertzel et al., 2007).

There are several surveys of security SDL, either aiming to provide a comprehensive state-of-the-art report as in the IATAC Software Security Assurance report (Goertzel et al., 2007) or comparing processes (Davis, 2005) and practices (Davis et al., 2004). According to Davis (2005), the focus areas of security SDL are:

- Security engineering, including the elicitation of security requirements, secure design and implementation, and security testing,
- Security assurance through external reviews, verification, and validation,

- Organizational and project security management, providing policies, roles, and planning on an organization-wide and project level,
- Security risk management.

Security SDL can be roughly categorized according to their weight. On the one end of the spectrum, light-weight methods provide practices as enhancements to existing processes. On the other end, elaborate methods define the entire development process in detail. Two methods that claim to fall into the category of light-weight enhancements to existing processes are McGraw's *Seven Touchpoints of Software Security* (McGraw, 2006) and OWASP's *Comprehensive, Lightweight Application Security Process* (CLASP, OWASP). The Touchpoints include a risk management framework and are particularly tailorable to development processes already in use. The CLASP is also targeted at enhancing existing processes with light-weight security practices, but provides a broader set of guidance for the individual roles in development contexts, including a description of 24 activities, seven best practices, and a list of typical vulnerabilities.

The Microsoft *Trustworthy Computing Secure Development Lifecycle* (Microsoft SDL) is a more integrated approach to secure software development, based on a waterfall model, which is heavily enhanced with security activities. The SDL draws from four high-level principles: secure by design (design, implementation), secure by default for adequate configuration, secure deployment (tools, updates, documentation), and communication (e.g. with users on vulnerabilities). Microsoft reported a reduction of security bulletins in two case studies for subsequent releases after implementing the SDL (Lipner, 2004). De Win et al. (2009) compared Microsoft's SDL, CLASP, and the Touchpoints from each method's activities and process-wise on how the method affects the process. They noted that all three development processes have undergone validation in industry projects and cover a broad spectrum of the development lifecycle. They criticized that the methods often do not provide the methodology on how to implement the practices and lack thorough cost and usability trade-offs.

Similar to the Microsoft SDL, the *Team Software Process for Secure Software Development* (TSP-Secure) is an integrated approach and builds upon the Team Software Process, for which a reduction of defects in case studies by a factor of 10 to 100 has been reported (Davis and Mullaney, 2003). TSP-Secure introduces security practices and roles, and multiple defect removal points (filters) between phases (Davis, 2005). *Correctness by Construction* also originates in the reliable software development (Hall and Chapman, 2002), but is considerably more rigorous in its approach. The method heavily relies on formal methods and verification to achieve low defect rates. When applied to security-critical development, system states are categorized by their impact on the system's security and critical states accordingly verified more rigorously.

In contrast to the specific descriptions of methods, such as the Microsoft SDL or TSP-Secure, the aforementioned CMM focuses on the process management to improve the software quality through organizational, project management, and assurance measures. To cover the specific security risks apart from defect-caused vulnerabilities, CMM require additional security engineering and security risk management to adequately mitigate malicious intent, for instance malicious code introduced by developers (Davis, 2005). Several models enhance CMM with security practices. One example is the *Systems Security Engineering – Capability Maturity Model* (SSE–CMM, ISSEA, 2003; Menk, 1996), which has also been published as ISO/IEC 21827. SSE–CMM does not define a specific process, but rather extends CMM and proposes security practices. The additional "Base Practices" are concerned with security best-practices for development and organization.

Apart from secure development methods, assurance frameworks were developed to guarantee specific levels of security in software products. The most commonly implemented framework is *Common Criteria* (CC, ISO/IEC 15408-1:2009, 2009). For CC evaluations, organizations that plan to acquire evaluated systems develop a Protection Profile (PP) that describes the security requirements.

The requirements are formulated in an implementation-independent way, both for functional and non-functional (assurance) requirements. A system developer then develops a Security Target (ST), an implementation-specific security requirements document that is evaluated to comply with the PP. Lastly, the system developer provides the product (Target of Evaluation, TOE), which is evaluated against the ST. There are several evaluation levels that differ in the rigor of the developer's implementation practice, including testing, reviews and formal verification (Davis, 2005).

A.3. Security practices

Numerous practices have been proposed to improve the security of the software. We categorize the practices by the development activities that the practices primarily relate to:

A.3.1. Pre-process practices

Before the development process begins, security practices for the project inception may be necessary. The Microsoft SDL advises to check the applicability of the security development method and conduct general tool and personnel organization, for example, assigning a suitable security adviser (Lipner, 2004). CLASP proposes to identify security metrics at this point. Moreover, if not yet existing, the organizational policy on project-independent security requirements should be defined. The policy formulates global security requirements, which are then evaluated for their applicability to the current project (OWASP).

A.3.2. Process-accompanying practices

A number of practices need to be constantly applied throughout the SDL and thus accompany the process. One process area of this kind is education, awareness, and motivation of team members. The Microsoft SDL, for instance, requires baseline education for every team member to increase the awareness and teach security-engineering basics not only to developers, but also to management staff (Lipner, 2004). Security knowledge management frameworks help to organize and share the acquired information among team members and across project boundaries (McGraw, 2006). CLASP suggests a proactive sharing of security artifacts with all developers to improve awareness. To further increase motivation, CLASP advocates the institutionalized accountability for defects and rewards for prevented issues (OWASP).

The risks that the product and process are prone to constantly change throughout the development process. New attack vectors are discovered and need to be mitigated, but also the development context changes over the course of the project. McGraw (2006) developed a risk-management framework that is deeply integrated into the SDL to accommodate evolving risks. McGraw's Touchpoints also include the continuous execution of an improvement program, primarily targeting the measurement strategy and the development process (McGraw, 2006). Typical metrics are, for example, the attack surface, reported defect rates, or security test coverage (De Win et al., 2009). Goertzel et al. (2007) list eleven metrics with very different approaches and target audiences.

A.3.3. Requirements engineering

From a security perspective, the requirements engineering activity poses several challenges. Firstly, software developers generally consider security requirements as quality attributes, formulated as non-functional requirements, or as negative ones, describing events that should not occur. The problem with negative and non-functional requirements is that these types of requirements are difficult to implement and test. One approach is to not document non-functional and negative requirements at all, but turn them into functional requirements (Goertzel et al., 2007). Similar challenges originate from the different levels of abstraction between high-level policies and functional requirements (Devanbu and Stubblebine, 2000). Conversely, there is the tendency to describe security requirements through mechanisms, unnecessarily anticipating design and implementation decisions (Firesmith, 2003). Moreover, there is the secondary goal problem with security: Software developers strive to implement functional requirements; secondary goals, such as security aspects, can be problematic, particularly with a low customer security-awareness (Tondel et al., 2008).

To meet the challenges that the engineering of security requirements poses, a wealth of methods and activities have been proposed. In a survey of different elicitation methods, Tondel et al. (2008) identified eight task categories, including defining central concepts, high-level objectives, and identifying threats and assets. Security requirements engineering methods range from very detailed, complex, and broad approaches, such as SQUARE (Mead and Stehney, 2005), to very simple, misuse/threat-only approaches as in McGraw's Touchpoints (van Wyk and McGraw, 2005). SQUARE, the Security Quality Requirements Engineering, provides guidance for eliciting, categorizing, and prioritizing security requirements (Mead and Stehney, 2005).

One starting point for the elicitation of security requirements are business requirements, laws and regulation, and contractual obligations that need to be fulfilled by the software product, as proposed, for instance, by the Touchpoints (McGraw, 2006). To turn the high-level and non-functional security requirements into functional requirements, practitioners need to identify the threats that should be mitigated. There are two distinct approaches: Either explore the threats starting from known attack patterns or starting from the resources and assets to be protected, thus with the attacker or user perspective, respectively. Attack patterns have the problem that they are limited to the attacks in the current knowledge base of threats and miss further attacks (Goertzel et al., 2007). More generally, threat modeling approaches can be conceptual or technical, functionality- or attack-driven, and systematic or pragmatic (De Win et al., 2009). Threat modeling should be conducted as a requirements and as a design activity because the architecture may introduce additional attack vectors not foreseeable from functional requirements (OWASP; Viega, 2005).

Several pragmatic methods have been proposed to elicit threats. As the negative counterpart to use-case definition, abuse cases or abuser stories define activities that a malicious user must not be able to conduct (McDermott and Fox, 1999; Sindre and Opdahl, 2005). The Touchpoints describe threat modeling with abuse cases, defining threat agents, anti-requirements of what might go wrong and the attack pattern of how an threat agent causes the anti-requirement (McGraw, 2006). The SDL threat modeling method version 1, STRIDE has a very pronounced attacker perspective of specific attack vectors (Shostack, 2008). Although STRIDE's very pragmatic approach from typical attacks might be very accessible to developers, it might be too narrowly defined because of its explicit limitation on specific attack vectors (Hernan et al., 2006). Another method to analyze threats from the attack perspective are attack trees, attack paths modeled in a tree structure (Moore et al., 2001). Attack trees are based on the observation that attacks often occur in several steps and have several subtasks that need to be completed to succeed. Beginning from the threat target, for each node, the subgoals (either AND or OR decompositions) are collected recursively, resulting in a tree structure. A path from a leaf subgoal to the root is an attack path.

Threat mitigation measures turn negative requirements into functional requirements. Haley et al. (2004) propose problem frames with threat information as constraints that are subsequently transformed into functional requirements. Also, specific processes have been defined to manage non-functional requirements and relate non-functional to functional requirements (Chung and Nixon, 1995), or refine non-functional requirements into subgoals until implementable (Yu and Cysneiros, 2002; Mouratidis and Giorgini, 2007).

The number of security requirements can quickly explode for non-trivial systems, requiring a prioritization according to the risks. Risk assessment usually includes the aspects of threat and vulnerability assessment as well as the impact analysis. For requirement risk assessment, the Microsoft SDL proposes the DREAD model that takes the damage potential, the reproducibility, the exploitability, the number of affected users, and the discoverability into account (Howard and Leblanc, 2002). Threats are ranked according to each of the categories, resulting in a relative risk score from the mean of each of the threats' rankings. Another requirements trade-off may be necessary between security and usability (Braz et al., 2007).

To validate the completeness of and the lack of conflicts between security requirements, the requirements can be analyzed, for example, informally in externally audits (OWASP). More formally, Liu et al. (2003) analyze requirements as relationships between actors, stakeholders, users, and aggressors to analyze security requirements. Correctness by construction uses the programing language Z for this purpose (Hall and Chapman, 2002). Having a formal model of the requirements may enable an analysis of the security requirements with formal methods, for example, the verification that a high-level policy is complied with.

A.3.4. Architecture and design

In the architecture and design activity, the individual measures to address the security requirements are decided upon and result in the security architecture. While plan-driven development traditionally distinguishes between the requirement and design phases, the ties between these phases are very close for the security architecture. In addition to mitigating threats, the security architecture the architecture often at the same time introduces additional attack surface, and thus calls for further functional security requirements. With insufficient security requirements engineering, a typical effect is that the security architecture is only added as an afterthought, potentially resulting in suboptimal designs (Devanbu and Stubblebine, 2000).

For designing the security architecture, Saltzer and Schroeder (1975) compiled a list of security design principles. The list is still largely applicable and includes principles, such as least privilege and economy of mechanism (Beznosov and Kruchten, 2004). A broader list of 33 practices is provided in the NIST publication on engineering principles for security (Stoneburner et al., 2004). The problem with design principles is that the principles are rather abstract and, thus, difficult to apply in practice. More pragmatic are security design patterns that allow practitioners to solve security challenges in a structured manner by employing best-practice approaches. Schumacher and Roedig (2001) propose a security-pattern template and give examples for VPN security.

Software architectures can be modeled in graphical modeling languages, such as UML. Designers can benefit from integrating security constraints and measures: SecureUML focuses on access control and integrates role-based policies into the model-driven development (Lodderstedt et al., 2002). UMLsec takes a broader approach and introduces UML stereotypes that impose constraints on the UML model (Jürjens, 2002).

An analysis of the architecture may strengthen the confidence in the design. Manual, misuse-case- or risk-based, analyses are often conducted through architecture reviews (Pauli and Xu, 2005; McGraw, 2006). A formal model of the architecture allows for automatic or semi-automatic analyses using formal verification: For example, the formal language Z provides for logic-based modeling, and CSP for the formalization of process models. For design-by-contract modeling in UML invariants can be specified in OCL (Goertzel et al., 2007). The design may then be verified against the requirements.

A.3.5. Implementation

Defects in the source code of software are a common cause of security vulnerabilities (Software Engineering Institute, 2002). As one of the aforementioned unintentional errors, developers may forget to properly validate input. As an intentional error, not following the architecture may lead to vulnerabilities, for example, when developers circumvent protective layers (Sohr and Berger, 2010). In implementation the focus thus lies on producing reliable (defect-free) and correct (adhering to the specified requirements and design) software. On the lowest layer of implementation lies the syntax and semantics of the program code. There have been conflicting findings on whether there are differences in the proneness for defects for different languages, for example, from empirically evaluating the rate of defects for different platforms (Whitehat Security, 2010) or analyzing the "edit distance" to vulnerabilities (Chinchani et al., 2003). More accepted is the fact that adhering to secure programming strategies improves the reliability of software. A wealth of strategy lists have been published, either agnostic (Peine, 2005) or specific to the platform or language (Seacord, 2005). Agnostic principles include defensive programming, defense in depth (Stallings and Brown, 2008), and assuming a hostile environment (Jones and Rastogi, 2004).

A.3.6. Assurance

Concurrently or subsequent to implementation, analyses and tests can assure that the software behaves as specified: securely and deterministically. Analyses are either performed on the source code, statically, or on executed code, dynamically. Static and dynamic analyses may be conducted manually, semi- or automatically. A typical method for manual static analysis is source-code review, as mandatory in the Microsoft SDL (Lipner, 2004). Static-analysis tools abstractly execute the source code to discover defects (semi-)automatically (Chess and West, 2007). While early static analysis focused on syntax and dangerous constructs through lexical analysis, current tools detect defects in memory allocation and malicious information flows (McGraw, 2006). A further form of static analysis is the verification with formal methods, proving that certain constraints hold (Mills et al., 1987; Hall and Chapman, 2002).

For dynamic analysis (testing), program code is executed and the results are compared with expectations. Manual security-focused tests complement manual functionality tests that target the overall software quality, applying typical attack patterns (Allen et al., 2008). However, not all attack vectors are testable because of resource constraints, so that risk-based testing is often suggested, which focuses tests on the most critical parts (McGraw, 2006). Manual security tests occur as controlled attacks as penetration tests, either with knowledge of the system internals (white box) or without (black box).

Automatic tests are employed in software development to continuously guarantee that the functionality works correctly. For security tests, the test cases may be enriched with explicit testing of the security mechanisms. Other automatic tests use fuzzers to generate input data to automate black-box testing and discover input handling defects (Del Grosso et al., 2005).

A.3.7. Operations and maintenance

Security activities in the SDL do not end when the software is released or deployed. First of all, it is important to guarantee that the software product is deployed unchanged to prevent the addition of malicious functionality (OWASP). After deployment, the operation environment needs to be kept secure and security incidents need to be handled. Ideally, the developers provide the assumptions on the operational environment that guided the architectural decisions (Lipner, 2004) as well as on secure configuration, such as on auditing and backups (Stallings and Brown, 2008; Fraser, 1997).

For security incidents, a response plan should be prepared, including communication policies to customers and security update procedures (Lipner, 2004). An incident management process (Killcrece, 2008) should cover prevention (following vulnerability reports and technological developments), detection (e.g. log file analysis), the triage process (prioritize reports), and response (containment of the damage, vulnerability fixing, updates). Often, dedicated computer security incident response teams (CSIRT) are formed in organizations to systematically handle the incidents. It is important that lessons learned from the incidents flow back into the development process to prevent recurring vulnerabilities.

B. Agile development

While the Agile Manifesto stems from 2001, one important aspect, the iterative and incremental development to address imprecise and changing requirements, has a long history in software development. It has been employed since the 1950s with varying parameters (Larman and Basili, 2003), including combinations of up-front specification with time-boxed, incremental development. The common goal was to avoid the problems of single-pass, document-driven development with "big-bang" deployment. While the plan-based, waterfall development process model was still seen as the "ideal model," it was already accepted that the waterfall method was not suitable in many projects with changing requirements, such as the NASA Shuttle program. Since the 1980s, several development methods such as the Spiral Model have been proposed that strongly emphasize incremental development. According to Nerur and Balijepally (2007), such paradigm shifts from plan-based to adaptive are not specific to development and can be observed in other disciplines, such as management, as well. In the late 1980s and the 1990s, the first of today's agile methods were formed, including the Dynamic Software Development Method (DSDM) as one of the earliest from the model-driven development community, but also Scrum and Extreme Programming (XP). In contrast to earlier iterative and incremental approaches, the agile methods take a more comprehensive approach to provide for dynamic contexts: the agile values.

B.1. Empirical findings on agile development

Since the spreading of agile development methods and practices in industry, empirical research has been conducted to back the claims of agile practitioners and counter the argument of missing scientific evidence. While the absolute number of empirical studies on agile development is constantly increasing, the quality of the evidence is still challenged (Dybå and Dingsøyr, 2009). Most existing studies are surveys or case studies; surveys rather measure the perception of the development method, case studies are difficult generalize. When experiments are conducted, these are typically undertaken in academic contexts with student participants since it is difficult to control the variables and have control groups in industry context. Despite these shortcomings, the existing literature does give indications on the consequences of employing agile development.

The literature review in the following originates from two sources. Firstly, in an oft-cited meta study, Dybå and Dingsøyr (2008) compiled the evidence from studies on agile development that had been published until 2005. From the 1,900 scientific publications on agile development, they found 36 studies with empirical evidence of acceptable rigor. The second source is a systematic literature research for agile development studies published between 2005 and 2010 through searches at IEEE Xplore, ACM Digital Library, Springer Link, Science Direct and Wiley Interscience on the search terms "agile development empirical". Studies were selected based on whether original empirical research was presented, the acceptability of the scientific methods and whether the results concerned the effects of agile development. The results are grouped by the effect in the following:

B.1.1. Software quality and development productivity

The effects of agile development on software quality and productivity have been studied with several empirical methods, including surveys, case studies and experiments. The perception of agile methods of students was investigated by Melnik and Maurer in two publications. Students from very distinct backgrounds broadly agreed that agile development improved code quality and productivity (Melnik and Maurer, 2002, 2005). Similarly, professionals also perceive agile development to increase quality and productivity as shown in a large survey of developers from the software industry (Parsons et al., 2007). In a survey at Microsoft, Begel and Nagappan (2007) found that the practitioners thought that increased quality, quick releases, and faster responses are among the most important benefits of agile development.

Other findings have been derived from case studies. In two industry case-studies on Lean software development, quality problems were found in one of the cases because of missing social skills resulting in error-prone software (Middleton, 2001). In contrast, a migration to agile development resulted in above-average quality and average to above-average productivity when compared to industry benchmarks (65% and 35% improvement in pre- and post-release defect-rates, respectively: Layman et al., 2006, 2004). Another industry case-study showed a 13% reduction of reported defects in agile teams when compared against a baseline team (Ilieva et al., 2004). In a case study at Ericsson, a higher product quality with less fault slips, a reduced rise of maintenance costs, and improved testing was found after the migration from plan-driven to agile development. Although there was a higher release frequency and a reduction of "waste," no productivity gains were observed in this study (Petersen and Wohlin, 2010). At Intel, a combination of Scrum and XP worked best and cherry-picking practices resulted in reduced defect density and ahead-of-schedule delivery (Fitzgerald et al., 2006). Further findings were reported on specific agile practices. A qualitative field study indicated that pair programming and test-first development, two important agile practices, improved the software quality (Tessem, 2003).

Formal experiments with students in academic contexts have resulted in mixed findings. In one experiment, two teams of 16 and 17 students worked in plan-based or agile development. The agile team achieved consistently better code metrics, but developed a less consistent user interface (Wellington et al., 2005). In a larger experiment with 93 students, the students were divided into agile and plan-based approaches with 20 teams in total, of which each team was assigned to one of four real customers. This experiment showed no difference in internal or external quality (Macias et al., 2003).

B.1.2. Motivation and satisfaction

Motivation and satisfaction have several indirect effects, and are thus an important indicator of software development success (Baddoo et al., 2006). Motivated developers achieve higher productivity, remain longer in the organization, and take less sick-leave (Beecham et al., 2008). For agile development, a large-scale survey of professional agile developers showed that the satisfaction is perceived to be higher (Parsons et al., 2007). Similarly, in a Web-based survey of 55 professional agile developers, most developers approve their companies' development process, report greater satisfaction, and see XP practices favorable (Mannaro et al., 2004).

In case studies, the results are similar. In a Canadian organization that shifted from plan-driven to agile development, developers stated that agile development is a "pleasant experience" in contrast to the unpleasant previous development model (Bahli and Zeid, 2005). Along the same lines, a multi-case study at three large independent software vendors showed increased motivation from agile practices such as the empowerment of the engineering (Karlström and Runeson, 2005). In two case studies on the characteristics of successful agile development teams, several aspects are seen as motivating: higher responsibility, preserving the quality of life, and increased respect and trust. Also, the team receives motivating feedback from customer satisfaction and encouragement from each other (Robinson and Sharp, 2004).

In contrast, no positive motivational effect was found in an industrial pilot case study with only a limited number of agile practices (Auvinen et al., 2006). Moreover, apart from factors that positively affect the motivation in observed mature XP teams (self-organization, regulating information load, communication-rich team), de-motivating aspects were found (career aspects with team effort) in another industry case-study (Beecham et al., 2007). Also, Ilieva et al. (2004) found in a case study that specific practices have negative side-effects, for example, being exhausting in case of pair programming.

B.1.3. Internal and external communication

Software development depends on knowledge to create products that deliver added value. Knowledge is passed between stakeholders, such as customers and developers, through communication (Korkala et al., 2006). When studying communication in software development, one may distinguish between internal communication between software developers in the development team and external communication with external stakeholders, such as customers. In agile development, verbal communication is the primary form of knowledge passing and program code is practically "talked into existence" (Mackenzie and Monk, 2004). At Microsoft, developers perceived the improved communication as the most important benefit from agile development (Begel and Nagappan, 2007). Communication has also been an important aspect in case studies on agile development. In a case study at F-Secure, positive effects on internal communication, both informal and formal, were observed, but also that, if the proper communication media are lacking, such as open office space, negative effects can occur (Pikkarainen et al., 2008). At Ericsson, improved communication was also an important effect of agile development, mostly credited to the concentration of developers in one place and the reduction of formalized documentation (Petersen and Wohlin, 2010).

At F-Secure, agile practices improved the common understanding of short-term goals between the customer and the team. The communication success is very dependent on the specific communication skills with less formalized channels (Pikkarainen et al., 2008). In three case studies, a reduction in defects was linked to the more informative communication media (Korkala et al., 2006). Agile development gives the developers a better high-level view of the customer's goals (Mann and Maurer, 2005). Similarly, it was found that the communication improved the requirements definitions (Dagnino et al., 2004). In contrast, in a study from the customer perspective, the customer stakeholders perceived that there was no change in the amount of communication after the introduction of agile methods (Svensson and Höst, 2005b).

B.1.4. Conclusion

Studies on agile development methods and specific agile practices indicate increased software quality and productivity with no difference in the worst case as well as higher job satisfaction. In a large-scale ex-post-facto study, Misra et al. (2009) identified important factors for the successful adoption of agile development, including customer satisfaction, customer collaboration and commitment, decision time, corporate culture, qualitative control, personal characteristics, societal culture, and training/learning. However, generally, the evidence remains rather weak (Dybå and Dingsøyr, 2008). Most studies are either surveys, showing primarily the developers' perceptions, or case studies that depend on the specific circumstances. Only very few of the studies are experiments and then only in artificial academic environments. Interestingly, we can observe that the results from surveys, representing the developer perception, are more positive than from case studies or experiments.

The overall corollary from reviewing the empirical findings thus is that we still need better studies with broader scopes of validity, particularly in professional contexts. Moreover, most studies focus on the programming practices. More empirical work is needed on management-oriented aspects and the core agile ideas (Dingsøyr et al., 2008). Because the field is still nascent, qualitative studies should currently be the primary target (Dybå and Dingsøyr, 2008).

B.2. A comparison of agile development methods

The definition of which software development methods actually are "agile" is unclear. According to van Vliet (2008), many methods incorporate agile aspects but are not "purely" agile. The prototyping development models and the Spiral model have similar incremental and iterative approaches. These models are agile in the sense of "working software" and "responding to change", but may also be implemented in a plan-driven way (van Vliet, 2008; Boehm, 1988). Rapid Application Development (RAD) also employs agile practices, such as time-boxing, incremental development, user involvement, and small development teams, but is a heavy-weight and very structured process (van Vliet, 2008).

In essence, each agile development method is a collection of practices, which are supported by values and principles. Kent Beck states for XP that the combination of the XP practices "amplify" each individual practice's effect (Beck and Andres, 2004). Concas et al. (2008) found that the consequences of abandoning practices for specific phases as measured in quality metrics were dramatically negative. Conversely, agile development proponents, such as Kent Beck, downplay the idea of "method purity", stressing the importance of the agile values, such as sustainability and producing value (Beck and Andres, 2004). In a highly-experienced development context, there have been positive results from augmenting standard processes with agile practices, leading to higher performance (Ramasubbu and Balan, 2009). At Microsoft, many teams use agile methods that are loosely based on Scrum (Begel and Nagappan, 2007). Similarly, Parsons et al. (2007) and Fitzgerald et al. (2006) found that combinations of XP and Scrum worked best, for example, with Scrum providing project management and progress monitoring, and XP additional practices.

Several agile development methods have been formulated since the late 1980s. To analyze the breadth in agile development, we compare popular methods – Scrum and Extreme Programming, and, less widely-spread, Crystal Clear, Lean development, the Dynamic Software Development Method (DSDM) and Feature-driven Development (FDD). The comparison should enrich the reasoning on agile development and its relation to security in software development (cf. Chapter 3). While neither comprehensive nor representational, the comparison covers a wide range of approaches to agile development and, thus, offers a good overview of the agile "ecosystem" (Highsmith, 2002). Each method is described briefly followed by a systematic comparison. Other methods, such as Kanban, which focuses on "pull" work assignments and on limiting the work-in-progress items to improve the development flow (Kniberg and Skarin, 2010), are still evolving and for that reason not included in the comparison here.

In the following, we discuss each of the studied development models briefly. The values, principles, and practices of the models are contrasted in Tables B.1 and B.2.

	Scrum	XP	Crystal Clear	Lean	DSDM	FDD
Project management				*Never push beyond limits*		
Process improvement	Sprint retrospectives		*Reflective improvement*			
Process model	Scrums, Sprints	Weekly, quarterly cycles	*Incremental delivery*	Start-up, steady-state, transition, renewal phase	Overlapping iterative phases	Five processes, two iterative
Delivery	Sprint result	Incremental deployment	Frequent delivery	*80% solution today; Complete, don't construct*	Frequent delivery	Developing by feature
Reporting	Burn down chart		*Progress by delivery*	*Product growth is feature growth*	15 work products	*Visibility of results*
Requirements	Sprint backlog	Stories	Staging		Several prototypes as documentation	Build feature list
Customer participation	Sprint planning meeting	*Real customer involvement*	Easy access to expert users	*Active customer participation*	*Active user involvement*	Domain walk-through
Planning	Product backlog	Slack	Release plan		Requirements are baselined at high level	Plan by feature
High-level goal	Sprint goal	Pay-per-use		*Satisfy the customer; Best value for money*	*Fitness for business purpose*	

Table B.1.: The principles (emphasized) and practices of agile development methods

	Scrum	XP	Crystal Clear	Lean	DSDM	FDD
Design and implementation		Incremental design; Single code base	Policy standards	*Everything is changeable; Needs determine technology; Minimalism; Domain, not point solutions*	Iterative, incremental development	Domain object modeling
Code ownership		Shared code				Class ownership
Code review		Pair programming	Two user viewings per release			Inspections
Testing		Test-first development	Automated tests		Testing throughout the lifecycle	Unit test
Building		Continuous integration; 10-minute build; Daily deployment	Frequent integration			Regular builds
Configuration management			Configuration management		All changes are reversible	Configuration management
Team and people	Self-organizing teams	*Whole team*; Team continuity	*Personal safety*	*Project is team effort*	*Collaborative, cooperative approach*	Feature teams
Work approach		*Energized work*	*Focus*		*Empowerment to make decisions*	
Communication	Daily stand-up meetings	Informative workspace; *Sit together*	*Osmotic communication*			

Table B.2.: The principles (emphasized) and practices of agile development methods (continued)

B.2.1. Scrum

Scrum implements "empirical process control" through feedback loops between development and process adaption (Schwaber and Beedle, 2001; Highsmith, 2002). Scrum emphasizes the roles and responsibilities in the process, and high-level practices for requirements management, and thus focuses on the project management perspective. The product owner (customer) is in charge of setting priorities, developers in the team create the product, and a Scrum master is responsible for an effective Scrum process. Central to the Scrum process are the iterating development phases of 30 days, called "sprint", each resulting in a deliverable product. The sprint is planned in a planning meeting and the product owner places work items into the sprint backlog, a list of features that need to be completed by the team within the sprint.

While a sprint is running, daily stand-up meetings, in which each team member states the current status, are the main institutionalized form of communication. Further informal communication among developers is encouraged. At the end of each sprint, a sprint review meeting is held to present the progress to the product owner. Moreover, the team conducts sprint retrospectives to identify potentials of improvement in the Scrum process as part of the feedback loop.

B.2.2. Extreme Programming

Extreme Programming (XP) is a bottom-up development method in that specific programming practices play a central role in defining the development process (Beck, 1999; Beck and Andres, 2004)[1]. The XP development process is similarly structured as the Scrum process, but the iterations are shorter with weekly cycles. In a planning game, the customer chooses the work, which is described in stories, for the subsequent iteration. In addition, XP proposes quarterly cycles for long-term, "big-picture" planning. In contrast to Scrum, XP implements more specific practices concerned with the project environment and the development, such as informative workspaces and pair programming.

B.2.3. Crystal Clear

Crystal Clear is part of the Crystal family of agile development methods. The Crystal methods should provide "barely sufficient" process guidance and, thus, offer different method weights for different project types (Cockburn, 2004; Highsmith, 2002). As a side effect, the Crystal family is an example of how different the actual implementations of agile processes are for specific development contexts in terms of team size and product criticality. Crystal Clear is the most liberal with a small number of characteristics and leaves the team room for individual practices. When compared to XP, for instance, with its strict approach to development practices, Crystal Clear should be easier to implement and can even serve as a backup if the introduction of XP fails. An interesting aspect that is explicitly formulated as a practice in Crystal is "personal safety": Participants need to trust each other and no-one should be hindered by fear of personal consequences of mistakes.

B.2.4. Lean Development

Lean Development is the adaption of lean-production principles as implemented by Toyota in manufacturing to software development (Poppendieck and Poppendieck, 2003) and has a pronounced management perspective (top-down). Accordingly, the first of four process phases, a start-up phase, is used to identify and reduce project risks. In a "steady state", development cycles (shorter than 60

[1]This analysis refers to the second edition of the XP book since most practices from the first edition are still present in the "primary practices" and "corollary practices" of the second edition.

days) integrate incremental analysis, design, tests, and integration. The last two phases, transition and renewal, are enacted to continue development after the customer accepted a product's delivery. In Lean Development, everything should be changeable, resulting in one principle to "build the application to be more tolerant" to changes (Highsmith, 2002). This principle is in contrast to the principle of choosing the simplest approach in XP. Similarly, a Lean-Development principle is "Domain, not point solutions", thus explicitly not targeting a very specific and simple solution, but broader ones that are often more complex.

B.2.5. Dynamic Software Development Method

The Dynamic Software Development Method (DSDM) originates in the Rapid Application Development (RAD) community and formalizes RAD practices (Highsmith, 2002; Stapleton, 2003). DSDM defines three interrelated development phases that are iteratively applied. In the functional modeling phase, requirements are defined by way of functional prototypes. The design/build phase produces design prototypes, which are them turned into the final product and deployed in the implementation phase. With 15 defined types of process deliveries and 11 roles, DSDM has a considerable process complexity. However, most deliveries may occur in form of prototypes, thus corresponding to the agile principle of early running code.

B.2.6. Feature-driven Development

Feature-driven development (FDD) combines model-driven development with agile practices (Palmer and Felsing, 2001). The FDD method is precisely and formally defined. The method is divided into 5 processes, of which three (developing an overall model, building a feature list, and planning features) are run once and two (designing a feature and building a feature) are run iteratively for every feature. The up-front processes are the main differences to the simpler agile methods like Scrum. While these processes should only take up a limited amount of project resources and are ongoing in case of later changes, the focus is on having a more precise plan and less need for rework and refactoring. In this respect, FDD is more plan-driven then the other agile methods (Highsmith, 2002). On the other hand, FDD is reported to be particularly well suited for scaling agile development to larger projects. Development roles are more differentiated, for example, including a chief-programmer role, responsible for assigning work items to individual programmers. Another difference between FDD and other agile methods is that, instead of a collective code ownership, as in XP, FDD requires individual class ownerships by developers. When compared to DSDM, the other analyzed agile method with a complex process model, FDD shows a stronger focus on modeling than prototyping (Highsmith, 2002).

B.2.7. Key characteristics of agile methods

The agile development methods evolved independently, prior to the publication of the Agile Manifesto, and differ in coverage and the approach to their specification, regarding principles, practices, roles, and concreteness (Abrahamsson et al., 2003). Due to this heterogeneousness agile methods were often described alongside each other as by Highsmith (2002), without a systematic comparison. Cohen et al. (2004b) target in their systematic comparison formal aspects, such as team sizes and iteration lengths, and which practice of each method supports which development phase. Abrahamsson et al. (2003), on the other hand, systematically compared which part of the SDL the agile methods cover and whether the processes are adaptable. Qumer and Henderson-Sellers (2008) take

Individuals and interactions over processes and tools	Process formalization, Averseness to change, Process overhead
Working software over comprehensive documentation	Process formalization, Averseness to change, Process overhead, Size/reliability
Customer collaboration over contract negotiation	Formalization, Size/reliability
Responding to change over following a plan	Process formalization, Averseness to change

Table B.3.: Direct interrelations between agile values and agility dimensions

a partly quantitative approach to the comparison of agile methods along dimensions of scope, agile features, agile values, and process characteristics. Quantitative metrics of agility are assigned to each development phase and practice according to "agile features", such as flexibility, speed, and "leanness". The sum of agility values as well as qualitative evaluations result in an overall degree-of-agility score. However, the equal weights of the difference aspects result in findings that are difficult to interpret.

In practice, the decision of which agile method to apply cannot be taken based on a one-dimensional score, but needs to compare the given context to the characteristics of the methods. Accordingly, the comparison given in Tables B.1 and B.2 aligns the concepts, principles, and practices as characteristics of each of the agile methods according to the topic. The low-level comparison in the tables reveals the focuses of the studied agile development methods by highlighting which areas are covered to what extent. The methodology of the analysis was to start off from the characteristics for each method and contrast these for the aspects of project management, requirements elicitation, design/implementation and team/human aspects. The characteristics are based on primary and secondary practices and principles, if given, as well es extracts from textual descriptions of the processes in primary sources.

To further systematically analyze the agile methods, we examine four major dimensions of agility for a high-level comparison. The four dimensions – process formalization, averseness to change, process overhead, project size/reliability – are strongly interrelated with the agile values as defined in the Agile Manifesto (Beck et al., 2001) as shown in Table B.3. In contrast to the model of Qumer and Henderson-Sellers (2008), which estimated the overall "degree of agility", even though interrelated as well, these dimensions offer a more differentiated picture of the continuum of agility. We discuss the rationale for the relative and qualitative findings on the methods for each of the dimensions in the following and present the results schematically in Figure B.1.

Process formalization

Depending on the method's background, each has a distinct approach on how formal and concrete the definitions of the processes and practices are – in contrast to a more liberal approach of teams adapting process and practices as needed in specific environments. High process formalization often increases the method's averseness to change (fixations from formalized procedures) and also entails higher process overhead (from prescribed activities), but may increase the process reliability (activities are prescribed) and its adequacy for large projects (more formalized interaction). Similarly, process formalization conflicts with agile values, for example, with the focus on individuals and (implicit) interactions, and on working software.

Lean development has a high-level perspective from management on practices and thus a low formalization. Scrum does not formalizes any explicit development practices and only simple process

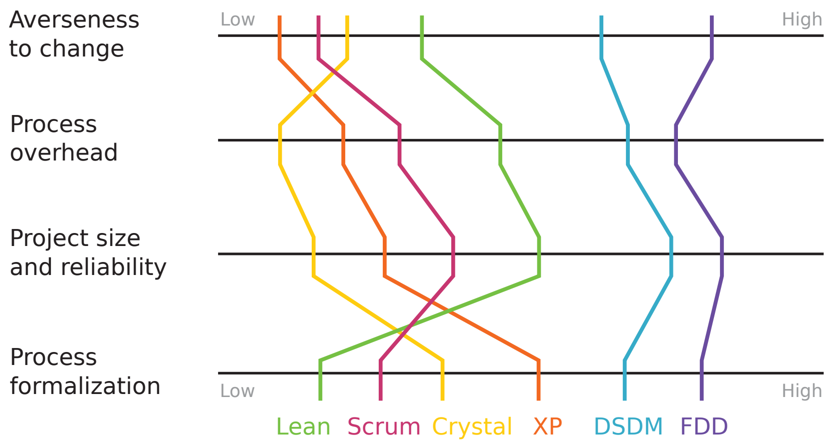


Figure B.1.: Characteristics of the compared agile methods

management practices. Crystal Clear even defines less process management than Scrum, but describes some development practices for testing, building, and tool support. XP, although not very detailed on the specific process, has a wealth of very specific practices for development and the project environment. In contrast, DSDM has very specific process descriptions, naming subprocesses and work products as well as some development practices, such as testing. FDD is very specific in terms of processes as well, but also formulates more guidance on development, concerning for example tool support, testing and code ownership.

Averseness to change

Depending on the process, methods may be more or less open to changes to an original plan and to the process itself. While larger teams may work more efficiently with less volatile requirements – since this improves parallel development –, changes to requirements are likely to occur during the development process if the resulting product should deliver added value. Scrum, XP and Crystal Clear are similar with respect to their openness to changes. XP has weekly iterations and quarterly high-level planning, which both allow modifications of the plan. Scrum advocates planning for sprints of 30 days, limiting changes to between the sprints. Crystal Clear describes longer iterations and a release plan. Lean development is more ambivalent with respect to the rigidness of requirements. On the one hand, changeability is part of the method philosophy (“everything is changeable”) and development is conducted in short, time-boxed iterations. On the other hand, there are up-front feasibility studies in the start-up process. In DSDM, high-level requirements are formulated early on and formalized in prototypes at several stages, but re-entry into previous phases are allowed. In FDD, an overall plan is formulated up-front to enable the definition of a high-level object model so that the main changes later on occur on feature-level and leave the object model unchanged.

Process overhead

A high process formalization may lead to more overhead when implementing and operating a method. Process complexity is dominated by the number and depth of hierarchy of different processes and phases that are enacted as part of the development method. Another aspect is the number of required practices and of roles that are defined as part of the method. Crystal Clear and XP advocate simple iterative development, with the addition of longer-term release plans or quarterly cycles, respectively. Scrum only increases the process complexity marginally by inserting simple inner-iteration loops, daily scrums. Both Scrum and Crystal Clear suggest reflective process improvements through retrospectives. Lean development has a slightly higher number of distinct processes with start-up, transition and renewal phases in addition to a steady-state phase that is similar to development iterations. In contrast, DSDM and FDD have more complex process models, comprised of multiple phases or processes with multiple steps each. DSDM has separate processes for functional modeling, designing and building, and implementation. FDD implements five processes, of which three are run once and two iteratively for each feature.

Appropriateness for large projects and high reliability

Larger projects typically require more defined communication processes since "osmotic" communication loses its efficacy (Highsmith, 2002). Similarly, high reliability often requires more defined processes to implement formal process requirements and traceability (Poppendieck, 2002). Some of the analyzed agile methods are explicitly tailored for small teams and medium reliability requirements. Crystal Clear has been specifically targeted only at small teams and medium reliability requirements, although other methods from the Crystal family should be applicable in larger projects. XP is also primarily targeted at small teams with practices such as "sit together". Beck and Andres (2004) explicitly state that XP alone is not sufficient for high-reliability projects. Scrum has also been developed with small teams in mind, else practices such as stand-up meetings are not viable. Still, there are Scrum adaptations for larger projects, involving multiple small Scrum teams that are coupled through meetings of individual members from each team. Lean development demands that the process should not be applied to contexts that it is not designed for ("Never push beyond limits"). It has been reported to work well on projects with 50 person-years of effort. DSDM and FDD have both been developed in larger project contexts with high reliability requirements.

B.2.8. Conclusion

The high-level comparison of the agile methods indicates that the methods are compartmentalized into two groups. On the left-hand side, Figure B.1 shows the more liberal methods Crystal Clear, XP, Scrum, and Lean development. On the right, the more heavy-weight agile methods FDD and DSDM represent the other end of the spectrum of agile methods. These two methods have more complex processes but are claimed to be better suited for larger, high reliability projects. Two distinct philosophies can be contrasted here: The liberal and optimistic view that development teams can tailor the most efficient process for their particular development environment against the pessimists who rather specify the process in detail to prevent failures or turmoil in projects. Despite the popular view of agile development as light approach, the comparison shows that there is a broad spectrum of methods that adhere to the agile values, including more more complex models.

B.3. Agile practices

Most of the popular agile practices are only explicitly formulated in one agile method: XP. As seen in the previous section, many of the agile methods provide rather abstract guidance. When practices are recurring between methods, they are similar, but their selection varies widely. Below, the individual practices are categorized into common high-level practices. From the wealth of practices in the agile methods, we discuss those that are either broadly applied in the agile ecosystem or are exemplary for the implementation of agile values. Two large surveys give insights into the actual use of agile practices. In a study by Begel and Nagappan (2007) on the dominant practices at Microsoft, most are employed "at least sometimes" by between 60 and 80% of the respondents. Test-driven development

Agile values (Beck et al., 2001)	Principles in Highsmith (2002)	Common practices
Individuals and interaction	Rely on people Encourage collaboration Technical excellence	Communication and trust Team effort and responsibility Reflective process improvement Inner quality
Working software	Deliver something useful The simplest thing possible	Iterative, incremental development Automated testing
Customer collaboration	Encourage collaboration	Customer participation
Responding to change	Be adaptable	Iterative, incremental development Inner quality Tool support

Table B.4.: Agile values, principles, and the corresponding practices

and pair programming were less frequently used (50 and 35% "sometimes", respectively). In a large survey, Parsons et al. (2007) found that eight of twelve practices were used by more than 10% of the respondents. The dominant practices of both studies can be related to the high-level practices described in the following. In addition, Table B.4 relates the common high-level practices to the agile values from the Agile Manifesto and Highsmith's principles.

Iterative and incremental development In iterative and incremental development, the development process is cyclic, instead of continuing in one pass through the phases of development. The software is developed piece by piece and requirements elicitation, designing, implementation and deployment are conducted repeatedly. The advantage is that the plan may be adjusted between iterations while the development is ongoing. Studying the requirements engineering approach in agile processes, Cao and Ramesh (2008) found that it is an iterative process with intense communication with customers, but no formal process. The observed benefits are an improved understanding of customer needs and adaption to dynamic changes. On the other hand, several challenges from agile requirements engineering were identified, including problems with non-functional requirements. Short-term planning and adapting to change is incorporated into XP through weekly and quarterly development cycles and in FDD through planning each feature separately when it should be implemented. Crystal Clear and DSDM also require frequent delivery of working software, while FDD mandates developers to not only plan, but also develop each feature separately. In addition to improved planning and requirement elicitation, the progress can also be monitored more precisely and potential release-date slips can be detected earlier and prevented in iterative development. Crystal Clear thus calls for progress monitoring through the delivery of working software.

Reflective process improvement Agile practitioners not only consider the plan to be changeable, but also the process. Agile development is often called "empirical software development", since the development teams are encouraged to tailor the development process to the development context. To integrate developers into the process design, reflections are built into several agile methods, in Scrum by proposing sprint retrospectives after each iteration, in Crystal Clear, more broadly, by calling for reflective improvement. In their study on reflection in agile development, Talby et al. (2006) found that reflections were perceived as highly efficient by team members.

Customer participation Together with the iterative development, the close integration of the customer allows for changes in plans and re-evaluation of priorities during development, resulting in more precise requirements. Moreover, the customer's control of the development is increased. All studied agile methods have customer participation as an important aspect: in Scrum as part of the sprint planning meetings, XP as the corollary practice "real customer involvement", Crystal Clear as the principle of easy access to customers, and in FDD through the domain walk-throughs. In practice, an on-site customer, as one variant of customer participation, is perceived as "an arena for discussions" (Svensson and Höst, 2005a). The planning game, a practice where customers prioritize requirements, is perceived to improve the collaboration between customer and developers (Svensson and Höst, 2005b). In a small independent software vendor, customer participation was found to incur higher costs, but improved the motivation and the confidence of the developers (Hanssen and Fægri, 2006).

Communication and trust Agile development strives to create only the necessary documentation to reduce the overhead and problems of outdated documentation, and acknowledging the value of implicit documentation, such as source code and tacit knowledge. For example, XP advocates informative workspaces and in DSDM prototypes are used as part of the documentation. Instead, agile methods encourage direct and informal communication between stakeholders and developers. Crystal Clear suggests "osmotic communication", which may be fostered through XP's "sit together" practice: All developers should share one room to lower the communication threshold. In Scrum, communication is encouraged through short daily stand-up meetings. The aforementioned empirical findings on communication (Section B.1) indicate that there is, indeed, more communication among participants of agile development, but also that challenges arise when communication does not work as intended. For example, in studies on story cards and the effect of a shared wall for communication, physical cards showed to be very important internally for industrial development teams (Sharp and Robinson, 2008). Since communication works effectively only in an environment of trust and low risk for the individual, Crystal Clear also calls for "personal safety" of the participants.

Team effort and responsibility A well-working team can improve the communication and motivation to develop more productively (see Section B.1). The team effort and responsibility is emphasized in agile methods in several ways. XP proposes "team continuity" to these ends. Lean Development stresses that a project is a team effort, and DSDM calls for a collaborative and cooperative approach. To decrease the barriers to communication between the different roles in a development project, agile methods rely on cross-functional teams. In XP this approach is part of the "whole team" practice. FDD explicitly employs cross-functional feature teams. The planning game, for example employed in XP, integrates all team members in the planning activities. Mackenzie and Monk (2004) found that the planning game reduces the gap between developers and project manager.

The members of development teams can typically identify obstacles to efficient development. Thus, in agile development, the team has a higher responsibility for the development process, taking advantage of the knowledge of each participant and, at the same time, increasing motivation. In addition to the reflective process improvement (see above), Scrum promotes "self-organizing teams" and DSDM "empowers" teams to make decisions.

Inner quality A range of different types of defects threaten the quality of software (cf. Section 3.1). In addition, maintainability is of importance to sustain the continuous, incremental development in agile processes. Accordingly, FDD requires code reviews to keep the quality of source code high. XP advocates pair programming, a practice in which two developers share one desktop as a pair. XP proponents claim that one pair is more efficient than two individual developers: Pair programming should lead to a higher focus and to finding superior solutions to development challenges. Some studies show shorter development time and higher quality for pair programming as well as pair programming to be more enjoyable (Hulkko and Abrahamsson, 2005; Tessem, 2003). Interestingly, in an experiment with student and professional developers, the professional developers were slower in pair programming than the students as the professional developers refactored more intensively (Höfer and Philipp, 2009). Similarly, a case study showed that pair programming might not consistently result in higher productivity and improved quality. Instead, pair programming is most useful for learning (Auvinen et al., 2006) and complex tasks, and, consequently, more often employed at the beginning of a project (Hulkko and Abrahamsson, 2005).

In addition to programming procedures, agile methods propose practices related to code ownership to improve the quality and maintainability. FDD suggests individual class ownership to have individual team members hold responsibility of the maintainability of the source code. XP, in contrast, advocates a shared-code approach in which all developers equally care for the produced source code and build up redundant source-code knowledge.

Automated testing Agile developers need to have a high confidence in the reliability of the developed program. One reason is that working software is delivered frequently and lengthy quality-assurance phases would be inefficient. In addition, refactoring is part of the daily routine with continuous changes to the original plan. Automated test support allows developers to efficiently assess whether the previously implemented functionality is still working as intended. As a secondary aspect, test cases can be used as implicit documentation since test cases are typically more verbose than the functional code and help to clarify which functionality is implemented and what the semantics of the programming interfaces are. Accordingly, automated testing is a common requirement in agile methods: FDD requires unit tests, Crystal Clear "automated tests", and DSDM suggests testing throughout the lifecycle.

More specifically, XP proposes test-driven development (TDD) with a test-first practice: the developer iteratively first implements a test case for new functionality and followed by as much functionality as necessary for the test case to succeed. Proponents of TDD state that TDD improves the design by requiring the developer to first think about an appropriate interface and then implement the underlying functionality. Additionally, TDD results in a high test coverage. Scientific literature is similarly broad on TDD as on pair programming. Tessem (2003) found in a case study with students and researchers in an academic setting that Test First was perceived to increase the software quality. In industrial case-studies at Microsoft and IBM, TDD lowered the pre-release defects rates by between 40 and 90% at an estimated 15 to 25% extra effort (Nagappan et al., 2008). In a comparative case-study, it was found that TDD did not change the design much, measured in object-oriented metrics, but increased the test coverage (Siniaalto and Abrahamsson, 2007). An experiment in an academic setting with undergraduate students in 12-week projects showed that test-first groups spent more time on testing. Regardless of test-first or test-last, higher testing effort improved the external product quality with a non-linear correlation (Huang and Holcombe, 2009).

Tool support To support the frequent changes to requirements and lean approach to documentation, agile methods heavily rely on adequate tool support. Crystal Clear and FDD explicitly require configuration management, which includes revision-control systems for source code. DSDM has the more high-level requirement that all changes are reversible. Other tools support progress monitoring, for example, Scrum's burn down charts. The build process is supported through continuous-integration tools, allowing for "frequent builds" (XP) of the entire system, "frequent integration"

(Crystal Clear), or “regular builds” (FDD). In addition to configuration management and build support, Kelter et al. (2002) found tools for development and documentation, communication and coordination adequate for agile development in their study on agile tools. They demand that tools in agile processes need to offer flexibility, support communication and adapt to the specific environment.

C. Practitioners’ perspectives on security in agile development

To expand the limited evidence on how agile practitioners approach security, this appendix[1] presents a study on the practitioners’ perspectives on security in agile development. Due to the lack of evidence in the area, we apply an exploratory approach through interviews with agile practitioners. Aiming to understand the effects of agile development concerning security, one question is how developers and customers interact on security. A further goal is to study what implications the security awareness and expertise of developers have and how they employ security practices.

C.1. Study design

The goal of this study is to expand on the theoretical findings on security-sensitive agile development through an exploration of the problems and their alleviation in typical agile development projects. Since there is little prior research in this area, we chose an exploratory approach and conducted semi-structured interviews. While limited in sampling size, the interviews allowed us deeply explore the problems and alleviations in practice. Based on the prior work, we identified the following areas to cover in the interviews:

- *Customer involvement:* Due to its theoretical nature, prior research is thin on the implications of customer involvement, particularly, how customers relate non-functional and functional security requirements.
- *Developer security awareness and expertise:* Prior research recommends security training for developers. How security-aware are developers typically and where do expertise and awareness originate?
- *Effects of “agile” on security:* Theoretical works assume a number of mostly negative effects of agile methods on security. Which effects do practitioners see from agile methods and practices?
- *Security practices:* Several security practices are suggested to improve security in agile development. Are they present and how well do they integrate?
- *Authorization:* Authorization is a particularly dynamic security measure (Sinclair et al., 2008). How is it affected by agile development?

The sampling of the participants was conducted in a way to represent a wide variety of agile development contexts. The sampling dimensions included the interviewee’s process role and project characteristics, such as team size and development platform. Table C.1 lists the ten interviewees from nine companies. The interviewees were directly contacted primarily through agile development meet-ups. The interviews lasted between 30 and 60 minutes, five were conducted on the telephone

[1]This study has been published in an earlier, but similar form in the proceedings of ARES 2011 (Bartsch, 2011b).

Pseud.	Role	Affiliation type	Project characteristics
Moritz	Developer	Small to medium dev. comp.	External, 7 dev., JavaEE
Ben	Developer	Small to medium dev. comp.	External, 3–6 dev., Rails
Max	Developer	Small dev. company	External, 4–6 dev., Rails
Herbert	Developer	Small dev. company	External, 3 dev., Rails
Stefan	Developer	Medium-sized company	Internal, safety-critical, 11 dev., C#
Niels	Developer	Medium-sized company	Internal, security-critical, 10 dev., JavaEE
Wolfgang	Developer	Consultant developer in medium-sized company	Internal, 4–9 dev., Rails
Günther	Coach	Consultant	Internal, ca. 1000 dev.
Klaus	Customer CEO	Medium-sized company	External, 1–3 dev., Rails
Jan	Business analyst	Consultant, in medium-sized organization	External, 3 dev., 2 QA at customer, Delphi/ASP.NET

Table C.1.: Interviewee sampling

	Literature	Mentions
Process/lifecycle model		
Process dynamics	Poppendieck (2002)	–
Emergent requirements, design changes	De Win et al. (2009); Davis (2005); Beznosov and Kruchten (2004); Goertzel et al. (2006)	6
Security is difficult to retrofit	Chivers et al. (2005); Wäyrynen et al. (2004)	–
Assurance does not fit with lifecycle model	Beznosov and Kruchten (2004); Goertzel et al. (2006)	1
Communication/interaction		
Neglected non-functional security reqs.	Goertzel et al. (2007); Cao and Ramesh (2008)	–
Missing requirements traceability	Poppendieck (2002)	–
Missing assurance objectivity	Beznosov and Kruchten (2004); Goertzel et al. (2006)	1
Lack of/outdated security documentation	Goertzel et al. (2006); Beznosov and Kruchten (2004)	2
Lack of customer awareness/sec. reqs.	–	5
Trust in team and individuals		
Implicit trust in developers	Goertzel et al. (2006)	–
Lack of specialist expertise on team	Goertzel et al. (2006, 2007); Wäyrynen et al. (2004)	–
Neglected practices from pressure	–	1

Table C.2.: Challenges in literature and the number of mentions in this study

and five in face-to-face settings. Based on detailed notes on each interview, we structured the statements by common concepts iteratively with each interview and clustered related aspects to derive the findings on problems and alleviations (cf. Tables C.2 and C.3, respectively, which include the relation to the literature from the previous sections and the number of interviewees who mentioned each concept).

C.2. Findings

C.2.1. Unclear security requirements

The interviewees portrayed the security awareness among customers as covering a broad range, from marginally to highly aware. Five interviewees mentioned problems related to the lack of security awareness. Ben noted that the security awareness on the customer side "depends on the specific person that you talk to." Interviewees generally observed that the awareness on the customer side is often driven by specific values at risk and requirements and threats need to be discussed in concrete terms (6 mentions).

	Literature	Mentions
Process/lifecycle model		
Defining and auditing the process	–	3
Adapting the process for security	–	3
High-level security architecture up-front, early/late security sprint	Sullivan (2008); Beznosov and Kruchten (2004); Ge et al. (2007)	3
Iterative, incremental security design	Aydal et al. (2006); Chivers et al. (2005)	6
Individual sec. activities per sprint	Sullivan (2008)	–
Close customer integration	–	6
Security requirements		
Abuse cases	Kongsli (2006); Boström et al. (2006); Heikka and Siponen (2006); Mellado et al. (2006)	–
Security requirements traceability	Poppendieck (2002)	–
Explicit risk analysis	Aydal et al. (2006)	2
Implicit security requirements	–	7
Concrete requirements and threats	–	6
Non-functional reqs. as done-definition	–	2
Design, implementation, assurance		
Secure design and implementation	Beznosov and Kruchten (2004)	2
Test cases/code as documentation	Wäyrynen et al. (2004)	2
Automatic sec. testing, static analysis	Beznosov and Kruchten (2004); Kongsli (2006); Tappenden et al. (2005); Erdogan et al. (2010)	4
Security review meeting	Kongsli (2006)	2
Implicit/explicit code reviews	–	4
Formal change and integration proc.	Beznosov and Kruchten (2004)	2
Early secure deployment	Kongsli (2006)	–
Security awareness and expertise		
Prevalent security	–	7
Implicitly and explicitly spreading awareness and expertise	–	6
Rotating experts on team	Wäyrynen et al. (2004)	1
Security training	Ge et al. (2007)	2
Self-teaching security expertise	–	4
Holistic development approach	–	5

Table C.3.: Mitigations in literature and the number of mentions in this study

> *Max: Security "is only of interest [to the customer] when money-aspects are concerned."*

Security incidents involving similar systems increase the customer's high-level security awareness:

> *Moritz: "The shop leak at [newspaper], that's where customers wake up."*

> *Klaus: "There was a hacker attack in the newspapers on a shop in [close-by town, redacted] last week."*

Security requirements are discussed, "if at all, then only unspecific, together with functions" (Klaus) and are characterized as being approached from the positive side ("what is allowed" instead of "what is forbidden").

Jan's case is an exception in that the customer was very security-aware and developed very specific security requirements because the developers were rather unaware.

C.2.2. Implicit security requirements

Often, developers derive the security requirements from the functional requirements as implicit security requirements (7). For example, Moritz' development team has good domain knowledge in electronic commerce and can therefore propose adequate security requirements. For Herbert's team, the developers' best practices led to agreements between the customer and developers to minimize risks and outsource the handling of payment data to a service provider.

According to Klaus, the trust relationship between the customer and the development team is very important in this respect. Customers often cannot comprehend the technical background for decisions on security measures. The necessary level of trust is built up during the close development participation. The trust on the technical level is perceived to be similar to the trust on the inner quality of the software product, for which customers usually assume that the developers conduct adequate quality-assurance measures without specifically prescribing exact procedures.

C.2.3. Close customer participation

Six interviewees mentioned that, irrespective of the customer's security awareness, close customer participation improves the elicitation of security requirements through their domain knowledge. The security requirements are usually refined over several discussions and iterations (6 mentions). For example, developers propose technical approaches, which are discussed with the customer, then implemented and adapted in subsequent iterations. In Jan's project, authorization requirements were difficult to elicit bottom-up due to large variations among the customer stakeholders, causing overly complex requirements. The simplified, top-down model that was implemented instead then needed to be adapted in production over time, with changes even after 1.5 years in production. Another reason for discussions are functional changes: Moritz reported that over the course of the project, functional changes, such as new interfaces between the ERP and the shop system, repeatedly required security discussions. In his experience, adaptations to the original plan are mostly necessary when the best practices of the developer team do not fit as expected in the customer context.

There was a broad consensus among the interviewees that authorization requirements change frequently, often once per iteration. The reasons that were named are the evolving functionality, new user groups accessing the system and the realization that permissions are inappropriate in production. The style of customer interaction on authorization varies. A permission matrix can be appropriate, but more common are natural language discussions without a document as basis.

> *Max: Developers refer "first to the configuration then to source code and then to the behavior of the application."*

In some projects, such as Jan's, a process handbook is provided by the customer as a high-level foundation of authorization restrictions. Two interviewees mention that missing or outdated security documentation is a problem. In Jan's case, a five-page document is necessary to describe the high-level authorization concept used in the application in addition to seven separate lists with the actual permissions that are maintained in parallel to the actual permission configuration.

C.2.4. Awareness and expertise spreads among developers

Seven interviewees described security as being generally present within the developer teams and the expertise as being rather homogeneous. Exceptions are interns and new employees:

> *Max: They "need some time to take up the necessary security awareness."*

There are several ways in which the security awareness and expertise is built. Generally, security expertise and awareness spreads between developers (6 mentions) and security is part of informal discussions.

> *Moritz: "There is an exchange of information between the people of the team and between the teams, 'how have you handled the problem?' "*

Self-taught awareness and expertise from the news and blogs was also mentioned frequently (4). One reason is the motivation of feeling "responsibility for the project" due to the holistic development approach (5). Holistic development also increases the awareness and expertise: New developers, for example, rotate through multiple project teams to work on various parts of a project. Comparing the motivation to pre-agile development, Stefan stated that the improved communication among developers and quality assurance helped. Other motivating factors were the external audits and the inner-company competition for quality between teams.

Less common are explicit trainings for security requirements that are required from regulations (2). Other institutionalized knowledge sharing occurs through security review meetings (2). Moritz also reported on coding Dojos (collective development sessions), in which developers train specific quality-improving agile practices, such as test-driven development.

C.2.5. Developing securely

Three interviewees mentioned that there are problems of integrating assurance into the agile lifecycle, or, similarly, of neglected assurance practices from the pressure of short iterations. The full-stack tests can be more difficult with agile processes because the effort is too high for rigorous testing on each iteration. Short iterations can cause pressure to make visible progress, in some cases even causing practices, such as test-driven development, to be neglected.

Two interviewees stated that the iterative and incremental development allows them to find more direct approaches and keep the software design simpler as recommended in secure software design. Herbert reported that the developers always attempt to keep the authorization model simple to prevent side effects by asking the customer which limitations are actually necessary. The design also benefits from the holistic development approach (5) since it offers a more complete picture for the individual developers. If non-functional requirements, such as quality and high-level security aspects, should be made explicit, they can be formulated as "done definitions" (2) that specify when a feature is considered complete. For instance, this may include a security review.

C.2.6. Employing low-overhead assurance

In most cases, the assurance practices are integrated into the development process. Four interviewees mentioned testing and test-driven development, typically conducted by the development team. In contrast, the customer took over most of the testing activities in Jan's project to achieve the necessary quality. Penetration tests were conducted in two cases, either as part of the done criteria, or in parallel to the development on the production system. Code reviews are more common (4), for example, as a four-eye principle on each software repository check-in or as part of the done criteria. Three interviewees reported that the practices are adapted over time to changing contexts, so that, for example, the code-review practices evolved with the company and development processes.

C.2.7. Adapting the process model to high criticality

In three cases, the processes were defined and externally audited in parallel to development to assure the process quality. The same three projects also employed dedicated security or release iterations before the actual release to create the required documents and thoroughly test the product. Explicit risk analysis and management was mentioned by two interviewees, either assessing the risk for individual features before deployment or following regulations on a systematic risk management process, including explicit risk meetings and sign-off procedures.

C.3. Discussion

This study explored the problems with security in agile development and respective alleviations through interviews with practitioners. The interviews offer only subjective data and are prone to researcher or participant bias. Since the sample size is limited for interviews, we focused on covering a broad range of development contexts. The results are, by study design, not sound and representative, but extend the prior theoretical findings with a practical perspective and offer a description as an initial hypothesis for further research.

Overall, the problems and alleviations from prior work are mentioned in the study where applicable to the study's scope. Tables C.2 and C.3 contrast the key concepts of challenges and alleviations from literature with those mentioned by study participants for security in agile development. Six of the challenges from literature were not mentioned in the study, four are present in both, and two challenges mentioned in the study were not covered by previously. Three of the challenges not mentioned by participants are outside of the core focus of the study and apply to highly critical contexts, such as "Requirements traceability" and "Implicit trust". Others may not be as critical as seen in the theoretical prior work ("Lack of specialist expertise", "Retrofitting security"). Not previously covered problems are the very practical issues with the customer awareness and pressure leading to neglected security practices. In case of the alleviations, four of the 14 alleviations in literature were not mentioned in the study and participants stated 11 further alleviations. The alleviations from literature without mention were primarily very specific ("Abuse cases", "Individual security activity per sprint"), or applied to high criticality ("Requirements traceability"). The alleviations not present in previous work primarily leverage the strengths of agile development ("Close customer integration", "Spreading awareness and expertise"), but also regard specific problems with security ("Implicit security requirements", "Concrete requirements"). We discuss the key results in the following:

Institutionalize customer involvement Despite not covered in prior work, a good interaction concerning security between developers and customers is a prerequisite to adequate security measures.

In this study, the interaction differs greatly among projects and customers can often only state unclear security requirements, leading to implicit security requirements. Non-functional requirements need to be elicited together with customers and then iteratively turned into functional requirements. Thus, *customers and developers need a common understanding of the roles in the project regarding security*, including:

- Which side has how much security awareness and expertise,
- Who is responsible for triggering discussions of non-functional and functional security requirements.

Risk assessments are often only implicitly part of functional discussions. While not necessarily as an explicit security iteration as suggested in literature (Sullivan, 2008), *customers and developers should explicitly address the risks and the non-functional security requirements early in the project* to have a common framing for the later decisions on security measures and implementation practices. In discussions, *developers should focus on concrete threats and security requirements to improve the common understanding* (West, 2008).

Foster developer security awareness and expertise The overall security in a project depends on the security expertise of the individuals, both on the customer or developer side. This corresponds to the agile value of "individuals and interaction over processes and tools" (Beck et al., 2001). The developers in this study for the most part feel responsible to craft secure systems and are motivated to learn and spread security expertise, also supported by the communicative nature of agile processes. Nevertheless, this cannot be taken for granted, as shown in Jan's case. *The spreading of awareness and expertise among developers should thus be further fostered.* Not only through explicit security trainings (Ge et al., 2007) and experts on teams (Wäyrynen et al., 2004), as suggested in literature, but also by motivating the exchange between developers, for example, through agile practices with positive effects on internal communication.

Continuously improve the process for security Three teams in this study are actively adapting the development processes over time for it to optimally suit the environment. While this has not been the focus in literature, *developers should consider security aspects of the development process in retrospectives and adapt it to meet security needs.* Retrospectives are also a good place to consider the integration of secure development practices, for example, from SSE–CMM (ISSEA, 2003). Three cases in this study show that it is possible to integrate heavy-weight security practices, in contrast to the skeptical prior work.

Promote implicit documentation Literature and two cases in this study point to the problem of outdated security documentation in agile development. Not only test cases (Wäyrynen et al., 2004), but also other artifacts, such as authorization policies, can provide implicit documentation. *Developers should consider to move the security documentation into the code where possible* to create current documentation as the basis for discussions with customers.

D. Literature on problems surrounding authorization

As part of the existing body of literature on authorization, numerous problems surrounding this measure have been identified. Relevant publications are listed in the following table. Each publication has a specific focus, targeting one or more authorization perspectives in one of the variety of authorization contexts. We refer to this list as part of the analysis of problems in prior literature (cf. Section 2.3), particularly noting the respective study design and the perspectives covered.

Source	Study design	Environment	Focus/scope	Perspective
Sikkel and Stiemerling (1998)	Subjective: interviews	Private and public organizations	How users specify policies	Authoring
Whalen et al. (2006)	Subjective: survey, interviews	Medium-sized research laboratory	Individuals' problems from mechanisms	Authoring, making, functional
Bauer et al. (2009)	Subjective: interviews	Diverse organizations	Challenges for policy professionals	Authoring, making
Smetters and Good (2009)	Objective: historical policy data	Medium-sized corporation	Usage of authorization features and models	Authoring, making
Inglesant et al. (2008)	Subjective: interviews	E-Scientist collaboration	Authoring through controlled natural language	Authoring
Bauer et al. (2008a)	Subjective: interviews, objective: usage data	Physical security in offices with smartphone system	How can users implement ideal policies	Making, authoring
Ahern et al. (2007)	Objective: usage data, subjective: interviews	Social networking, photo sharing	Privacy decisions in mobile/online sharing	Authoring, making
Kim et al. (2010)	Subjective: interviews	Home, allowing guest access	What kind of policies are needed?	Authoring, making
Wool (2004)	Objective: policy data	Firewall rules in organizations	Mistakes in firewall rules	Authoring
Sinclair et al. (2008)	Field study	Financial organization	Security in organizations	Functional, making
Gallaher et al. (2002)	Case study, survey	Organizational authorization	Benefits of RBAC	Authoring
Jaeger et al. (2004)	Case study	Linux Kernel	Authorization enforcement	Developing
Vaniea et al. (2008b)	Laboratory: usability	Privacy in organizations	Natural language policy authoring	Authoring
Herzog and Shahmehri (2006)	Laboratory: usability	Java application policy authoring	Setting of Java policies	Authoring, Developing
Maxion and Reeder (2005)	Laboratory: usability	OS file permissions	Prevent mistakes in file permission setting	Authoring
Reeder et al. (2007)	Laboratory: usability	Enterprise privacy policies	Challenges in privacy policy authoring	Authoring

Table D.1.: Literature on problems surrounding authorization (continued on next page...)

Source	Study design	Environment	Focus/scope	Perspective
Zurko and Simon (1996)	Laboratory: user-centered design	–	Policy editing usability	Authoring, Developing
Zurko et al. (1999)	Laboratory: usability	Distributed research environments	Usable policy editor	Authoring, Developing
Rode et al. (2006)	Laboratory: usability	Ad hoc file sharing	Challenges in policy authoring	Authoring, making
Reeder et al. (2008)	Laboratory: usability	File permissions	Challenges in file permission setting	Authoring
Brostoff et al. (2005)	Laboratory: usability	E-Scientist collaboration	Improve policy authoring tool	Authoring
Reeder et al. (2011)	Laboratory: usability	File permissions	Conflict resolution	Authoring
Bertino et al. (2008)	Analytical	Enterprise authorization	Role-mining and usability	Authoring
Blakley (1996)	Analytical	Organizational authorization	General problems in information security management	Authoring
Chadwick and Sasse (2006)	Analytical	E-Scientist collaboration	Policy formulation	Authoring
Al-Shaer and Hamed (2003)	Analytical	Firewall rules in organizations	Discover anomalies and support editing	Authoring
Yee (2005)	Analytical	General security mechanisms	Design guidelines for system security	Authoring
Johnson et al. (2009)	Analytical	File sharing in organizations	Laissez-faire file-sharing	Functional, making
Kapadia et al. (2004)	Analytical	Organizational authorization	Provide feedback on denials	Functional
Becker and Nanz (2008)	Analytical	Organizational authorization	Support policy editing, delegation	Authoring
Bonatti et al. (2006)	Analytical	Web access control	Explaining proof failures in authorization decisions	Authoring
Vaniea et al. (2008a)	Analytical	Large policies	Linking policy problems to the source	Authoring
Joshi et al. (2005)	Analytical	GTRBAC policies	How the model affects policy complexity	Authoring

Table D.1.: Literature on problems surrounding authorization

E. A list of approaches to authorization problems

Authorization in information systems has been extensively researched in the last decades (cf. Section 2.4.4). To assess how well the requirements on usable authorization (cf. Section 6.1) are already addressed by prior research, we list important works in the following. We base the elicitation of open research questions (cf. Section 6.3) on this body of work. Apart from approaches directly from literature, the table also includes the interrelation between the requirements as these build upon another – for example, Req 1 *Authorization measure* is addressed through Req 5 *Circumventions*.

Approach	Description	Type	Argument	Source
Req 1 ***Authorization measure***				
Circumventions	How and when to circumvent authorization measures	Requirement		Req 5 *Circumventions*
Change requests	Ability to request changes with reasonable effort	Requirement		Req 9 *Change requests*
Decision-making	Precise decisions to prevent over-entitlements at low effort	Requirement		Req 6 *Policy decisions*
Procedures	Decisions by authoritative staff and acceptable coordination effort	Requirement		Req 10 *Change procedures*
Reviews and monitoring	Monitoring functional staff for their compliance in using the measure and regularly reviewing permissions for over-entitlements, both at low effort	Requirement		Req 7 *Reviews and monitoring*
Policy changes	Formalizing the policy decisions correctly and efficiently	Requirement		Req 8 *Policy changes*
Controls integration	Reliably integrate authorization enforcement and decisioning in systems, expending low effort	Requirement		Req 15 *Controls integration*
Req 2 ***Functional productivity***				
Reduce disruptions	Integrate in workflows	Recommendation	Field study	Whalen et al. (2006)
Reduce disruptions	Minimal friction	Principle	Analytical	Johnson et al. (2009)
Reduce disruptions	Adequate restrictions from the formalize policy decisions	Requirement		Req 8 *Policy changes*
Reduce disruptions	Adequately restrictive policy from adequate policy decisions	Requirement		Req 6 *Policy decisions*
Reduce disruptions	Enable and communicate requests to change restrictions with low disruptions and prompt request processing	Requirement		Req 9 *Change requests*
Reduce disruptions	Reduce disruptions through acceptable circumventions	Requirement		Req 5 *Circumventions*
Reduce disruptions	Flexible restrictions with adequate authorization models, e.g. implementing delegation and policy override	Requirement		Req 12 *Model usability*
Req 3 ***Regulatory requirements***				

Table E.1.: Approaches in literature and requirement interdependence (continued on next page…)

Approach	Description	Type	Argument	Source
Measure effectiveness	Effectively enforce regulatory requirements through authorization	Requirement		Req 1 *Authorization measure*
Compliant procedures	Implement procedures to guarantee adequate decisions and formalization of policy changes	Requirement		Req 10 *Change procedures*
Compliant decision-making	Honor regulatory requirements in policy decisions	Requirement		Req 6 *Policy decisions*
Req 4 ***Individual satisfaction***				
Functional work	Reduce unnecessary disruptions in functional work	Requirement		Req 2 *Functional productivity*
Circumventions	Guide circumventions to reduce disruptions	Requirement		Req 5 *Circumventions*
Efficient measure	Reduce necessary effort for the changing of policies	Requirement		Req 1 *Authorization measure*
Efficient measure	Reduce necessary effort for the management of authorization measure	Requirement		Req 1 *Authorization measure*
Req 5 ***Circumventions***				
Reduce disruptions	Reduce disruptions to reduce necessity of circumventions	Requirement		Req 2 *Functional productivity*
Increase risk awareness	Foster informal rules on circumvention	Recommendation	Field study	Section 5.2
Increase risk awareness	Training, education, awareness, motivation	General approach		Roper et al. (2005)
Increase security awareness	Training and awareness campaigns	Best practices		ISO/IEC 27002:2005 (2005)
Guide circumventions	Foster formal rules on circumvention	Recommendation	Field study	Section 5.2
Monitor circumvention	Derive insights on authorization problems from circumventions	Recommendation	Field study	Section 5.2
Disciplinary measures	Formal and fair disciplinary process	Best practices		ISO/IEC 27002:2005 (2005)
Req 6 ***Policy decisions***				
Adequate restrictions	Task analysis	General approach		Hollnagel (2006)
Adequate restrictions	Less restrictive, support social controls	Recommendation	Field study	Whalen et al. (2006)
Decision guidance	Provide high-level policies on authorization changes	Recommendation	Field study	Section 5.2
Decision guidance	Guidance to enable adequate decisions and reduce the decision complexity through support	Requirement		Req 11 *Decision guidance*
Improve expertise	Increase the expertise and awareness of decision makers	Recommendation	Field study	Section 5.2
Improve expertise	Collaborative action learning	General approach		Ngwenyama (1993)
Model expressiveness	Improve authorization models	Recommendation	Field study	Section 5.2

Table E.1.: Approaches in literature and requirement interdependence (continued on next page. . .)

Approach	Description	Type	Argument	Source
Model expressiveness	Expressiveness of authorization model influences the granularity and thereby the number and precision of decisions	Requirement		Req 12 *Model usability*
Functional staff integration	Integrate functional staff in security management	Principle	Analytical	Church (2008)
Functional staff integration	Integration of tasks, monitoring, configuration	Approach	Laboratory experiment	Rode et al. (2006)
Functional staff integration	Integration of policy decisions into workflow UI	Approach	Analytical	Karp and Stiegler (2010)
Functional staff integration	Transparency of authorization decisions and the implemented policy to facilitate change decisions by non-experts	Requirement		Req 14 *Mechanism usability*
Authorization requirement elicitation	Goal-driven requirements for privacy policies	Method	Field study	He and Antón (2009)
Authorization requirement elicitation	Requirements analysis	Approach	Analytical	Koch and Parisi-Presicce (2003)
Authorization requirement elicitation	Scenario-based role engineering	Method	Field study	Neumann and Strembeck (2002)
Authorization requirement elicitation	Evolutionary role engineering	Method	Analytical	Takabi and Joshi (2010)
Req 7 ***Reviews and monitoring***				
Monitoring practice	Discover unintended information disclosures, non-compliance, access violations	Best practices		ISO/IEC 27002:2005 (2005)
Monitoring practice	Auditing in health care information system for authorization	Recommendation	Field study	Røstad and Edsberg (2006)
Monitoring implementation	Enable efficient and effective monitoring through adequate implementation of mechanism	Requirement		Req 15 *Controls integration*
Review practice	Regular reviews	Best practices		ISO/IEC 27002:2005 (2005)
Review practice	Self-review of staff	Approach	Field study	Sinclair et al. (2008)
Review tool support	Comprehensible policies, adequate policy review tool for precise and efficient reviews	Requirement		Req 13 *Policy and tool usability*
Req 8 ***Policy changes***				
Concrete changes	Concreteness reduces cognitive load	General approach	Analytical	Blackwell et al. (2008)
Concrete changes	Use examples	Approach	Field study	Inglesant et al. (2008)
Concrete changes	Employ email metaphor	Approach	Analytical	Johnson et al. (2009)
Concrete changes	Security by designation	Principle	Analytical	Yee (2005)
Improve usability	Comprehensible policies, adequate editing and supporting tools, and usable policy syntax for precise changes with low effort	Requirement		Req 13 *Policy and tool usability*

Table E.1.: Approaches in literature and requirement interdependence (continued on next page…)

Approach	Description	Type	Argument	Source
Reduce complexity	Small rule sets	Recommendation	Field study	Wool (2004)
Reduce complexity	Expressiveness of authorization model to formalize decisions adequately at low effort	Requirement		Req 12 *Model usability*
Reduce complexity	Adequately flexible and restrictive policy decisions for reduced authoring effort	Requirement		Req 6 *Policy decisions*
Improve procedures	Effective and efficient communication with policy makers	Requirement		Req 10 *Change procedures*
Functional staff integration	Integrate functional staff in security management	Principle	Analytical	Church (2008)
Req 9 ***Change requests***				
Procedure adequacy	Adjust degree of centralization and formalization	Recommendation	Field study	Section 5.2
Change effort	Reduce the (perceived) change lead-time and change effort	Recommendation	Field study	Section 5.2
Procedure awareness	Define and communicate procedures	Recommendation	Field study	Section 5.2
Req 10 ***Change procedures***				
Management approach	Process-based role management	Recommendation	Field study	Sinclair et al. (2008)
Process model	Multi-level policy lifecycle	Method	Analytical	Dai and Alves-Foss (2002)
Process model	High-level policy lifecycle	Method	Analytical	Rees et al. (2003)
Process model	High-level policy change procedure model	Method	Analytical	OGC (2007c)
Process model	RBAC policy lifecycle model	Method	Field study	Kern et al. (2002)
Process model	Technical policy lifecycle model	Method	Analytical	Mönkeberg and Rakete (2000)
Change management	Security configuration management	Best practices		ISSEA (2003)
Change management	Change management approaches	General approach		Joeris (1997)
Validation and verification plan	Plan of which artifacts need to be assured by whom	Best practices		ISSEA (2003)
Procedure adequacy	Adjust degree of centralization and formalization	Recommendation	Field study	Section 5.2
Procedure awareness	Define and communicate procedures	Recommendation	Field study	Section 5.2
Req 11 ***Decision guidance***				
Authorization requirement elicitation	Policy requirements based on organizational structures	Method	Analytical	Crook et al. (2003)
Authorization requirement elicitation	Combination of bottom-up and top-down role analysis	Method	Field study	Kern et al. (2002)
Authorization requirement elicitation	Role mining	Approach	Analytical	Bertino et al. (2008)
Authorization requirement elicitation	The use of inheritance in organizational policies	Recommendation	Analytical	Moffett (1998)
Risk assessment	Practices for the specification of security needs	Best practices		ISSEA (2003)

Table E.1.: Approaches in literature and requirement interdependence (continued on next page...)

Approach	Description	Type	Argument	Source
Risk assessment	Business requirements surrounding authorization	Best practices		ISO/IEC 27002:2005 (2005)
Awareness of consequences	Awareness of consequences from decisions	Recommendation	Field study	Ahern et al. (2007)
Decision support	Decision support from prior decisions to prevent mistakes	Recommendation	Field study	Ahern et al. (2007)
Decision support	Provide decision support	Recommendation	Field study	Section 5.2
Decision support	Risk-based analysis of permission assignments	Approach	Analytical	Nissanke and Khayat (2004)
Decision support	Market mechanisms for risk-based decision guidance	Approach	Analytical	Molloy et al. (2008)
Decision support	Model of incentives to manage escalation risks	Approach	Analytical	Zhao and Johnson (2009)
Req 12 ***Model usability***				
Cognitive effort	Learnability, reduced cognitive effort, maintainability	Recommendation	Analytical	Blakley (1996)
Cognitive effort	Simpler models, patterns	Recommendation	Field study	Smetters and Good (2009)
Cognitive effort	Adequate conflict resolution algorithm	Approach	Laboratory experiment	Reeder et al. (2008)
Cognitive effort	Specificity-based conflict resolution works best	Approach	Laboratory experiment	Reeder et al. (2011)
Model expressiveness	Precise limitation of scopes, handling exceptions	Recommendation	Field study	Sikkel and Stiemerling (1998)
Model expressiveness	Improve models to handle unexpected events	Recommendation	Field study	Bauer et al. (2009)
Model expressiveness	Fine-grained controls	Recommendation	Field study	Ahern et al. (2007)
Model expressiveness	Policies based on presence, logging, asking for permission; grouping of people	Approach	Laboratory experiment	Kim et al. (2010)
Model expressiveness	Logging, notifications, transitivity, grouping granularity	Approach	Field study	Bauer et al. (2008a)
Flexibility	Improve flexibility	Recommendation	Analytical	Blakley (1996)
—Delegation	Delegation formalization for RBAC	Approach	Analytical	Ahn et al. (2003)
—Delegation	General option of delegation for users	Principle	Analytical	Johnson et al. (2009)
—Policy override	Requirements for optimistic security	Recommendation	Analytical	Povey (2000)
—Policy override	Policy override in clinical information systems	Approach	Field study	Denley and Smith (1999)
—Policy override	Policy override model in health care environment	Approach	Field study	Ferreira et al. (2009)
Flexibility	Uno-tempore authorization through negotiation	Approach	Field study	Stiemerling and Wulf (2000)
Flexibility	Authorization through audits	Approach	Analytical	Cederquist et al. (2007)
Flexibility	Risk-based authorization decisions	Approach	Analytical	Diep et al. (2007)

Table E.1.: Approaches in literature and requirement interdependence (continued on next page…)

Approach	Description	Type	Argument	Source
Flexibility	Risk-based multi-level security	Approach	Analytical	Cheng et al. (2007)
Req 13 ***Policy and tool usability***				
Policy usability	Comprehensible authorization model to enable comprehensible policies	Requirement		Req 12 *Model usability*
Policy-editing tool usability	Make controls simpler to manage	Recommendation	Field study	Whalen et al. (2006)
Policy-editing tool usability	Better authoring tools	Recommendation	Field study	Smetters and Good (2009)
Policy-editing tool usability	Requirements: integrates management of policy aspects, convenient interface, consistency checks, consequences of rules, applicable rules	Recommendation	Analytical	Zurko and Simon (1996)
Policy-editing tool usability	Privacy-editing requirements: group objects, consistent terms, comprehensible default rules, policy structure, rule conflict resolution	Recommendation	Laboratory experiment	Reeder et al. (2007)
Policy-editing tool usability	Smooth transition between authoring phases	Approach	Laboratory experiment	Vaniea et al. (2008b)
Policy-editing tool usability	Support communication of authors and makers	Recommendation	Field study	Bauer et al. (2009)
Policy-editing tool usability	Appropriate level of abstraction	Method	Analytical	Dai and Alves-Foss (2002)
Policy-editing tool usability	Developer and user mode, usage of the error log	Recommendation	Field study	Inglesant et al. (2008)
—Presentation and visualization	Error avoidance through effective permission display; necessary information in screens; no misleading constructs in UI	Approach	Laboratory experiment	Maxion and Reeder (2005)
—Presentation and visualization	Presentation of whole policy, visualization of exceptions and overrides	Approach	Laboratory experiment	Reeder et al. (2008)
—Presentation and visualization	Good visualization of large policy	Recommendation	Analytical	Bertino et al. (2008)
—Presentation and visualization	Clear consequences, understandable terms and identifiers	Recommendation	Analytical	Yee (2005)
—Presentation and visualization	Comprehension of consequences	Approach	Laboratory experiment	Rode et al. (2006)
—Presentation and visualization	Help on semantics	Recommendation	Laboratory experiment	Herzog and Shahmehri (2006)
—Presentation and visualization	Visualize analysis output, show effects of changes	Approach	Laboratory experiment	Vaniea et al. (2008b)
—Editing support	Anomaly detection and supported editing	Approach	Analytical	Al-Shaer and Hamed (2003)
—Editing support	Explaining decisions through proof explanations	Approach	Analytical	Bonatti et al. (2006)
—Editing support	Intentional access management through decision support	Approach	Laboratory experiment	Cao and Iverson (2006)
Supporting tools	Role mining by clustering users and permissions from business processes	Approach	Analytical	Kuhlmann et al. (2003)
Supporting tools	Role mining by clustering users and permissions from similar users	Approach	Analytical	Schlegelmilch and Steffens (2005)
—Analysis tools	Change impact analysis	Approach	Analytical	Fisler et al. (2005)

Table E.1.: Approaches in literature and requirement interdependence (continued on next page…)

Approach	Description	Type	Argument	Source
—Analysis tools	Role analysis tool	Approach	Analytical	Gofman et al. (2009)
—Assurance tools	Policy analysis with UML and OCL	Approach	Analytical	Sohr et al. (2008)
—Assurance tools	Analysis of organizational controls	Approach	Analytical	Schaad et al. (2006)
—Assurance tools	Traceability from high-level requirements to policy	Approach	Analytical	Verhanneman et al. (2005)
—Assurance tools	Analysis of inherent policy quality	Approach	Analytical	Dougherty et al. (2006)
—Assurance tools	Analysis of inherent policy quality: XACML coverage	Approach	Analytical	Martin et al. (2006)
—Assurance tools	Analysis of inherent policy quality: dominance, conflicts, coverage	Approach	Analytical	Agrawal et al. (2005)
—Assurance tools	Policy test-case generation	Approach	Analytical	Hwang et al. (2008)
—Assurance tools	Verification of property-set quality	Approach	Analytical	Martin et al. (2008)
Language usability	Concrete syntax: distance between human requirement and abstract syntax	Recommendation	Laboratory experiment	Pane et al. (2001)
Language usability	Controlled natural language, iterative procedure	Approach	Field study	Inglesant et al. (2008)
Language usability	Tool support: syntax highlighting	Approach	Laboratory experiment	Vaniea et al. (2008b)
Language usability	Procedural support: iterative transformation of natural language into policy	Approach	Laboratory experiment	Vaniea et al. (2008b)
Tool, language usability	Comprehensible authorization model to implement usable tools and languages	Requirement		Req 12 *Model usability*
Req 14 ***Mechanism usability***				
Transparency	Make authority transparent	Principle	Analytical	Yee (2005)
Transparency	Make access control decisions visible: easier to see what has been configured	Recommendation	Field study	Whalen et al. (2006)
Transparency	Feedback on why requests are denied	Approach	Analytical	Kapadia et al. (2004)
Transparency	Explaining denials, necessary changes	Approach	Analytical	Becker and Nanz (2008)
Transparency	Principle of transparency	Principle	Analytical	Johnson et al. (2009)
Transparency	Suggest responsible policy maker	Approach	Field study	Bauer et al. (2008b)
Transparency	Comprehensible model to enable comprehensible and transparent authorization decisions	Requirement		Req 12 *Model usability*
Decision integration	Integrate policy decisions in workflow	Approach	Analytical	Karp and Stiegler (2010)
Delegation	Delegation assistance	Approach	Analytical	Brucker et al. (2009)
Model implementation	Interactive decisions, reactive delegation	Approach	Field study	Bauer et al. (2008a)
Model expressiveness	Expressive authorization model to implement usable mechanism	Requirement		Req 12 *Model usability*
Req 15 ***Controls integration***				
Enforcement integration	Need to support developers in integrating controls	Recommendation	Field study	Herzog and Shahmehri (2006)

Table E.1.: Approaches in literature and requirement interdependence (continued on next page...)

Approach	Description	Type	Argument	Source
Manage controls	Maintenance and review of controls	Best practices		ISSEA (2003)
Integrated development	Integrate policy specification into software development	Method	Field study	He and Antón (2009)
Vulnerability management	Technical vulnerability management	Best practices		ISO/IEC 27002:2005 (2005)
Security patterns	Authorization enforcement patterns	Approach	Analytical	Priebe et al. (2004)
Enforcement architecture	Decoupling authorization from application logic	Principle	Analytical	Beznosov et al. (1999)
Enforcement architecture	OS-based controls	Approach	Analytical	Harrison et al. (1976)
Enforcement architecture	API-based controls in Java	Approach	Analytical	Gong and Ellison (2003)
Enforcement architecture	Language-based enforcement	Approach	Analytical	Goguen and Meseguer (1982)
Enforcement architecture	Aspect-oriented programming for authorization	Approach	Analytical	Kiczales et al. (1997)
Enforcement validation	Analysis tools to identify missing enforcement	Approach	Field study	Jaeger et al. (2004)
API usability	Cognitive dimensions model	General approach	Analytical	Clarke (2004)

Table E.1.: Approaches in literature and requirement interdependence

F. Policy override calculus

As described in Chapter 8, policy override allows functional users to temporarily extend their permission to prevent interferences from missing permissions. When defining an authorization policy with override options, security and domain experts not only need to assign the normal privileges of each role. For the normal privileges, experts usually evaluate the authorization requirements from a *need-to-know* or *least-privilege* perspective. In policies for policy override, a second, *hard* boundary needs to be defined in addition to the *soft* boundary, which is derived from the need-to-know evaluation. The hard boundary is a result of the balancing of risks and benefits. The risks arise from access to additional, not routinely needed knowledge and abilities of each role. Benefits are the potential advantages of policy override from the higher flexibility.

The decision-making of which override extent is appropriate is difficult, as stated in the policy-override study (Section 8.3), since various threats and risks as well as several factors relating to the benefits need to be considered. When the authorization-override model allows fine-grained settings of who may override to what extent, a considerable amount of effort needs to be spent to analyze risks and gains for each case. To help with this decision and integrate functional staff in the policy-making task (P1), this chapter[1] proposes one approach of supporting the decision-making through the Policy Override Calculus. The calculus guides the decision and gives an estimate of how appropriate specific hard boundaries for individual roles are. Through a systematic collection of risk and benefit inputs and the appropriateness output, the calculus reduces the cognitive effort for the policy maker (P2). The calculus employs formulae based on qualitative risk-management principles and combines approaches from several disciplines, including risk and decision sciences, and criminology for the risk and decision models, and the socio-technical for a comprehensible decisioning model (P4). The Policy Override Calculus estimates the override adequacy by employees' roles and override extent. The calculus is implemented in an override risk management tool, which we evaluate briefly on the adequacy of its guidance and the approach of data collection for the calculus (Req 11, Q11.1, Q11.2).

F.1. Decisions on policy override

To the best of our knowledge, there have been no prior publications on decision support for the configuration of policy override. However, risk assessment has been been included in override authorization models by Cheng et al. (2007) in their optimistic authorization model as discussed above. Without explicitly offering override functionality, the risk-based access control model RAdAC balances operational needs versus security risk (Choudhary, 2005). Britton and Brown (2007) analyzed risk factors to be used in the RAdAC model. Similarly, Diep et al. (2007) described an authorization model with context-based decisions that includes a quantitative risk assessment on each action. For a similar mechanism, Dimmock et al. (2004) suggest a Prolog-based risk and trust decision-making mechanism, in which each action is checked against a risk function for the current state, ensuring that it is below a certain threshold. Molloy et al. (2008) implement authorization through a trading or auction metaphor, employing a risk calculation function. These models use quantitative methods

[1]The Policy Override Calculus was first proposed at the SIN conference 2010 (Bartsch, 2010b) and appears here in a slightly revised form.

to automatically judge about the risk of actions and, thus, authorization. In practice, quantitative risk assessment is missing the necessary reliability for automatic decision-making in many contexts because of insufficient input data quality and many speculative factors (Straub and Welke, 1998). Qualitative methods are also more common in practice. Therefore, this chapter focuses on qualitative decision-support, which should be more practical.

In the broader realm of information security, other decision-support approaches have been proposed. For example, in the Quality of Security Service (Irvine and Levin, 2000), the functional user chooses a variable amount of security according to the current needs and balances with respect to performance and fidelity. Beresnevichiene et al. (2010) use a utility function to model the economic benefit from security investments. Taking security usability into account, Parkin et al. (2010) suggest a tool to allow information security officers to choose an adequate password policy. These approaches lack the fit for the specific problems of deciding on authorization, though.

F.2. Risk management

Risk assessment is the second major building block for information security decisions. Systematic evaluation of system security through threats and security features has a long history (Hoffman et al., 1978). The underlying risk assessment approaches can be – again – categorized according to their quantitative or qualitative nature. Quantitative approaches, based on numerical calculations of values and probabilities, have for example been proposed for information security by Guarro (1987). Post and Diltz (1986) suggest a stochastic dominance approach to quantitative risk analysis to more precisely compare different mitigation options. Jaisingh and Rees (2001) describe the Value at Risk methodology, which results in a monetary evaluation of the current risk. More common in practice is qualitative risk assessment, such as the example given in ISO/IEC 27005:2008 (2008), that results in a relative risk estimation. It is also common to combine qualitative and quantitative approaches, for example by combining several risk assessment methods and only using quantitative calculations in the last step as discussed by Rainer et al. (1991). Karabacak and Sogukpinar (2005) use questionnaires to collect qualitative inputs, which are then combined with quantitative methods to normalize responses.

There is a wealth of publications available on risk management (Baskerville, 1993; Campbell and Stamp, 2004). First, there are national and international standards on risk management. ISO/IEC 27005:2008 (2008) provides a general framework for risk management in IT systems. The U.S. FIPS-65 standard, withdrawn in 1995, applied the well-known quantitative approach “Annualized Loss Expectancy” (ALE: NIST, 1975). The most widely-used management approaches are of qualitative nature, though. Among other problems, quantitative estimations suggest a precision that is not realistic with the available input (Stoneburner et al., 2002; Peltier, 2005). For example, humans are very imprecise when estimating frequencies of occurrences (Gilovich et al., 2002) and risk estimations depend on the context (Borge, 2001; Adams, 1999). Also, many aspects, such as attacker motivation in case of insider threats, are difficult to quantify. A well-known qualitative approach is the NIST Special Publication 800-30 (Stoneburner et al., 2002) that supersedes the quantitative FIPS-65 standard. Other wide-spread approaches are CRAMM (The CRAMM Manager, 2005) and OCTAVE (Alberts et al., 2003). These standard methods enact very similar processes: They start with a qualitative valuation of assets or impacts of incidents. Then, threats and vulnerabilities are identified and categorized with likelihood estimations, again by way of qualitative values. In a risk assessment step, the inputs are combined into risk estimations and, in the risk management part, mitigations are chosen (United States General Accounting Office (GAO), 1999). To further support the risk assessment, the manual aggregation can be converted into calculations. There are primarily

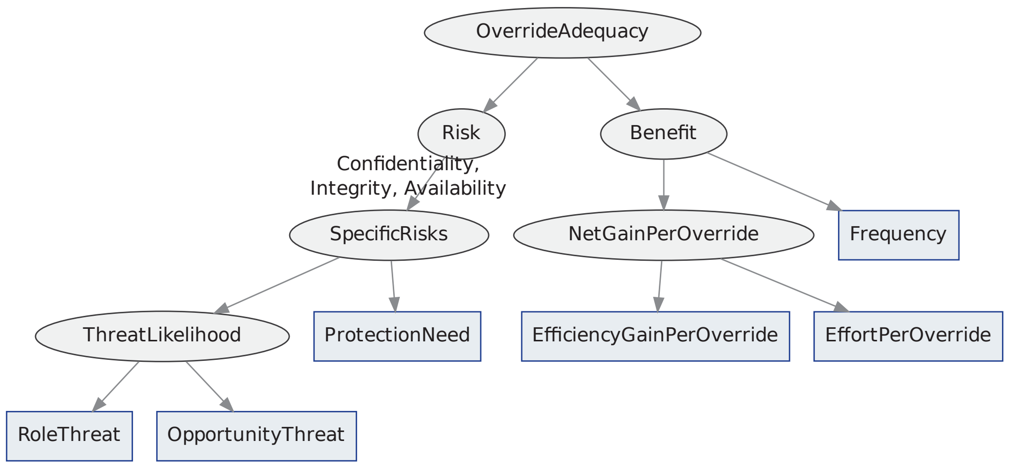


Figure F.1.: Calculus formula overview

two options for risk assessment: either quantitatively, through calculations of values at risks and probabilities of incidents, or qualitatively (Alter and Sherer, 2004).

The risk management processes are very powerful methods for general assessment of risks and may serve as a basis for the input to more specific risk assessment. However, to derive the appropriate override privileges – the objective of this chapter – the standard processes operate at a too high-level and are too coarse. Specifically, we, for example, need to differentiate between different roles. Moreover, to reduce the effort, we need to focus on relevant threats.

F.3. A calculus for the Policy Override configuration

Accounting for the above remarks on prior work, the proposed policy override calculus needs to aim for qualitative evaluation of the relevant risks from specific granting of authorization: In the policy override calculus, the adequacy of override is derived by role and override extent from qualitative data collected from domain and security experts. The calculus does not employ a quantitative assessment because of the problems indicated above. For the policy override calculus, a qualitative and relative risk that allows one to rank the risks should suffice as decision support. The calculus estimates the additional risk for specific extents when compared to the default authorization of a specific role.

The primary factors of the override adequacy calculus are the risk and benefit associated with specific override extents. As depicted in Figure F.1, both risk and benefit are derived from a number of inputs that are combined through formulae. In risk management, the implementation of possible mitigations are often decided based on a risk/cost relationship (Hoffman et al., 1978). Similarly, we balance risks against potential benefits for the policy override calculus, thus replacing costs with the benefits of policy override. The reasoning is that the lost benefit from not allowing override is analogous to the costs of mitigating the risk caused by policy override. By not allowing policy override, the potential override benefits are lost, but, equally, the risks mitigated. First, we will consider the risks that an organization is exposed to when granting override options to users. The qualitative values that are collected for the input components are either *Normal*, *High* or *Very High*, abbreviated “N”, “H” and “V” in the tables.

		Protection Need		
	⊗	Normal	High	Very High
Threat	Normal	N	N	N
	High	N	H	H
	Very High	N	H	V

Table F.1.: Specific Risk: (Protection need) ⊗ (Threat likelihood)

F.3.1. Risk

In information security, it is common to aggregate the individual risks for the security objectives confidentiality, integrity and availability into a single risk value, as suggested by Stoneburner et al. (2002) and ISO/IEC 27005:2008 (2008). The policy override calculus follows this approach. The *Risk* is a function of a system user's role and a specific privilege extent, for example permitting access to only individual objects of a type or all related to an organization's branch.

$$\begin{aligned} Risk(role, extent) = \propto\, (&SpecificRisk(\text{Confidentiality}, role, extent),\\ &SpecificRisk(\text{Integrity}, role, extent),\\ &SpecificRisk(\text{Availability}, role, extent)) \end{aligned}$$

Following Lenstra and Voss (2004), the aggregation operator ($\propto$) is defined as the maximum of the individual risks. The individual risks do not need to be weighted in this aggregation as weights are implicitly present in each individual risk's value through the protection need, as described below.

Specific Risk For each security objective, confidentiality, integrity and availability, the *Specific Risk* is calculated for each user role and override privilege extent. In risk models, risk is typically defined as the expected loss resulting from a threat as calculated from the product of an incident's potential damage and its likelihood (Stallings and Brown, 2008). Following this approach for the calculus, the *Specific Risk* is derived as the individual risk to the security objective from the *Protection Need*[2] as the expected impact and the *Threat Likelihood* as the likelihood of the incidents.

$$\begin{aligned} SpecificRisk(objective, role, extent) = \mathsf{ProtectionNeed}(objective, extent) \otimes\\ ThreatLikelihood(objective, role, extent) \end{aligned}$$

The $\otimes$ operator is defined as a look-up matrix shown in Table F.1 that follows loosely the *Risk-Level Matrix* proposed in the NIST 800-30 guidelines, in which the risk levels are determined through multiplication (Stoneburner et al., 2002).

One operand for the calculation of the *Specific Risk* is the impact of incidents that is approximated through the *Protection Need* as defined in *BSI IT-Grundschutz*, the German information security standard for baseline protection (Bundesamt für Sicherheit in der Informationstechnik (BSI), 2008). In the standard, the protection levels are defined as follows:

- *Normal:* The impact of any loss or damage is limited and calculable.
- *High:* The impact of any loss or damage may be considerable.
- *Very High:* The impact of any loss or damage could be of catastrophic proportions threatening the survival of the organization.

[2] Throughout the formulae, sans-serif fonts are used to signal direct input data.

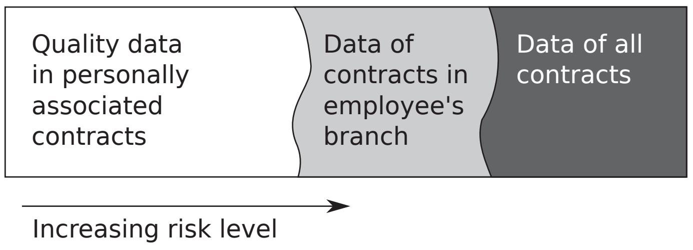

Quality data in personally associated contracts
Data of contracts in employee's branch
Data of all contracts
Increasing risk level

Figure F.2.: Example of risk varying according to the privileges of a role

The IT-Grundschutz standard defines the protection need levels for the impacts regarding the violation of laws, regulations or contracts, privacy rights, physical injury, the ability to complete tasks at hand, internal and external effects and financial consequences (Bundesamt für Sicherheit in der Informationstechnik (BSI), 2008).

The protection need values are differentiated according to the exposure extent. A company with several branches may be able to absorb the losses of a single incident at one branch. On the other hand, if the whole company is affected, the damage might be disastrous. The potential damage thus depends on the extent of data exposed. An example for the different risk levels by privilege extent is shown in Figure F.2. In the example, the disclosure of only some contracts' quality data has a much smaller impact than the disclosure of all of the company's contracts.

Threat Likelihood The second operand for the *Specific Risk* is the *Threat Likelihood*, defined as the likelihood of the occurrence of an incident. The Threat Likelihood is calculated for a given security objective and role for the case that a specific policy override extent has been assigned to that role:

$$\begin{aligned} \mathit{ThreatLikelihood}(objective, role, extent) = \\ \mathsf{RoleThreat}(role) \odot \\ \mathsf{OpportunityThreat}(objective, extent) \end{aligned}$$

Two aspects are taken into consideration to arrive at an estimation of the threat likelihood: the threat induced by the personal factors of users in specific roles (*Role Threat*) and the differences in threat caused by different privileges (*Opportunity Threat*). This separation is based on criminological models on insider threat. The insider threat models in literature separate the aspects of the capability of users, the motivation and the opportunity in the information systems (CMO model) (Schultz, 2002; Wood, 2000; Willison, 2006; Willison and Backhouse, 2006). The capability is of minor interest for the risk in the case of policy override, since the risks are expected to originate from ordinary activities similar to an insider's daily routine rather than sophisticated attacks. On the other hand, motivation and opportunity of users are of high importance. To facilitate the collection of input data for opportunity and motivational factors with a reasonable effort, these factors are collected by the two separate input types, Role Threat and Opportunity Threat. This separation can also be found with Magklaras and Furnell (2002), who suggest a similar insider threat model that depends on reasons and system role. Schultz (2002) also uses the privilege of users as attributes of insiders in his CMO model, further backing the modeling of the Opportunity Threat in the calculus.

Role Threat As described above, the Role Threat captures threat likelihood aspects that differ regarding the system roles. Intuitively, for personal factors, experts need to evaluate how trustworthy a group of employees in a specific role is (Shaw et al., 1998). Magklaras and Furnell (2002) define

	Intentional	Accidental
Threat likelihood factors	Trust in employees; Employee Satisfaction; Seniority; Previous issues with employees of specific roles; Deterrents	Frequency of application usage; Application usability; Work environment; Training level
Threats to confidentiality	Selling of data	Inappropriate data handling
Threats to availability, integrity	Intentional data manipulation	Accidental removal, modification of data

Table F.2.: Likelihood factors and threats related to the role-threat likelihood

a taxonomy of insider threat considering the user's system role, reasons of attack and the attacks' consequences. We follow their approach on the reasons of attack and differentiate *accidental* from *intentional* threats. On the intentional side, following the CMO model, the role threat foremost relates to the motivation that causes insiders to harm the employer. The NIST 800-30 risk management guide lists human threat sources (Stoneburner et al., 2002). From practice, Randazzo et al. (2005) assembled a detailed study on insider threat incidents in the banking and financial sector and describe different motivations. Further motivations are described in a U.S. CERT report on critical infrastructure sabotage, including disgruntled employees, previous sanctions and personal predispositions that may be mapped to users in roles (Moore et al., 2008). According to Cappelli et al. (2009), insiders tend to occupy foremost lower-level, non-technical positions, another factor that supports the correlation between the system roles and threat likelihood. Similarly, a study by the Association of Certified Fraud Examiners (ACFE) (2006) of 1134 fraud and abuse incidents, not limited to computer system activities, shows that the perpetrator's position and department affects incident frequencies.

In addition to motivational factors, threats from accidental incidents need to be considered. Accidental aspects can be derived from previous incidents for specific roles, the training level of employees in a role and similar aspects (Bedny and Meister, 1999; Endsley, 1995). A selection of aspects and typical incidents relating to the system role that security experts need to take into account for the Role Threat estimation is listed in Table F.2. The qualitative threat values for the Role Threat correspond to an estimation of the likelihood of incidents. The input value *Normal* stands for highly unlikely, *High* for unlikely and *Very High* for likely that an incident occurs.

Opportunity Threat The second dimension of the insider threat likelihood is the opportunity that is caused by the extent of privilege in override cases. The Opportunity Threat is differentiated by privilege since the differences of data exposure and of possible actions in the system influences the threat likelihood. In the criminological CMO model, opportunity is one of the three primary categories. According to the Rational Choice Perspective from the Situational Crime Prevention theory, crimes are deliberate and purposive even though the decisioning of perpetrators may be imperfect (Willison and Backhouse, 2006). Thus, the motivation might be higher with increased opportunity that causes, for example, higher interests for the data for espionage and higher impact in case of sabotage. Financial gain motivates most perpetrators (Randazzo et al., 2005). In an insider threat prevention guide, Cappelli et al. (2009) list motivations for theft and modification for financial gain and business advantage, stating that insiders acted mostly for financial gain by stealing and selling data or modify data for their own or friends' profit. Another insider threat aspect is that "opportunity makes a thief" as described for insider threats in the Audit Commission (1994) report.

The Opportunity Threat is estimated independently from personal factors, which are already considered for the Role Threat. Moreover, the opportunity threat likelihood depends only on the mo-

	Threat	Likelihood factors
Confidentiality	Selling data for industry espionage	Data value to competitor; Risks in selling
Integrity, Availability	Data manipulation	Gain from fraud; Value of sabotage to competitor

Table F.3.: Likelihood factors and threats related to the opportunity-threat likelihood

		Role Threat		
	$\odot$	Normal	High	Very High
Opport.	Normal	N	H	V
	High	N	V	V
	Very High	H	V	V

Table F.4.: Threat Likelihood: (Role Threat) $\odot$ (Opportunity Threat)

tivational factors for harmful actions so that only intentional aspects are taken into account. The varying impacts from accidental incidents caused by different privileges are already taken into account for the Protection Need. A selection of relevant aspects of threats to Confidentiality, Integrity and Availability are given in Table F.3.

Threat Likelihood Aggregation Automatically aggregating the Role Threat and Opportunity Threat values is a key challenge. The aim is to aggregate the different input dimensions of the threat likelihood for specific roles with the values for the specific privileges. The proposal of the look-up operator $\odot$ is shown in Table F.4. It is based on the assumption that "opportunity makes a thief" (Audit Commission, 1994). The operator result starts out from the threat estimation for the specific role, which is then modified according to opportunity effects. For Very High opportunities, even solid employees might be tempted to commit a malicious insider act so that the aggregated threat is High. For High and Very High Role Threat, the aggregated threat increases with higher opportunities.

F.3.2. Benefit

While there are potential risks related to allowing policy override, an organization may significantly benefit from the increased flexibility. The benefits are estimated with the following formula for users of a role with a specific override extent configuration:

$$\begin{aligned} \mathit{Benefit}(role, extent) = {} & \mathsf{Frequency}(role) \times \\ & \mathit{BenefitPerOverride}(role, extent) \end{aligned}$$

The benefit is calculated from the frequency of override incidents and an estimate of the benefit that may be achieved on average from each override incident. Following quantitative calculations, the $\times$ operator acts similar to multiplication as shown in Table F.5.

Frequency The daily routine of system users often varies significantly according to the role. In the same way, each role may have different frequencies of situations where quick responses are needed. Thus, domain experts estimate the frequency of override cases by role. Aspects to consider should include the structuredness of daily routine, the frequency of unforeseeable incidents and whether there is direct customer contact. Existing application log data and interviews can provide the basis for this factor.

		Override Frequency		
	×	Normal	High	Very High
Benefit	Normal	N	H	H
	High	H	H	V
	Very High	H	V	V

Table F.5.: Benefit: (Benefit per Override) × (Override Frequency)

		Effort per Override		
	−	Normal	High	Very High
Gain	Normal	N	N	N
	High	H	N	N
	Very High	V	H	N

Table F.6.: Net Gain per Override: (Gain per Override) − (Effort per Override)

Benefit per Override The second component for the calculation of the benefit is the benefit that may be gained in each override incident:

$$\begin{aligned} \mathit{BenefitPerOverride}(role, extent) = {} & \mathsf{EfficiencyGainPerOverride}(role, extent) - \\ & \mathsf{EffortPerOverride} \end{aligned}$$

Benefit Per Override follows the intuition that each case of policy override brings benefit, but also causes auditing effort for the superior who is responsible for auditing the case. Depending on the auditing implementation, auditing effort may reduce the benefit from policy override cases, particularly with high numbers of policy override cases. Accordingly, Benefit Per Override derives the net gain that is achieved in each override case from the efficiency gain in each case and the auditing effort per case (−). Efforts estimated as *High* and *Very high* reduce the net gain similar to subtraction as shown in Table F.6.

Efficiency Gain per Override The efficiency gain per override action is estimated separately for each role and permission extent. Benefit scenarios help with this step. In the example of a quality operator in Figure F.2, domain experts may foresee cases in that the company may profit to a larger extent from the access of data of the whole assigned branch because the operator might jump without any bureaucracy from one local contract to another. On the other hand, the experts may find it unlikely that the same employee would need to switch branches quickly. Aspects to be considered are the company gain per override, the time saved by not requiring a formal delegation of permissions and the likelihood of work in the context of a specific extent. The qualitative input values should only be above *Normal* if there is additional gain when compared to the original privilege extent of a role. In addition to benefits from the functional perspective, it is also necessary to consider the effort from saved administrative tasks, which is often significant (Gallaher et al., 2002). Instead of assigning new permissions for temporarily required authorization, the functional user may resort to policy override, reducing the administrative overhead.

Effort per Override The additional effort that the company needs to invest per override is foremost caused by the need to audit override actions afterwards. For the Policy Override Calculus, domain experts estimate the typical effort that is spent on auditing the actions per override.

	$\bowtie$	Aggregated Risk Normal	High	Very High
Gain	Normal	N	L	L
	High	H	N	L
	Very High	V	H	N

Table F.7.: Override Adequacy: (Aggregated Risks) $\bowtie$ (Net Gains)

F.3.3. Override adequacy

From the risk and benefit estimations, the policy override adequacy is determined as decision support for the policy override authorization configuration for each role with a specific override extent assigned:

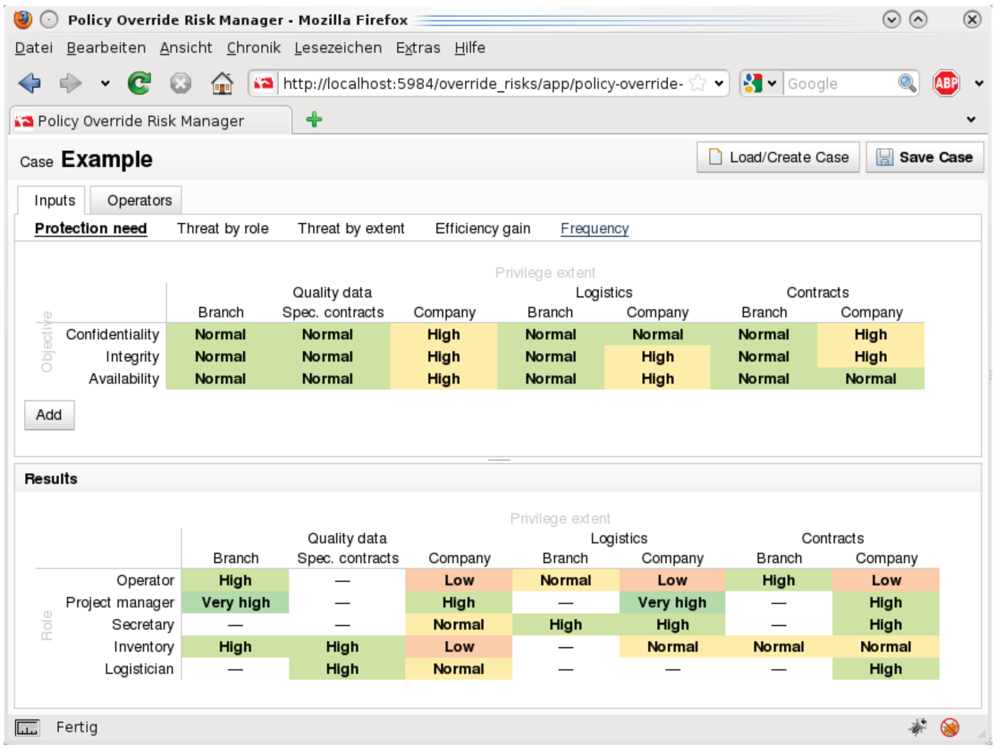
$$Adequacy(role, extent) = Risk(role, extent) \bowtie Benefit(role, extent)$$

The $\bowtie$ operator balances risks with benefits. The interpretation of the outcome of this operator depends on the company policy with regard to acceptable risks. The U.S. FIPS 191 guideline suggests to calculate risk/cost relationships for the balancing of security-mechanism costs against risks based on qualitative data (NIST, 1994). Similarly, the look-up table for the operator definition given in Table F.7 is derived from the risk/benefit relationship. In this case, the costs are the lost benefits from not employing override as a way to mitigate the risks from override actions. The adequacy values are calculated by quantifying Normal as 1, High as 2 and Very High as 3. The result from the ratio $a = Gain/Risk$ is interpreted as Low for values $a < 1$, Normal for $1 \leq a < 1.5$, High for $1.5 \leq a < 2.5$ and Very High for values $a \geq 2.5$. Results from this operator do not offer an absolute estimation of the override adequacy. Rather, the results help by providing an order of the most suitable role/privilege extent combinations.

F.4. Preliminary case studies

For the evaluation of the proposed calculus, the Web-based Override Risk Assessment tool was developed, depicted in Figure F.3, to facilitate the collection of input data and calculation of the override adequacy. In the override risk tool, input tables are provided for each of the input types. If necessary, even the operator tables may be modified here. The resulting override adequacy is continuously calculated and displayed in the bottom area, so that stakeholders who provide the input for the evaluation can directly see the consequences of inputs. Inputs and results are colored according to the qualitative value to ease the collection of data and interpretation of results.

We evaluate the proposed override calculus in two distinct environments as case studies. The context of the first evaluation is the medium-sized enterprise from the policy override study described in Section 8.3. With policy override in place, the employees have already built up experience with the policy override concepts. To evaluate the developed calculus, the author worked together with the company's managing director to collect the necessary data. Parts of the results from the analysis with the Override Risk Assessment tool are shown in Table F.8 by role and privilege extent. The privileges correspond to application areas in the authorization policy. Five roles of employees from the authorization policy are shown. Employees with the *Inventory* role manage stock in the warehouse. *Logisticians* are office workers concerned with logistics coordination. *Operators* perform the quality control tasks. *Secretaries* and *Project managers* fulfill administrative and project-specific


Policy Override Risk Manager - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://localhost:5984/override_risks/app/policy-override-
Google
Policy Override Risk Manager
Load/Create Case
Save Case
Inputs
Operators
Add
Fertig

Case **Example**

Protection need | Threat by role | Threat by extent | Efficiency gain | Frequency

Privilege extent

Objective	Quality data: Branch	Quality data: Spec. contracts	Quality data: Company	Logistics: Branch	Logistics: Company	Contracts: Branch	Contracts: Company
Confidentiality	Normal	Normal	High	Normal	Normal	Normal	High
Integrity	Normal	Normal	High	Normal	High	Normal	High
Availability	Normal	Normal	High	Normal	High	Normal	Normal

Results

Privilege extent

Role	Quality data: Branch	Quality data: Spec. contracts	Quality data: Company	Logistics: Branch	Logistics: Company	Contracts: Branch	Contracts: Company
Operator	High	—	Low	Normal	Low	High	Low
Project manager	Very high	—	High	—	Very high	—	High
Secretary	—	—	Normal	High	High	—	High
Inventory	High	High	Low	—	Normal	Normal	Normal
Logistician	—	High	Normal	—	—	—	High

Figure F.3.: Override Risk Assessment tool

management duties, respectively. Due to space considerations, the individual input values are not shown.

As shown in the result table, the calculus only outputs values for permissions not already assigned for the conventional *need-to-know* permission. The results are very heterogeneous. For example, the results strongly suggest policy override for *Project managers* with two *Very high* and two *High* results. *Operators*, on the other hand, should only receive override privileges for some extents as indicated by the *Low* values. After the data collection, the author interviewed the managing director from the studied company on the plausibility and relevance of the results. The managing director agreed with the results and decided to adapt the original override policy. The override calculus thus supported the domain experts' intuition in the complex evaluation of risks and gains.

The second evaluation context is a large enterprise from the food industry with about 1,000 employees. In this case, an IT management person used the Override Risk Assessment tool with assistance. While the first evaluation focused on usefulness of the adequacy results, the second case was aimed at the usability of the calculus. Primarily, the evaluation explored the necessary effort and the comprehensibility of the involved risk and benefit concepts. For that reason, the policy override calculus was only collected for part of the company's roles and an interview was conducted on the studied context and the participant's impressions. In the studied environment, no explicit risk assessment had been conducted for authorization policy decisions before. Thus, the estimation of protection needs and threat likelihoods were initially perceived as a challenge. Still, the concepts that are used in the calculus could be applied. One minor issue was that the available levels to choose from are abstract and natural-language categories instead of "Normal" and "High" could improve the process. In the company's ERP system, only a flat role structure has been implemented, resulting in a large number of roles. The participant thus saw a high effort required to fully analyze the company with the policy override calculus. Here, similar to the initial role engineering, external support might

	Privilege Extent						
Override	Logistics		Contracts		Quality data		
Adequacy	Branch	Company	Branch	Company	Spec. contracts	Branch	Company
Inventory		N	N	N	N	N	L
Logistician				H		H	N
Operator	N	L	H	L		H	L
Secretary	H	H		H			N
Project manager		V		H		V	H

Table F.8.: Example result for the override adequacy by role and privilege extent

be necessary. On the other hand, he was unsure whether the role-based distinction was fine-grained enough to estimate personal threat likelihoods.

From the two small-scale evaluations, we can draw the conclusion that the current design of the calculus and implementation of the Override Risk Assessment tool may be helpful in small to medium enterprises. For large enterprises, the effort may be high, at least when there is no structured risk assessment data available. Still, further research and larger-scale evaluations are necessary to show the broader validity of the proposed approach.

F.5. Conclusion

This chapter addresses the challenge of formulating the override policy. Since not only the need-to-know is considered, but two levels of risks need to be balanced against the benefits, the task is more complex than defining the default policy. To support this task, a novel calculus estimates the adequacy of policy override for specific roles and extents. The qualitative result is derived from indicator inputs, including the protection needs, threats, and benefits.

The preliminary evaluation of the calculus shows that it may be viable for small to medium enterprises for insights into the adequate extent of policy override. Domain experts of a medium-sized company could improve the policy-override configuration from the calculus results and gave positive feedback on the usefulness of the results (P1). In large enterprises, the implementation of an override risk assessment is more challenging, though. Regarding the aim of the calculus to explore how policy decisions can be guided (Req 11), we can derive the following insights on open research questions:

Q11.1 *What kind of guidance for authorization-related decisions is effective and efficient?* Adequate guidance depends strongly on the context. In small to medium enterprises, it is plausible that a risk-based policy-wide calculus of permission adequacy is useful in guiding the decisions and reducing the decision effort (P2). For environments with a large number of functional roles and unstructured permissions, the calculus approach may require too much effort.

Q11.2 *How can the information that is required for decision guidance be elicited?* The calculus took a one-time approach to input collection, which may work for smaller contexts. However, the approach can be prohibitively expensive for larger contexts. For the latter, a more process-oriented approach that integrates with existing procedures for policy changes might be more adequate.

Future work involves the further development and evaluation of the calculus on a larger scale. This may include re-using the calculus results for the creation of conventional policies, using different levels of abstraction for input collection, and experimenting with directly employing the calculus result for override permissions without expert intervention. Similarly to improve the viability for larger environments, we should consider re-using data from Information Security Management Systems and existing risk assessments to reduce the effort of collecting the input data.

G. Principles applied in the artifacts

In this dissertation, we explored the usefulness of eight Authorization Principles by developing six artifacts. Each of the artifacts employed a subset of the principles as summarized in Table 6.4. The respective insights on the principles from their application in the artifacts are listed below in Table G.1. Moreover, the evaluation of the artifacts resulted in numerous recommendation that relate to principles as listed in Table G.2. Both aspects are further discussed in Section 11.1.5. Lastly, we discuss how the principles apply to contexts outside of the scope of this thesis.

Artifact	Coverage	Insights	Limitations (argument)
P1 Interaction and participation			
Enforcement	Integration of policy changes and application development	Better reactions to problems of staff due to separation of concerns	Single case study (Objective empirical/analytical)
Policy	Participation of functional staff in policy discussion	Difficult due to cognitive load, unknown semantics; context-dependent	Needs validation in practice (Subjective empirical)
Change support	Participation of functional staff, interaction between perspectives	Helps in the communication of change options and consequences, but context-dependent	Limited policy expressiveness, needs validation in practice, (Subjective empirical)
Policy override	Integration of functional staff in policy making for the own permissions	Need to support functional staff's understanding of what is possible in override – and what is appropriate	Single case study (Subjective empirical)
INDUSE	Interactions between perspectives	Defined interrelations can be useful and can be indicated by INDUSE	Needs validation in practice (Subjective empirical)
Decision support	Participation of functional staff in decisions; foster interactions of policy maker and author	Context-dependent; functional staff can provide decision factors; improved communication, transparency, and security awareness; support discussions	Prototype only, needs validation in practice (Subjective empirical)
P2 Reduced burden			
Enforcement	Implementation effort of controls	Reduced effort from the separation of concerns	Single case study (Objective empirical/analytical)
Policy	Comprehending and reviewing policy	Adequate management-mode reduces effort, depends on context; policy semantics are problematic	Needs validation in practice (Subjective empirical)
Change support	Identifying and choosing policy-change option	Improvement over interaction with policy, but abstraction can be problematic, depends on context	Needs validation in practice (Subjective empirical)
Policy override	Interference with functional tasks and effort of policy changes	The override cases indicate a reduction of both efforts, but caused additional effort for policy makers	Single case study (Objective empirical/analytical)
Decision support	Effort of change request (functional staff) and comprehensive decisions (policy makers)	Needs comprehensible factors, honor actual processes, efficiency through participation, flexibility in collection, integration in process	Prototype only, needs validation in practice (Subjective empirical)

Table G.1.: Insights from the application of the Authorization Principles (continued on next page…)

Artifact	Coverage	Insights	Limitations (argument)
P3 Concreteness			
Change support	Concrete policy-change goals and alternatives	More effective and efficient, but also leads to problematic abstraction	Needs validation in practice (Subjective empirical)
Decision support	Decision-factor collection and presentation	Contextualizing factors is necessary for comprehensibility	Prototype only, needs validation in practice (Subjective empirical)
P4 Multidisciplinary			
Enforcement	Controls integration: Software engineering, socio-technical	Improved comprehensibility of controls, changeability of enforcement	(Artifact development)
Policy	Management-mode design: Socio-technical, inf. security	Improved comprehensibility of model and representation	(Artifact development)
Change support	Algorithm, tool design: Socio-technical, information security, artificial intelligence	Increases concreteness of changes	Context-dependent adequacy (Artifact development)
Policy override	Model, implementation: Socio-technical, information security	More effective security through informal rules and mechanisms (limits, monitoring)	Requires socio-organizational measures (Artifact development)
INDUSE	Process model, activities: Organization sciences, software engineering, inf. security	Effective relation of activities from diverse perspectives	Further organizational research for prescriptive process model (Artifact development)
Decision support	Factors, prototype: Risk and decision, socio-technical	Contextualizing factors (risk communication, mental model)	(Artifact development)
P5 Individuals' behavior			
Policy override	Offer convenient alternatives to circumvention	Reduced circumventions, but excessive override use	Single case study (Objective/-subjective empirical)
Decision support	Influence behavior through participation	Behavior may be influenced by transparency, improved communication	Prototype only, needs validation in practice (Subjective empirical)
P6 Security awareness			
Decision support	Awareness through participation in decision process	Indications of improved security awareness through completing form	Prototype only, needs validation in practice (Subjective empirical)
P7 Design for dynamics			
Enforcement	Separation of concerns of policy and functional changes	Reduced necessity of functional or policy changes	Single case study (Objective empirical/analytical)
Policy override	Flexibility of the model	Reduced disruptions of functional work and policy-changes quantity	Single case study (Objective/-subjective empirical)
INDUSE	Interaction and procedures for system and policy changes	The process must fit the context's dynamics and INDUSE can indicate adequate processes	Needs validation in practice (Subjective empirical/analytical)
Decision support	Decision support for dynamic changes	Context-dependent; needs to be integrated in procedures for low overhead	Prototype only, needs validation in practice (Subjective empirical/analytical)
P8 Tailor for context			
Policy	Adequacy of management mode	Depends on individual's background, organizational context	Needs validation in practice (Subjective empirical)
Change support	Adequacy of policy change-support	Depends on complexity of policy, model	Needs validation in practice (Subjective empirical)

Table G.1.: Insights from the application of the Authorization Principles (continued on next page...)

Artifact	Coverage	Insights	Limitations (argument)
INDUSE	Adequate processes for contexts; adaptability of process model	Contexts require tailored processes (e.g. formality, centralization), as described by INDUSE for a broad range	Needs validation in practice (Subjective empirical/analytical)
Decision support	Context-dependency of supporting decisions	Highly context-dependent; needs to fit process and to be flexible regarding security needs	Prototype only, needs validation in practice (Subjective empirical)

Table G.1.: Insights from the application of the Authorization Principles in artifact development

Recommendation	Principles	Additional aspects
Authorization Problems Study (Section 5.2)	Subjective, qualitative	
Foster acceptable circumventions	P6 P5 P7	Satisfaction, decision-making
Monitor circumventions and the use of additional controls	P5 P2	Observation, consequences
Define and communicate procedures	P1 P5	Procedures
Reduce the (perceived) change lead-time and change effort	P8 P5	Procedures
Adjust the degree of centralization and formalization	P8 P6	Procedures
Provide high-level policies and guidance on decisions	P2 P6	Decision-making
Increase the expertise and awareness of policy makers	P6 P3 P1	Decision-making
Improve authorization models and management tools	P2	Intuitive UI
Modes of policy changes (Section 7.3.5)	Subjective, qualitative	
Support policy decisions	P2 P3 P1	
Take the context into account	P2 P8	
Support the problem-solving strategies	P2 P8	
Improve the user experience	P2 P5	Satisfaction, intuitive UI
Policy override in practice (Section 8.3.5)	Objective/subjective, qualitative	
Awareness for appropriate override usage and availability	P6 P5	Intuitive UI
Adequate tool support	P2 P7	
Organization for consequences	P2 P6 P5 P8	Observation, proc., consequences
Decision support	P2	Decision-making
Authorization processes in practice (Section 9.5.4)	Subjective, qualitative	
Integrate functional staff	P1 P8 P5	Decision-making, procedures
Establish operative activities	P7 P5 P6	Observation, consequences
Integrate systems development	P1 P7 P8	
Improve stakeholder interaction	P1 P8	Procedures
Decision-support study (Section 10.6)	Subjective, qualitative	
Adequately embed collection in procedures	P2 P8 P7	Procedures
Contextualize factors	P8 P3	
Respect the sensitivity of factors	P5	Satisfaction
Offer flexibility in data collection	P2 P8	Flexibility
Reuse existing data	P2	Observation
Support discussions	P8 P1 P5	Decision-making, procedures

Table G.2.: Relation of the Authorization Principles to recommendations

G.1. Applicability of the principles beyond organizational contexts

Whereas we developed the Authorization Taxonomy (cf. Chapter 4) and used it to limit the scope of this thesis to organizational authorization contexts, we already considered briefly for the authorization requirements how they apply to non-organizational example contexts (cf. Sections 4.1.2 and 6.2.2, and Tables 4.4 and 6.3). The primary finding there was that the priorities of the individual requirements are diverse. Most requirements are less critical for contexts of less complexity and

P1 *Interaction and participation*	−	Not for functional/administrative interaction, often less dynamic
P2 *Reduced burden*	+	Diverse expertise, though less complex and dynamic
P3 *Concreteness*	+	Diverse expertise
P4 *Multidisciplinary*	−	Fewer organizational aspects
P5 *Individuals' behavior*	+	Diverse expertise, security needs
P6 *Security awareness*	+	Diverse expertise, security needs
P7 *Design for dynamics*	−	Often less dynamic
P8 *Tailor for context*	+	Very diverse contexts

Table G.3.: Importance of principles in non-organizational contexts

dynamics, but there is no trend of specific requirements being generally of less importance. Continuing from that preliminary analysis, we will now examine analytically how the requirements and Authorization Principles relate to the characteristics of non-organizational contexts – summarized in Table G.3, based on the characteristics described below and the correlation between characteristics given in Table 11.3.

Even though the scope allowed us to make a number of concrete assumptions on the breadth of the context factors, the actual restrictions of the scope were limited: We only allowed for organizational resource owners (not individual authors or entity owners), human principals (excluding e.g. applications as principals), and organizational authorization scopes (not personal, inter-personal, or inter-organizational). Further assumptions relied on an organization as a formative structure for individuals (resource owners, policy makers, policy authors, principals). However, even if the formative structure is absent, individuals are still under the influence of power structures of society – based on morale and ethics, social norms, and laws (cf. e.g. Lessig, 1998).

Generally, authorization measures outside of organizational environments have to fit the context (P8 *Tailor for context*). The *security needs* will often depend on personal characteristics of the *resource owner* (e.g. risk tolerance for privacy) instead of organizational characteristics or an organization's regulatory obligations (Req 3 *Regulatory requirements*). Availability may be similarly important as in organizational contexts since the acceptability of the measure highly depends on appropriate permissions (P5 *Individuals' behavior*). For the *principals*, the primary difference is that we need to consider non-human principals, such as smartphone applications, which we explicitly excluded for the scope of this thesis. Non-human principals need to be judged differently than human ones (e.g. based on the trustworthiness of the application's developer), leading to a different approach to decision-making (Req 6).

In case of the *security expertise*, non-organizational contexts can be considerably different from organizational ones: Whereas organizations generally have the ability of influencing awareness and expertise of stakeholders, society only has indirect and rather remote means (e.g. school education or public awareness-campaigns). Thus the motivation of individuals and comprehensibility of the measure may need to receive increased attention (P5, P6, P2, P3). Conversely, the *complexity* factor can reduce the impact of many problems, since non-organizational context often have lower quantities and dynamics – reducing the effort of decisions (Req 6, Req 11), and the need for expressive models (Req 12), usable policies (Req 13), and P7 *Design for dynamics*.

Regarding design parameters, the *management* measures are less significant, since administrative and functional roles seldom separate. Only the interaction with development will remain, even though to a lesser extent since the distance from development tends to be larger (P1 *Interaction and participation*). Primarily, this removes the necessity for adequate procedures (Req 10) and change requests (Req 9). Generally, organizational measures are less important (P4 *Multidisciplinary*), including reviews and monitoring (Req 7). In case of *enforcement*, while models on power structures –

such as laws and social rules (Lessig, 1998) – apply, we can assume that non-technical enforcement will be less effective without the authority of the organization, in some cases leading to increased need for technical means (Req 1).

Overall, we find a reduced importance for the Authorization Principles related to the complexity and dynamics of the context, but wide variations regarding security expertise and the security needs (cf. Table G.3). This also results in a somewhat reduced significance for integration and participation (P1) since there is often a smaller number of distinct perspectives outside of organizations. Thus, while a large proportion of requirements and principles still apply, the priorities are shifted considerably and warrant further research.