Complex Systems Informatics and Modeling Quarterly
CSIMQ, Issue 7, June/July 2016, Pages 1–24
Published online by RTU Press, https://csimq-journals.rtu.lv
http://dx.doi.org/10.7250/csimq.2016-7.01
ISSN: 2255-9922 online

Modeling Enterprise Authorization: A Unified Metamodel and Initial Validation

Matus Korman, Robert Lagerström and Mathias Ekstedt

Software Systems Architecture and Security Research Group, Department of Electric Power and Energy Systems, Royal Institute of Technology, Osquldas väg 10 100 44, Stockholm, Sweden

{korman, robertl, mekstedt}@kth.se

Abstract. Authorization and its enforcement, access control, have stood at the beginning of the art and science of information security, and remain being crucial pillar of security in the information technology (IT) and enterprises operations. Dozens of different models of access control have been proposed. Although Enterprise Architecture as the discipline strives to support the management of IT, support for modeling access policies in enterprises is often lacking, both in terms of supporting the variety of individual models of access control nowadays used, and in terms of providing a unified ontology capable of flexibly expressing access policies for all or the most of the models. This study summarizes a number of existing models of access control, proposes a unified metamodel mapped to ArchiMate, and illustrates its use on a selection of example scenarios and two business cases.

Keywords: Access control, authorization, enterprise architecture, EA modeling.

1 Introduction

Authorization and its enforcement (access control) have been a crucially important pillars of enterprise information technology (IT) security, both on a technical level (in computer systems, databases, networks, etc.) and an organizational level (access policy and its human enforcement). In parallel to their role in IT and IT architectures, authorization and access control are essential in physical premises such as airports and industrial facilities, thus it is easy to imagine the importance of authorization and access control being functioning appropriately and being well-aligned with the enterprise. However, major enterprise architecture (EA) modeling languages, such as ArchiMate [1], currently neither support modeling authorization and access control, nor provide extensions that would enable practitioners to do so in an elegant, defined, and generic manner.

Research and practice in the area of authorization and access control is decades old. In fact, they are even more older if we extend our view beyond the boundaries of IT. Within IT, a plethora of different access control models have been proposed (a subset of them is listed in Table 1). Several of them have become widely adopted in a variety of IT systems. For instance, discretionary access control (DAC) implemented using access control lists (ACLs) (e.g., see [2], p. 35) and role based access control (RBAC) [3], [4] resound most with their origins dating back to 70's and 80's, respectively. While these and a few more models have been employed extensively, there are some fresh candidates on the verge of larger-scale adoption, such as the attribute based access control models (ABAC) [5], [6], [7], [8], not to mention their more recent and sophisticated risk-adaptive variants [9], [10].

This study attempts to address the challenge of flexibly modeling policies of authorization according to the most well-known access control models, in terms of EA. The purpose of the study is three-fold: (1) to give an overview of well-known access control models; (2) to enable EA practitioners to easily model authorization in enterprise architectures; and (3) to aid the development of security policies in organizations. The study presents a number of existing access control models, describes them in terms of conceptual modeling, and proposes a unified metamodel (seen as an ontology or modeling grammar). The unified metamodel can be used to describe access policies, which any of abovementioned models or their combinations might represent, and which a single access control mechanism or a set of different mechanisms might implement – be it in a well-delimited scope such as within a single IT system, or across an entire enterprise. A unified metamodel is needed in order to (1) be able to combine the different models of access control in a single EA model and to (2) propose a single extension to an established EA modeling language. The proposed unified metamodel is formed as a prospective extension of the popular EA modeling language ArchiMate. Subsequently, five simple illustrative scenarios are presented, to exemplify several different ways of modeling authorization, to demonstrate and partially validate the applicability of the metamodel.

The study takes a standpoint in EA and EA modeling [11]. EA is defined as *"a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure;"* it attempts to capture the essentials of the business, IT and its evolution (ibid.). In this light, EA can be adopted and used for a multitude of purposes including documentation, communication support, design, analysis, transformation, decision-making, etc. [12].

In EA, the modeling plays the role of a central enabler. Namely, the models typically represent most of what is being reasoned about in EA as a discipline or as an organizational process. In different environments, a number of EA frameworks have been widely adopted (e.g., the Zachman framework [13], DoDAF [14], TOGAF [15], etc.). EA frameworks define their modeling concepts, several of them provide even modeling techniques and own metamodels [16]. TOGAF, one of the most popular and widely adopted EA frameworks [17], provides a process/method for EA work without defining an own modeling technique. Nevertheless, TOGAF is complemented by ArchiMate, both of them are standards of The Open Group, harmonized together to effectively deliver EA [11]. ArchiMate was chosen in this study from a variety of architectural design languages [18], since it was found to be well-adopted, supported by several tools, and well suited for high-level EA modeling [19]. Although UML [20] and AADL [21] are found to surpass ArchiMate in terms of general adoption (i.e., not EA-specific) [18], both appear to be more suited for modeling technical systems and solutions (e.g., software architectures) than higher-level architectures commonly used in EA. Although UML models have been demonstrated to be able to represent a subset of what ArchiMate models can represent, and vice versa, both languages are scoped and optimized for modeling different types of architectures, which makes them rather complementary than alternative in the practice of EA [22]. Hence, this study follows the line of several other studies proposing contributions to EA-modeling, such as modeling business strategy [23], value [24], capabilities [25], or even access control [26].

The ultimate aim of this study is to support the development of enterprises through furthering their abilities to model authorization and access control in enterprise architectures so that the models could be easily aligned with (i) specific purposes of a given modeling task, and (ii) related parts of the enterprise's existing EA models.

This article is an extension of a conference publication [27]. The article is organized as follows. Section 2 summarizes the fundamentals of modeling access control, and presents the common terms, concepts and models of access control used in this study. Section 3 describes the method and design of the study. Section 4 presents the unified metamodel as the result of the study. Section 5 presents several examples of using the unified metamodel to model authorization and

access control. Section 6 discusses the unified metamodel and other findings, compares them with some similar approaches, concludes the study and outlines future work in this line of research.

2 Fundamentals

Models of access control are typically described formally (e.g., [28], [29], [30], [4], [5]). Formal modeling (e.g., using mathematical set theory) enables one to soundly and canonically describe the models' static representation, their operation, and verifying their correctness or other properties. However, this approach renders highly impractical for the use in enterprise-wide contexts, since it forces a high degree of detail in models of specific access control configurations. In effect, the modeler would typically have to model every single entity, the subject to access control, and detailed assignments between such entities, which scales poorly with the size of an enterprise architecture. Additionally, solely set-theoretical notations of access control models provide little visual comprehensibility, a paramount attribute in the practice of EA.

In addition to formal modeling, propositions of several access control models have been complemented by freely conceptual, visual illustrations (e.g., [3], [4], [5], [9]). Although such illustrations show a degree of similarity with EA metamodels, they are, contrary to formal models, lacking an established, unified syntax, and compatibility with modeling languages such as ArchiMate [1] or UML [20]. The challenge occurs when the modeler has to embed models of authorization and access control (e.g., an enterprise authorization policy or a specific configuration of access control) into [existing] models of a specific enterprise. Hence, the two extremes call for a middle way, which would provide compatibility with common EA modeling languages and be visually highly comprehensible, although not necessarily describing the representation and operation of the access control models with formal, mathematical exactitude and rigor. A promising such middle way is conceptual modeling according to a defined, unified language, since it provides immediate visual comprehensibility, compatibility with established modeling languages (e.g., ArchiMate), and at the same time a level of exactitude and consistency that suits formal analysis and machine processing.

Similar approach to that of this study has been adopted by Basin et al. [31], proposing an approach titled "model driven security", building on an extended metamodel of RBAC [3] called SecureUML [32], and providing a semantically well-founded modeling language and code generation process. Somewhat similarly, the works of Gaaloul & Proper [33] and Gaaloul et al. [26] propose an access control model for the use in EA modeling. However, the approaches are based exclusively on RBAC, which makes them inapplicable or impractical for modeling a number of other nowadays commonly used models, as explained later in this section (e.g., RBAC cannot emulate an arbitrary configuration of ABAC and its derivatives; and scales poorly [34] at emulating DAC, especially at representing a DAC-equivalent configuration as an RBAC configuration). Slimani et al. [35] and Muñante et al. [36] propose approaches for modeling access control in a more generic manner, however, both do not express an arbitrary ABAC configuration, not to mention the more recent models. This study treats the most well-known and widely adopted models of access control, as well as some prospectively powerful newcomers, presents conceptual models of these access control models, and proposes a unifying metamodel mapped to ArchiMate. On top of [27], this paper provides more detailed descriptions of the fundamentals of access control and the different models; more detailed account of the methodical aspects and choices that have been made in the study; mapping to ArchiMate; additional validation and illustration of the use of the proposed unified metamodel; and an extended discussion.

The rest of this section introduces the terms specific to access control used throughout the paper, lists and briefly describes the models of access control treated, and presents conceptual models corresponding to those models of access control. Initially, a distinction between an access control model, policy, and mechanism should be made. While a *model of access control* describes an access control system; *a policy* describes a set of access enforcement requirements for the system;

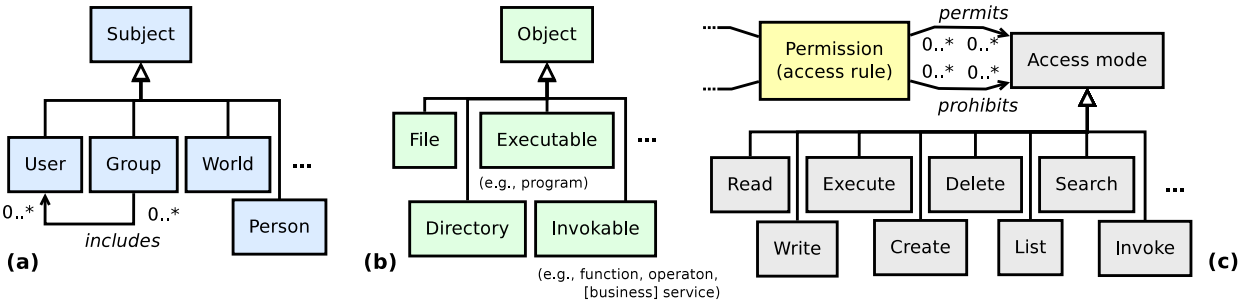

Figure 1. Subject, object, access mode – the most basic terms of access control, exemplified

Table 1. Summary of access control models, policies and mechanisms studied

Name	Character	Property	Domains	Policy	References
Access matrix	Mechanism	*any*	*any*	DAC*	[37]
Access control list (ACL)	Mechanism	*any*	*any*	DAC*	[29] (p. 35)
Protection bits	Mechanism	*any*	OS	DAC*	[34] (p. 14)
Capability ticket	Mechanism	*any*	*any*	DAC*	[2] (p. 134)
Protection ring/domain	Mechanism	*any*	*any*	MAC*	[29]
Lattice model	Model	*any*	*any*	MAC	[29], [38]
Bell LaPadula (BLP)	Model, policy	Conf:ty	*any*	MAC	[28]
Biba	Model, policy	Integrity	*any*	MAC	[29]
Brewer-Nash (Chinese wall)	Model, policy	Integrity	*any*	MAC	[30], [39]
Clark-Wilson	Policy	Integrity	OS	MAC	[40]
Graham-Denning	Model	*any*	*any*	DAC	[41]
Harrison-Ruzzo-Ullman (HRU)	Model	*any*	*any*	DAC	[42]
Role-based access control (RBAC)	Model	*any*	*any*	*any*	[3], [4]
Attribute-based access control (ABAC)	Model	*any*	*any*	*any*	[5], [6], [7], [8]
Usage control model (UCON)	Model	*any*	*any*	*any*	[43]
Risk-adaptive access control (RAdAC)	Model	*any*	*any*	*any*	[9], [10]
Token-based access control (TBAC)	Model	*any*	*any*	*any*	[44]

* *denotes typically, however, not exclusively*; OS *denotes operating systems*

and *a mechanism* describes a part of an implementation of the system. Table 1 summarizes the access control models, policies and mechanisms treated within this study. The vocabulary specific to the access control can be found in Table 2, which is for more clarity complemented in Figure 1. The conceptual models corresponding to the different models and policies of access control, briefly described below, can be found in Figure 2.

2.1 Discretionary Access Control

The family of discretionary access control (DAC, Figure 2a) models is based on the identity of subjects, and on access rules stating what the subject is allowed and/or denied to do. In DAC,

Table 2. Common vocabulary of access control

Term	Description
Subject/ requestor	An entity capable of performing actions in a system under consideration (SUC). For example, a program running on an operating system.
Object/ resource	An entity within a SUC, which has to be protected from unauthorized access. For example, an object can be a document or a system operation.
Access mode	The way, in which a subject can access an object within a SUC. Examples are read, write, execute, delete, create, search, or list contents.
Access rule/ permission/ prohibition/ access right	A rule specifying a mode of access for a subject to an object – either by permitting it (more common), or by prohibiting it. In a yet more generic sense, a single access rule may also specify multiple modes of access for multiple subjects to multiple objects.
User	A user is an entity, external to a SUC, capable of acting within the SUC. A user can be a subject, often having the privilege to further create subjects in a SUC (e.g., run programs). A non-subject user might only be allowed to manipulate subjects, however, not itself access objects directly – as taken from a strictly technical perspective that discerns a human actor from a computer program that actually performs data operations within an IT SUC.
Session	A temporally constrained window of usage, typically authenticated (e.g., by a log-in procedure), in which a user can act within a SUC via subjects – or as a subject itself in case users are treated directly as subjects by the SUC (a rather abstract distinction with regards to the notion of agency, i.e., who consciously performs acts within the SUC).
Classification	A security designation of an object (e.g., a document), which indicates, e.g., the highest secrecy of information contained therein, according to a predefined scheme (e.g., a mathematical lattice defining a partially ordered set of security labels, or simpler, a full order such as in Figure 5b.
Clearance	A security designation of the eligibility of a subject to access object having a certain level of classification, in a certain access mode. Specifics depend on the model of access control under consideration.
Security label	A mark associated with an object or a subject, which carries a specific security meaning. A security label typically denotes a specific classification (of an object) or clearance (of an object).
Attribute	A characteristic of an entity such as a subject (e.g., organizational affiliation or business role), an object (e.g., minimum amount of credits needed for access or classification), or the environment (e.g., time of day, threat level, or other environmental conditions). It can be seen as a function that takes as input an entity (e.g., a subject, object, or the environment) and returns a specific value based on the properties/state of the entity.
Token	An attribute extended through its possible dependence on volatile, dynamic properties or items such as cryptographic tokens (e.g., a Kerberos token), devices (e.g., a smart card), biometric tokens, or risk tokens, which change based on subject behavior and/or other conditions.

subjects can decide over other subjects' permissions (access rules) to objects [2] (hence the *discretion*) – typically only to those objects owned by the subjects themselves.

The most common logical representation of a DAC configuration is an access control matrix [37]. An access control matrix has two dimensions, one containing subjects and the other – objects, while each combination of a subject and an object in the matrix is assigned a set of access rules (i.e., a set of permitted and/or forbidden access modes of the subject over the object). In the more advanced cases (Graham-Denning model [41] and Harrison-Ruzzo-Ullman model [42]), upon a need to regulate subjects' control over other subjects, a subject can be seen as a type of object, so

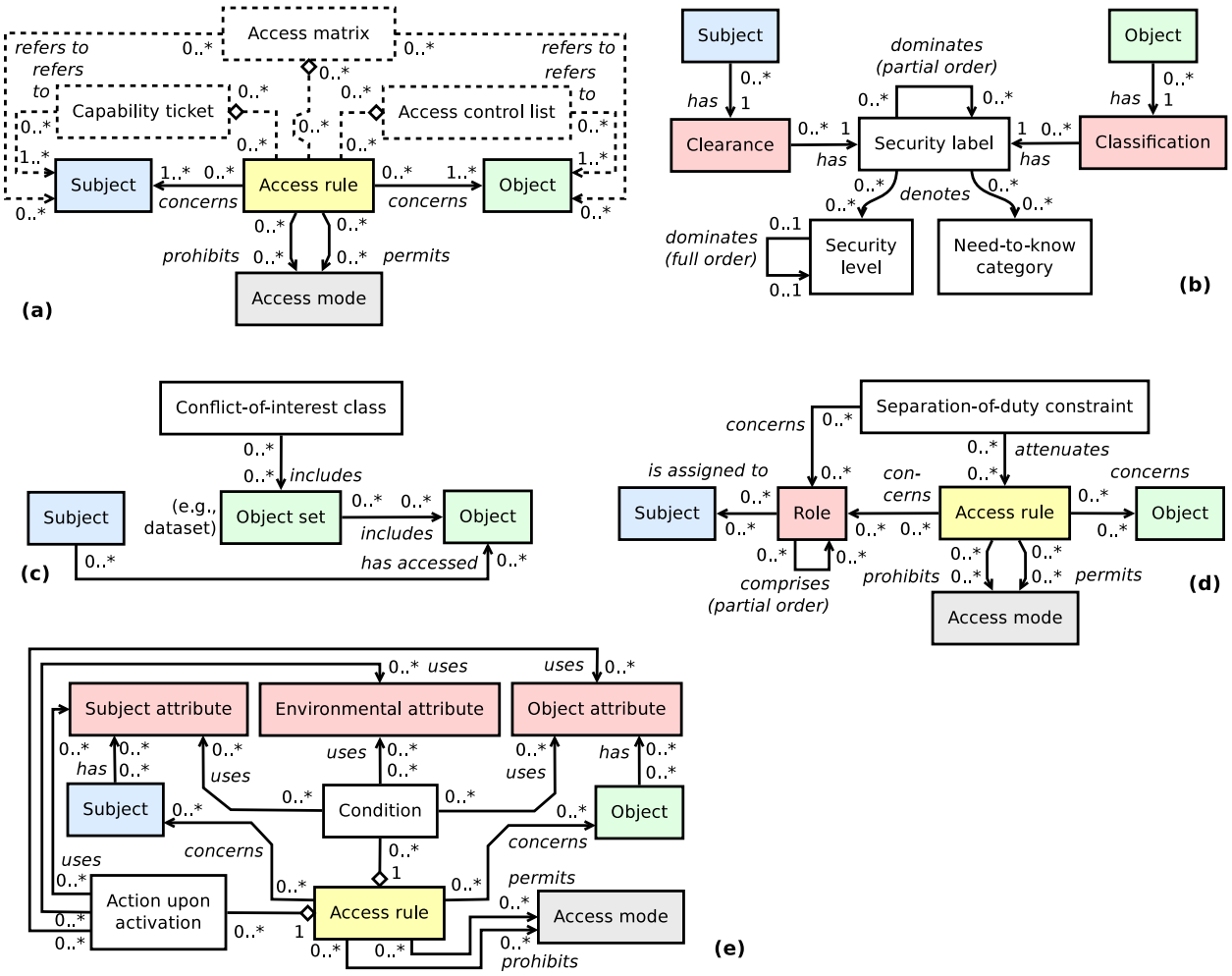

Figure 2. Generic metamodels for expressing configurations of (a) DAC; (b) BLP and Biba; (c) Brewer-Nash (Chinese wall); (d) $\text{RBAC}_{0,1,2,3}$; and (e) ABAC. In (a), the dashed items describe implementation aspects/alternatives.

that the set of objects includes all subjects in addition to other objects in the system. In practice, an access control matrix is typically represented either by multiple access control lists (ACL) (see [29], p. 35) when decomposed by objects (e.g., record for each file listing users and their permissions over the files), or capability tickets when decomposed by subjects (e.g., record for each user listing files and the users' permissions over the files) [2]. Hence, each ACL is bound to a single object (resource in the system) and each capability ticket to a single subject (e.g., a program or a user), while the set of all ACLs across a system can express an authorization configuration equally to a set of capability tickets across the system.

To date, DAC is likely the most prevalent access control model, thanks to its simplicity, the freedom it gives to subjects (and the actual users acting through it), and its extensive legacy. DAC is the default and even the only model used in most operating systems, many databases, and a multitude of web applications. ACL is the most commonly used mechanism in DAC implementations [34]. An example of DAC can be found in a typical Windows or UNIX filesystem.

Finally, the details of Graham-Denning model [41] and Harrison-Ruzzo-Ullman (HRU) [42] model, besides those mentioned previously, are omitted from consideration, since they only have implications on the dynamic aspects of an access control, not the conceptual representation of an access configuration that is of interest to this study.

2.2 Mandatory Access Control

The family of mandatory access control (MAC, Figure 2b) features somewhat greater diversity than that of DAC. Mandatory access control has largely become a synonymous to the term *lattice-based access control* [38], the security levels of which are structured as a lattice. For brevity, only a few well-known models are presented below.

The Bell-LaPadula (BLP) model [28] and Biba model [29] are built on the same basis. Both models use need-to-know categories (e.g., project numbers) for regulating access to objects in a DAC-like fashion, and security labels denoting security levels for classification of objects and clearance of subjects. Both models consider two modes of access – reading and modification. Biba additionally considers invocation (i.e., calling upon another subject) which can consequentially be viewed as modification under the invoked subject's clearance. However, while BLP protects confidentiality, Biba protects integrity. In BLP, reading an object is allowed to a subject if the subject's clearance is equal or higher than the object's classification, and writing is allowed if it is equal or lower. In Biba, reading an object is allowed if the subject's clearance is equal or lower than the object's classification, and writing (and invocation) is allowed if it is equal or higher. Although the difference between BLP and Biba makes the two policies conflicting, they can be combined if security labels denote security levels for confidentiality and integrity separately [38].

The Brewer-Nash model [30] (Chinese wall, Figure 2c) differs in that its configuration changes dynamically according to the history of each subject's access. The model defines a term *conflict-of-interest class*, which groups datasets, or rather *object sets* (e.g., data of different banks), and regulates access as follows: a subject can read an object only if the object is in the same object set as an object already accessed by the subject, or if the object belongs to an entirely different conflict-of-interest class; a subject can write [to] an object only if it also can read the object, and if no such object can be read that is in a different object set from the one for which write access is requested and which at the same time contains unsanitized (i.e., not anonymized) information.

Finally, the Clark-Wilson [40] model has been omitted from further consideration in this study, both because it applies to the level of operating systems, and because its level of detail is far greater than what is typically modeled for the purposes of EA practice (the model specifies e.g., transformation procedures, integrity verification procedures, constrained and unconstrained data items).

2.3 Role-Based Access Control

Role-based access control (RBAC, Figure 2d) [3], [4] is technically a non-discretionary model, in which subjects are granted access based on the roles they take on themselves for a specific session. To exemplify, a subject in form of a human user can take on different roles among those assigned to her/him; e.g., when logging in for a single session, Jane can take on the role of a system administrator, a financial analyst, or a teller.

Several types of RBAC have been identified [4] according to their features. $RBAC_0$ denotes a minimal version, in which a subject can only take on a single role for a session, and there are no constraints for separation of duties. $RBAC_1$ augments $RBAC_0$ with hierarchies of role inclusion (inheritance) in form of a partially ordered set. $RBAC_2$ augments $RBAC_0$ with constraints (e.g., expressing that two specific roles must not be assigned to a subject at the same time – also called static separation of duties). $RBAC_3$ combines $RBAC_1$ and $RBAC_2$, which additionally enables constraining for dynamic separation of duties (e.g., a subject must not take on itself two specific roles within a single session, although it may be possible for the user using two separate sessions).

In a final comment, although RBAC can be configured to emulate DAC and MAC [45], it can be impractical, particularly in the case of DAC [34].

2.4 Attribute-Based Access Control

Attribute-based access control (ABAC, Figure 2e) [5], [3] is one of the more recent models, which, although being the fastest growing one [8] and seemingly on the verge of a large-scale adoption [45], [7], is not yet as widely known as RBAC. To date, although ABAC may be well known in certain circles of academics and practitioners, it is less known in others, which might also benefit from familiarity with it. The major advantages of ABAC over DAC, MAC and RBAC are far greater expressiveness (essentially only limited by the used computational language), richness, greater precision, and flexibility. In fact, ABAC no longer requires specifying individual relationships between subjects and objects [8]. In brief, ABAC builds on *attributes* that can be characteristic to subjects, objects, and the environment (see Table 2). Access rules can specify (e.g., using script-like expressions) under what conditions to provide what type of access. A condition (cf. Figure 2e) means a specification of a given match of attributes upon which access should be granted, or prohibited.

On top of ABAC, UCON [43] presents a feature enrichment. Among others, UCON proposes *mutable attributes* (changeable as a consequence of access in addition to administrative actions), predicates that have to be evaluated prior to a usage decision (*authorizations*), and predicates that verify mandatory requirements for the access (*obligations*).

ABAC has been used as the base for the design of XACML [46], an open standard defining a declarative policy language for access control.

The invention of ABAC has been preceded by numerous extensions to RBAC (e.g., by spatial, temporal, task-, organization- and decision-related aspects). However, this study does not treat them in favor of the more encompassing and generic ABAC.

2.5 Risk-Adaptive and Token-Based Access Control

Recently, risk-adaptive access control (RAdAC [9], [10]) and token-based access control (TBAC [44]) have been proposed.

On top of ABAC, RAdAC considers measures of risk related to access decisions. The measures of risk can be arbitrary, e.g., based on subjects' behavior and trust; specific ways, probabilities, and consequences of misusing objects; or environmental attributes. Hence, the differences from ABAC are mostly of a functional nature, nevertheless major in terms of the increased adaptability and so outlook for greater effectiveness in protecting assets.

TBAC, unlike all other models considered in this study, does not yet appear to have received academic attention. Although its modeling-representational differences from ABAC and RAdAC seem few and minor, the concept of TBAC broadens the perspectives of application and implementation of ABAC and RAdAC in a way highly relevant for EA practice and modeling. Example tokens are listed in Table 3.

3 Method

The study process can be divided into five phases, as outlined in Figure 3. This study has been inspired by the design science research paradigm [47].

The *research contribution* is a design artifact, a unified metamodel for flexibly modeling authorization in enterprises, mapped to ArchiMate. The metamodel can be seen as an extension of a popular EA language, which can represent the access configurations of most access control models used to date, and allows EA practitioners to flexibly model authorization in their enterprises, which further aids development of security policies. According to [48], the artifact falls into the category of frameworks.

The *relevance of the problem* consists in the lack of availability of a modeling language or its part, which would allow EA practitioners to flexibly model authorization realized by the different

Table 3. Examples of TBAC tokens according to [44]

RM OSI-like layer	Example tokens	Control functions
User	One-time password (OTP) token, biometric tokens	User and resource-level access control
Application	Kerberos token, RDF token	User and resource-level access control
Presentation	Data tokens, USB tokens, DOM (document object model) tokens	Coding and conversion controls (XML firewall)
Session	OTP as a session token	Session control via OTP
Transport	EAP-TLS (RFC 2716) token	Flow, port and header control (network firewall)
Network	Packet token, posture token, path token, periphery token	IP address and routing control
Data link	MAC address token, TPM token, L2TP token	Physical media access control

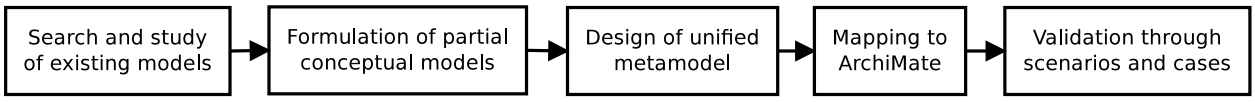

Figure 3. Outline of the study design

access control models used to date, across an entire enterprise, and so enrich existing and new EA models of other kind with the aspects of authorization and access control.

The *process of design evaluation* uses a set of example scenarios and two business cases to test and demonstrate the unified metamodel's qualities.

The *research rigor* is supported by the search for relevant access control models to be considered in the study, the extraction of the conceptual representation of the access control models' configuration (cf. Section 2 and Figure 2), subsequently the process of semantic unification of these partial metamodels into a single unified metamodel, and finally the validation of the unified metamodel on a set of example scenarios and business cases.

The design of the unified metamodel can be seen as a *search* for the following, prioritized from the most important to the least: (1) accommodating all of the features of the partial metamodels within the designed unified metamodel, (2) maximizing the flexibility provided by the unified metamodel to the user, (3) maximizing the ease of modeling using the unified metamodel, (4) providing extensibility of the unified metamodel to aid its adaptation and further development, and (5) maximizing the usefulness of the unified metamodel's mapping to ArchiMate.

This article is intended for both academic audience familiar with EA, who might find use of the artifact or the approach presented in this study in their own research efforts, further develop the artifact, as well as for practitioners who might be facing the needs of modeling authorization and access control in terms of EA.

The first phase consisted in collecting and studying different mechanisms, models, and policies of access control. This study phase employed a combination of electronic search engines such as Google and Google Scholar, and a form of the snowballing method of performing literature review [49]. However, it must be admitted that due to time constraints the process of literature review has not been exhaustive.

An initial set of publications was identified through electronic search, after which mostly backward snowballing, together with further electronic search attempts, were used. Publications sought after were such that proposed or described access control models, mechanisms or policies. Table 1 provides a summary of the different models, mechanisms, and policies identified and

chosen for further consideration. An important criterion for each model to be chosen for further consideration, was its relevance for the study in terms of the combination of (1) its generally known and reported degree of adoption (e.g., DAC, RBAC scored highest); (2) the number of [preferably fresh] online search hits and academic citations related to the model (satisfying score in almost all models except the freshest such as RAdAC and TBAC); as well as (3) the model's reported (e.g., ABAC) or due to yet little academic and other descriptive work rather subjectively judged (e.g., RAdAC, TBAC) potential to increase in enterprise adoption in the near future. It also needs to be mentioned that less known and less adopted models derived from a major one such as RBAC, which have additionally been surpassed by another major model such as ABAC (in terms of the newer major model being able to emulate the less adopted one), have been left out from consideration.

The second phase consisted in formulating a set of concise yet sufficiently expressive conceptual models of the models of access control collected in the previous phase. The models of access control are briefly explained in Section 2, and the corresponding conceptual models are depicted in Figure 2. In order to formulate the conceptual models, a [sub-]design process has been used within the study. The criterion for the design of the conceptual models (also called partial metamodels) of the different access control models has been a correct semantic representation of the configuration of each respective access control model. In other words, each of the partial metamodels has to be able to represent any configuration of the respective access control model, or be trivially extensible to do so. At the same time, each of the partial metamodels must not fail to represent any information in the access configuration, which is necessary for a correct function of the access control model, upon applying the configuration to it in a real setting. The process of evaluation of the correctness of the design of each of the partial metamodels has had the form of a mental evaluation, both by analytic reasoning and by verification against a set of concrete, ad-hoc formulated scenarios of configuration of each respective access control model.

The third phase consisted of designing a unified metamodel, such that it shall be able to express arbitrary configurations of all models and policies taken into consideration in the study, as well as arbitrary combinations of the models and policies. In addition to the expressiveness for purposes of EA modeling, the metamodel has been sought to provide high amount of flexibility to the modeler, mainly to avoid forcing models into a certain given level of detail, while rather letting the modeler choose the degree of detail aligned with the purpose of the modeling task at hand; further it was sought to achieve ease of modeling and extensibility of the metamodel, all evaluated during the design process based on analytical reasoning and ad-hoc formulated scenarios of access configuration.

The fourth phase consisted of identifying points of equivalence or correspondence between elements of the designed unified metamodel and the ArchiMate language [1], for the purpose of mapping the former onto the latter in the form of an extension. During this process, it was sought to maximize the flexibility of the mapping, evaluated based on analytical reasoning and ad-hoc formulated scenarios of relating an authorization model based on the proposed unified metamodel to the ArchiMate model. The flexibility was further evaluated regarding to configurations assumed to commonly occur in enterprises. Subsequently, the mapping was performed and described.

The last, fifth, phase consisted of both illustrating the use of the unified metamodel for the purpose of modeling simple scenarios of access policies (configurations) and validating the sought-after design properties of the unified metamodel to a certain degree, limited by convenience. This phase resulted in five simple example scenarios, modeled according to the unified metamodel, and two business cases: the first related to authorization in an IT landscape of an organization in the public sector; the other related to a widely used passage system. For the first case, a personal interview was conducted. For the other case, authors' experience was used and a few documents were additionally studied. The choice of the example scenarios and business cases was driven mostly by the need to illustrate access policies of the different types of access control models considered in the study, partly by convenience, and partly by the commonality of the access control

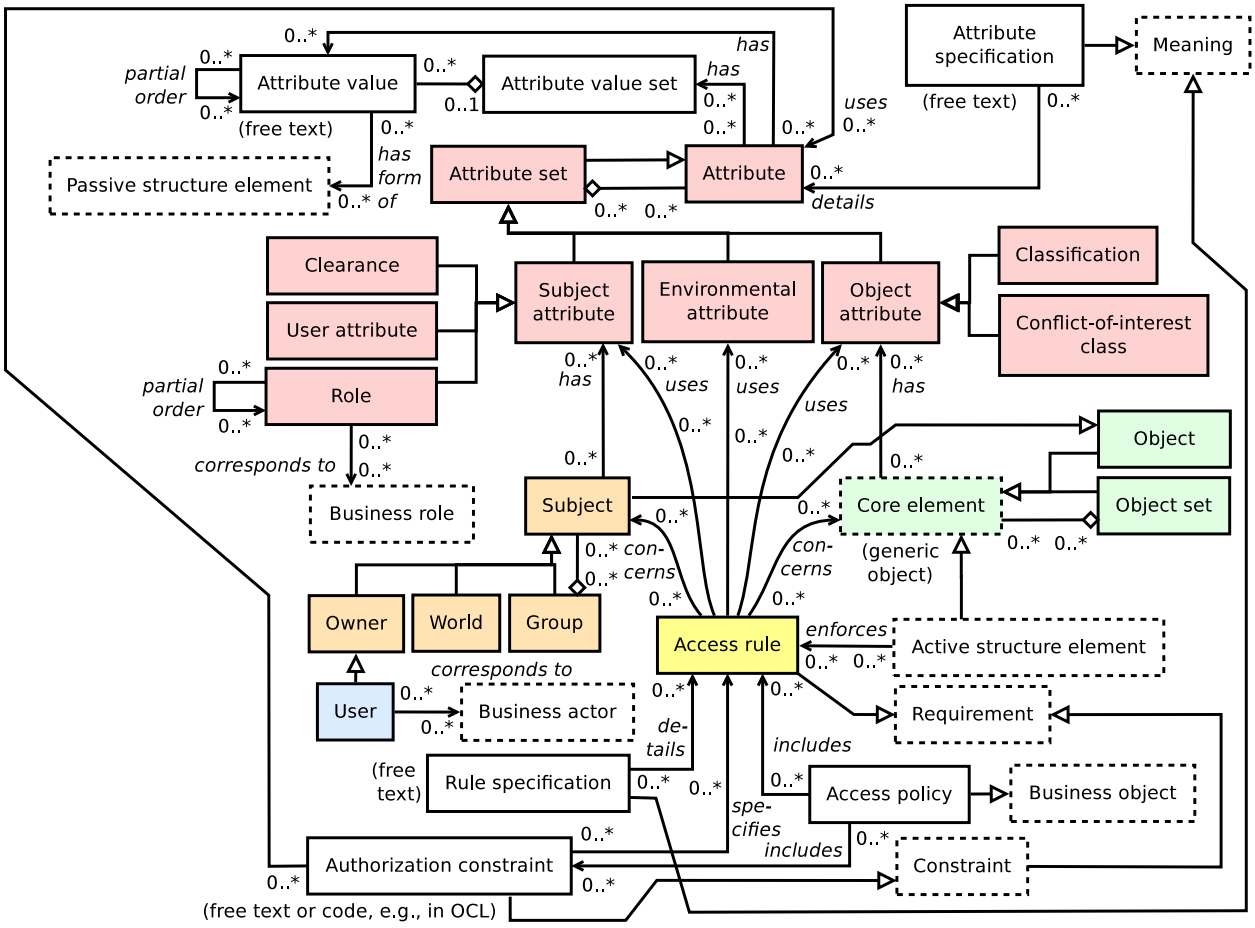

Figure 4. Proposed unified metamodel for modeling authorization. The dashed entities are defined by ArchiMate [1].

models in enterprises, since a comprehensive exemplification and validation of all features of the unified metamodel would require a thorough and lengthy treatment, which fell beyond the limitations of this study. Hence, it has to be admitted that the validation is not complete.

4 Metamodel for Modeling Authorization

This section describes the metamodel and motivates certain features of its design. The unified metamodel for modeling authorization is depicted in Figure 4.

Structurally and syntactically, the unified metamodel mostly builds on the conceptual model of ABAC (cf. Figure 2e), because of ABAC's ability to encompass or emulate most of the other access control models' functions.

For structural simplicity and semantic similarity of the terms, the entity `Attribute` semantically comprises both *attribute* from ABAC and *token* as used in TBAC. Also, items such as *role* or *clearance* can be modeled simply as an *attribute*. For more clarity, however, the unified metamodel retains a number entities, namely, `Role`, `User Attribute`, `Clearance`, `Classification` and `Conflict-of-interest class`, since those are expected to occur commonly. Less generally common entities (e.g., *predetermined explicit authorization* (i.e., decision-based authorization) or *location*) can be instantiated from the closest fitting child class of `Attribute` rather than having a separate class. `Role`, unlike other children of `Attribute`, allows the modeler to define arbitrary partial orders, to capture configurations of $\text{RBAC}_{1,3}$ [4].

Since the name of a modeled attribute might not suffice to capture its full nature and its range of values, the modeler can further specify attributes textually (e.g., by free text or references), using `Attribute specification`. At the same time, the modeler can specify partial orders

(e.g., lattices) of attribute values using Attribute value and group them into sets (e.g., for security levels and need-to-know categories), using Attribute value set. Moreover, attribute values can be linked to instances of ArchiMate's Passive structure element, to denote values that might already be modeled using ArchiMate.

An Object can group arbitrary sets of ArchiMate's Core elements. At the same time, the unified metamodel as a potential extension of ArchiMate is not proposed with the ambition to change the core of ArchiMate in any way, but rather extend its contents in an "append-only" fashion. Hence, the generic entity Object presented in Figure 4 is modeled as a child of ArchiMate's Core element rather than vice versa.

Subject figures as a child of Object, since a subject itself can be an object. Subject, much like Attribute, is also further categorized into the commonly occurring Owner, Group, World (i.e., anyone), and even User denoting an intelligent actor (e.g., human), for the case its distinction from Subject is desirable to a model.

Access rule can connect to a Subject and an Object, although it is not necessary, e.g., in case of ABAC. Access rule can relate to Attributes of any kind, also multiple ones. It can also relate to ArchiMate's Active structure element, e.g., to denote dependency on a system that realizes its enforcement, etc. As with Attribute specification, Rule specification can help further specify an Access rule. Finally, an Access rule might be a part of a specific Access policy.

Various authorization constraints (e.g., cf. $RBAC_2$ [4]) might need to be modeled, using Authorization constraint. Similarly to Access rule, the modeler can also relate an Authorization constraint to an Access policy.

Finally, three patterns occur repeatedly in the design of the proposed unified metamodel. First, a specific form of grouping is used at Attribute, Subject and Object represented by ArchiMate's Core element: The grouping entity (titled a *-set* or *-group*), inherits from its immediate base entity, and aggregates a set of its instances. This allows arbitrary tree-like grouping under the name of the base entity (e.g., Subject). Second, relations of partial order allow the modeler to create arbitrary lattice-like hierarchies. Third, multiplicities of relations are highly permissive, and in most cases allow 0..* rather than the more constraining 0..1 or 1..*, to provide higher flexibility.

The proposed metamodel includes bindings to ArchiMate entities, in Figure 4 distinguished from others using dashed lines. Table 4 lists and motivates the bindings.

There are several entities that do not inherit from an ArchiMate entity. First, Attribute value set, which only serves the [abstract] grouping purpose. Second, Attribute, and its child classes, since they could have different forms corresponding to different ArchiMate entities, implying the need of relying on multiple inheritance, which is avoided on purpose.

It has been mentioned that the proposed metamodel builds extensively on ABAC, as can also be seen comparing it (Figure 4) with the partial metamodel of ABAC in Figure 2e. The core of the proposed metamodel was built on the partial metamodel of ABAC, which was further extended to match the semantic compatibility with RBAC, Bell-LaPadula, Biba, Brewer-Nash, etc. In the process, evaluations have been made as to how to best model conditions translating the match of different attributes into a Boolean value (cf. Figure 2e). In order to avoid introducing excessive complexity of the metamodel, the choice became to let conditions be modeled directly within Authorization constraints. Finally, the metamodel was mapped to ArchiMate, to be able to enrich [existing] ArchiMate models, which introduced a few additional elements and features (e.g., linking to and using Core element as an object).

5 Example Scenarios and Cases

This section presents five simple, example scenarios of the metamodel's usage, and two business cases, which are based on real implementations.

Table 4. Mapping of the proposed unified metamodel to ArchiMate

Relation (entity – entity)	Relation character	Motivation
Any core element used as a generic object	Implied association (twice)	Compatibility with ArchiMate and avoidance of the need to additionally connect instances of `Object` with ArchiMate elements.
Object attrib. – Core element	Association	*(see above)*
Access rule – Core element	Association	*(see above)*
Object – Core element	Specialization	The modeler might need to model an object, which does not have an equivalent in the core of ArchiMate.
Object set – Core element	Specialization	As above, a new child entity of ArchiMate's Core element is added/proposed here, to allow object grouping.
Access rule – Requirement	Specialization	An access rule is seen as a requirement an access control system has to fulfill.
Auth. constr. – Constraint	Specialization	An authorization constraint, as the name implies, is a constraint, and thereby also a requirement.
User – Business actor	Association	In an enterprise context, a human or an intelligent computer agent can be considered a business actor. Moreover, the modeler might need to model multiple user identities for a single business actor, or vice versa.
Role – Business role	Association	Roles in RBAC typically reflect business roles in an organization (e.g., teller or system administrator).
Attrib. spec. – Meaning	Association	Textual descriptions having semantics further dependent on a specific business context.
Attrib. value – Passive struct. element	Association	Attribute value can be a piece of data, an artifact, a business object or a representation of a business object. To avoid relying on multiple inheritance, it associates with their base entity (passive structure element).
Rule spec. – Meaning	Association	Textual descriptions having semantics further dependent on a specific business context.
Access policy – Business object	Association	Normative content, commonly also having form of a physical or electronic document.
Access rule – Active structure element	Association	Access rule is enforced by an active structure document – be it a node or device, an application component or a business actor (e.g., a human being).

5.1 Example DAC Scenario: File System

Description. Let us have a school computer file system, one teacher, and two students. The students, belonging to a group called "Students", are allowed to read contents of the course study directory, and execute a program for exam submission. The teacher, belonging to a group called "Teachers", is allowed to read and write grade records, and read the contents of an exam directory, which stores exams submitted by students. The example configuration is depicted in Figure 5a.

Evaluation and findings. As can be seen in Figure 5a, the example is modeled using a DAC mindset, which in this simple case does not imply a prohibitively large amount of authorization

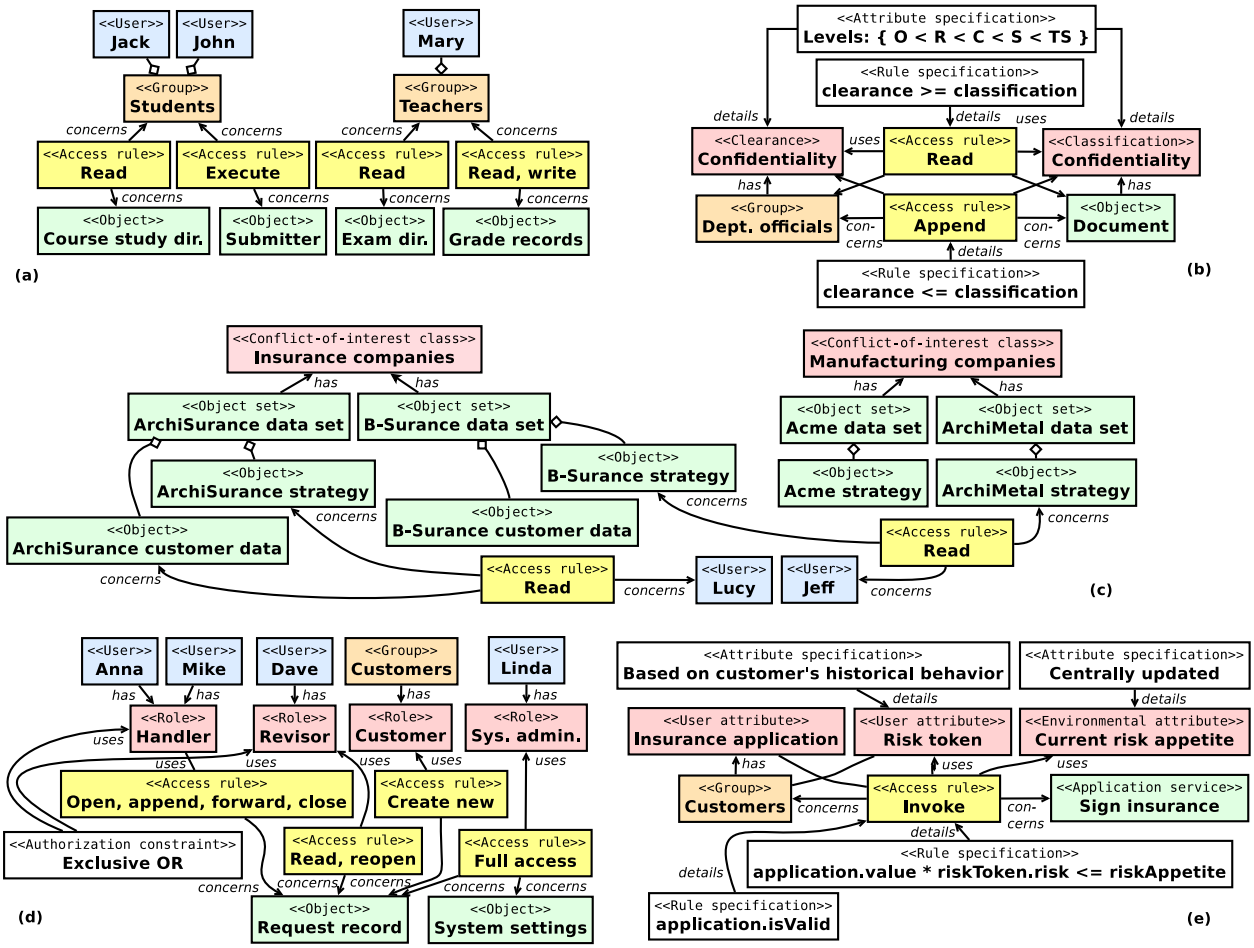

Figure 5. Example scenarios of a (a) DAC; (b) Bell-LaPadula; (c) Brewer-Nash; (d) RBAC; and (e) ABAC/RAdAC/TBAC model configuration. The scenarios are instance models corresponding to the unified metamodel depicted in Figure 4, and each describes a specific access policy.

entities to model, especially `Access rules`. In a more complicated case, the authorization configuration might become too large and unorderly. However, employing an ABAC mindset, the amount of `Access rules` and even other entities could have been decreased by specifying the authorization logic by a single `Access rule` together with a single `Authorization constraint` containing script-like or free-text specification of when what access mode shall be granted or prohibited. From a modeling-technical perspective, doing so would transfer some semantic representation of the access policy from graphical representation (by classes and associations defined in the metamodel) into textual one (a script or free text within an `Authorization constraint`). This is an example of flexibility offered by the unified metamodel; and what is a wiser way to model DAC certainly depends on concrete circumstances faced by the modeler.

5.2 Example MAC Scenario 1: BLP Multilevel Security

Description. Let us have an environment with multilevel security policy according to the Bell-LaPadula model [28]. Let us consider only a single group of users called “Department officials” and their authorizations to read and append to protected documents. The example configuration is depicted in Figure 5b.

Evaluation and findings. The configuration appears to be rather small in size and well-scalable (since for a certain access policy, only objects and subjects might need to be added). However, there is a theoretical possibility that some `Rule specifications` and/or `Attribute`

`specifications` would need to represent complex specifications, and so contain large amounts of text. A quick and easy shortcut to this, although being none of authorization-specific solutions, would be to use references as the names of the entities that uniquely represent documents or other resources that contain the full respective specifications.

5.3 Example MAC Scenario 2: Brewer-Nash (Chinese Wall)

Description. Let us consider a firm providing business consulting services to its customers. Let us have two insurance companies (called ArchiSurance and B-Surance), and two manufacturing companies (called Acme and ArchiMetal). These two types of companies make up a conflict-of-interest class each. For simplicity, let us model one data set for each of the companies, some specific group of data within each, and let us not consider the possibility to anonymize the data. Let us further have two consultants involved in projects with different customers and having read some of the customers' data. Let one consultant (Jeff) be involved in a project for B-Surance, which requires reading documents describing the company's strategy. Let the same consultant have been previously involved in another project dealing with ArchiMetal's strategy. Note that each of the companies belongs to a distinct conflict-of-interest class, thanks to which having access to both companies' datasets is acceptable. Now, among all the insurance and manufacturing companies, the consultant is only allowed to access data belonging to B-Surance and ArchiMetal, the companies already having cooperated with. Let us consider another consultant (Lucy), having read the customer data and strategy of ArchiSurance. Now, this other consultant does not have the freedom to access B-Surance's data set, due to the access to a competitor's [non-anonymized] data set, but still has the freedom to choose whether to cooperate with and hence read the data of Acme or ArchiMetal, both belonging to another conflict-of-interest class. The example configuration is depicted in Figure 5c.

Evaluation and findings. The Brewer-Nash policies described using the proposed unified metamodel do not seem to scale very well with regards to human comprehensibility – for instance, imagine having a multitude of conflict-of-interest classes, a multitude of different object sets, and objects within these. The visual comprehensibility of the whole model would quickly reach its limits. On the other hand, the Brewer-Nash model, perhaps more than others, invites to employing automated analysis within the metamodel, and thus enabling it to warn modelers upon linking a `Subject` with an `Object` (through an `Access rule`) in a way that would violate the Brewer-Nash constraints. Even without such an automated analytical support, modeling a Brewer-Nash policy alone might help holding an overview of the different conflict-of-interest classes and the related object sets. It can also help evaluating whether a concrete user may be allowed to access a specific data set according to the policy.

5.4 Example RBAC Scenario: Request Tracking System

Description. Let us have a request tracking system with two types of objects – system settings and request records, a few users, a group representing customers, and four roles, each having different access: a system administrator, a customer, a request handler, and a revisor. Further, let it be forbidden to combine the roles of request handler and revisor. The example configuration is depicted in Figure 5d.

Evaluation and findings. RBAC appears to be particularly straight-forward and easy to model, allowing concise models (even of advanced RBAC configurations involving a tree structure of roles with static and dynamic constraints), yet not requiring complicated, algorithm-like specifications of access rules. On the other hand, combining many different modes of access into a single `Access rule` (as in Figure 5d) has shown itself as a potential challenge, since the entity names can become large and so less comprehensible and visually pleasing. While breaking such an `Access rule` into multiple ones might not truly pose an improvement, using a system of shorter codes

(e.g., a single letter) for each pre-defined mode of access, might be beneficial. Lastly, what is not apparent from the simple example scenario, is the challenge of having many different objects. In reality, attempt to model one or more access rules for each access-controlled `Object` in a system would have been highly inefficient. As a solution, `Objects` can be grouped into `Object sets`, potentially notably simplifying the entire model of the access policy, given that such groupings can be applied to the model of a given system.

5.5 Example ABAC/RAdAC/TBAC Scenario: Insurance Application System

Description. Let us have an automated processing of insurance applications in an insurance company. Let the company use a risk token, which calculates risk value for each customer based on the customer's history; and let there be a risk appetite setpoint providing a threshold for how risky is the deal the company may sign at a given moment. Let the system register the customer's insurance request as a user attribute automatically upon the customer applying through a web-based form. Finally, let us only allow the system to invoke an insurance signage service if the insurance application is valid, and if the risk that the signed deal would pose to the company does not exceed the risk appetite. The example configuration is depicted in Figure 5e.

Evaluation and findings. Modeling ABAC-like access policies has shown to pose a challenge stemming from the very strength of ABAC: its expressiveness, which by far exceeds that of its predecessors. The challenge is to concisely describe the access rules, which may have the extent of a script or an algorithm, or even an entire computer program. While no immense complexity of the access rules might be reasonable to expect in the general case (although possible with ABAC), the rules can be complicated enough for several lines of free text or code to be needed for an adequate description. A solution to the problem is to only use references (e.g., URI) in the names of `Rule specifications` that can be linked with a document providing a description of each such access rule.

5.6 Business Case 1: Enterprise IT Landscape

This case is constructed based on a part of an IT landscape of an organization in the public sector, and presented in a highly simplified form, allowing the elements essential to authorization and access control to come forth.

Description. The organization administers a large number of user accounts and objects, over a hundred thousand of each. As a peculiarity, although the IT landscape features a central authentication, it does not feature any central authorization, which implies the challenging need to keep the access configuration synchronized, consistent and secure across the many different systems being used on a daily basis. A related challenge is that many of the systems implement their own flavors of access control models, which results in that the configuration is often different from one system to another, despite it being desirable for the configurations to be equal. Another interesting aspect to consider is the regular need to add up to thousands new subjects and hundreds new roles/groups and objects. In other words, the authorization configuration is both large and rather heavily changing – a combination making it impractical and perhaps meaningless to model any close to its full detail for the purposes of EA.

Figure 6 depicts the model of authorization and a few other aspects in Business case 1. In brief, most of the IT usage in the landscape is authenticated using two domain controllers, and an identity management system, which stores the access configuration. The access configuration resembles $RBAC_0$ most, however, as the configuration is synchronized onto systems other than the central identity management system, its parts even become implemented in terms of DAC. The major source of changes in the access configuration are two external systems and one internal system. Machines other than workstations (e.g., enterprise systems) are typically administered through local administrator accounts (not governed by a central access configuration), as can be seen in the model.

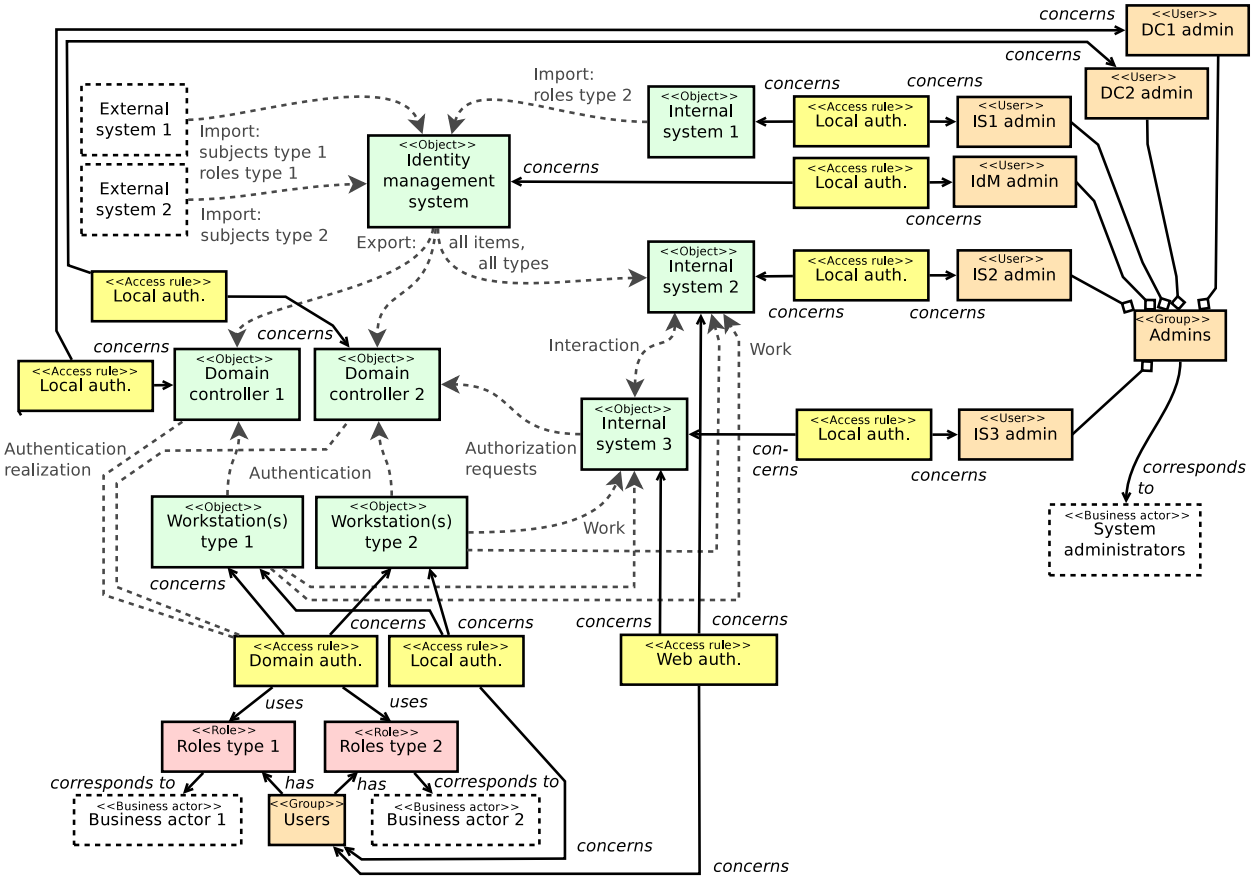

Figure 6. Model of Business case 1: Authorization in an IT landscape (simplified, anonymized). The dashed elements are not modeled according to the unified metamodel proposed in Section 4.

Evaluation and findings. Unlike the example scenarios described previously, the model in Figure 6 does not describe detailed access policies on the level of system resources such as documents or functions. While greater detail of the model would show a more realistic use of modeling authorization in an entire IT landscape, the size of the model would be excessive. Providing an overview of how authorization is realized in an entire IT landscape also useful for supporting communication, analysis, and development of security architecture of an enterprise.

This business case demonstrates the ability to express policy configurations in a highly concise manner – without detailing individual access modes that access rules provide to users. At the same time, the proposed metamodel allows the possibility to detail access rules in a comprehensive and rigorous way, similar to that used in the example scenarios in Figure 5.

A challenge was encountered in modeling the same type of local authorization for a number of separate systems. The need to model elements with a higher amount of repetition might rightfully evoke the feeling of insufficiencies in the design of the modeling approach. There might be at least two solutions leading to more concise modeling of the access policy in this case. The first alternative would be to model an `Object set` that includes all of the administered systems, and subsequently using a single `Access rule` connected to all the different administrative `Users`. It would additionally be advisable to model a `Rule specification` attached to the `Access rule`, detailing which user has access to which system. The second solution would resemble the first, except from the use of the `Object set`, and the need to connect the common `Access rule` to each of the systems directly.

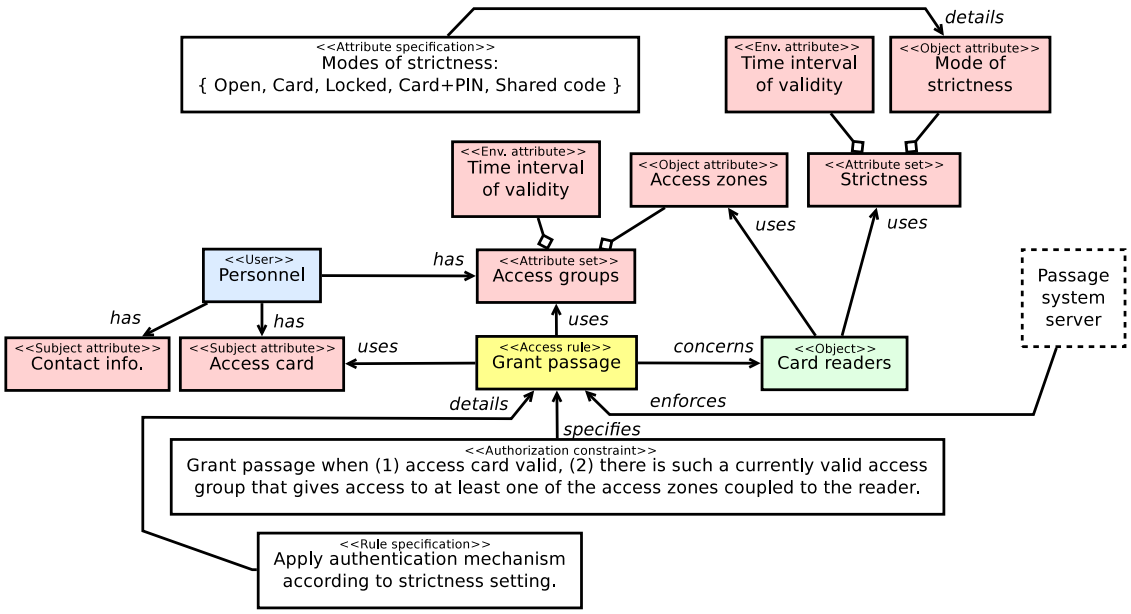

Figure 7. Model of Business case 2: Authorization configuration of a passage system in an office building (simplified, omitting concrete users, access zones and other settings).

5.7 Business Case 2: Passage System

This case is constructed from a set of features and an implementation of a commercial passage system.

Description. The passage system consists of a server with a database, several hundreds of card readers, and a range of other components. In the passage system, access zones can be defined. These access zones can then be coupled to an arbitrary set of card readers. Each user owns an access card, and can be associated with an arbitrary number of access groups, each of which defines the time interval during which it is valid, and the access zones to which it provides access. Lastly, each card reader can operate in different modes of strictness, dependent on the time and configuration (e.g., during daytime, doors are either open or passable by simply striping a valid access card; while during nighttime, both a card swipe and a PIN-code entry are required). Figure 7 depicts the model of authorization in case 2.

Evaluation and findings. The model presented in this case is rather abstract, lacking details about a concrete authorization configuration that shows concrete users, access groups, card readers, etc., in contrast with the modeling approach used in Business case 1 and the illustrative examples depicted in Figure 5.

An alternative would have been to model the authorization configuration of the entire passage system in detail, which would, however, result in a large model with lesser comprehensibility, and the need for more frequent updates. Among other, this case shows the possibility to use the proposed metamodel to describe a concept of authorization, not only its specific configuration. In fact, describing the concept can be combined with describing a detailed configuration in a single model.

In this case, an actual imperfection of the metamodel becomes apparent. Let us consider having defined an `Attribute` such as time interval of validity (see Figure 7) and a set of `Attribute values` related to that attribute. Let us further wish to model a number of concrete `Users` having access, and bind each `User` to its actual value of the validity interval. Currently, the proposed metamodel does not offer a possibility to directly connect an `Attribute value` with a `Subject` or `Object`, which would be useful in a set of possible scenarios. The described imperfection can be resolved by adding an association between the classes `Attribute value` and `Subject`, as well as `Attribute value` and `Object`, both being of type "0..* to 0..*".

A potential downside of such solution would, however, be a possible ambiguity upon reusing the same `Attribute value` by multiple `Attribute`s. In order to eliminate that problem, the cardinalities on the side of `Attribute` would have to be restricted to 0..1 on the associations between `Attribute value` and `Attribute`, as well as `Attribute value set` and `Attribute`.

6 Discussion and Conclusions

To begin with, information obtained during an interview that led to formulating Business case 1 shed some light at the topic of logging and log analysis as a functional part of an access control approach over a longer period of time. All of the models of access control used in this study focus on strictly controlling access according to a predefined policy, or that in combination with other measurements such as the amount of risk related to an act of granting access. However, some enterprise environments practice a more laid back style of managing access. Namely, a more benevolent configuration of access control may be allowed, all the while activity logs are being captured and analyzed, and if an undesirable act of overstepping an adequate amount of access is detected, its originator will be penalized, or the access control configuration will be made stricter. Unfortunately, the concept of logging has not become a part of the unified metamodel, although it might be, thought, worth of further investigations.

Another side issue relates to centrally managed accounts accompanied by locally managed ones, especially local accounts providing administrative privileges to servers including those servers that realize the central management and unification of access control. There might be a system enabling the unification of access control within a domain. However, if there exist separately managed local accounts providing privileged access to resources (as Business case 1 in Section 5 shows), the overall level of security might be impaired. Depending on what level of detail the modeler chooses, such potentially vulnerable patterns of configuration can become more visible to architects and managers, and hence be earlier addressed for improved security.

The proposed metamodel offers the possibility to describe arbitrary configurations of all of the access control models summarized in Table 1, and their combinations. Hence, it can be considered highly comprehensive. However, using a single metamodel that is able to represent configurations of many different models of access control using a single grammar, might be counterintuitive for a modeler who needs to model an access policy according to an older access control model with its specific terminology, such as Bell-LaPadula [28], and who lacks familiarity with the abstractions of newer access control models such as ABAC (for example, the entity `Clearance` or `Classification` is defined as a specialization of `Attribute`, which might possibly be confusing). Fortunately, a modeler can easily be guided to the particular model that is needed for a specific task, to avoid requiring the understanding of the entire metamodel and the underlying access control models prior to using it.

The proposed metamodel offers a high degree of modeling flexibility, as demonstrated in Section 5, which emerges mainly from the presence of four features: (1) broad possibility to group items or present them as groups/sets (e.g., attributes, subjects, objects) with the possibility of introducing further details; (2) the possibility to arbitrarily textually specify attributes, attribute values, access rules, policies, and constraints; (3) the conceptual redundancy provided (e.g., the modeler can model a DAC or a RBAC model both as an ABAC model, or entirely avoiding the use of attributes in the former case while making use of `Role` in the latter case; and (4) the possibility to exploit the permissive cardinality constraints to make abstractions similar to grouping, and so to reduce the number of modeled instances and connections. However, offering high amounts of flexibility through many choices might have a price tag in terms of the lack of enforcing uniformity, from which a lack of consistency and orderliness in instance models (i.e., the different access policies modeled) might result. The level of such consistency and orderliness then, to a large extent, depends on the discipline of the modeler.

The proposed metamodel also offers extensibility, such as the possibility to add new types of attributes (as specializations of the class `Attribute` or any of its child classes). Similarly, further types of `Subject` can be added.

The application of the proposed metamodel to the example scenarios and business cases in Section 5 resulted in finding a deficiency in the metamodel, which is described in that section, together with the solutions for improvement. Another and more general deficiency of the proposed metamodel is the difficulty of enforcing machine-readability of instance models that correspond to the metamodel. This becomes particularly clear in relation to the use of free text in `Rule specifications`, `Attribute specifications`, `Authorization constraints`, `Access rules`, etc. In this particular case, an approach that provides high amount of flexibility in modeling, introduces difficulty for a potential implementation of automated analytical capabilities of the unified metamodel (e.g., automatic detection of policy violations), i.e., machine processing as a part of the metamodel. A solution to the problem would be to use a more machine-friendly representation of the different specifications, e.g., a formal language such as OCL [50]. However, such an enforcement could pose too high requirements on the process of modeling access policies, and so make the unified metamodel less practical to use. This tradeoff might need to be resolved better in the future.

The validity of the study's contribution can be threatened by an eventual omission of an impacting model of access control or a major impacting issue that may exist to date. For example, the issue of boundary-crossing decentralization relevant for cloud computing, has not been extensively treated in this study. Another threat to validity might consist in a design process potentially resulting in a suboptimal artifact (i.e., the unified metamodel), potentially jeopardizing its usefulness to practitioners and academics. Although the generalizability of the metamodel to all existing models of access control is difficult to evaluate, the consideration of well-known and highly generic models of access control such as ABAC, provides an outlook for a high degree of generalizability of the proposal. Similar concern relates to how applicable will the metamodel remain over time, which depends on the innovations taking place within the domain of access control. An interesting aspect to mention is the ability to not only model authorization and access control within IT, but also beyond its boundaries. One of the ideas behind proposing a highly generic metamodel of representing authorization configurations, is to truly be able to model authorization in any relevant domain within an enterprise (e.g., passage and checkpoint control, handling postal packages, etc.).

Although the proposed metamodel of this study shares many conceptual likenesses with the results of Gaaloul & Proper [33], Gaaloul et al. [26], Basin et al. [31], Slimani et al. [35] and Muñante et al. [36], it surpasses these works in terms of its comprehensibility, i.e., the breadth of coverage of the different existing models of access control. Additionally, this study shares much likeness in terms of its ArchiMate mapping compared to that proposed in Gaaloul & Proper [33]. However, the latter is more direct and constraining (e.g., the entity `User` inherits from ArchiMate's `Business actor` and `Role` inherits from ArchiMate's `Business role`, rather than associating with them), which leads to lesser modeling flexibility in comparison to the mapping proposed in this study. Unlike Lodderstedt et al. [32], the ambition with this approach has not been to offer or further develop template-based code generation, since this approach is delimited to the use within EA, acknowledging that software engineering is a different domain of application, the contribution to which would require a new research effort.

In terms of conceptual modeling, this study has summarized a number of relevant models of access control including a few recent ones; presented an ArchiMate-mapped unified metamodel capable of expressing configurations of all the individual models of access control treated; and, finally, provided five illustrative examples of using the metamodel in distinct scenarios, and two additional Business cases based on real setups.

In the future, enriching the unified metamodel with automated analysis is intended, enabling the metamodel to warn about risky patterns of configuration (perhaps easily imaginable for the Brewer-Nash model), or deviations from best practices. Additionally, the metamodel could be used in analyzing attributes related to a given access control implementation and configuration, enterprise needs and maintenance processes (e.g., amount of maintenance demands, cost, modifiability [51], and security through the likelihood of being in a state of misconfiguration), and so help enterprises optimize their architecture.

Acknowledgments

This study has been financed by SweGrids, the *Swedish Centre for Smart Grids and Energy Storage* (www.swegrids.se).

References

[1] The Open Group, "ArchiMate 2.1 Specification, Technical Standard," Van Haren Publishing, Zaltbommel, http://www.opengroup.org/archimate/, 2013.

[2] W. Stallings and L. Brown, Computer Security: Principles and Practice, 2nd ed. Pearson Education, 2012.

[3] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pp. 224–274, 2001. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31540-4_4

[4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," Computer, vol. 29, no. 2, pp. 38–47, 1996. [Online]. Available: http://dx.doi.org/10.1145/501978.501980

[5] X. Jin, R. Krishnan, and R. Sandhu, "A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC," in Data and applications security and privacy XXVI. Springer, 2012, pp. 41–55. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31540-4_4

[6] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," NIST Special Publication, vol. 800, p. 162, 2014. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-162

[7] R. Wagner, "Identity and Access Management 2020," 2014. [Online]. Available: http://www.issa.org/resource/resmgr/JournalPDFs/feature0614.pdf

[8] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-Based Access Control," Computer, vol. 48, no. 2, pp. 85–88, February 2015. [Online]. Available: http://dx.doi.org/10.1109/MC.2015.33

[9] R. W. McGraw, "Risk-Adaptable Access Control (RAdAC)," in Privilege (Access) Management Workshop. NIST–National Institute of Standards and Technology–Information Technology Laboratory, 2009.

[10] R. A. Shaikh, K. Adi, and L. Logrippo, "Dynamic Risk-Based Decision Methods for Access Control Systems," Computers & Security, vol. 31, no. 4, pp. 447–464, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2012.02.006

[11] M. Lankhorst, Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29651-2

[12] R. Lagerström, T. Sommestad, M. Buschle, and M. Ekstedt, "Enterprise Architecture Management's Impact on Information Technology Success," in System Sciences (HICSS), 2011 44th Hawaii International Conference on. IEEE, 2011, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2011.187

[13] J. A. Zachman, "A Framework for Information Systems Architecture," IBM systems journal, vol. 26, no. 3, pp. 276–292, 1987. [Online]. Available: http://dx.doi.org/10.1147/sj.263.0276

[14] Department of Defense Architecture Framework Working Group, "DoD Architecture Framework, Version 1.5," Department of Defense, USA, 2007.

[15] The Open Group, TOGAF Version 9.1. Van Haren Publishing, Zaltbommel, 2011.

[16] U. Franke, D. Höök, J. König, R. Lagerström, P. Närman, J. Ullberg, P. Gustafsson, and M. Ekstedt, "EAF^2 – a Framework for Categorizing Enterprise Architecture Frameworks," in Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD'09. 10th ACIS International Conference on. IEEE, 2009, pp. 327–332. [Online]. Available: http://dx.doi.org/10.1109/snpd.2009.98

[17] R. Sessions, "Comparison of the Top Four Enterprise Architecture Methodologies," 2007. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb466232.aspx

[18] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What Industry Needs from Architectural Languages: A Survey," Software Engineering, IEEE Transactions on, vol. 39, no. 6, pp. 869–891, 2013. [Online]. Available: http://dx.doi.org/10.1109/TSE.2012.74

[19] A. Q. Gill, "Agile Enterprise Architecture Modelling: Evaluating the Applicability and Integration of Six Modelling Standards," Information and Software Technology, vol. 67, pp. 196–206, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2015.07.002

[20] O. M. Group, "Unified Modeling Language (UML)," 2014. [Online]. Available: http://www.omg.org/spec/UML/2.4.1/

[21] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The Architecture Analysis & Design Language (AADL): An Introduction," DTIC Document, Tech. Rep., 2006.

[22] C. Armstrong, J. Baker, I. Band, S. Courtney, H. Jonkers, V. Muchandi, and M. Owen. (2013) Using the ArchiMate® Language with UML®. [Online]. Available: https://www2.opengroup.org/ogsys/catalog/W134

[23] A. Aldea, M.-E. Iacob, J. van Hillegersberg, D. Quartel, L. Bodenstaff, and H. Franken, "Modelling Strategy with ArchiMate," in Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, 2015, pp. 1211–1218. [Online]. Available: http://dx.doi.org/10.1145/2695664.2699489

[24] A. Aldea, M. E. Iacob, J. van Hillegersberg, D. Quartel, and H. Franken, "Modelling Value with ArchiMate," in Advanced Information Systems Engineering Workshops. Springer, 2015, pp. 375–388. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-19243-7_35

[25] C. L. Azevedo, M.-E. Iacob, J. P. A. Almeida, M. van Sinderen, L. F. Pires, and G. Guizzardi, "Modeling Resources and Capabilities in Enterprise Architecture: A Well-Founded Ontology-Based Proposal for Archimate," Information Systems, vol. 54, pp. 235–262, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.is.2015.04.008

[26] K. Gaaloul, S. Guerreiro, and H. A. Proper, "Modeling Access Control Transactions in Enterprise Architecture," in Business Informatics (CBI), 2014 IEEE 16th Conference on, vol. 1. IEEE, 2014, pp. 127–134. [Online]. Available: http://dx.doi.org/10.1109/cbi.2014.26

[27] M. Korman, R. Lagerström, and M. Ekstedt, "Modeling Authorization in Enterprise-Wide Contexts," in Short and Doctoral Consortium Papers Presented at the 8th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling (PoEM 2015), 2015.

[28] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations," DTIC Document, Tech. Rep., 1973.

[29] K. J. Biba, “Integrity Considerations for Secure Computer Systems,” DTIC Document, Tech. Rep., 1977.

[30] D. F. Brewer and M. J. Nash, “The Chinese Wall Security Policy,” in Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium. IEEE, 1989, pp. 206–214. [Online]. Available: http://dx.doi.org/10.1109/secpri.1989.36295

[31] D. Basin, J. Doser, and T. Lodderstedt, “Model Driven Security: From UML Models to Access Control Infrastructures,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 15, no. 1, pp. 39–91, 2006. [Online]. Available: http://dx.doi.org/10.1145/1125808.1125810

[32] T. Lodderstedt, D. Basin, and J. Doser, “SecureUML: A UML-Based Modeling Language for Model-Driven Security,” in UML 2002 - The Unified Modeling Language. Springer, 2002, pp. 426–441.

[33] K. Gaaloul and H. Proper, “An Access Control Model for Organisational Management in Enterprise Architecture,” in Semantics, Knowledge and Grids (SKG), 2013 Ninth International Conference on. IEEE, 2013, pp. 37–43. [Online]. Available: http://dx.doi.org/10.1109/skg.2013.12

[34] V. C. Hu, D. Ferraiolo, and D. R. Kuhn, Assessment of Access Control Systems. US Department of Commerce, National Institute of Standards and Technology, 2006. [Online]. Available: http://dx.doi.org/10.6028/NIST.IR.7316

[35] N. Slimani, H. Khambhammettu, K. Adi, and L. Logrippo, “UACML: Unified Access Control Modeling Language,” in New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on. IEEE, 2011, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/ntms.2011.5721143

[36] D. Munante, L. Gallon, and P. Aniorté, “An Approach Based on Model-Driven Engineering to Define Security Policies Using OrBAC,” in Availability, Reliability and Security (ARES), 2013 Eighth International Conference on. IEEE, 2013, pp. 324–332. [Online]. Available: http://dx.doi.org/10.1109/ares.2013.44

[37] B. W. Lampson, “Protection,” ACM SIGOPS Operating Systems Review, vol. 8, no. 1, pp. 18–24, 1974. [Online]. Available: http://dx.doi.org/10.1145/775265.775268

[38] R. S. Sandhu, “Lattice-Based Access Control Models,” Computer, vol. 26, no. 11, pp. 9–19, 1993. [Online]. Available: http://dx.doi.org/10.1109/2.241422

[39] R. S. Sandhu, “Lattice-Based Enforcement of Chinese Walls,” Computers & Security, vol. 11, no. 8, pp. 753–763, 1992. [Online]. Available: http://dx.doi.org/10.1016/0167-4048(92)90131-A

[40] D. D. Clark and D. R. Wilson, “A Comparison of Commercial and Military Computer Security Policies,” in 2012 IEEE Symposium on Security and Privacy. IEEE Computer Society, 1987, pp. 184–184. [Online]. Available: http://dx.doi.org/10.1109/sp.1987.10001

[41] G. S. Graham and P. J. Denning, “Protection: Principles and Practice,” in Proceedings of the May 16-18, 1972, spring joint computer conference. ACM, 1972, pp. 417–429. [Online]. Available: http://dx.doi.org/10.1145/1478873.1478928

[42] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, “Protection in Operating Systems,” Communications of the ACM, vol. 19, no. 8, pp. 461–471, 1976. [Online]. Available: http://dx.doi.org/10.1145/360303.360333

[43] J. Park and R. Sandhu, “The UCON ABC Usage Control Model,” ACM Transactions on Information and System Security (TISSEC), vol. 7, no. 1, pp. 128–174, 2004. [Online]. Available: http://dx.doi.org/10.1145/984334.984339

[44] R. Radhakrishnan, “The Fifth and Final Frontier of Access Control Model,” 2012. [Online]. Available: http://www.isaca-washdc.org/presentations/2012/201211-session3_article.pdf

[45] R. Sandhu, “The Future of Access Control: Attributes, Automation and Adaptation,” 2013. [Online]. Available: http://profsandhu.com/miscppt/coimbatore_131219.pdf

[46] OASIS, “Extensible Access Control Markup Language (XACML) Version 3.0,” 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

[47] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” MIS Quarterly, vol. 28, no. 1, pp. pp. 75–105, 2004.

[48] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi, “Design Science Research Evaluation,” in Design science research in information systems. Advances in theory and practice. Springer, 2012, pp. 398–410. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29863-9_29

[49] C. Wohlin, “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering,” in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2014, p. 38. [Online]. Available: http://dx.doi.org/10.1145/2601248.2601268

[50] O. M. Group, “Object Constraint Language (OCL),” 2014. [Online]. Available: http://www.omg.org/spec/OCL/2.4/

[51] R. Lagerström, P. Johnson, and D. Höök, “Architecture Analysis of Enterprise Systems Modifiability–Models, Analysis, and Validation,” Journal of Systems and Software, vol. 83, no. 8, pp. 1387–1403, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2010.02.019