Enterprise Models as Drivers for IT Security Management at Runtime

Anat Goldstein, Sietse Overbeek

Institute for Computer Science and Business Information Systems,
University of Duisburg-Essen, Germany

{Anat.Goldstein, Sietse.Overbeek}@uni-due.de

Abstract. This paper describes how enterprise models can be made suitable for monitoring and controlling IT security at runtime. A holistic modeling method is proposed that extends enterprise models with runtime information, turning them into dashboards for managing security incidents and risks, and supporting decision making at runtime. The requirements of such a modeling method are defined and an existing enterprise modeling language is extended with relevant security concepts that also capture runtime information to satisfy these requirements. Subsequently, the resulting modeling method is evaluated against the previously defined requirements. It is also shown that common metamodeling frameworks are not suitable for implementing a modeling environment that results in suitable IT security dashboards. This leads to suggesting implementation of the modeling environment using the eXecutable Modeling Facility.

1 Introduction

In recent years, Information Technology (IT) security has become a major topic as well as a substantial challenge. On the one hand, there is an ever increasing need to protect IT resources as enterprise assets and business processes increasingly depend on IT. On the other hand, enterprises become more exposed to IT security threats due to an increase in Internet connectivity and availability of sophisticated malware. Designing and managing secure IT systems involves technical complexities that are caused, for example, by prevalent use of distributed computing and by frequent technological changes. It requires a deep understanding of the organizational structure and its operations, which are also subject to frequent changes with the continuous effort of the enterprise to stay competitive. Designing secure IT systems also requires the participation of various stakeholders, such as: IT professionals, security experts and business managers. These stakeholders do not share a common language and possess different views of the problem. Also, as IT security solutions are required to be economically justified, a cost-benefit analysis of possible solutions is called for. These challenges stress the need for methods and tools for supporting IT management with designing, realizing and managing IT security systems. Such methods and tools should account for technical, economical, business and managerial aspects [1, 2].

Conceptual models are often used for reducing complexities and bridging communication gaps. In recent years, an extensive amount of research is oriented towards development of methods for modelling IT security. In particular, there are several methods that bridge the gap between business and technical perspectives, for example, by extending business process (BP) models (e.g., [3-5]) or use case diagrams (e.g., [6, 7]) with security concepts. These methods use models for security requirements analysis and design, for risk identification and some even support further code generation of security policies based on requirements defined in the model. However, so far there are almost no methods that use models for monitoring and controlling IT security at runtime, e.g., for monitoring security incidents, for analyzing their effect on BPs and on IT resources, and for quickly selecting an appropriate response. In addition, most of the existing modeling methods do not provide holistic support for IT security. Instead, they are focused on particular perspectives of IT security.

As part of our ongoing research, a modeling method is developed that provides holistic support for the analysis and design of secure IT systems and for managing IT security. It is holistic in two ways. Firstly, it covers and integrates three different enterprise perspectives: Organizational, strategic and technological. Secondly, it covers several tasks – the analysis, design, implementation and management of IT security. In this paper, however, the focus is on the management of IT security only. In particular, the developed modeling method extends an enterprise modeling environment so that created enterprise models (EM) can be used as dashboards, allowing managers to monitor and control the security of enterprise systems, that is, systems that support the managing of BP executions, employees, concrete IT resources and so forth. Security dashboards can be used to analyze monthly costs of various security controls, summarize attack information on different IT resources and identify business process instances that might be affected by an occurrence of a security incident. In other words, in this paper we propose using EMs as drivers for IT security management at runtime.

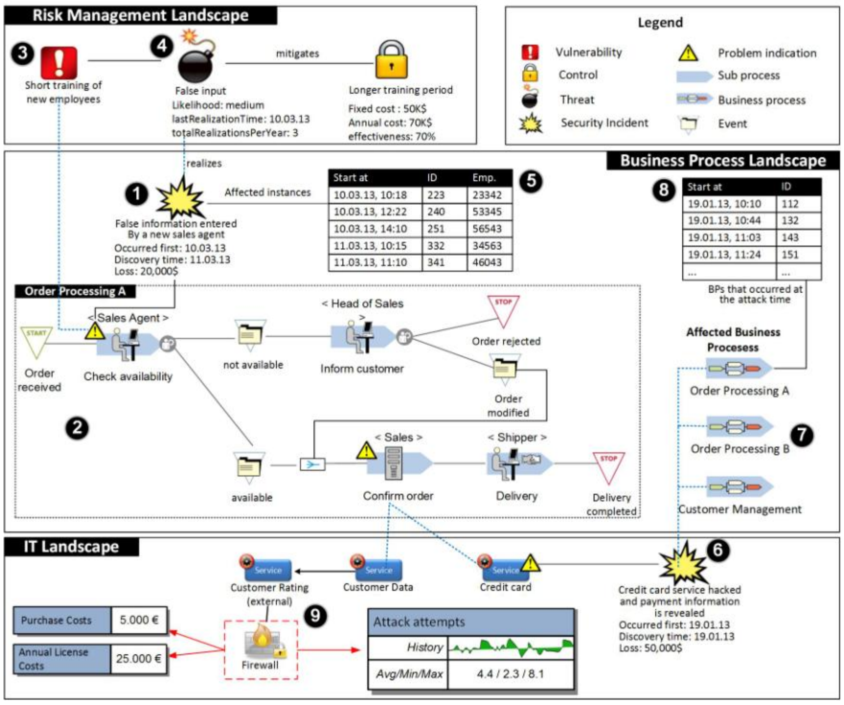
Our approach is based on and extends a multi-perspective enterprise modelling (MEMO) framework [8]. MEMO provides a number of domain specific modelling languages (DSML) to describe different aspects of the enterprise, for example, business processes, the organizational structure, IT resources and strategic goals. Extending MEMO with IT security concepts allows for creating EMs that can be used to describe IT security from various perspectives.

The rest of this paper is organized as follows: In section 2, use scenarios for managing IT at runtime are provided. In section 3, IT security models at runtime are defined and in section 4 we define the requirements for supporting such models at runtime. In section 5, we describe the development of a modeling method for creating IT security models at runtime and in section 6 we evaluate the method. The paper ends with conclusions and a discussion of future work.

2 Use Scenarios for Managing IT Security at Runtime

Our modeling method is based on the Multi-perspective Enterprise Modelling (MEMO) approach [8]. MEMO includes a high-level conceptual framework that represents a “ball park view” of an enterprise [8]. It includes three *generic perspectives*, namely: Strategy, organization and information systems. Each of these perspectives can be further detailed into various aspects, e.g., resource, structure, process and goal. Each perspective is supported by a domain-specific modeling language (DSML), which provides specific concepts and abstractions for describing these aspects. For example, OrgML is a DSML for modeling the structure of the organization [9] and its BPs [10], and ITML is a DSML for modeling IT resources in use. The semantics and abstract syntax of all MEMO-DSMLs are specified using a meta modeling language (MML). Sharing the same metamodel supports the integration of the DSMLs and thereby the integration of the different perspectives. For example, it is possible to define that a BP activity is performed by a particular organizational unit and requires the use of a certain IT resource. Therefore, MEMO provides a foundation for the modeling of IT security in organizations, as it allows extending perspectives with security concepts and integrating them to allow for holistic modeling of IT security.

Fig. 1 illustrates two scenarios in which EMs are used for managing security at runtime. It shows three landscapes: A risk management landscape, a BP landscape and an IT landscape. The BP landscape shows three BP types: *Order processing A*, *Order processing B* and *Customer management*. The detailed structure of the BP type *Order processing A* is shown as well. The figure shows two scenarios in which security threats are realized by actual security incidents.



Start at	ID	Emp.
10.03.13, 10:18	223	23342
10.03.13, 12:22	240	53345
10.03.13, 14:10	251	56543
11.03.13, 10:15	332	34563
11.03.13, 11:10	341	46043

Start at	ID
19.01.13, 10:10	112
19.01.13, 10:44	132
19.01.13, 11:03	143
19.01.13, 11:24	151
...	...

Purchase Costs	5.000 €
Annual License Costs	25.000 €

Attack attempts	
History	
Avg/Min/Max	4.4 / 2.3 / 8.1

Fig. 1. Use scenarios for managing IT security.

In the first scenario, false information was entered on 10 March 2013 by a new sales agent (❶) in the *Check availability* sub process (❷), as a result of a too short training period for the new agent. *Short training period of new employees* has been identified as a *vulnerability* in the risk management landscape (❸), that can be exploited by the *False input threat* (❹). Thus, the first security incident in fact realizes a threat that has been modeled in the risk management landscape. It is then possible to determine which instances of this process were affected (❺) and, therefore, contain incorrect information. Subsequently, these instances can be analyzed and incorrect information can be repaired.

The second scenario shows that a security incident occurred on 19 January 2013 that is related to a Credit card service (❻). As a result, payment information has been revealed. The credit card service is an IT resource and part of the IT landscape. This service relates to the three BP types shown in the BP landscape (❼), which implies that it is possible to know which instances of these BP types are affected by this incident. For example, it can be seen that at least four instances of the *Order processing A* type were affected on 19 January 2013 (❽). A possible way to mitigate the risk that a service is hacked is to install a firewall (❾), such as is the case with the *Customer rating* service. The firewall can be supplemented with statistics on its attacks and related costs in order to support management of risk mitigation.

Having explained the two scenarios, IT security models at runtime can be defined.

3 IT Security at Runtime

Usually, enterprise models refer to what is known as the *type level* [11]. This means that they describe types or classes of BPs (e.g., processing of an order, procurement of supplies, etc.), of IT resources (e.g., Oracle DB server, customer data Web service, ERP system), and of organizational entities (e.g., sales agent position and a QA officer role), rather than particular instances (e.g., a processing of an order that occurred at a certain time, an instance of an Oracle DB server, or a certain employee that fills a certain position). This is illustrated in Fig. 2. The upper part of the figure contains an excerpt of a BP type called *order processing*, an IT service type called *credit card* that is used in the *confirm order* step of the BP and a threat type *credit card data theft*, which is associated with the *credit card* service.

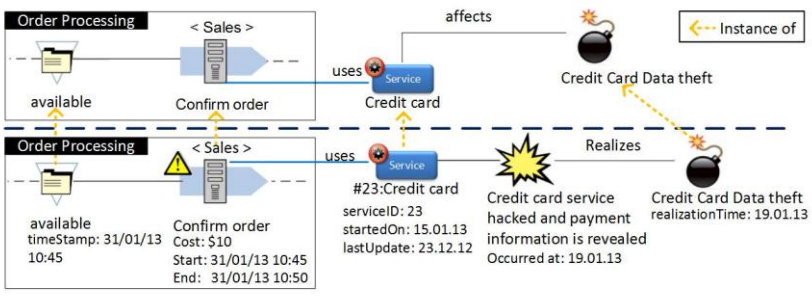


Fig. 2. Type level and instance level.

These are all examples of *types* as they abstract from instance details such as the exact start time and end time of a business process step. However, sometimes it is useful to represent actual instances of the running system, for example, for monitoring and controlling the running system. The lower part of Fig. 2 presents a runtime execution of the *order processing* BP that occurred at a certain time. It also presents an instance of the *credit card* service that is used by the BP instance and a security incident which affects it. The security incident realizes the instance of the threat from the upper part of Fig. 2. Models that represent the type level are designated as type models and models that represent instances are designated as instance models.

When it comes to modeling frameworks, e.g., UML [12] and MEMO [8], it is common to distinguish between four levels of abstraction. In addition to the instance level (referred to as M_0) and the type level (M_1), there is a metamodel level (M_2) that captures the modelling language for creating type models, for example, the UML or in our case MEMO-DSMLs. On top of that, the meta meta model level (M_3) is used to define properties of all metamodels [13]. For example, the UML is defined by the Meta Object Facility (MOF) and all MEMO-DSMLs are defined using MML. In the rest of the paper it is shown how this language hierarchy is used and extended to support the integration of types and instances and to foster runtime models of IT security.

Our intention to support real-time monitoring and analysis of security incidents demands for integrating enterprise models with runtime information of the enterprise software systems that are used to manage business process control flows, IT resources and employees. Such kind of integration is known as ‘modelling at runtime’. A ‘model@runtime’ is a conceptual model that is a “causally connected self-representation of a system” [14]. In order for a model to be causally connected with a system it needs to always represent the correct state of the system. In addition, changes to the model should result in correct system changes [15]. By abstracting from the runtime properties of M_0 instances, models@runtime promise rendering runtime behaviour more understandably for different stakeholders and support analyzing the system’s current state [16]. Following this, we aim at extending MEMO-DSMLs with IT security concepts that are supplemented with capabilities of runtime models. Equipped with these capabilities, EMs can be used for capturing information within the EM that is aggregated from the instance level, e.g., the average time to complete an activity or the number of realizations of a threat. They can also be used for visualizing concrete instances of the M_0 level and for navigating between model types and representations of their instances. In this way, the extended EM can support management of IT security.

4 Requirements of the Modeling Method

The scenarios presented in section 2 illustrate that in order to use EMs for managing IT security at runtime, the following requirements should be satisfied:

Req1: In order to comprehensively model IT security, various perspectives of the enterprise should be considered. The EM should integrate IT security concepts that are relevant for various enterprise perspectives (e.g., BPs, IT resource and organizational units). This point is stressed in [1] and it is dubbed as *horizontal integration*.

Req2: The EM should integrate concepts that not only represent abstractions of the type level such as types of BPs. It should also represent abstractions of the runtime (instance) level, such as executed BP instances. This is dubbed as *vertical integration*. More specifically, the DSMLs that are used for creating enterprise models should include both type-level abstractions such as event name or sub process type and instance-level abstractions such as an event time stamp.

Based on the definitions in section 3, a further requirement can be specified:

Req3: Type models should be aware of their instances and able to interact with them. This also implies a need for synchronization of the different models, so that in case M_0 instances change it is automatically reflected in the runtime models.

The vertical integration of types and instances could be taken one step forward, by using type level entities for the actual creation of their M_0 instance, which are represented in the enterprise software systems. This would foster reuse and facilitate conformance of M_0 instances to their type models. Thus, the next requirement is defined:

Req4: Type level entities should be used for defining their corresponding M_0 instances. Nevertheless, using type models for the creation of M_0 objects has been identified as a challenging task [17], as discussed in the following section.

5 Developing a Method for IT Security Models@Runtime

In this section, the development of a modeling method that addresses the aforementioned requirements is described.

5.1 Extending MEMO to Support IT Security Models@Runtime

The first step is to extend MEMO-OrgML and MEMO-ITML with meta concepts that support the scenarios as illustrated in section 2. It should be noted that a holistic modeling approach for IT security should include more concepts than those that are relevant for the use scenarios (c.f. [1]). However, this is sufficient to reach our objective of illustrating how an EM language is extended in order to create EMs that serve as dashboards for IT security management.

An IT security incident is an event that might have a negative effect on the organization. The incident exploits one or more vulnerabilities of the organization, e.g., a weakness of an IT resource or of a BP, and it has a set of impacts. An IT security incident can realize a threat - a potential event, situation or action that might cause harm to the organization by exploiting its vulnerabilities. Threats and vulnerabilities are usually identified during security risk analysis. A threat is created by a threat-source – an external force to the security system that has potential or intention to cause harm. When the threat-source comes from within the organization, it can be associated with an organizational unit, e.g., employee, position or role. The above concepts represent both type and instance level properties. For example, a threat type is modeled in advance (usually during security risk analysis) and belongs to the type level. Then, at runtime, a threat can be realized by actual security incidents at a certain times.

In order to use the meta concepts not only for modeling types but also for representing runtime instances, we enhance the meta concepts with abstractions of runtime properties such as the realization date of a threat or the name of a threat source. Adding runtime concepts to the metamodel makes sense because otherwise it would be the responsibility of the modeler to account for them in every enterprise model that is created [18]. This would reduce the reusability of the resulting models [8]. To support this requirement, the MEMO notion of an intrinsic feature [8] is used. This notion allows defining an entity, attribute or association that is only relevant on the instance level as ‘intrinsic’. Intrinsic features cannot be instantiated at the type level, but only on the M_0 level. Thus, a security incident is defined as an intrinsic entity. The attribute *realizationTime* of a Threat is defined as an intrinsic attribute.
In order to support online decision making, type concepts should also include attributes that are calculated based on instance values. For example, we can calculate a threat average realization cost based on the concrete costs of its instances or we can calculate the number of threat realizations per year. Such attributes are called *derivable* attributes. Fig. 3 describes a simplified OrgML meta model which is extended with the above security concepts. Intrinsic features are marked with a boxed letter ‘i’ and derivable features are marked with a boxed letter ‘d’). Due to space limitations, Fig. 3 does not depict ITML concepts, except for a general concept ‘IT Resource’ which is a surrogate for any IT resource type.

5.2 Developing a Corresponding Modeling Tool

So far, extensions to MEMO-DSMLs have been described in order to capture IT security concepts in general and runtime properties in particular. By doing so, *Req1* and *Req2* are addressed. In order to address requirements *Req3* and *Req4,* a corresponding modeling environment should be developed. The IT-security related concepts that we specified have in part been implemented within the existing meta modeling environment MEMO Center, which is based on the Eclipse Modeling Framework (EMF) [19]. For this purpose, the respective meta models had to be extended first. Subsequently, new model editors were generated based on the extended metamodels. Unfortunately, the EMF-based MEMO Center was not suitable for satisfying *Req3* and *Req4*. As with most metamodeling facilities, the EMF is limited by the semantics of its implementation language, in this case Java, which supports only two levels of abstraction: Class and Object.

Type models that are created within the newly generated modeling editors are represented by Java objects that can be dynamically changed by the user. For example, a BP event type is represented by an object that is created and modified by the modeler to define the event name and its other properties. Being Java objects, they could not be further instantiated for creating M_0 instances. Instead, these objects can be used for generating new classes that represent M_0 instances and corresponding editors (using the same technique that was used for generating classes of the type-level modeling editors). This results in two independent sets of classes / editors, one for representing types and one for representing instances. This is problematic with respect to our intention to integrate between the type level and the instance level. Firstly, it is difficult to synchronize the evolution of the type level with the instance level: Whenever a type model changes (e.g., a BP model is updated, which is likely to happen a lot) all the classes representing corresponding instances should be regenerated. Secondly, when M_0 elements are created or changed, e.g., when a new security incident occurs or when a BP step is completed, corresponding changes should be immediately reflected in the relevant type model editors. For example, when a certain threat is realized, the value of *totalRealizationsPerYear* of that particular threat type should be increased. This requires a support for a synchronization mechanism between the two editors.

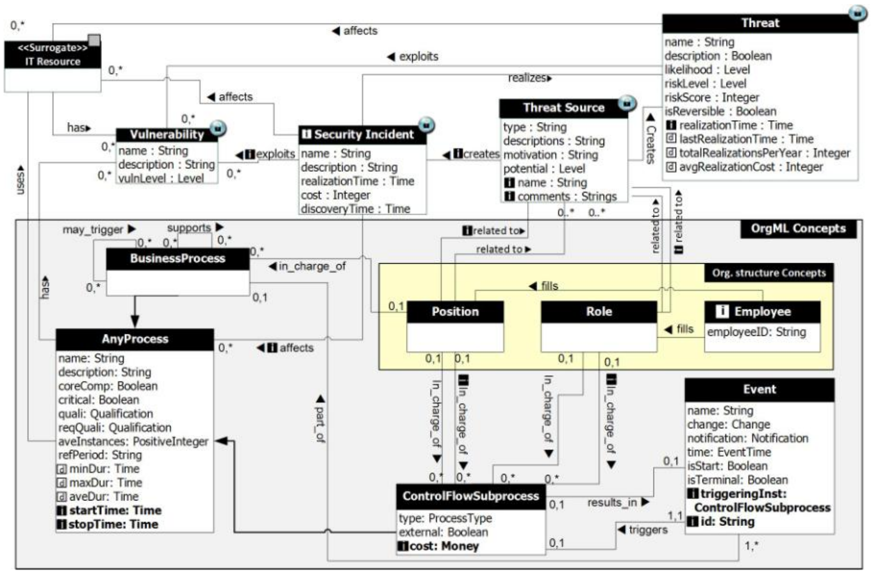


Fig. 3. An excerpt of the OrgML metamodel that illustrates the use of intrinsic features.

Because of these reasons, we started to re-implement the meta modeling environment using eXecutable Modeling Facility (XMF). XMF is a programming language that is accompanied by a modeling tool, the Xmodeler [13]. Both are implemented within the Eclipse framework. Its syntax has some similarities to Smalltalk and Lisp. XMF allows accessing and modifying its own specification and its runtime system. Furthermore, it includes tools for building compilers for further languages, which makes it possible to execute code of different programming languages in one runtime system. Therefore, XMF can be seen as a meta programming language. Implementing the MEMO modeling environment using XMF results in an environment that does not only include standard features of modeling tools, such as enforcing language syntax and semantics and creation of corresponding modeling editors. Furthermore, it features a common representation of models and code, which enables both a tight integration of models on different classification layers and also using (meta) models at runtime. In particular, a meta type defined on M_2 can be instantiated into type entities on M_1 that in turn can be instantiated into M_0 objects. Entities at each level are able to interact with all their instances and vice-versa. These features facilitate the development of IT security dashboards that integrate and synchronize the M_0 and M_1 levels. A rudimentary implementation of MEMO Center with XMF, which demonstrates its capabilities for creating runtime models, is presented in [20].

6 Evaluation

An approach has been proposed to extend an EM environment in order to utilize EMs as dashboards for managing IT security. In section 4, we have defined four requirements that the targeted modeling method should satisfy. The presented modeling method is measured against these requirements. The first requirement, which concerns the integration of IT security concepts that are relevant for various enterprise perspectives, has been partially addressed. As the focus of this paper is on a modeling approach that allows EMs to serve as IT security dashboards, only a limited set of concepts that are mainly related to risk analysis have been presented. However, horizontal integration is facilitated as MEMO is at the basis of our method. It provides an infrastructure for adding IT security concepts that are relevant for the various enterprise perspectives. The second requirement, which concerns including abstractions of the type and instance levels, is supported by our method by using intrinsic features to describe runtime properties and derivable features that aggregate runtime information of instances on M_0. Although not discussed in detail, using XMF for the implementation of MEMO allows for satisfying the third and fourth requirements. First of all, by inheriting from the core concepts of XMF, the MEMO-DSMLs can be used to specify type-model concepts that can be instantiated into M_0 concepts without requiring generation of code. This satisfies the fourth requirement. Secondly, M_1 concepts created with the MEMO-DSMLs have the ability to access their instances, which serves as a foundation for integrating and synchronizing between different levels of abstraction at runtime, as defined by the third requirement.

7 Conclusions and Future Research

In this paper, we present a modeling method that extends EMs to serve as dashboards for IT security management. Such dashboards can support real-time management of IT security incidents and risks, which allows for analyzing their effect at runtime and enable managers to respond quickly and efficiently. While we have focused on the IT security domain, the presented approach could be applied to other domains as well.

It has been explained why common metamodeling facilities such as the EMF are not sufficient for extending EMs to serve as dashboards, resulting in a re-implementation of MEMO in XMF. In the future, we intend to continue with the development of the modeling environment in XMF and to extend it with additional IT security concepts. We also intend to supplement the modeling method with corresponding process models that would guide the use of such IT security models@runtime in various scenarios. So far it is not discussed how M_0 instances are created. It is assumed that somehow they are created by instantiating type-entities on M_1. In future research, the integration of EMs with the actual enterprise system should be supported. One way to do that is by using EMs for realizing the enterprise software systems, so that EMs are integrated with them and are able to monitor them. This kind of enterprise software system is known as a self-referential system [17]. This is our ultimate future goal. A less demanding approach is to use information that is collected by enterprise systems, e.g., by collecting information of BP executions, of concrete IT resources and of system intrusions through dedicated interfaces. This information can be used to create corresponding instances within the EM environment. Yet, in this approach synchronizing between the EM environment and the model that is represented in the enterprise system remains problematic.

References

1. Goldstein, A., Frank, U.: A Language for Multi-Perspective Modelling of IT Security: Objectives and Analysis of Requirements. In: BPM International Workshops, Tallinn, Estonia. Revised papers, 132, pp. 636–648. Springer, Berlin (2012)
2. von Solms, B.: Information Security – A Multidimensional Discipline. Computers & Security 20, 504–508 (2001)
3. Rodriguez, A., Fernandez-Medina, E., Piattini, M.: Security requirement with a UML 2.0 profile. In: ARES 2006. IEEE Computer Society (2006)
4. Hafner, M., Breu, R., Agreiter, B., Nowak, A.: SECTET: an extensible framework for the realization of secure inter-organizational workflows. Internet Research 16, 491–506 (2006)
5. Wolter, C., Menzel, M., Meinel, C.: Modelling Security Goals in Business Processes. In: Kühne, T. (ed.) Modellierung 2008. 12. - 14. März 2008 Berlin, pp. 197–212.(2008)
6. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Eng 10, 34–44 (2005)
7. McDermott, J.P., Fox, C.: Using Abuse Case Models for Security Requirements Analysis. In: 15th ACSAC, pp. 55–64. IEEE Computer Society (1999)
8. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. Softw Syst Model (2013)
9. Frank, U.: MEMO OrgML (1): Focus on Organizational Structure. ICB-Report 48 (2011)
10. Frank, U.: MEMO OrgML (2): Focus on Business Processes. ICB-Report 49 (2011)
11. Kühne, T.: Matters of (Meta-) Modeling. Softw Syst Model 5, 369–385 (2006)
12. OMG: OMG Unified Modeling Language Infrastructure, http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/ (2011)
13. Clark, T., Sammut, P., Willans, J.: Applied metamodelling: a foundation for language driven development (2008)
14. Blair, G., Bencomo, N., France, R.B.: Models@ run.time. Computer 42, 22–27 (2009)
15. Song, H., Huang, G., Chauvel, F., Xiong, Y., Hu, Z., Sun, Y., Mei, H.: Supporting runtime software architecture: A bidirectional-transformation-based approach. Journal of Systems and Software 84, 711–723 (2011)
16. France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. In: FoSE 2007, pp. 37–54. IEEE Computer Society, Los Alamitos, CA (2007)
17. Frank, U., Strecker, S.: Beyond ERP systems: An outline of self-referential enterprise systems. Requirements, conceptual foundation and design options. ICB-Report 31 (2009)
18. Atkinson, C., Kühne, T.: The Essence of Multilevel Metamodeling. In: UML 2001. proceedings, 2185, pp. 19–33. Springer, Berlin, London (2001)
19. Eclipse: Eclipse Modelling Project, http://www.eclipse.org/modeling/
20. Goldstein, A., Johanndeiter, T., Frank, U.: Business Process Runtime Models: Supporting Process Monitoring in a Modeling Environment. A working paper (2013)