Joint Proceedings of the Workshops On the Globalization of Modeling Languages (GEMOC 2013) and Towards the Model Driven Organization (AMINO 2013)

Benoit Combemale[1], Julien De Antoni[2], Robert B. France[3], Balbir Barn[4], Tony Clark[4], Ulrich Frank[5], Vinay Kulkarni[6] and Dan Turk[3]

[1] University of Rennes
[2] University of Nice Sophia Antipolis, France
[3] Colorado State University, USA
[4] Middlesex University, UK
[5] Universty of Duisburg-Essen, Germany
[6] Tata Consultancy Services, India

Table of Contents

First Workshop On the Globalization of Modeling Languages (GEMOC 2013) *

Benoit Combemale[1] and Julien De Antoni[2] and Robert B. France[3] and Frédéric Boulanger[4] and Sébastien Mosser[5] and Marc Pantel[6] and Bernhard Rumpe[7] and Rick Salay[8] and Martin Schindler[9]

[1] University of Rennes 1, France
[2] University of Nice Sophia Antipolis, France
[3] Colorado State University, USA
[4] Supélec, France
[5] Polytech'Nice Sophia Antipolis, France
[6] INPT ENSEEIHT, France
[7] RWTH Aachen University, Germany
[8] University of Toronto, Canada
[9] MaibornWolff, Germany

Abstract. The first edition of GEMOC workshop was co-located with the MODELS 2013 conference in Miami, FL, USA. The workshop provided an open forum for sharing experiences, problems and solutions related to the challenges of using of multiple modeling languages in the development of complex software-based systems. During the workshop, concrete language composition artifacts, approaches, and mechanisms were presented and discussed, ideas and opinions exchanged, and constructive feedback provided to authors of accepted papers. A major objective was to encourage collaborations and to start building a community that focused on providing solutions that support what we refer to as the globalization of domain-specific modeling languages, that is, support coordinated use of multiple languages throughout the development of complex systems. This report summarizes the presentations and discussions that took place in the first GEMOC 2013 workshop.

1 Introduction

Modern software-intensive systems serve diverse stakeholder groups and thus must address a variety of stakeholder concerns. These concern spaces are often associated with specialized description languages and technologies that are based on concern-specific problem and solution concepts. Software and system engineers are thus faced with the challenging task of integrating the different languages and associated technologies used to produce various artifacts in the different concern spaces.

GEMOC 2013 was a full-day workshop that brought together researchers and practitioners in the modeling languages community to discuss the challenges associated with integrating multiple, heterogeneous modeling languages. Supporting coordinated use

* This workshop is supported by the GEMOC initiative (`http://gemoc.org`) and the ReMoDD initiative (`http://www.cs.colostate.edu/remodd`)

of modeling languages leads to what we call the globalization of modeling languages. The languages of interest for the participants ranged from requirements to runtime languages, and included both general-purpose and domain-specific languages. Challenges related to engineering composable languages, semantic composition of languages and to reasoning about systems described using heterogeneous languages were discussed

The workshop GEMOC 2013 was co-located with MODELS 2013 in Miami, FL, USA, on September 29th, 2013. In this report we document the various presentations, as well as the enthusiastic and intense discussions that took place during the workshop. The workshop report is organized as follows. Section 2 gives a broad overview of the workshop, including the topics of interest and relevant application domains. Section 3 describes the paper review and selection process, and the structure of the workshop. Section 4 summarizes the presentations made in the first half of the workshop day. Section 5 then presents the main points raised in the discussions that followed the presentations. The intent of the discussions was to position each presented approach in a language and model composition landscape. An initial global landscape was introduced and discussed in the second half of the workshop day. The results of that discussion are summarized in Section 6. Conclusions are drawn in Section 7.

2 Workshop Overview

Software intensive systems are becoming more and more complex and interconnected. Consequently, the development of such systems requires the integration of many different concerns and skills. These concerns are usually covered by different languages, with specific concepts, technologies and abstraction levels. While the use of multiple languages eases the development of target concerns, it also raises language and technology integration problems at the different stages of the software life cycle. In order to reason about the global system, it becomes necessary to explicitly describe the different kinds of the relationships that can exist between the different languages used in the development of a complex system. To support effective language integration, there is a pressing need to reify and classify these relationships, as well as the language interactions that the relationships enable.

In this context, the workshop attracted submissions that include outlines of language integration approaches and case studies, or that identify and discuss well-defined problems about the management of relationships between heterogeneous modeling languages. The goal was to facilitate discussions among the participants that lead to an initial classification of useful kinds of language relationships and their management.

The Call for Papers explicitly solicited contributions that describe case studies on coordinated use of multiple modeling languages, and/or practical experience, opinions and related approaches. Authors were invited to submit short papers describing (i) their language integration experience, or (ii) novel approaches for integrating modeling languages. Authors were also invited to store full versions of models used to illustrate their novel approach or experience in the Repository for Model Driven Development (ReMoDD). This allowed us to share the actual models with participants and the wider modeling community before, during and after the workshop.

The topics of interest for GEMOC 2013 was:

- Composability and interoperability of heterogeneous modeling languages
- Language integration challenges, from requirement to design, to analysis and simulation, to runtime.
- Model and metamodel composition
- Multi-paradigm modeling and simulation

Submissions describing practical and industrial experience related to the use of heterogeneous modeling languages were also encouraged, particularly in the following application domains:

- Cyber-Physical Systems, System of Systems
- Smart City, Smart Building, Home automation
- Complex Adaptive Systems
- Internet of Services, Internet of Things

3 Workshop Organization

Benoit Combemale, Julien Deantoni and Robert B. France organized and chaired the program committee (PC) for this GEMOC edition. The workshop website[10] and the call for papers (CfP) was online, quite 6 months before the workshop date. Apart from the workshop website, the CfP was announced on different professional mailing lists (e.g., SEWORLD, pUML, planetmde).

We received 11 submissions and finally accepted 8 papers, resulting in an acceptance rate of 72%. Each submission was reviewed by at least three members of the PC. Papers were selected based on their relevance to the topics for the workshop and the reviews provided by PC members. The organizers are very grateful to PC members for performing this important service to the community and for the quality of their work. The PC consisted of: Walter Cazzola, (DICo, University of Milano, Italy), Benoit Combemale (University of Rennes 1, France), Julien DeAntoni (University of Nice Sophia Antipolis, France), Robert B. France (Colorado State University, USA), Jeff Gray (University of Alabama, USA), Jean-Marc Jézéquel (University of Rennes 1, France), Jörg Kienzle (McGill University, Canada), Marjan Mernik (University of Maribor, Slovenia), Pieter J. Mosterman (MathWorks, USA), Gunter Mussbacher (University of Ottawa, Canada), Bernhard Rumpe (RWTH Aachen University, Germany) and Eugene Syriani (University of Alabama, USA).

The format of the workshop reflected the goals of the workshop: To provide constructive feedback on submitted papers and models on the coordinated use of different modeling languages, and to foster collaborations and community building. The format of the workshop was that of a working meeting. Hence, it was less focused on presentations and more on producing and documenting a research roadmap that identifies challenges, different forms of language integration, and relates existing solutions.

The workshop day was split into two parts. In the first part, an introduction and short presentations of the accepted papers were given. The second part of the day was dedicated to open discussions of the presented contributions and other related topics. We lead the discussion towards a classification of existing and proposed forms of language integration. This workshop report is the result of the beneficial discussions we had.

[10] Cf. `http://gemoc.org/gemoc2013`

4 Papers and Models Summary

The papers and the associated talks are available on the workshop web pages (`http://www.gemoc.org/gemoc2013`). The associated models are available in the ReMoDD repository (`http://www.remodd.org`).

[1] Toward Denotational Semantics of Domain-Specific Modeling Languages for Automated Code Generation (Danielle Gaither and Barrett R. Bryant: University of North Texas) This contribution advocates the use of denotational semantics instead of operational ones in order to specify, at the language level, the behavior of models defined using Domain Specific Modeling Languages. The key purpose is to ease the development of Automated Code Generators from these languages through their denotational semantics. Denotations are provided as functions that interprets the models by manipulating metamodel's elements. In future work, the use of mathematical functions should ease the composition of language semantics through function composition. This contribution is applied on a restricted subset of a dedicated modeling language for Role Playing Games.

[2] From Sensors to Visualization Dashboards: Challenges in Languages Composition (Sebastien Mosser, Ivan Logre and Philippe Collet: University Nice-Sophia Antipolis ; Nicolas Ferry: SINTEF IKT) This contribution describes a full fledged use case from the Internet of Things based on the SensApp platform: bikes equipped with sensors and ad-hoc networks transmit different kind of data that are gathered, analysed and forwarded to end users, which can configure their own interfaces to browse the consolidated data. The authors' purpose is to describe the use case and the associated issues, not to advocated a specific solution. The use case contains eleven kinds of models that must communicate and cooperate: Topology, Behavior, Communication, Component, Data, Computation, Resource, Requirement, Task, User Interface and Variability. All these models must be consistent and reusable: sophisticated composition operators are thus needed to build the whole system model from the concern models. The authors identify some relations between the models: co-exists with, uses, constraints, implements. An interesting question is then: are these relation between languages or models?

[3] Heterogeneous Model Composition in ModHel'X: the Power Window Case Study (Frédéric Boulanger, Christophe Jacquet and Cécile Hardebolle, Supélec) This contribution details the use of the ModHel'X toolset for heterogeneous model execution for a Car Power Window system. Strongly inspired by Ptolemy, ModHel'X provides Multi Paradigm Modeling focusing on the semantic adaptation between the various involved Models of Computation. They use the Tagged Event Specification Language (TESL): a DSML for expressing time and control constraints between different heterogeneous parts of the system. TESL takes the synchronous part of the Clock Constraints Specification Language (CCSL), which is part of the UML/MARTE standard, and extends it with time tags on events and relations between the time scale of clocks. The purpose of selecting the synchronous subset of CCSL is to ease the implementation of tools around TESL. Adding a time tag to events, and establishing relations between time tags is more

efficient for simulation because it allows arbitrary precision on the date of events without requiring high frequency clocks. The Car Power Window models include Discrete Event (communication between parts), Timed Finite State Machine (controller) and Synchronous Data Flow (mechanical parts) with adaptation between DE and TFSM, and between DE and SDF.

[4] Railroad Crossing Heterogeneous Model (Matias Ezequiel Vara Larsen, University Nice-Sophia Antipolis ; Arda Goknil, INRIA Sophia Antipolis) This contribution focuses on the composition of heterogeneous models applied to a Railroad Crossing Management System (RCMS). Models are conforming to languages expressed with four language units: Abstract Syntax (AS), Domain Specific Actions (DSA), Domain Specific Events (DSE) and Model of Concurrency (MoC). These units are put in consistency through the DSE, which are expressed using the Event Constraint Language (ECL), an extension of OCL with events and event relations. From an ECL specification on a language, an automatic transformation of a model is provided to create the CCSL constraints, i.e. the execution model of the model. These constraints are then solved to drive the execution of the models. Composition operators express coordination between the DSE of two or more languages and are used to create constraints written in CCSL that are combined with the ones provided by the execution models of the languages. The whole approach is illustrated with the Rail Crossing Management System that combines a Barrier Detection Controller modelled using Timed Finite State Machine and a Barrier Motor Controller modelled using fUML.

[5] On the Challenges of Composing Multi-View Models (Matthias Schöttle and Jörg Kienzle: McGill University) This contribution targets the specification of complex systems using separation of concerns. The key aspects is a strategy to integrate metamodels and the associated model composers to build multi-view formalisms including multi-view composers. Each metamodel is provided with an internal model composer. This proposal relies on an asymmetric approach: one of the metamodels is selected as independent metamodel which is unchanged, and the other metamodel is integrated in this one. The independent metamodel composer is thus unchanged, and the whole multi-view composer is derived automatically from the integration of the second metamodel. This proposal is applied to the Touch RAM toolset implementing Reusable Aspect Models that allow specifying and reusing concerns in the development of complex systems. RAM provides structural, behavioural and protocol modelling using class, sequence and state machine diagrams.

[6] Using partial model synthesis to support model integration in large-scale software development (Marsha Chechik and Rick Salay: University of Toronto) This contribution targets the synthesis of model stubs that enable the simulation of models specified using interface contracts. In the development of complex systems with many stakeholders, model interface contracts allow to loosen development schedule constraints. However, these contracts cannot be executed and thus the whole system can only be simulated when all models have been defined. The synthesis of models satisfying those contracts provide stubs that can be used in that purpose. These models can be refined incrementally when additional data are available during the development. This proposal

is applied to the models of a webmail system relying on sequence diagrams to describe interaction scenarios and Fluent Linear Temporal Logic to express system invariants. Other experiments are planned to extend the proposal to other kind of models of computation.

[7] Enhance the Reusability of Models and Their Behavioral Correctness (Papa Issa Diallo and Joël Champeau: ENSTA Bretagne) This contribution targets the preservation of model semantics during a Model Driven Engineering process that combines several executable languages and simulation tools using model transformations from languages to languages. In that purpose, the authors rely on the COMETA framework to specify high level Models of Computation relying on the low level MoC provided by each model execution toolset. This approach constrains the usual execution of the models inside a given tool in order to provide a common semantics for the various models of the same system during the development phases. This proposal is applied to the integration of two toolsets involved in the development of safety critical systems: Rhapsody UML and ForSyDe-SystemC.

[8] Black-box Integration of Heterogeneous Modeling Languages for CPS (Markus Look, Antonio Navarro Perez, Jan Oliver Ringert, Bernhard Rumpe and Andreas Wortmann: RWTH Aachen University) This contribution targets the integration of six independently developed modeling languages and its application to the robotics domain. These languages includes a component & connector architecture description language, automaton, I/O table, class diagrams, OCL, and a Java DSL. The resulting MontiArcAutomaton modeling framework allows to model the logical software architecture and the system behavior of robotics applications. Even the languages were not developed for the robotics domain in the first place, the existing language components could be completely reused using the language composition approaches of the MontiCore framework.

5 Why, What, Where, How Composing Languages and Models?

An important part of the workshop was dedicated to discussions on how to realize the vision of globalized modeling languages. The aim was to map out a landscape of language and model composition issues that reflected ideas raised in the previous presentations and the experiences of workshop participants. To structure the discussions, the organizers proposed to follow the Why? What? Where? How? pattern as reported in the rest of this section.

5.1 Why?

Composition of models is highly desired, because the history of computer science has shown that development can only be managed by dividing a complex task or product into smaller easier sub-structures. This in particular holds for models which we like to decompose along sub-structures of the product, e.g. into components [2], or along viewpoints that allow us to look at the same components from different perspectives [5]. However, decomposition is useful only if the different realizations obtained through the design of the subsystems, can be composed again to obtain a realization of the whole system. Structural decomposition of models needs a composition operator present within the modeling language, for example, in finite state machines, where composing two state machines leads to another one or in class diagrams, where diagrams are basically merged along classes with the same name.

When different viewpoints are modeled in different languages, model composition also becomes language composition, where models of different languages need to become interoperable. This is necessary for example for the UML, where structural and behavioral models describe the same system, but also in the recently emerging and potentially more interesting domain of distributed systems, where several different, but similar languages are used to describe the behavior of system components [2].

We can identify four main reasons for using models written using different formalisms:

– The decomposition of a complex system into different parts that belong to different technical domains which have their own modeling formalisms and tools;
– The change in the level of abstraction during the design of a system, for instance from a non-deterministic automata for specification, to VHDL for the concrete realization;
– The need for different models of a system that model different aspects, or offer differents views on the system and therefore use different modeling formalisms: the timing or power consumption model of a system may use a different formalism than the functional model;
– The need for different models of a system for different activities during the design process: a model used for code generation and a model used for verification may use different formalisms.

These four reasons lead to four corresponding issues:

– How does one compose the structure and behavior of heterogeneous models?
– How does one check that a model is a refinement of a more abstract model?
– How does one synchronize different views on different aspects of a system?
– How does one check the consistency of different models of a given system?

Only if these forms of composition are well understood will we be able to carry out integrated code generations, simulations, validations and verifications on integrated models. These composition issues are rooted in the semantics of the models, with the added complexity of the definition of a mapping between concepts that belong to different semantic domains and are represented according to different syntaxes.

Even though the composition of models and languages is somehow related to the forms of composition for products, we need to be explicitly aware that these are different forms of compositions. That is why it is necessary to explicitly clarify what the composition is about. In particular, it would be very nice if a specification formalism provides a composition operator for its specifications in such a way that this composition conforms to the composition of the specified components.

Identifying the relationships that exists between different languages is useful to support the understanding of language composition, from a systemic point of view [2]. Understanding this kind of relations leads to a classification of existing composition relationships, e.g., uses (a computation model uses a data model), implements (a task model implements a use case). An off-the-shelf classification of composition in the context of DSML provides guidance to engineers who have to perform such compositions.

Additionally, besides the technical forms of composition, the ability to compose has also an organizational consequence. If we can compose individually developed models, we can decompose the team into smaller sub-teams developing individual parts. This is why composition of models is an important requisite for successful projects developing complex products.

5.2 What?

Formally, composition is an operator taking at least two arguments and producing one result. Model composition thus involves manipulation of models. Each model has a syntax and a semantics. Composition can operate on the syntax, for example deriving an integrated new model describing the information originally contained in both models [8]. Composition could also work on the semantics only [4]. This, for example, works well with denotational semantics, when the models are mapped onto a common semantic domain and the composition is denoted using composition in the semantic domain only [1]. As another possibility we could explain composition at the code generation level, by explaining how the code, which has been derived from each model individually, is linked together [7]. In any case, this usually requires an understanding of the interfaces between models, and respectively, their derivatives [4,3,7].

Furthermore, since models are developed incrementally across the development lifecycle and at different rates in different sub-teams, requiring models to be complete before composition can cause project delays. To address this we must also allow for the meaningful composition of partially complete models [6]. Such compositions must preserve the information that is known without biasing the information that is still unknown.

Relying on an existing classification of existing composition supports the identification of the elements to compose during the process of designing a given composition. One can rely on the existing relations to guide its own development and support incremental approaches. The fact that a language L “implements” concepts modeled by a language L’ implies that both elements (the concept to be implemented in L’ and its associated artefacts in L) exists at the time of the composition, even if developed asynchronously. On the contrary, if L “uses” elements from L’, one cannot use L to work while the model in L’ is not complete.

These considerations hold for any kind of modeling languages, structural or behavioral. However, from a practical point of view, in a top-down approach we develop structure first and thus need decomposition on structure on the higher level. Typically behavioral aspects are only modeled and thus composed on a more fine-grained details level in the later phases of the development. On the other hand, a bottom-up approach involves composing already developed parts of a system to produce a complete system. However, in bottom-up approaches, the composition is usually made at the model level by making interoperable homogeneous interfaces and usually do not address composition of syntactic elements.

5.3 Where?

We outlined in the previous section how the composition can operate on different parts of languages and models. It is also interesting to note that the composition can be specified and applied at different level: From the language level to some compiled code. Additionally, the specification of the composition and its application can also be done at different levels. For instance, a composition operator can be specified between two languages in order to create a new language [8]. In this case, both the specification and the application of the composition is done at the language level. Another composition operator could be specified between two languages in order to create relations between the models of the respective languages, but without explicitly creating a new language [4]. It is important to identify at which level the specification and the application of the composition is made to understand the impact on the reuse of the composed language tooling. On this point, several aspects of the involved languages have to be considered:

- concrete and abstract syntax
- semantics
- consistency checks
- code generators

This also includes their resulting infrastructure, for example, lexer, parser, symbol tables, or editors. For an efficient and agile composition of existing languages, these aspects should be reused as much as possible and only a minimal amount of additional glue-code should be necessary. The level at which the specification and the application of the composition is realized depends on the main objective of the composition (i.e. the why). For instance, because the semantics of a language combination has to be considered, the composition is preferably defined on the language level. However, in a reuse perspective the linkage should take place as late as possible to be able to provide the same language component for different compositions. The idea behind this is the same as for framework or libraries in programming languages which can be used as precompiled components without providing the sources. This speeds up the development process significantly as the libraries do not have to be compiled again and again for every usage while it restrains the possibility of analysis of the composed system.

5.4 How?

In order to be able to compose different languages and models without developing the languages for each combination from scratch again or specifying the matching parts for each model combination we need a concise and consistent definition of the interfaces of languages and models. An approach similar to that used in existing programming languages, where methods or classes have an interface for its usage encapsulating the complexity of the actual implementation, may be applicable. If languages provide interfaces that hide their internal complexity, a library of language components could be created allowing the composition of languages by writing a minimal amount of glue-code for each combination.

The need for an interface holds for models as well. Each model should provide an interface which can be used from other models without knowing the internal details [3,6,7]. This means that the modeling languages have to provide two concepts: one for describing the interface of a model and one for referencing or calling such interfaces. These concepts can even be used to combine models whose languages were originally developed independently. To give an example let us take the concept of a method call in a programming language like Java which was originally designed as a reference to a method signature within the same language. However, this concept could be reused to call other interfaces as well, e.g. an interface to a Statechart. This only requires that a method call allows one to specify all aspects requested by the Statechart interface. If this holds only a mapping of the concepts of the reference (the method call) and the targeted interface (of the Statechart) is needed. In this way two languages can be composed by adding an additional semantics to an existing language concept without changing the original language itself. It can also be seen as scheduling the new events from the statechart (start, entry in a state, trigger, etc) together with the events of the original java program and specifically here with the events of its method call (call, return)[4,5].

By using such approaches, a composition of languages can be reduced to a mapping between the language concepts needed to combine models along their interfaces. Only if the existing language concepts are not sufficient to define such a mapping, an extension of one or both languages is needed for the composition.

And finally the composition of the language tooling have to be considered. However, if all of these aspects are implemented using clear interfaces, the composition could be derived from the already defined mapping of the language concepts even automatically.

When only partial information is known about a model, techniques such as model synthesis can be used to construct an approximate, but well-formed, model based on the known information. Such approximate models can be used as stand-ins for partially complete models in composition operations [6].

The various criteria about compositions lead to various implementations. Such implementations can be hard coded [6,8], based on predefined operators [5], or specified thanks to a dedicated DSL [4,3] Additionally, many important properties can be used to characterise the implementation, for example implementations may be asymmetric or symmetric, or may be transient or not. These properties should be clarified and we should understand how the why of the composition is influencing such choices.

6 Conclusion

The first edition of the workshop met its goals and thus can be considered a success. The discussions that took place during the workshop were of very high quality and provided insights into some of the pressing problems associated with the use multiple modeling languages in development projects. A key result is a deep appreciation by the participants of the need to develop support for globalizing modeling languages. The ongoing research that was reported in the workshop and the discussions that took place are a good indication that community around these problems is emerging. For more information about the GEMOC initiative please visit the following website: `http://gemoc.org`.

7 Acknowledgments

GEMOC 2013 was supported by the GEMOC initiative (cf. `http://gemoc.org`) that promotes research seeking to develop the necessary breakthroughs in software languages to support global software engineering, i.e., breakthroughs that lead to effective technologies supporting different forms of language integration, including language collaboration, interoperability and composability. Part of this initiative, the workshop was partially supported by the ANR INS project GEMOC (grant #ANR-12-INSE-0011).

GEMOC 2013 was also supported by the ReMoDD project, an NSF funded project (award number CNS-1305381).

References

1. Gaither, D., Bryant, B.R.: Toward Denotational Semantics of Domain-Specific Modeling Languages for Automated Code Generation. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
2. Mosser, S., Logre, Y., Ferry, N., Collet, P.: From Sensors to Visualization Dashboards: Challenges in Languages Composition. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
3. Boulanger, F., Jacquet, C., Hardebolle, C., Dogui, A.: Heterogeneous Model Composition in ModHel'X: the Power Window Case Study. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
4. Vara Larsen, M., Goknil, A.: Railroad Crossing Heterogeneous Model. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
5. Schottle, M., Kienzle, J.: On the Challenges of Composing Multi-View Models. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
6. Chechik, M., Salay, R.: Using partial model synthesis to support model integration in large-scale software development. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
7. Diallo, P.I., Champeau, J.: Enhance the Reusability of Models and Their Behavioral Correctness. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)
8. Look, M., Navarro Perez, A., Ringert, J.O., Rumpe, B., Wortmann, A.: Black-box Integration of Heterogeneous Modeling Languages for Cyber-Physical Systems. In: GEMOC workshop 2013 - International Workshop on The Globalization of Modeling Languages, Miami, Florida, USA (September 2013)

Towards the Model Driven Organization Introduction to the 1st AMINO Workshop

Held at the ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems MODELS 2013 Miami Florida USA

Balbir Barn[1], Tony Clark[1], Robert France[2], Ulrich Frank[3], Vinay Kulkarni[4], and Dan Turk[2]

[1] Middlesex University, London, UK
[2] Colorado State University, Colorado, USA
[3] Universty of Dusibirg-Essen, Essen, Germany
[4] Tata Consultancy Services, India

Organizations such as banks and public sector institutions can be thought of as being structured in three key layers. The *strategic* layer defines what an organization must achieve in terms of its high-level *goals*, the *tactical* layer defines how an organization plans to behave and thereby achieve its goals, and the *operational* layer defines the day-to-day running of the organization in a manner that is consistent with the organization's plans. The operational layer of a modern organization is implemented in terms of a collection of inter-connected IT systems that form an *organizational platform*. An organization seeks to *align* its high-level goals with its platform so that its strategy is properly supported by its IT infrastructure. Expressing and achieving alignment remains a key challenge for modern organizations.

Essentially organizations exist to accomplish specific goals. In the case of a *not-for-profit* organization, the goals are often expressed as a *mission statement*. In the case of a *for-profit* organization, the mission should also include achieving profit for investors. Frequently these goals are not consciously or explicitly written down or even clearly recognized. But, even so, they still exist. Whether goals are explicitly identified and documented or not, organizations wish to know whether the goals are achieved and, where possible, concrete metrics may be used to help answer these questions.

Whilst top-level organizational goals can often be captured as relatively simple mission statements, the internal processes, resources, IT systems and structures that are put in place to realise the goals are usually highly complex. Measuring aspects of a typical large multi-national corporation in order to determine its internal consistency, to measure quality metrics, to perform any number of change-based activities, and ultimately to establish consistency between the organizational goals and the human- and IT-centric processes by which it operates, is a difficult task.

In the field of software system development, it is accepted that various forms of modelling need to be brought to bear on the problem of measuring complex system properties, controlling its behaviour and to establish that an implementation satisfies its specification. The key feature at play is *abstraction* where unnecessary detail is elided leaving important aspects of a system to be represented in a language that is suitable for the stakeholders and exhibits appropriate properties for reasoning, transformation, simulation *etc.* Our claim is that the same use of abstraction through modelling can be applied to the problems of managing an organization, leading to the idea of a Model-Driven Organization:

> **Def:** A *Model-Driven Organization* (MDO) is an institution that uses models, both formally and informally, in conscious and intentional ways, throughout the organization to define who it is and what it does, to help better and more quickly train employees, to carry out, assess, and improve its functioning, and to more effectively develop and modify systems that support the organization.

The use of models in system development has established the fields of Model Driven Architecture (MDA) and Model Driven Engineering (MDE). Although these fields use models as abstractions of systems in order to establish properties, they predominantly use models to help with system construction during the design phase. In some cases part or all of the program code for a system is generated from models.

Although MDA/MDE may be viewed as providing a contribution, the MDO vision is much broader than system development. An MDO uses models at multiple levels: Organizational level models, not just system / software level models; conceptual, as well as technical models; strategic, tactical, and operational models. MDO models relate to much more than just the IT systems of an organization and need not be expressed in a formal language. The models must address people-centric aspects of an organization in addition to the systems-level aspects. We can see this by comparing the definition of the MDO as given above, with the following two common definitions for MDE and MDA:

> **Def:** *Model-Driven Engineering* (MDE) is a software development methodology which focuses on creating and exploiting domain models (that is, abstract representations of the knowledge and activities that govern a particular application domain), rather than on the computing (or algorithmic) concepts.

> **Def:** *Model-Driven Architecture* (MDA) is a software design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models. Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems. It was launched by the Object Management Group (OMG) in 2001.

Our claim is that a model-based approach can be used in many ways within an organization. Models can be used to describe the organization itself, its structure, its processes, its business rules, its regulatory constraints, *etc.* This can help in a wide variety of organization-centric use-cases including training new employees and helping outsiders better understand and effectively interact with the organization. These models may be able to help potential investors compare organizations, or even make it easier to merge organizations. They may make it easier to compare the functioning of organizations, or systems within organizations, to allow decision-makers to better determine which parts of merged organizations to keep and which to eliminate after a merger takes place, for instance. Models may be used to help in the analysis of an organization, to determine and clearly identify its mission, and assess how well it is meeting its goals.

If the models are formalized and automated, it may be possible to ease and/or automate some types of changes within an organization. And, of course, models have had a long history of helping to automate parts of organizations.

Achieving the MDO vision will involve expertise from many different disciplines. Experts in modelling will be required to design suitable languages both domain-specific and general purpose using appropriate meta-technologies. Expertise in model processing including transformation, synchronization and model-checking will be required. Experts in the field of Organization Theory will be required to understand the structures and processes to be modelled. Information Systems experts will be required to analyse and represent the ontologies and complex data used within an organization. Management theorists will be required to understand the human-centric aspects of the problem and where models can be used to facilitate all stakeholders. Experts in tools will be required to determine how to offer the features of the MDO in an effective way.

The goals of the AMINO workshop are to work towards a better understanding of where models can be used to address all aspects of organizational development, operation and management use-cases. The workshop took the form of an invited speaker and presentations of 7 peer reviewed papers. The rest of this introduction provides an overview of the presentations.

In *Goals, Domains, and Enterprise Architecture in the Model-Driven Organization* Desmond D'Souza argues that goal models are fundamental to achieving the MDO. He presents a notation and method for incremental model construction in terms of goals and their constraints over domains. The paper concludes by indicating how goals can be exploited in other MDO areas including architectures and migration plans.

In *Multimodel-Driven Software Engineering for Evolving Enterprise Systems* Richard Paige, Radu Calinescu, Dimitrios S. Kolovos, Nicholas Matragkas and Dave Cliff address the issue of organizational evolution and how to analyse the Quality of Service (QoS) issues that arise. The paper proposes a multimodel-driven approach (MMSE) and concludes with some thoughts about a research agenda that will lead to solutions in this area.

Organisations consist of many different roles that link to business processes and provide access to information at various security levels. In *UML/OCL based Design and Analysis of Role-Based Access Control Policies* Oliver Hofrichter, Martin Gogolla, and Karsten Sohr discuss an approach to modelling the access control and use OCL to address a case study based on EasyChair.

As discussed above, a key feature of the MDO is goal-IT alignment. In *Metamodel Patterns for Expressing Relationships Between Organization Model Concepts and Software Implementation Concepts* Jens Gulden discusses this issue and provides patterns for mapping between the two levels.

The state-of-the-art in Enterprise Modelling and Enterprise Architecture uses enterprise frameworks to represent and reason about aspects of an organisation or even its entirety. Few of these frameworks use modelling technologies and techniques. In *MDE Support for Enterprise Architecture in an Industrial Context: the TEAP Framework Experience* Hugo Bruneliere, Jordi Cabot, Stphane Drapeau, Flavien Somda, William Piers, Juan David Villa Calle and Jean-Christophe Lafaurie describe a new framework called TEAP that applies Model Driven Engineering techniques to the construction of an enterprise framework.

In many ways organisations are more complex that standard software systems and the structures, information, resources and processes involved are less precise. Therefore it is important to understand the reasoning behind design decisions and the reasons for organisational change. In *Introducing Argumentative and Discursive Enterprise Leading and Management* Sebastian Bittmann, Balbir Barn, Tony Clark and Oliver Thomas develop this theme in terms of *argumentation theory* as a basis for recording the intentions behind organisation-related actions.

There are many different organizational use-cases involved in achieving an MDO. One is *mergers and acquisitions* whereby two organisations become one. In *(Multi-) Modeling Enterprises for Better Decisions* Sagar Sunkle, Vinay Kulkarni, and Hemant Rathod argue that in order to represent and reason about such complex use-cases it is necessary to use multi-models.

The MDO is likely to involve the use of models in many different ways. Some of these may simply be to help understand the organisation, but others may help to operationalise it. In *Enterprise Models as Drivers for IT Security Management at Runtime* Anat Goldstein and Sietse Overbeek discuss the technique of models@Runtime and its use to achieve IT security.

Goals, Domains, and Enterprise Architecture in the Model-Driven Organization

Desmond D'Souza

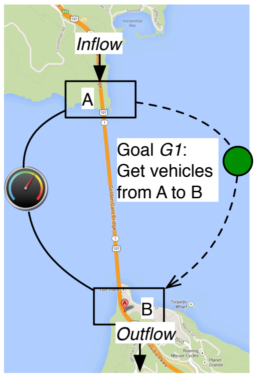
Kinetium, Inc.

desmond.dsouza@kinetium.com   www.kinetium.com

In a Model-Driven Organization (MDO), all aspects of planning, designing, implementing, deploying, operating, and evolving the organization and its supporting infrastructure are based on models. Goals form an anchor for other models in an MDO. This paper is a summary of my keynote talk at AMINO 2013 in Miami, Florida, in which I covered a variety of ways to use Goal Models as a recurrent focal point to improve model coherence in an MDO.

Goals, Domains, and Refinement

A goal is a property desired over a domain. In Figure 1(a), a goal of the Golden Gate Bridge is shown with a green circle *G1: Get vehicles from A to B* i.e. given some *domain property* about the *inflow* of vehicles at *A*, establish a corresponding *outflow* at *B*. A goal is anchored to domain elements that are its subject matter, and evaluates to true or false over any domain instance. *G1* is anchored at end-points *A* and *B*, controlling *outflow* based on *inflow*. A dashboard wired to the anchor points can, in principle, monitor the goal.



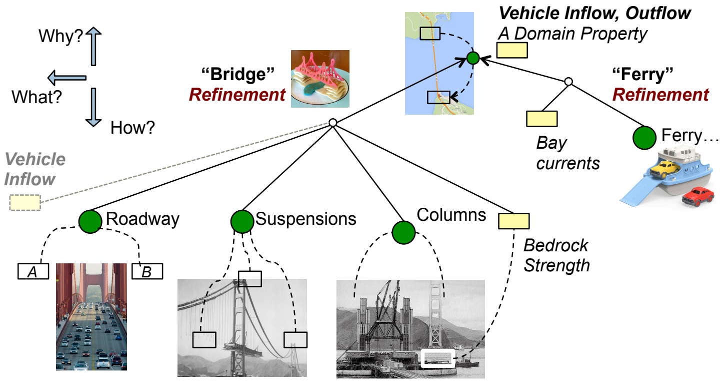


Figure 1: (a) A goal with its end-points and a monitor.   (b) Goal and domain refinement.

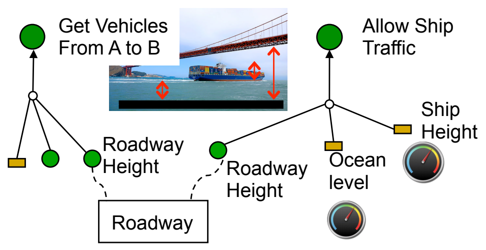
A goal *refinement* consists of sub-goals and domain properties that, combined, satisfy a parent goal across its end-points. The *bridge refinement* (small white circle in Figure 1(b)) moves *inflow* from *A* to *B* via a roadway, supported by cables suspended from columns built on bedrock, and establishes sub-goals for each of these based on their mutual load and support properties, bedrock properties, and inflow. Domain properties (yellow rectangles) are facts or assumptions about the domain, also anchored on domain elements. A goal can have alternative refinements: *G1* can be met by a *bridge-refinement* or a *ferry-refinement*. Choosing one or the other uncovers different domain properties: *bedrock strength* is vital for bridge columns, and *bay currents* is for a *ferry*, while *inflow* and *outflow* is part of the problem context of *G1* itself and common to both. When analyzing any part of a goal model, only certain refinements apply, and the corresponding domain model is *composed* from the domain properties of all applicable refinements. Goal refinement, domains, and architecture choices are intrinsically intertwined.

The domain model can range from a simple average vehicles-per-hour, to a detailed "film-strip" with individual vehicles moving along the bridge. Goals are predicates over that model, and evaluate to true or false on a domain instance. An *objective* associated with a goal can quantify how well the goal is met, based on measurable domain attributes and often in an aggregate sense.

By making *intention* explicit, goals answer some key questions (Figure 1(b)):

1. *What* are we trying to accomplish? The goal specification, *G*, with end-point domain elements, gives a precise success criteria over any domain instance.
2. *Why* do we want to accomplish *G*? Refinement answers this in the form: because if we meet *G*, given additional domain property X and assuming we meet other sibling goals, then in concert we meet the higher level goal.
3. *How* will we accomplish *G*? Examine the refinement of *G* and follow the "line of reasoning" through its domain properties and sub-goals.
4. *How well* does refinement *R1* meet *G*? Provided all children of *R1* can be expressed in a sufficiently detailed form, evaluate *G*'s objective function against domain instances assuming that all the sub-goals are met.

Monitoring a goal is clearly useful, but *assumed* domain properties are also candidates for monitoring. As a secondary goal, the bridge should accommodate shipping traffic, which introduces assumptions about ocean level and ship height, and goals about minimum roadway height (Figure 2(a)). The assumptions can be monitored, as changes could jeopardize a goal. In the larger feedback loop of an extended enterprise with its environment, assumption tracking must happen in some form, whether proactive or reactive.



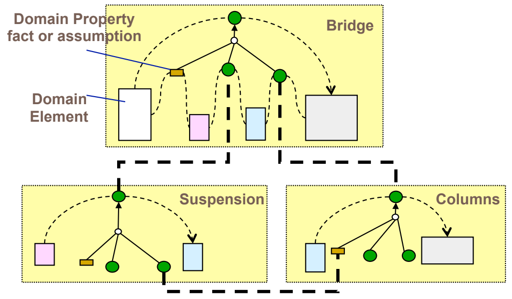


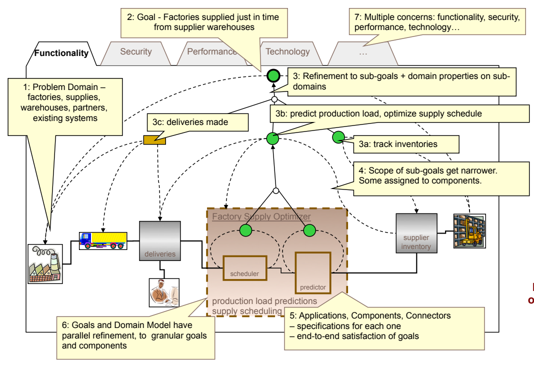
Figure 2: (a) Monitoring assumptions. (b) Federated goal models and alignment.

Goal models can be federated. Figure 2(b) shows federated goal models for the bridge, suspension, and columns. Note that each goal "frames" a problem; goals and end-points must align from child to parent goal model; one model's assumption can be (or otherwise depend on) another model's goal; and goal models have different projections of shared domains. Since dependencies are between properties of domain elements and not directly between elements, value, risk, and impact can be assessed in terms of affected properties.

If the underlying domain fits within a governance structure, goals can be extended with strategic dependency relations between the person desiring the goal, the one responsible for meeting the goal, and the one with authority over any domain element needed in refining the goal.

Goals, Architecture, and Architecture Style

An *architecture* of a system is a *model* describing system *properties* to be understood and analyzed *together*. Architectural components are part of the domain model. Goals and domain properties demarcated by refinement define *"together"*-ness and are part of architecture. The "ends" in "end-to-end architecture" are precisely framed by goals.


2: Goal - Factories supplied just in time from supplier warehouses
7: Multiple concerns: functionality, security, performance, technology…
Functionality
Security
Performance
Technology
…
1: Problem Domain – factories, supplies, warehouses, partners, existing systems
3: Refinement to sub-goals + domain properties on sub-domains
3b: predict production load, optimize supply schedule
3c: deliveries made
3a: track inventories
4: Scope of sub-goals get narrower. Some assigned to components.
deliveries
Factory Supply Optimizer
scheduler
predictor
supplier inventory
production load predictions
supply scheduling
6: Goals and Domain Model have parallel refinement, to granular goals and components
5: Applications, Components, Connectors
– specifications for each one
– end-to-end satisfaction of goals

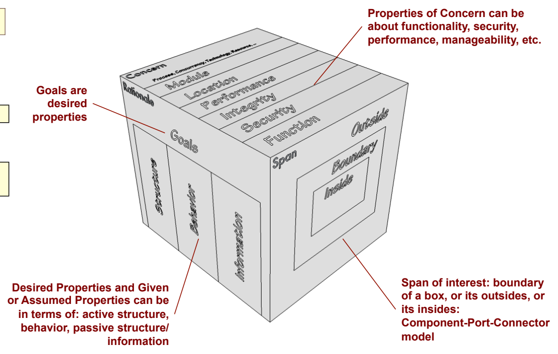

Properties of Concern can be about functionality, security, performance, manageability, etc.
Goals are desired properties
Concern
Process, Concurrency, Technology, Resource, …
Module
Location
Performance
Integrity
Security
Function
Rationale
Goals
Span
Outside
Boundary
Inside
Structure
Behavior
Information
Desired Properties and Given or Assumed Properties can be in terms of: active structure, behavior, passive structure/ information
Span of interest: boundary of a box, or its outsides, or its insides: Component-Port-Connector model

Figure 3(a) Example of fractal refinement. 3(b) Scheme for fractal goals with architecture.

Figure 3(a) sketches an example of the approach used in a fractal manner for factory supply optimization: the granularity of goals and domain elements, including software components, events simple or complex, and other behaviors, can all be refined recursively; 3(b) shows a general scheme for incorporating goals as a fractal structure alongside other architecture descriptions.

An architecture description can obscure, suggest, or reveal intention. In Figure 1(b), hanging cables longer than some length need to be reinforced as their own weight adds to the cable strain at the top. Below are three architecture descriptions of the cables, ranging from obscuring to revealing intention.

1. Cables 1-6 are normal, and cables 7-9 are reinforced.
2. For every cable: *(weight, strength, load)* must satisfy a *strength rule*.
3. Cables = map *cable_reinforcer* basic_cables

The first reveals no intent; it simply lists the final result of implicit reasoning. The 2nd reveals intent as a *checkable* rule, without helping shape the architecture. The 3rd provides an intention-revealing *transformation* that is generative.

An *architecture style* defines a set of architectures, and is described like any object type: attributes that name relevant architectural elements (*cables*), attributes of those elements (*weight, strength*), functions that determine attributes from others (*cable_reinforcer*), invariants (*strength rule*), desired and given domain properties. Like the *cables* example, architecture styles can span a spectrum from check-only (given an architecture, return Pass/Fail) to generative (given an architecture, evaluate to a transformed architecture). Since architectures include goals and domain properties, the most generative kind is an "architecture compiler": given goals and domain context, produce an architecture realization.

Goals and Roadmaps

Roadmapping is a decision-making technique used to support medium to long-term strategic planning, examining linkages between the domains of technology and business capabilities, organizational objectives, and the customer and market environment, considering multiple perspectives and their relationships over time. An MDO could analyze goals, facts, assumptions, and architecture elements in these domains on a timeline; key assumptions could bear monitoring over this timeline. Figure 4 illustrates what such a roadmap looks like.

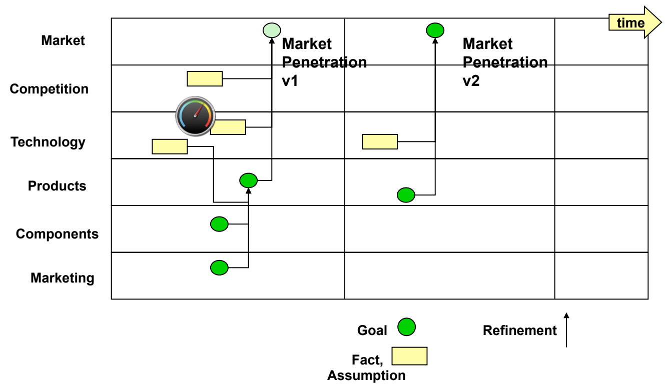
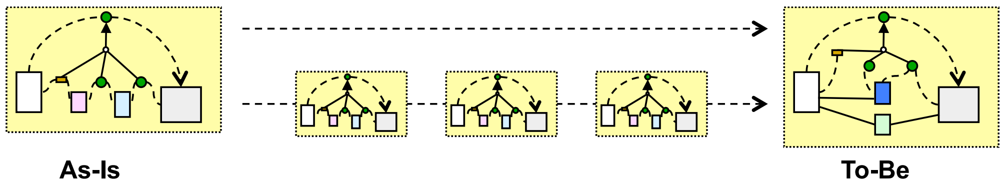


Figure 4. Example of roadmap based on goal modeling.

Goals and Migration Plans

Large-scale architecture initiatives deliver an *as-is* and a *to-be* architecture, and an MDO could use goal models in developing these. A *migration plan* is a sequence of staged architecture changes from *as-is* to *to-be*.



Migration planning is a difficult problem, with its own goals and domains. The obvious domain is a filmstrip of architectural stages, but there are peripheral components, applications, processes, people, skills, and regulations to consider; other roadmaps to co-ordinate, such as infrastructure upgrades. The obvious goal is to convert *as-is* to *to-be*; but there are others, such as minimizing disruption risk to critical business processes. Migration planning even has its own *architecture styles*, such as *Legacy Coexistence* and *High-Risk First / Fail Early*.

Acknowledgements

This paper is based on a modeling approach we have evolved over several years (called MAp, www.kinetium.com). It uses ideas from KAOS, Problem Frames, Catalysis, and Model Driven Engineering. I also owe thanks to past collaborators including Tony Clark and Hans Gyllstrom. The keynote slides also discussed goal-models and portfolio management, excluded here for lack of space.

Multimodel-Driven Software Engineering for Evolving Enterprise Systems (Position Paper)

Richard F. Paige[1], Radu Calinescu[1], Dimitrios S. Kolovos[1], Nicholas Matragkas[1] and Dave Cliff[2]

[1] Department of Computer Science, University of York, UK.
{richard.paige, radu.calinescu, dimitris.kolovos, nicholas.matragkas}@york.ac.uk

[2] Department of Computer Science, University of Bristol, UK.
dc@comp.bristol.ac.uk

Abstract. We advocate the use of multimodel-driven software engineering for the principled evolution of enterprise systems whose stakeholder concerns are captured using multiple interdependent models. *Enterprise systems* that evolve are increasingly common in healthcare, transportation, e-government and defense. These important systems must be regularly extended with new components satisfying interdependent functional, governance and quality-of-service (QoS) requirements that are modelled using different domain-specific languages. We describe key challenges associated with modelling, reasoning about QoS properties, and evolving such systems. The concepts of this engineering paradigm are presented in the context of a statistical reporting project carried out in collaboration with healthcare organisations.

1 Introduction

The recent advent of technologies ranging from cloud and mobile computing to smart sensor networks has led to the emergence of new types of data and applications at an extremely fast rate. This trend is amplified further by equally rapid changes in public information governance. The open data movement, in part spearheaded by the UK Government [1] and recently embraced by all G8 Governments [2], has opened up a wide range of public datasets for research and commercial use. Healthcare, transportation, education and local government are only a few of the areas in which new public datasets are released (e.g., `data.gov`, `data.gov.uk`) on a daily basis.

These developments have created business and research exploitation opportunities for both public and commercial organisations. To exploit these opportunities – and to comply with changes in information governance and other requirements – such organisations must *evolve* their enterprise systems on a regular basis. Information systems supporting new business processes must be engineered and integrated *on the fly* with existing enterprise systems, and must then be updated frequently in response to new stakeholder requirements and governance policies. Plausible examples of such evolving enterprise systems are encountered in the health and social care domain. In the UK for instance, the recently enacted Health and Social Care Act 2012 [3] required an organisation to "*establish and operate a system for the collection or analysis of information of a description specified in the request*" of "*any person*", at any time.

The stringent demands of *evolving enterprise systems* cannot be achieved using today's software engineering approaches. Traditional enterprise system engineering approaches comprise manual processes that cannot respond to such demands in a timely manner, and are costly and error prone. Model-driven software engineering - which automates development processes by synthesising software artefacts from models - is challenging to apply, as the concerns of new information systems (e.g., functionality, governance and quality-of-service) cannot easily be captured by a single model.

We envisage that overcoming the limitations of existing approaches and achieving the goals of evolving enterprise systems requires *multimodel-driven software engineering* (MMSE). This software engineering paradigm will automate key processes of evolving enterprise systems whose concerns are described by multiple interdependent models, when these models are specified by different stakeholders, in different domain-specific languages. Significant research challenges must be addressed to achieve this vision. A key concern is integrating QoS models throughout the engineering lifecycle. In particular, the research community will need to address the open research questions of how to devise multimodel transformation techniques that consider inter-model dependencies (where some are QoS models), and how to co-evolve sets of interdependent metamodels. Another key challenge is the joint analysis of quality-of-service (QoS) models for dependability, performance and resource usage, to identify system configurations that deliver effective QoS trade-offs.

Our paper summarises these key challenges, and sets a research agenda for the delivery of the software engineering formalisms, techniques and tools needed to automate the development, analysis, adaptive configuration and evolution of information systems specified by sets of interdependent functional, governance and QoS models.

2 Background

Recent advances in model-driven software engineering (MDSE) address many challenges of developing traditional software systems. MDSE is a software development paradigm that aims to use (software) models as the main development artefact instead of code [4]. Achieving this aim within a problem domain involves the use of domain-specific languages (DSLs) to define models of the systems to develop [5]. Automated transformations can be applied to them to generate models of lower abstraction levels, which ultimately can be transformed into code. The advantages of MDSE over other approaches to software development include significant improvements in software quality and development efficiency, and increased reusability of software components [4, 5]. The research efforts to exploit these advantages produced a broad range of effective MDSE modelling languages and tools (e.g., [6, 7]).

More recently, the MDSE paradigm was extended to also cover the post-development stages of the software lifecycle. In this extended MDSE approach, models continue to be used as the primary artifact in the maintenance and evolution of software systems. A key challenge of using MDSE in this context is that the models and metamodels used at different levels of abstraction may change asynchronously as the software system evolves, creating ripple effects on related artefacts such as model transformations and validation constraints. Different aspects of this challenge have been addressed by recent research on *model management* [8], *model-metamodel co-evolution* [9], and *incremental and bidirectional model synchronisation* [10].

In parallel with these advances, the *performance* (or *QoS*) *engineering* area of MDSE uses performance, reliability and cost models as key artifacts in all stages of the software lifecycle [11, 12]. Model-driven QoS engineering aims to ensure that software systems satisfy their QoS requirements "by construction" when initially delivered [13], and continue to do so as they *self-adapt* in response to changes in environment, requirements or internal state [14]. QoS models may be developed explicitly or may be synthesised from annotated variants of the structural and behaviour models used in the traditional MDSE process [15], and typically need to be updated continually at run time based on the observed system behaviour [16]. Such efforts can be linked to relevant modelling standards such as the MARTE and QoS profiles for UML [17, 18].

3 A motivating scenario

In the UK, there are organisations responsible for accumulating and managing data related to health and social care. They act as a trusted repository and broker for such information, and also provide statistical expertise and domain knowledge relevant to such data. In particular, they may produce and provide statistical reports on health and social care data to a variety of stakeholders. Stakeholders may include casual browsers of health and social care data – who may simply be interested in learning what information or reports are available – to sophisticated commercial users of data/reports, who may base important commercial decisions on what they acquire from this trusted broker. Additionally, the organisation may be required to provide information to government departments or ministers. As such, information in the form of customised reports and data may be requested by commercial, public or governmental stakeholders, at any time, and the organisation must be able to respond to such requests. In particular:

– QoS requirements may be applicable to requests coming from stakeholders, e.g., a certain data quality, a report available within a certain hard or software deadline.
– Information governance requirements and policies may also be applicable, requiring the health and social care organisation to determine whether prospec-

tive customers are permitted to access either raw data, or pseudonymised data, of a certain type or kind. Checking compliance with information governance policies is particularly time consuming; it would be beneficial to be able to determine if a customer was permitted to access particular data items before continuing with the rest of the procurement process, but sometimes this is not possible – research (see the next point) may need to be carried out in parallel with checking compliance.
– A customer may request an existing type of report, or a report that is similar to one that is currently available. But they may also request reports that are new and novel, for which a *research process* must be carried out. In such a process, the organisation may have to 'buy in' expertise it may not have, may need to investigate new statistical methods or practices, and may need to synchronise the research process with parallel business processes that are in place to ensure proper billing and compliance with governance policies and regulations.

Such an organisation would potentially benefit from the application of different models: for capturing business processes; for capturing QoS properties and requirements; for capturing data and interrelationships; and for capturing information governance rules. Such models could be used for understanding the complexity of the organisation, analysing the effectiveness or potential bottlenecks in a particular stakeholder interaction, and for analysing the effects of *evolution*, e.g., through new QoS requirements or stakeholder requirements. However, the stakeholders interested in the organisation's research, data and results can change at any time, leading to new report/data requirements. As such, the models relevant to the organisation should be considered highly volatile, and may be subject to evolution at any time.

4 Multimodel-driven software engineering

Fast and robust evolution represents a key requirement for a growing number of important enterprise systems. Achieving this requirement needs software engineering technology capable of developing "on demand" information systems that satisfy the overlapping concerns of different stakeholders, and of integrating them into evolving enterprise systems on the fly. We envisage that this role will be played by *multimodel-driven software engineering* (MMSE) – a novel software engineering approach that will combine:

– *multimodel-driven automated code generation*—to take advantage effectively of interdependent models associated with different but overlapping areas of concern, and specified by different classes of stakeholders in distinct and co-evolving domain-specific languages;
– *multimodel-driven QoS engineering*—to ensure that evolving enterprise systems achieve the performance, dependability and cost-related requirements specified across the interdependent models mentioned above.

Significant research is required to provide the theoretical foundation for the MMSE vision and open-standards MMSE tools that realise this theory. Extending the applicability of MDSE to large-scale, evolving software systems whose characteristics spread multiple domains is a hard open problem [4, 19], whose solution involves addressing several major challenges:

1. *Transformations of interdependent models.* Identifying the dependencies among multiple concern-specific models and devising model transformations that comply with these dependencies is notoriously difficult and error prone [4].
2. *Co-evolution of interdependent metamodels.* Managing the co-evolution of heterogeneous sets of interdependent metamodels and the synchronisation of their associated models is a complex problem that is not addressed by existing MDSE approaches [19].
3. *Cross-analysis of interrelated QoS models.* Analysing the relationships and tradeoffs between interacting performance, dependability and cost attributes specified across multiple models is a complex and non-scalable task that is deemed a major challenge for QoS engineering [11].

The research agenda in the next section suggests research objectives that need to be pursued by the software enginnering community in order to tackle these challenges and to realise the vision of multimodel-driven software engineering.

5 MMSE research agenda

Research objective 1. To develop a theoretical foundation comprising formalisms, algorithms, model transformations and techniques for multimodel-driven software development, and for the management and co-evolution of interdependent metamodel and model sets.

Addressing this objective requires the development of new techniques for multi-modelling, including the following multimodel-aware code generators and co-evolution approaches:

1. DSLs for modelling the architecture, business processes and governance rules associated with enterprise information systems. These DSLs must be defined using *generic modelling concepts* (generic in the sense that they can be used throughout the enterprise domain, and derived from existing modelling languages or frameworks such as ArchiMate and TOGAF), wherein metamodels are specified in terms of *required* arguments. This will allow substantial sharing between DSLs.
2. Formalism based on OCL and OCL extensions such as the Epsilon Validation Language [8], for defining *generic inter-dependencies* between the DSLs from (1). These formalisms must allow instantiation of the generic modelling concepts from (1) and support the specification of *strong* (i.e., must-hold) and *weak* (i.e., may-hold) consistency rules on models. These formalisms need to be elaborated to allow such rules to be defined for QoS models.

3. A theory of *generic code generation* for transforming multi-models into platform-specific enterprise information systems application code. The transformations must be capable of instantiating generic parameters and of producing *monitors* to be used to detect whenever QoS properties (not guaranteed to hold via construction) have been violated. The novelty here is that the code generation must take into account QoS models as well as other enterprise systems domain models.
4. A theory of co-evolution for generic multi-models, investigating patterns of generic metamodel change (including changes to parameters), as well as defining strategies for co-evolving generic models, metamodels and interdependencies (from (1) and (2)) and code generators (from (3)).

Research objective 2. To devise a suite of scalable QoS engineering techniques for: (i) co-analysing the relationships between QoS concerns specified through multiple mathematical models; and (ii) identifying effective tradeoffs between conflicting QoS concerns. The real challenge here is to be able to manage QoS models in a structurally identical way to other MDSE models.

Achieving this objective will require the use of a combination of stochastic, Markovian and queueing models to *co-analyse QoS properties* including: (i) dependability (e.g., availability and reliability); (ii) performability (e.g., response time and throughput); and (iii) resource usage (e.g., battery power, data storage capacity, bandwidth and cost). New research results are needed to enable the co-analysis of interdependent QoS attributes specified by heterogeneous mathematical models, in order to identify and exploit effective tradeoffs between the dependability, performability and resource usage requirements of complex software systems. To accomplish this, the research must develop the following new theoretical contributions for the co-analysis of interdependent QoS models:

1. Techniques for the automated co-extraction of stochastic, Markovian and queueing QoS models from enhanced structural models specified using QoS-related UML profiles [20, 17, 18]. These techniques could build on results from [21, 22, 12, 15, 23], extending them with support for extracting new types of QoS models, and with the ability to identify and encode the interdependencies among multiple QoS models.
2. A high-level formalism for specifying (i) the QoS constraints that an information system must comply with at all times; and (ii) the *utility* associated with the achievable trade-offs between the dependability, performability and resource usage of these systems.
3. Algorithms for translating the QoS constraints and utility levels that system developers and operators provide in the high-level formalism from (2) into verifiable low-level properties expressed in temporal logics augmented with probabilities, event rates, costs and rewards.
4. Quantitative analysis techniques for the co-analysis of the formally expressed sets of QoS properties from (3) against the mathematical QoS models from (1). These techniques will need to take into account the dependencies among multiple QoS models, in order to identify trade-offs between performabil-

ity, dependability and resource usage that satisfy the QoS constraints and achieve a high utility.

Research objective 3. To develop an open-standards MMSE platform.

This multimodel-driven software engineering platform needs to include:

1. An integrated toolset supporting multimodel-driven software development, QoS engineering and metamodel/model management. Some of this technology already exists, but extensions to existing model management platforms (such as ATL or Epsilon) need to be made to fully support QoS models, particularly for code generation.
2. A methodology comprising methods for the effective engineering of evolving enterprise systems.

The MMSE platform must integrate, implement and hide the complexity of the formalisms, algorithms and techniques devised by the research described under the research objectives 1 and 2, enabling developers of large-scale evolving software systems to exploit these theoretical results effectively without the need for expert training. Extending an established platform (e.g., the Eclipse platform) for achieving this would speed up the adoption of MMSE considerably.

Research objective 4. To employ MMSE in the development of exemplar evolving enterprise systems.

It is essential that emerging MMSE techniques and tools are evaluated in the development of real-world evolving enterprise systems, as part of joint projects between the research community and organisations whose business processes are supported by such systems. The wide dissemination of the results of these projects, ideally as open-sourced exemplars, is essential to the early adoption of the much needed MMSE technology.

6 MMSE technology delivery in health and social case

We recently embarked on a new project to devise MMSE technology components, and to use them for the development of a prototype evolving enterprise system, in collaboration with a health and social care partner. This will allow us to validate, refine and extend our preliminary version of several MMSE technology components, and to contribute significantly to the efforts to achieve the objectives stated in the UK Health and Social Care Act 2012 [3]. As part of this, our planned MMSE platform will be instantiated with specific types of models relevant to a health and social care industry partner. In particular, we envision building QoS models relevant to assessing the timeliness of treatment or care, information governance models capturing the policies, rules and procedures related to health and social care, etc. This approach is summarised in Fig. 1.

Our research will be carried out while being driven by scenarios and use cases from the health and social care domain. A plausible use case that we have available involves using multimodel-driven software engineering to support *reporting*

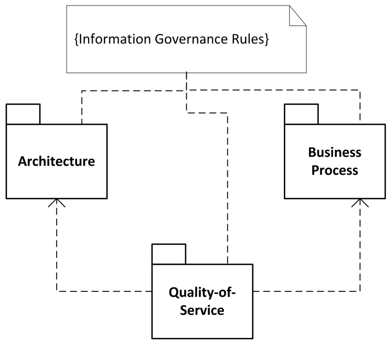


Fig. 1. Conceptual model of instantiation of the MMSE approach

scenarios. Such scenarios involve collecting different types of raw pseudonymised data (e.g., number of incidents of stroke in a particular region) as well as statistical data (e.g., percentage of residents of a certain age being supported by a care provider in a region). Such data may need to be collated and presented in a report for a particular stakeholder group - like a health trust (who may have responsibility for reporting care outcomes to government), or even a government minister.

The data and statistics either gathered or produced for these reports must be audited and validated, produced within strict time bounds, for specific costs (for example, some reports may be produced for a commercial organisation for a fee). The type, quantity and complexity of the data that is being managed, and the reports that are being generated, may change at any time – that is, new IT systems may be brought in to the report-generating organisation (e.g., from new health care providers) and integrated into the report generating process.

Such systems clearly require handling of multiple models and QoS concerns, as well as the need to handle evolution of models. What is particularly challenging about this scenario is that the types of models evolve in different ways and at different rates. Consider Fig. 1: the architecture of such an enterprise system at one of our health and social care partners does not change frequently, but the QoS requirements do (e.g., on a problem-by-problem basis: different customers require their results at different rates), and the information governance rules also change frequently (though not as frequently as QoS models) due to new legislation and Department of Health rules. As such, the multimodel evolution problem is extremely challenging in this context.

7 Conclusions

We have motivated the need for and the challenges of multimodel-driven software development (MMSE), particularly focusing on evolving enterprise systems. We have described a research agenda for developing the MMSE theory and tools required to work with such systems, as well as a motivating scenario in the health and social care domain. Currently, we are developing the MMSE infrastructure for addressing the research objectives of this agenda in the health and social care domain, and are identifying suitable enterprise systems scenarios with two practicing health and social care partners.

References

1. Nigel Shadbolt, Kieron O'Hara, Tim Berners-Lee, Nicholas Gibbins, Hugh Glaser, Wendy Hall, and M. C. Schraefel. Linked open government data: Lessons from data.gov.uk. *IEEE Intelligent Systems*, 27(3):16–24, 2012.
2. UK Cabinet Office. G8 Open Data Charter and Technical Annex, G8 communiqué and documents, June 2013. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/207772/Open_Data_Charter.pdf`.
3. The Health and Social Care Information Centre, Part 9, Chapter 2 of the Health and Social Care Act 2012, 2012. Available at `http://www.legislation.gov.uk/ukpga/2012/7/enacted`.
4. Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *ICSE Workshop Future of Softw. Eng.*, pages 37–54.
5. M. Fowler. *Domain-Specific Languages*. Addison-Wesley, 2010.
6. Dimitrios S. Kolovos, Richard F. Paige, and Fiona Polack. The Epsilon transformation language. In *ICMT 2008*, pages 46–60.
7. Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2008.
8. Dimitrios S. Kolovos, Richard F. Paige, Louis M. Rose, and James R. Williams. Integrated model management with Epsilon. In *ECMFA 2011*, pages 391–392.
9. Louis Rose, Dimitrios Kolovos, Richard Paige, and Fiona Polack. Model migration with Epsilon Flock. In *ICMT*, LNCS 6142. 2010.
10. H. Giese and R. Wagner. From model transformation to incremental bidirectional model synchronization. *Softw. & Syst. Modeling*, 8(1):21–43, 2009.
11. R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM*, 55(9):69–77, 2012.
12. Stephen Gilmore, Laszlo Gonczy, Nora Koch, Philip Mayer, Mirco Tribastone, and Daniel Varro. Non-functional properties in the model-driven development of service-oriented systems. *Softw. & Syst. Modeling*, 10(3):287–311, 2011.
13. Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio component model for model-driven performance prediction. *J. of Systems & Softw.*, 82(1):3 – 22, 2009.
14. Radu Calinescu, Lars Grunske, Marta Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. Dynamic QoS management and optimisation in service-based systems. *IEEE Trans. Software Eng.*, 37(3):387–409, May 2011.
15. M. Tribastone and S. Gilmore. Automatic translation of UML sequence diagrams into PEPA models. In *QEST 2008*, pages 205 –214, 2008.
16. I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by runtime adaptation. In *ICSE 2009*, pages 111–121.
17. Object Management Group. UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms v1.1, 2008.
18. Object Management Group. UML Profile for Modelling and Analysis of Real-Time and Embedded Systems (MARTE) v1.1, 2011.

19. Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John McDermid, and Richard Paige. Large-scale complex IT systems. *Commun. ACM*, 55(7):71–77, 2012.
20. Object Management Group. UML Profile for Schedulability, Performance and Time v1.1, 2005.
21. Radu Calinescu and Marta Kwiatkowska. CADS*: Computer-aided development of self-* systems. In *FASE 2009*, pages 421–424, 2009.
22. M. L. Drago, C. Ghezzi, and R. Mirandola. Towards quality driven exploration of model transformation spaces. In *MoDELS 2011*, pages 2–16.
23. Murray Woodside, Dorina C. Petriu, Dorin B. Petriu, Hui Shen, Toqeer Israr, and Jose Merseguer. Performance by unified model analysis (PUMA). In *5th Intl. Workshop on Software and Performance*, pages 1–12. ACM, 2005.

UML/OCL based Design and Analysis of Role-Based Access Control Policies

Oliver Hofrichter, Martin Gogolla, and Karsten Sohr

University of Bremen, Computer Science Department
Database Systems Group, D-28334 Bremen, Germany
{hofrichter,gogolla,sohr}@informatik.uni-bremen.de
http://www.db.informatik.uni-bremen.de

Abstract. Access control plays an important part in IT systems these days. Specifically Role-Based Access Control (RBAC) has been widely adopted in practice. One of the major challenges within the introduction of RBAC into an organization is the policy definition. Modeling technologies provide support by allowing to design and to validate a policy. In this work we apply a UML and OCL based domain-specific language (DSL) to design and to analyze the access control of the conference management system EasyChair. For the first time EasyChair is formally described in connection with RBAC. Our activities are located on three levels: (a) the re-engineering of the system's access control policy is located at the policy level, (b) the framework level summarizes activities concerning the RBAC metamodel (e.g. enhancements), and (c) at the configuration level, we configure a concrete policy using the conference management system options. As a result, both a DSL developed in previous work is checked for the need of enhancements, and the re-enginered EasyChair access control policy is analyzed. For validation purposes a frequently used UML/OCL validation tool is utilized.

Keywords: Metamodel, RBAC, Policy Analysis, Validation, UML, OCL

1 Introduction

Nowadays large organizations deal with a huge amount of data. Parts thereof have to be protected so that no unauthorized access can occur. At this point, access control comes into play.
Access control regulates *who* can access *what* kind of data under *which* circumstances. Different access control models exist. An access control model defines the concepts available during the creation of an access control policy [19]. An access control policy contains the concrete rules restricting access to objects in an organization. Nowadays the Role-Based Access Control (RBAC) model [16] is frequently used [13]. The idea behind RBAC is to prevent the direct assignment of permissions to users. Instead roles are interposed. In 2004 RBAC was adopted as an ANSI standard [2]. The initial RBAC model is referred to as RBAC$_0$ and comprises the core concepts of roles, permissions, users, sessions and relations

between these concepts. It has been enhanced by additional concepts in further versions of the RBAC model: $RBAC_1$ enhances the core RBAC model by adding role hierarchies; $RBAC_2$ introduces authorization constraints as restrictions on the RBAC functions and relations. RBAC96 cumulates the features of $RBAC_0$, $RBAC_1$ and $RBAC_2$.
Since an organization's access control policy is the basis for the decision whether access to a resource is authorized or not, it is very important that the access control policy meets the organization's security requirements. However, often access control policies are so complex that their rules get opaque. Often the formulation of authorization constraints results in additional properties the policy designer might not be aware of. That is why tool support for the design and for the validation of access control policies is desirable. In [9] a domain-specific language (DSL), based on the Unified Modeling Language (UML) [14] and the Object Constraint Language (OCL) [20], is presented which allows for designing and analyzing RBAC. For validation purposes, the UML-based specification environment (USE) [6] is applied. The organization's security administrator who faces the challenge of designing a policy for a large organization with a huge amount of resources worthy of protection can deploy the RBAC framework and in doing so is enabled to consider security requirements in early stages of software development life-cycle.
In this contribution we analyze the implemented access control policy of the EasyChair conference management and propose few minor modifications of the original RBAC DSL. EasyChair [4] was selected because using roles and role-based access control in the domain of scientific conferences seems obvious and among the conference management systems EasyChair is most commonly used. Since there was no documentation of the access control policy implemented by the EasyChair developers, role engineering had to be conducted in the first step. The EasyChair access control policy was reconstructed by (1) exploring the graphical user interface (GUI) and (2) constructing typical workflows in the work with the EasyChair system. In this way potential roles, accessible objects, access operations and authorization constraints could be mined.
The rest of this paper is organized as follows: In Section 2 the DSL based approach is introduced. In Section 3 the RBAC DSL is deployed to visualize excerpts from the re-enginered EasyChair RBAC policy. Other related work is presented in section 4. Section 5 concludes this work and commands a view on further developments in the future.

2 The Deployed DSL Based Approach

This section firstly introduces the RBAC DSL and the applied validation tool and based on this classifies the activities performed in our contribution.

2.1 RBAC DSL

The RBAC DSL allowing for the design and analysis of RBAC policies and deployed in the following is presented in [9].

It is based on a metamodel, consisting of a UML description of the core RBAC concepts and a set of authorization constraints expressed by OCL invariants. The UML part defining the abstract syntax of the DSL is depicted in Figure 1.

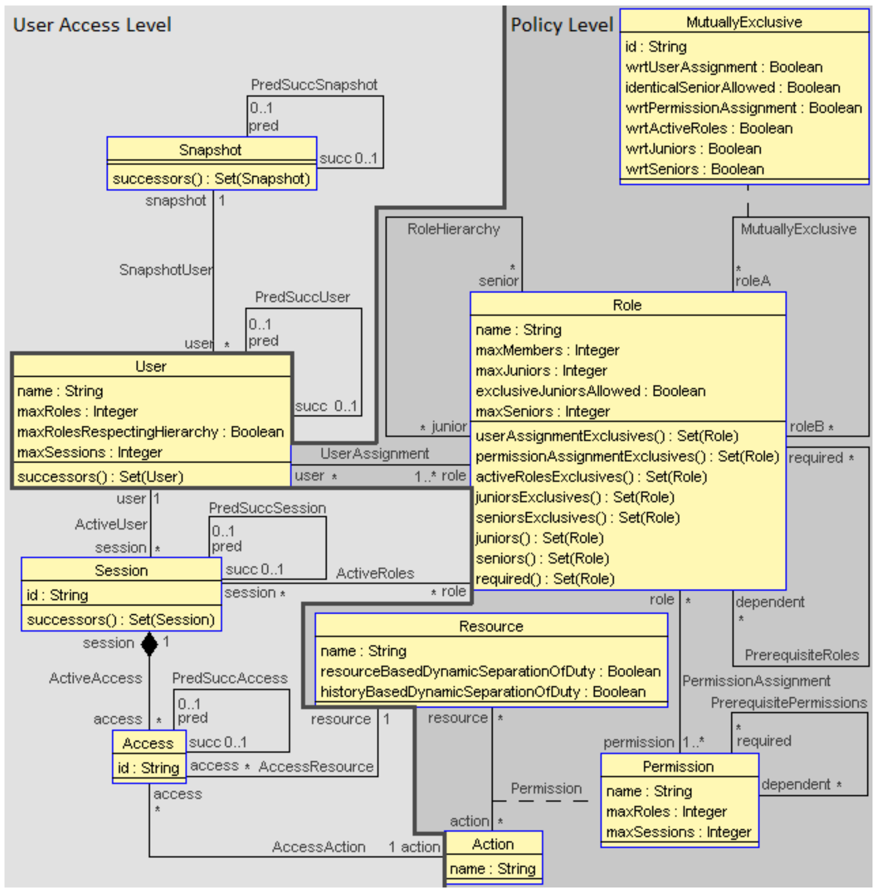


Fig. 1. Structure of RBAC metamodel

A detailed description of the OCL invariants is presented in [7]. The RBAC metamodel differentiates two levels: the dark gray shaded policy level and the light gray user access level. The policy level involves all those elements from the RBAC model the policy designer comes in contact with during the process of the design of a policy. Among these are roles, users, permissions (represented as actions on resources) and the relations between these elements such as user assignment. Similar to the elements of the user access level, these elements are represented as classes and associations. The user access level represents concrete activities of users in the context of an access control policy. The concept of snapshot allows for the specification of dynamic constraints. A concrete access control policy is represented by an instantiation of the RBAC metamodel.

2.2 Validation Tool USE

The RBAC DSL serves as a basis for the validation. The validation is conducted by the UML-based specification environment (USE) [6]. This tool can be used to check if a policy design meets the intuition of a policy designer which is based on organizations' security requirements. In order to validate a security policy the analyst has to generate system states representing the policy. System states are represented as object diagrams. The development can be done by creation and manipulation of objects, attributes and links on a GUI or by writing command files. The created snapshots are compared with the specified RBAC model. If the modeled system states violate the defined constraints the user obtains precise feedback (cf. Figure 5) about the cause for the violation.

2.3 Analyst's Activities

This work is based on previous work [9] and extends it. The use case diagram in Figure 2 classifies the activities performed in the present work in the context of preceding activities. The activities were located on three different levels: (a) the upper part summarizes activities concerning the RBAC metamodel on the framework level. (b) activities concerning the RBAC policy which is an instance of the RBAC metamodel are located below at the policy level. (c) at the bottom of the figure the configuration level is depicted. At this level a concrete access control policy is configured. The analyst activities are promoted in the present paper (highlighted through a bold rectangle in the figure). The analyst activities span over all the three levels as we will describe in the following.
The original RBAC metamodel was developed by the RBAC metamodel development team represented as an actor in the diagram. The remaining involved actors are the EasyChair development team, the system's end-user (conference organization) and the analyst. We assume the EasyChair development team initially chose an access control model [19] at the beginning of the design phase of EasyChair. Based on this decision it defined an access control policy. It is assumed in this contribution, that EasyChair uses the RBAC model [16]. Finally the EasyChair developers defined policy configuration options for the end-users and implemented these and the according access control rules in the EasyChair system. The organizer of a scientific event as the end-user of EasyChair makes choices from the configuration options and customizes EasyChair to the needs of the respective event this way. The analyst's main activities are the policy reengineering, the policy validation and the extension of the RBAC metamodel if necessary. For the policy's definition roles, permissions and authorization constraints had to be mined. In order to mine authorization constraints the analyst also acted on the configuration level.

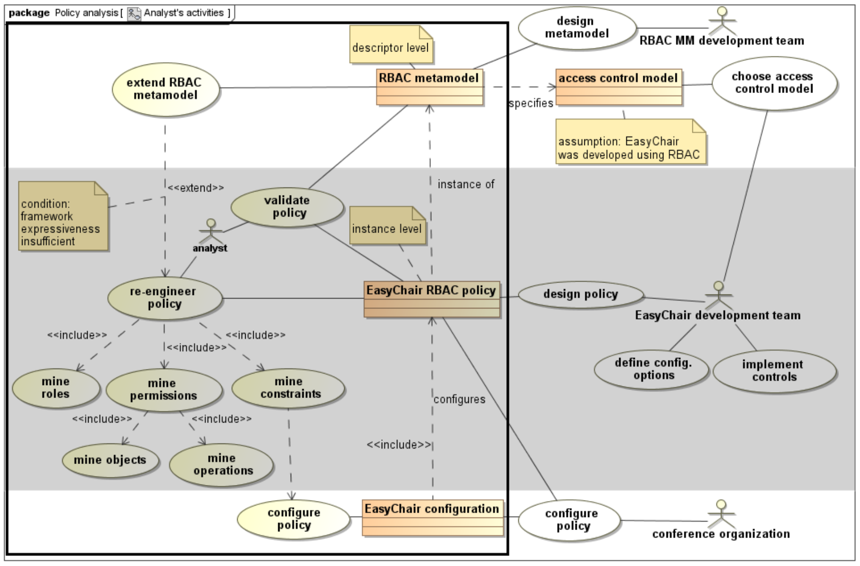


Fig. 2. Classification of analyst's activities

3 EasyChair RBAC Policy

In this section excerpts from the reverse engineered security policy of EasyChair are presented. A more detailed modeling of EasyChair's RBAC policy is presented in [7]. The following modelings serve on the one hand as example visualizations for the reverse engineered access control policy of EasyChair and describe on the other hand possibilities to validate concrete system states against a previously defined security policy. In the following figures, elements from the policy level are shaded in gray.

3.1 Modeling of the Simplest Conference

The UML object diagram in Figure 3 serves as an introductory modeling. The diagram represents the simplest of all possible academic conferences in EasyChair. The diagram depicts three users that respectively access one resource by three different actions one after another. In the first system state a user assigned to role `author` writes a paper. This paper represents the accessible resource. In the following state another user associated with the role `pcMember` reviews this paper. In the final system state a user in the role `chair` decides on the acceptance respectively the refusal of the paper. Security administrators would proceed in the same way to create organization's access control policies: by creating and manipulating objects, links and attributes.

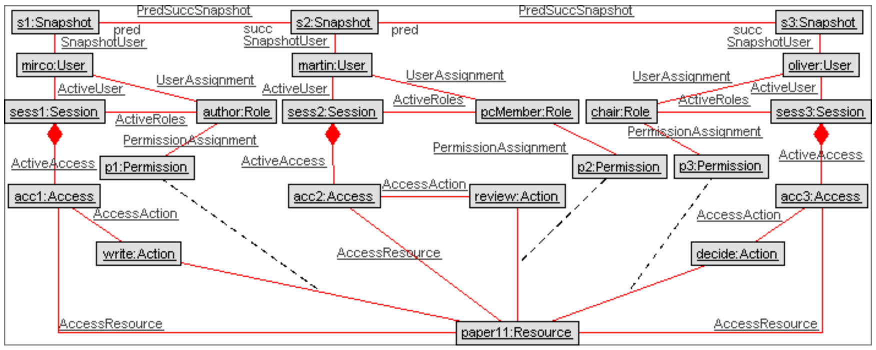
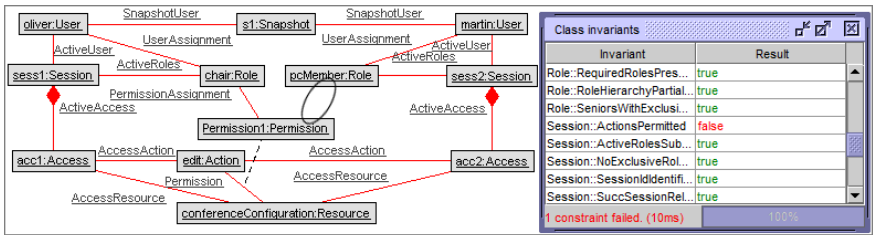


Fig. 3. Representation of the simplest academic conference in EasyChair

3.2 Policy Validation Example: Missing Permission

The previous modeling served as a visualization of an excerpt from the policy. In the following sections examples of possibilities to analyze the policy are presented.
In Figure 4, a concrete situation from the EasyChair system is depicted together with the Class invariants view of the USE tool. The situation was manually constructed on the basis of a concrete EasyChair configuration.
The policy specifies a permission for editing the administration area of EasyChair. The so-called conference configuration is represented as a resource. The user depicted in the left part of Figure 4 is associated to the role `chair`. Since this role is associated with the necessary permission for editing the conference configuration, no constraint is violated. The right part of the figure shows the access of another user to the protected resource. This user is member of the role pcMember. This role is not equipped with the needed permission. Therefore the validation tool reports on a violation of an OCL constraint depicted in Listing 1.1.



Invariant	Result
Role::RequiredRolesPres...	true
Role::RoleHierarchyPartial...	true
Role::SeniorsWithExclusi...	true
Session::ActionsPermitted	false
Session::ActiveRolesSub...	true
Session::NoExclusiveRol...	true
Session::SessionIdIdentifi...	true
Session::SuccSessionRel...	true

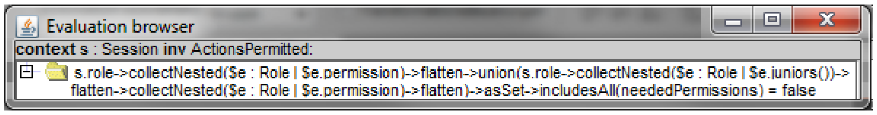
1 constraint failed. (10ms) 100%

Fig. 4. Policy validation example: missing permission

Listing 1.1. Invariant `ActionsPermitted`

```
context s:Session
  inv ActionsPermitted:
    s.access->forAll(a |
      let neededPermissions = a.action.permission
          ->select(p | p.resource = a.resource) in
      neededPermissions->notEmpty() and
      s.role.permission->union(s.role.juniors().permission)->asSet()
        ->includesAll(neededPermissions))
```

The invariant `ActionsPermitted` serves as an example for the realization of OCL restrictions on the RBAC functions and relations supplementing the UML description of the RBAC concepts. The invariant in Listing 1.1 is defined in the context of an RBAC session. A session is associated with users, roles and accesses. It activates a user's membership in a role and the concrete access to a resource. As depicted in the RBAC metamodel in Figure 1, any number of accesses can be made by a user activated in a session. The invariant specifies for all the accesses that the activated roles have to be equipped with all the permissions that are needed for the execution of the respective operation (represented as action in the RBAC metamodel) on the regarded object (represented as resource in the RBAC metamodel).
Figure 5 shows an excerpt from the Evaluation Browser view in the USE tool. It enables a policy designer to identify the position where the constraint is violated. The excerpt shows that a role exists that is not equipped with all the needed permissions.



```
context s : Session inv ActionsPermitted:
s.role->collectNested($e : Role | $e.permission)->flatten->union(s.role->collectNested($e : Role | $e.juniors())->
flatten->collectNested($e : Role | $e.permission)->flatten)->asSet->includesAll(neededPermissions) = false
```

Fig. 5. Feedback concerning constraint violation

3.3 Policy Validation Example: Handling of Dynamic Constraints

Whilst the preceding modeling investigated a restriction on the static level, the following modeling serves as an example for the handling of dynamic constraints. The aim is to model the following condition: a PC member is only allowed to read the other PC member's reviews if she has already submitted her own review.
In Figure 6 a state from the EasyChair system that violates this restriction is depicted. The object diagram consists of two snapshots. In both of the snapshots a user associated to the role pcMember is regarded. In the second snapshot the regarded user accesses the other PC member's reviews. This is only permitted if there is an access to the resource through the write action first. However, this access is missing here. So the validation tool again reports on a violation of an OCL constraint responsible for the realization of this authorization constraint.

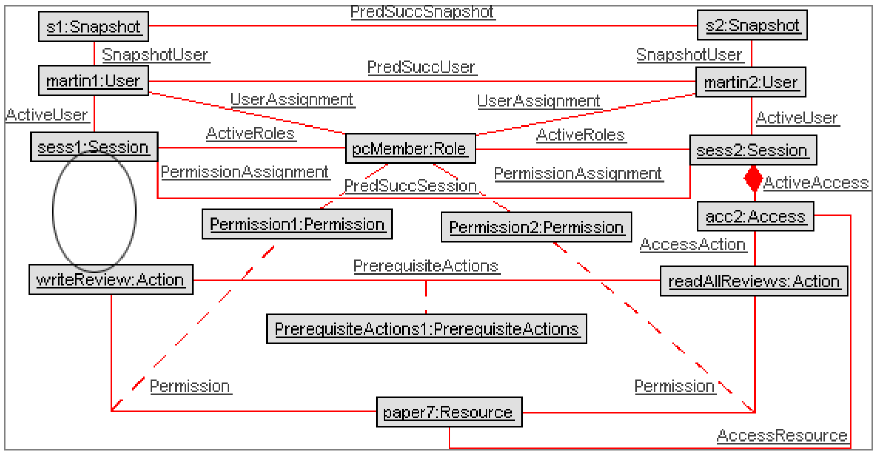


Fig. 6. Policy validation example: handling of dynamic constraints

3.4 Metamodel Extension

The extensibility of the RBAC metamodel is shown in [17]: the metamodel is extended by support of delegation and revocation concepts. In some places in this contribution the modeling of the re-engineered EasyChair RBAC policy also involved extensions to the original RBAC metamodel [9]. A detailed description of the extensions is presented in [7]. For example the metamodel was extended to prevent that reviewers have the same affiliation as authors. For the realization the classes `User` and `Action` and the association class `MutuallyExclusiveActions` were expanded. On top of this an additional invariant was introduced. The class `User` was expanded by the attribute `affiliation`. The class `Action` was expanded by an operation for identifying mutually exclusive Actions regarding the attribute `wrtAffiliation` introduced to `MutuallyExclusiveActions`. These ingredients are brought together by the OCL invariant `checkAffiliation` defined in the context of the class `Resource`.
In the same way the RBAC metamodel can be extended if an organization's policy designer is confronted with situations that can be expressed by the existing metamodel.

4 Related Work

A classification of RBAC related publications since the adoption of role theory in information security [16] is presented in [5]. Three major classes of RBAC research were identified in [15]. In the present paper, a hybrid approach for the definition of roles was applied. Other approaches are top-down [12] and bottom-up (also labeled as role mining). The RBAC framework consists of a family of models [16]. Each family member represents an access control model. In the present contribution a concrete access control policy is designed and analyzed. The relationship between access control models and access control policies is described in [19]. Interoperability problems of access control policies are addressed by the standard access control policy language XACML [21]. Juerjens combined Model-driven development and security by integrating security related information in UML [8]. Different approaches for modeling RBAC properties in UML/OCL exist. In [10] a UML profile for a modeling environment is presented. The approach uses (customized) class diagrams and activity diagrams. For verification of RBAC properties, OCL is applied. In [18] UML is also used to describe security policies. Contrary to our approach the UML models are transformed to Alloy for analysis purposes. An access control policy analysis approach for mobile applications using the UPPAAL model checker is presented in [1]. A survey of Model-driven security offers [3].

5 Conclusion and Future Work

In this contribution, we have successfully applied a UML and OCL based DSL for the investigation of a concrete access control policy. In doing so, the DSL itself was validated and described how the DSL in combination with a UML and OCL validation tool can be employed to discover and eliminate undesired policy properties which do not meet the security requirements.
The policy design and validation were carried out by the same group of persons. In order to validate concrete system states against a previously defined policy, a separation of the responsibilities as in software quality management [11] is of prime importance. Since the conducted role mining based on the exploration of the GUI, simulation of concrete work steps in EasyChair and interviews with domain experts the policy designed for investigation already based on findings of the investigation. This is why in this work the RBAC DSL was primary applied to formalize and visualize the reengineered RBAC policy of EasyChair.
For the future, an automatic transformation from concrete RBAC policies into concrete syntax used by RBAC DSL (USE) would be desirable. To harmonize the different application-specific access control policy languages the language XACML was developed. A further step could be to develop a transformation of the XACML representation of a policy to the USE syntax. The other direction, namely the transformation from RBAC policy formulated in RBAC DSL into a representation supported by concrete target systems like Tivoli or DirXMeta-Role, could be of interest as well.
Moreover, the usability of the applied approach has to be improved. For example at the moment the policy level and the user access level have to be modeled in the same diagram. Separate diagrams and syntactic representations of (our DSL) modeling elements would improve usability within an enterprise.
Another interesting aspect that has to be considered in future work is the evolution of the RBAC metamodel. This comprises among other things the development of a concept for handling different metamodel versions and both domain-specific metamodel changes and ones concerning access control model concepts.

References

1. Abdunabi, R., Ray, I., France, R.B.: Specification and analysis of access control policies for mobile applications. In: Conti, M., Vaidya, J., Schaad, A. (eds.) SACMAT. pp. 173–184. ACM (2013)
2. ANSI: Role Based Access Control (2004), ANSI/INCITS 359-2004
3. Basin, D.A., Clavel, M., Egea, M.: A decade of model-driven security. In: Breu, R., Crampton, J., Lobo, J. (eds.) SACMAT. pp. 1–10. ACM (2011)
4. EasyChair Conference System. Internet, `http://www.easychair.org/`
5. Fuchs, L., Pernul, G., Sandhu, R.S.: Roles in information security - A survey and classification of the research area. Computers & Security 30(8), 748–769 (2011)
6. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-Based Specification Environment for Validating UML and OCL. SCP 69, 27–34 (2007)
7. Hofrichter, O.: Analyse und Modellierung der rollenbasierten Zugriffskontrolle für ein IT-System zur Verwaltung wissenschaftlicher Konferenzen. Master's thesis, University of Bremen (2012)
8. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development. In: Jézéquel, J.M., Hußmann, H., Cook, S. (eds.) UML. LNCS, vol. 2460, pp. 412–425. Springer (2002)
9. Kuhlmann, M., Sohr, K., Gogolla, M.: Comprehensive Two-Level Analysis of Static and Dynamic RBAC Constraints with UML and OCL . In: Baik, J., Massacci, F., Zulkernine, M. (eds.) Proc. SSIRI. pp. 108–117. IEEE (2011)
10. Montrieux, L., Wermelinger, M., Yu, Y.: Tool support for UML-based specification and verification of role-based access control properties. In: Gyimóthy, T., Zeller, A. (eds.) SIGSOFT FSE. pp. 456–459. ACM (2011)
11. Myers, G.J.: Art of Software Testing. John Wiley & Sons, Inc., New York, NY, USA (1979)
12. Neumann, G., Strembeck, M.: A scenario-driven role engineering process for functional RBAC roles. In: SACMAT. pp. 33–42 (2002)
13. O'Connor, A.C., Loomis, R.J.: 2010 Economic Analysis of Role-Based Access Control. Tech. Rep. RTI Project Number 0211876, NIST (2010)
14. OMG (ed.): UML Superstructure 2.4.1. OMG (Aug 2011)
15. Sandhu, R.S.: Future Directions in Role-Based Access Control Models. In: Gorodetski, V.I., Skormin, V.A., Popyack, L.J. (eds.) MMM-ACNS. LNCS, vol. 2052, pp. 22–26. Springer (2001)
16. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-Based Access Control Models. IEEE Computer 29(2), 38–47 (1996)
17. Sohr, K., Kuhlmann, M., Gogolla, M., Hu, H., Ahn, G.J.: Comprehensive Two-Level Analysis of Role-Based Delegation and Revocation Policies with UML and OCL. Information and Software Technology 54(12), 1396–1417 (2012)
18. Sun, W., France, R.B., Ray, I.: Rigorous Analysis of UML Access Control Policy Models. In: POLICY. pp. 9–16. IEEE Computer Society (2011)
19. di Vimercati, S.D.C.: Access Control Policies, Models, and Mechanisms. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security (2nd Ed.), pp. 13–14. Springer (2011)
20. Warmer, J., Kleppe, A.: The Object Constraint Language: Getting Your Models Ready for MDA. Object Technology Series, Addison-Wesley, Reading/MA (2003)
21. OASIS XACML. Internet, `http://docs.oasis-open.org/xacml/`

Meta-model Patterns for Expressing Relationships Between Organization Model Concepts and Software Implementation Concepts

Jens Gulden

Information Systems and Enterprise Modeling
University of Duisburg-Essen
Universitätsstr. 9
45141 Essen, Germany
jens.gulden@uni-due.de

Abstract. The abstraction gap between organization models and models of software artifacts is of fundamental ontological nature, and bridging this gap cannot be achieved with solutions located either on the technological abstraction level or on the conceptual level separately.
The work presented in this article describes a meta-model based approach to explicate design decisions on how to map conceptual organizational model constructs to software implementation specifications, from which software for supporting an organization's work can subsequently be developed or generated. With the described meta-model patterns, one methodical component of a development method is made available to systematically guide the development of enterprise software systems, based on knowledge given in organization models.

Keywords: Organization Modeling, Software Development, Business Process Model Implementation, Meta-modeling

1 Aligning organization models and software implementation

One central research goal in information systems science is to achieve an alignment between conceptualized enterprise models (EMs) and the enterprise systems (ESs) that are used to support their realization [7, 10]. With the use of information technology (IT) systems as supporting units in organizations, this task also covers the behavior of software, and it becomes a managerial task to make sure that software systems in organizations operate in alignment with their business purpose [5].

From this constellation, a dilemma arises in managing organizations. On the one hand, it is an inherent managerial task to align the ideas and conceptualizations of strategic goals with the real actions going on in an organization. On the other hand, once software gets involved, a high degree of technical expertise is required to understand the operation of software, or even to develop software according to intended managerial conceptualizations.

Traditionally, there is a methodical gap between describing organizational structures and processes on the one hand, and software components and functionality on the other hand, because organizations and software systems are understood and constructed with different terminology and on different levels of abstractions, typically also by differently educated groups of people.

In EMs, dedicated modeling language elements are used to express knowledge about structures and processes in organizations, e. g., about who is responsible for performing actions, involved resources, and strategic goals intended to be realized by organizational means. With the use of EMs, a chance opens up to closer involve the users of software systems into the process of developing and configuring software. Building software from enterprise models is desirable, because once a dedicated relationship between enterprise models and software functionality has been established by a development method, involved users and responsible stakeholders can adapt the software according to their business needs, without having to deal with programming or technical details.

This article investigates the research question, how design decisions made during a development process from conceptual organization models to ESs can be formally captured in a model structure, and thus be made available to further automatic evaluation, e. g., to code generation mechanism or runtime interpreters. This is done by elaborating a set of meta-model patterns, which allow to instantiate model instances that carry knowledge about how conceptual model elements are understood in technical terms. Incorporating these patterns as parts of meta-models of intermediate models used during the software development process provides one possible solution for capturing design decisions in the desired way, and further make use of this formalized knowledge in a partially automatized software development process.

The following section takes a look at existing research that has covered comparable work or is located in the same area of research. Section 3 presents the introduced meta-modeling patterns and sketches some methodical steps, which would embed the use of these patterns into an overall software development method to get from organizational EMs to supporting ESs. In section 4, an example of applying the proposed approach in a prototypical ES development setting is presented. The final section 5 summarizes the presented work and takes a look at how the suggestions can further be integrated with other methodological research in the field of organization modeling.

2 Related work

A number of research questions are addressed when enterprise models are consulted for deriving executable software, especially when business process models are to be interpreted as executable workflow models.

In [15], a method is suggested to convert models in the Business Process Modeling Notation (BPMN) to executable Business Process Execution Language

(BPEL) workflows. Other process modeling languages are not looked at, neither are other enterprise perspectives, such as organization models. The method is limited to generate BPEL models, which are to be manually revised by software developers.

Another approach for "bridging the gap between business models and workflow specifications" is discussed in [2]. The central idea of the proposed procedure is to methodically guide human modelers, i. e., domain stakeholders, architects and developers, through a process of human modeling actions to transform a given conceptual business process model to an executable workflow model. The methodical procedure is designed in a way to ensure that the resulting workflow model fulfills the criterion of the soundness meta-property.

In [1], an approach is suggested, which explicates relationships between conceptual elements in business process models, and workflow elements, through an individual type of model, called the Business-IT Mapping Model (BIMM). The approach appears like a specialization of the one presented here, since the general notion of an explicit mapping between business-level model concepts and implementation concepts using a mapping model is a building block in this work. The approach, however, is not generalized to map to arbitrary variants of target architecture platforms expressed via implementation strategy meta-models.

Enterprise models comprise more than business process models only, such as actor and resource models, business rule models, or goal models. This is taken into account by [17], in which a general methodical approach is suggested for developing software from EMs. The approach uses a specifically adapted conceptual modeling language to capture enterprise knowledge. Additionally, several link types are introduced, instances of which can reference from elements of the conceptual model to elements of implementation-level modeling languages. Implementation-level elements are not further described by the proposed approach, it seems to be inherently assumed that existing modeling techniques for technical artifacts can directly be applied for this task.

[12, 16] discuss a number of conceptual mismatches between BPMN [8] and BPEL [11, 13], which in the first place is BPMN's flow oriented process models, versus BPEL's block-oriented approach. A flow-oriented way of modeling processes makes use of interconnecting sequence elements between individual process-members (i. e., between process-steps and events, if applicable). In contrast to the flow approach, a block-oriented way of expressing sequence-flows makes use of specific language constructs, which determine, in what way inner elements of the block are executed, e. g., `If`-blocks to express conditions, `While`-blocks to form loops, or `Flow`-blocks to indicate parallelism.

Development methods, which consult models for expressing different layers of system abstraction in a software development process, can generally be subsumed under the term Model-Driven Architecture (MDA) [14]. Although MDA approaches make use of the notions of computation independent models (CIMs), platform independent models (PIMs), and platform specific models (PSMs), they only consider isolated models on each of these abstraction layers, without interlinking constructs that capture the design decisions leading from one level to the other. The meta-model patterns presented in the work at hand provide orthogonal modeling constructs which fulfil this task, and can potentially be used in combination with the standard model types suggested by MDA.

3 Meta-model patterns for bridging between different levels of abstraction

Methodical means for performing the required bridging between abstract domain-specific enterprise model concepts on the one hand, and technically concrete constructs describing desired target output artifacts on the other hand, can be offered by explicit language constructs in a dedicated mapping model language. A mapping model entry is a modeling construct, which allows to formally express how conceptual elements from the enterprise models are interpreted in technical terms. The use of such a construct as a central part of the development procedure allows for a controlled bridging between both levels of abstraction.

Traditional business conceptualizations regard ESs as a kind of IT resources that are involved when performing specific processes [3]. However, this conceptualization does not allow for understanding ESs as a kind of formal representation of parts of the organization itself. Since ESs are actively acting automatic entities inside the organization, these entities necessarily encapsulate formal knowledge about the organizational action system and the process contexts they are applied in. In this sense, ESs are more than production resources to foster efficient process execution. They both reflect and shape the processes they are involved in.

As a consequence, in descriptions of organizations' action systems, there is an internal connection between the action system, and the ESs that occur as part of these descriptions. Whenever an ES is incorporated in the description of an organization's action system, it can be inherently assumed that the ES contains formal internal descriptions of selected aspects of the action system, too, since otherwise the software could not successfully contribute to the processes it is intended to support. This connection makes it attractive to reason about a software development approach which interconnects both EMs and ESs, as it is carried out in this work, and justifies the assumption that it is possible to derive formal software system descriptions from organization models using a defined engineering method.

Implementation strategies represent formalized descriptions of technical design decisions about the desired software system to be developed or configured, on a computation dependent, yet platform independent, level. They serve as bridge concepts between the interpretation of enterprise models, and technical realizations of software artifacts. With the notion of implementation strategies, a group of model elements is introduced into the software development process, which can systematically capture design decisions made during system development. Which implementation strategies to apply, is either decided by human software architects and developers, or automatic rules can be formulated before-hand, which allow the automatic association of implementation strategies with enterprise model concepts.

Implementation strategies can additionally be enriched by human-readable descriptions of the design rationales behind the chosen decision, which offers an additional level of documentation and justification of design decisions taken to build an overall system.

After implementation strategies are specified and referenced from a mapping model, code generation templates can be used to transform the chosen implementation strategies to software artifacts. Fig. 1 sketches the idea behind a mapping model entry relating implementation strategies on the right-hand-side to conceptual model elements of the left-hand-side.

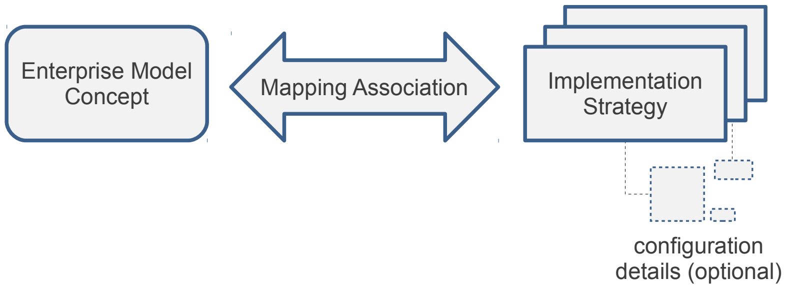


Fig. 1: Pattern of a single mapping association

Examples of implementation strategies used to describe the implementation of web applications are shown in the meta-model excerpt in Fig. 4. They describe dedicated technological means available in a web application setting, without already specifying implementation details on how the technology is realized.

The implementation strategy concept provides an abstraction over technological artifacts, while not being concerned with the actual implementation of these artifacts. This way, it offers an adequate means of abstraction to serve the purpose of a linking concept between interpreted domain-specific concepts in enterprise models on the one hand, and design decisions for their technical realizations on the other hand.

A formal meta-model representation of the mapping between a process step element in a business process, and one or more assigned process step implementation strategies, is shown in Fig. 2 a). The abstract meta-class `ProcessMember` on the left-hand-side represents a business process step specified in a conceptual business process model. `AbstractProcessMemberImplementation` on the right-hand-side is a place-holder for any possible concrete implementation strategy that can be decided to be applied to the given business process element, depending on technical capabilities available for the ESs to be created or configured. The meta-class `ProcessMemberMapping` in the middle represents the type of a binding element, which declares an instance of the specified mapping when design decisions are captured in models of this meta-model pattern.

Reasonable mapping structures for capturing knowledge about how to bridge between different perspectives and levels of abstraction need not simply consist of a one-to-many mapping from an organization model element to implementation strategies. Instead, the meta-model patterns suggested here cover specific semantic aspects of different conceptual elements in organization models. As a consequence, the mapping of process sequence steps, which interconnect individual process steps in business process models, resolves to specifying three independent dimensions of what it means to proceed a step further in a process. From a conceptual point of view, sequences may lead across boundaries of actor responsibilities, resources and spatial or timely distribution. To provide sufficient design decision knowledge about the implementation of sequence concepts, both aspects of either passing the control flow to a different actor, and/or passing the control flow to another spatially distributed system responsible for performing the next process-step now or later, have to be taken into account. A third orthogonal dimension is the handling of conditions, under which sequence steps are taken or ignored.

These three dimensions of sequence implementations are represented by the corresponding meta-classes `AbstractActorResolverImplementation`, `AbstractControlFlowImplementation`, and `AbstractConditionImplementation` in the meta-model pattern. Fig. 2 b) shows the corresponding meta-model excerpt of this example.

Other meta-model patterns for mapping actor concepts, resource concepts, and other types of enterprise model elements, can be constructed accordingly.

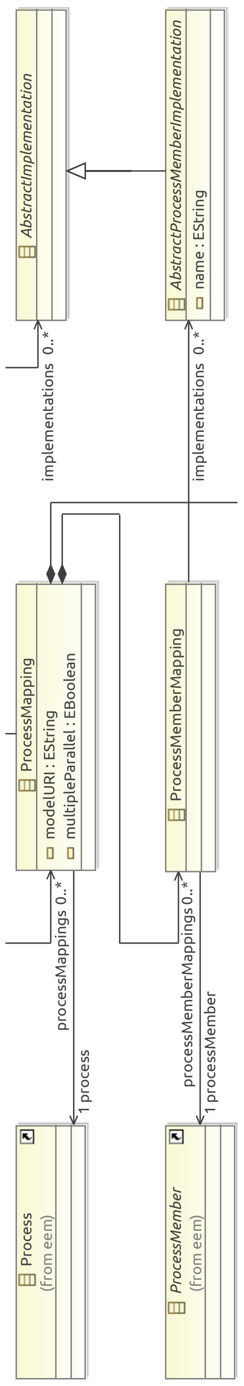
The combined use of a mapping model and implementation strategies provides dedicated methodical abstractions for coping with the requirements to bridge the abstraction gaps between conceptual enterprise model specifications, and ES implementations.

4 Example application

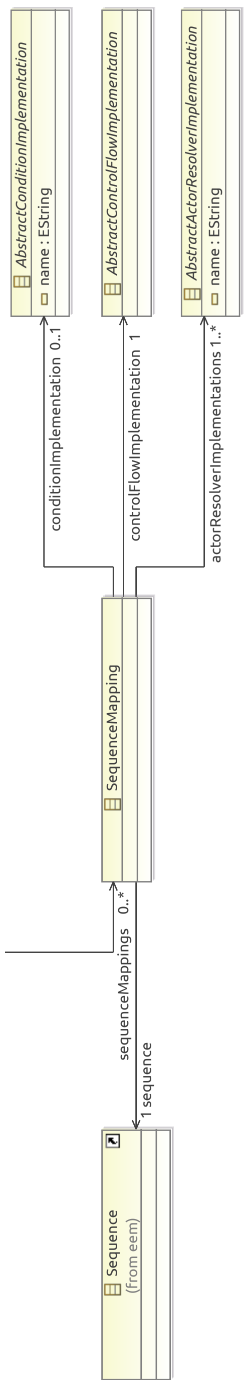
This section introduces a simple web shop example to demonstrate the use of the proposed approach. The example uses enterprise models in the MEMO [4] language as conceptual models describing the socio-technical environment of the software to be generated. The application architecture resembles a traditional web application environment, with web-server and web-client running on physically remote machines, communicating through the internet via the Hyper-Text Transfer Protocol (HTTP).

Fig. 3 shows an excerpt from the MEMO process control flow model in the example, in which organizational roles and resources from other perspectives are referenced.

The meta-classes suggested by the web implementation strategy meta-model are shown in Fig. 4, and described in the following. To enrich the set of available event implementation strategies, the `EventLinkHasBeenFollowed` meta-



(a) Process mapping pattern


AbstractConditionImplementation
name : EString
AbstractControlFlowImplementation
AbstractActorResolverImplementation
name : EString
conditionImplementation 0..1
controlFlowImplementation 1
actorResolverImplementations 1..*
SequenceMapping
sequenceMappings 0..*
1 sequence
Sequence
(from eem)

(b) Sequence mapping pattern

Fig. 2: Excerpts of the mapping meta-model showing the process mapping pattern and the sequence mapping pattern

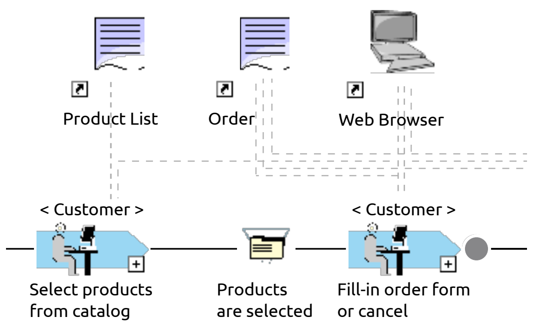


Fig. 3: Excerpt from a MEMO process control flow model referencing elements from other organization model perspectives

class has been included in the meta-model as a subclass of the mapping meta-model's abstract meta-class `ArchitectureSpecificEventImplementation`. It allows to describe that an ES reacts on user actions on a web page.

To resolve concrete users that fulfill an actor role, the `WebSessionUser` meta-class is part of the meta-model. It subclasses the abstract meta-class `ArchitectureSpecificActorResolverImplementation` from the mapping meta-model, and allows to describe an additional actor resolver implementation strategy, implemented e. g. based on session ids. Session ids are a concept specifically available on the underlying technological platform of web applications.

5 Conclusion and future work

The presented approach forms one building block of an overall development method for creating ESs from EMs, which is described elsewhere [6]. The described model types, the mapping model, and one or more implementation strategy models, can be integrated into various possible procedures for deriving software from conceptual models. They offer a general construct for explicating understanding of two distinct conceptual domains, and corresponding interrelationships expressed for the purpose of deriving software functionality from organization models.

One proposal of such an overall development method has been made in [6]. Further methodical integrations are subject to future research, possibly the proposed approach can be used within existing methodical frameworks, such as the Rational Unified Process (RUP) [9].

Applications for the proposed meta-modeling patterns other than software development are possible. When examining the possible range of mappings that can be constructed between organization models and software models, research on the semantics of conceptual modeling languages is inherently part of the evaluation. In combination with this work, further theoretical insight can be gained into the expressiveness of organization and enterprise modeling languages, which can result in scientifically justified suggestions for improving future organization modeling languages.

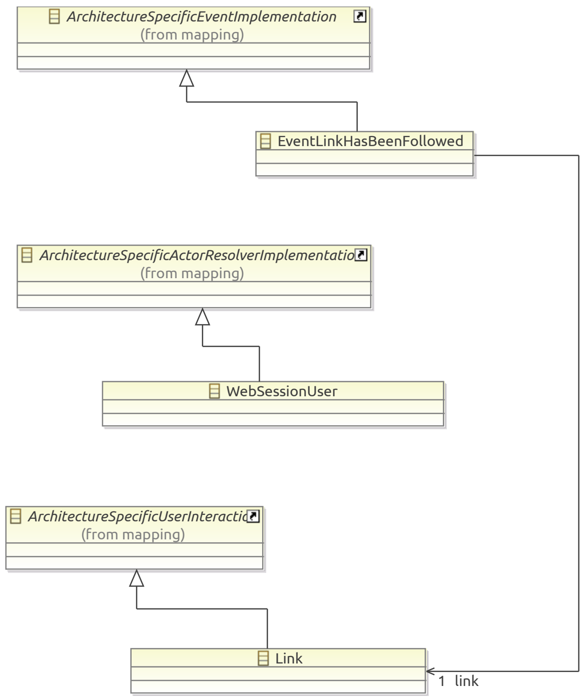

ArchitectureSpecificEventImplementation
(from mapping)
EventLinkHasBeenFollowed
ArchitectureSpecificActorResolverImplementatio
(from mapping)
WebSessionUser
ArchitectureSpecificUserInteractic
(from mapping)
Link
1 link

Fig. 4: Example implementation strategy meta-model excerpt for a web application architecture

References

1. Stephan Buchwald, Thomas Bauer, and Manfred Reichert. *Bridging the Gap Between Business Process Models and Service Composition Specifications*, pages 124–153. IGI Global, Hershey, 2011.
2. Juliane Dehnert and Wil M. P. van der Aalst. Bridging the gap between business models and workflow specifications. *International Journal of Cooperative Information Systems*, 13(3):289–323, 2004.
3. Joaquim Filipe and José Cordiero, editors. *Enterprise Information Systems*, Berlin/Heidelberg, 2008. Springer. 10th International Conference ICEIS 2008, Barcelona, Spain, June 2008.
4. Ulrich Frank. Multi-perspective enterprise modelling: Background and terminological foundation. Technical Report 46, ICB Institute for Computer Science and Business Information Systems, University of Duisburg-Essen, Essen, December 2011.

5. Wim Van Grembergen and Steven De Haes. *Enterprise Governance of IT: Achieving Strategic Alignment and Value*. Springer, New York, 2009.
6. Jens Gulden. *Methodical Support for Model-Driven Software Engineering with Enterprise Models*. Logos, Berlin, 2013. PhD thesis.
7. John C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1):4–16, 1993.
8. Business Process Management Initiative. Business process modeling notation 2.0 (bpmn 2.0), 2011.
9. Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley, Upper Saddle River, 3rd edition, 2003.
10. Martin Op't Land, Erik Proper, Maarten Waage, Jeroen Cloo, and Claudia Steghuis. *Enterprise Architecture*. Springer, Berlin Heidelberg, 2009.
11. Jan Mendling. Business process execution language for web service (bpel). *EMISA Forum*, 26(2):5–8, 2006.
12. Jan Mendling, Kristian Bisgaard Lassen, and Uwe Zdun. On the transformation of control flow between block-oriented and graph-oriented process modeling languages. *International Journal of Business Process Integration and Management (IJBPIM). Special Issue on Model-Driven Engineering of Executable Business Process Models*, 3(2):96–108, 2008.
13. OASIS Web Services Business Process Execution Language (WSBPEL) Technical Committee. Web services business process execution language version 2.0, 2007. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.
14. Object Management Group. Mda guide version 1.0.1, 2003. `http://www.omg.org/mda`.
15. Chung Ouyang, Marlon Dumas, Wil M. P. van der Aalst, Arthur H. ter Hofstede, and Jan Mendling. From business process models to process-oriented software systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(1):1–37, 2009.
16. Jan Recker and Jan Mendling. On the translation between bpmn and bpel: Conceptual mismatch between process modeling languages. In Thibaud Latour and Michael Petit, editors, *CAiSE 2006 Workshop Proceedings - Eleventh International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2006)*, pages 521–532, 2006.
17. Iyad Zikra, Janis Stirna, and Jelena Zdravkovic. Bringing enterprise modeling closer to model-driven development. In *The Practice of Enterprise Modeling, 4th IFIP WG 8.1 Working Conference, PoEM 2011 Oslo, Norway, November 2-3, 2011 Proceedings*, volume 92 of *Lecture Notes in Business Information Processing*, pages 268–282. Springer, 2011.

MDE Support for Enterprise Architecture in an Industrial Context: the TEAP Framework Experience

Hugo Bruneliere[1], Jordi Cabot[1], Stéphane Drapeau[2], Flavien Somda[3], William Piers[2], Juan David Villa Calle[1], Jean-Christophe Lafaurie[3]

[1]AtlanMod Team (Inria, Mines Nantes & LINA), Ecole des Mines de Nantes, 4 rue Alfred Kastler, 44307 Nantes, France
{hugo.bruneliere,jordi.cabot, juan-david.villa_calle}@inria.fr
[2]Obeo, 7 boulevard Ampère, Espace Performance La Fleuriaye, 44481 Carquefou, France
{stephane.drapeau,william.piers}@obeo.fr
[3]Capgemini, 16 mail Pablo Picasso – CS 81515, 44015 Nantes, France
{flavien.somda,jean-christophe.lafaurie}@capgemini.com

Abstract. Model Driven Engineering (MDE) is often applied to support software engineering processes (i.e., from reverse to forward engineering, including maintenance and/or evolution tasks). However, as promoted by the Model Driven Organization (MDO) initiative, it can also be relevant in more business-oriented and strategic decision-making activities such as Enterprise Architecture (EA). EA is the process of translating business vision and strategy into effective change by better describing the enterprise's future state and thus enable its evolution. Even if several approaches have already proposed different kinds of support to deal with the company's EA, an integrated MDE framework combining EA data federation, EA standard adaptation and multiple viewpoint support is still missing. This paper reports on our ongoing experience of building the TEAP MDE framework (based on the TOGAF standard and SmartEA tooling) notably addressing these three challenges in an industrial EA context.

Keywords: MDE, EA, Federation, Adaptation, Traceability, View/viewpoint.

1 Introduction and Motivation

Model Driven Engineering (MDE) has already been largely applied in the general context of supporting software engineering processes (concerning both forward and reverse engineering) or when dealing with interoperability problems (e.g., data exchange, component adaptation) between different systems, environments, tools, etc. More recently, the so-called Model Driven Organization (MDO) initiative has been showing that (business-) strategic or decisional levels within companies, administrations, etc. could also benefit similarly from the application of MDE.

In this area, Enterprise Architecture (EA) [6] implies the effective representation and manipulation of many different aspects of an organization, such as notably its information system as well as depending services and people. There have been different initiatives during the last 30 years aiming to provide a unified EA representation framework, from the widely known Zachman Framework [21] to the U.S. DoDAF [9], British MODAF [15], Open Group ArchiMate [8] and currently the Open Group TOGAF standard [18]. However, fully and efficiently coping with EA is a real challenge [7] despite of the existing tools [14]. Thus *Modeling*, in the very large sense of dealing with *representations of reality*, has already been proposed as a possible solution in the EA context [5] although real effective applications of MDE have been much rarer. Among them we can cite LEAP [3], which provides a light and generic EA framework and language aiming notably at facilitating the analysis of EA representations (models) via their execution/simulation.

Complementary to this initiative, the main objective of our TOGAF Enterprise Architecture Platform (TEAP) collaborative project [19] is to provide (benefiting from MDE capabilities) a lightweight support to other standard industrial EA activities, more particularly to EA governance and decisional processes as commonly performed manually by the enterprise architects. In particular, the industrial partners in TEAP (namely Capgemini, DCNS and Obeo), based on their long-term expertise in EA and their concrete use cases, have identified some MDO shortcomings:

1. The capability of obtaining an initial cartography of the organization's system (here in terms of EA) from the relevant available information and data.
2. A standard (EA) representation facilitating interoperability that, at the same time, is flexible enough to be specialized for specific contexts and scenarios.
3. Support for the efficient handling of several views over the organization's system according to different viewpoints (here business, functional, technical, etc.).

The paper reports on the TEAP ongoing experience to target these MDO limitations in an industrial context while identifying relevant improvements to the MDE techniques themselves. We focus on three main MDE-based approaches allowing to:

- **Federate** heterogeneous data sources to **integrate** relevant EA information.
- **Adapt** more easily a standard EA solution to customer needs and potentially **trace** its different usages.
- **Support** multiple **views/viewpoints** over the same EA repository.

Resulting from this TEAP project, the SmartEA [17] tooling (continuously under development) is implementing a model-based EA framework integrating progressively the three MDE-based approaches mentioned before. Sections 2, 3 and 4 respectively introduce them with more details. Section 5 concludes by discussing our experience in TEAP and by summarizing both ongoing and future related works.

2 Model Driven Federation of Heterogeneous Data Sources

Within the context of EA, the amount of information to be considered is very large. Moreover, it can come in many different forms and quality levels (e.g., date, origin, completeness, relevance, etc.) and from several distinct data sources (e.g., XML documents, Excel files, databases, documentation, etc.). For the architects to deal more efficiently with this heterogeneity, it is important to provide them with a more advanced support for (semi-)automatically initiating their EA representation from this plethora of available data. For instance, the business processes of the organization are often already documented, at least partially or in a semi-structured format (e.g., in Excel). Thus, being able to create some EA representations from this business process information would be very helpful according to our industrial partners.

We call **data federation** such a "discovery + integration" process that populates an initial repository of interconnected models representing the company's EA.

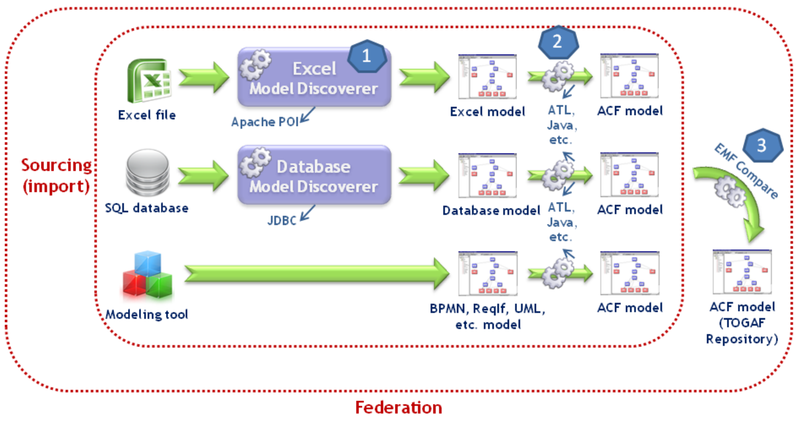


Fig. 1. Overview of the TEAP model driven data federation architecture

As shown in **Fig. 1**, our repository stores EA models (that conform to ACF, our TOGAF implementation). The objective is to get early model representations of the information from the different data sources so that we can benefit from MDE techniques when analyzing/handling them. Thus, model discoverers [1] have been implemented to automatically inject the needed initial *data* models (#1 in previous figure). Model-to-model transformations are then specifying the required *data-to-EA* transformations (#2 in figure), using DSLs (e.g., ATL [4]) or GPLs with model handling APIs (e.g., Java with EMF [13]). Finally, newly generated EA models can be integrated as part of the reference EA model(s) thanks to automated model comparison followed by manual merging decision (e.g., using EMF Compare [11]) (#3 in figure).

3 Model Driven Adaptability

Another fundamental characteristic of EA is the need for adaptability. Even if based on a well-known standard representation (e.g., TOGAF [18]), the concrete application of EA in different organizations often requires extending or specializing the core EA metamodel by reusing concepts coming from other metamodels. For instance, in our case, the core TOGAF metamodel had to be directly related with BPMN for business processes and ReqIf for requirements specifications: the EA information could more easily be linked to the data coming from different teams inside the company.

Within the context of TEAP, we address these two aspects of **adaptability** and corresponding **traceability** (between the extended EA elements and the related ones) from an MDE perspective. We first establish links (with different semantics such as *extension*, *trace*, etc.) between elements from two or more models. These links are then used to provide a global integrated representation of the different involved models, thus proposing a more general picture of the EA.

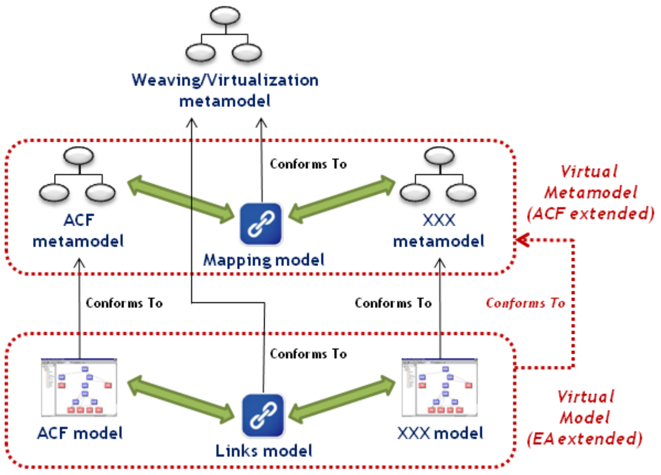


Fig. 2. Overview of the TEAP model driven adaptability and traceability architecture

As shown in **Fig. 2**, our proposal combines model weaving and virtualization techniques. Virtualization (Virtual EMF [20]) allows transparently accessing a set of related models as if they were composing a single model. The possible link types are defined at metamodel-level in a reusable *mapping model* (a weaving model) using different kinds of relationships (e.g., *isEquivalentTo*, *extends*, *refines*, etc.). Once such a mapping is specified (creating the *virtual metamodel*), a *virtual model* is automatically available based on a particular *links model (*i.e., model element-level links).

4 Multiple Model Views/Viewpoints Over a Central Repository

EA is about establishing an integrated representation of a whole organization. This is challenging as it implies visualizing EA models that can be very large and complex, notably because of the many different EA *building blocks* addressing several aspects of organizations (e.g., strategic, organizational, technological, etc.). Thus, for the framework to be actually usable, several interconnected *views* over the same EA repository are needed, targeting different user types/roles. This requires having specific viewpoints on the EA data, combining one or more predefined views.

As working with Obeo in the project, we rely on Obeo Designer [16] to support the smooth integration of several views, distinct or complementary, while working on a central EA repository. As shown on **Fig. 3**, this tool allows the definition of different graphical representations for the same model element. Thus, an element is displayed in one form or the other depending on the user's role or activity type, using an automated lock mechanism. Each viewpoint corresponds to a set of specified representations: diagrams, tables, matrices or trees that can be modified and/or extended if necessary. To realize this, Obeo Designer combines MDE techniques for model handling (EMF [13]), model comparison (EMF Compare [11]), graphical editing (GMF [12]) and model distribution (CDO [10]). The coupling between the concrete representations and the abstract syntax is minimized as much as possible to favor reusability.

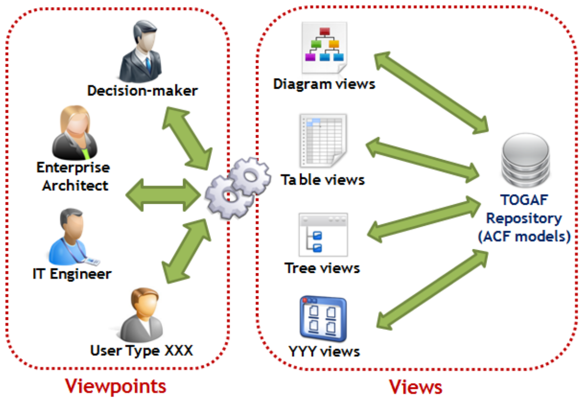


Fig. 3. Overview of the TEAP multiple model views/viewpoints

5 Discussion and Further Works

Based on our experience so far in the TEAP project, we can say that the effective combination and integration of several MDE techniques have shown to be able to bring benefits to some of the current industrial MDO shortcomings. This is particularly true when dealing with problems related to heterogeneity, adaptability or visualization. However, the main finding is that some adaptations and/or improvements from an MDE perspective have been necessary in order to tackle the targeted MDO challenges. This has notably been realized based on the constructive feedback received from the different EA experts (i.e., the end-users in our case) involved in the project.

Some of these aspects are the following. Working on the federation problem, we have to deal with quite different data sources (e.g., Excel sheets, Power Point schemas) than the ones usually considered in more standard Model Driven Reverse Engineering processes (source code, XML files, etc.). This is forcing us to modify the available model discovery support and to regularly upgrade it with new supported input formats, only having a partial (explicit) structure in some cases. While addressing the adaptability/traceability issue using model virtualization techniques, the tool integration aspects have highlighted the necessity of being able to virtualize not only models but also metamodels (which was not the case before). This has notably been required to improve model virtualization usability with already existing solutions (e.g., SmartEA in TEAP). Finally, concerning multiple views/viewpoints, we have realized that the problem was not so much on the tooling/feature side but rather on the human aspects: more particularly, how to agree on the best concrete syntax (i.e., representation) to use for each specific and different group of users [2].

We are convinced that the work in the project will continue to help us getting new relevant concrete insights, not only on how MDE can benefit EA (and potentially other related fields) but also on how EA, as a natural application field for MDE, can be valuable to guide the improvement of some of the current MDE techniques.

Acknowledgements. This work is funded by the DGCIS and the Pays de la Loire region (French FUI #13 - TEAP). We also thank Pierre-Xavier Fouille from DCNS for his very valuable feedback on both EA practices and the developed solutions.

References

1. Bruneliere, H., Cabot, J., Jouault, F., Lennon, Y., Madiot, F.: MoDisco: a Generic and Extensible Framework for MDRE. In: ASE 2010. Antwerp, Belgium. (2010)
2. Canovas, J., Cabot, J.: Enabling the Collaborative Definition of DSMLs. In: CAiSE 2013. Valencia, Spain (2013)
3. Clark, T., Barn, B., Oussena, S.: A Method for Enterprise Architecture Alignment. In: PRET 2012. Gdansk, Poland. (2012)
4. Jouault, F., Allilaire, F., Bezivin, J., Kurtev, I.: ATL: a Model Transformation Tool. In: Science of Computer Programming – Special Issue on EST (72), pp. 39-39. (2008)
5. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Third Edition, Springer (2013)
6. Mentz, J., Kotze, P., van der Merwe, A.: A Comparison of Practitioner and Researcher Definitions of Enterprise Architecture using an Interpretation Method. In: Advances in Enterprise Information Systems II, pp. 11-26. (2012)
7. Zachman, J. A.: Enterprise Architecture: The Issue of the Century. In: Database Programming and Design, pp. 1-13. (1997)
8. ArchiMate®, Open Group, *http://www.opengroup.org/subjectareas/enterprise/archimate*
9. DoDAF, U.S.A., current version 2.02, *http://dodcio.defense.gov/dodaf20.aspx*
10. Eclipse CDO Model Repository project, *http://www.eclipse.org/cdo/*
11. Eclipse EMF Compare project, *http://www.eclipse.org/emf/compare/*
12. Eclipse Graphical Modeling Framework (GMF), *http://www.eclipse.org/modeling/gmp/*
13. Eclipse Modeling Framework (EMF) project, *http://www.eclipse.org/modeling/emf/*
14. Magic Quadrant for Enterprise Architecture Tools, by Gartner, *http://www.gartner.com/technology/reprints.do?id=1-1CR3TJ5&ct=121108*
15. MODAF, United Kingdom, *https://www.gov.uk/mod-architecture-framework*
16. Obeo Designer product, *http://www.obeodesigner.com/*
17. SmartEA, Obeo, *http://www.obeosmartea.com*
18. TOGAF®, Open Group, *http://www.opengroup.org/subjectareas/enterprise/togaf*
19. TEAP collaborative project (French FUI), *http://teap-project.org/*
20. Virtual EMF EclipseLab, *https://code.google.com/a/eclipselabs.org/p/virtual-emf/*
21. Zachman, J. A.: The Zachman Framework™ – The Official Concise Definition, *http://www.zachmaninternational.com/index.php/the-zachman-framework*

Introducing Argumentative and Discursive Enterprise Leading and Management

Sebastian Bittmann[1], Balbir Barn[2], Tony Clark[2], Oliver Thomas[1]

[1] University Osnabrueck, Germany
[2] Middlesex University, London

Abstract. Leading an enterprise requires, obviously, decision making. However, these decisions require explanations in order to make it possible for stakeholders to get an understanding about the enterprise's strategic direction. This is even more important when these stakeholders are in charge to transpose such strategic decision into their tactical or operational work. Enterprise modelling may be capable of depicting strategies per se, but it is rather a vessel of communication than of explanation. Whilst, a strategy may be accordingly modelled, those who receive such a model needs to purposeful interpret and successfully implement it. However, without any insights, justifications or references that go beyond the claim of a model, it is difficult to embrace the theory of the actual modeller. Therefore, in this paper argumentative modelling will be specifically applied to the domain of strategic management. Moreover it will be elucidated how modelled strategic arguments can be used as a basis for enterprise architecture alignment and management. As it will be shown in the paper, the application of argumentative modelling overcomes classical restrictions and makes it possible to support a discourse, which can be later on used as an explanation for the intentions of the modeller.

1 Introduction

Strategies are a central component of the enterprise's success [1], as they guide the enterprise within an often unstable environment as well as they position the enterprise within a competitive and challenging market. So, whether the strategies are planned or emergent [2], they decide about the transition of the external requirements into business values by means of the enterprise's resources and capabilities. Although the strategy of an enterprise shall be incorporate by every artefact of the enterprise, the impact is often difficult to measure. This relates mainly to the often natural-language-based spread strategies [3]. Despite the available approaches for providing a strategy by means of a conceptual model [4] and further the consideration of strategic aspects in enterprise architectures [5], a possible benefit of the conceptualisation of a strategy is missing. For example, based on the strategy itself, the *management*, the *argumentative evaluation* of and *reasoning* about IT-artefacts is not possible, because of rather straightforward perspective on enterprise transformations [6, pp. 11-14]. Therefore the motivation of conceptualising the strategy for documentary reasons may be not sufficient to take the extra effort.

In order to motivate the relevance for conceptualising a strategy accordingly, in this paper, the concept of a strategic argument will be introduced that was derived from the theory of argument by Stephen Toulmin and explicitly from the conception of such an argumentative theory for modelling languages, namely the Argumentative Modelling Language [7]. Thereby, a relation between an IT-artefact and the respective part of the strategy becomes obvious, which enables a possible justification of the IT-artefact and further the motivation for an adaptation of the artefact based on changing conditions. The rest of the paper is outlined as follows. Initially, the theoretical conception behind the strategic argument is introduced in section 2. Following, the conception of the strategic argument as well as its relation towards both, the strategic and enterprise architecture management will be discussed in section 3. Successively, the approach will be evaluated in a case study in section 4. Lastly, the paper ends with a conclusion in section 5.

2 The Argumentation Modelling Language

The Argumentation Modelling Language (ArgML) was derived from the work by Stephen Toulmin, who proposed an argumentation theory from the field of jurisprudence [8]. Using a form of argument as proposed by the ArgML, enables the depiction of the theory that lies behind a conceptual model [9, 10]. Moreover the form of argument as proposed by Toulmin enables an evaluation of the underlying theory [11]. The conception of the ArgML is a complete exclusion of natural language and exclusive to semi-formal modelling languages, respectively domain-specific modelling languages. So any argument proposed by means of the ArgML follows a strictly specified syntax and is interpretable by clearly defined semantics. Therewith the abstract syntax of the ArgML is given by Figure 1. A detailed explanation of the key concepts that are either adopted based on Toulmin's theory or added with respect to requirements of the formalisation process.

The prime concept is the *argument*. Any argument is a container for a multitude of claims and their rebuttals. An arguments comprises claims that are specified by means of an uniquely chosen *language*. Generally, such a language should satisfy the purposes of the resulting model, so usually a domain-specific language should be chosen that can be characterised as semi-formal. Therewith, its specification offers various concepts for expressing claims. Every *claim* that is included by the argument and specified by the respective language embodies knowledge, which shall be established. It is expressed through a model, which follows the syntax of the respective argument's associated language. Respectively, a model is an instance of the chosen language of an argument used for expressing the designated claims. As the knowledge offered by claims may become established, those claims will be delegated by *grounds* in order to propose, respectively establish, new and upcoming knowledge by means of new claims. To conclude from grounds to claims, warrants define the implication from already grounded knowledge to newly claimed knowledge. *Warrants* as well as qualifiers are universal expressions, which are specified by the OCL. However, *qualifiers* define the validity of the qualifying claim. If the qualifier is satisfied, the claim is valid. A qualifier is expressed through OCL. Both the warrants and qualifiers refer explicitly to the different concepts offered by the language by means of their OCL expressions. Finally, a rebuttal is a further claim that contradicts a previous made claim, by means of conflicting statements. Based on the semi-formal language specification the resulting contradictions can be automatically identified.

Based on such formalisation, which are embodied by the abstract syntax of the ArgML (cf. Figure 1), the formalisation of arguments based on domain-specific languages becomes possible. With the use of semi-formal arguments, the actual discourse, which was held by the respective modeller, can be comprehended afterwards. Additionally, from the discourse, the actual resulting model can be derived, by combining the various different and valid claims. [7]

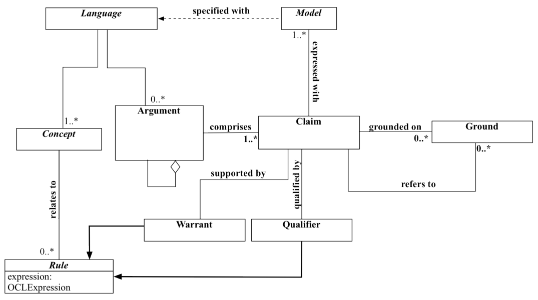


Fig. 1. Toulmin's Argumentation Theory [7]

3 Modelled Arguments of the Domain of Strategic Management

3.1 Phases of Strategic Management

With respect to strategic management, the strategic management process (SMP) [12] shall be used as a basis for alignment of the succeeding work. The SMP basically can be divided by four different phases. These are the analysis, specification, implementation and control phase. During the *analysis phase*, the SMP evaluates the environment as well as the enterprises possibilities and resources. Hence, the strength and weaknesses as well as the opportunities and challenges faced by the enterprise are evaluated. Subsequently, during the *specification phase*, a strategy will be stated that should be pursued by the enterprise. The strategy provides a plan for achieving a certain set of goals that were defined with respect to the previous analyses. After the specification, the strategy will be *implemented and enacted*. Hence, the strategy finds its concretisation by means of the different actions taken by the employees of the enterprise. Finally, the proper implementation needs to be audited during the *control* phase. Mainly the two initial phases are of relevance for the proposed approach, as these focus rather on concepts. With respect to the argumentation theory, the analysis phase provides facts on which strategic steps that claim the complete strategy may be grounded. Additionally, further concepts are needed that justify a strategic step on the facts of the analyses.

3.2 Strategic Arguments

Applying the ArgML to a specific domain, namely the strategic management, requires the various concepts of the ArgML to become domain-specific. Respectively, the concepts need to be adapted or specialised towards the specific requirements of the specific domain. Thereby, the ArgML represents a certain reference model [13], which has to be adapted towards the requirements of the specific domain, namely strategic management. Therefore, the ArgML has to support the grounding of claimed strategic steps. Based on their grounding, it needs to possible to justify the strategic steps regarding their expected benefit and practicability [14, pp. 53-130].

So arguing a strategy requires at least two parts: the explanation of the *possibility*, respectively if it's possible to resolve the strategy and there is a need for explaining the *expected benefit* is. Both these parts can be structured by means of the concepts of the ArgML. Beginning with the *claim*, which represent a certain statements that is sought to be established; its purpose is the proposition of a specific, rather atomic, strategic step. Based on this multitude of claims, those that ultimately become valid form the actual strategy of the enterprise [4].

Nevertheless, the proposed claims need to be grounded in order to justify the strategic argument with respect to the expected benefit and the practicability. Thereby, a *ground* of the ArgML is specialised by a strategic argument to any circumstance either owned by the enterprise as a form of resource or inhabited in the external environment. Accordingly, grounding on the enterprise's resources further justifies the possibility to implement the strategic step by the enterprise. Grounding on the environment, enables the elucidation whether a strategic step leads to a better position within the market or improves the overall competitiveness. So, a ground is a statement about an irrefutable state of the enterprise or its environment. So with having both concepts applied to the strategic management domain, the phase of analysis and specification of the strategic management can be purposefully supported by means of conceptual models.

To simply refer to a ground from a claim jeopardises the chance for insights, rebuttals and onward improvement [15]. Thereby, explanations are required that describe a certain design for realising a specific set strategy, against other possible design decisions. The concepts of a *qualifier*, enables a more specific statement about the required resources and external requirements. The qualifier embodies a certain set of rules that are capable of evaluating the grounds properly in order to make a statement about the practicability of the strategic step. If for example, the enterprise owns a certain IT-System, a strategic argument might refer to such a resource. However, the qualifier is able to state the exact amount of time this IT-System will be needed for the realisation. Thereby, refutations between two arguments, namely *rebuttals* can be identified based on the references and whether the respective requirements can be aligned or not. The concept of a *warrant* in a strategic argument represent the possible benefit the enterprise might have with following a specific strategic step. Hence, it must provide the conclusion for realising a strategic step, which is in the case of strategic management, the achievement of a specific strategic goal. Such a goal represents a change in the external or internal circumstances that ultimately should lead to a better position of the enterprise. Thereby within the perspective of argumentation, a strategic goal shall be viewed as a transition from a current to a desired future state. Next to the desired benefits, the warrant further has to provide the expected effort, which has to further be accounted by the prediction of a future state. On account of this, the warrant reveals the sense of pursuing a strategic step based on the circumstances of the enterprise. Ultimately the general conceptualisation can be gained from Figure 2.

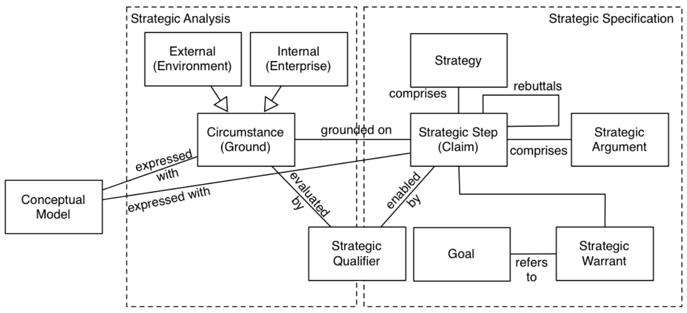


Fig. 2. Conception of a Strategic Argument

3.3 Implementation of Strategic Arguments within an Enterprise Architecture

As it was elucidated in the previous chapter, the strategic arguments can be used for the specific requirements evolved with reference to the analysis and specification of the enterprise's strategy. However, in order to illustrate the beneficial contribution, the concept of enterprise architecture management as a reference for the evaluation of possible applications of strategic arguments within the enterprise. An enterprise architecture subsumes the relevant artefact for the enterprise and their interrelation [16, 17]. In general, these artefacts are considered with strategical, such as product and services, as well as process, application and technological related issues [5, 18].

For the general purposes of this paper, the LEAP approach [19] was chosen, which represents a lightweight, component and layer based approach for enterprise architecture management. The general conception of a component as an artefact enables the illustration of general rules for applying the strategic argument throughout the enterprise architecture. Furthermore, the layered perspective of LEAP, fosters the refinement of strategic arguments to concrete software components and the generation of strategic directions based on technological innovations. Hence, the benefits of the approach can be elucidated by the use of a middle-out approach for enterprise arguments, without restricting the approach to the strategy and rather top-down or the technology and rather bottom-up, but enabling a possible alignment between technological and organisational innovations [20]. LEAP, although supports graphical representation by means of diagrams, uses textual representations. Respectively LEAP uses OCL statements for definitive specification of the enterprise architecture. The most general statement is `"(C,o)[n=v;...]`when Q" , whereby `C` is a class name, `o` an object identifier, `n` a name of a specific field and `v` is a value. `Q` is an OCL constraint that should satisfy the proper creation of `o`. So in order to enable the use of strategic arguments in LEAP, the form of statement needs to be properly adapted. Therefore it is necessary to enable references to enable a justification of the decisions. Respectively, whilst it is already possible to propose claims, it must be further necessary to ground these claims. So with

```
(C,o)[n=v;...] requires (R,QS) targets (G,WS) when Q
```

a specific form of the OCL constraint is given that satisfies the requirements of the conception of a strategic argument. Therefore `R` has been introduced for referencing an already available artefact of the enterprise architecture or an external circumstance and `QS` describes how this reference qualifies as a ground for the argument. Additionally `G` references any goal that should be targeted by means of the introduction of the artefact `o` and `WS` gives the justification of the achievement of the respective goal by means of the introduced artefact.

4 Case Study

In the following, an evaluation of the respective introduced approach will be undertaken by means of a case study. The case study focuses on an enterprise, whose primary business is the online retail of commercials. For that particular purpose, the company uses an own developed platform, on which the respective customers can purchase, request customer services and other services regarding compliance. Thereby, the enterprise mainly focuses on extending its customer base and additionally the increase of the customer value by means of cross- and up-selling.

The upcoming Figure 3 illustrates the enterprise architecture on a current state and additionally, several arguments that claim adaptations within the enterprise architecture. So, in an excerpt, the enterprise architecture includes three different business processes that were proposed to target the achievement of the initial strategy. These business processes are further supported by certain applications that were coordinated on the application layer. Ultimately, the enterprise architectures include certain technology for the realisation of the application systems. However, while the workflow management system has already been introduced, it hasn't been yet used for the realisation of an application system or a business process.

So, initially, as depicted by Figure 3, an automation of the customer services was proposed based on the respective goal, namely the sale increase. This was grounded on the opportunities a workflow management system offers and additionally, the available time of the customer consultants that prior were in charge for executing the services. So, based on the ground and the respective goal to achieve and strategy to resolve, the argument proposes an automation or partly automation of the business processes "Order Processing" and "Customer Services". The formalisation of this argument regarding customer services is given in the following.

```
context CustomerServices
self.includeComponent(CRM[components=(WfMS,CustomerDatabase), ...])
and self.includeComponent(SelfServicePortal[components=WfMS,...])
   requires CustomerConsultants.avgWeeklySpentTime(OrderProcessing)
            >36000s and TechnologyLayer.includes(WfMS)
   targets SaleIncrease >= 0.1
```

Additionally, a further argument proposed the focus on the customer relation with the motivation of an increase of five per cent of the customer value. Such argument is grounded on the customer requirements and additionally, on the previous analysed potential of cross- and up-selling. So, in order to resolve such a strategy, the business processes of customer services and customer compliance were sought to adapt to the upcoming and *justified* customer needs, as given below.

```
context CustomerServices
self.includeComponent(CustomerServiceSystem
   [components=(CustomerServices,CustomerDatabase), ...])
   and self.excludeComponent(WfMS)
   requires CustomerReBuyPotential>0.5
   targets AvgRevenuePerCustomerIncrease>0.05
```

However, on a later stage, respectively the design of the respective business processes based on the proposed strategic arguments, it occurs that although prior in harmonisation, two of the goals provided by the strategic arguments are in conflict. Such a conflict results from the necessity for the process "Customer Services" for being compliant to both prior discussed strategic arguments. With the current business process landscape of the company, the respective business process has to be designed with reference to automation potential as well as customer relation. These directives are in conflict, as the customer relation needs to be fostered by a customer consultant and requires human interactions. Hence, the identification of a rebuttal within the set goals requires a specific level of concretisation by the respective strategic step.

In parallel and untouched from this conflict, a request from the operative has arisen that demands the integration with the ticket manager and the customer

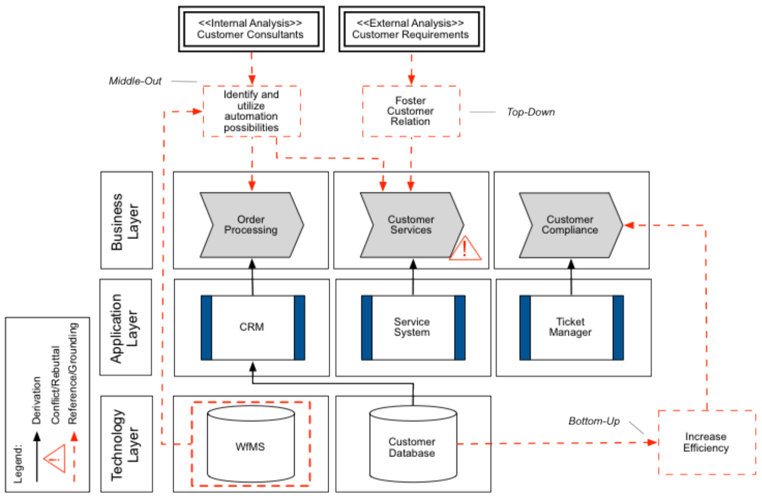


Fig. 3. Initial Derived Enterprise Architecture on a Conceptual Layer supplemented with Strategic Arguments

database, which was initially only used by the CRM System, for the more efficient performance of the customer compliance process.

```
context CustomerCompliance
self.includeComponent(TicketManager.includeComponent(CustomerDatabase))
    requires TicketManager.datamodel
      .intersection(CustomerDatabase.datamodel).includes(Customer)
    targets EfficiencyIncrease > 0.1
```

So, as depicted by Figure 3, the actual conflict between two arguments only become visible, after the conceptual realisation of the respective artefacts, as the design requires balancing between opportunities. Thereby, design decisions within the business processes are contrary to each other, but both were based on a strategic direction that initially didn't reveal a conflict.

However, with the identification of the actual conflict and documentation of the design decisions, the different strategic steps can be purposefully refined in order to propose a less ambiguous strategy with respect to enterprise architecture management. With the strict grounding on the specific *targets* and *requirements* of the artefact, the alignment to the business strategy of the company was possible, as inference could be made to the initial intentions, which was derived from the overall enterprise strategy. Moreover with the refinement of the strategic arguments, the enterprise architecture can be purposefully aligned. With a more elaborated strategy as well as the introduction of additional artefacts, the alignment was feasible, without jeopardising the overall strategy implementation. The upcoming Figure 4 represents the aligned enterprise architecture that specifically embodies the adapted artefacts towards the three initially proposed strategic arguments. Specifically, the strategic steps that aimed at the customer relation were adapted. Therewith a self service business process was added with reference to those services that won't necessarily benefit from human interactions.

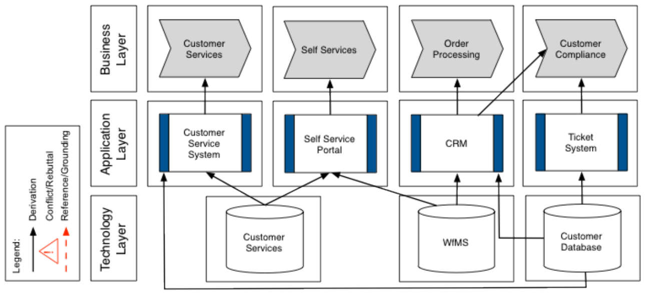


Fig. 4. Aligned Enterprise Architecture based on Strategic Arguments on a Conceptual Layer

5 Conclusion

With the presented approach, a manner of enterprise architecture management was introduced that uses a form of argument for purposefully managing the evolution of the enterprise architecture in order to enable the purposeful alignment with the overall enterprise strategy. With the use of arguments a misalignment between the strategy and the enterprise architecture becomes identifiable with a concretisation of the actual artefacts by means of their design directives. Thereby, design decisions of artefacts can be supported by their underlying rationale derived from the strategic directives. As on first sight, strategic steps seem in harmony, later on, a misalignment can be revealed based on a more complete specification. With having the design decision's rationale the adaptation as well as the alignment of the respective artefacts can become more directed and contributing. Additionally, the approach supports bottom-up and middle-out, next to rather strategic top-down, implementations and thereby, is able to return developed insights to the overall strategy of the enterprise.

References

1. Porter, M.E.: Competitive Strategy: Techniques for Analyzing Industries and Competitors. The Free Press, New York (1980)
2. Mintzberg, H., Waters, J.A.: Of strategies, deliberate and emergent. Strategic Management Journal **6**(3) (July 1985) 257–272
3. Hoppenbrouwers, S., Proper, H.A., Van Der Weide, T.P.: A Fundamental View on the Process of Conceptual Modeling. In Delcambre, L., Kop, C., Mayr, H.C.,

Mylopoulos, J., Pastor, O., eds.: Proceedings of the 24th International Conference on Conceptual Modeling. Volume 3716 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg (2005) 128–143
4. Kaplan, R.S., Norton, D.P.: Having trouble with your strategy?: Then map it. In: Focusing Your Organization on Strategy - with the Balanced Scorecard. 2nd edn. Harvard Business School Publishing Corporation (2000)
5. Winter, R., Fischer, R.: Essential layers, artifacts, and dependencies of enterprise architecture. Journal of Enterprise Architecture **3**(2) (2007) 7–18
6. Greefhorst, D., Proper, E.: Architecture Principles: The Cornerstones of Enterprise Architecture(Google eBook). (2011)
7. Bittmann, S., Barn, B., Clark, T.: A Language Oriented Extension to Toulmins Argumentation Model for Conceptual Modelling. In: 22nd International Conference on Information Systems Development (ISD2013). (2013)
8. Toulmin, S.E.: The Uses of Argument. Updated edn. Cambridge University Press, Cambridge, UK (2003)
9. Barn, B.S., Clark, T.: A domain specific language for contextual design. In Bernhaupt, R., Forbrig, P., Gulliksen, J., Lárusdóttir, M., eds.: Human-Centred Software Engineering. Volume 6409. Springer, Berlin Heidelberg (October 2010) 46–61
10. Barn, B.S., Clark, T.: Revisiting Naur's programming as theory building for enterprise architecture modelling. In: CAiSE'11 Proceedings of the 23rd international conference on Advanced information systems engineering, Berlin, Heidelberg, Springer (June 2011) 229–236
11. Gregor, S.: The nature of theory in information systems. MIS Quarterly **30**(3) (September 2006) 611–642
12. Näsi, J.: Information systems and strategy design. Decision Support Systems **26**(2) (August 1999) 137–149
13. Thomas, O.: Management von Referenzmodellen: Entwurf und Realisierung eines Informationssystems zur Entwicklung und Anwendung von Referenzmodellen. Logos, Berlin (2006)
14. Johnson, G., Scholes, K., Whittington, R.: Exploring Corporate Strategy: Text and Cases. (2008)
15. Plataniotis, G., Kinderen, S., Proper, H.: Capturing Decision Making Strategies in Enterprise Architecture A Viewpoint. In Nurcan, S., Proper, H., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T., Bider, I., eds.: Enterprise, Business-Process and Information Systems Modeling SE - 24. Volume 147 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2013) 339–353
16. Zachman, J.A.: A framework for information systems architecture (1987)
17. Aier, S., Riege, C., Winter, R.: Unternehmensarchitektur Literaturüberblick und Stand der Praxis. WIRTSCHAFTSINFORMATIK **50**(4) (September 2008) 292–304
18. Rychkova, Irina; Wegmann, A.: A Method of Functional Alignment Verification in Hierarchical Enterprise Models. In Latour, T., Petit, M., eds.: Proceedings of the CAISE06 Workshops and Doctoral Consortium, Namur, Presses universitaires de Namur (2006) 244–253
19. Clark, T., Barn, B.S., Oussena, S.: LEAP. In: Proceedings of the 4th India Software Engineering Conference on - ISEC '11, New York, USA, ACM Press (February 2011) 85–94
20. Henderson, J.C., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming organizations. IBM Systems Journal **32**(1) (January 1993) 4–16

(Multi-) Modeling Enterprises for Better Decisions

Sagar Sunkle, Vinay Kulkarni, and Hemant Rathod

Tata Research Development and Design Center
Tata Consultancy Services
54B, Industrial Estate, Hadapsar
Pune, 411013 INDIA
{sagar.sunkle,vinay.vkulkarni,hemant.rathod}@tcs.com

Abstract. Enterprises typically depend on past experience and human expertise for responding to changes. This is no longer cost effective in the increasingly dynamic world of enterprises. Enterprise models are required that describe the enterprise as well as prescribe courses of action in the face of change. Owing to complexity of enterprises, multiple models need to be employed when addressing specific problems. Toward this end, we present an approach that uses number of specialized models focused on decision making. Our contributions are- first, we show how these models can be used in concert and second, we present how these models can be used in a real world case study of merger of two enterprises. Our ongoing research suggests that in spite of several challenges, this approach provides promising first steps toward enabling enterprises in responding to changes with more certainty.

Keywords: Enterprise Models, Multi-Models, Decision Making, Enterprise Analysis

1 Introduction

For the past 17 years, we have helped 70+ enterprises in their IT automation efforts. In recent times, we have witnessed that the need for automation and its actual realization in an enterprise is driven by multiple change drivers in various dimensions. Modern enterprises display a characteristic connectedness between these dimensions; change in any one dimension affects the others. Our experience suggests that lack of enterprise-wide context that encompasses all dimensions, is the key cause of enterprises' inability to change cost effectively.

This led us to posit that a model of enterprise is needed that is capable of catering to all relevant dimensions of enterprise while being machine processable and analyzable [1, 2]. Furthermore, specialized modeling languages could be used with these enterprise models for decision making in enterprises [3, 4]. Our past experience and ongoing investigations suggest that we essentially need both descriptive [5] and prescriptive models of enterprise [6]. In this paper, we describe various models that we think are needed for better decision making. Our specific contributions are twofold-first, we systematically elaborate on how these models can be used in-concert, and second, we put the proposed research in the context of Mergers and Acquisitions (M&A), a special case of enterprise transformation. By elaborating how our approach can be used for solving prevalent M&A problems in general and M&A of two large banks in specific, we bring out further facets of our proposed research.

The paper is arranged as follows. Section 2 motivates the need for multiple kinds of models of an enterprise and based on our experience and investigation puts forth the kinds of models needed for better decision making. Section 3 describes how we are applying the proposed approach to a real world M&A case study. Section 4 concludes the paper.

2 Enterprise (Multi-) Models

2.1 Motivation

Enterprises are complex systems of systems and responding to change is extremely cost-, time-, and effort- intensive [2]. This is felt both when aiming at sustainable enterprises, i.e., maintaining business-as-usual in the face of change [7] and tackling enterprise transformation scenarios, i.e., changing enterprises substantially in response to change [8]. If we consider business, IT, and infrastructure to be roughly the key dimensions of any enterprise [5], we can see that changes in one dimension are affected by change drivers in other dimensions. Business change drivers such as economic and socio-political drivers, and more specifically drivers like dynamic supply chains, mergers, acquisitions, and divestitures, and globalization and regulatory compliances, etc., eventually lead to changes in IT and infrastructure. Similarly, changes demanded by mobile and cloud technology eventually result in changing the way business is performed.

As far as IT systems are considered, enterprises use them in particular to derive mechanical advantage through automation of operational processes catering to their strategic, tactical and operational goals [9]. Large enterprises traditionally operate in siloized manner for ease of management and control. This results in IT departments knowing only their local context. As a result, IT systems are developed independently and individually to service globally mandated goal, be it transactional (i.e., business-as-usual) or transformational in a very specific context. This leads to a plethora of IT systems servicing same goal from the global context. Furthermore, technological requirements in specific contexts may vary, resulting in widely non interoperable technologies. Together these problems result in highly escalated cost of IT to business.

In general, management relies solely on expert judgment about how to tackle changes across dimensions of enterprise and in particular resolving issues pertaining to IT systems. Our take on this is it is better to make machine-processable and analyzable models of enterprise as a whole as well as of its key parts that are respectively descriptive and prescriptive in nature [10], so that decision making is automated to the extent possible, yet always carried out in the context of all of an enterprise.

2.2 Proposed Models

In using models for better decision making in enterprises, we need models that act as inventories of all relevant information, i.e., models that are descriptive. An ArchiMate-based enterprise model that captures information about business, IT, infrastructure entities and relations while conforming to ArchiMate metamodels would be such a model.

We also need models that can use such information and prescribe course(s) of actions based on some criteria. This core distinction is at the basis of several kinds of models we are proposing for better decision making in enterprises as shown in Figure 1.

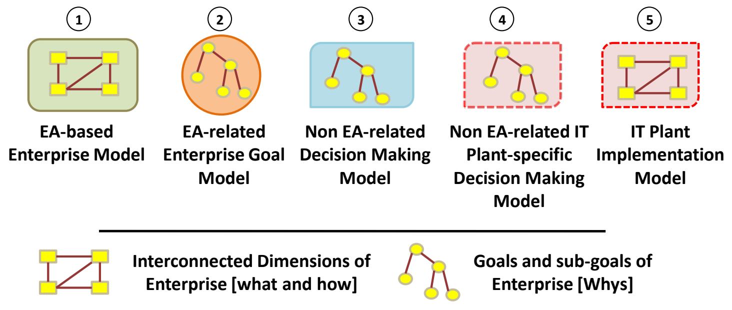

1
EA-based Enterprise Model
2
EA-related Enterprise Goal Model
3
Non EA-related Decision Making Model
4
Non EA-related IT Plant-specific Decision Making Model
5
IT Plant Implementation Model
Interconnected Dimensions of Enterprise [what and how]
Goals and sub-goals of Enterprise [Whys]

Fig. 1: Different Kinds of Models Required for Better Decision Making in Enterprises

EA-based Enterprise Model(s) In Figure 1, ① shows an enterprise model that should be based on an enterprise architecture (EA) framework. This is a descriptive model that inventories the information about key dimensions of an enterprise. This model can be queried for stakeholder-specific information and can also be used for some informative analyses pertaining to given EA framework. Our initial results in creating such a model were presented in [1].

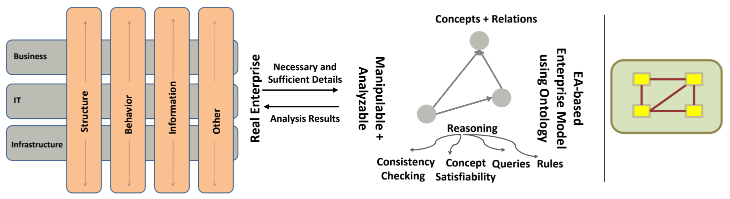


Fig. 2: EA-based Enterprise Model using Ontological Representation [1]

Figure 2 shows how an EA-based ontology representation, in this case based on ArchiMate, can essentially make enterprise models machine-processable and analyzable. Note that ① kind of models capture *what* and *how* of an enterprise but not *whys*.

EA-related Enterprise Goal Model ② in Figure 1 shows a prescriptive model focused on capturing *whys* or intentions behind decisions taken in enterprise. For this, we suggested first steps using intentional modeling language i* for capturing goals in enterprise models based on ArchiMate. In contrast to goal descriptions provided by the EA framework, such as ArchiMate motivation extensions, our approach enables solving problems faced by enterprise, and select the optimum strategy from the available ones [3, 4]. This is illustrated in Figure 3. We used a bidirectional metamodel mapping to represent problems faced by our own model-driven development unit in the concerned case study.

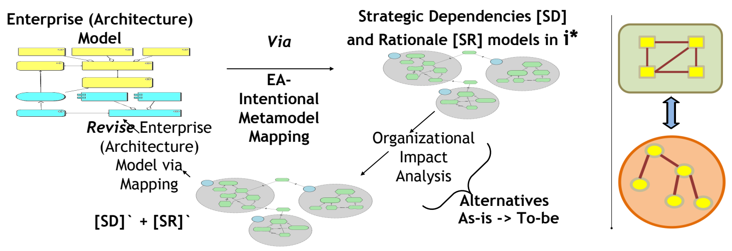

Enterprise (Architecture) Model
Via
EA-Intentional Metamodel Mapping
Strategic Dependencies [SD] and Rationale [SR] models in i*
Organizational Impact Analysis
Alternatives As-is -> To-be
Revise Enterprise (Architecture) Model via Mapping
[SD]` + [SR]`

Fig. 3: Intentional Model Related to the Enterprise Model [3]

Non-EA-related Decision Making Model The prescriptive models of Figure 3 are relatable with enterprise models, i.e., based on similarity in concepts of actor, behavior, resources one could relate these models [3, 4]. But we are currently also investigating the utility of prescriptive models that are not relatable to enterprise models in the sense that there is no metamodel to which these models conform that could be mapped to enterprise metamodels in this case ArchiMate metamodels. (3) in Figure 1 illustrates such models. We describe one such model using an example of workforce planning.

Generally, spreadsheets are used to encode workforces planning parameters to monitor and to take actions based on parameter values. Essentially, operational decisions are represented using formulaic language supported by spreadsheets suitable for quantification which is cryptic and spread over number of spreadsheets. The information about why certain formulae are used remains largely unclear. What-if and if-what analyses, i.e., evaluation of alternate strategies from as-is state of enterprise to its to-be state and search of strategy that might have led the enterprise to its current as-is state respectively, are difficult to explicate with such representation. These analyses need to be conducted frequently as for a service company such as ours, workforce situation remains in flux. We are therefore investigating the use of system dynamics (SD) models to address the problem of workforce planning. The detailed description of how we propose to use SD models to capture issues of attrition, on roll skills, and on the job headcount, and how recruitment from campus, lateral hiring, on-boarding process etc. affect these is out of the scope of this paper.

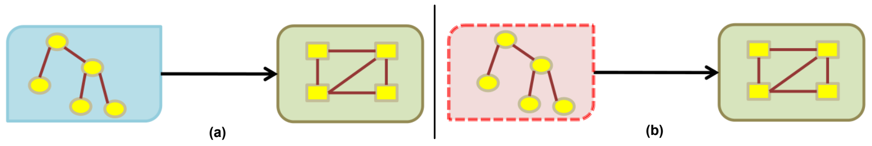


Fig. 4: (a) Using Non-relatable Prescriptive Models with Enterprise Models (b) IT Plant-specific Prescriptive Models with Enterprise Models

The core concepts of stocks, flows, and influencing variables in SD represent drastically different abstractions than those found in enterprise metamodels. Instead of metamodel mapping therefore, we propose to use these models in-concert by explicating the analysis results in the form of strategies to achieve goals which are represented with bidirectional mapping to enterprise models. This is shown in Figure 4 (a). One obvious example where the problems of workforce planning are directly related to enterprise-wide context is finding the steps required for an organization to increase revenue by 10% quarter on quarter. Cost benefit analysis of workforce planning would need to see how revenue earned changes as a function of on roll manpower, project mix to bid, and bid success ratio etc. The results of this analysis need to be put in the overall enterprise context, in the ArchiMate sense, at the business layer, perhaps in terms of actions that resource management department may take to satisfy the strategic goal of increased revenue.

Non-EA-related IT Plant-specific Decision Making Model In Section 2.1 we referred to the problems of IT systems of an enterprise that must be addressed in both business-as-usual and transformational situations. The set of interacting IT systems of an enterprise, and technology and hardware infrastructures underneath them is what we refer to as an IT plant. We believe that to address the problems of local optimality with respect to a given property, functionality overlapping, and non-interoperability between IT systems, we need to model IT systems and their interactions with specialized models. These kinds of models are shown as **4** in Figure 1. Their usage with enterprise models is shown in Figure 4 (b).

We expect these models to be graph-like where IT systems are nodes and the edges are interactions such as depending on one another, accessing same set of data, simply relaying data, and so on between them. The analysis essentially focuses on interaction patterns of the systems of an IT plant. Such models may be constructed by observing in automated manner how IT systems use one another and how changes in underlying technology platforms and hardware infrastructure of an IT system affects other IT systems of given IT plant. By constructing models based on interaction patterns of IT systems and refining them over a period of time, we think that problems of optimality with respect to given criteria, non-interoperability, and overlapping functionalities could be addressed effectively. This constitutes part of our ongoing work. Like models illustrated earlier in Figure 4, results of IT plant-specific model analysis will have to be put in the context of enterprise as shown in Figure 4 (b).

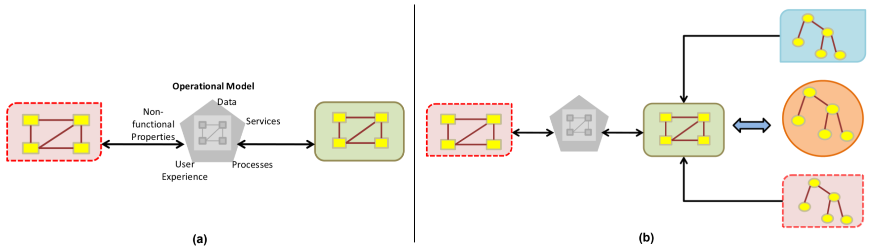


Fig. 5: (a) Relating IT Plant Implementation Model with Enterprise Model (b) Using (Multi-) Models of Enterprise in Concert

IT Plant Implementation Model While strategic and tactical goals focus on the long-term orientation of the organization and short-range planning respectively, operational goals focus on implementing tactical goals at the ground level. This is also more relevant in the case of IT as business rather than IT as support function of business. We are proposing that there should be a bidirectional traceability between enterprise model and IT plant implementation models via operational models that translate strategic level requirements of data, services, processes, user experience, and non-functional properties to implementation level specifications from which IT plant may be generated. (5) in Figure 1 shows such models which are illustrated in overall context in Figure 5 (a).

We are aware of stark differences in levels of abstraction and granularity of concepts between enterprise models and IT plant implementation models and the operational models with data, service, process, non-functional properties, and user experience requirements in between are expected to bridge this gap. Earlier in Figure 1, we did not show the operational model, because we are not certain of what will be the nature of these models or if these models are required at all. We believe that it should be possible to automate to the extent possible translation of results of various analyses obtained with the rest of the models illustrated in Figure 1 into an actionable form.

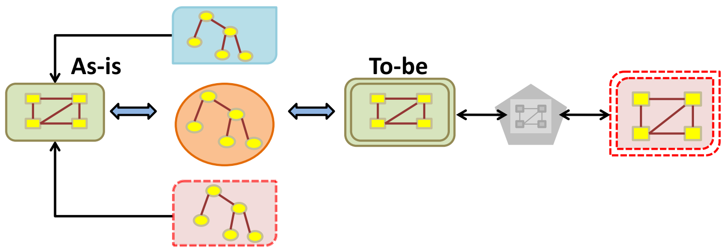


Fig. 6: Using (Multi-) Models of Enterprise in Concert for Enterprise Transformation

Figure 5 (b) shows various models illustrated so far together. While the enterprise model remains the **single version of truth**, results of analyses using non-EA-related decision making models of IT- and non-IT-specific models are to be explicated in conjunction with enterprise goal models for better decision making. Since ultimately our focus is on IT systems, we have proposed to connect enterprise models to IT plant implementation models via operational models (of kinds of requirements). In the next section we review further issues pertaining to multi-modeling of enterprises.

While Figure 5 (b) can be used straight away for decision making in business-as-usual situations, Figure 6 shows how transformational situations can be addressed using proposed models. Note that Figure 6 does not show as-is IT plant implementation models for the want of space. The as-is and to-be EA-based enterprise models are connected via EA-related goals models. Other decision making models may be used for specific problems in transformation and eventually IT plant implementation of to-be enterprise model may be obtained.

In the next section, we show how the proposed models may be applied to a case study of M&A of two wealth management companies taking into account various issues outlined above.

3 Proposed Application to M&A

Background The case study concerns two large independent Wealth Management (WM) (retail brokerage) companies (WM1 and WM2) which came together to form WM3. The combined retail brokerage house has 10000+ Financial Advisors, managing multi-billion dollars in client assets across 700+ locations in country X. WM1 and WM2 both provide WM products such as credit, lending, annuity, insurance, banking, etc. and services like brokerage, advisory, financial planning, wealth planning, retirement planning, and trust etc., to wealthy individuals and small-to-medium size businesses across X.

WM3 was formed the expressed strategic goal of tripling WM1's revenue and gross margin in 5 years. WM3 also has strategic growth viewpoint where it needs to provide new and innovative products and services to its clients. This requires a renovated state-of-the-art IT platform to compete with its more aggressive peers and ever increasing tech-savvy clients. In the following, we discuss how various decision making models may be used to address specific problems in conjunction with the enterprise model.

Using non EA-related Decision Making Models for Business Aspects Several tactical goals were devised that would contribute positively to key strategic goal. Three of these tactical goals were optimize/rationalize branch and back-office operations, rationalize wealth management products and services, and of course, integrate workforces of WM1 and WM2. Each of these three goals can be achieved using SD or similar models, which are essentially examples non EA-related decision making models.

For the tactical goals mentioned above, particularly for workforce integration of WM1 and WM2, SD models could be used in a manner similar to the way they are used in workforce planning as discussed earlier in Section 2.2. For optimization/rationalization of branch and back-office operations, an important fact to consider is that WM1 and WM2 were competitors prior to the merger with several branches in the same locality. Consolidating the branches and their operations would result in recurring cost savings every year. The decision to which ones to keep as-is, which ones to merge, and which ones to let go of depends on branch operations related parameters such as whether it is owned/leased, cost and duration of the existing lease, importance of the location from WM3 perspective, floor capacity, terms and conditions, future growth potential at that location, etc. Decisions on optimal branch structure, e.g., how many/which branches should form a complex, how many/which complexes should be part of a region (western/eastern/northern/southern), etc., need to be taken to better manage WM3. The WM1/WM2 back-office operations also need to be optimized along similar lines.

WM1 and WM2 operate in similar business domain and hence have similar and even overlapping product and service portfolios. For rationalization of wealth management products and services, it is necessary to look at these products and services portfolios from WM3 business model perspective and take decision on keeping as-is, enhancing (modifying/merging/re-branding), decommissioning (retiring) the mix of products and services. The parameters that would be of relevance in taking decisions include product capabilities, channels, WM3 requirements, integration with other products/services, 3rd party (product/bank/vendor) involvement, etc. Also, it makes sense to look at cross-selling opportunities within WM1 and WM2 in terms of existing clients for products/services that were otherwise not available before the merger.

Using non EA-related IT Plant-specific Decision Making Models With regards IT platforms of WM1 and WM2, key goals were to integrate IT platforms of WM1 and WM2 so as to obtain optimum IT platform functionality in WM3, optimizing WM3 IT platform capacity, and come up with optimum data conversion/migration from WM2 to WM1 in WM3.

In case of optimum WM3 IT platform functionality, the alternatives are to build a new IT platform or enhance one of the existing WM1 or WM2 IT platforms to support the WM3 target operating model arrived at separately. In order for the target IT platform to support WM3 operating model, decisions need to be made on which applications to keep as-is from WM1 or WM2, which ones to enhance (modify/merge-functionality), which ones to build from scratch (nothing can be reused from WM1/WM2), which ones to decommission and when. Note that problems of IT systems of enterprise described in Section 2 get accentuated in this case because of the size and legacy of WM1 and WM2 combined together and therefore need further efforts in capturing interaction patterns of IT systems of WM1 and WM2 themselves and possible interactions between IT systems of WM1 and WM2.

To resolve the problem of optimizing WM3 IT platform capacity, current size of WM1 + WM2 needs to be considered along with the future growth plans for WM3 which is that the capacity of the WM3 platform needs to be doubled. It means the existing applications (selected for WM3) should be able to handle 3 times their current volume (# of transactions, clients, branches, financial-advisors, employees, etc.) without (negatively) impacting performance. This would require changes such as optimizing/ re-writing database queries, re-architecting / re-designing, adding more hardware, etc. The changes would cut across multiple layers and applications. There could be problems similar to Y2K that need to be addressed. For example, 3 digits were sufficient to accommodate WM1 Branches, but WM3 is going to have more # of branches and 3 digits may no longer be sufficient to accommodate WM3 branches.

For data migration problem, existing and historical data needs to be converted from the source to the target platform and applications. This includes both business critical and non-critical data. Since there is terabytes of data involved and limited conversion (live cut-over) time-window available because of the nature of the business, there is very limited scope for making an error (in speed and quality of the conversion) during the entire process. The converted data should also comply with the regulatory requirements applicable for WM3. We are currently investigating how above mentioned problem descriptions can be represented using decision making models of IT plant as described in Section 2.2.

Decision making in M&A Problems in Enterprise Context Figure 7 extends the transformational situation captured in Figure 6, and shows the merger of WM1 and WM2. This merger was initiated by WM1, therefore the as-is and to-be states are depicted as that of WM1. EA-based enterprise models of WM1 and WM2 are descriptive models which capture all relevant information about WM1 and WM2 in a manner explained in [1]. The EA-related enterprise goal model using intentional modeling represents the strategic goal of revenue increment and its further breakdown into sub-goals as in [3, 4]. The decomposition of goals and sub-goals here needs to continue till results of analysis on non EA-related decision making models including those that are IT plant-specific can be plugged in suitably in the EA-related enterprise goal model. Chosen alternatives to resolve specific problems are eventually implemented in the IT plant implementation of WM3 via bridge provided as discussed in Section 2.2.

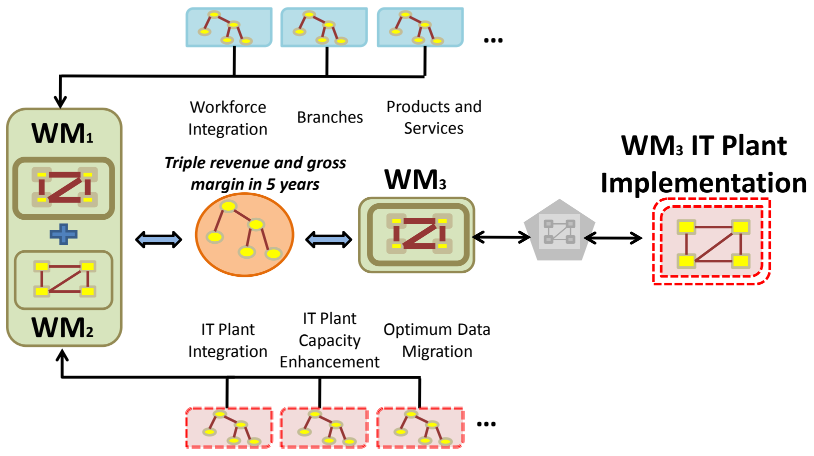


Fig. 7: Using (Multi-) Models of Enterprise in Concert for M&A

As we proceed with our approach, we found some issues that need to be addressed which we enlist below-

- **Integrating multi-models of enterprise** As discussed, multi-(level and formalism)models are needed to address problems faced by enterprises. Using common ontology to map concepts from different modeling languages to each other [11] and level-agnostic metamodeling [12] may provide some help to address this issue.
- **Keeping models and reality in sync** Reality may already have changed till the time various models are completed. We think agile concepts applied to enterprise modeling would help along the lines discussed in [13]. Furthermore, multi-models focused on specific concerns may help in keeping models and aspects of reality they capture in sync.
- **Relating strategic goals with properties of operational elements** Strategic goals and desirable properties of operational elements like IT systems are at different level of abstraction. We need ways of computing properties [14] and deliberating their tradeoff before plugging them in various decision making models.
- **Treating various uncertainties** Some of the uncertain aspects of enterprise modeling are variation in meaning of concept based on modeling language, modeling the reality completely and accurately, reconciling modeling information spread over multiple sources and multiple levels of abstraction, and ways in which enterprise phenomena affect each other. Probabilistic methods are suggested for this [15], but further research is needed for application to specific kind of uncertainty.

Many of the above issues have been recognized already by several researchers in different contexts. Complexity and size of enterprise models make addressing them as effectively as possible even more pertinent.

4 Conclusion

We proposed combination of EA-based enterprise model and set of models specialized for decision making pertaining to specific aspects of enterprise as descriptive and prescriptive models respectively. Initial treatment of real world merger of two large enterprises using our multi-model approach indicates that problems faced by enterprise that demand specific ways of solving can be modeled and solved in separation yet maintaining the overall enterprise context. Furthermore, modeling enterprise concerns down to IT plant itself means that both business-as-usual and transformational situations can be tackled. Initial results suggest that multi-models of enterprise help in separating and localizing decision making in enterprise.

References

1. Sunkle, S., Kulkarni, V., Roychoudhury, S.: Analyzing Enterprise Models Using Enterprise Architecture-based Ontology. In: MoDELS. (2013) Accepted.
2. Kulkarni, V., Sunkle, S.: Next Wave of Servicing Enterprise IT Needs. In: IEEE Conference on Business Informatics. (2013) Accepted.
3. Sunkle, S., Kulkarni, V., Roychoudhury, S.: Intentional Modeling for Problem Solving in Enterprise Architecture. In: Proceedings of International Conference on Enterprise Information Systems (ICEIS). (2013) Accepted.
4. Sunkle, S., Roychoudhury, S., Kulkarni, V.: Using Intentional and System Dynamics Modeling to Address *WHY*s in Enterprise Architecture. In: ICSOFT-EA. (2013) Accepted.
5. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer (2005)
6. Saat, J., Winter, R., Franke, U., Lagerström, R., Ekstedt, M.: Analysis of IT/Business Alignment Situations as a Precondition for the Design and Engineering of Situated IT/Business Alignment Solutions. In: HICSS, IEEE Computer Society (2011) 1–9
7. Laverdure, L., Conn, A.: SEA Change: How Sustainable EA Enables Business Success in Times of Disruptive Change. Journal of Enterprise Architecture **8**(1) (feb 2012)
8. Rouse, W.B., Baba, M.L.: Enterprise transformation. Commun. ACM **49**(7) (2006) 66–72
9. Ross, J.W., Beath, C.M.: Sustainable IT Outsourcing Success: Let Enterprise Architecture Be Your Guide. MIS Quarterly Executive **5**(4) (2006)
10. Greefhorst, D., Proper, E.: Architecture Principles: The Cornerstones of Enterprise Architecture (The Enterprise Engineering Series). 2011 edn. Springer (June 2011)
11. Roque, M., Vallespir, B., Doumeingts, G.: Interoperability In Enterprise Modelling: Translation, Elementary Constructs, Meta-modelling And Ueml Development. Computers in Industry **59**(7) (2008) 672–681
12. Henderson-Sellers, B.: On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages. Springer Briefs in Computer Science. Springer (2012)
13. McGovern, J., Ambler, S.W., Stevens, M., Linn, J., Sharan, V., , Jo, E.: The Practical Guide to Enterprise Architecture. Prentice Hall (2003)
14. Lagerström, R., Saat, J., Franke, U., Aier, S., Ekstedt, M.: Enterprise Meta Modeling Methods - Combining a Stakeholder-Oriented and a Causality-Based Approach. In Halpin, T.A., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R., eds.: BMMDS/EMMSAD. Volume 29 of Lecture Notes in Business Information Processing., Springer (2009) 381–393
15. Johnson, P., Lagerström, R., Närman, P., Simonsson, M.: Enterprise Architecture Analysis With Extended Influence Diagrams. Information Systems Frontiers **9**(2-3) (2007) 163–180

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.

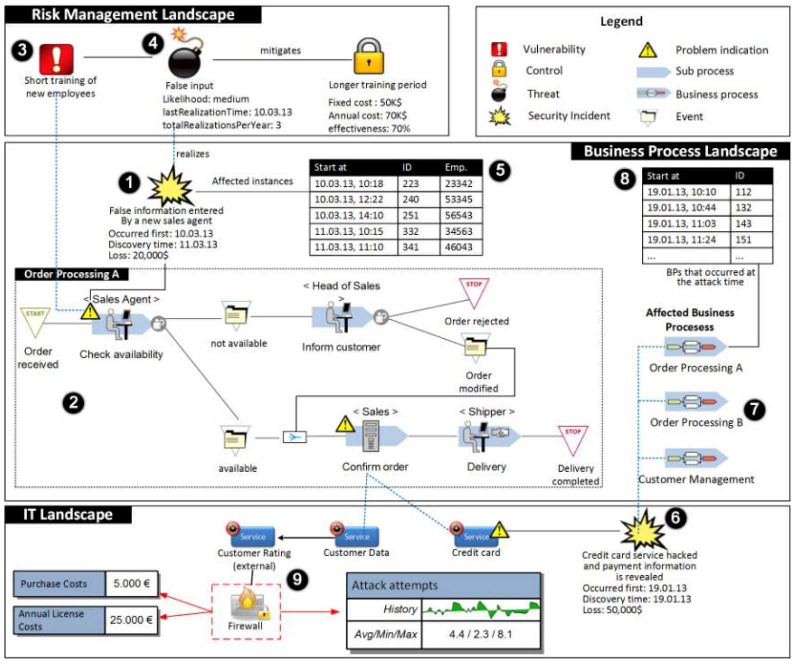
Our approach is based on and extends a multi-perspective enterprise modelling (MEMO) framework [8]. MEMO provides a number of domain specific modelling languages (DSML) to describe different aspects of the enterprise, for example, business processes, the organizational structure, IT resources and strategic 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.

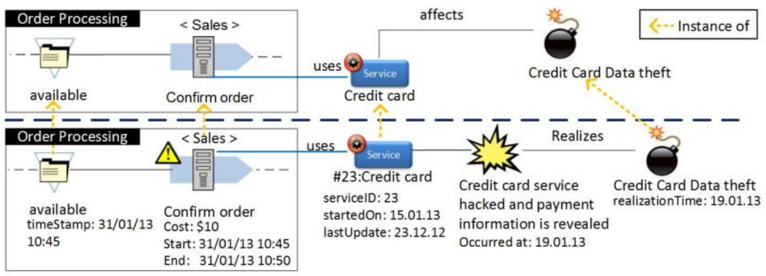


Fig. 2. Type level and instance level.

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.

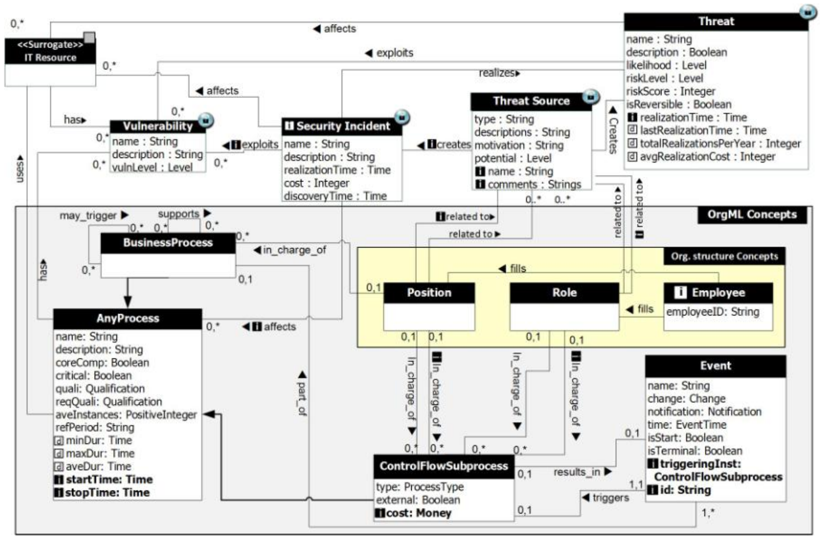


Fig. 3. An excerpt of the OrgML metamodel that illustrates the use of intrinsic features.

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)