Noname manuscript No.
(will be inserted by the editor)

Eliciting Security Requirements and Tracing them to Design: An Integration of Common Criteria, Heuristics, and UMLsec

Siv Hilde Houmb · Shareeful Islam · Eric Knauss · Jan Jürjens · Kurt Schneider

Received: date / Accepted: date

Abstract Building secure systems is difficult for many reasons. This paper deals with two of the main challenges: (i) the lack of security expertise in development teams, and (ii) the inadequacy of existing methodologies to support developers who are not security experts. The security standard ISO 14508 (Common Criteria) together with secure design techniques such as UMLsec can provide the security expertise, knowledge, and guidelines that are needed. However, security expertise and guidelines are not stated explicitly in the Common Criteria. They are rather phrased in security domain terminology and difficult to understand for developers. This means that some general security and secure design expertise are required to fully take advantage of the Common Criteria and UMLsec. In addition, there is the problem of tracing security requirements and objectives into solution design, which is needed for proof of requirements fulfilment.

This paper describes a security requirements engineering methodology called SecReq. SecReq combines three techniques: the Common Criteria, the heuristic requirements editor HeRA, and UMLsec. SecReq makes systematic use of the security engineering knowledge contained in the Common Criteria and UMLsec, as well as security-related heuristics in the HeRA tool. The integrated SecReq method supports early detection of security-related issues (HeRA), their systematic refinement guided by the Common Criteria, and the ability to trace security requirements into UML design models. A feedback loop helps reusing experience within SecReq and turns the approach into an iterative process for the secure system life-cycle, also in the presence of system evolution.

Keywords Security requirement elicitation, Common Criteria (CC), UMLsec, heuristics, and secure design.

1 Introduction

Modern society heavily relies on networked software systems. These systems are inherently complex and highly integrated, and operate in a heterogeneous environment. This makes the engineering of these systems difficult and exposes them to security risks that may have serious implications, such as threatening the privacy of people and the financial well-being of companies. In addition, these systems must comply to an increasing number of regulations and standards, such as Sarbanes Oxley (SOX), the EU Privacy Directive, national privacy laws, etc.

These serious security risks and the vast amount of legal constraint make it important to address security up-front; i.e., in the early phases of development. However, systematic elicitation, negotiation, and validation of requirements is in general very difficult [24]. An additional challenge is

This work was partly supported by the Royal Society Industrial Fellowship on *Automated Verification of Security-Critical Software (VeriSec)*, the Royal Society Joint International Project on *Model-based Formal Security Analysis of Crypto-Protocol Implementations*, the EU FP7 Integrated Project *Security Engineering for Lifelong Evolvable Systems* and the *German Research foundation(DFG project InfoFLOW, 2008-2011)*.

Siv Hilde Houmb
Connected Objects Laboratory, Service Platform Group, Telenor GBDR, Otto Nielsens vei 12, 7004 Trondheim, Norway
E-mail: siv-hilde.houmb@telenor.com

Shareeful Islam
Fakultät für Informatik, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
E-mail: islam@in.tum.de

Eric Knauss · Kurt Schneider
Software Engineering Group, Leibniz Universität Hannover, Welfengarten 1, D-30167 Hannover, Germany
E-mail: {eric.knauss,kurt.schneider}@Inf.Uni-Hannover.de

Jan Jürjens
Chair for Software Engineering(14), Technische Universität Dortmund, Baroper Straße 301, 44227 Dortmund, Germany
E-mail: http://jurjens.de/jan

to ensure that the set of identified security requirements is consistent and complete; i.e., that no necessary security requirements are left undiscovered, and that the set of security requirements jointly indeed enforces the security needs.

1.1 Security requirements challenges

During requirements elicitation, vague and un-documented demands and desires from multiple stakeholders must be detected and merged with more conscious and documented requirements. This task is inherently difficult due to the different backgrounds, tacit assumptions [62], and styles of communication [52] among stakeholders. Experience has shown that stakeholders are often unable to voice their requirements [22]. There is an additional challenge in security: many stakeholders are not even aware of the threats and the consequent losses they may face.

Standardization organizations such as the European Telecommunications Standards Institute (ETSI) are committed to include security requirements in their standards. The industrial background of the work presented in this paper is the IPTV standardization activities undertaken by ETSI (see Section 4.1), and in particular the IPTV security requirements elicitation activities. While there usually is technical and architectural expertise available, security expertise is a scarce resource, also among the member companies of ETSI. Therefore, security knowledge and experience need to be made available to non-security technicians.

Security knowledge and experience can be explicit or tacit. Explicit information is either documented or somehow easily accessible. For example, security standards, security checklists and vulnerability bulletins contain explicit experience and knowledge. Security experts usually have experience and tacit knowledge about security issues. They are not aware of their tacit and unspoken experience, but they use it all the time. Tacit knowledge is often needed to make best use of explicit knowledge. Furthermore, the major challenges in in experience and knowledge management are the elicitation and the reuse of tacit knowledge. Ideally, security experts can turn security principles into guidelines, e.g., for identification and authentication of users to a system. Developers and other stakeholders can then use these guidelines and by that reduce the need for involvement of security experts. However, in many cases this will not be possible: e.g. key management for cryptographic operations requires a deep understanding of security that can hardly be captured by checklists. Thus, there is a need for additional supportive constructs.

Furthermore, the terminology used in many documents, even though they present knowledge explicitly, is not intended to be understood by security novices. There are many reasons for this, one being that there are underlying processes and methodologies or ways of thinking or working that are assumed in these documents, i.e., not stated explicitly. One such source is the security standard ISO 15408:2007 Common Criteria for Information Technology Security Evaluation [17], here referred to as the Common Criteria. The Common Criteria provide support in building a secure system by offering classes of functional security components that a developer can select from. These classes address security principles such as identification and authentication, encryption, security management, and audit. Together with the security refinement guidelines, these classes support a security expert in eliciting security requirements. However, the Common Criteria are formulated using domain-specific security terminology, which makes it hard for non-security developers or stakeholders to take full advantage of the guidelines in the standard. Hence, the security requirements elicitation support is inherent in the standard, but requires security knowledge and experience to be accessed. So far there is no thorough description of the security requirements elicitation process implied in the Common Criteria, nor an extensive demonstration of its use in practise, and both are required for the developer community to benefit from this tacit information.

In addition, to ensure correct employment of security principles in a secure design, the security requirements need to be unambiguous and specific. Therefore, a Common Criteria based security requirements elicitation methodology was standardized by ETSI's standardization program Telecoms & Internet converged Services & Protocols for Advanced Networks (TISPAN). This methodology supports a non-security expert to hierarchically elicit security requirements from overall functional system descriptions and functional requirements. The methodology was developed during a series of security requirements projects, and applied within the IPTV standardization project discussed in this paper (cf. Section 4.1). The Common Criteria document is large in size (three parts of together more than 600 pages and a significant amount of additional material, such as existing Protection Profiles [18]), tailored for security evaluation and certification rather than security requirements elicitation support, and few people really understand all aspects of the Common Criteria. Making use of the Common Criteria requires an understanding of the security assurance paradigm and a deep general security domain knowledge. To extract the security requirements, requirements engineering expertise is also necessary. In the IPTV project, the methodology used at ETSI helped structuring the security requirements interpretation and negotiation process, helped in the formulation of the security requirements, but provided no support for requirement tracing to design. Also, a need for better reuse of security experience was evident, as the process turned out to heavily rely on the presence of experts with experience in both security and requirements engineering to run smoothly. However, the way the Common Criteria were used proved

to be effective and has been adopted in the security requirements elicitation and tracing methodology described in this paper.

1.2 Reusing security expertise: SecReq

This paper describes SecReq, a security requirements elicitation and tracing methodology that is built upon the methodology applied at ETSI. SecReq enhances the ETSI methodology with security requirements elicitation and writing support, as well as requirements analysis and tracing capabilities. Those added elements are supported by a systematic use of different sources of security expertise and experience, and by integrating three existing techniques, namely the Common Criteria (from the ETSI methodology), the heuristic requirements editor HeRA, and the model-based security engineering approach UMLsec.

In order to raise awareness early in the requirements process, SecReq provides support for identifying potential security relevant aspects of service descriptions and functional requirements. This has been identified as being a challenging and time-consuming task in the ETSI methodology. For structuring and refining security issues systematically, SecReq contains a more explicit description of the process implied in the Common Criteria.

In this paper we describe the four-fold contributions of SecReq to security requirements elicitation and tracing:

(i) explicit description and demonstration of the underlying security requirements elicitation process, guidelines and refinement steps of the Common Criteria,
(ii) direct support for writing better security requirements by supporting the information flow between the stakeholders who are involved,
(iii) support for identifying potential security aspects in service descriptions or requirements, and
(iv) security requirements tracing to secure design.

SecReq integrates three techniques that address the four above-mentioned aspects of security requirements elicitation and tracing:

(1) The Common Criteria standard, with its underlying security requirements elicitation process as an extension of the ETSI methodology, to guide the elicitation process from identifying overall security objectives to specific security requirements. This covers (i).
(2) The HeRA (Heuristic Requirements Assistant) tool applies security-relevant heuristics to requirements and service descriptions in order to identify potential security issues. HeRA raises awareness and provides feedback while people write requirements. This covers (ii) and (iii).
(3) The UML extension UMLsec to trace security requirements to secure design and to analyse and verify that the design solution complies with the security requirements. This covers (iv).

1.3 Overview of the three techniques integrated in SecReq

The Common Criteria based security requirements elicitation part of SecReq (technique 1 of SecReq) takes high-level security needs (objectives) through several refinement steps to produce specific security requirements, which are at a refinement level suitable to formulate design solutions. A demonstration of this is given in Section 5, where SecReq is explained using the IPTV security requirements elicitation project mentioned earlier (cf. Section 4.1).

The HeRA tool [50] (technique 2 in SecReq) supports technical experts, as well as security experts, in identifying potential security issues. HeRA is integrated with the Common Criteria based requirements method (technique 1) and together these two techniques make up the elicitation phase of SecReq. HeRA contains a requirements editor that allows technicians to enter system functional information such as service requirements. The input to this editor is checked against security-related heuristics. In particular, these heuristics search for keywords and patterns that may indicate security-relatedness. This search for security keywords are in SecReq used, among other things, to help a developer in selecting appropriate parts of the Common Criteria security requirements knowledge, and thus, HeRA works closely together with the Common Criteria based method.

The UML security extension UMLsec (technique 3 in SecReq) contains security principles expressed as UML stereotypes. Within the SecReq integrated approach, these stereotypes are incorporated into security-related heuristics within the HeRA tool. When a stereotype occurs in a text during elicitation, HeRA issues a warning or advice. The Common Criteria are another source of heuristics. Furthermore, UMLsec, together with the Common Criteria refinement process, is also used for tracing security requirements from elicitation into UML design models. Finally, the UMLsec analysis module is used to analyse and verify that the requirements are met by the resulting UML design models.

Integrating the Common Criteria, the HeRA tool, and UMLsec is not straight-forward and posed numerous challenges. For example: to apply UMLsec in the industrial application at hand, we had to extend the set of UMLsec stereotypes with new ones to deal with the specific security requirements in the telecommunication application domain, as explained in Section 5.3. We envision further extensions to the UMLsec stereotypes. In order for the HeRA tool to support the elicitation process, we had to derive the security-related heuristics based on the UMLsec stereotypes we wanted to identify. In addition, we had to add facilities to HeRA that allowed the refinement of security requirements once they were detected. In order to link UMLsec models to the input

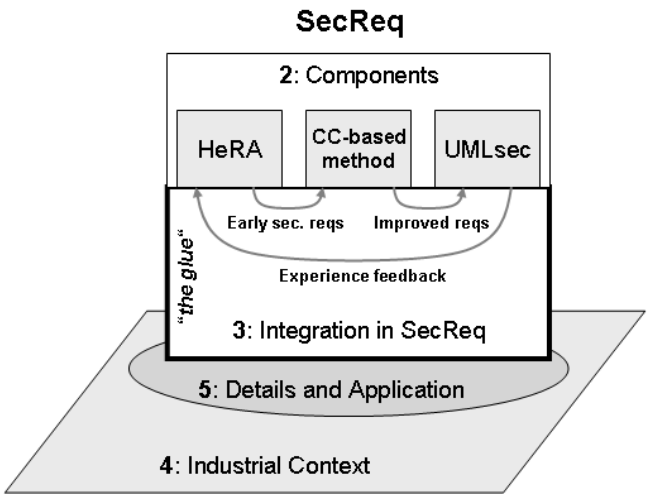

Fig. 1 Overview of sections: components, application and integration of techniques in SecReq

from the Common Criteria on the one hand and to the tool-based process using the HeRA tool on the other, we had to develop an approach which allows us to incrementally develop the UMLsec models in parallel with the incremental development of an associated goal-tree, as explained in Section 2.3. Also, we developed an approach which allows the user to trace the security requirements through the SecReq process using the UMLsec models, as visualized in figure 9.

Nevertheless, SecReq does contribute to making developers not familiar with security able to elicit security requirements. Furthermore, it provides an effective way to reuse security experience across projects, and for accessing tacit security knowledge. This is particularly important in cases where only limited security expertise is available. For these cases, SecReq adds synergy that enables stakeholders to better master the challenging task of developing complex security-critical systems.

1.4 Overview of this paper

The paper is structured as outlined in Figure 1, which gives an overview of how the SecReq approach is presented in the different sections in this paper (referred to in the figure by their section numbers). In Section 2, we present the three techniques as components of SecReq. In Section 3 we describe how SecReq exceeds its components by integrating them in a non-trivial way: they are "glued together" by flows of security experience that might otherwise be neglected. In Section 4, we provide the industrial context under which the main parts of SecReq were developed, as well as present the Internet Protocol Television (IPTV) project (the example project discussed in this paper). Section 5 explains the details of SecReq and how we applied SecReq in the IPTV project introduced in the section before.

In Section 6, we provide a critical discussion of the various parts of SecReq, and outline some of our experiences from the practical application of SecReq. In Section 7, we discuss different approaches relevant for SecReq. We conclude the paper in Section 8 and give an outlook on future work.

2 Heuristics, Common Criteria, and UMLsec

This section provides background information related to the three techniques of the SecReq approach.

2.1 The role of the Common Criteria in security requirements elicitation

The Common Criteria project was established in 1995 as a response to the national and regional security evaluation standardization initiatives. These include:

- Trusted Computer System Evaluation Criteria (TCSEC), also known as the orange book [25].
- Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) [35].
- Information Technology Security Evaluation Criteria (ITSEC) [26].

The project was led by the International Standardization Organization (ISO) and the aim was to harmonize these diverse standardization initiatives (TCSEC, CTCPEC, ITSEC, etc). In 1999, after four years of rigorous evaluation of the existing standards and current industrial practise, the project published ISO 15408: Common Criteria for Information Technology Security Evaluation (Common Criteria) version 2.1 [16]. The standard has since gone through several revisions until version 3.2 revision 2 [17], which was released in September 2007. This is the version that SecReq is built upon.

The Common Criteria consist of three parts:

Part 1: Introduction and General Model [18],
Part 2: Security Functional Components [19], and
Part 3: Security Assurance Components [20] and an evaluation methodology (CEM) [13].

The CEM methodology is different from parts 1-3 as it is a guidance tool aimed only at the Common Criteria evaluator, who must be formally certified to carry out evaluations of IT products. Parts 1-3 provide general guidelines to the developers and the customers of IT products, as well as to the evaluators. Note that the Common Criteria operate under the security assurance paradigm. Security assurance refers to the level of confidence in that the system delivers the specified security functionality, rather than the level of security functionality (often simply referred to as security level).

The main contribution of the Common Criteria is a framework that permits comparability between results of independent security evaluations. This is done by providing a common set of requirements for the security functionality of IT products, and for the assurance measures that are applied to these products during an evaluation. The evaluation process is used to establish confidence in the fulfilment of particular security functionalities. The process also helps the customers of IT products to determine whether the identified functionalities meet their security needs. The IT product or its part being evaluated is referred to as Target of Evaluation (ToE). A ToE can include any combination of hardware, firmware or software, the development of these and the operational environment that they are intended to work in or that they are being evaluated for. Hence, a ToE is a specific configuration of an IT product. Details can be obtained from the Common Criteria portal (http://www.commoncriteria.org).

The three groups with general interest in Common Criteria evaluations are consumers, developers and evaluators. Consumers are aided by Common Criteria evaluations and Common Criteria information in procuring IT products. They may have alternative IT products to choose from and Common Criteria information can thus be used to compare different IT products. Consumers or developers may also use Protection Profiles (PP) to specify security requirements. A PP contains security requirements targeted at a specific type or group of IT products. Developers use the Common Criteria to help them identify security requirements and to market their products. An evaluator uses the Common Criteria to aid in formulating judgements about the conformance of ToEs to the security requirements in the evaluation of a particular product.

The Common Criteria standard recognizes two types of evaluations:

(1) ST/ToE evaluation and
(2) PP evaluation

The standard provides support for developing all three constructs (ST (Security Target), ToE, and PP). Here ST denotes the Security Target, which is the construct used to formulate the functional security requirements for a particular ToE and can be considered as the security requirements specification for a ToE. A PP is the same for a group or a specific type of ToE, and lists implementation-independent functional security requirements for a particular IT product type, such as smart cards. A ToE can also be evaluated against a PP, which is made implementation-specific by constructing an ST. Thus, a PP states the security objectives and the general functional security requirements for an IT product type that can be made ToE specific in an ST. Figure 2 shows how PP, ST and ToE relate to the development phases.

Common Criteria Part 2 [19] specifies the security principles along with their internal dependencies and supports

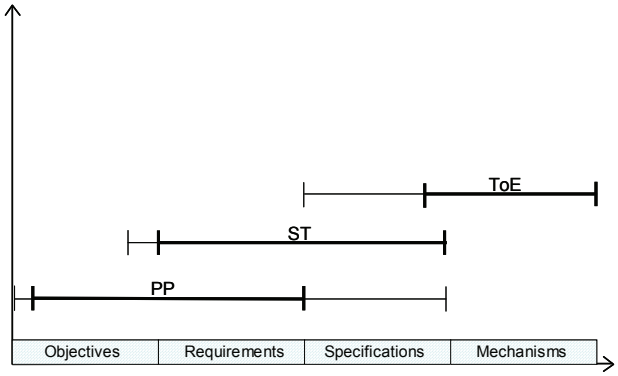

Fig. 2 PP, ST, and ToE in relation to phases of a development process

with a refinement process to refine high-level security statements to specific security requirements. Part 2 consists of 11 security functional *Classes*, each refined into sets of *Families*, *Components* and *Elements*. In Section 5, we present the refinement process for the Identification and Authentication (FIA) class.

Classes are the most general grouping of security requirements, which means that all members of a class share a common general focus. An example of a class is the FIA class, which is focused at identification of users, authentication of users and binding of users as subjects to objects. Classes are refined into one or more families. A family is a grouping that shares a more specific focus, but that differs in emphasis or rigor. An example of a family is the User Authentication (FIA_UAU) family, which is part of the FIA class. This family concentrates on the authentication of users. Each family within a class contains a set of components to represent increasing strength or capability. The components may be partially ordered but in some cases, there is only one component in a family and thus ordering is not applicable. An example of a component is FIA_UAU.3 (Unforgeable authentication), which is used to specify unforgeable authentication of subjects to objects. The components are constructed from individual elements. An element is the lowest abstraction level in CC part 2 that is relevant for security requirement elicitation and is formulated in such a way that the fulfilment of an element is verifiable in practise. An example of an element is FIA_UAU.3.2, which specifies that use of copied authentication data shall be prevented.

The SecReq approach performs the security requirement refinement activities along the abstraction levels of classes, families, components and elements, that, together with the PP, ST and ToE activities and constructs, define the underlying security requirements elicitation process of the Common Criteria. The first rule for a refinement is that a security objective or requirement at some abstraction level has to meet both the refined and unrefined construct. If a refinement does not meet this rule, the resulting refined construct is considered to

be an extended construct and shall be treated as such. For an example of a valid refinement consider FIA_UAU.2.1: "The ToE Security Functionality (TSF) shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user." This can be refined to: "The TSF shall require each user to be successfully authenticated by *username/password* before allowing any other TSF-mediated actions on behalf of that user." TSF refers to the security functionality of a system and is comprised of all hardware, software, and firmware of the ToE that must be relied upon for the correct enforcement of the security requirements. The second rule of refinement is that a refined requirement shall be related to the original requirement. For example, refining an audit requirement with an extra element for masquerade prevention is not allowed.

This hierarchy of security requirements refinement levels and the associated refinement rules provide the underlying elicitation process of SecReq.

2.2 Heuristic for security requirements elicitation

When requirements analysis start, stakeholder wishes should be captured and documented as requirements. These initial requirements tend to be vague and weakly documented, but are nevertheless important. In addition, stakeholders may have conflicting or unclear goals and perspectives. Therefore, it is crucial that the requirement engineers have good communication skills and the ability to understand the ongoing processes between stakeholders (which might be explicit or implicit), as well as to understand the stakeholders' views and early requirements. However, as the domain knowledge and familiarity with the involved stakeholders and customers may be rather limited at the beginning, this is a challenging task. Misunderstandings and errors in meeting notes, use cases, or other requirements-related documents can easily lead to design flaws or even wrong design that later cause severe and costly problems. Errors can be simple typos, usage of synonyms or homonyms, problematic sentence structures, and inconsistent process descriptions, that all somehow makes the requirements imprecise. Furthermore, in e.g. neuro-linguistic programming (NLP) grammatical patterns are identified as potential reasons for misunderstanding [8]. Correcting these errors may be costly, but when it comes to security requirements, errors can even open up for severe security attacks. Thus, getting the security requirements right from the early stages on, will save on cost, but will most probably also reduce the chance of severe security attacks that may affect company reputation, lead to loss of customer data, loss of customer privacy, etc.

Security requirements appear throughout the elicitation process, the stakeholders are simply not aware of them. E.g., when stakeholders discuss general requirements in meetings, they are often not aware that they also discuss security-related topics. One way to deal with this and to make the stakeholders aware of the security-relatedness, is to work with a requirements support tool that prompts whenever indicator terms or expressions that relate to security occur. Security-relatedness can e.g. be recognized by certain words or patterns of words, such as the combination of "Internet" and "payment", which then should be investigated in more detail. Such a prompt will elicit more considerations and possibly even make hidden security requirements explicit. The earlier a security requirement can be identified, the easier it is to trace and to cover it throughout implementation. Therefore, rapid feedback mechanisms are needed.

HeRA (Heuristic Requirements Assistant) is a tool that incorporates the concept of heuristic requirements elicitation [50]. In SecReq, HeRA uses heuristic rules to represent security related experience. Users of HeRA receive warnings and hints that are triggered when they type a certain term or pattern. Rules are created from the experience of security experts, and from observations in earlier projects [51]. Findings can be codified as rules. E.g., considering the refinement of requirements, a security expert could specify a rule that whenever a requirement contains the keyword "authenticate" a refined requirement with the specific authentication mechanism has to exist. HeRA supports writing and editing rules, and integrates them during run-time. Rules are organized in layered sets of rules: There is a layer of basic general rules, with more security-specific layers on top.

HeRA is one instance of a family of tools that are based on Fischer's architecture for domain oriented design environment (DODE) [30]. The central part of this architecture is a construction component; in the case of HeRA, requirements are "constructed" using a general-purpose requirements editor, or a use case editor. Both editors check heuristic rules. In Figure 10, HeRA's requirements editor is shown. Each requirement has an ID and a textual description by default, but additional attributes can be added (e.g., relevant UMLsec stereotypes). The second DODE component of HeRA that we use in SecReq is the argumentation component: Security experts can use HeRA mechanisms to codify their findings from earlier projects. As a rule set in HeRA, those heuristics will support future users who are less proficient in security. HeRA applies these rules to the input in the requirements editor and then analyses it. Based on this analysis, HeRA's argumentation component gives context-sensitive feedback to the input.

In HeRA, heuristic rules are defined in JavaScript and can access the data model of the requirements editor. There are wizards that allow users to generate JavaScript code for a rule, along with a description and parameters (e.g. certain keywords). Rules can be changed during runtime and the results become visible immediately. Figure 3 shows how a rule can be defined in HeRA, and in Section 5 we will see work-

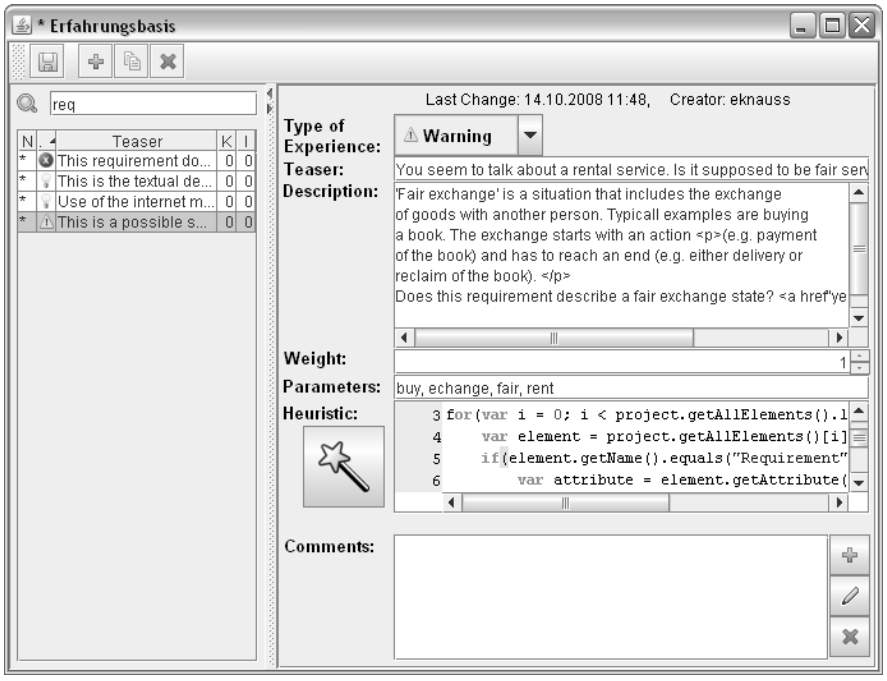

Fig. 3 Definition of a heuristic rule in HeRA

ing examples of such rules for detecting security-relatedness; here referred to as security heuristics.

The underlying idea of requirement tools in the DODE family is to create awareness and improve feedback about domain specific knowledge (coded into the heuristic rules) during documentation of requirements. Fischer investigated the effects of a DODE in different domains and reports its effectivity [30]: The argumentation component analyses the user's actions and issues computer-based *critiques*. During an activity like typing requirements this triggers breakdowns [66]. In this way, authors are interrupted and made aware of a problem they may have overlooked. They can decide to fix it, finish their work and return to the warning later, or give feedback to the warning. Feedback results in an improved rule set for the next project. Fischer calls this the SER-cycle [31]:

Seeding A knowledge based system needs to be initialised with knowledge. Otherwise it would not be helpful; hence it would not be used.

Evolutionary Growth While the system is being used, additional experiences are added in response to warnings. In HeRA, comments or explanations may be given. Even ignoring a warning provides feedback. Heuristic rules can be changed based on feedback.

Reseeding At some point, the experience base needs to be cleaned up. In HeRA, the security expert would usually remove heuristics that were often ignored and change others that were commented on.

HeRA works with different sets of heuristic rules. The basic set covers general warnings like typos. Additionally, there are requirements specific rules, like the detection of ambiguities. In [49] we investigated how this rule sets can be used to increase the quality of use case descriptions (based on the template and guidelines provided by Cockburn [15]). We found that HeRA helped to achieve better quality. In fact, the issues brought up by HeRA were not found during quality gate reviews in similar projects. Some of the findings in the reviews never caused a heuristic in HeRA to fire. We concluded that HeRA's constructive feedback complements analytical quality assurance with constructive assistance [49], but cannot replace it.

Therefore, in SecReq we define an additional role to make the elicitation of security requirements more effective. This role is the security instructor, which ideally should be someone with experience and knowledge within both security and requirements engineering. The reason for this is that an experienced moderator can guide the stakeholders in how to use HeRA, as well as capture new security concerns not currently covered by HeRA. Furthermore, often stakeholders tend not to see the link to security, even when they are prompted by HeRA, and a moderator can help mitigating this risk. However, there is usually a severe shortage of experienced moderators, so SecReq does not depend on one being present. Figure 4 shows the information flow in a moderated elicitation situation. An information flow model according to the FLOW notation [65] depicts the flow of requirements within a project. Flow of requirements is denoted by black arrows. Grey errors indicate the flow of experiences. Unlike requirements, experiences are not specific to a single project.

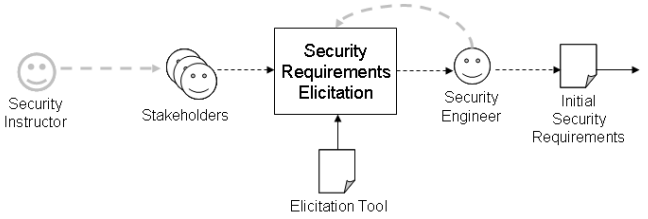

Fig. 4 Information flow in an elicitation situation

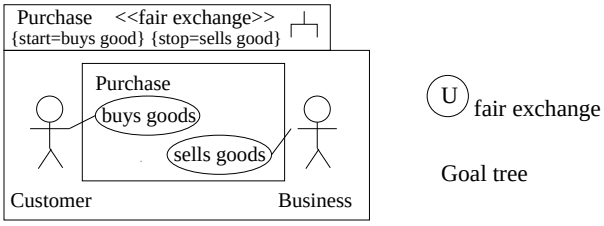

Fig. 5 Use case diagram with goal tree

FLOW distinguishes between documented information (requirements, experience) and oral or informal communication. Solid lines and document symbols indicate documented information, while dashed lines stand for informal or oral representation of information, which is called "fluid" in FLOW. Information and experience people have in mind is fluid. It is denoted by a face symbol. In Figure 4, an experienced requirements engineer uses her personal, un-documented experience (grey dashed arrow) to support stakeholders in their elicitation effort. Ideally, the stakeholders should receive a briefing by an experienced security instructor before the elicitation starts. The instructor will provide insights (i.e., experiences - grey arrow) and raise awareness for security issues. The requirements engineer receives information on security requirements (black arrow) from the elicitation step by listening to discussions, carrying out interviews, etc. HeRA supports these needs in an elicitation situation through its ability to capture the security experience of e.g. a security expert explicitly by adding new rules to its heuristics, and by allowing the security instructor (expert) to merely guide the elicitation process. I.e., it is the stakeholders that perform the actual elicitation, supported by HeRA, while the security instructor observes, guides and inserts new heuristic rules if necessary. E.g., if the security instructor discovers security concerns in e.g. functional documents not currently covered by the heuristics of HeRA, he can update the heuristics as part of the elicitation process. This way, HeRA learns from new experience.

2.3 UMLsec for security requirements analysis

The Unified Modeling Language (UML) consists of different types of diagrams for describing different views on a system. It offers rich extension mechanisms in the form of labels that can be used to provide additional data. These can be either stereotypes (written in guillemets or double angle brackets «stereotype») or tag-value pairs (written in curly brackets such as {tag=value}). Using these extension mechanisms, one can construct an extension of the UML notation for a given application domain.

UMLsec is such a UML profile for security-critical systems that can be used to express and analyse security requirements [45]. The purpose of security analysis is to achieve a satisfactory level of confidence that the relevant security requirements are properly fulfilled. For the analysis, the design solution is expressed using UML diagrams and the security requirements are expressed as UMLsec stereotypes and tags and integrated with the diagrams. This way, one can trace the security requirements into the solution design. Moreover, one can also backtrack from the solution design to the security requirements, as each «stereotype» has a precisely defined semantics that links the design to the requirement.

By developing a security goal tree alongside with the development of the solution design specification, the security objectives are refined in parallel by giving more system details in subsequent UMLsec diagrams. Goals are refined in parallel in an iterative fashion by refining the UMLsec diagrams with additional system details as they become available as part of a given UML based development process. The resulting goal tree and subsequent UMLsec diagrams then link the solution design to the security objectives.

The UMLsec approach is supported by a number of automated analysis tools, which are based on a formal semantics of the relevant fragment of UML [42,43]. It has been applied to a number of application domains such as service-oriented systems [56]. In the following, we give a short background overview of the part of UMLsec relevant for SecReq using examples.

Requirements Capture We employ use case diagrams to capture security requirements. To start with our example, Figure 5 gives the use case diagram describing the situation to be achieved together with the (yet trivial) goal tree: a customer buys goods from a business in a way that should provide a fair exchange of the goods against the payment. The semantics of the stereotype «fair exchange» is, intuitively, that the actions "buys goods" and "sells goods" should be linked in the sense that if one of the two is executed then eventually the other one will be (where these actions are specified on the next more detailed level of specification). The "U" in the goal tree stands for "undetermined" – it is not yet known whether the goal will be *satisfied* ("S") or *denied* ("D") [14, 44].

In general, the idea of the UMLsec extension is to use UML diagrams that would also be used for a system that is not security relevant (such as use case diagrams), and add the security relevant information as stereotypes or tagged

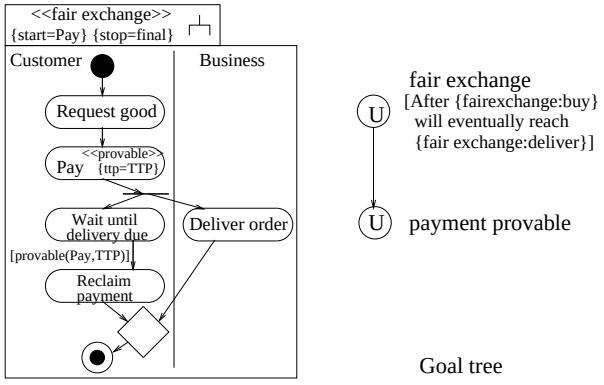

Fig. 6 Activity diagram and goal tree

values. Note that there are also diagrammatic notations similar to use case diagrams which are specifically used to express security-relevant information, such as Misuse case diagrams [68], Abuse case diagrams [55], or Secure Tropos diagrams [60,54]. These specialized notations can often be used successfully together with the UMLsec extension of the general-purpose notation UML. E.g. for Secure Tropos this is explained in [61]. However, this is not in scope of the current paper.

Analysis We use activity diagrams to explain use cases in more detail. Following our example, Figure 6 explains the use case of Figure 5 and the associated goal tree in more detail by giving an activity diagram and a refined goal tree. The activity diagram is separated in two *swim-lanes* describing activities of different parts of a system (here Customer and Business). Two tag-value pairs {start=Pay} and {stop= final} are used to mark certain actions. Now any such diagram fulfils the security requirement fair exchange given in the diagram in Fig. 5 if the condition holds that if one of the start actions is executed, then eventually one of the stop actions will be.

Design: object and sequence diagrams An object is an entity with well-defined boundary and identity that encapsulate state and behaviour. State is represented by attributes (for instance properties of a user of a system) and behaviour is represented by operations, methods, and states. Object diagrams provide instance level information and represent different objects and their interfaces (e.g. dependencies) to analyse the system behaviour. Sequence diagrams show the interaction of objects or components. Both object and sequence diagram are part of the security analysis and the associated goal tree is used to trace security objective and sub-security-objective along with design diagrams. Additionally, relevant stereotypes are incorporated within these diagrams so that the design contains security requirements related information.

Implementation: deployment diagrams Deployment diagrams describe the underlying physical layer; we use them to ensure that security requirements on communication are met by the physical layer. Deployment diagrams consist of nodes as modelling element represents the system components to the physical structure of the system and links among the nodes. User node specifies the external actor's components, whereas system node specifies the internal system actor's components. These components from different nodes are connected by link stereotype.

3 SecReq overview: integrating security techniques

The goal of SecReq is to assist in all steps in the security requirements elicitation and to provide mechanisms to trace security requirements from high-level security objectives to the design of a secure system. SecReq combines three distinctive techniques that have been integrated to meet this goal. These techniques are:

- Use of the Common Criteria standard and its underlying security requirements elicitation and refinement process;
- the HeRA tool with its security-related heuristic rules, and
- UMLsec stereotypes, secure design principles and the UMLsec security analysis tool-set.

These three techniques are integrated in a way supporting the two purposes of SecReq:

(i) security requirements elicitation and
(ii) security requirements tracing and mapping to design, as shown in Figure 7.

The security requirements elicitation is supported by the Common Criteria requirements refinement process and by the HeRA tool. Some heuristic rules in HeRA are derived from the security principles in the Common Criteria, others come from UMLsec stereotypes, and others are stimulated directly by observations of security experts. Tracing security requirements is enabled by UMLsec and goal trees.

The crucial resource of security expertise and experience is used to tie all parts of SecReq together. As the grey lines in Fig. 7 show, the HeRA tool supports security requirement elicitation and uses the expertise of security engineers and the experience expressed in stereotypes of UMLsec. The requirements elicitation approach reuses experience from the elicitation (via the security expert) and is guided by the documented experience in the Common Criteria etc. UMLsec can build on a significantly improved input of security requirements as a basis for its analysis. When new security issues occur, they may be encoded into UMLsec stereotypes and models. The latter are fed back to HeRA where they contribute to detect the new kind of issues in stakeholder interviews.

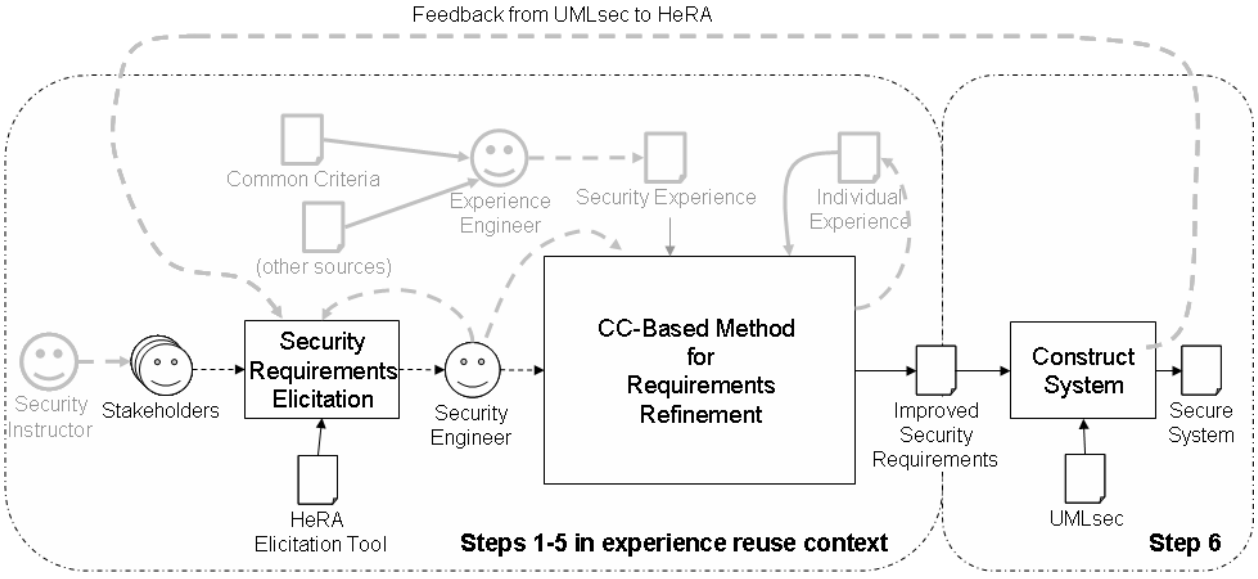

Fig. 7 The main elements of the SecReq method as an information flow model. Black parts represent requirements and their flow, while grey parts stand for experience. Document symbols and solid lines indicate documented information. Dashed lines and human face symbols represent information that is kept in mind or transferred orally or informally (which is called "fluid" in FLOW terminology [65]). Note the fluid feedback from Step 6 (UMLsec) to the elicitation activity. Problems identified in UMLsec design models are used to improve heuristics in security requirement analysis. Other experience reuse mechanisms are depicted above the CC-based method box. Elicitation using HeRA provides input to Steps 1-5: security issues discovered by applying the heuristics.

The integration of the HeRA tool, the Common Criteria based requirements elicitation method, and UMLsec goes beyond putting one technique next to the other. Instead, the flow of experience and expertise glues all parts together and facilitates an evolutionary learning cycle. The resulting specifications run through an integrated set of filters. UMLsec stereotypes convey parts of that experience on security concerns. They are both used to assist in eliciting security requirements and in tracing these from high-level security statements (security objectives) to secure design. Firstly, UMLsec stereotypes assist in the elicitation activities involving the HeRA tool: UMLsec stereotypes are incorporated into the heuristic rules of HeRA. Secondly, UMLsec stereotypes are identified from the security objectives, sub-security-objectives, security requirements and specific security requirements that were elicited. UMLsec stereotypes help to make the security needs (in the objectives and requirements) explicit, verifiable, and traceable.

3.1 Security requirements elicitation and analysis (SecReq)

SecReq is a security requirements elicitation and tracing method built on the Common Criteria, the HeRA tool and UMLsec. The elicitation part consists of five steps that take a developer through a series of refinement steps starting from system objectives and functional requirements and ending with specific security requirements at an early stage.

These requirements should be verifiable and measurable expressions about the mandatory, desired and optional security features of the system. Furthermore, the requirements expressions should be on the refinement level of Common Criteria elements and according to the SMART principles [53]. There are several definitions of SMART available. In SecReq, we use the adaptation of the SMART terminology of ETSI TC TISPAN. This means that each elicited security requirement must be *Specific*, *Measurable*, *Achievable*, *Realizable*, and *Traceable* (SMART). *Specific* means that the requirement must be described in such a way that it is well defined and clear to anyone who has a basic knowledge of the project and security requirements. *Measurable* means that the requirements must be described in such a way that one can verify whether it has been met. *Achievable* means that the requirements must be possible to meet and include a description that can be used to determine whether it has been met. *Realizable* means that the requirements should be possible to meet given the system and physical constraints, and given the project resource and schedule constraints. *Traceable* means that the requirements should be expressed in such a way that it is possible to trace them into the solution design, and eventually into the implementation. The security functional elements of the Common Criteria are the proper abstraction level to analyse for fulfilment of SMART. This refinement level further makes it possible to analyse for fulfilment of the requirements in the solution design using UMLsec, which is the requirement tracing and analysis activity of SecReq as shown in Figure 7.

In the process of going from functional descriptions to specific security requirements, we make use of the abstraction levels of the various constructs in the Common Criteria: *Class*, *Family*, *Component*, and *Elements*, the refinement specifications given in Common Criteria part 2, and the HeRA tool with its security-related heuristics.

The SecReq method consist of the following six steps:

Step 1: Specify security objectives from system objectives and functional requirements
This step involves deriving security objectives from system objectives and functional requirements. Other sources for security objectives are the system architecture, concept descriptions, or any other relevant system information available, such as existing standards. Common Criteria security functional classes offer guidance throughout this step. HeRA supports this refinement by identifying security relevant requirements and asking the right questions for the refinement.

Step 2: Associate a security functional class to each security objective
During this step, for each security objective, the relevant security functional requirement class is selected from the Common Criteria. Again, HeRA could guess an appropriate security functional class based on heuristics.

Step 3: Refine security objectives to sub-security-objectives
The task of this step is to refine each security objective into one or more sub-security-objectives. Each sub-objective should comply with one or more of the Common Criteria families contained within the relevant functional class from the previous step. The HeRA tool is also used to aid this step by identifying additional candidate sub-security-objectives, and by guiding the refinement process.

Step 4: Refine sub-security-objectives to security requirements
This step involves refining each sub-security-objective into one or more security requirements supported by the contained components of the relevant Common Criteria family (i.e., the family used for the sub-security-objective). The HeRA tool is also used to aid this step, similar to that of Step 3.

Step 5: Refine security requirements to specific security requirements
This is the last step of the requirement elicitation part of SecReq and refines each security requirement into one or more specific security requirements supported by the elements contained in the relevant Common Criteria components (i.e., the components used for the security requirements). Similar to Step 3 and 4, the HeRA tool supports this step by identifying potential additional security requirements. Again, it guides the refinement process with a dialogue based assistant.

Step 6: Analyse the security requirements using UMLsec
This is the analysis part of SecReq. The goal of this step is to check whether the secure design contains the mandatory, desired and optional specific security requirements that were elicited. This includes checking the UML models specifying the design for fulfilment of the requirements, and it enables tracing from the solution design, via specific security requirements, back to the security objectives. This analysis requires that all security requirements elicited in Steps 1–5 be specified on the abstraction level of Common Criteria security functional elements (specific security requirements). The security analysis with UMLsec models is supported by the development of a security goal tree in parallel to the development of the UMLsec model.

Security requirements elicitation in SecReq Security requirements elicitation in SecReq thus consists of a tight integration of a Common Criteria based requirements refinement process and the HeRA tool. HeRA contains, among others, security specific heuristic rules derived from the UMLsec stereotypes and the security principles of the Common Criteria. In particular, elicitation in SecReq follows the first five steps of the process described above, which produces specific security requirements. These requirements are refined from the high-level system descriptions and security needs, which are called security objectives. The process is iterative and whenever new system information becomes available or new high-level security issues appear, the process iterates back to the relevant refinement step.

The five steps are refinement steps that follow the hierarchical refinement structure of Common Criteria part 2, the security functional components. As discussed earlier, the Common Criteria standard structures security needs or principles into five refinement levels, where the most abstract level is called a security functional class and the most refined level is called security functional element. Note that elements both represent a security principle abstraction level and contain specification of the necessary security actions to be undertaken by the involved parties. It is this refinement process that drives the elicitation process, as there is a direct link between the Common Criteria classes and elements. One class represents one security principle that is refined in such a way that following the class structure intuitively results in specific security requirements that can be mapped directly to the design. The Common Criteria standard also contains specification of dependencies between the classes (security principles), which also guide the elicitation process.

Fig. 8 illustrates the role of the Common Criteria and the HeRA tool in the five step elicitation process of SecReq. Drawn in grey, the Common Criteria are considered solid, documented experience. The Common Criteria control all five steps, which are indicated by the flows coming in from

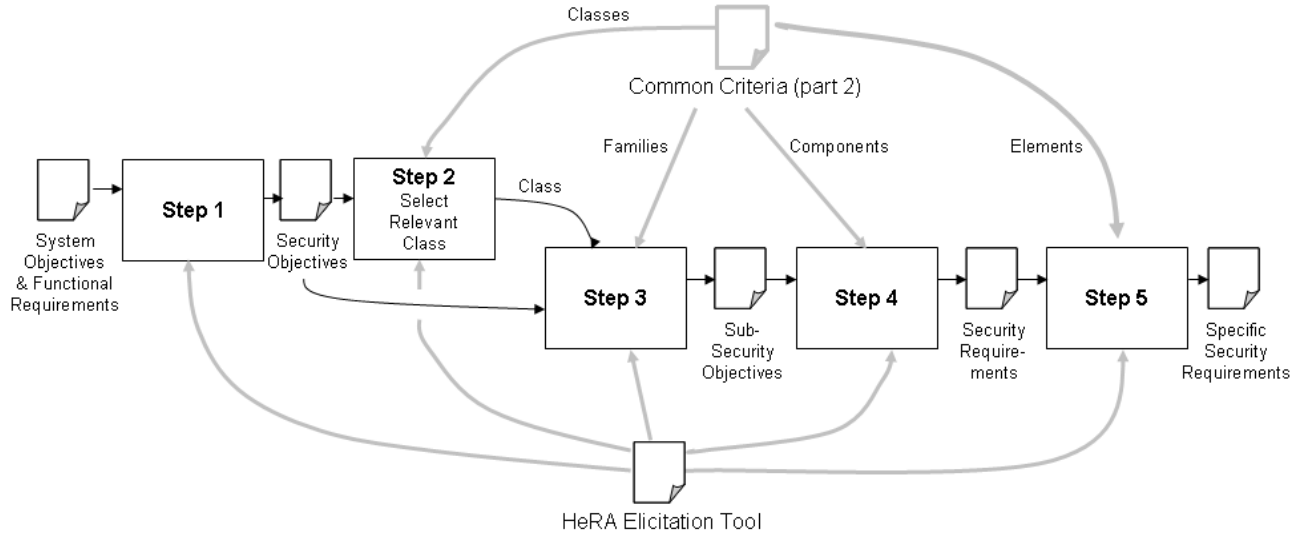

Fig. 8 The role of the Common Criteria and HeRA in the requirements elicitation process

the top. HeRA is supporting all steps at the same time, which is indicated by arrows from HeRA to the bottom of the steps. Fig. 8 focuses on these experiences from Common Criteria and from HeRA. Further input, such as incoming fluid requirements, are omitted in this figure. Note that the iterative nature of the elicitation process is not shown here.

The HeRA tool is the elicitation support tool of SecReq. The tool can support all five elicitation steps, but in particular the difficult Step 1. Heuristic rules derived from experience notify stakeholders when they describe issues that seem to be security-related. UMLsec stereotypes are one of the sources of experience that are captured in heuristic rules in HeRA. They are most helpful in identifying potential security issues from system objectives, functional requirements and the like. HeRA has two main tasks in the elicitation process:

(i) identify potential security issues in system description of various abstraction levels, such as high-level system objectives descriptions, service functional requirements etc. (by means of a security instructor as shown in Fig. 4), and
(ii) guide the developer in formulating security objectives, sub-objectives, security requirements and specific security requirements, in the five steps respectively.

Task (ii) is crucial to produce formulations that are both specific and verifiable. The task is supported by the security principles and the security functional requirements hierarchy of Common Criteria part 2.

The heuristics within HeRA are triggered at any of the elicitation steps when technical experts use the HeRA editors and run into a suspicious word or pattern. At this point, they are supposed to discuss related security issues and potentially pin down additional security requirements. Note that these additional security requirements must also be on the refinement level of specific security requirements. On this level it is possible to trace them into the security design. In addition, we ensure that all security requirements are refined to a detail level that allows one to analyse their fulfilment in Step 6.

Security requirements tracing to design in SecReq In addition to an effective and tool-supported elicitation process, in Step 6 we move from specific security requirements to secure design. The goal of this step is to achieve a satisfactory level of confidence that the design can properly fulfil the security requirements, and thus meet the security goals. This forward tracing can be done at an early stage of development. If required, tracing can also be done backwards from the design to the requirements, and finally to the security objectives.

Successful security requirements elicitation relies on the assumption that the requirements are later fulfilled in a secure design. The design represents the solution space for the requirements. Achieving a satisfactory solution and checking it for the fulfilment of the requirements involve:

(i) producing a solution representation in the design and
(ii) tracing requirements from the high-level security statements (security objectives), via the elicitation refinements, to secure design.

Without (i), the security requirements merely represent good intension, and without (ii), there is no control regarding whether or how the security requirements are realized. Thus, there cannot be any security assurance.

SecReq uses UMLsec for advancing security requirements to secure design, and for tracing the requirements from security objectives to design. One part of the requirement tracing is to specify the relationships between the security objectives, sub-security-objectives, security requirements and specific security requirements. This is done using a goal tree, where security objectives are the goals, and where these goals

are linked to the specific requirements through the two intermediate refinement levels (sub-security objectives and security requirements). By doing so, each goal tree specifies the dependency among the involved security refinement level constructs. In addition, the goal tree is integrated with the UMLsec diagrams and hence the security objectives can be traced to and enforced by the UML design diagrams.

A UMLsec design specification requires a set of constructs that must be identified to map the elicited requirements to the design. First, the relevant UMLsec stereotypes must be identified. As the UMLsec stereotypes are used as input to the heuristic rules during the elicitation phase, some of these are already identified. Nevertheless, it is necessary to go through the goal trees and map the security constructs to UMLsec stereotypes. When that is done, the involved actors and their interactions for each UMLsec stereotype must be identified. Note, however, that any conflicts between the UMLsec stereotypes identified on any of the layers in the goal tree must be resolved before the actors and their interaction can be specified. The actors and interactions are always specified on the refinement level of the specific security requirements; i.e., the lowest level in the goal tree. If there are more than one specific security requirements associated with a security objective, several sets of actors and interactions are the result of this activity. The final step of moving requirements into design is to choose the relevant diagram type. UMLsec is modelled using both behavioural and structural UML diagrams as discussed in Section 2.3. Depending on the UMLsec stereotype and the actors and interactions, one or several UMLsec diagrams are created, which together represent the realization of the security requirement in the design.

4 Industrial Context and Case Study

In this section we outline the industrial context that the SecReq methodology was developed and applied within. One of the projects to which it has been applied is introduced as an example.

4.1 Industrial Context

Parts of the development of SecReq were carried out within the context of the standardization organization ETSI. ETSI is the main standardization organization for Telecommunication (Telco) in Europe and has worldwide influence. ETSI is based on voluntary contributions from its members, which are typically companies with interest in Telco standards, such as companies that provide equipment, software or services to the Telco domain. ETSI consists of a number of programs, such as the "Telecoms & Internet converged Services & Protocols for Advanced Networks (TISPAN)". Within TISPAN, there are eight working groups with different perspectives and focuses, such as technology, protocols, security, user, etc. TISPAN's main aim is to specify the requirements and architecture for the Next Generation Network (NGN). Due to the heavy workload of some of the tasks of TISPAN, ETSI employs a number of experts in various fields each year, including security experts. These security experts work at ETSI for a limited time and are representatives from the ETSI members. The SecReq approach builds on an approach used by a group of such security experts. The methodology was later extended with the HeRA tool, and with additional features of requirements tracing and fulfilment analysis using the UML extension UMLsec. This work was done in collaboration with security experts employed at ETSI.

SecReq was developed in an iterative manner through actual security requirements elicitation projects in TISPAN. We use one of these projects, namely the IPTV standardization project, as a running example to demonstrate the methodology. The IPTV project was carried out between March and August 2007 and resulted in a set of IPTV specific security requirements that achieved formal acceptance by the TISPAN members. This means that the resulting IPTV security requirements are part of an ETSI standard and will be used by the ETSI members when implementing IPTV solutions for NGN.

4.2 Case study: Internet Protocol Television (IPTV)

IPTV is a system where a digital television service is delivered by using the Internet Protocol (IP) over a network infrastructure, which may include delivery by a broadband connection. A general definition of IPTV is television content that, instead of being delivered through traditional broadcast and cable formats, is received by the viewer through the technologies used for computer networks. IPTV is one of the subsystems of the NGN and provides a variety of services, such as Video on Demand (VoD) for residence users, and may be bundled with Internet services such as Web access and VoIP. More information can be found in [71, 72].

There are several other standardization efforts for NGN in general and IPTV in particular, such as that of ATIS[1],

[1] ATIS is a United States based body committed to rapidly developing and promoting technical and operations standards for the communications and related information technologies industry worldwide using a pragmatic, flexible, and open approach.

International Telecommunication Union (ITU-T)[2], and Open Mobile Alliance (OMA)[3].

Standardization deals with specifying concepts at a general level. The goal is to derive specifications that make it possible for all stakeholders involved to together provide services to the end-users. This involves describing what the underlying network looks like, how the system should use the network, how end-users should relate to the service, etc. Therefore, future providers of the different parts of NGN in relation to IPTV have interest in and participate in the IPTV standardization efforts. This includes companies such as access network providers, service providers, IPTV providers, and content providers. This leads to a development context with multiple stakeholders having different, often conflicting, goals. This situation is common in standards development, as well as industrial development, and puts constraints on the elicitation process.

SecReq deals with this challenge by performing requirements elicitation in a step-wise manner, which makes it possible to continuously communicate concrete results at various levels of abstraction, and hence gain repeatable feedback from the core stakeholders throughout the elicitation process. The process allows its users to iterate back at any point in time and provides results that can be used as a basis for discussion from early on in the elicitation.

5 SecReq and its application at IPTV

This section explains the SecReq process outlined in Section 3 in more detail, and describes how it was applied in the IPTV standardization project introduced in Section 4.

5.1 Overview of the application of SecReq to IPTV

The SecReq approach supports eliciting and tracing the specific security requirements with the help of the Common Criteria and the heuristics in the HeRA tool to secure design. In the IPTV case, we identified specific security requirements to ensure non-forgeable identities so that end-users and named groups of end-users can be successfully identified and authenticated before being allowed to perform any service action. Steps 1 to 5 of SecReq support identifying and refining the security requirements by using the Common Criteria and security-related heuristics. In step 6, a set of UMLsec stereotypes are identified from the security objectives, sub-security-objectives, and refined security requirements that were elicited in order to trace the specific security requirement into secure design diagrams, and by that achieve the security objective.

Figure 9 gives an overall view of how the SecReq approach was applied in the IPTV case study. The left part of the figure shows the activities needed for the security requirement elicitation supported by the Common Criteria and the heuristics, and includes the relevant security objectives, heuristics, and the security requirements that were elicited. The right part of the figure shows the activities followed to create the secure design supported by UMLsec, and includes the relevant stereotypes and UMLsec diagrams. E.g., the security objectives identified, such as access control, which are addressed by the specific security requirement: "Successful authentication and identification before any action is permitted". This again is based on the refinement process provided by the Common Criteria through its classes, families, components and functional elements, and assisted by the security-relevant heuristics of HeRA. As described earlier, these heuristics represent security expertise and knowledge and are used to identify potential security aspects from e.g. functional requirements.

Furthermore, the security objectives and specific security requirements are traced to the UML design diagrams by identifying the relevant UMLsec stereotypes (such as «fair service delivery»,«authentication provable», «non forgeable identity», «secure links», etc), and by developing the associated goal tree to ensure that the UMLsec models keep track of all security requirements that are needed. The stereotype «fair service delivery» represents the elicited security requirement from the IPTV application that end-users and named groups of end-users require to subscript and prove identity and authenticity before requesting any IPTV service action. Also, the service providers are supposed to deliver the service requested by the legitimate user in a manner that is fair for both involved parties. The stereotype «authentication pro vable» specifies that users are required to prove the authenticity whenever requesting any service action. Furthermore, the provider is required to send authentication provable information to the successful authenticated user before starting the delivery of service. This gives the user the possibility to reclaim the service in cases where the service delivery is interrupted or not delivered according to agreement. The stereotype «non forgeable identity» specifies that every subscribed user shall have unique identity to support the IPTV specific security requirement. The stereotype «secure links» enforces security to the communication link established between the user and the provider. Both «authentication provable» and «non forgeable identity» refine the «fair service delivery» stereotype. All identified

[2] ITU is the leading United Nation's agency for information and communication technologies. As the global focal point for governments and the private sector, ITU's role in helping the world communicate spans three core sectors: radio communication, standardisation, and development. ITU also organizes TELECOM events and was the lead organizing agency of the World Summit on the Information Society.

[3] OMA is the leading industry forum for developing market driven, interoperable mobile service enablers. OMA focuses on service enabler architectures and open enabler interfaces that are independent of the underlying wireless networks and platforms.

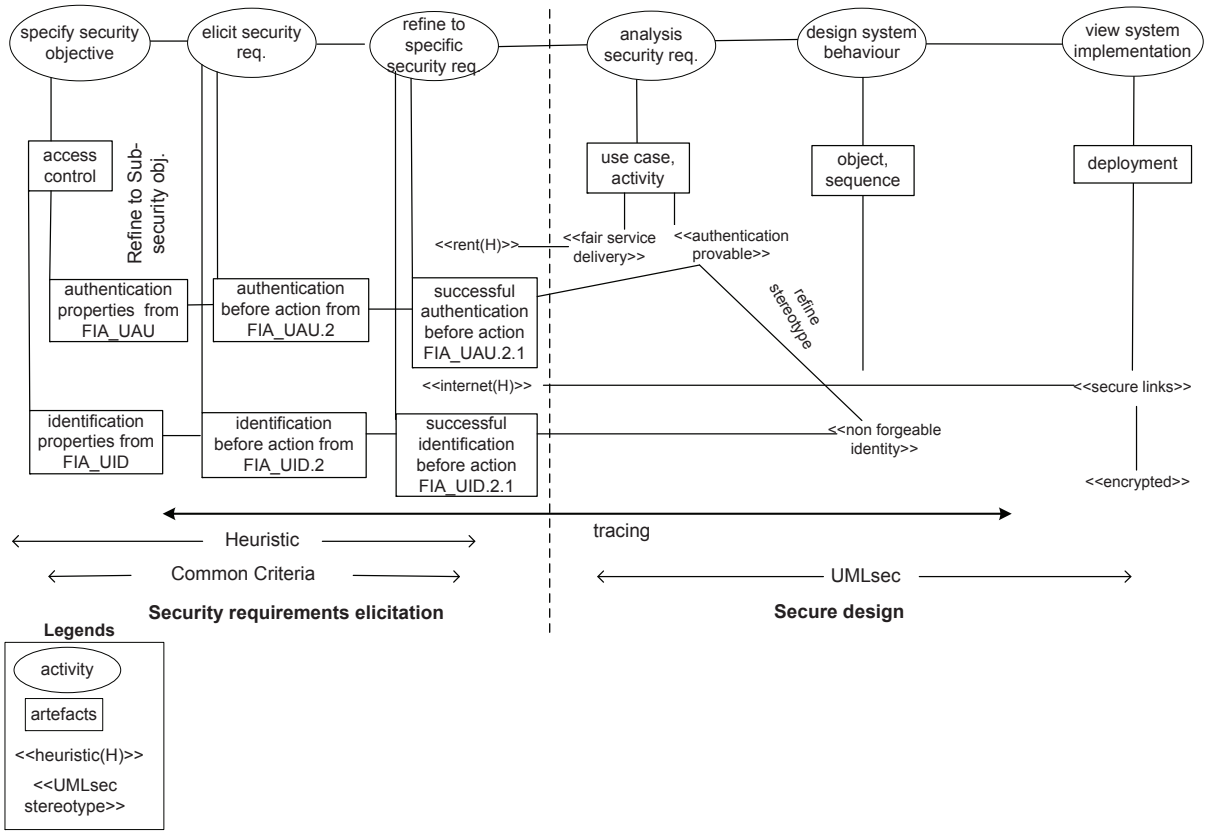

Fig. 9 UML diagrams and UMLsec stereotypes in SecReq

UMLsec stereotypes support the IPTV security, sub-security-objective and relevant specific security requirements. Since these stereotypes have a precise semantic meaning, they can also be used to trace back from the UMLsec analysis phase (Step 6 of the SecReq process; cf. Section 3.1) to the original specific security requirements (Steps 4 and 5), and eventually back to the security objectives (Steps 1 and 2). In the next sections, we will demonstrate parts of SecReq by discussing how it was applied to the IPTV project.

5.2 Steps 1–5: Common Criteria supported Security Requirements Elicitation

In the SecReq method, five steps lead through the process of specifying security requirements. The goal is to define requirements in a way that allows for requirements tracing and fulfilment analysis using UMLsec, and to generate feedback from the analysis. This demands high quality and a high-level of details for the requirements specified. The HeRA tool supports requirements engineers in this task throughout the five steps, by identifying potential security requirements, and by means of providing support to the refinement activities of each step. The kind of support offered differs only in details between each step. Therefore, we start by explaining how the Common Criteria are used in aiding security requirements elicitation, and then we walk through an example demonstrating how HeRA was applied. This separation in the demonstration of Steps 1-5 does not reflect that there is an actual separation in practise, but is merely done for readability purposes.

5.2.1 Step 1: Specify security objectives from system objectives and functional requirements

Step 1 starts with collecting and refining the necessary information about the system at the appropriate abstraction level. As a minimum, the information should contain some high-level description of the main objectives of the system and some functional descriptions, such as some details on the system architecture and/or the assumptions posed upon it, and the functional requirements. These pieces of information are then analysed with the aim of specifying the set of system objectives, and to get an overview of the functional requirements already specified, as these can be used as the starting point when specifying security objectives.

Definition **System Objective** is a high-level goal of the system in meeting the expectations of the end-users of the system.

A system objective is the highest level of abstraction and is met by a set of functional requirements. Existing system

objectives may give rise to additional functional requirements and vice versa. Both system objectives and functional requirements are useful in the process of deriving the specific security requirements, which are needed in order to comply with the SMART principles. The output of Step 1 is a set of security objectives.

Definition **Security Objective** is a high-level goal of the system in providing a secure environment for end-users of the system for a particular system objective.

There might be several security objectives relevant for one system objective. This means that one might have to add more than one security enhancement to each core functionality of the system. The number of security objectives will vary from system to system. In the following, we use the IPTV project as a case study to elaborate on the relationship between system objective and security objective.

Looking at our IPTV case study, one of the main functional services offered by IPTV is Video on Demand (VoD). An example of a narrative system objective for VoD is given below.

IPTV System Objective: IPTV should offer all categories of end-users (note there might be more than one category, e.g., parental control) to select and watch video and clip content over some Internet connection as part of an interactive television system.

The system objective is either extracted from existing functional requirements or from the high-level description of the system given as input to Step 1. If the first case, we use the existing functional requirements to abstract system objectives. As the set of functional requirements might not be complete, additional system objectives may be specified from the available high-level description of the system or system goals. Please note that to abstract means to move one abstraction layer up.

IPTV Functional Requirement VoD.1: IPTV shall offer the ability to specify different service groups for VoD to allow for categorizing of end-users.

From the above functional requirement, we can deduce that IPTV service providers would like to offer different service levels for VoD, i.e., to provide higher service levels at higher charges. We can also deduce that there are categories of end-users and that these need to be distinguishable, i.e. parental control. The latter is particularly relevant for security.

From the above system objective and functional requirement, we can derive the relevant high-level security goals or security objectives. For simplicity reasons, we limit the example to one security objective.

IPTV Security Objective: The IPTV service should restrict the access to services for end-users and named groups of end-users according to a pre-defined access control policy.

5.2.2 Step 2: Associate a security functional class to each security objective

The above security objective addresses the issue of end-users or groups of end-users need to be distinguishable and separable to the system. Therefore, the system needs to be able to uniquely identify end-users, and to ensure that the identified end-users are properly authenticated to the system, as specified in the VoD.1 functional requirement. We use the Common Criteria part 2 security functional classes as support when formulating the above security objective. More specifically, we examine the security objective to identify the relevant security functional class or classes (which means that there might be several classes that are of relevance). This is Step 2 of SecReq. That is, we do a concept match between each of the security functional classes and the security objective. In this context, the security functional class FIA (Authentication and Identification) is highly relevant.

We can derive more security objectives relevant for the functional requirement VoD.1. By examining the functional classes, we see that the classes Security Management (FMT, dealing with management of user roles) and ToE Access (FTA, dealing with the control of user sessions) also are of relevance. However, to focus the demonstration of SecReq, we limit the security requirement elicitation to the FIA class.

5.2.3 Step 3: Refine security objectives to sub-security-objectives

Step 3 takes the security objectives from Step 2 and refines each into sub-security-objectives, which are expressions of the security goals of the system at a grainer level of abstraction. As it is not easy to choose the appropriate abstraction level, we use Common Criteria part 2 to support this refinement as well. In particular, we use the refinement process between component classes and their contained component families to go from a security objective to sub-security-objectives.

Definition: **A Sub-Security-Objective** is a refinement of a security objective and is a detailed description of the relevant part of the secure environment for end-users of the system specified by the security objective.

The relationship between security objectives and sub-security-objectives in SecReq is an extension to the refinement step from class to family in Common Criteria Part 2. The specifics of the relationships are:

- A security objective can be refined into *One or more* sub-security-objectives connected by the operator "and",
- A security objective can be refined into *One or more* sub-security-objectives connected by the operator "or", and
- A security objective can be refined into a set of *at least three* sub-security-objectives connected by the operators "and" and "or".
- Nothing else is a sub-security-objective.

If we examine the IPTV security objective from above we see that the security features needed are identification and authentication of end-users and named groups of end-users. The two directly relevant FIA families are: FIA_UID (user identification) and FIA_UAU (user authentication). The other families of this class are also relevant, but to focus the demonstration of SecReq they are not considered further in this paper.

To ease the process of selecting the relevant families from a class, we use the functional requirement to support the process. If we look at the functional requirement VoD.1, the focus is primarily on the actual user identification and authentication activities, and for this only the FIA_UID (user identification) and FIA_UAU (user authentication) are directly relevant. The other families are also of relevance, but only indirectly. In practise, several of the security objectives and functional requirements will interrelate and thus other security objectives will ensure the proper protection of secrets etc.

IPTV Sub-Security-Objective 1: The IPTV service shall include features to define identification properties of end-users and named groups of end-users (derived from FIA_UID).

IPTV Sub-Security-Objective 2: The IPTV service shall include features to define authentication properties of end-users and named groups of end-users (derived from FIA_UAU).

5.2.4 Step 4: Refine sub-security-objectives to security requirements

Step 4 takes the result from Step 3 and refines the sub-security-objectives into security requirements supported by the refinement step between functional families and functional components in the Common Criteria. This means that the security requirements should be stated at an abstraction level similar to the security functional components of Common Criteria. The abstraction level should also align with the abstraction level of the functional requirements.

Looking at the IPTV case again, we start by examining the specification of the functional components of the two functional families FIA_UID and FIA_UAU.

The user identification family (FIA_UID) defines the conditions under which users shall be required to identify themselves before performing any other actions that are to be mediated by the security functions of IPTV, and which require user identification. The user identification family is made up of two sequential components: FIA_UID.1 and FIA_UID.2, where the latter represents a stricter enforcement than the first. FIA_UID.1 specifies timing of identification and is used in situations where one can allow users to perform certain actions before being identified to the system. FIA_UID.2 is used to specify that users must identify themselves before they can perform any other actions on the system. For the IPTV functional requirement that we are examining in this paper, it is necessary to prevent any actions of the user before identifying and authenticating them to the IPTV service. The reason for this is that we want to restrict the browsing of the VoD catalogue so that children can only view content pre-approved by the parents. This gives the following security requirement:

IPTV security requirement 1: The IPTV service shall have features to ensure that end-users and named groups of end-users always are identified before any other actions are allowed (FIA_UID.2).

The user authentication family (FIA_UAU) consists of eight components, where the first two are similar to the UID components in their specification of timing. These are FIA_UAU.1 (Timing of authentication) and FIA_UAU.2 (User authentication before any action).

Similar to user identification, we want to prevent any IPTV service action before user identification and authentication is performed successfully. This means that we need to specify the timing of the authentication the same way we did for the identification. Hence, we choose FIA_UAU.2 rather than FIA_UAU.1, as component 2 is used to restrict all actions and to disallow any action before authentication.

IPTV security requirement 2: The IPTV service shall have features to ensure that end-users and named groups of end-users always are authenticated before any other action is allowed (FIA_UAU.2).

5.2.5 Step 5: Refine security requirements to specific security requirements

Step 5 follows the same type of refinement process as for the other steps. The only difference is that the targeted abstraction layer consists of the security functional elements. The result of Step 5 should be specific security requirements; i.e., requirements expressions that are specific, measurable, achievable, realizable, and traceable (following the SMART principle discussed in Section 3.1). Please note that the SMART principle is not used to validate the requirements, but only to guide the formulation of the specific security requirements.

Going back to the IPTV case study we have two security requirements that we need to refine. We perform this

refinement by examining the associated functional components of each security requirement. For IPTV security requirement 1, we see that the FIA_UID.2 component has one associated functional element, which is FIA_UID.2.1. This element specifies that the security features of a system shall require each user to be successfully identified before allowing any other actions on behalf of that user. Tailored for the IPTV service and security requirement number 1, this gives the following specific security requirement:

IPTV specific security requirement 1: The IPTV service shall require all end-users and named groups of end-users to be successfully identified before allowing any IPTV service actions to be executed on behalf of that end-user.

Specific security requirement number 2 is derived from the FIA_UAU.2 component, which also has one functional element, namely FIA_UAU.2.1. This element specifies that the security features of a system shall require each user to be successfully authenticated before allowing the execution of any other actions on behalf of the end-user. Tailoring this to the IPTV service by taking security requirement number 2 into consideration gives the following:

IPTV specific security requirement 2: The IPTV service shall require all end-users and named groups of end-users to be successfully authenticated before allowing any IPTV service actions to be executed on behalf of that end-user.

As can be seen from the above specific security requirements, there are still some security decisions left to be made. E.g., the mechanisms for identity control need to be specified.

Conforming to the SMART principle does not guarantee anything in respect to the actual fulfilment of a security requirement. Also, there may be dependencies and interrelations between the security requirements. As identification is a pre-requisite for authentication, there is clearly a dependency between these two specific security requirements. These issues are handled in the analysis step of SecReq (Step 6) discussed in Section 5.3. However, first we will give a short example of how the security-related heuristics and HeRA were used to support the activities of Steps 1-5.

5.2.6 Step 1 – 5: Heuristic Support for Security Requirements

Throughout Steps 1-5 of the SecReq process, the HeRA tool offers support. The HeRA tool observes requirements input and raises warnings and hints when security-related input is detected. This additional layer facilitates reflection and raises awareness for security issues whenever requirements are captured and documented. It also helps in classifying requirements according to their abstraction level. This is done by dialogue based wizards.

Preparation: Before the HeRA heuristical editor can analyse and criticise security-requirements-elicitation, it has to be *seeded* with an initial set of heuristics from the security domain. We start with the stereotypes and tags provided in UMLsec, to create an initial set of heuristics. These heuristics range from a certain pattern of interaction in a Use Case to simple keywords (e.g. web, online, etc.).

HeRA can observe a wide range of heuristic rules. To get started, we only need to create an initial set that is useful in our context. Additional security related patterns and keywords will appear over time and lead to an evolutionary growth of the rule set when more experience is collected.

The advantage of starting with UMLsec stereotypes as a seed is that these are directly linked to a pre-defined and well tested set of security related issues. Furthermore, UMLsec makes a good starting point as we can configure the security related advices in HeRA based on the definition of the UMLsec stereotypes and thus make direct use of the implicit security experience captured in UMLsec. E.g., by linking the security requirement to one of the UMLsec keywords we can provide step-by-step support in formulating security requirements to fulfil the SMART principles. In addition, UMLsec includes analysis mechanisms that can be used for tracing security requirements into solution design.

Identification of potential security requirements: We illustrate how the HeRA tool supports any of the five steps with a simplified example. The example starts with a discussion with the technical experts involved in the IPTV project. During this discussion, one of the domain experts formulates a requirement - she is not aware whether it contains security-related aspects:

> Customers should be able to rent a movie over the Internet. This will be movie-on-demand.
> (Statement 1)

One of the participating domain experts from a company types this requirement into the HeRA heuristic-based editor. The editor automatically applies the related set of heuristics and fires two reactions:

> **(heuristic 1) «rent»** You seem to talk about a rental service. Is it supposed to be fair service delivery?
>
> **(heuristic 2) «Internet»** Use of the Internet makes this a security-challenged requirement. You may want to consider different options.

The HeRA screen is shown in Figure 10, where the typed text is on the left-hand side and the HeRA reaction is on the right-hand side. In a group of domain experts with little security experience, this reaction may cause confusion and in-depth discussions. They may not know what "fair service delivery" means, so they press "more" and access the additional information shown in Figure 10. Further comments

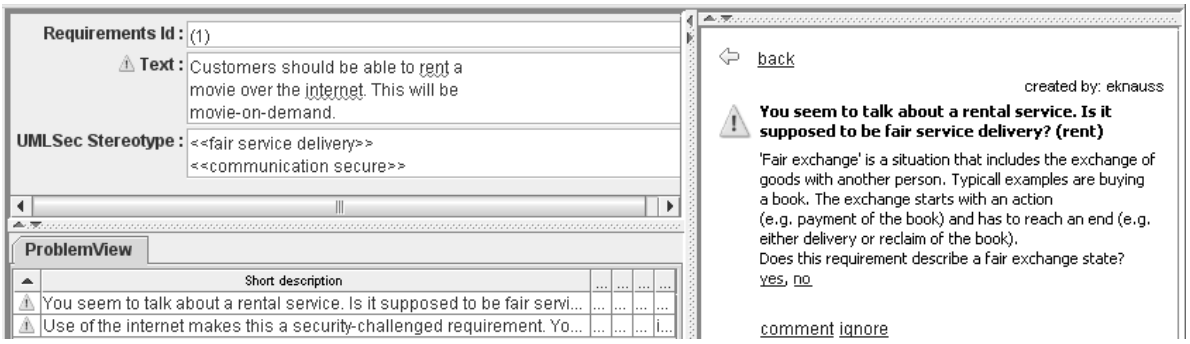

Fig. 10 Screenshot of a heuristic security warning in HeRA.

explain the issue of fair service delivery. By the next warning of the same type, the domain experts will know the meaning of this already and will not need to read the explanation again. Domain experts agree: they want to ensure *fair service delivery* in this case.

> Domain expert clicks on "yes" button: fair service delivery. (Statement 2)

By using the UMLsec stereotypes and associate tags (such as «fair service delivery», «Internet») during the elicitation of general requirements we are able to mark and link statement (1), the decision made in statement (2), and the UMLsec capabilities of dealing with design models for fair service delivery. Adding administrative data on participants, time, etc., provides an excellent basis for tracing, despite the informal and open setting.

Refinement of identified security requirements: In the scenario, statement (2) triggers an assistant that enriches the security information by asking follow-up questions:

> **(heuristic 1.2)** You specified this requirement (1), is it supposed to be a 'fair service delivery'?
> Please indicate which action causes the exchange to start (e.g. payment) and which action concludes it (e.g. delivery).

At that point, the fair service delivery annotation will be used to remind the designer when he starts building design UML models. Note that this mechanism can be used throughout the different abstraction levels in the steps 1 – 5 in order to map requirements to corresponding UMLSec stereotypes. The information captured by HeRA can be handed to the UMLSec Tools via XMI.

5.3 Step 6: Security Requirements Analysis with UMLsec

We now continue with *Step 6* of the SecReq approach in order to analyse the security requirements. We capture the specific security requirements and integrate them into UML diagrams by using the UMLsec stereotypes. In parallel to the UMLsec models, we develop a security goal tree that keeps track of the security objectives and their sub-objectives that need to be enforced, and their mutual dependencies. Our aim is thus to provide a satisfactory level of confidence that the design will correctly enforce the security requirements, in a way that allows one to trace back to the security requirements from the security design models.

5.3.1 Requirements: use case diagrams

To initiate the security analysis, use case diagrams are used to capture the IPTV security requirements that were elicited in Steps 1-5 of the SecReq approach, as explained in the previous subsections. Continuing with the IPTV case study at Step 6 of the SecReq approach, Figure 11 gives the use case diagram describing the situation to be achieved together with the initial (trivial) goal tree. An end-user or named group of end-users may request IPTV services (e.g. Video on Demand (VoD), Internet services (VoIP), etc.) from the providers (e.g. access network provider, service provider, IPTV provider and content provider). Prior to the service request, the user needs to complete the subscription formalities for any service or group of services. When subscribed, the user needs to prove her identity to the provider before carrying out the service request. If the service delivery is initiated, this means that the provider has already successfully recognised the user. Depending on the service type, the provider continues to deliver the service and finally finishes delivery whenever requested by the user. Service delivery and the end of the delivery need to occur in such a way that the transaction is fair from the point of view of both ends (in that the user only pays for the services requested and received, and in that the service provider delivers exactly what the user pays for). This security requirement can be specified using the «fair service delivery» stereotype, which has the associated tagged values {start}, {continue}, and {stop} with system actions as values. More specifically, this stereotype is supposed to capture the requirement that, when the workflow of the transaction will be later specified (i.e. using an activity diagram), that

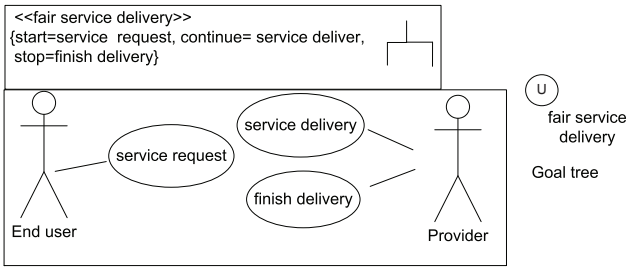

Fig. 11 Use case diagram and goal tree

workflow will also have to satisfy the «fair service delivery» property, as defined below in Sect. 5.3.2. Correspondingly, the security goal is at this stage in the goal tree marked with "U", which stands for "undetermined"; that is, it is not yet known whether the goal will be *satisfied* ("S") or *denied* ("D") [14].

5.3.2 Analysis: activity diagrams

The use case diagrams that capture the IPTV specific security requirements are now elaborated further using activity diagrams. Figure 12 elaborates the use case of Figure 11 using an activity diagram, and provides the associated and refined goal tree. As can be seen in the figure, the stereotype «fair service delivery» with the associated tags {start}, {continue}, and {stop} has now been refined. All three tags take (service, action) pairs as values, where service specifies the type of service that the user request and action specifies the possible actions involved within the requested service, and possible actions are: subscription request, service request, deliver service and finish delivery.

The refined scenario and analysis start with the user requesting an IPTV service using the tagged value {start=subscription request}. This request is eventually approved by the provider upon fulfilment of certain pre-defined conditions (such as payment, signature of agreement etc.). We omit the details here and just mention that this happens between the 'subscription requests' activity on the user side and the 'id & auth data' activity on the provider side in Fig. 12. Then, the provider generates and sends identification and authentication data to the user. Upon reception of the user identification and authentication data from the provider, the user approves the subscription. The user then issues a service request and sends the earlier received subscription authorization user identity and authentication data. If the authorization is successful, the user receives authentication provable information from the provider as a proof of authentication. This is shown by the «authentication provable» stereotype with tags {action} and {data}. This stereotype specifies that the user has to provide proof of authenticity before the provider initiates the delivery of service and is therefore specified as a security requirement in Fig. 12. The {action} tag holds the list of actions (e.g. subscription request, generate IDAuth, service request, prove authentication, authSucceed, service delivery, and finish deliver). The {data} tag contains a list of values associated with the actions (e.g. authentication and identification, authentication proved). The security requirement («authentication provable») is satisfied if each execution of the activity diagram fulfils the following two conditions:

- whenever the {action=service request} is executed by the subscribed user, then the {action=prove authentication} is eventually executed by the provider;
- whenever the user has successfully authenticated by sending {data= identification and authentication}, the provider is required to send {data=authentication proved} and continue with {action=service deliver}.

These two constraints specify the conditions that need to be satisfied for the requirement («authentication provable») to be meet, which can be by automatically verified using the UMLsec tool suite [74]. What the UMLsec tool suite does is to check whether the activity diagram satisfies these constraints. The stereotype «authentication provable» refines the «fair service delivery» stereotype with associate tags {start}, {continue}, and {stop}. We therefore extend the goal tree addressing «fair service delivery» and define 'authentication provable' as a sub-security objective of 'fair service delivery'.

5.3.3 Design: object diagrams

We consider every end-user to have received identity and authenticity data at the time of subscription. This identity information needs to be unique so that each user can provide a non-forgeable identity to the provider. The stereotype «non forgeable identity» has associated tags {group}, {subscription}, and {user_identity}. Tag {subscription= subs_number} specifies the subscription number. We assume that every group would be under a specific subscription number. Therefore tag {group=subs_number.group_number} identifies user group. User can be individual end user directly under a specific subscription or under a group. The tag {user_identity} requires two different values *subs_info* and *ID*. The *subs_info* determines which subscription the user belongs to (e.g. directly under subscription or under a group). The *ID* is used to ascertain that each user should have unique identity values. This because subscriptions are used to specify the services available to a specific user or group of users. Figure 13 shows an object diagram that displays an example for a possible combination between end users (who may be members of certain groups) and their IDs. It further extends our goal tree with the objective non-forgeable identity as a sub-objective of «authentication provable», in order to be able to uniquely identify every user. Prior to any service action specified by the stereotype «fair service delivery», the

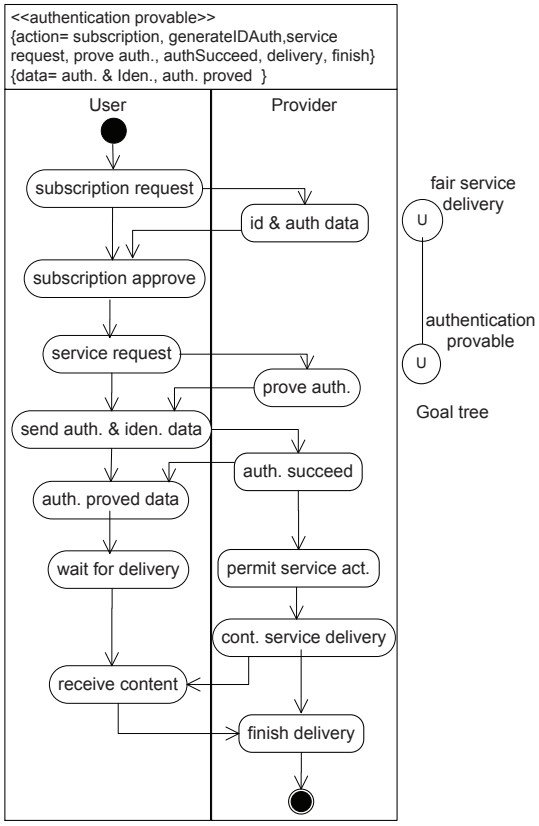

Fig. 12 Activity diagram and goal tree

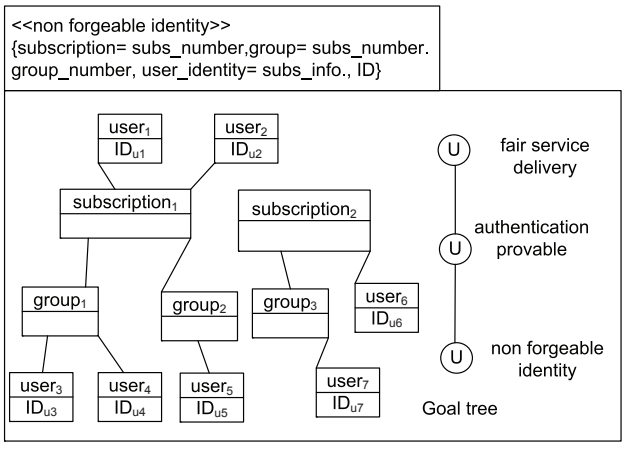

Fig. 13 Object diagram for unique user identity and goal tree

subscribed user is required to prove non-forgeable identity and authenticity.

Figure 13 shows unique user identity within groups or individuals under specific subscription. The relevant formulas for the unique user identity derive from the Figure 13 and are given below:

- $\forall\, user_1,\ user_2 \cdot\ (user_1 \neq user_2 \Rightarrow IDu_1 \neq IDu_2)$
- $\forall\, group_1 \cdot user_3,\ group_1 \cdot user_4 \cdot (user_3 \neq user_4 \Rightarrow IDu_3 \neq IDu_4)$
- $\forall\, group_1 \cdot user_3,\ group_2 \cdot user_5 \cdot (group_1 \neq group_2 \wedge user_3 \neq user_5 \Rightarrow IDu_3 \neq IDu_5)$
- $\forall\, subscription_1 \cdot user_1,\ subscription_2 \cdot user_6 \cdot (subscription_1 \neq subscription_2 \wedge user_1 \neq user_6 \Rightarrow IDu_1 \neg IDu_6)$
- $\forall subscription_1 \cdot group_1 \cdot user_3, subscription_2 \cdot group_3 \cdot user_7 \cdot (subscription_1 \neq subscription_2 \wedge group_1 \neq group_3 \wedge user_3 \neq user_7 \Rightarrow IDu_3 \neq IDu_7)$

The relevant rules based on the formulas are given below:

- Two different end-users under the same subscription should have different user IDs.
- Two different end-users belonging to the same group and under the same subscription should have different user IDs.
- Two different end-users belonging to different groups and under the same subscription should have different user IDs.
- Two different end-users belonging to different groups and under different subscriptions should have different user IDs.
- Two different end-users under the different subscription should have different user IDs.

5.3.4 Design: sequence diagrams

As part of the security analysis, one can specify the system behaviour using UML sequence diagrams, which in the case of the IPTV example are used to describe the communication between the end-user and the provider. We assume the communication link to be untrustworthy on the underlying physical layer (e.g. an unprotected Internet link). We thus need to design protection to prevent an intruder from attempting to gain unauthorised access, or to obtain user authentication information.

Figure 14 explains the sequence of messages exchanged between the user and the IPTV service provider using a sequence diagram. The user subscription request is approved using the user profile management function by sending the unique ID and authentication data. Thus, only valid subscribed users can prove identity and authenticity. The sequence of message exchanges between the user and the provider needs to be secure. Therefore, we need to attain the security objective 'communication secure' so that message exchange between user and providers through the unprotected Internet link is prevented from any unauthorised access.

Our goal tree is thus extended by including the additional sub security objective «communication secure», as shown on the right side of Figure 14. To achieve this sub objective, we need to include cryptographic operations for encrypting

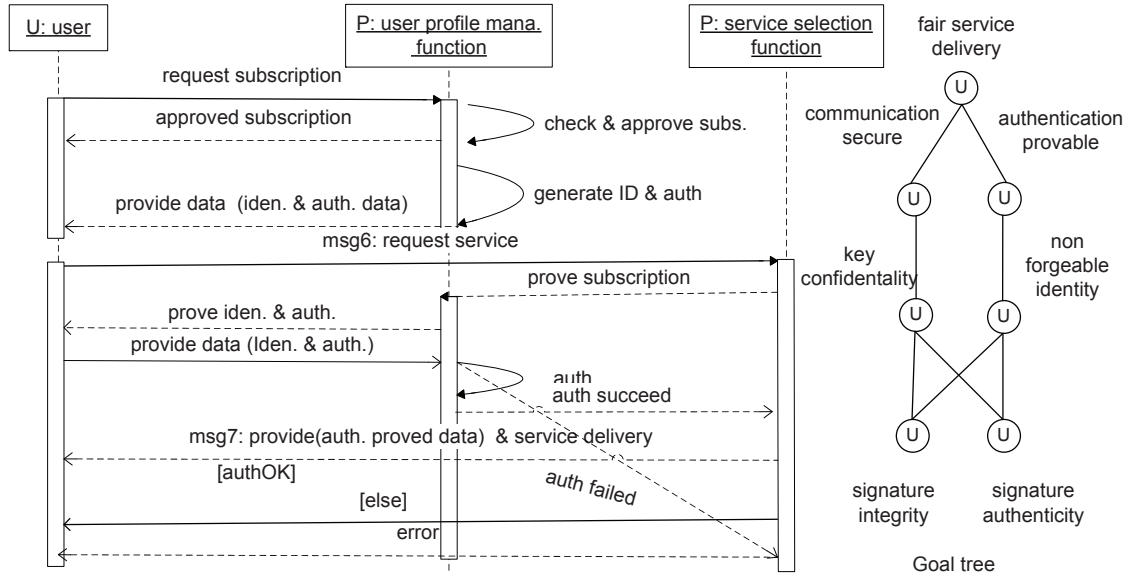

Fig. 14 Sequence diagram and goal tree

and signing messages and by that ensure secrecy of the information contained in the messages and message authenticity to prevent masquerade attacks and identity theft. Therefore, our goal tree further includes *key confidentiality* for encryption, *signature authenticity* for non-forgeable identity, and *signature integrity* so that the non-forgeable identities cannot be modified during transmission. We formalise two constraints that need to be satsfied before any service action can be executed:

- $\forall\, h \in\ Histories\, (\exists\, t \cdot h_t = msg6_{UE} \Rightarrow \exists t' < t \cdot h'_t = generateIDAuth_{UE})$
- $\forall\, h \in\ Histories\, (\exists\, t \cdot h_t = msg7_{SSF} \Rightarrow \exists t' < t \cdot h'_t = authSucceeded_{UP})$

The above equations based on Fig. 14 specify that no user-initiated request for service action by means of message $msg6_{UE}$ are processed before identification and authentication data have been generated by $generateIDAuth_{UE}$. Similarly, providers deliver the requested service by message $msg7_{SSF}$ only after the user has successfully authenticated, which results in the generation of the $authSucceeded_{UP}$ message. If the authentication fails for any reason, an error message is sent to the user instead of delivering of service. Note that this is an additional security objective to that identified in the previous subsections, which means that the UMLsec analysis has discovered additional security aspects, namely the need for *secure communication*.

In Figure 14, the actors of the IPTV system (i.e., *User* and *Provider*) exchange messages to support fair service delivery. The message exchange involves cryptographic operations (such as public-key encryption combined with digital signature) to ensure non-forgeable user authentication, and to make sure that the message exchange between the relevant objects is secure. Initially, the provider's user profile management function UPMF sends the user unique ID and authenticity data d encrypted with user public key K (denoted by $\{d\}_K$) when the user subscription is approved. The user decrypts the data with her private key K^{-1} resulting in $\{d\}_{K^{-1}}$ (for simplicity we assume the use of RSA type encryption). When the subscribed user U requests a service action, again the provider's user profile management function P:PPMF responds and requests the user to prove identity and to authenticate before the service action. This time the provider's user profile management function P:PPMF sends a session key K_s encrypted with the user public key K. Thus, all further communication between the user and the provider can take place using this session key for the given requested session. The subscribed user U sends her unique ID and authentication data signed with her private key and encrypted with the session key K_s . P decrypts the received message with the same session key K_s, verifies the signature with U's public key, and finally recovers the message. This way, the User's non-forgeable identity and integrity can be ensured.

5.3.5 Implementation: deployment diagram

Deployment diagrams describe the underlying physical layer. In SecReq, we use these to ensure that security requirements to the communication are met by the physical layer. Continuing with the IPTV case study, Figure 15 shows the deployment diagram for the physical layer of the communication link between the end-user node and the provider management function and service control node. Here, we assume that the end-user equipment node is connected with the provider nodes using the Internet Protocol over a network infrastructure with stereotype «Internet». Different nodes of

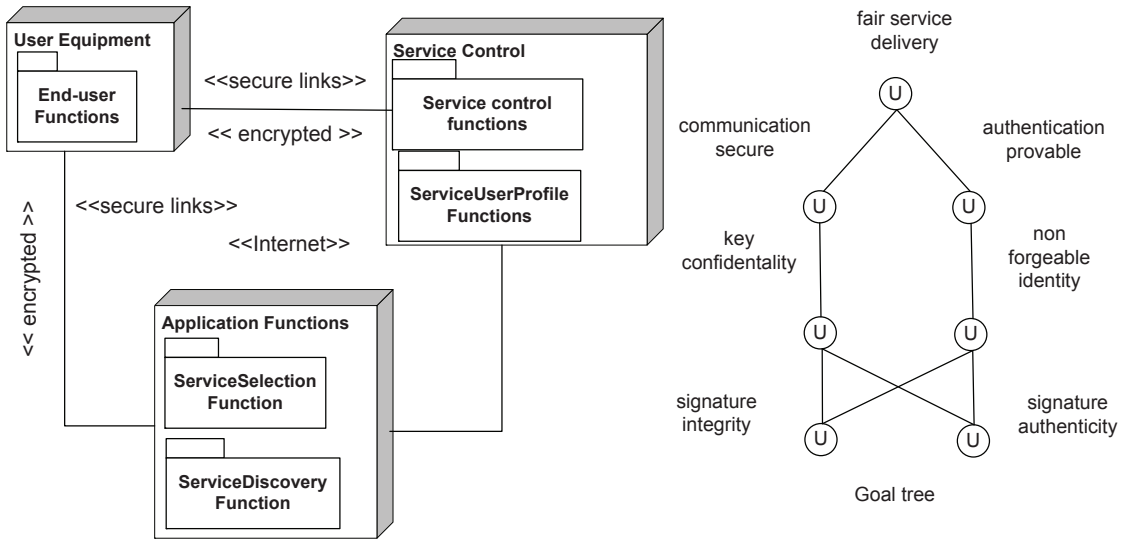

Fig. 15 Deployment diagram and goal tree

the provider, such as the management function node and the service content control node, may be connected via Intranet or Internet. The system specification requires that the data exchanged in this invocation is guaranteed confidentiality, which is specified using the stereotype «secure links». This stereotype is used to ensure that security requirements on the communication between the nodes are supported by the physical situation. This stereotype requires the «encrypted» stereotype on the underlying communication link in cases of an Internet connection, as all user identification and authentication information must be protected from unauthorised users.

This concludes Step 6 of the SecReq approach, which has been demonstrated using the IPTV application.

6 Discussion

This section gives an overview of the lessons learnt from using SecReq for security requirements elicitation in the IPTV project, and provides a general discussion of SecReq by highlighting some of the strengths and weaknesses of the methodology.

6.1 Lessons learnt from the IPTV project

Elicitation following the methodology resulted in 46 IPTV security requirements, where 19 were common IPTV security requirements, 13 were content specific requirements, 9 were network and service specific requirements, and 3 were availability specific requirements.

The IPTV work was reasonably successful compared to similar efforts. The reason for this was three-folded: (1) the methodology used was step-wise in an iterative and facilitated rapid feedback loop, (2) the requirements engineering team worked closely together with the rest of the development (standardization) team, which is a result of the working method of ETSI with formal meetings and approval procedures, (3) some of the members of the requirements engineering team had good connections with the stakeholders involved and were good communicators. Overall, the time spent on elicitation was effective and relatively short compared to similar projects that did not follow the step-wise process. This was because the methodology forced a particular structure on the elicitation process and made it possible to set reasonable deadlines. What was not achieved was a complete tracing and analysis of the requirements, as this part of SecReq was developed later. This is now being done, but has yet to be completed.

The IPTV project had a small core requirements drafting group, which seems to comply well with the underlying iterative and feedback-oriented process of SecReq. Rather than having a large elicitation group, our experience using the methodology indicates that it might be more effective to separate into a core and a supportive elicitation group. Such a distribution of roles makes the people involved accountable for the progress. This ensures that everybody involved knows who is responsible for what and when and in which form it should be delivered. This avoids deadlocks, which happens when several activities rely on the presence of one or a few persons (such as a security expert or instructor). SecReq enables and encourages feedback between core and supportive groups at the end of each step. However, the methodology does not directly specify any distribution of work or roles. This was made explicit for the IPTV case. A desired extension of the methodology would be to enable the users of the

methodology to specify and model the distribution of roles and responsibilities, and for SecReq to keep track of these. In addition, it will ease the use of SecReq in practise if the methodology is extended with a precise description of the expected abstraction level of the input and output from each step, and how these relate to the activities of each step.

6.2 Evaluation of strengths and weaknesses of SecReq

From a technical point of view, the most difficult task of the methodology is step 1, where security objectives are identified from functional descriptions, such as functional requirements. This has been the observation from several projects using SecReq to elicit security requirements. Step 1 requires expertise on at least three dimensions: (i) information structuring and analysis, (ii) requirements engineering, and (iii) security. The reason is that it is rarely intuitive what the overall security goals and objectives are, and it is not easy to simply extract these from highly abstract system information, incomplete sets of functional requirements and early draft system architecture. HeRA provides some support, but there is definitively room for improvement. The obvious cases are easy to identify. However, often the security association is not as obvious as that demonstrated in Section 5. Hence, future work includes investigating the details of the activities involved in Step 1 to better understand how this step can be supported, and in particular how HeRA can be extended to provide even better support for identifying potential security aspects from functional descriptions. The success rate of this activity is currently not satisfactory and it is still so that the presence of a security expert or instructor is crucial for the successful execution of this step. However, as more security knowledge and experience are formulated as security-related heuristics in HeRA, we foresee that step 1 will be less dependent on the presence of security expertise.

The disadvantage of the Common Criteria is its size and that it requires security expertise to benefit from its advices. SecReq deals with this problem to some extent by adopting parts of the ETSI method which include guidelines on how to affiliate the security functional component part of the standard (part 2) when eliciting security requirements. However, there is still work to be done before stakeholders with none or little security expertise can take full advantage of the Common Criteria. One possibility is to work out a security requirement repository from the security functional components and formulate these as security-related heuristics in HeRA.

Nevertheless, even if one is able to elicit the correct set of security requirements, there is still no guarantee that these requirements will be correctly represented in first the solution design and then the implementation. This gives rise to the need for tracing security requirements to the solution design and finally to the implementation. Hence, it is important not only to be able to elicit security requirements, it is equally important to ensure requirements fulfilment at the later stages of the development. The first transformation in this perspective is from security requirements to secure design. SecReq uses UMLsec for requirements tracing and fulfilment analysis. The main advantage of using UMLsec is that it includes tool-support that can automatically verify whether the security requirements are correctly addressed by the design. UMLsec thus extends SecReq from being a pure elicitation technique to become an elicitation and tracing technique. This is a strong benefit for SecReq in practical use, and means that SecReq can be used to bridge the gap between security requirements elicitation activities and development activities. Often these two types of activities are carried out separately with little or no interaction, which may result in that the end product contain few or even none of the identified security requirements.

However, SecReq is limited to supporting the design phase with UML notation, and hence, solution designs must be specified using UML diagrams. This is reasonable as the industry does use UML in development. However, the level of diagram details needed for a proper requirements analysis is not usually provided. E.g., often, only UML class and sequence diagrams are used in industrial development projects. This means that to exploit the full abilities of the UMLsec security analysis tools, the user would need to construct the missing diagrams. However, this additional investment may be worthwhile, depending on the required level of assurance. In terms of industrial application of an approach like SecReq, it is important to keep in mind that development projects often have many both conflicting and political issues that must be dealt with under limited time, resource and budget constraints. On the other hand, in case of limited resources, one can still apply the UMLsec analysis tools at whatever models that are available, which already deliver some (if relatively limited) level of assurance.

Security requirements must be unambiguous to set up a realistic analysis environment. More specifically, security requirements can be usefully categorized according to the SMART principles. This is true even though the current analysis activity in SecReq cannot, in its current version, address the *R* (Realizable) attribute of SMART in its every possible aspect: To check whether a security requirement is realizable the following two core questions are of most interest: (i) can the security requirement be satisfied given the system and physical constraints, and (ii) can the security requirement be satisfied given the project resource and schedule constraints. UMLsec does not cover(ii) and this must be checked otherwise, i.e., by checking resource and time employment estimates against the project resource and schedule plan. UMLsec can however address the first issue: One can for example use UMLsec to check for conflicts between

the security requirement and the existing design or physical infrastructure (for example modelled using a deployment diagram).

7 Related work

The proposed SecReq combines three different techniques, and this section discusses work of relevance to each of these. We first discuss the security standards, then related work about information flow modeling and heuristics, and finally security requirement elicitation and model based security engineering.

Security Standards: Security standards cover security management, risk management and secure systems development. The aims of these standards are either to support the overall and detailed security management in an organisation, or to support secure systems development by providing principles for security requirements specification and composition, building a security architecture, and creating secure code. Security management and risk management are not within the scope of our approach. So, we focus on security standards relating to secure system development.

Security evaluation standards concern the process of secure software development and give recommendation for this task. These standards target the IT product creation process (including requirements elicitation, design, implementation, and maintenance). Trusted Computer System Evaluation Criteria (TCSEC) [25] is the oldest known standard for evaluation and certification of information security in IT products. The standard was developed by the Department of Defence (DoD) in the United States in the 1980s. TCSEC evaluates systems according to the six predefined classes: C1, C2, B1, B2, B3 and A1. These classes are hierarchically arranged, meaning that A1 is the strongest and C1 is the weakest (actually TCSEC groups these classes into four categories; A, B, C and D, but category D is not used for certification). Each class contains both functional and assurance requirements. The functional requirements are divided into authentication, role based access control, obligatory access control and logging and reuse of objects. TCSEC was also known as the Orange Book and targeted military IT systems development. The standard was to some extent also used to evaluate industrial IT products without much success.

The United Kingdom, Germany, France and the Netherlands produced versions of their own national evaluation criteria as a response to the development of TCSEC. These were harmonised and published in 1991 under the name Information Technology Security Evaluation Criteria (ITSEC) [26]. ITSEC certification of a software product means that users can rely on an assured level of security for any product they are about to purchase. As for TCSEC, ITSEC certifies products according to the predefined classes of security (E0, E1, E2, E3, E4, E5 and E6). The standard was mainly a European standard. In addition, there is the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) [35]. CTCPEC combined the TCSEC and ITSEC approaches and was published in 1993 by the Canadian intelligence agency Communications Security Establishment (CSE).

TCSEC, ITSEC and CTCPEC have since been harmonized into the standard ISO 15408:2005 "Common Criteria for Information Technology Security Evaluation" [17] (or short: Common Criteria) by the International Organization for Standardization (ISO). The idea behind the Common Criteria was to merge the existing approaches into a world-wide framework for evaluating security properties of IT products and systems. The standard incorporates experience from TCSEC, ITSEC, CTCPEC and other relevant standards and industrial experience, and provides a common set of requirements for the security functions of IT products and systems. The standard also provides a program called "Arrangement on the Recognition of Common Criteria Certificates in the field of IT Security (CCRA)" and an evaluation methodology called "Common Methodology for IT Security Evaluation (CEM)". Together these ensure equality and quality of security evaluations, and that results from independent evaluations can be compared to aid decision makers (e.g. customers) in choosing among alternative IT products.

Lately several attempts have been made on tailoring the Common Criteria to various application domains. ETSI has undertaken several years of work on tailoring the Common Criteria to the Telco domain in the TISPAN program. SecReq builds on this experience and has adapted parts of the ETSI method [73,40]. SecReq extends the ETSI method by making the underlying requirement refinement process explicit in our step-wise process and by semi-automatic requirements identification and formulation support and by requirements tracing to secure design.

Information flow modelling: Information flow models were used by Winkler to increase traceability in software projects in [78]. In [23], Damian et al. use social networks to describe communication in software projects. She differentiates media from transfer information, and identifies patterns like "bottleneck". In [65], Schneider et al. propose a simple graphical notation for describing the flow (path) of information such as requirements. Security requirements are a special case of requirements. This work distinguishes between so-called "solid" (document-based) and "fluid" (e.g. spoken, email, informal) representations. Unlike the work of Winkler in [78], fluid information is modelled explicitly. Dashed lines and faces denote flow and storage of fluid information, whereas solid lines and document symbols represent solid information representation. In [70], Stapel et al. show interesting observations on this modelling approach in a financial institution. In [4], Allmann et al. and in [69], Stapel et al.

further used it in the automotive industry to describe and improve the relationship between a car company (OEM) and its subcontractors. Specific support tools can be built once an information flow problem has been identified [64]. However, making information flow explicit across solid and fluid representations stimulates heuristic approaches that can be applied before any given solid document exists. In this paper, figure 4 shows information flow model, and figures 7 and 8 show solid and fluid information flow within the context of the Common Criteria and the SecReq approach.

Heuristics: Computer-based feedback on requirement documents has often been reported to enhance quality of requirements specifications. Most of these approaches are focused on automatic evaluation of the Software Requirements Specifications (SRS) [29,77,58].

In [77], Wilson et al. define this approach as a quality model for the SRS and computer accessible indicators of the quality, as in the ARM tool. These indicators can point to possible issues in the SRS. Indicators are mostly not only based on keyword lists, but also include some more sophisticated natural language (NL) processing (e.g. for detecting under-specification in the QuARS tool [28,29]).

In [28], Fabrini et al. report that QuARS criteria can effectively evaluate SRS's quality. They also claim that QuARS not only helps to assess the quality of the SRS, but also helps to improve it. However, no quantitative evaluation of this aspect can be found. In [58], Melchisedech takes this approach one step further: ADMIRE relies upon specific process and information models that allow one to evaluate the dependencies between requirements more closely. As opposed to these approaches, we do not analyse the requirements after their creation to improve the documentation. Instead, we aim at constructively enhancing the knowledge about the system to construct. In addition, we focus on the narrowed domain of elicitation of security related requirements. This helps to capture experiences of security experts and apply them very early during requirements engineering – potentially even before comprehensive documentation exists that could be reviewed by these experts.

In [10], Breaux et al. work on extracting the security requirements from legal documents. Like our heuristics, their work is also based on analysing natural language. However, their work concentrates on the gap between legal and product requirements. We do consider legal documents to be a valuable source of security requirements, but in this paper we concentrate on another source: implicit security requirements hidden within the product requirements. We also concentrate on capturing the experience of security experts, which leaves us with a heuristic approach as opposed to the systematic analysis of codes of law.

HeRA has been presented in previous work [49,50] by Knauss et al., where the main focus was on using heuristics to constructively improve quality of requirement artefacts. In [49], we showed how HeRA could be integrated with complementing analytical quality assurance measures. In the SecReq approach, we extend the use of HeRA, and in particular HeRA's established mechanisms to identify potential requirements, with security relevant heuristics. In [49], Knauss demonstrated the ability to improve a requirement defect based on local feedback. However, this is not sufficient for SecReq that uses and refines knowledge about possible security requirements at all stages throughout the development process. The Common Criteria help to refine such security requirements candidates, if necessary. These refinement steps can be supported by HeRA for the refinement of the collected additional information. This information is handed to the UMLsec approach, where it is used to consistently insert tagged values in the UMLsec models. Based on this information, we can automatically create traceability links from the initial requirements over the refined security requirements to the UMLsec models.

Security Patterns Security patterns help to leverage experiences during design and implementation of systems [67]. By organising these experiences as patterns, developers can judge whether the pattern matches the given situation. This leads to the problem that developers need to know the pattern, which cannot be assumed by non-security-experts. Thus, it could pay off to create heuristics based on the security patterns. This is not currently part of our approach but will be addressed in future work.

Security Requirement Elicitation: Over the last few years, a significant amount of work has been carried out on security requirements elicitation. We discuss those approaches that are closely related to our approach.

In [75], Toval et al. discuss the Spanish Public Administration's adaptation of the Common Criteria framework, MAGERIT, and an extension that focuses on requirements reuse, SIREN (SImple REuse of software requiremeNts). While MAGERIT focuses on risk identification and management and includes processes to control such activities, SIREN provides a reuse repository that structures security requirements into categories and hierarchies. Among other things, this repository can be used to support the treatment assignment part of MAGERIT. E.g., security measures used to treat the identified risks in MAGERIT can be translated into reusable security requirements in SIREN. SIREN also provides requirements tracing by means of inclusive and exclusive dependencies, and forwards and backwards tracing. Inclusive dependencies specify 'AND' conditions between two or more requirements, also across requirements documents, while exclusive dependencies specify 'OR' conditions between requirements, i.e., mutually exclusive. Forwards tracing are from requirements to later development artifacts, such

as design, and backwards tracing are from lower refinement levels to more abstract expressions. The authors also talk about providing explicit tracing by using UML notation. SecReq is similar to SIREN, but provides requirements reuse by means of security related heuristics stored in HeRA. HeRA also provides semi-automatic requirements elicitation and formulation support, which SIREN does not, and SecReq includes a formal requirements tracing process and notation using the UML notation UMLsec.

In [60], Mouratidis et al. introduce Secure Tropos as a security goal driven approach for integrating security related concepts into the Tropos methodology. The approach considers security constraints such as privacy, integrity, etc. throughout the development stage, from the early requirements analysis to the implementation. These constraints can be effective in eliciting and analysing the security requirements. In [61], Mouratidis et al. combine Secure Tripos with the UML security extension UMLsec [45] and by that advance Secure Tropos from merely dealing with architectural design to also handle detailed design.

In [21], Crook et al. and in [36], Haley et al. formulate a vision for the requirements engineering community towards providing a "bridge between the well-ordered world of the software project informed by conventional requirements and the unexpected world of anti-requirements associated with the malicious user". In [36], Haley et al. propose a framework for representation and analysis of security requirements. It allows the security engineer to represent and analyse security requirements. In [33, 34], Giorgini et al. and in [54], Massacci et al. propose an extension of the i*/Tropos requirements engineering framework to deal with security requirements.

In [68], Sindre et al. propose a misuse case driven approach to elicit security requirements at an early stage. A visual link is established between use cases and misuse cases that are used to guide the analysis of functional requirements against security requirements and the threat environment. In [59], Mellado et al. propose a Common Criteria based Security Requirements Engineering Process (SREP). SREP makes use of several Common Criteria constructs, such as the security functional components, protection profile, and security assurance components to elicit and analysis security requirements. Another approach similar to SREP is the Security Quality Requirement Engineering Methodology (SQUARE) [57]. SREP and SQUARE are asset-based and risk-driven methods both providing a nine steps procedure for eliciting, categorising, and prioritising security requirements. However, SREP differs from SQUARE as it integrates knowledge and experience from the Common Criteria and Information Security Standards, such as ISO/IEC 27001 [1], while eliciting security requirements. In [39], Islam et al. identify a set of security risks and its impact to software quality attributes and investigate how eliciting security requirements at an early stage can address these risks. Human factors relating to overall organisational security in particular with security risk management are represented by Islam et al. in [38].

In [76], Whittle et al. present an executable misuse case modelling language which allows modellers to specify misuse case scenarios in a formal yet intuitive way and to execute the misuse case model in tandem with a corresponding use case model. In [79], Yskout et al. present an approach for the transformation of security requirements to software architectures. In [5], Arenas et al. discuss the use of requirements-engineering techniques in capturing security requirements for a Grid-based operating system. In [27], Elahi et al. examine how conceptual modelling can provide support for analysing security trade-offs, using an extension to the i* framework. In [32], Flechais et al. present an approach which integrates security and usability into the requirements and design process, based on a UML meta-model for defining and reasoning over the system's assets.

The work presented here differs from these approaches in that SecReq provides explicit support for tracing security requirements to security design, provides heuristics-based tool-support for reusing security knowledge, and includes Common Criteria best practises developed at ETSI for security requirements elicitation and specification support.

Model-based Security Engineering: In [6], Baldwin et al. and in [48], Kearney et al. use UML for risk-driven security analysis that focuses on the assessment of risk and analysis of requirements for operational risk management. In [63] Ray et al. and in [37] Houmb et al. propose to use aspect-oriented modelling for addressing access control concerns. Functionality that addresses a pervasive access control concern is defined in an aspect. The remaining functionality is specified in a so-called primary model. Composing access control aspects with a primary model then gives a system model that addresses access control concerns.

In [7], Basin et al. and in [12], Brucker et al. show how UML can be used to specify access control in an application and how one can then generate access control mechanisms from the specifications. The approach is based on role-based access control and gives additional support for specifying authorisation constraints. In [3], Alam et al. present usage scenarios for access control in contemporary health care scenarios and show how to unify them in a single security policy model. Based on this model, the SECTET [2] framework for Model Driven Security is specialised towards a domain-specific approach for the health care scenarios, including the modelling of access control policies, a target architecture for their enforcement, and model-to-code transformations. A formally based process for model-based security engineering was presented in [11].

The approach for model-based security engineering using UML used in this work (namely UMLsec) is presented in

[45]. Tool support for UMLsec is presented in [47] and some industrial applications were reported in [9,41,46].

In order to integrate the UMLsec method into the SecReq approach presented in this paper, it had to be adapted and extended in non-trivial ways. For example, linking the UMLsec models to the input from the Common Criteria on the one hand and using the HeRA-toolset as tool-based process on the other, SecReq incrementally develops the UMLsec models in parallel with the incremental development of an associated goal-tree, as explained in Section 2.3. Also, we had to develop an approach which allows the user to trace the security requirements through the SecReq process using the UMLsec models, as visualized in Fig. 9. Finally, to apply UMLsec in the industrial application at hand, we had to extend it with several new stereotypes which deal with specific security requirements in the telecommunication application domain, as explained in Section 5.3 (as is usually the case when applying UMLsec to a new domain; compare [9,41, 46] for similar situations).

8 Conclusion and Future work

The paper presents SecReq, a Common Criteria driven security requirements elicitation and tracing approach. SecReq consists of the following three techniques:

(i) Common Criteria to support security requirements elicitation, and in particular, the underlying security requirements elicitation process of Common Criteria,
(ii) The HeRA tool with its security-related heuristics, which are used to discover additional security aspects in functional descriptions that are not covered or simply overlooked, and
(iii) UMLsec for security requirements fulfilment and traceability analysis.

The main aim of SecReq is to extend security requirements engineering by seamlessly integrating elicitation, traceability and analysis activities. The motivation for this is that requirements engineering activities are often executed by other people than those writing the code, and often without much contact between the two groups. This applies in particular to security requirements, which is a major quality attribute of today's systems. It is therefore important to develop both the ability of the people involved in the development to identify potential security aspects, and the capabilities of the development team to solve these needs in practice through secure design. Currently, SecReq uses UMLsec analysis for tracing between requirements and design and vice versa. Future work includes extending the analysis capabilities into the code development phase, along the lines described in [80].

Future work also includes extending the heuristics with relevant regulatory constraints, such as Sarbanes Oxley (SOX), the EU Privacy Directive and similar. We also plan to look into how to use the heuristic concepts implemented in the HeRA tool to externalise a generic threat analysis. This can be done to some extent by having heuristics interact with users based on past experiences. Security experts should be able to phrase things like "Every time I have to explain the same threats associated with a fair exchange again" as a heuristic. This way the stakeholders not experts in security will profit as they get feedback at the earliest possible stage. Security experts will also profit, as they then can insert generic security expertise in a reusable manner, which leaves more resources to concentrate on the more project specific issues. Furthermore, SecReq already supports system evolution to some degree by means of a feedback loop. Currently, this helps reusing experience within SecReq and turns the approach into an iterative process for the secure system lifecycle. However, there is still work to be done on making this more seamlessly, and hence we plan to further investigate the challenge of system evolution for the case of secure systems development.

When it comes to the security-related heuristics, both the Common Criteria and UMLsec provide input to these. However, we still have not been able to fully explore the capabilities of Common Criteria. E.g., the security functional components can also be used as a security requirement repository, which would make the elicitation activities in Steps 1-5 easier. We plan to look into additional ways that we can support the first five steps of SecReq, and particularly the identification of the relevant classes, families, components and elements of the security functional components.

However, SecReq does not yet fully explore the capabilities inherited from the Common Criteria. E.g., the security functional components can also be used as a security requirements repository, which would make the elicitation activities in Steps 1–5 easier. We plan to look into additional ways to support the first five steps of SecReq, and particularly the identification of the relevant classes, families, components and elements from Common Criteria part 2.

References

1. ISO/IEC 27001:2005 Specification for Information Security Management, October 2005.
2. M. Alam, M. Hafner, and R. Breu. Model-driven security engineering for trust management in SECTET. *Journal of Software*, 2(1), February 2007.
3. M. Alam, M. Hafner, M. Memon, and P. Hung. Modeling and enforcing advanced access control policies in healthcare systems with SECTET. In J. Sztipanovits, R. Breu, E. Ammenwerth, R. Bajcsy, J.C. Mitchell, and A. Pretschner, editors, *Workshop on Model-based Trustworthy Health Information Systems (MOTHIS@Models)*, 2007.
4. C. Allmann, L. Winkler, and T. Kölzow. The Requirements Engineering Gap in the OEM-Supplier Relationship. *Journal of Universal Knowledge Management*, 1(2):103–111, 2006.
5. A. Arenas, B. Aziz, J. Bicarregui, B. Matthews, and E. Y. Yang. Modelling security properties in a grid-based operating system

with anti-goals. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security(ARES)*, pages 1429–1436, 2008.

6. A. Baldwin, Y. Beres, S. Shiu, and P. Kearney. A model based approach to trust, security and assurance. *BT Technology Journal*, 24(4):53–68, October 2006.
7. D.A. Basin, J. Doser, and T. Lodderstedt. Model driven security: From UML models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1):39–91, 2006.
8. V. Berzins, L. C. Martell, and P. Adams. Innovations in Natural Language Document Processing for Requirements Engineering. In Barbara Paech and C. Martell, editors, *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs: 14th Monterey Workshop 2007*, Lecture Notes In Computer Science, pages 125–146, Berlin, Heidelberg, 2008. Springer-Verlag.
9. B. Best, J. Jürjens, and B. Nuseibeh. Model-based security engineering of distributed information systems using UMLsec. In *29th International Conference on Software Engineering (ICSE 2007)*, pages 581–590. ACM, 2007.
10. T. D. Breaux and A. I. Antón. Analyzing regulatory rules for privacy and security requirements. *IEEE Transactions on Software Engineering, Special Issue on Software Engineering for Secure Systems*, 34(1):5–20, 2008.
11. R. Breu, K. Burger, M. Hafner, J. Jürjens, G. Popp, G. Wimmel, and V. Lotz. Key issues of a formally based process model for security engineering. In *16th International Conference "Software & Systems Engineering & their Applications" (ICSSEA 2003)*, 2003.
12. A.D. Brucker, J. Doser, and B. Wolff. A model transformation semantics and analysis methodology for SecureUML. In *MoDELS 2006*, volume 4199 of *Lecture Notes in Computer Science*, pages 306–320. Springer-Verlag, 2006.
13. ISO 15408:2007 Common Criteria for Information Technology Security Evaluation: Evaluation Methodology, Version 3.1, Revision 2, CCMB-2007-09-004, September 2007.
14. L. Chung. Dealing with Security Requirements During the Development of Information Systems. In *5th International Conference on Advanced Information Systems Engineering (CAiSE 1993)*, pages 234–251. Springer, 1993.
15. A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley Professional, January 2000.
16. Common Methodology for Information Technology Security Evaluation, Evaluation methodology, Version 3.2, Revision 2, CCMB-2009-09-004, September 2007.
17. ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2, CCMB-2007-09-001, CCMB-2007-09-002 and CCMB-2007-09-003, September 2007.
18. ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2: Part 1; General Model, CCMB-2007-09-001, September 2007.
19. ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2: Part 2; Security Functional Components, CCMB-2007-09-002, September 2007.
20. ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2: Part 3; Security Assurance Components, CCMB-2007-09-003, September 2007.
21. R. Crook, D.C. Ince, L. Lin, and B. Nuseibeh. Security requirements engineering: When anti-requirements hit the fan. In *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 203–205. IEEE Computer Society, 2002.
22. D. Damian, L. Izquierdo, J. Singer, and I. Kwan. Awareness in the Wild: Why Communication Breakdowns Occur. In *Proceedings of Second International Conference on Global Software Engineering*, pages 81–90, Munich, Germany, 2007.
23. D. Damian, S. Marczak, and I. Kwan. Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks. In *Proceedings of 15th IEEE International Requirements Engineering Conference (RE 2007)*, New Delhi, India, 2007.
24. A. M. Davis. *Just Enough Requirements Management: Where Software Development meets Marketing*. Dorset House Publishing, 2005.
25. Department of Defense. DoD 5200.28-STD: Trusted Computer System Evaluation Criteria, August 15 1985.
26. Department of Trade and Industry. The National Technical Authority for Information Assurance, June 2003. http://www.itsec.gov.uk/.
27. G. Elahi and E. Yu. A goal oriented approach for modeling and analyzing security trade-offs. In *ER 2007*, volume 4801 of *Lecture Notes in Computer Science*, pages 375–390. Springer-Verlag, 2007.
28. F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. An Automatic Quality Evaluation for Natural Lan-guage Requirements. In *Proceedings of the Seventh In-ternational Workshop on RE: Foundation for Software Quality (REFSQ 2001)*, Interlaken, Switzerland, 2001.
29. F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the use of an Automatic Tool. In *SEW '01: Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*, page 97, Washington, DC, USA, 2001. IEEE Computer Society.
30. G. Fischer. Domain-Oriented Design Environments. *Automated Software Engineering*, 1:177–203, 1994.
31. G. Fischer. Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments. *Automated Software Engineering*, 5:447–464, 1998.
32. I. Flechais, C. Mascolo, and M. A. Sasse. Integrating security and usability into the requirements and design process. *International Journal of Electronic Security and Digital Forensics*, 1(1):12–26, 2007.
33. P. Giorgini, F. Massacci, and J. Mylopoulos. Requirement engineering meets security: A case study on modelling secure electronic transactions by VISA and Mastercard. In I.-Y. Song, S. W. Liddle, T. W. Ling, and P. Scheuermann, editors, *22nd International Conference on Conceptual Modeling (ER 2003)*, volume 2813 of *Lecture Notes in Computer Science*, pages 263–276. Springer-Verlag, 2003.
34. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 167–176. IEEE Computer Society, 2005.
35. Government of Canada. The Canadian Trusted Computer Product Evaluation Criteria, January 1993.
36. C. B. Haley, R. C. Laney, J. D. Moffett, and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008.
37. S. H. Houmb., G. Georg, R. B. France, J. M. Bieman, and J. Jürjens. Cost-benefit trade-off analysis using BBN for aspect-oriented risk-driven development. In *Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems*, pages 195–204. IEEE Computer Society, 2005.
38. S. Islam and W. Dong. Human factors in software security risk management. In *LMSA '08: Proceedings of the first international workshop on Leadership and management in software architecture*, pages 13–16, New York, NY, USA, 2008. ACM.
39. S. Islam and W. Dong. Security Requirements Addressing Security Risks for Improving Software Quality. In *Workshop-Band Software-Qualitätsmodellierung und -bewertung (SQMB '08), Technical Report TUM-I0811, Technische Universität München, 2008,Munich, Germany*, 2008.
40. J. E. Rossebø and S. Cadzow and P. Sijben. eTVRA, a Threat, Vulnerability and Risk Assessment Method and Tool for eEurope.

In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 925–933. IEEE Computer Society, 2007.

41. J.Jürjens and R.Rumm. Model-based Security Analysis of the German Health Card Architecture. *Methods of Information in Medicine*, 47(5):409–416, 2008. Special section on Model-based Development of Trustworthy Health Information Systems.
42. J. Jürjens. Secure information flow for concurrent processes. In C. Palamidessi, editor, *CONCUR 2000 (11th International Conference on Concurrency Theory)*, volume 1877 of *Lecture Notes in Computer Science*, pages 395–409. Springer-Verlag, 2000.
43. J. Jürjens. Formal semantics for interacting UML subsystems. In B. Jacobs and A. Rensink, editors, *5th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS 2002)*, pages 29–44. International Federation for Information Processing (IFIP), Kluwer Academic Publishers, 2002.
44. J. Jürjens. Using UMLsec and goal-trees for secure systems development. In G. B. Lamont, H. Haddad, G. Papadopoulos, and B. Panda, editors, *Proceedings of the 2002 Symposium of Applied Computing (SAC)*, pages 1026–1031. ACM Press, 2002.
45. J. Jürjens. *Secure Systems Development with UML*. Springer, 2004.
46. J. Jürjens, J. Schreck, and P. Bartmann. Model-based security analysis for mobile communications. In *30th Intern. Conference on Software Engineering (ICSE 2008)*. ACM, 2008.
47. J. Jürjens and P. Shabalin. Tools for secure systems development with UML. *Intern. Journal on Software Tools for Technology Transfer*, 9(5–6):527–544, October 2007. Invited submission to the special issue for FASE 2004/05.
48. P. Kearney and L. Brügger. A risk-driven security analysis method and modelling language. *BT Technology Journal*, 25(1), January 2007.
49. E. Knauss and T. Flohr. Managing Requirement Engineering Processes by Adapted Quality Gateways and critique-based RE-Tools. In *Proceedings of Workshop on Measuring Requirements for Project and Product Success*, Palma de Mallorca, Spain, November 2007. in conjunction with the IWSM-Mensura Conference.
50. E. Knauss, D. Lübke, and S. Meyer. Feedback-Driven Requirements Engineering: The Heuristic Requirements Assistant. In *International Conference on Software Engineering (ICSE'09), Formal Research Demonstrations Track*, Vancouver, Canada, 2009.
51. E. Knauss, K. Schneider, and K. Stapel. Learning to Write Better Requirements through Heuristic Critiques. In *Proceedings of 17th IEEE Requirementes Engineering Conference (RE 2009)*, Atlanta, USA, 2009.
52. S. N. Lindstaedt and K. Schneider. Bridging the Gap between Face-to-Face Communication and Long-term Collaboration. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, Phoenix, USA, Nov 1997. ACM.
53. M. Mannion and B. Keepence. SMART Requirements. *ACM SIGSOFT: SE Notes*, 20(2):42–47, April 1995.
54. F. Massacci, J. Mylopoulos, and N. Zannone. Computer-aided support for secure tropos. *Automated Software Engineering*, 14(3):341–364, 2007.
55. J. P. McDermott and C. Fox. Using Abuse Case Models for Security Requirements Analysis. In *Proceedings of the 15th Annual Computer Security Applications Conference*, pages 55–. IEEE Computer Society, 1999.
56. M.Deubler, J.Grünbauer, J.Jürjens, and G.Wimmel. Sound development of secure service-based systems. In Marco Aiello, Mikio Aoyama, Francisco Curbera, and Mike P. Papazoglou, editors, *Proceedings of the 2nd international conference on Service oriented computing(ICSOC)*, pages 115–124. ACM, 2004.
57. N.R. Mead and T. Steheny. Security quality requirements engineering (square) methodology. *SIGSOFT Softw. Eng. Notes*, 30(4):1–7, 2005.
58. R. Melchisedech. *Verwaltung und Prüfung natürlichsprachlicher Spezifikationen*. PhD thesis, Fakultät Informatik, Universität Stuttgart, Stuttgart, 2000.
59. D. Mellado, E. Medina, and M. Piattini. A common criteria based security requirements engineering process for the development of secure information system. *Computer standards & interfaces*, 29:244–253, June 2007.
60. H. Mouratidis, P. Giorgini, and G. A. Manson. Integrating security and systems engineering: Towards the modelling of secure information systems. In J. Eder and M. Missikoff, editors, *15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, volume 2681 of *Lecture Notes in Computer Science*, pages 63–78. Springer-Verlag, 2003.
61. H. Mouratidis, J. Jürjens, and J. Fox. Towards a Comprehensive Framework for Secure Systems Development. In Eric Dubois and Klaus Pohl, editors, *CAiSE*, volume 4001 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2006.
62. M. Polanyi. *The Tacit Dimension*. Doubleday, Garden City, NY, 1966.
63. I. Ray, R.B. France, Na Li, and G. Georg. An aspect-based approach to modeling access control concerns. *Information & Software Technology*, 46(9):575–587, 2004.
64. K. Schneider. Generating Fast Feedback in Requirements Elicitation. In *Requirements Engineering: Foundation for Software Quality (REFSQ 2007)*, 2007.
65. K. Schneider, K. Stapel, and E. Knauss. Beyond Documents: Visualizing Informal Communication. In *Proceedings of Third International Workshop on Requirements Engineering Visualization (REV 08)*, Barcelona, Spain, 9 2008.
66. D.A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.
67. M. Schumacher, E. F. Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons Ltd., 2006.
68. G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering Journal*, 10(1):34–44, 2005.
69. K. Stapel, E. Knauss, and C. Allmann. Lightweight Process Documentation: Just Enough Structure in Automotive Pre-Development. In Rory V. O'Connor, Nathan Baddoo, Kari Smolander, and Richard Messnarz, editors, *Proceedings of the 15th European Conference, EuroSPI*, Communications in Computer and Information Science, pages 142–151, Dublin, Ireland, 9 2008. Springer.
70. K. Stapel, K. Schneider, D. Lübke, and T. Flohr. Improving an Industrial Reference Process by Information Flow Analysis: A Case Study. In *Proceedings of PROFES 2007*, volume 4589 of *LNCS*, pages 147–159, Riga, Latvia, 2007. Springer-Verlag Berlin Heidelberg.
71. ETSI TISPAN. ETSI TS 182 027 V.2.0.0: IPTV Architecture; IPTV functions supported by the IMS subsystem. Standard, February 2008.
72. ETSI TISPAN. ETSI TS 182 028 V.2.0.0: IPTV Architecture; Dedicated subsystem for IPTV functions. Standard, January 2008.
73. TISPAN, ETSI. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN): Methods and Protocols; Part 1: Method and Proforma for Threat, Risk, Vulnerability Analysis. Technical Report ETSI TS 102 165-1 V4.2.1, European Telecommunications Standards Institute, 2006.
74. UMLsec tool, 2001-08. http://www.umlsec.de.
75. A. Toval, J. Nicolás, B. Morosa, and F. García. Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach. *Requirements Engineering Journal*, 6:205–219, 2002.
76. J. Whittle, D. Wijesekera, and M. Hartong. Executable misuse cases for modeling security concerns. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 121–130, New York, NY, USA, 2008. ACM.
77. W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated quality analysis of Natural Language requirement specifications. In *Proceedings of PNSQC Conference*, 1996.

78. S. Winkler. Information Flow Between Requirement Artifacts. In *Proceedings of REFSQ 2007 International Working Conference on Requirements Engineering: Foundation for Software Quality*, volume 4542 of *Lecture Notes in Computer Science*, pages 232–246, Trondheim, Norway, 2007. Springer Berlin / Heidelberg.
79. K. Yskout, R. Scandariato, B. D. Win, and W. Joosen. Transforming security requirements into architecture. In *International Conference on Availability, Reliability and Security*, pages 1421–1428, 2008.
80. Y. Yu, J. Jürjens, and J. Mylopoulos. Traceability for the maintenance of secure software. In *24th International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, 2008.