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 beforehand, 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.

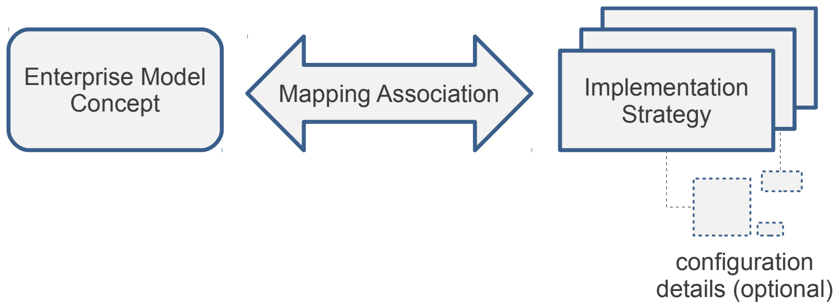


Fig. 1: Pattern of a single mapping association

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.

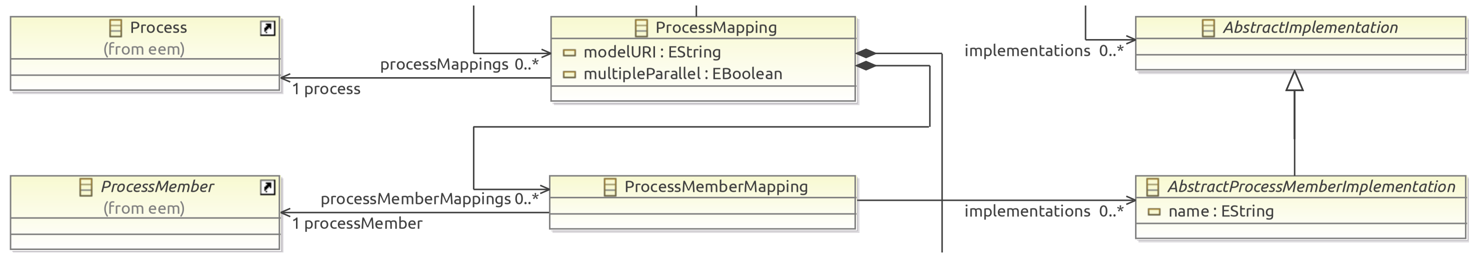
The combined use of a mapping model and implementation strategies provides dedicated methodical abstractions for coping with the requirements to bridge the abstraction gaps between conceptual enterprise model specifications, and ES implementations.

4 Example application

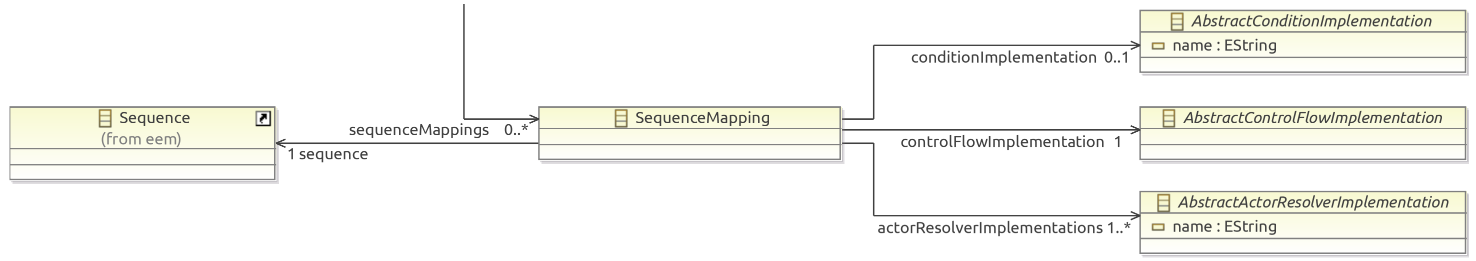
This section introduces a simple web shop example to demonstrate the use of the proposed approach. The example uses enterprise models in the MEMO [4] language as conceptual models describing the socio-technical environment of the software to be generated. The application architecture resembles a traditional 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


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

(b) Sequence mapping pattern

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

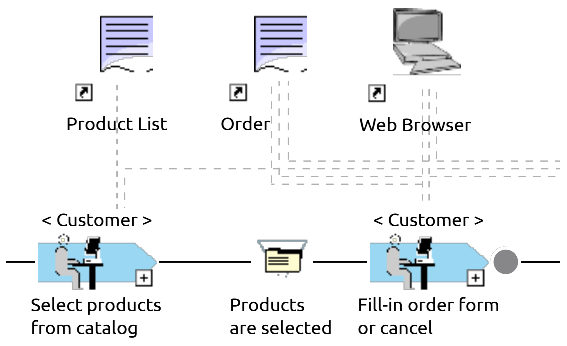


Fig. 3: Excerpt from a MEMO process control flow model referencing elements from other organization model perspectives

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.

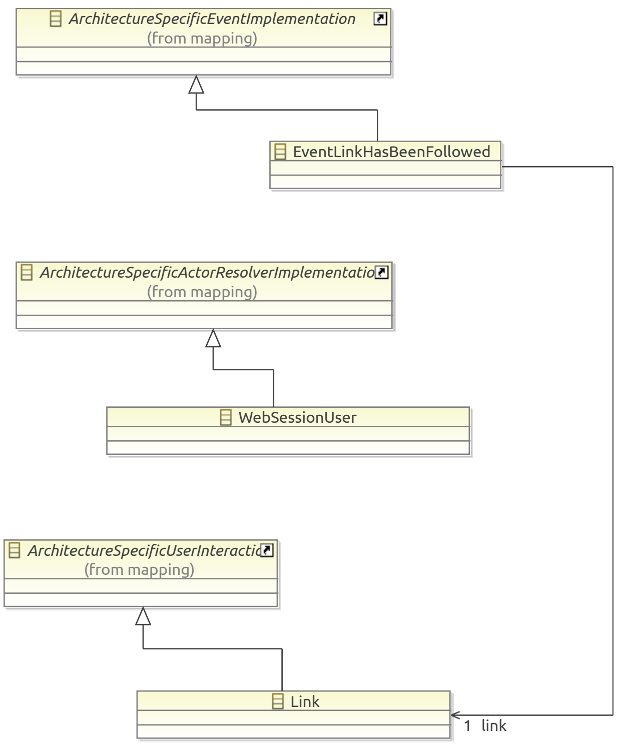

ArchitectureSpecificEventImplementation
(from mapping)
EventLinkHasBeenFollowed
ArchitectureSpecificActorResolverImplementatio
(from mapping)
WebSessionUser
ArchitectureSpecificUserInteracti
(from mapping)
Link
1 link

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.