Modeling the Transformation of Application Landscapes

Stefan Hofer

Modeling the Transformation of Application Landscapes

Stefan Hofer

Software Engineering Group, Department of Informatics, University of Hamburg, Germany
hofer@informatik.uni-hamburg.de

Abstract. Many of today's IT projects transform application landscapes. Transformation is a challenging task that has significant effect on an organization's business processes and the organization itself. Although models are necessary to accomplish this task, there are no specialized modeling approaches for transformation. We describe what such a specialized modeling approach should be capable of. This will allow the adaption of existing approaches and thus support the transformation of application landscapes.

Key words: application landscape, enterprise architecture, transformation, migration, co-evolution

1 Introduction

Enterprises use models of application landscapes for many purposes. One of them is to support the *transformation* of application landscapes like in the following example:

> An insurance company introduces a new customer relationship management (CRM) system which replaces a legacy application and provides new possibilities for handling documents. In addition, a back-office system replaces the previous custom-made software for commission calculation. Millions of data records need to be migrated to the new systems. Several applications (e.g. offer calculation, electronic application form, and bookkeeping) need to be adapted so they can exchange data with the new systems.
> Releasing all changes in a "big bang" approach seems too risky, so the company opts for an incremental transition in two steps:
> 1. Transition to an intermediate to-be landscape.
> 2. Transition to the final to-be landscape.
>
> The transformation affects the users because they have to adapt their work processes. For example, the "offer calculation" process has to be altered to avoid data duplication. Instead of entering the customer's data into the offer calculator, it has to be entered into the CRM system and then imported by the calculator.

Transformation may be triggered by business needs, legislation and strategic technological decisions. An organization, its business processes and its application landscape are interwoven and changes to one of them are likely to affect the others. This effect is called *co-evolution* (see [14]). Thus, transformation requires knowledge about applications and their dependencies, knowledge on how applications support business processes, and knowledge on how users work with applications. To acquire that knowledge, a vast amount of information has to be gathered and analyzed; information that changes frequently and cannot be gained by measuring and automated analysis only. Observations, interviews and assumptions add to the body of acquired knowledge. Models are a means to record this knowledge and make it accessible.

A large number of modeling tools[1] and notations support modeling of application landscapes. We claim that despite they have been in use to help transforming IT landscapes, they are not adjusted well enough for that purpose. For example, they give little guidance on how to create models from a huge amount of – possibly contradicting – information. Furthermore, it is often neglected in modeling that technical aspects and domain aspects of an application landscape need to be analyzed in conjunction with one another.

Our claim will be elaborated by presenting these contributions:

- We define what transformation of application landscapes means and how models are used in that area.
- We present six requirements that should be fulfilled by modeling approaches that are used to support application landscape transformation.
- We will show how well existing modeling approaches fulfill these requirements.

In conclusion, the reader will see that modeling approaches are currently not well suited for application landscape transformation and that adapted approaches would be useful.

The contributions presented in this paper are the result of ongoing research. Five real-world projects were analyzed to identify what characterizes "transformation" and to derive requirements for modeling approaches. The projects were conducted by companies from different domains, namely banking, logistics, and wholesale. The author was actively involved in three of those projects. Literature ([6] and [13]) served as additional input for the concept of transformation. All results presented in this paper were evaluated by seven experts that helped transform application landscapes as project manager, IT department manager, consultant, or software architect. None of the experts are affiliated with the author's employer or research group.

2 Transformation of Application Landscapes

The term *application landscape* "refers to the entirety of the business applications and their relationships to other elements, e.g. business processes in a company"

[1] For example, Buckl et al. list 41 of such tools in [2].

[4, p. 12]. More precisely, we will use this term only if the applications are used in the context of human work and some of the applications are directly used by people. As a counterexample, a group of applications that jointly and fully automatically carries out business processes is not considered an application landscape in the context of this paper.

Transformation is inevitable in the life cycle of application landscapes. We use the term to describe substantial, business-critical changes in an application landscape that have significant impact on an organization's business processes and on the people that work with the applications. Hence, transformation is a form of *co-evolution* which means "that the evolution of one domain is partially dependent on the evolution of the other [...], or that one domain changes in the context of the other." [14]

In this paper, we focus on the impact of transformation on business processes and on the people that execute them. This view was influenced by the concept of *application orientation* which is one of the main pillars of a software development approach called *Tools & Materials approach* (see [23]): "Application orientation focuses on software development, with respect to the future users, their tasks, and the business processes in which they are involved." [23, p. 4]. Accordingly, we aspire towards an application oriented approach to transformation of application landscapes. Yet, there are other aspects of application landscapes that are of importance for transformation, such as costs, maintainability, security, and scalability. They will, however, not be covered by this paper.

Transformation is usually carried out as a project that includes several types of activities:

- *Collect information*: Technical aspects (e.g. dependencies) and business aspects (e.g. supported business processes) of the application landscape have to be gathered. This is either done before or during modeling.
- *Evaluate and decide*: The goals of the transformation have to be defined and the current state of the application landscape has to be evaluated. For all affected applications, the necessary changes need to be identified.
- *Plan*: Planning activities for transformation are comparable with "regular" IT projects. A lot of tasks have to be aligned in order to fit into the overall roadmap.
- *Involve the organization*: Lots of communication is required to explain the goals, decisions and deadlines to all those affected by the transformation.
- *Execute*: The transformation is executed. The operations that constitute a transformation are:
  - bringing an application into service
  - changing an application
  - placing an application out of operation

  More accurately, these operations are relevant to transformation projects only if they affect several applications (e.g. by adding a new dependency to the application landscape). Hence, "end-to-end" tests of business processes are required to ensure that the application landscape works as expected.

In this paper, projects that include these activities and transform an application landscape are called *transformation projects*. Although the list describes what activities are typical for transformation projects, it may not be complete. As mentioned in Sect. 1, this list of activities was derived from real-world transformation projects and evaluated by experts.

3 Modeling in Transformation Projects

The requirements we will lay down in this paper aim at increasing the value that models provide to transformation projects. To understand what value that is and who benefits from it, we will discuss the following questions:

- *How* and *what for* are models used?
- *What kinds* of models are relevant for transformation projects?
- *What* is modeled?
- *Who* uses the models?

We will answer these questions in the following subsection. Examples for use of models in transformation projects conclude this section.

3.1 Characteristics of Modeling in Transformation Projects

In transformation projects, models are often used to evaluate the current state of an application landscape and to develop possible future states. The main purpose of the latter is to explore possibilities and to anticipate the consequences of transformation. This is of particular importance for the activities summarized as "evaluate and decide" in the previous section. In general, "the purpose of a model covers a variety of different intentions and aims such as perception support for understanding the application domain, explanation and demonstration [...], optimization of the origin, hypothesis verification through the model, construction of an artifact or of a program" [20, p. 86].

Transformation projects require models that depict the context in which an application landscape is used – the terminology of the domain, business processes and how the application landscape supports these processes. Models that represent a domain are called *conceptual models*. They may be interpreted as a "collection of specification statements relevant to some problem" [12, p. 42].

Other types of models used in transformation projects show the internal structure of an application landscape – its software systems and their dependencies. Models depict either what software systems and dependencies exist generally in the landscape or how they work together during the execution of certain business processes. Yet other types of models focus on the dependencies between software systems and the hardware they run on.

The activities described in Sect. 2 involve various stakeholders like domain experts, IT experts, managers, and users. Since different kinds of models are commonly used to support these activities, modelers are relevant stakeholders

too. A stakeholder's view on the application landscape is shaped by their goals and activities and may differ substantially from other stakeholders' views. A view can be expressed with one or several models.

3.2 Examples

The following examples illustrate how models are used in transformation projects:

- As-is models of the structure of the application landscape foster discussion about the applications that might be affected by a transformation. Also, the models serve as a baseline for the development of to-be models that show possible future states of the application landscape's structure.
- As-is and to-be models are used to analyze which dependencies have to be added, deleted, or modified and which interfaces need to be changed or introduced.
- Some transformation projects are carried out in several releases, transitioning incrementally from the as-is state to the to-be state. Models are used to determine the technical and process-related dependencies. This allows to decide which changes will be carried out in which transition.
- Models are used to develop to-be processes that fit the to-be state of the application landscape. The transformation's consequences on the way users work with their applications are communicated with the help of such models.
- To-be models allow cross-checking the planned to-be state of the application landscape and its intended use. Such cross-checks are useful to detect shortcomings in the planned to-be states.
- Models are used to develop test cases that ensure that the application landscape supports the business processes as expected – during transitions and in the final to-be state.

Creation and use of models like in these examples require a suitable modeling approach. In the next section, we will describe what constitutes such an approach.

4 Six Requirements for a Modeling Approach for Transformation Projects

A graphical modeling approach should provide a modeling language and a modeling procedure (see [10]). In addition, we consider tool support essential for use in real-world projects. Furthermore, any modeling approach should provide means to achieve high quality of models. Quality attributes that generally apply to models are for example correctness, consistency and comprehensibility (see [15]).

We claim that there are additional requirements to modeling approaches that are due to the nature of transformation projects. Also, we argue that stakeholders would benefit from using a modeling approach that meets these

requirements. The following collection of requirements was compiled from an application-oriented point of view. Hence, the requirements focus primarily on how application landscape, business processes and the people who execute them are intertwined. 12 requirements – derived from the same five real-world projects described in Sect. 1 – were evaluated by the same experts who assessed what activities are typical for transformation projects. The six requirements rated as most relevant are:

Requirement 1: The modeling approach should make the available information manageable.

A model is an abstraction of an original (e.g. an application landscape) that contains only selected properties of that original. It is generally assumed that all the properties of the original are known and that the selection of properties that are represented in the model is driven by the goal of the model. However, this assumption does not hold in the context of application landscapes. There is extensive information available about an application landscape and its use. In large organizations this information changes constantly. Since transformation projects do not only require technical information but also information on how the landscape is used, the problem is aggravated:

> "The only complete specification of a system is the system itself, and the only complete specification of the use of a system is an infinite log of its actual use [...]." [5, p. 255].

For these reasons, it is not possible to create a "complete" model within limited time and effort. Information on complex application landscapes is both incomplete and beyond comprehensibility. Therefore, a modeling approach should provide some guidance on how to create and use models in such an environment.

Requirement 2: The modeling approach should be able to express contradictions.

It is tempting to assume that models of application landscapes show facts. After all, applications are technical systems and information about them can be measured or at least gathered automatically. But even if that were the case with all the technical information, some interpretation is necessary to create models from that information. In addition to technical information, transformation projects require information about how people use an application landscape (see Sect. 3). In some cases, log files can be used to analyze application usage (see [21]). But to a certain extent modelers have to rely on interviews and observations. Hence, modeling in the context of application landscapes is a *social process*. It has to account for the personal goals and needs of the people involved. For example, a stakeholder might present an assumption as a fact, withhold information, or (consciously or not) falsify information. In such an environment, contradictions will emerge.

Requirement 3: The modeling approach should be able to express *how* an application landscape supports business processes.

In transformation projects, stakeholders use models to understand which business processes depend on which applications. However, this information does not suffice to plan how work processes and organizational units are affected by the transformation. This requires knowledge of *how exactly* applications are used in business processes. In particular, this information is needed for testing.

Requirement 4: The modeling approach should be able to express an application landscape's dependencies even for business processes that use several applications and are carried out by more than one organizational unit.

Models of such processes are prone for errors as the people that are involved in them usually are familiar with fragments of the process only. The information they can provide on how application landscapes and business processes work together may be inaccurate. The division of work results in little understanding of overall processes. Modeling approaches should consider that.

Requirement 5: The modeling approach should be able to express dependencies between applications even if they cannot be mapped to technical interfaces.

There are various kinds of dependencies in an application landscape like calls (of functions, methods etc.), shared data, shared hardware (e.g. same network segment), and shared runtime environments (e.g. virtual machine). At least in theory, some information about dependencies can be gathered by analyzing interface access. This increases confidence in the information that is depicted in a model. But there is another kind of dependency that does not correspond to any technical interface and can only be recognized by analyzing business processes: Dependency by time and order. For example, a stakeholder may use the results of one task (carried out with application A) to decide, how to carry out another task with application B. If application A was to be changed or replaced in a transformation project, the way how stakeholders use application B could be affected. Such dependencies have to be considered in transformation projects and in modeling.

Requirement 6: The modeling approach should be able to express how an application landscape changes over time.

As illustrated by the example in Sect. 1, application landscapes undergo a series of changes until their desired state is reached at the end of a transformation project. Thus, it is important for stakeholders to know how and when changes will affect their work processes. This is not just a matter of project planning but of communication.

In the next section, we will evaluate how well existing modeling approaches meet the requirements.

5 Evaluation of Existing Modeling Approaches

The goal of this evaluation is to test our claim that existing approaches are not suited well enough for transformation projects. As mentioned in Sect. 1 there is a large number of modeling languages, frameworks and tools that deal with application landscapes. Our evaluation focuses on approaches that fulfill certain criteria or are open for extension so that the criteria could be fulfilled by adapting the approach. The criteria are:

- The approach consists of a modeling notation, a methodology for creation and use of models, and tool support.
- The approach can express different views on an application landscape (as described in Sect. 3). It is able to depict technical information and domain knowledge.

Since these criteria are possibly matched by many professional modeling tools we chose one tool as a representative and omitted other commercial products from the evaluation.

In the following sub-sections we will give a short introduction to the approaches that were included in the evaluation. The section is concluded with the results of the evaluation.

5.1 UML

The *Unified Modeling Language* (*UML*, see [19]) has its origins in the area of software engineering but lays claim to be much more versatile:

> "UML is a general purpose language, that is expected to be customized for a wide variety of domains" [17, p. 211].

UML is adaptable and can be modified to depict application landscapes. Such an adaption is reported by Heberling et al. in [8]. Countless modeling tools support UML. However, UML does not include a methodology for how to create or use models:

> "[UML] is methodology-independent. Regardless of the methodology that you use to perform your analysis and design, you can use UML to express the results." [16]

UML was included in the evaluation for its extensibility and widespread adoption.

5.2 ArchiMate

ArchiMate was developed to model enterprise architectures (which application landscapes are a part of). It does not include a methodology but an informal description of usage scenarios that are expressed as a collection of *viewpoints* (see [18]). Since version 2.0, ArchiMate can be used in combination with *The*

Open Group Architecture Framework (*TOGAF*, see [9]) for enterprise architecture development. However, TOGAF does not provide any guidelines on how to create or use ArchiMate models.

As UML, ArchiMate is a standardized and established modeling language that is supported by many tools and was thus included in the evaluation.

5.3 EAM Pattern Catalog

The EAM patterns described in the *Enterprise Architecture Management Pattern Catalog* [2] are a collection of problems and fitting solution patterns in the area of IT enterprise architecture management. This descriptive approach presents best practices for analysis, graphical representation, and information modeling. The patterns aim at enhancing existing approaches. For example, the catalog's *methodology patterns* concretize TOGAF and the *viewpoint patterns* show applications of UML, ArchiMate, and *software maps* (see [3]). Due to the nature of this approach, the criterion of tool support can be neglected.

Methodology patterns describe the use of models. Since this approach tends to interpret graphical models as mere visualization of an underlying information model, there is no guidance on how to create models. Yet, this approach was included in the evaluation because it is grounded in practice and many of the patterns are implemented by professional modeling tools.

5.4 MEMO

Multi-perspective enterprise modeling (*MEMO*, see [7]) is a framework for the development of domain specific modeling languages for use in enterprise modeling. Examples of such languages are the business process modeling language *OrgML* and the *IT Modeling Language* (*ITML*, see [11]) for IT management.

All MEMO languages share a common meta-model that ensures interoperability of languages. MEMO is meant to be extended and provides a meta-methodology for modeling that can be used to develop a language-specific methodology. A tool prototype demonstrates that tool support is feasible.

MEMO was included in the evaluation because it provides adequate concepts to create a modeling approach for transformation projects.

5.5 ADOit 5.0

ADOit is a commercial tool for architecture management developed by the BOC Group [1]. Its meta-model can be adapted to meet the requirements of transformation projects. Yet, already the meta-model's default configuration suffices to model technical and domain aspects of application landscapes. Although ADOit is not coupled to a specific methodology for creation and use of models there are predefined views and queries that suggest certain usage scenarios.

ADOit was included in the evaluation because of its adaptable meta-model and the availability of a free-of-charge community edition. The tool is relevant

to the German market and documentation is available. The vendor proved to be accessible for discussion. However, it should be noted that these criteria might also be met by other vendors (and their tools, respectively).

5.6 BEN

The *Business Engineering Navigator* (*BEN*, see [22]) developed at the University of St. Gallen is an approach for managing IT enterprise architecture. It offers support for modeling IT and its relation to business processes. BEN provides some guidance on how to analyze models of application landscapes and tool support is available. It was included in the evaluation because it covers application landscapes from a business engineering perspective.

5.7 Evaluation

In this section, we rate how well the approaches that were described briefly in the preceding sections meet the requirements laid down in Sect. 4. The results are summed up in Table 1 and explained in the remainder of this section. We use the following values for the rating:

- ++ Requirement is fulfilled.
- \+ Rudimentary but insufficient solution for requirement.
- = Requirement not fulfilled but approach offers means for enhancement.
- – Requirement not fulfilled.

Table 1. Results of the Evaluation

Requirement	UML	ArchiMate	EAM-Patterns	MEMO	ADOit	BEN
1 manageable information	=	=	+	=	=	–
2 contradictions	+	–	–	–	=	–
3 business process support	++	+	+	+	+	–
4 dependencies across boundaries	++	++	+	+	+	+
5 non-technical dependencies	=	=	–	=	=	–
6 change over time	–	+	+	=	++	+

Requirement 1 (manageable information): To make the amount of information manageable, one can simply include less of it in a model – either by constricting the modeling language or by switching to a more coarse-grained, generalized perspective that omits detail. Even if a modeling approach does not support these mechanisms explicitly they can always be applied by convention. If an approach enforces such conventions (like UML by providing *profiles*, see [19]) we rated it with "=". The EAM patterns approach was rated "+" because for every concern it addresses corresponding *information model patterns*. These patterns help a modeler to identify which information is needed to provide a model for the given concern.

Requirement 2 (contradictions): Contradictions will catch a stakeholder's eye during modeling or during use of models. The approaches presented above give little guidance on how to model and contradictions are not mentioned at all. However, some approaches offer generic means to at least annotate contradictions. UML provides *stereotypes* and *tagged values* (see [19]) for that purpose and ADOit's meta-model could be adapted to achieve a similar possibility.

Requirement 3 (business process support): Several approaches can express which activities an application is involved in. UML offers *activity diagrams* and *partitions* (see [19], often called "swim lanes" in other approaches). ADOit allows modelers to link activities to applications. In addition, it provides several queries to analyze the usage of an application landscape in business processes. ArchiMate's *business layer* only allows for modeling of coarse-grained process chains but they can be linked with *services* and *components* in the *application layer*. Several similar viewpoint patterns can be found in the EAM pattern catalog. MEMO's meta-model includes a relationship between business processes and applications (see [7]) which allows for the creation of a MEMO language that fulfills this requirement.

Requirement 4 (dependencies across boundaries): The dependencies described in this requirement can be expressed with UML's activity diagrams and ArchiMate (for example with the *introductory* and *layered* viewpoint described in [18]). An overview of such dependencies is provided by so-called *Process Support Maps* that show which applications support which business processes. Such visualizations are described in the EAM patterns catalog (e.g. viewpoint patterns V-29 and V-30, see [2]), in ADOit, and in BEN.

Requirement 5 (non-technical dependencies): To express dependencies between applications that cannot be mapped to technical interfaces, both modeling language and methodology have to be considered. Since no approach provides both, none was rated "+" or "++". Approaches with extensive possibilities to depict dependencies or generic relation types were rated "=".

Requirement 6 (change over time): The approaches included in the evaluation provide three different means to express how an application landscape changes over time:

The first is provided by EAM patterns, BEN, and ADOit which allow to model the *life cycle* of applications. The second possibility is a model of the application landscape that combines as-is and to-be applications. ArchiMate (viewpoint "Implementation and Migration", see [18]) and ADOit support this kind of model. Additionally, ADOit offers a time-based filter that helps tracking changes over time. Third, BEN's tool support allows for pairwise comparison of models. This functionality can be used to compare as-is and to-be models with each other.

6 Conclusions and Further Work

In this paper, we introduced a type of project that deals with extensive changes to an organization's application landscape – *transformation projects*. We look at transformation from an *application-oriented* point of view that focuses on how an application landscape is used in business processes. This point of view relies on models of the application landscape and its use.

We argued that specific requirements for modeling exist in this area and that it would be beneficial for modeling approaches to meet these requirements. This would improve their suitability for transformation projects. Six such requirements were presented. An evaluation of existing approaches showed that some approaches provide means to fulfill some of these requirements. None of the approaches met all the requirements. However, it is not necessary to invent a completely new modeling approach for transformation projects. We plan to show that existing approaches can be complemented so that they are better suited for transformation projects. Therefore, we will create an enhanced approach to show the feasibility of this idea. The enhanced approach will then be evaluated to test our initial claim: Fulfilling the presented requirements leads to a more useful modeling approach for transformation projects.

References

1. ADOit Product Website, http://www.boc-group.com/products/adoit/
2. Buckl, S., Ernst, A., Lankes, J., Matthes, F.: Enterprise Architecture Management Pattern Catalog. Technical report, Technical University Munich (2008)
3. Buckl, S., Ernst, A., Lankes, J., Schweda, C., Wittenburg, A.: Generating Visualizations of Enterprise Architectures using Model Transformations. In: Reichert, M., Strecker, S., Turowski, K. (eds.) EMISA 2007. LNI, vol. P-119 pp. 33–46. GI, Bonn (2007)
4. Buckl, S., Ernst, A., Matthes, F., Schweda, C.: An Information Model for Managed Application Landscape Evolution. In: Journal of Enterprise Architecture, vol. 5, pp. 12–26 (2009)
5. Carroll, J.: Making Use. Scenario-Based Design of Human-Computer Interaction. MIT Press (2000)
6. Engels, G., Hess, A., Humm, B., Jung, O., Lohmann, P., Richter, J., Voß, M., Willkomm, J.: Quasar Enterprise. dpunkt.Verlag, Heidelberg (2008)
7. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. In: Journal of Software and Systems Modeling, online first article, Springer (2012)
8. Heberling, M., Maier, C., Tens, T.: Visual Modelling and Managing the Software Architecture Landscape in a large Enterprise by an Extension of the UML. In: Position Papers of the 2nd Workshop on Domain-Specific Visual Languages at OOPSLA (2002)
9. Jonkers H., Band, I., Quartel, D., Franken, H., Adams, M., Haviland, P., Proper, E.: Using the TOGAF 9.1 Architecture Content Framework with the ArchiMate 2.0 Modeling Language. Technical report, The Open Group (2012)

10. Karagiannis, D., Kühn, H.: Metamodelling Platforms. In: Bauknecht, K., Min Toja A., Quirchmayer, G. (eds.) Third International Conference EC-Web 2002. LNCS. vol. 2455, pp. 182–195. Springer, Berlin (2002)
11. Kirchner, L.: Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements. Technical report, University Duisburg-Essen (2007)
12. Lindland, O., Sindre, G., Solvberg, A.: Understanding quality in conceptual modeling. IEEE Software, vol. 11, pp. 42–49 (1994)
13. Matthes, F., Wittenburg, A.: Softwarekarten zur Visualisierung von Anwendungslandschaften und ihren Aspekten. Technical report, Technical University Munich (2004)
14. Mitleton-Kelly, E., Papaefthimiou, M.: Co-Evolution Of Diverse Elements Interacting Within A Social Ecosystem. In: International Workshop on Feedback and Evolution in Software and Business Processes (2000)
15. Mohagheghi, P., Dehlen V., Neple, T.: Towards a Tool-Supported Quality Model for Model-Driven Engineering. In: Chaudron, M. (ed.) Models in Software Engineering. LNCS, vol. 5421, pp.74–88. Springer, Berlin (2008)
16. Object Management Group: Introduction to OMG's Unified Modeling Language, http://www.omg.org/gettingstarted/what_is_uml.htm
17. Object Management Group: OMG Unified Modeling Language (OMG UML) Infrastructure 2.4.1, http://www.omg.org/spec/UML/2.4.1/
18. The Open Group: N116 ArchiMate 2.0 Viewpoints Reference Card, https://www2.opengroup.org/ogsys/catalog/n116
19. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language reference manual. Addison-Wesley, New York (2005)
20. Thalheim, B.: The Science and Art of Conceptual Modelling. Transactions on Large-Scale Data- and Knowledge-Centered Systems, vol. 6, pp. 76–105 (2012)
21. Van der Aalst, W.: Intra- and Inter-Organizational Process Mining: Discovering Processes Within and Between Organizations. In: Johannesson, P., Krogstie, J., Opdahl, A. L. (eds.) The Practice of Enterprise Modeling. LNBIP, vol. 92, pp. 1–11 (2011)
22. Winter, R.: Business Engineering Navigator. Springer, Berlin (2010)
23. Züllighoven, H.: Object-Oriented Construction Handbook. dpunkt.verlag, Heidelberg (2005)