Fakultät für Informatik
der Technischen Universität München

A Pattern-based Approach to Enterprise Architecture Management

Alexander M. Ernst

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Martin Bichler
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Florian Matthes
2.	Univ.-Prof. Dr. Dimitris Karagiannis, Universität Wien / Österreich

Die Dissertation wurde am 09.09.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03.03.2010 angenommen.

Abstract

Patterns have proven to be a useful and practical approach to address complexity in various domains like software engineering, project management, or architecture. They document general, reusable proven practice solutions to common problems in a given context. Thereby they identify driving forces, known usages, and consequences. This thesis presents a pattern-based approach to Enterprise Architecture Management (EAM).

EAM is about the alignment of business and IT, which is a major issue of today's medium and large size enterprises. Exemplary reasons for this are demands for higher flexibility to support new business demands or higher agility to adapt to new market requirements. Additionally, there are legal regulations like the Sarbanes Oxley Act, which require a documentation of the enterprise architecture of a company. Different methodologies to support EAM have been developed by academia and practice with all of them having certain drawbacks like being too exhaustive or being too company-specific.

This dissertation investigates the use of patterns for EAM, following a concern-driven approach, to overcome these drawbacks. Three kinds of EAM patterns are introduced. Methodology patterns (M-Patterns) documenting methodologies to address typical problems in EAM in a step-wise manner. Viewpoint patterns (V-Patterns) detailing best practice visualizations, which can be used in the execution of M-Patterns. Information model patterns (I-Patterns) recommend information model fragments that have proven to be useful in the documentation of enterprise architectures. In contrast to EAM patterns, EAM anti patterns document solutions, which have proven not to work in practice to prevent typical mistakes in EAM.

Experiences from academia and practice have been collected, structured, and made available for future reuse in the EAM Pattern Catalog using EAM patterns and EAM anti patterns. Four usage scenarios for the catalog are elaborated and applied in this thesis: Develop an enterprise-specific EAM, evolve and assess an existing EAM, specify EAM requirements, and conduct scientific research.

Six industrial case studies, one academic workshop, a lecture at the Technische Universität München, and a survey show that the approach is adopted by academia as well as by practitioners and is able to pave the way for future developments in EAM.

Zusammenfassung

Patterns haben sich in verschiedenen Domänen, wie dem Software Engineering, dem Projektmanagement oder der Architektur, als ein hilfreicher und praktikabler Ansatz für den Umgang mit Komplexität bewährt. Sie dokumentieren allgemeingültige, wiederverwendbare und bewährte Lösungen zu wiederkehrenden Problemstellungen in einem gegebenen Kontext. Dabei identifizieren sie die auf die Lösung einwirkenden Kräfte, bekannte Nutzungen und zu berücksichtigende Konsequenzen. Diese Dissertation stellt einen Pattern-basierten Ansatz für das Enterprise Architecture Management (EAM) vor.

EAM beschäftigt sich mit dem aneinander ausrichten und unterstützen von Geschäft und IT. Dies ist heute eines der wichtigsten Themen mit dem sich mittelständische und große Unternehmen beschäftigen müssen. Exemplarische Gründe hierfür sind die Forderung nach einer größeren Flexibilität um neue Geschäftsanforderungen zu unterstützen oder eine größere Agilität um schneller auf neue Marktanforderungen zu reagieren. Darüber hinaus existieren gesetzliche Bestimmungen wie der Sarbanes Oxley Act, welche die Dokumentation der Unternehmensarchitektur verlangen. Verschiedene Methodiken zur Unterstützung von EAM wurden bereits von der Wissenschaft und der Praxis entwickelt. Bisherige Ansätze haben jedoch alle einige Nachteile, wie einen unüberschaubaren Umfang oder sie sind zu unternehmensspezifisch.

Diese Dissertation untersucht die Nutzung von Patterns für das EAM, unter Berücksichtigung eines Concern-getriebenen Ansatzes um diese Nachteile zu beseitigen. Drei Arten von EAM Patterns werden vorgestellt. Methodology-Patterns (M-Patterns) dokumentieren Methodiken um typische Probleme im EAM schrittweise zu lösen. Viewpoint-Patterns (V-Patterns) detaillieren best-practice Visualisierungen, die im Rahmen von M-Patterns genutzt werden. Informationsmodell-Patterns (I-Patterns) schlagen Informationsmodellfragmente vor, die sich für die Dokumentation von Unternehmensarchitekturen als nutzbringend bewährt haben. Im Gegensatz zu EAM Patterns dokumentieren EAM Anti Patterns Lösungen, die sich in der Praxis als nicht zielführend herausgestellt habe, um typische Fehler im EAM zu vermeiden.

Erfahrungen aus Wissenschaft und Praxis wurden gesammelt, strukturiert und für die zukünftige Nutzung in Form von EAM Patterns und EAM Anti Patterns im EAM Pattern Catalog zur Verfügung gestellt. Vier Nutzungsszenarien wurden für den EAM Pattern Catalog entwickelt und in dieser Arbeit angewandt: Entwicklung eines unternehmensspezifischen EAM, Weiterentwicklung und Bewertung eines existierenden EAM, Spezifikation von Anforderungen im EAM und Durchführung von wissenschaftlicher Forschung.

Sechs industrielle Fallstudien, ein akademischer Workshop, eine Vorlesung an der Technische Universität München und eine Umfrage zeigen, dass der EAM Pattern Ansatz sowohl von der Wissenschaft, als auch von der Praxis angewendet wird und das er als Grundstein für die weitere Entwicklung von EAM dienen kann.

Acknowledgment

This work was created during my activities as a research assistant at the Chair for Informatics 19 (sebis) at the Department of Informatics of the Technische Universität München. Here, I would like to thank all those, who supported me in my research.

First of all, I would like to thank my doctoral advisor, Prof. Dr. Florian Matthes, for providing the opportunity to work on an interesting research topic, and for his suggestions and advice that greatly contributed to the success of this work. I would also like to thank Prof. Dr. Dimitris Karagiannis, for our conversations about the subject and for being the second reviewer of my thesis.

This work is part of and based on sebis' research into enterprise architecture management and system cartography. Thus, I would like to thank my colleagues in the system cartography team for their patience in discussing ideas, developing approaches, writing publications, and reviewing texts. This cooperation greatly influenced my research on the pattern-based approach to EA management. My thanks go to Sabine Buckl, Dr. Josef Lankes, Christian M. Schweda, and Dr. André Wittenburg.

I would also like to thank the students which contributed to my research on EAM patterns. Jana Leitel provided me with an overview about existing EA frameworks. Martin Böhme, Thomas Dierl, Katharina Pflügler, and Alexander Schneider for applying the pattern-based approach to EA management in case studies at project partners of the research project. In addition, I want to thank Andreas Heusler and Gregor Bender for their help on the EAM Pattern Catalog Wiki.

Discussing my research with practitioners and applying EAM patterns in real world case studies greatly contributed to my research, allowing me to keep my work focused, and evaluate its results. Thus, my thanks also go to the practitioners who gave me the opportunity to discuss my work with them, and apply it in practice.

Last but not least, I am most grateful to my family, especially to my parents, for their support and constant encouragement during my doctoral work, and to my beloved wife, for her support, patience, and understanding.

Garching b. München, September 2009

Alexander M. Ernst

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction and Overview

Contents

Today's medium and large size enterprises increasingly have to care about the management of their information technology (IT). One of the reasons is that IT is more than only a supporting function for running the business of an enterprise. It has become an enabler for new business prospects. This is often referred to as the *alignment between business and IT*. Enterprise architecture (EA) management is a prominent and emerging approach promising to improve the aforementioned alignment [ARW08].The Open Group [Op04] presents a similar point of view. EA "enables to achieve the right balance between IT efficiency and business innovation. It enables managed innovation within the enterprise".
Complexity and scope are other challenges important for today's enterprises. Leist and Zellner [LZ06] state that "as information systems and technologies grow in complexity and scope, the need for a coherent and comprehensive modeling approach becomes of paramount importance". The growing complexity and required agility of enterprises in the 21st century is also emphasized in [FG98, DH08a] resulting in a growing demand to address those issues. In addition, quality and timeliness require enterprises to face the issue of EA [Za97].
The importance of managing the EA does not only emerge from within the enterprise. It is further promoted by legal regulations like the Sarbanes Oxley Act [Co02], which require a documentation of the EA of a company.

Zachman in [Za97] summarizes the importance of EA: "The issue of enterprise architecture is critical to the survival of every enterprise of any substance in the moderately near future". An indication for the validity of this statement can be found in [LW04a], which presents an overview about literature on EA and concludes that the number of publications in this area has continually increased.
This thesis proposes a pattern-based approach for improving currently existing approaches for EA management by introducing characteristics of engineering to EA management in order to support academia and practice in the future development of EA management.
The intended audience for this thesis are people concerned with managing the IT of an enterprise ranging from software developers looking for an overview how their business applications are embedded in the EA, via enterprise architects trying to document and plan the future development of the EA, to business managers, which need to have an overview about the EA, its future development, and its support for business requirements.

1.1. Motivation

The growing importance of managing the EA led to the development of a multitude of methods for EA management by the academic community (cf. [Fr02, We03, La05a, Wi07a, Wi07b, Ai09]), standardization bodies (cf. [Do07, Op09a]), practitioners (cf. [De06, Ca07a, En08]), or tool vendors (cf. [se05, Ma08a]).
All of these methods have their specific advantages and disadvantages, but there is none, which has yet been accepted as a kind of standard approach. Different reasons for this absence of a standard are discussed in the EA management community and are documented in reports about the state of the art in EA management e.g. by Buckl et al. [Bu09d]. One reason is that methods for EA management are documented on a too generic level to solve problems occurring in the EA management context. Existing methods are also often too company-specific and can therefore not be used as a general solution to existing problems. Making the situation even more difficult, they cannot adequately be adapted to the company-specific requirements. Especially the last reasons are important, because there is no one size fits all solution to EA management. The requirements are often too company-specific to be addressed by a monolithic reference method.
Besides the issues mentioned above, EA is a rather young discipline, which is more an art based on experiences [We03] than based on dedicated methods, which are used for systematically addressing problems as in mature engineering disciplines. A few engineering-oriented approaches to EA management are currently in development, see e.g. the business engineering navigator by Aier et al. [Ai09], but none of them can be considered to be widely accepted in the EA community. Typical frameworks for managing the EA also only partially contribute to the solution of the aforementioned problems as they may be helpful for an architect in his daily work, but knowing them does not make one an architect, because experience and knowledge is required to instantiate, i.e. apply a framework [Mo09b]. Therefore, an approach to address these problems is required.

1.2. Contribution

The goal of this thesis is to analyze existing approaches for EA management concerning their weaknesses and drawbacks, which restrict their application, in order to develop an engineering-based approach to EA management. The analysis results yield the thesis' two research challenges. On the one hand the available knowledge about EA management in academia and practice had to be collected, documented, and made available in a form that supports the EA management approach. On the other hand the principles of an engineering method had to be derived and applied to EA management, a discipline where their application is currently limited.
A prominent approach to document knowledge in software engineering are patterns, which originally were introduced in architecture. In architecture patterns are used to document recurring solutions to common problems in a given context. Alexander [Al79] states that "every person has a pattern language in his mind [...] At the moment when a person is faced with an act of design, he does not have time to think about it from scratch." This way of thinking is widely-used in engineering and can also be applied to EA management. Another idea of Alexander is applied in this thesis. It is the idea that "each concrete building problem has a language" [Al79]. In the case of this thesis the building problem is EA management, which is addressed by a pattern language. This pattern language for EA management is the contribution of this thesis concerning the first research challenge.
Concerning the second research challenge the contribution of this thesis is a method for EA management, based on principles from mature engineering disciplines, like mechanical engineering or architecture, which is useful for academia and practitioners. In engineering well accepted and utilized methods to address typical recurring problems are available. This systematic approach is applied to EA management and may help to improve the maturity of EA management. The constituents of the method for EA management – models, languages, and processes and roles – are thereby documented as patterns.
Knowledge about proven practice solutions from academia and practice has been collected, structured, and made available for reuse by novice and experienced users in form of the *Enterprise Architecture Management (EAM) Pattern Catalog* using *EAM patterns* and *EAM anti patterns*. This addresses the second research challenge and constitutes the second contribution of this thesis. The development of the EAM Pattern Catalog took 16 months and resulted in the EAM Pattern Catalog including more than 320 pages and 120 patterns. The EAM Pattern Catalog is the basis of the pattern language for EA management, which has been made available in a wiki and includes 164 (as of 2009-09-05) patterns. The EAM Pattern Catalog Wiki features 385 registered users and is used to revise existing patterns and to document additional ones in a community-based approach.
EAM patterns make up the building blocks, which can be integrated to a company-specific and concern-oriented approach to EA management, based on proven practices from academia and practice. The patterns can also be used to inspire and assess existing EA management approaches or to specify goals and requirements for EA management. Due to the self-containedness of patterns, they foster utilization and iterative extension by academia. Six industrial case studies, one academic workshop, a lecture at the Technische Universität München, and a survey show the adoption and the usefulness of the pattern-based approach to EA management.

1.3. Research Approach

Different approaches to conduct research are known in information systems (IS) research. The goal of these approaches is to provide rigor and relevance to research. Hevner et al. [He04] distinguish between two distinct paradigms. *Behavioral science* seeks to develop and verify theories that explain or predict human or organizational behavior in order to comprehend reality. By contrast, *design science* seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts to shape and/or change reality. This paradigm roots in engineering and uses a problem-solving process. Thereby, the knowledge and understanding of a design problem is gained by building and applying an artifact.
According to March, Salvatore, and Hevner et al. [MS95, He04], design research creates four types of artifacts.

Constructs are the vocabulary and symbols, which are used to describe problems within the domain and to specify their solutions.

Models are the abstractions and representations of reality used to describe tasks, situations, etc.

Methods are algorithms and practices, used to perform goal-directed activities.

Instantiations are implemented and prototypic systems intended to perform certain tasks.

Braun et al. [Br05a] determined that "regarding the basic research approach, method construction is clearly part of design science". This result and the description of design science by [He04] as "design science addresses research through the building and evaluation of artifacts designed to meet the identified business need", lead to the conclusion that a design science approach should be used in this thesis. Therefore, the first research challenge – document available knowledge in EA management – is best addressed in a design science fashion. Patterns for EA management are documented and evaluated in order to fulfill identified needs.
As a result, the type of artifacts presented above can be mapped to constituents of this thesis. In this thesis the patterns represent the constructs that are used to describe problems and corresponding solutions (see Chapter 4). These can be combined to higher level models (see Section 5.4), that are used by the developed pattern-based engineering method to EA management, which addresses concerns to achieve certain goals. The instantiation can be found in form of six case studies and a lecture at the Technische Universität München, applying the developed approach (see Chapter 6).
Hevner et al. [He04] provide seven design science research guidelines to assist researchers, reviewers, editors, and readers to understand the requirements for effective design-science research. The following listing presents these guidelines and shows how they are considered in this thesis.

Design as an artifact: The main result of this thesis are the documented patterns, which are included in the EAM Pattern Catalog. They are complemented by guidelines for their utilization and future development. Patterns and guidelines make up the artifacts of this thesis.

Problem relevance: The problem relevance was demonstrated in Section 1.1. [LW04a, ARW08] summarize the importance of EA management by providing an overview about EA and [Bu09d, Ma09] show that currently available approaches still retain potential for improvement. Benbasat and Zmud [BZ99] extend Hefner's criteria of relevance by demanding that IS research should focus on current technologies and business issues. This extended criterion is fulfilled by the foundation of this thesis (see Chapters 2 and 3) introducing EA management and patterns, as well as by the application of the pattern-based approach to EA management presented in Chapter 6 and by its evaluation in Chapter 7.

Design evaluation: The evaluation of the pattern-based approach to EA management is presented in Chapter 6 based on six case studies, an academic workshop, a lecture at the Technische Universität München. This is complemented by a comparison to existing EA management approaches and an online survey in Chapter 7.

Research contributions: The patterns documented in the course of this thesis, by collecting, filtering, and aggregating knowledge about EA management are the main research contribution. Their application in Chapter 6 shows that the developed pattern-based approach is applicable and valuable for academia and practice. A second contribution is the analysis of characteristics of mature engineering disciplines and their application on a method for EA management using a pattern-based approach.

Research rigor: The research rigor is given, as this thesis was conducted according to the design research process presented below.

Design as a search process: Documenting patterns is no one-time development approach. Patterns are typically mined, documented, and continually revised by a community. Therefore, a community has been initiated as part of the EAM Pattern Catalog Wiki (see Section 5.6). In addition, the evaluation of the pattern-based approach to EA management led to future enhancements and developments, which are presented in the outlook of this thesis (see Section 8.2).

Communication of research: The results of this thesis have been published and presented at various academic and non-academic conferences see e.g. [Bu07a, Bu08a, Bu08b, Er08, Bu09a, Bu09e, Bu09f, Bu09g, La09a]. This is complemented by the EAM Pattern Catalog Wiki providing the developed patterns for reuse, modification, and extension.

Summarizing, it can be said that this thesis complies with the guidelines proposed by Hevner et al. [He04] providing rigor to the research conducted in this thesis.
As mentioned before, the *design science research process* proposed by Offermann et al. [Of09] is used as a basis for this thesis. The research process is based on a combination

of qualitative and quantitative research methods. The goal of this combination is to "ensure relevance and scientific rigour by the interaction of practitioners and researchers, thereby improving the quality of created artefacts" [Of09]. In order to achieve this Offermann et al. [Of09] state that "empirical evaluations are best suited for generating accurate insight". In addition results should be presented to the observed practitioners to be able to include recent developments. Figure 1.1 shows the research process used in this thesis, which complies to the process proposed by [Of09]. Ellipses depict steps in the research process, arrows indicate transitions between these steps. Dotted lines indicate less frequently used transitions. The transitions from "case study / action research" depicted with dotted lines are exemplarily included. Similar transitions are also possible from "expert survey" and "laboratory experiment". The execution of this research process produces design science research results.

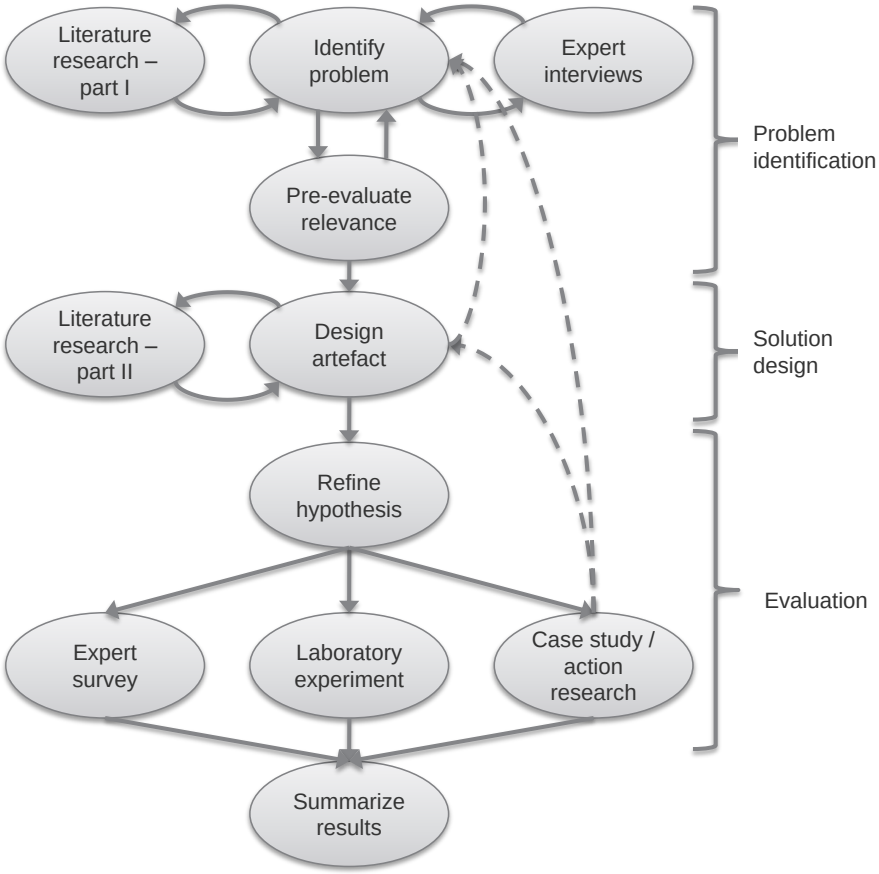

Figure 1.1.: Research process of this thesis based on [Of09]

The research process is divided into three main parts: Problem identification, solution design, and evaluation. These parts and the enclosed steps are described in the following, including the results achieved in each step.
Problem identification is the first process step, highly important to ensure the practical relevance of the problem. This demand is addressed in the research guidelines presented

above. Summarizing it can be said that EA management is in the domain of IS research and the practical relevance is given for this thesis.
For this thesis, available EA management approaches are analyzed, discussed, and shortcomings identified. In addition, problems arising in practical settings of EA management are identified in discussions with project partners and in interviews conducted as part of the *Enterprise Architecture Management Viewpoint Survey*. This is complemented by an extensive literature survey to analyze currently available approaches for EA management, see e.g. [LW04a], [Le07], etc. to ensure research rigor. The general idea to address problems in current EA management approaches with a pattern-based approach to EA management is published in Buckl et al. [Bu07a]. This leads to the general research hypothesis of this thesis.

> If a pattern-based approach to EA management is used, the results are superior to results of other EA management approaches.

Superior in this case means that the results do better fit the needs and address the concerns of a company occurring in the context of EA management.
The next step is the *pre-evaluation of relevance*, of the identified problem. In this thesis no pre-evaluation is conducted, because most companies at that time, if any, used a custom developed approach to EA management. See e.g. Buckl et al. [Bu09d] or Maicher et al. [Ma09] for a discussion on the utilization and utility of available EA management approaches. Nevertheless, the number of available approaches to EA management, see e.g. [LW04a, ARW08] and Section 7.1 indicate that they are regarded to be helpful in practice.
The second part is *solution design*. Using the results of the problem identification phase a pattern-based method for EA management, called EAM Pattern Catalog, is developed corresponding to the design artifact. The creation of the EAM Pattern Catalog complies to a "creative engineering process" [Of09].
Existing literature for managing EA, as well as information from project partners and additional practitioners is studied to identify proven practices in EA management. The results are documented in the initial version of the EAM Pattern Catalog, as a way to *design the artifact*. For more details about the creation of the EAM Pattern Catalog see Chapter 5.
For easier evaluation of the pattern-based approach to EA management the general hypothesis is refined and split into smaller hypothesis:

- The pattern-based approach to EA management is better applicable than other methods for EA management.
- The pattern-based approach to EA management is applicable in academia and practice.
- The pattern-based approach to EA management is useful for academia and practitioners.

The evaluation of the approach is conducted as an *expert survey*. Participants of the survey can rate the patterns for their applicability and utility in academia and practice. The results of the survey are used in the redesign of the design artifact, which leads to

version 1.0 of the EAM Pattern Catalog [Bu08a]. Afterwards, the EAM Pattern Catalog is made available to public in form of a wiki [se09a]. Additional articles (cf. [Bu07a, Bu08b, Er08, Bu09a, Bu09e, Bu09f, Bu09g, La09a, Mo09a] etc.) on patterns in EA management are published for further evaluation and communication of the idea.
The EAM Pattern Catalog is once more evaluated in six *case studies* (see Chapter 6) in order to prove relevance, applicability, and utility of the pattern-based approach to EA management. In addition, an *expert survey*, which is presented in Section 7.3, is conducted. This survey also compares the pattern-based approach to EA management to other existing approaches.
A *laboratory experiment* has not yet been conducted. EA management is a topic, which is only recently taught in universities. This results in the problem that it is difficult to use students for the comparison of the pattern-based approach to EA management to other approaches for EA management. In addition, it is difficult to find a company willing to invest time and money to use two different approaches for EA management in parallel in order to compare them. A first step to improve the situation is made by using the EAM Pattern Catalog in a lecture at the Technische Universität München (see Section 6.8). Therefore, it may be possible to conduct a laboratory experiment in the future.
All three hypotheses are supported by the results of the case studies and the survey, see Chapters 6 and 7. The results allow the conclusion that the general research hypothesis holds and that the pattern-based method to EA management is an improvement to the currently available support for EA management. The last step *summarize results* is done via this thesis.

1.4. Organization

This thesis is organized in eight chapters and complemented by the appendix presenting some exemplary patterns for EA management. Thereby, the chapters are grouped in five parts as shown in Figure 1.2.
The first part gives the *introduction and overview* (see Chapter 1), including the motivation and a summary of the contribution of the thesis, followed by the description of the applied research method.
Part two constitutes the *foundation* that the thesis relies on. It is divided into two chapters. Chapter 2 *enterprise architecture management* gives an introduction to EA and EA management, with a special focus on analyzing and applying characteristics of engineering to EA management. Therefore, information models, visualizations, and processes and roles for EA management are considered as the basis for an engineering method to EA management. The topic of EA management is further detailed by analyzing currently available approaches concerning their benefits and drawbacks. Chapter 3 introduces the concept of *patterns*. Here, it is elaborated where patterns originate from and what related approaches exist. Complementing, the process how a pattern is abstracted from a real world pattern is presented. During the evolution of the pattern concept, different pattern forms have been developed, which have strengths and weaknesses for different application areas. Therefore, the most used ones are introduced and compared.

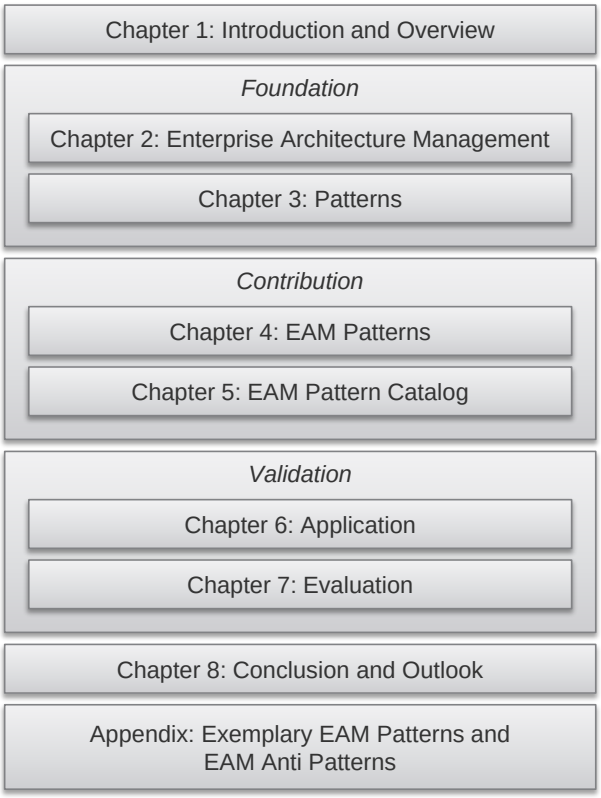

Figure 1.2.: Organization of this thesis

The third part uses the foundation introduced before to present the *contributions* of the thesis. Chapter 4 introduces *EAM patterns* and *EAM anti patterns* by applying the pattern concept to the area of EA management. After the introduction, four types of patterns for EA management are differentiated and their form of presentation is defined. This is complemented by guidelines on how to use EAM patterns to solve typical problems in EA management. The second part of the contribution (see Chapter 5) instantiates the EAM pattern concept by introducing the *EAM Pattern Catalog*. Due to the importance of relationships between patterns in a pattern language, the topic is elaborated in detail. This is followed by additional guidance for selecting and integrating EAM patterns in order to support the utilization of the EAM Pattern Catalog. As the development of a pattern language is no one-time endeavor, the EAM Pattern Catalog Wiki as a tool to ensure the continuous evolution of the EAM pattern language by a community of practitioners and researchers is presented.
The EAM pattern approach proposed in this thesis requires *validation* for its appropriateness to address the problems documented in the chapter about EA management. This is validated in the fourth part. Chapter 6 presents and analyzes six industrial case studies, an academic workshop, and a lecture at the Technische Universität München. The comparison to existing EA management approaches and the EAM Pattern Catalog online survey presented in Chapter 7 evaluate the benefits and drawbacks of the pattern-based approach to EA management.

Chapter 8 *concludes* the thesis and gives an *outlook* on future developments and extensions of the pattern-based approach to EA management.
The appendix presents selected EAM patterns and EAM anti patterns and describes their development in order to exemplify the approach.

CHAPTER 2

Enterprise Architecture Management

Contents

Modern enterprises face the challenge to survive in an ever changing environment. In order to support the transformation of the enterprise according to internal requirements like new business demands, legal regulations, or external requirements like changing market situations, a holistic perspective on the architecture of the enterprise has to be taken.
Managing the EA is a rather young discipline, which is more an art based on experiences [We03] then an engineering discipline like mechanical engineering. Those disciplines have well-established methods on how to address problems arising in their context. Shaw summarized in [Sh90] the various available definitions for engineering as "Creating cost-effective solutions to practical problems by applying scientific knowledge to building things in the service of mankind".
Figure 2.1 shows the evolution of an engineering discipline proposed by [Sh90]. The lower lines track the technology, and the upper lines show how the entry of production

skills and scientific knowledge contributing new capabilities to the engineering practice. In addition, typical characteristics are listed below each step.

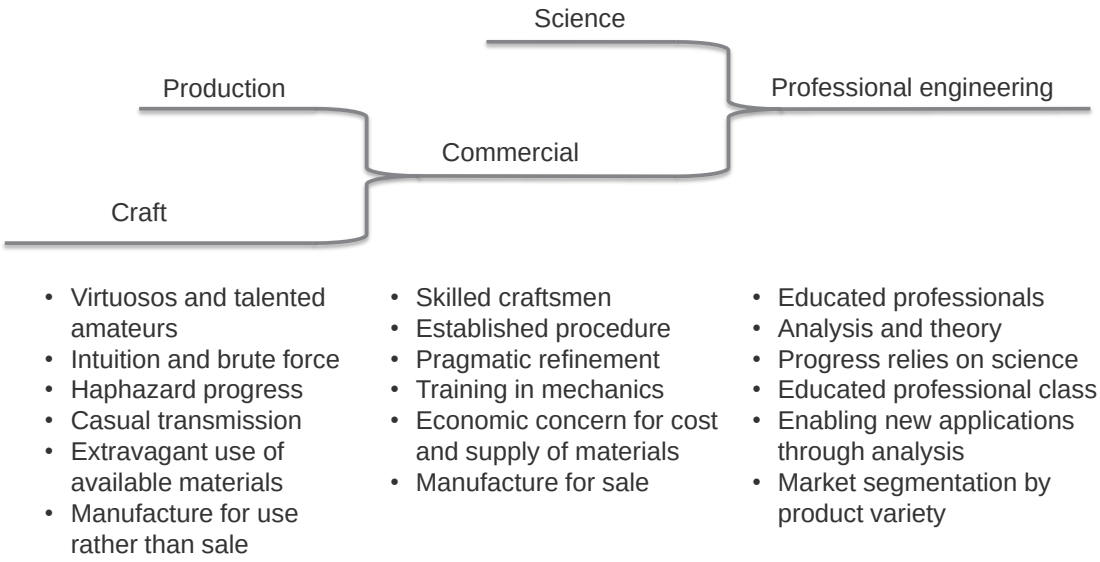

Figure 2.1.: Evolution of an engineering discipline [Sh90]

Attempts to apply engineering to EA management do already exist, see e.g. the business engineering navigator by Aier et al. [Ai09]. Pilkington [Pi09] calls such approaches *Enterprise Engineering*. It "is the application of engineering principals to the management of enterprises. It encompasses the application of knowledge, principles, and disciplines related to the analysis, design, implementation and operation of all elements associated with an enterprise" [Pi09].

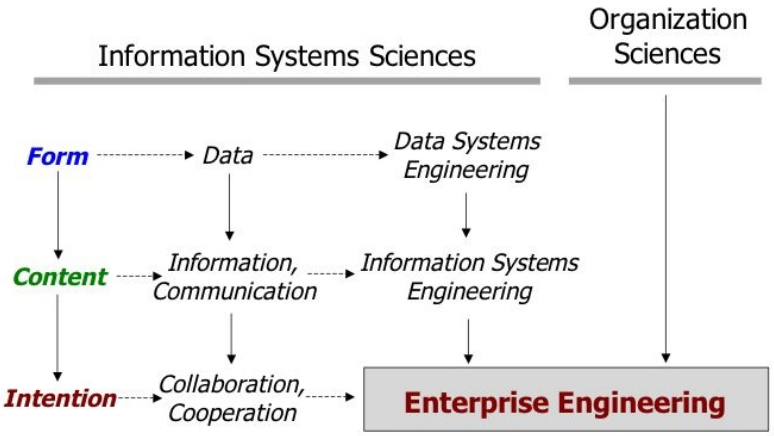

Figure 2.2.: Enterprise engineering and its related disciplines [DH08a]

Enterprise engineering is related to *Information Engineering*. According to Knolmayer [Kn09] information engineering is based on a design science paradigm. As a scientific discipline, information engineering is about the analysis of available and the

development of new methods for information design and the analysis of relationships between these methods, the implementation of the methods in tools and their integration in development environments, as well as the empirical evaluation of the methods usage and the tools usage.
Enterprise engineering is a design-oriented approach to help enterprises to adapt their strategies and to implement them effectively and flexibly [DH08a]. In order to achieve this, it merges *information systems sciences* and *organization sciences* (see Figure 2.2). The mission of the discipline enterprise engineering is "to develop emerging theories and associated methodologies for the analysis, design, engineering, and implementation of future enterprises" [DH08a]. Based on an analysis by Dietz and Hoogervorst [DH08a] two approaches for enterprise engineering have already emerged.

Enterprise ontology is an approach to construct and operate an enterprise in an implementation-independent way. It tries to reduce complexity by managing the enterprise with a high-level constructional model.

Enterprise architecture is a model of the enterprise's artifacts and their dependencies. It is used as a basis to apply principles that guide the design, engineering, and implementation of an enterprise.

This thesis focuses on enterprise architecture as it is according to Aier et al. [ARW08] wide spread and accepted in both academia and practice, which can be seen in various contributions, e.g. by [BH04, Ga04, Ju04, Me04, Si04a, Uh04, Wi04, Wö04]. Additional in-depth analyses of publications on EA up to the year 2004 by Langenberg and Wegman [LW04a] and by Schönherr [Sc09] focusing on publications between 2003 and 2008 also indicate the pervasion and importance of EA. The latter one also gives an overview about existing definitions of EA and their utilization in other publications.
Enterprise ontology will not be further detailed in this thesis, due to the higher importance and utilization of EA in academia and practice, but more information can e.g. be found in [AD06, Di06, DH08b, Op09c].
The rest of this section introduces available definitions for EA and EA management. These are used to derive the definitions throughout this thesis, as a foundation.
Aier et al. state that

> "the basic idea about EA is to map the most important artifacts of an enterprise together with their relationships to a model. The modeling pursues the goal to describe the mutual dependencies of the subjects of organization of an enterprise on an aggregated level. This is done for the current state for documentation and for analysis reasons, but as well for target states in order to plan the future development."

In addition, Aier et al. [ARW08] distinguish between different layers of an EA in order to be able to divide a complex problem into smaller sub-problems: Strategy, organization, integration, software, and IT infrastructure. A similar approach can be found in [Wi07a], where the layers business, business service, application and information, infrastructure service, and infrastructure are distinguished. These layers are accompanied by the cross functions strategies and objectives, requirements and projects, blueprints and patterns, and key performance indicators and metrics.

Lankhorst et al. in [Jo05] define EA as follows.

> "EA is the coherent whole of principles, methods and models that are used in the design and realization of an enterprise's organisational structure, business processes, information systems, and infrastructure."

In [RWR06], Ross et al. take a more business oriented focus, which is reflected in their understanding of EA.

> "EA is the organizing logic for business processes and IT infrastructure reflecting the integration and standardization requirements of the company's operational model."

The International Organization for Standardization (ISO) developed a standard for *Systems and software engineering – Architectural description* (ISO/IEC 42010) [IS07] pursuing a system-oriented approach. An enterprise can be considered to be a system that is made up of other systems [Ha05], e.g. of organizational units or other smaller enterprises. Therefore, the definition of architecture for systems in ISO/IEC 42010 can also be used for enterprises.

> "The fundamental conception of a system in its environment embodied in its elements, their relationships to each other and to its environment, and the principles guiding its design and evolution."

The Open Group Architecture Framework (TOGAF) (see Section 7.1.1) in most recent edition [Op09a] uses the ISO/IEC 42010 as a basis for its own definitions to stay consistent with a standard but also considers the accepted terminology of the TOGAF community. Therefore, TOGAF distinguishes between two definitions for architecture depending on the context. The first definition focuses on implementation issues.

> An architecture is "a formal description of a system, or a detailed plan of the system at component level to guide its implementation."

The second definition in TOGAF focuses on structural issues and guidelines.

> An architecture is "the structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time."

Like in the ISO/IEC 42010 [IS07], Northrop et al. in [No06] take a system-oriented approach for documenting and planning architectures. They call the approach *Ultra-Large-Scale Systems* (ULS) and define architecture as follows.

> "The architecture of a ULS system will consist of design rules (such as agreements on conventions, presuppositions, and constraints in the form of interfaces, standards, service-level agreements, etc.) that serve to decompose a system into component parts by decoupling design decisions that would otherwise be coupled. [...] Architecture is not purely a technical plan for producing a single system or closely related family of systems, but a structuring of the design spaces that a complex design process at an industrial scale will explore over time."

The system-oriented approach of ULS is closely related to the idea of an *extended enterprise architecture* [SA99, Op09a], which does not only cope with the artifacts of a single enterprise, but also includes partners, suppliers, and customers in order to integrate them in an optimal way.
Many more definitions for architecture in the context of enterprise architecture can be found in academia and practice. [ARW08, Ma09, Sc09] give an overview about those definitions. The various definitions lead to the problem that there is no general accepted definition of the term, ending up in confusion in discussions about this topic. This thesis therefore pursues a similar approach like TOGAF and resorts to ISO/IEC 42010 standard as a basis for the definition of EA.

> **Enterprise Architecture:** The enterprise architecture (EA) is the fundamental conception of the enterprise in its environment embodied in its elements, their relationships to each other and to its environment, and the principles guiding its design and evolution.

In addition to the problem that there is no generally accepted definition for EA, there is also the problem that often it is not clearly distinguished between EA and EA management. For this reason the rest of this section lays the ground for the understanding of EA management in this thesis.
Managing the EA is one of the major challenges of modern enterprises. It aims at aligning business and IT in order to optimize their interaction. The Open Group [Op04] emphasizes the importance of EA: "An effective enterprise architecture is critical to business survival and success and is the indispensable means to achieving competitive advantage through IT."
Documenting and managing the EA is an advanced and complex topic, which can be exemplified by managing the application landscape [Wi07a]. The applications landscape, which is an integral part of the EA, often includes a few hundreds up to a few thousand business applications and their interconnections. Those artifacts and their relations are only a small part of all the artifacts that need to be managed in an EA.
Thereby, managing the EA is a task that has to be executed as the need for a flexible IT is an integral concern of most companies. Another reason for the importance of EA management are regulations like e.g. the *Sarbanes Oxley Act* (SOX) [Co02], which determine the information a company has to have available about its EA.
Often EA management is conducted in a concern-driven or pain-point-driven way, meaning that EA management is not practiced for its own sake but it tries to address concerns occurring in an enterprise. A (system) concern is defined by ISO/IEC 42010 [IS07] as follows.

> **Concern:** Area of interest in a system pertaining to developmental, technological, business, operational, organizational, political, regulatory, social, or other influences important to one or more of its stakeholders

The *Report on Progress of Enterprise-Architecture-Management 2008* [Ma09] presents the understanding of EA management of different companies and research groups. To exemplify the difference between EA and EA management, three of them are presented in the following, leading to the definition for EA management used in this thesis.

For *Siemens* the most important aspect in EA management is the *alignment of business and IT*. Information about the EA artifacts dependencies is required for optimally and flexibly aligning the adaptation or design of the information systems to the requirements of the enterprise [Ma09].
Winter [Wi08] regards EA management as a tool for planning and steering the alignment of business and IT. Thereby, the systematic coordination between strategy, organization, and IT is supported. High efforts arise for collecting and maintaining information about the EA, requiring a certain level of abstraction, which is dependent on the selected EA management approach and to the enterprise itself.
Aier et al. [ARW08] summarize EA management to be based on a consistent, aggregate model of the enterprise in order to support decisions such as the design of business areas and products, corresponding business processes and organizational structures as well as application systems and infrastructure operations.
The different understandings presented above show that there is no sole definition, which everyone agrees on, i.e. that there is no commonly agreed understanding what EA management is and what it is not.
Concluding, it can be said that the alignment between business and IT and the coordinated steering of IT based on information about the EA are the most important aspects of EA management. This thesis uses the definition of EA management proposed by Matthes et al. in the Enterprise Architecture Management Tool Survey (EAMTS) 2008 [Ma08a], which also emphasizes theses aspects.

> **Enterprise Architecture Management:** Enterprise architecture management is a continuous and iterative process controlling and improving the existing and planned IT support for an organization. The process not only considers the information technology (IT) of the enterprise, also business processes, business goals, strategies etc. are considered in order to build a holistic and integrated view on the enterprise. Goal is a common vision regarding the status quo of business and IT as well as of opportunities and problems arising from theses fields, used as a basis for a continually aligned steering of IT and business.

The following section introduces methods as means to address common problems in engineering. Thereby, the constituents of a method are derived and linked to EA management.

2.1. Enterprise Architecture Management Methods

The previous section elaborated on the problem that various definitions for EA management exist. This is also documented in many other publications, e.g. [ARW08], [Bu09d], [Ma09], [Sc09], etc. These definitions vary regarding the scope, level of detail, and focus, thereby leading to different understandings of what EA management is about. As a result, a company establishing an EA management first has to define what should be included in the approach. This includes to identify the topics and concerns, which should be addressed.
A way to address this issue would be to define engineering processes most stakeholders can agree on, which detail what EA management is all about and explicate how to

address which problems. According to Kronlöf et al. [KSH93] an engineering process should always be based on methods, otherwise it is no engineering process. Therefore, the definition of what a method is, is important in the context of this thesis. This thesis resorts to the definition of a method in the context of systems engineering proposed in [KSH93].

Method: Within an engineering discipline, a method describes a way to conduct a process. In the context of systems engineering, a method is defined as consisting of:

- An underlying model
- A language
- Defined steps and ordering of these steps
- Guidance for applying the method

The *underlying model* refers to the classes of objects represented, manipulated, and analyzed by the method. This underlying model is often called a metamodel or an information model. For an analysis of such models in the context of EA management see Section 2.2.
The *language* refers the concrete means of describing the products of the method. In EA management those products are typically visualizations of any kind, like reports, documents, views, etc., which are further detailed in Section 2.3. The language is also often called a modeling technique.
Defined steps and ordering of these steps refers to activities by the user of a method. Approaches defining such steps or processes, are also known in EA management, but problems exist that limit their success. Some of those problems are detailed in the following. As a method is typically utilized by different users it is required to define roles, which can be associated to the defined processes. Therefore, this constituent of a method is also called *processes and roles*.
Guidance for method application is typically given via guidelines, or case studies, which show how a method can be applied in practice.
A similar definition of a method's constituents is given by Leist and Zellner [LZ06] based on various definitions in the context of method engineering (e.g. [He93], [Br96], [TRL96], etc.) and by Aier and Fischer [AF09]. A method comprises a metamodel, techniques/-modeling techniques, a procedure model, roles, and a specification document. The only difference to the previous definition is that the guidance for applying the method is split up in a definition of roles participating in the method and documents specifying the resulting documents. Another detailed analysis of the constituents of a method, which leads to the same result, can be found in [Br05a].
[KSH93] mentions that the "quality of an engineering process is strongly dependent on the quality of the methods used and, although methods alone are not sufficient, the quality of the process is definitely poor if methods are not used at all". This statement shows methods are an essential part in engineering.
The growing importance of methods in various domains and the need for systematic principles to develop situation-specific methods led to the discipline of *method engineer-*

ing [KW92]. Kumar and Welke [KW92] therefore developed a proposal for situation-specific method construction. Based on the definition by [To98] and [KW92] method engineering is understood as follows.

> **Method Engineering:** Method engineering is a change process by a method engineering group using a meta method and supporting tools to develop or maintain methods.

One of the most important problems in methods for EA management is that there is hardly any "one-size-fits-all" EA method [RA09a]. This means that the requirements for an EA management method differ widely in different companies and that they can hardly be addressed with a monolithic, all-embracing standardized method.
In larger companies, EA management, which involves a larger number of people, is typically supported by tools [Bu09d]. First of all, they are used for documenting information in one or more repositories and for creating demanded deliverables. Secondly, they ideally also support the steps or the processes defined in the method. Although there is a market for EA management tools for some years now, the market is still volatile. This means that new tools are introduced in the market, others are no longer available or were acquired by some other vendors. A result of this is that the included functionality increased over the years, which can be seen by comparing the two *Enterprise Architecture Management Tool Surveys* conducted in 2005 [se05] and in 2008 [Ma08a]. The importance of tools for managing the EA is also shown in analyses conducted by consulting companies, e.g. in [Pe04] or [Ja08]. Nevertheless, the surveys show that there still is potential for improvement in functionality and in support of EA management processes. Similar results have been gathered in [Bu09d].
Leist and Zellner [LZ06] use the previously introduced constituents of a method to analyze different EA frameworks. This thesis follows a similar approach, as it uses the concept of a method to structure EA management to introduce some basic concepts in the following sections. Information models for EA management are the first constituents of a method, which are further detailed.

2.2. Information Models for Enterprise Architecture Management

Managing the EA is not only about defining governance structures, management processes, or creating visualizations in Microsoft PowerPoint, Microsoft Visio, or any other software capable to create drawings. It is also about collecting and managing information for documentation reasons, as a basis for creating visualizations, and for supporting the defined management processes.
Typically information about EAs are stored in a structured (entities, attribute, relationships, etc.) way using spreadsheets, databases, or EA management repositories. Recent developments like [Bu09b] also consider starting with semi-structured information e.g. by using a wiki to collect information about the EA, which is little by little transformed to structured information.
A recurrent problem in EA management is that a standardized vocabulary is needed, e.g. like in engineering [Ar07, Ai08b]. In EA management communication is an important

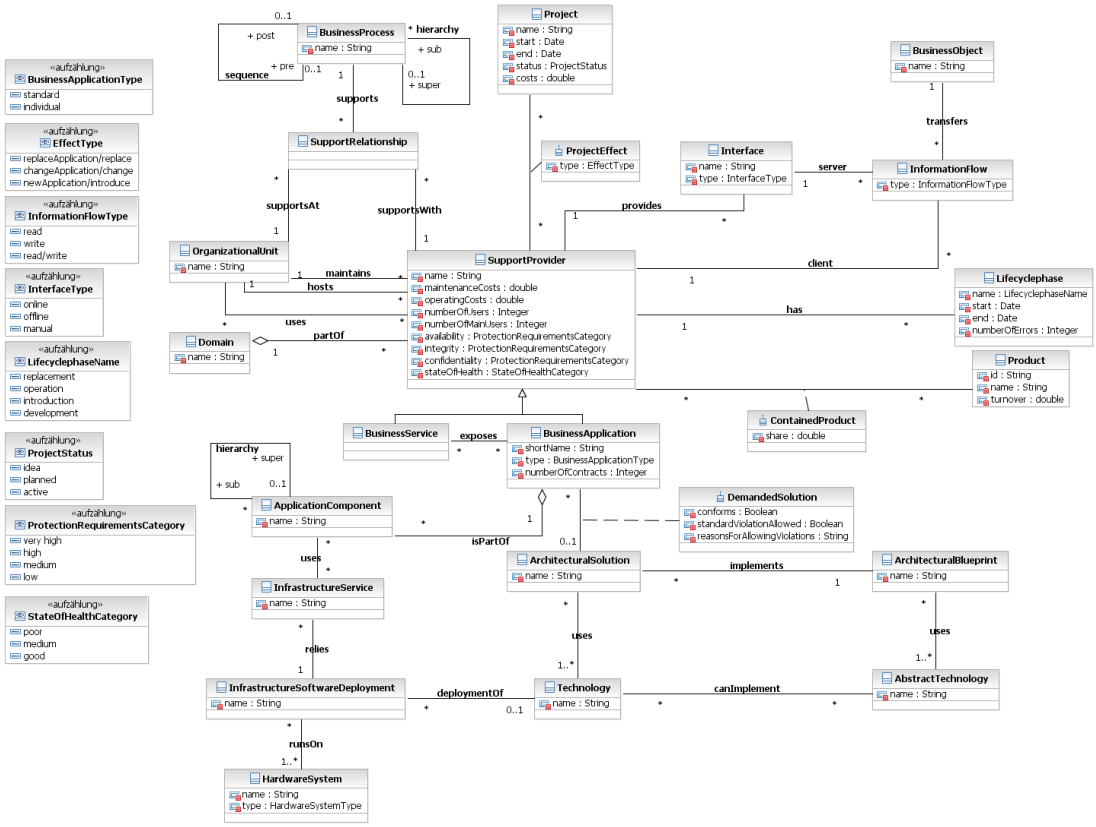

Figure 2.3.: An exemplary EA information model from [Pf08]

aspect, especially because EA management should be anchored across the whole company. If no common vocabulary is available, it is hard to work together cooperatively, as you cannot even use terms in the same way, because they are understood differently. In cases where a structured approach is pursued, it is required to develop a model of the EA. Such a model is called *information model*. Buckl et al. [Bu07a] define an information model as follows.

> **Information Model:** An information model specifies, which information about the enterprise architecture, its elements and their relationships should be documented, and how the respective information should be structured

This is usually achieved via a model expressed in a language suitable for conceptual modeling, as e.g. the Unified Modeling Language (UML) [Hi05], enriched with descriptions detailing the exact meaning of the concepts. An exemplary information model can be found in Figure 2.3. Similar definitions for the term information model can be found in [Wi07a], [BS96], or [Te99]. The benefit of the enriched descriptions, or a glossary, is that they can be used as a basis for a standardized vocabulary.
On the one hand an information model is an instantiation of a *metamodel* like the Meta Object Facility (MOF) [OM06]. According to [Fa04, Kü06, St98] a metamodel is a model

of a modeling language. On the other hand an information model can be instantiated to a *semantic model* representing real world information stored according to an information model, for example using a repository. For a more detailed elaboration on this topics see [Er06, Bu07b].
Information models are sometimes also called metamodels (cf. [Ai09] or [Op09a]) or enterprise models (cf. [GF96] or [GF96]). To clearly distinguish between the three previously described layers, the rest of this thesis uses the term information model.
When developing an information model, Stachowiak's general model theory has to be considered [St73]. Accordingly, a model is characterized by the following properties:

Mapping property: Each model is a model of something, e.g. a reproduction, representation of natural or artificial originals, etc.

Truncation property: A model in general does not include all properties of the represented original but only those, which are relevant for the creators or the users of the model.

Pragmatic property: The model is subdue to a purpose and its use for this purpose is only justified with respect to particular users, their objectives, applied techniques and tools, and period of time etc.

Especially the last two properties have to be considered during the development of an information model. Trying to represent every detail of an enterprise in one information model is not feasible. This would lead to an unmanageable, extensive information model, requiring high efforts for maintenance and for collecting information that should be stored according to the model. Therefore, the pragmatic property has to be considered: An information model is developed for a certain purpose and should therefore only include details required for this purpose[1]. A major problem is that this purpose often is not even known during the construction phase.
[Ai09] uses a similar argumentation for constructing information models. They differentiate between three criteria:

Width criterion: Information models have to address the information demand of their stakeholders, which is implied by their concerns. Those concerns can be address by viewpoints showing aggregated information in a form appropriate for the respective stakeholder. The criterion of width requires to include all artifacts and relationships required to create the viewpoints in order to address the concerns.

Depth criterion: This criterion is about preventing to document information on a too detailed level because this typically requires high efforts to maintain information about the artifacts under consideration. Defining a relevance criterion is helpful in deciding, if an artifact should be included or not. [Ai09] proposes such a relevance criterion. If changing an artifact of the EA has an impact on other artifacts, it should be included, otherwise it should most likely be omitted. This criterion is deduced from the idea of encapsulation and information hiding originating from [Pa72].

[1] An EAM anti pattern concerning the development of an Oversized Information Model can be found in Section A.4.1.

Pragmatic criterion: The third criterion accounts for the efforts and costs resulting from collecting and maintaining information according to an information model. Creating an information model it has to be considered, if the benefit of the collected information is higher then the effort required to collect it. If this is not the case, the concepts should not be included in the information model, although it might be considered relevant for some stakeholders.

Various documented examples for creating an information model can be found in academia (see e.g. [Ös07] or [Ma08a]) and in practice (see e.g. [La05b], [Br05b], [Pf08], or [Op09a]). All those examples show that there is no standard information model available fulfilling the requirement of a majority of companies concerned with the development of an EA management approach. Although, there is a demand for such a model, for example as a basis for a standardized exchange format to be able to exchange information between different EA management tools. A possible reason for the lack of a standardized information model is that the requirements are too different to be well addressed by one monolithic information model.
[Ar07] also analyzed the utilization of models and states four problems in their usage for architecture description due to the abstract nature of the object being designed and the descriptions of this design:

- Confusion exists with respect to the distinction among a model's presentation, content, and semantics.
- Capturing the diverse and abstract nature of information systems often requires the use of multiple large, complex, and interrelated models providing insight into the system from different viewpoints.
- In IT the technological building blocks, their abilities and their boundaries, are not as clear (and stable) as they are in the other disciplines.
- Socio-economical aspects are as important as IT aspects. This makes it harder to come up with a limited set of architectural descriptions, models, and associated languages.

As previously mentioned, visualizations as means for communication are important in EA management. Therefore, this topic is analyzed in the next section.

2.3. Visualizations for Enterprise Architecture Management

Visualizations are an integral part of EA management. The main reason for that is that they can facilitate the highly important communication between different stakeholders. Like in a town the different stakeholders, e.g. city major, merchants, urban planners, young families, etc. have to find ways to communicate with each other. In EA management these stakeholders are e.g. enterprise architects, business process owners, application owners, etc. In managing towns maps are used, which are intuitively understandable and can be used by all stakeholders.

A similar approach is used in EA management. In this case such visualizations are called software maps [Wi07a] or more general views, based on the definition of ISO/IEC 42010 [IS07].

> **View:** Representation of a system from the perspective of an identified set of architecture-related concerns.

The International Standards Organisation [IS07] thereby distinguishes between view and viewpoint. Their relationship can be compared to the one between object and class in object-orientation. A view is an instantiation of a viewpoint.

> **Viewpoint:** Conventions for the construction, interpretation, and use of an architectural view and its contributing architectural models.

In contrast to more mature disciplines like software engineering, where standardized viewpoints and notations like UML are available, EA management still lacks such standardized graphical languages.

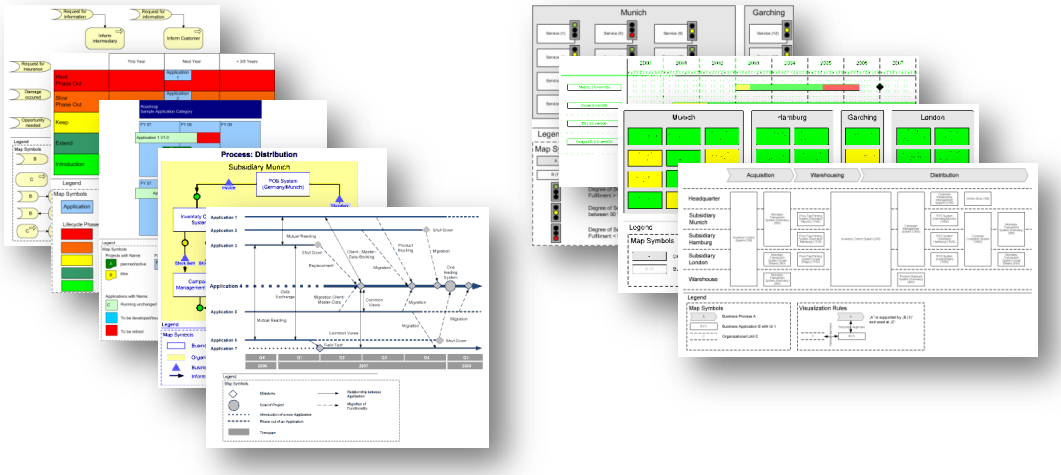

Figure 2.4.: Exemplary views used in EA management

Only a few groups are working on defining such a graphical language. Among them are the *Archimate* group (see e.g. [La05a, Op09b]) and the *System Cartography* group (see e.g. [Wi07a, Er06, Ma08b, se09d]). Other approaches for visualization can be found in tools for EA management. An overview about the capabilities of the major tool in the market can be found in [se05, Ma08a].
The absence of such a standardized language leads to the problem that each company uses self-defined viewpoints for EA management and tries to build its own "standard". This leads to rarely defined viewpoints, because the time required to adequately specify the viewpoints is usually not available in daily business. In addition, such viewpoints are often company-specific and can only be used within one company. Usually the exchange between different companies or more general different stakeholders is limited, although

it is very important for the future development of these viewpoints. Figure 2.4 shows some exemplary views gathered from industry.
In order to resolve the problem of completely independent viewpoints in different companies Wittenburg [Wi07a] defines three basic *software map types*, which are briefly described in the following sections as one example for visualization approaches currently available. For a detailed elaboration on *software maps* and the underlying *visualization model* see [Er06].

2.3.1. Cluster Map

The first software map type is called *cluster map*. It uses the concept of grouping visualization elements – *map symbols* [Er06, Wi07a] – in accordance to certain rules – *visualization rules* [Er06, Wi07a]. The grouping is called *clustering*. This is what this software map type is named after. Map symbols within a cluster are positioned randomly, but it is tried to position the elements in an appealing way. The same is true for the clusters themselves. Placing a cluster in the left upper corner does not mean anything else then placing it in the lower right corner, although placing elements at the same position on different cluster maps may support quick recognition of relevant elements. Clusters may also contain other clusters, whereas cluster maps containing more than three level of clusters may become hard to read and to use.

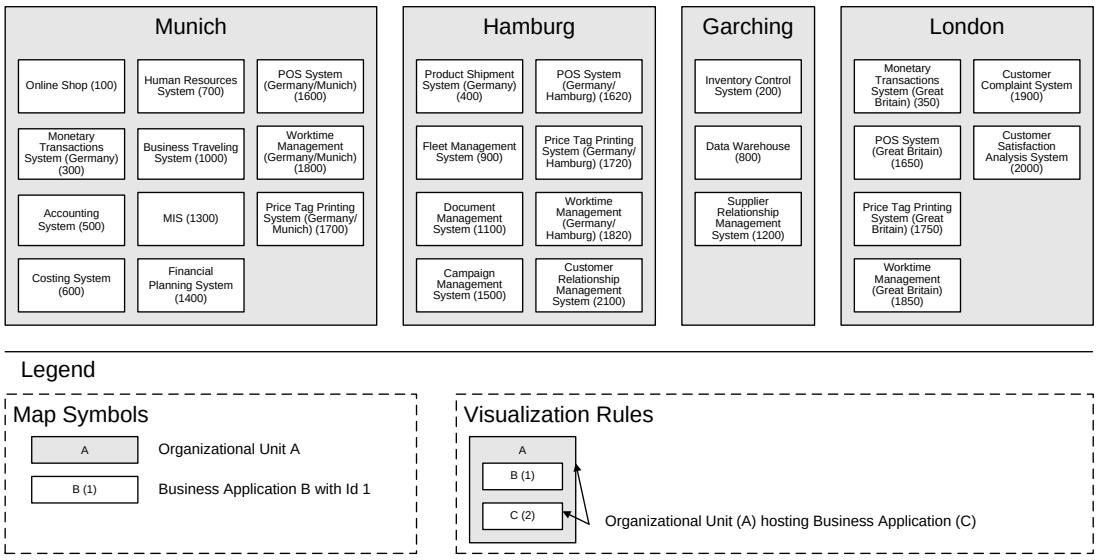

Figure 2.5.: A cluster map showing organizational units hosting business applications

Figure 2.5 shows an example for a cluster map. In this example business applications are grouped according to the organizational units, which host them. The cluster map could also be enriched by additional information, like e.g. interfaces between the business applications visualized as lines between the map symbols or cost of the business applications visualized via color-coding. Adding such information to a software map can be done using the layering principle (see Section 2.3.4). Other rules for the grouping could

be business applications based on (non) standard software, products being supported by business applications, etc.
Cluster maps can be used in various ways. The simplest usage scenario is to get an overview about which business applications are in use in a company. More advanced scenarios are to define distance measures based on the number of clusters hierarchies you have to pass to navigate from one business application to another using the interfaces visualized in the cluster map. Such a distance measure may help in identifying interfaces, which can be adapted more easily then others. In addition the costs for communicating with the application owners of all affected business applications are lower within one organizational unit (one cluster), whereas the required effort will be higher across organizational unit boarders.

2.3.2. Cartesian Map

A different software map type is the *cartesian map*, which has a matrix like structure. Different kind of information can be used for the two axes spanning up the matrix. Figure 2.6 shows an exemplary view, called *process support map*. On the x-axis the business processes are displayed, typically business processes of level 0 up to level 3, because it is important that the business processes form a linearly ordered sequence in a process support map. The y-axis is made up of organizational units in a random order. The cells of the matrix can be used to place business applications. This kind of visualization shows which business processes are supported by which business applications at which organizational units.

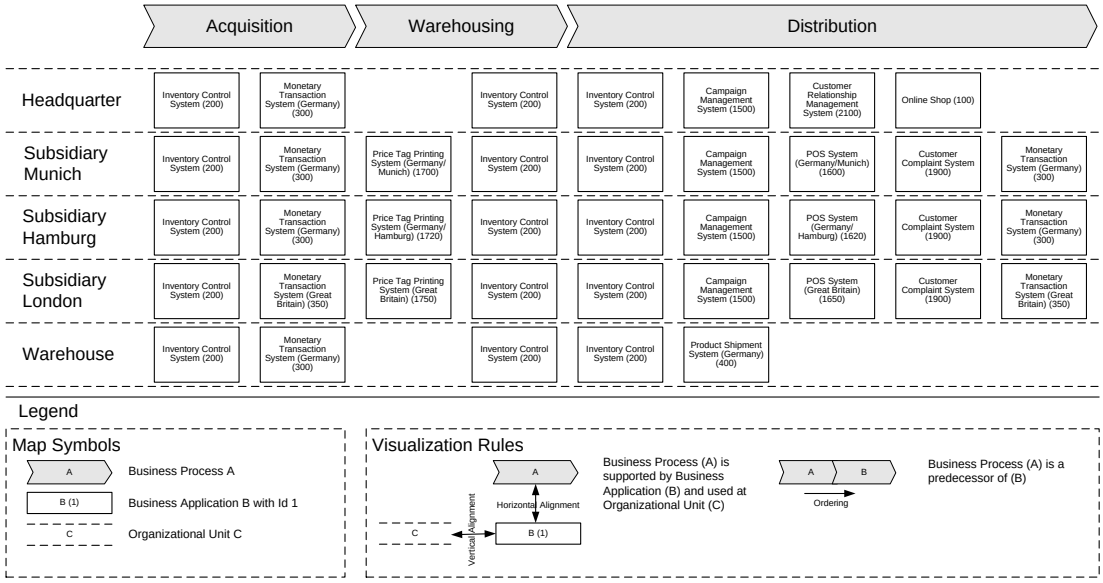

Figure 2.6.: A process support map showing business processes supported by business applications at organizational units

The main advantage of a process support map is that it shows information from the business and the IT domain in combination. It can e.g. be used for consolidating

the application landscape by homogenizing the business applications used at different organizational units for the same business process, also known as *vertical integration*. *Horizontal integration* is a similar approach using a process support map, trying to avoid media breaches, by reducing the number of business applications used for different business processes at one organizational unit.
Instead of organizational units, it is also possible to show e.g. products, domain, or other categorizations for business applications. In service-oriented architectures [KBS05] a similar visualization is used, showing services, instead of business applications.
The previously described variant of the process support map used ordinal scaled information on the x- and nominal scaled information on the y-axis. Another variant of a cartesian map shows interval scaled information, e.g. time, instead of ordinal scaled one. Such a viewpoint is called *interval map*.

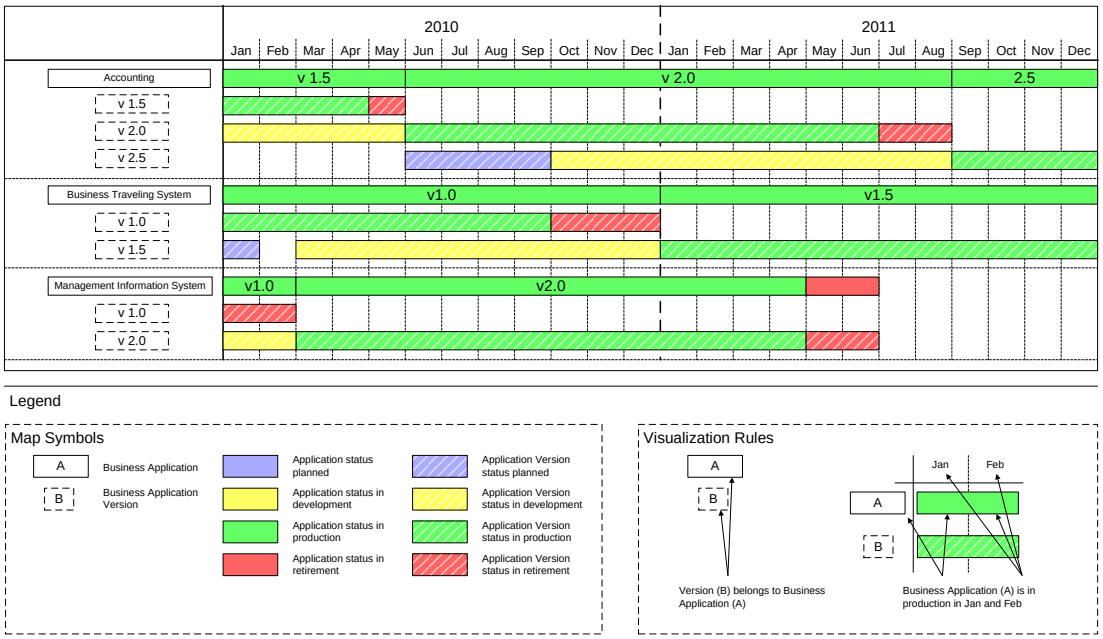

Figure 2.7.: An interval map showing the life cycle of business applications and their versions

Figure 2.7 shows an exemplary interval map. It has a Gantt-diagram like structure, with a timeline on the x-axis and business applications and their corresponding versions on the y-axis. Such viewpoints can be used to document and manage the life cycles of business applications or other elements of the EA, where time related aspects are of importance.

2.3.3. Graph-Layout Map

The *graph-layout map* does not rely on a base map for positioning of elements. Instead such viewpoints are typically graphs based on nodes and edges. For a detailed discussion of base maps see [Er06].
An example is given in Figure 2.8. It shows a business application and its relationships to other business applications. The business application, which is in the focus of interest,

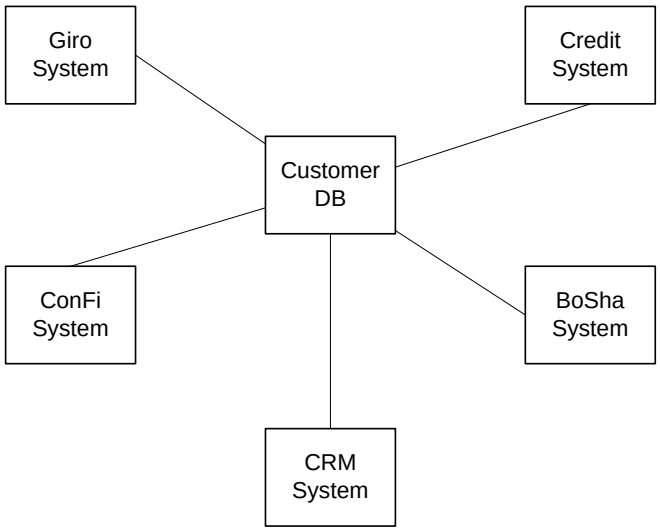

Figure 2.8.: A graph layout map showing the interfaces between business applications

is shown in the center of the visualization and the other business applications are placed around it, e.g. using a spring-based layout algorithm.
A benefit of graph-layout maps is that they can easily be generated from repository information. In contrast, a drawback is that small model changes usually result in drastic layout changes of the view.
Graph-layout maps can e.g. be used for dynamic analysis of information by navigating through a large amount of data or for showing relationships between different concepts.
A similar approach to visualize dependencies in software engineering is the so called *Sotograph* [BKL04].

2.3.4. Layering Principle

The different software map types described in previous sections can be enriched with additional information. In (geographic) cartography [Sl03, Tu01] the idea to use layers to show additional information like population density, average income, etc. is very common.
The same concept applies to software maps. Figure 2.9 shows a cluster map with multiple layers. The base map is made up of organizational units, which is referenced by the first layer showing the application systems. This is followed by a second layer showing information flows between the applications systems and a third showing key performance indicators for the application systems. The latter two layers both reference the application system layer, i.e. use this layer to derive positioning information.
A software map of this kind includes a lot of information, which may be hard to work with, but hiding certain layers allows to vary the information according to personal demands.

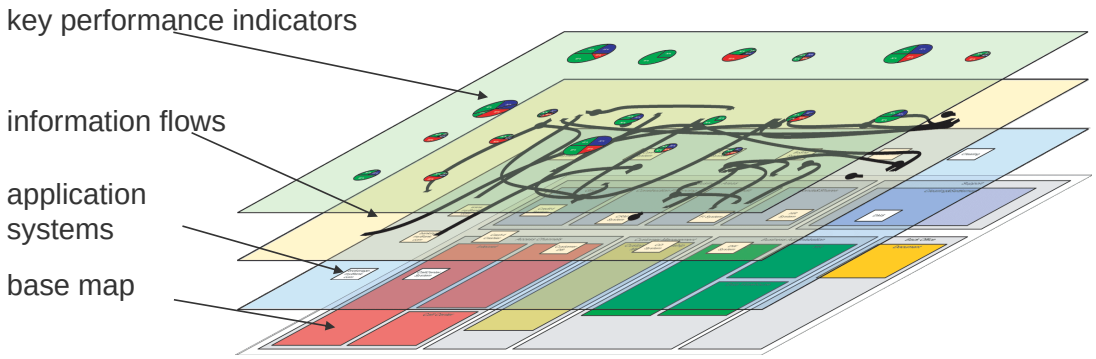

Figure 2.9.: The layering principle applied to software maps

2.3.5. Summary of Visualizations for EAM

Summarizing this chapter, the main problem of visualizations for EA management is a missing standardized graphical notation like the UML in software engineering. This leads to company-specific developments restricting their exchangeability and a limited future advancement of these visualizations.
Additionally, it turned out that the generalization of the visualization concepts for EA management by Wittenburg [Wi07a] is valuable but that the results are too general to be directly applied in practice.
Nonetheless, visualizations are very important, because communication is a major task in EA management. Therefore, a different way to document visualizations for EA management should be developed, which fosters the future development of viewpoints and the exchange between various groups.
Introducing information models and visualizations for managing the EA are an important step in the introduction of EA management in an enterprise. Nevertheless, processes and roles are required as a means for structuring and guiding the EA management approach. They will be introduced in the following section.

2.4. Processes and Roles for Enterprise Architecture Management

Various methods for managing the EA have been developed in recent years, see e.g. Dern [De06], Niemann [Ni06], Wittenburg [Wi07a], Keller [Ke07], Engels et al. [En08], Schekkerman [Sc08], Hafner and Winter [HW08], The Open Group Architecture Framework [Op09a] (see Section 7.1.1 for more details), etc.
This thesis gives a compact summary of the processes as described in two of the approaches as an introduction to processes for EA management. Wittenburg [Wi07a] developed the management processes shown in Figure 2.10 in cooperation with *BMW Group*.
Six different management processes are defined, which are grouped around and linked to the IT project life cycle. The *enterprise architecture management* process keeps everything together and coordinates the other processes.

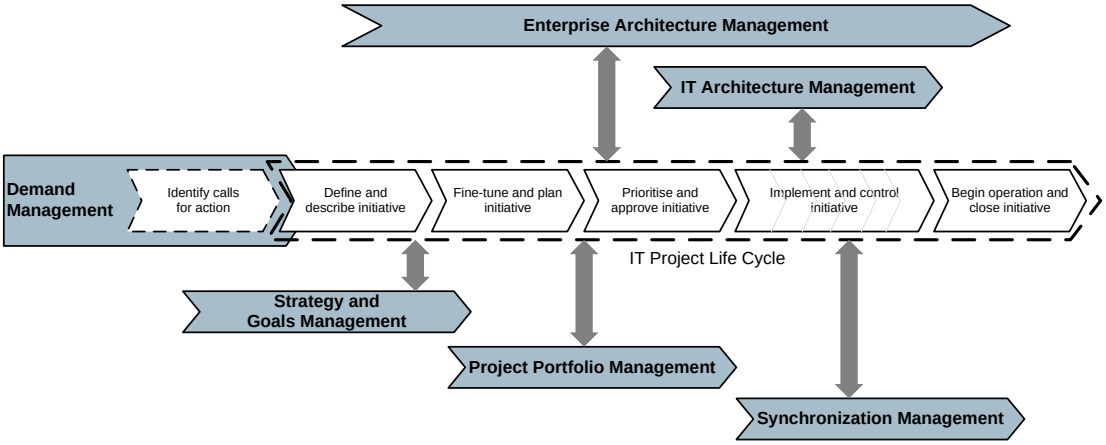

Figure 2.10.: Processes for managing an EA by Wittenburg [Wi07a]

Demand management collects requirements concerning changes to the EA for the whole enterprise, which may lead to one or more IT projects. Those requirements have to be documented in a standardized form to support comparison and automatic processing. They are the basis for new initiatives, which are triggered by the demand management. *Strategy and goals management* controls, if the new initiatives comply to the strategies and goals of the company. If they do not comply, it is analyzed whether an exception has to be made, e.g. because of the importance of the new initiative for the business. *Project portfolio management* cares about selecting the right projects to be conducted based the submitted initiatives and on existing constraints like strategies, costs, etc. *Synchronization management* takes care about the dependencies between projects, e.g. if a project is delayed and is the basis for a follow-up project the follow-up project also has to be delayed.
All IT projects have to take care of defined architectural standards. Those standards are defined by the *IT architecture management* process. Therefore, there is a relationship to the IT project life cycle.
A more detailed elaboration on these management processes, also detailing the deliverables exchanged between the processes, can be found in [Wi07a].
The second approach to be introduced in this thesis defines *processes for enterprise application architecture management*, which can be considered to be equivalent to EA management. It was developed by Hafner and Winter [HW08] by deriving a consolidated management process using three case studies at *Credit Suisse*, *Die Mobiliar*, and *HypoVereinsbank*.
The consolidated process model is shown in Figure 2.11. It is split up in four distinct phases, which all include at least two sub-phases.
The *architecture planning* phase assesses the current and the planned architecture for adaptation requirements. Thereby, also strategic requirements are considered. The assessment is used to derive architecture principles, which are used in a next step in the development of future architecture artifacts.
In the *architecture development* phase, additional operational requirements from IT and

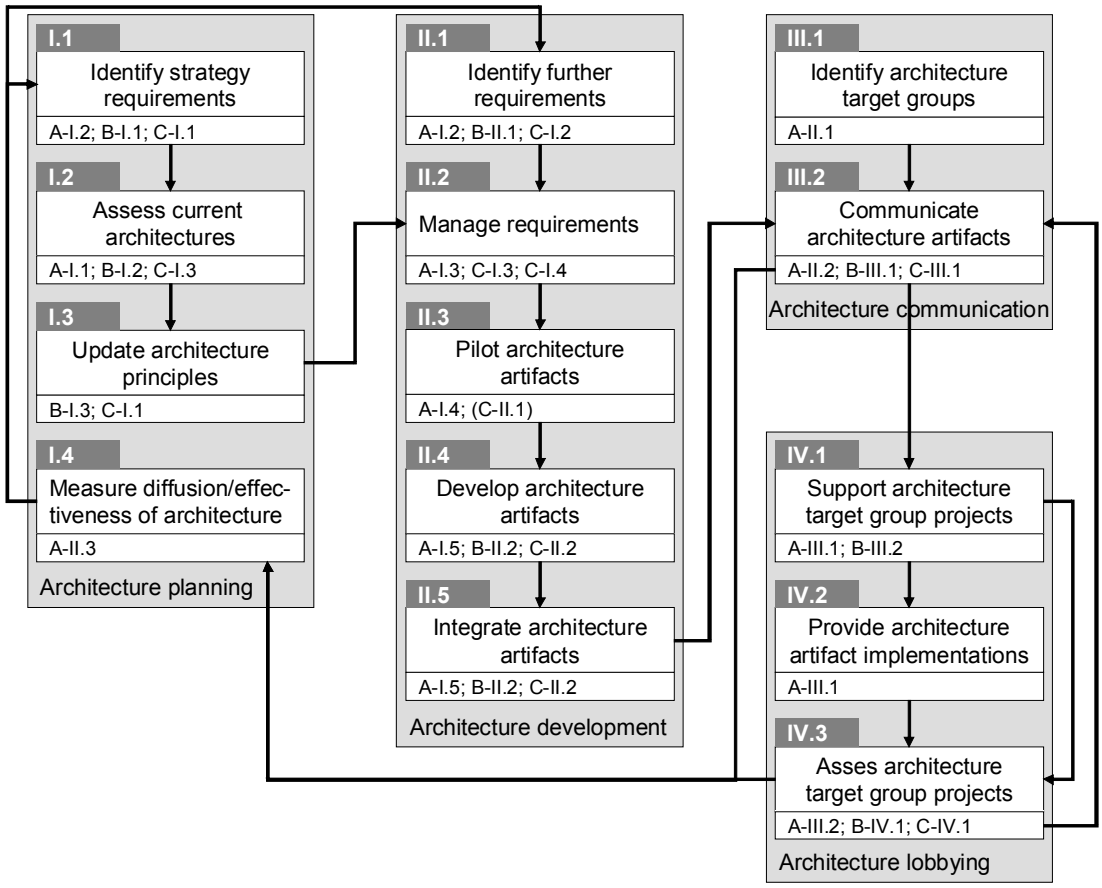

Figure 2.11.: Processes for managing an enterprise application architecture by Hafner and Winter [HW08]

the whole enterprise are collected, consolidated, and prioritized. Based on these requirements, architecture artifacts are developed and integrated with existing artifacts.
The *architecture communication* phase identifies groups of people concerned with EA management, which require training, information material, etc. and supplies them according to their needs.
The *architecture lobbying* phase provides assistance to projects and answers questions about the architecture. The assistance can be of operational but also of strategic nature. Additionally, architecture artifact implementations are provided, e.g. to relief development projects of infrastructure details. The last sub-phase uses information from communication of the architecture to derive new strategic and operational requirements, which are used to measure the diffusion and effectiveness of the architecture in phase architecture planning.
The main problem with currently available descriptions of processes for EA management is that they are either not detailed enough to be implemented, even though they give a good introduction, which processes or phases should be considered, or their extensiveness requires an intensive familiarization with and customization of the processes.

Besides this the general applicability of the process models still has to be evaluated, as they are typically derived from a limited set of case studies, which may not be representative for the majority of enterprises.
The resulting difficulties in establishing a process model within an enterprise are emphasized by [Ma09]. They found out that only about half of all analyzed companies posses a complete and consistent process model for their IT management processes.
Another problem is that role models are missing. Hafner and Winter [HW08] explicitly mention that their process model has to be supplemented by a role model and design techniques. The same is true for [Wi07a], which does not explicate the stakeholders or roles for the proposed management processes. As a result it is difficult to incorporate the right people when establishing an EA management approach.
EA management as a holistic approach has to include both sides, business and IT. This has been pointed out by Buckl et al. [Bu09d] and is also emphasized by Maicher et al. [Ma09]. The fact that at least some of the attempts to introduce EA management in an enterprise are advanced by IT department and not by the business additionally aggravates the situation. EA management can only be successful, if all stakeholders concerned with managing the EA takes part in the EA management process.
To incorporate the various stakeholders of the IT but also of the business side in one EA management approach, adequate governance structures and processes are required [Bu09d, Ma09]. Currently available literature and EA frameworks only offer limited support developing and implementing such structures. In addition, decisions about the EA have to be enforced within an organization, which often leads to problems, because the people who have to introduce these guidelines often do not have the power to enforce them [Bu09d].
In order to get all those different stakeholders to strive for the same goals in EA management, communication is an important factor [Bu09d]. This requires on the one hand that a common understanding of the EA management approach is given. A first step in the establishment of such an understanding is to define a glossary [Ma09]. On the other hand, adequate means for communication, like reports, visualizations, etc. have to be available (see Section 2.3). Focusing on common goals is additionally complicated because of the different topics of interest in EA management. Some of them are briefly introduced in [ARW08] and [Bu09d]. Therefore, the topics to be addressed should be defined.
The EA management processes require to gather and manage information about the EA [ARW08, Bu09d]. Thereby, it is important to carefully select, which information should be collected (see Section 2.2). Different processes on how to collect the required information are known, but none of them is established in a way that a majority of enterprises agrees on [ARW08, Bu09d].
After having established the processes required for managing the EA, it is important to measure and analyze them for potential for improvement and to control whether the defined goals have been reached. In order to achieve this, metrics and key performance indicators are of importance [Bu09d]. Similar is emphasized by Maicher et al. [Ma09]: "The demand to define and measure integrated IT management processes will increase in the next two to three years." Despite the importance of this field only few ideas have been published on this topic, e.g. [GSL07], [La08a], [Ad09], etc. providing an area for future research.

2.5. Summary

The first section of this chapter introduced a new approach to address the challenges of aligning the business and the IT of an enterprise, by utilizing concepts of mature engineering disciplines. One possibility to use these concepts is EA management. The term is widely used in academia and practice but currently no well accepted definition and common understanding of the term exists. Therefore, a definition for EA and EA management was derived from an overview about common definitions and understandings as a foundation for this thesis.
Engineering disciplines have developed methods to address the concerns arising in their context. In order to develop an engineering-oriented way to EA management, the term method was defined and applied to EA management. This included a summary of the currently available support for a method's constituents hinting to possible drawbacks in current EA management approaches. The section about information models for EA management (see Section 2.2) showed that although different approaches have been proposed by academia and practice no standard information model for EA management could yet be developed. Currently used information models are often too detailed to be maintained or are insufficiently documented restricting their future development. Similar problems are present in the context of visualizations for EA management (see Section 2.3). Although visualizations are important means for communication between different stakeholders, there is no accepted standard notation for EA management. This leads to the development of company-specific visualizations, restricting distribution in the EA management community and also restricting their future development. Processes and roles are also important parts of a method for EA management. They were introduced and analyzed for drawbacks and benefits in Section 2.4. The main problem is that the business side of a company often does not participate in the EA management approach in an adequate way. As EA management should be a holistic approach this necessarily leads to problems. In addition, complete and consistent process models for managing the EA are often not implemented within companies. This may be a consequence that currently available approaches are either not detailed enough, or to extensive to be introduced.

CHAPTER 3

Patterns

Contents

One way to address complexity is abstraction. According to Coplien [Co96] "much of design concerns itself with finding the right abstractions in a system, partitioning the system into mind-size, manageable chunks". This way of thinking lead to the concept of patterns, which will be introduced in this chapter.
The term pattern can be found in various disciplines like music (cf. [BM98]), education (cf. [LTW08]), project management (cf. [Co97a, Vö04]), architecture (cf. [AIS77]), software engineering (cf. [BC87]), etc. An exemplary compilation of over 700 patterns can be found in [Ri00], which focuses on software patterns, but the book also covers specialized topics like finite state machines, fire alarms, education, or patterns about patterns. Complementing, the utility of patterns is shown by Beck et al. [Be96] by presenting industrial experiences with design patterns. Erickson [Er00] gives three reasons for the fact that patterns are documented and utilized in so may different application areas:

Quality Pattern languages support the creation of systems that have what Alexander and his colleagues call "The Quality Without a Name". This is a shorthand for systems which really "work" for people, in all of the many meanings of that phrase.

Re-Use Pattern languages permit the re-use of the hard-won wisdom of designers, allowing the accumulation and generalization of successful solutions to commonly encountered problems.

Lingua Franca Patterns form a language that can be understood by all stakeholders.

To summarize, patterns are a popular approach because they provide the required quality, permit reuse, and can be understood by a wide variety of stakeholders.
Due to the wide spread use of the term, this section introduces definitions to derive their commonalities, preparing a definition of the term for the later usage in the thesis. This is followed by an excursus on approaches related to patterns and an elaboration on how patterns emerge. Finishing this chapter an overview about different pattern forms is given.

3.1. Pattern History and Definition

Patterns are a concept well-known and used in various disciplines. One of these disciplines is architecture, which also established one of the earliest definitions for patterns. Alexander et al. [AIS77] coined the following definition.

> Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.
>
> Each pattern is a three-part rule, which expresses a relation between a certain context, a problem and a solution.

Although, Alexander did not owe his breakthrough in architecture to the introduction, documentation, and usage of patterns, he nevertheless laid the ground for the usage of patterns in computer science. In 1987, Kent Beck convinced Ward Cunningham to use the concept of pattern, which he got to known by [Al64] and [Al79], for a project they both were working on. The goal of the project was to design an user interface for an application. Ward Cunningham came up with a five-pattern language that helped them take advantage of Smalltalk's strengths and avoided its weaknesses, which included the following patterns: Window per Task, Few Panes, Standard Panes, Nouns and Verbs, and Short Menus [BC87]. Ten years later these patterns have been included in a book called *Smalltalk Best Practice Patterns* [Be97].
In 1988, similar work was done by James Coplien. He started to catalog language specific C++ patterns, which he called *idioms*. Three years later this catalog had been published as *Advanced C++ Programming Styles and Idioms* [Co92] including the following definition of idioms, which form the lowest-level patterns.

> Programming idioms are reusable "expressions" of programming semantics, in the same sense that classes are reusable units of design and code.

At the same time as Beck and Cunningham made their first experiences with patterns, Erich Gamma was working on his Ph.D. thesis about object-oriented design [Ga91] and realized the importance of recurring design structures. Discussing his ideas with Richard Helm, Ralph Johnson, and John Vlissides finally resulted in the book *Design Patterns* [Ga95], which includes the following definition for patterns.

> Descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.

One year later Buschmann et al. published the book *Pattern-oriented software architecture* [Bu96] using the following definition.

> A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.

A newer publication by James Coplien and Neil Harrison called *Organizational Patterns of Agile Software Development* [CH04] includes another definition for patterns.

> A pattern is a piece of literature that describes a design problem and a general solution for the problem in a particular context.

The fourth volume of the *Pattern-Oriented Software Architecture* series [BHS07a] defines a pattern as follows:

> A pattern describes a particular recurring design problem that arises in specific design contexts and presents a well-proven solution for the problem. The solution is specified by describing the roles of its constituent participants, their responsibilities and relationships, and the ways in which they collaborate.

Vlissides [Vl97] supplements these definitions by the statement that a pattern should also have the following properties:

Recurrence, which makes the solution relevant in situations outside the original one.

Teaching, which gives you the understanding to tailor the solution to your variant of the problem. Most of the teaching in real patterns lies in the description and resolution of forces, and/or the consequences of application.

Speaking Name, by which to refer to the pattern.

Subsuming these definitions and remarks, the following definition of the term pattern is used in this thesis.

Pattern: A pattern is a general, reusable solution to a common problem in a given context.

This definition is complemented by a description of the constituents of a pattern in Section 3.4.
Typically patterns document proven solutions to recurring problems, but also the opposite approach is known. In cases where no proven practice solution is known, it is still worthwhile to document the solutions, which have proven not to work in order to prevent blind alleys. Such "solutions" are called *anti patterns*. Rising [Ri98] uses the following definition for anti patterns.

> An anti pattern is just like a pattern, except that instead of a solution it gives something that looks superficially like a solution but isn't one.

Cunningham [Cu09] cites Jim Coplien on a definition of anti patterns as follows:

> "An anti pattern is something that looks like a good idea, but which backfires badly when applied." It's not fun documenting the things that most people agree won't work, but it's necessary because many people may not recognize the anti pattern.

There are even whole books available about anti patterns, which do not solely focus on software engineering. In [Br98] Brown et al. describe anti patterns as follows.

> An AntiPattern is a literary form that describes a commonly occurring solution to a problem that generates decidedly negative consequences. The AntiPattern may be the result of a manager or a developer not knowing any better, not having sufficient knowledge or experience in solving a particular type of problem, or having applied a perfectly good pattern in the wrong context. When properly documented, an AntiPattern describes a general form, the primary causes which led to the general form; symptoms describing how to recognize the general form; the consequences of the general form; and a refactored solution describing how to change the AntiPattern into a healthier situation.

A similar definition can be found in a book on anti patterns by McCormick [Mc98].

> AntiPatterns are negative solutions that present more problems than they address. AntiPatterns are a natural extension to design patterns. Understanding AntiPatterns provide the knowledge to prevent or recover from them.

Based on these descriptions and definitions an anti pattern is in this thesis defined as follows.

Anti Pattern: An anti pattern documents a solution to a recurring problem in a specific context, which has proven not to work in practice.

3.2. Related Approaches

Miscellaneous related approaches are known, which follow a similar approach like patterns. They range from ancient Chinese philosophy with the *Yijing* (engl. *The Book of Changes*) as one of the five classics of the Chinese culture [Fl07] to *schemata theory* explicated in the rest of this section.
The term *schema* was originally introduced by the philosophers Plato and Aristotle [Ma95]. In Plato's *The Meno* [Pl99] a schema refers to essential commonalities, e.g. in music, shapes, or what makes up a brave man, for both concrete and abstract concepts [KS08]. Aristotle similarly speaks of form (or schema) when he talks about the

essence or nature of the thing, that is, which basic properties and characteristics make an object distinct [KS08].
Marshall [Ma95] summarizes that a schema is a structural unit that represents a concept, situation, event, plan, behavior, etc. in a generalized form, that is, it contains an abstract representation of multiple instances of the same kind. According to Kohls and Scheiter [KS08], *variables* make up the building blocks of schemata, whereby each variable can take a (constrained) range of values as input. The range of experiences a person had with a specific concept determines the space of variable values and what experiences the person is willing to accept as belonging to the concept without needing to modify the existing schema.
Kohls and Scheiter [KS08] summarize that schemata are organizational structures of memory that interrelate variables that frequently reoccur. The same is true for patterns as they are a logical structure that consists of variables [Al64].

3.3. Documentation of Patterns

Section 3.1 mentioned that one of the goals of patterns is to capture and document proven practices. Subsequently, the steps for doing so are discussed.

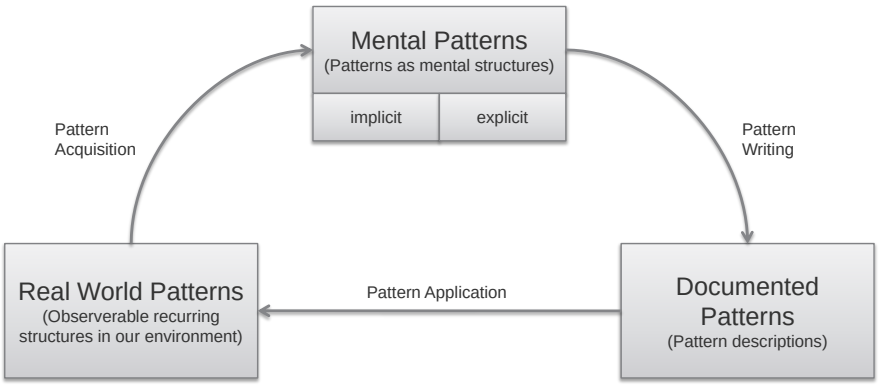

Figure 3.1.: Relationship between real world patterns, mental patterns, and documented patterns [KS08]

Vlissides [Vl97] states that "[...] people have had patterns in their heads for as long as there have been heads. What's new is that we've started naming the patterns and writing them down." This statement is in the following illustrated along an example. Cars exist in the real world (*real world pattern*) and are recognized as cars even though very different kinds of cars exist. Every person has its individual thoughts about what a car is. Some people may think about a sports car, others about a classic car (see Figure 3.2).
People observe a pattern in reality and acquire a *mental model* [PCV02] of the pattern. This is done implicitly and leads to an *implicit mental pattern*. In a next step this mental pattern has to be made explicit as an *explicit mental pattern* to be able to write it down as a *documented pattern*. At last this documented pattern can be applied to future real world problems.

Figure 3.2.: Exemplary images of different cars

In all three steps, the acquisition, the writing, and the application of the pattern, the person who performs these steps plays an important role. Therefore, the same pattern may be documented and understood in different ways by different people. Consequently, it is necessary to distinguish between real world patterns and documented patterns. According to Kohls and Scheiter [KS08] this distinction is acknowledged in the pattern community, although its implications have rarely be discussed.
The result is that no automatic or formal procedure to mine a pattern is available even though there are approaches to enrich patterns by formal methods like e.g. [Ta07] or [Zd07].
Nevertheless, patterns have become accepted in many different disciplines, because it is better to use patterns then to reinvent the wheel every time a problem has to be solved. That is exactly what patterns are all about, they document knowledge about solutions to recurring problems in a way that it can be reused by others having the same problem. Documenting patterns is not simply following an instruction but has to be learned and trained. Meszaros and Doble [MD98] present a pattern language for pattern writing, which documents best practices in how to document patterns and may be used by novice pattern authors as a starting point.

3.4. Pattern Form

The pattern form selected for documenting patterns is important for the applicability of the patterns. Although, various pattern forms are known, there are five essential elements included in all of them. Buschmann et al. [BHS07b] give an overview about these elements and their purpose, which is shown in the following listing.

Identification Name and classification for identifying pattern. The name should be mem-

orable, usable, and distinct. Additionally, it is important for creating a vocabulary for the domain under consideration (see [Co96]).

Context Situation giving rise to a problem. No problem exists in a vacuum, there is always a surrounding context. The context helps to determine where to use the pattern, and provides evidence that it is of general application.

Problem Set of forces repeatedly arising in the context. They constitute the obstacle, which are in the way of producing an efficient/elegant/powerful design. Usually there are some forces in conflict. That is why a pattern is required to solve them. For example, try optimizing code for speed, memory usage, and maintainability. This results in three conflicting forces. Some pattern forms reduce the problem to a single question or a summarizing formulation of the problem [Co96].

Solution Configuration to balance the forces. The solution is a description of the elements of the solution design, their responsibilities, relationships, and collaborations. It is not a concrete design or implementation, but a general, tailorable solution. "The solution part of a good pattern describes both a process and a thing: the 'thing' is created by the 'process'." [Al79]. Furthermore, a pattern tells about a form not only what it is but also what it does [Al64].

Consequences Consequences arising from application of the pattern. These are the results and trade-offs of applying the pattern, including the advantages and disadvantages.

To give a quick overview about eight of the more popular pattern forms the previously described essential elements are used in Figures 3.3, 3.4, and 3.5 as a structuring criteria. The overview is based on analyses by [Cu09, Co96]. Elements of the pattern form, which have not yet been described are described within the table. For a more detailed discussion about the different pattern forms see [Cu09] or [Co96].
All of these pattern forms have their specific advantages and disadvantages depending on the context they are applied in. Patterns may even be transformed between different pattern forms, although in some cases sections have to be added or removed, or have to be further detailed [BHS07b].
Therefore, it can be said that there is no ideal pattern form and selecting an existing pattern form or developing a new one should always be oriented at the experience of the author, the intent that the pattern author has in mind when documenting a pattern, and the target audience [BHS07b].

Essential Elements	Alexandrian Form	Alexander Pattern Language Form	Canonical Form (Coplien Form)
Identification	**Name**	**Name** **Picture:** Showing an archetypical example of the pattern.	**Name** **Alias** (optional)
Context	**Prologue:** One sentence per pattern that can be expected to precede this pattern. **Epilogue:** One sentence per pattern that can be expected to follow this pattern.	**Introduction:** A paragraph, which sets the context for the pattern. **Three delimiting diamonds**	**Context:** Setting the context of the pattern. **Example**
Problem	**Problem statement:** One or two sentences that summarize the problem solved by the pattern.	**Headline:** A headline in bold type, that gives the essence of the problem in one or two sentences.	**Problem** **Forces**
Solution	**Solution:** One or two sentences that tell what to do to solve the problem. **Diagram:** A picture or two, hand sketched or photographed, that illustrate the pattern (and sometimes the lack of the pattern).	**Body:** Presenting the background, motivation, variations. **Solution:** The solution, in bold type that shows how to solve the problem . **Diagram:** A diagram, that shows the solution as a labeled picture. **Three delimiting diamonds**	**Solution**
Consequences	**Discussion:** Anywhere from 4 to 40 paragraphs that illuminate the system of forces resolved by the pattern.		
Other sections		**See also:** A paragraph, ties the pattern to all the smaller related patterns that round out this one.	**Resulting Contex:** What does the context look like after the pattern has been applied? **Rationale:** Why does the pattern work? What is the history behind the pattern? (optional) **Known Uses** **Related Patterns**

Figure 3.3.: Comparison of different pattern forms – Part 1

Essential Elements	Gang of Four Form	Compact Form
Identification	**Name** **Intent:** A short statement that answers the question, what the design pattern does. **Also Known As:** Other well-known names for the pattern, if any.	**Name**
Context	**Applicability:** Showing the situations in which the design pattern can be applied. **Motivation:** A scenario that illustrates a design problem and how the class and object structures in the pattern solve the problem.	**Context**
Problem	**Problem**	**Problem** **Forces**
Solution	**Participants:** The classes and /or objects participating in the design pattern and their responsibilities. **Structure:** A graphical representation of the classes in the pattern using a notation based on the Object Modeling Technique (OMT). **Collaborations:** How the participants collaborate to carry out their responsibilities. **Sample Code:** Code fragments that illustrate how the pattern might be implemented.	**Solution**
Consequences	**Implementation:** What pitfalls, hints, or techniques should the user be aware of when implementing the pattern? Are there language-specific issues? **Consequences:** How does the pattern support its objectives? What are the trade-offs and results of using the pattern? What aspect of system structure can be varied independently?	
Other sections	**Known Uses:** Examples of the pattern found in real systems. **Related Patterns:** What design patterns are closely related to this one?	**Resulting context**

Figure 3.4.: Comparison of different pattern forms – Part 2

Essential Elements	Portland Form	Beck Form	Fowler Form
Identification	**Name**	**Title**	**Title**
Context	**Context:** The context and a brief description of the problem.	**Context** (optional, may include forward or back references)	
Problem	**Problem:** Cause of the problem followed by the forces that must be resolved in order, roughly, from strongest to weakest and with conflicts between the forces highlighted.	**Problem:** Always phrased in the form of a question the reader might have to ask themselves. For instance "How do I sort a Collection?". **Forces**	
Solution	**Solution introduced by "Therefore".** **Solution:** Describe a solution that resolves the strongest forces in this context.	**Solution:** The solution should include the name of the pattern in some form.	**Summary:** Do this so bad stuff won't happen to you.
Consequences			**Discussion:** The bad stuff that should be avoided by doing this pattern, and how this pattern helps to avoid the bad stuff.
Other sections	**Resulting context:** What has been resolved and what needs to be addressed next? What new possibilities are available at this point and what new problems have arisen? What possibilities are no longer available etc. **Summary:** Discussion about the greater context in which this pattern belongs, related patterns, and the specific relationships between those patterns and this one.	**Resulting context:** The resulting context may include forward or back references.	

Figure 3.5.: Comparison of different pattern forms – Part 3

3.5. Summary

This chapter presented the concept of patterns as a way to document proven practices in various domains like education, project management, software engineering, etc. and showed how this concept emerged in recent years. Additionally, related approaches from Chinese philosophy to schema theory were sketched and their influence on patterns was discussed.
How patterns are documented was introduced afterwards. This showed how real world patterns are acquired as mental patterns and are written down as a documented pattern. Additionally, this process was used to show what patterns are and what restrictions exist for patterns.
Concluding the chapter, eight different popular pattern forms were presented and compared to each other, to show that the form of a pattern should be oriented to the domain a pattern is applied in.

CHAPTER 4

EAM Patterns

Contents

Dealing with complexity is one of the most important challenges of system design [Co96]. As an EA can be considered to be a system of systems [Ha05], the same problem applies in this context. This is also one of the reasons why no fully satisfying approach to EA management has yet been developed. This chapter applies the concept of patterns, introduced in the previous chapter to a domain previously not addressed by patterns: EA management.

Proposing a new academic approach based on patterns may be difficult, because they are not invented but they document proven practice. Salingaros [Sa00] summarizes this problem. "A pattern is not usually invented, so creativity is subordinated here to scientific inquiry and observation. Although you can find novel ways to combine and relate patterns, creativity is reserved for the products arising from an application of the pattern language, not the process. Since patterns are derived empirically from

observations, they differ from scientific theory, which derives solutions starting from first principles. Nevertheless, discovered patterns provide a phenomenological foundation out of which scientific theories can grow." This thesis provides such a foundation by applying patterns to EA management.

4.1. Applying Patterns to Enterprise Architecture Management

In order to address the problems stated in Chapter 2, the concept of *patterns* can be applied. Using the pattern concept offers the possibility to profit from additional advantages. For example, according to Harrison and Avgeriou [HA07], an advantage of the pattern concept is that it enables architects to understand the impact of the architectural decisions at design time, because patterns contain information about consequences and context of the pattern usage.
Different definitions for patterns exist, as shown in Section 3.1, but they all exhibit a common ground. Patterns are general, reusable solutions to common problems and are dependent on their context.
These properties are the basis for the EAM pattern approach, which was initially introduced in [Bu07a] and was revised in [Er08]. An EAM pattern is defined as follows.

> **EAM Pattern:** An EAM Pattern is a proven practice-based, general, reusable solution to a common problem in EA management, for a given context, identifying driving forces, denoting known usages, and consequences.

Section 3.1 already showed that patterns also have counterparts, which are called anti patterns. The same is true for EA management, leading to the following definition.

> **EAM Anti Pattern:** An EAM Anti Pattern documents a solution, to a recurring problem in EA management, for a given context, identifying driving forces, and consequences, which has repeatedly proven not to work in practice, in order to prevent typical mistakes in EA management.

In order to clarify the difference and the relationship between the two pattern types a conceptual class diagram using the UML is shown in Figure 4.1
The attributes of the `EAM Pattern` class conform to the sections described in the EAM pattern form (see Table 4.1), except that the *see also* section is represented as the `see also` relationship and that the *problem* section is represented as the self-contained class `Problem`, with a relationship to the `Force` class. The class diagram also shows that three types of EAM patterns exist: M-Pattern, V-Pattern, and I-Pattern.
As already mentioned an `EAM Pattern` may have a `see also` relationship to other patterns, but it only `addresses` exactly one `Problem` as it is advised by Buschmann et al. [BHS07b]. The problem corresponds to a concern as introduced in Chapter 2. Both terms will be used in an equivalent way in the rest of this thesis. A `Problem` can thereby be addressed by more then one `EAM Patterns`. In this case a pattern author has to decide between two alternatives, introduced in [Bu96], in documenting a solution. The first one is to use the variant section of a pattern to describe the alternative solution to

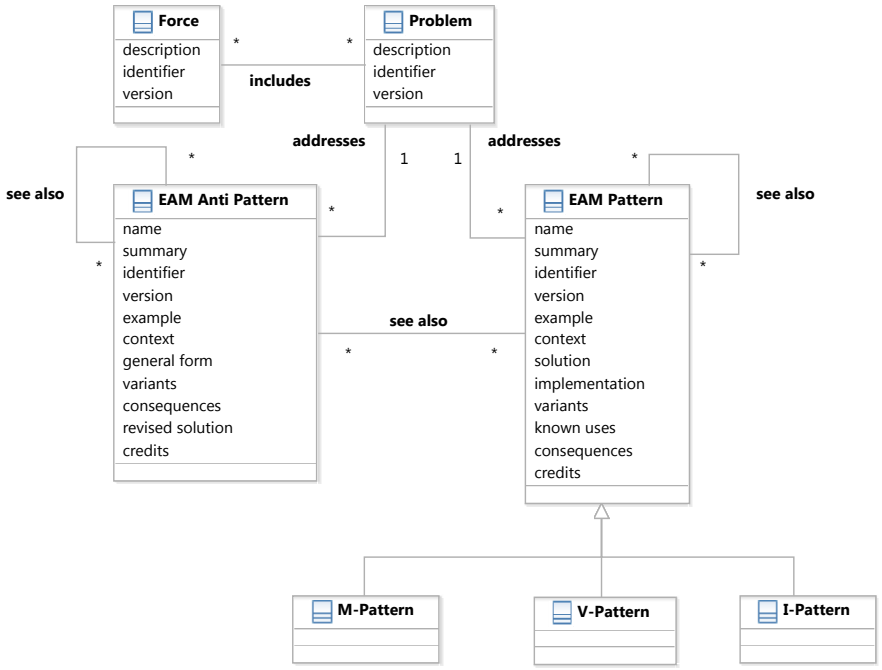

Figure 4.1.: Conceptual model of EAM patterns and EAM anti patterns

a problem. The second one is to create an independent second pattern and referencing it. Every `Problem` may `include` multiple `Forces` influencing the `solution` described in `EAM Patterns` or the `general form` in `EAM Anti Patterns`.
There is no fixed rule for selecting one of the two alternatives. As a guideline it can be said that, if the second alternative is chosen, both patterns should be extensive and self contained enough to be perceived as different entities. Although, references between both pattern should be established to indicate that both constitute solutions to the same problem. This decision is also related to the granularity of patterns and the relationships between them, which is discussed in Section 5.2 in more detail.
The `EAM Anti Pattern` class is the counterpart of the `EAM Pattern` class, which contains similar attributes, except that `solution` is called `general form`, `implementation` and `known uses` are omitted, and a `revised solution` is added . For a detailed elaboration on the differences see Section 4.1.3. The similarities go even further as `EAM Anti Patterns` do also address exactly one `Problem`. Additionally, there are two `see also` relationships. One is a self reference to be able to refer from one `EAM Anti Pattern` to another and the second one is to be able to refer to one or more `EAM patterns`. The last relationship can also be used for references in the opposite direction.
Every `EAM Pattern`, `EAM Anti Pattern`, `Problem`, and `Force` includes an `identifier` and a `version` attribute for consistently maintaining the parts of the EAM pattern and the whole pattern language.
Sections 4.1.1 and 4.1.2 further detail the structure of EAM patterns and their different types, followed by two sections detailing EAM anti patterns.

4.1.1. Form of EAM Patterns

Different forms to document patterns are known (see Section 3.4) in the pattern community. Therefore, a form for EAM patterns has to be selected or defined. According to Fowler [Fo06] selecting a pattern form is a personal choice and should be oriented to ones writing style and the ideas that should be conveyed.

Name of Section	Content of Section
Description	Short summary of the pattern to get a first look at its content.
Identifier	Unique identifier of the pattern to simplify referencing.
Version	Versioning information to keep track of changes on the pattern.
Status	Status information about the pattern: Operational, deprecated, in development.
Example	An example illustrating the problem to be addressed by the pattern. This example should be used by the other parts of the pattern.
Context	The situations in which the pattern may apply.
Problem	The problem a pattern addresses, including a discussion about its associated forces. Only one problem per pattern. Forces are *goals* and *constraints*, which occur in the context.
Solution	The fundamental solution principle underlying the pattern.
Implementation	Guidelines for implementing the pattern. These are only suggestions. E.g. the need to introduce a special board in an organization, a person who has to take care about the creation and the timeliness of a view, or the need to implement a process for data collection, etc.
Variants	A brief description of variants or specializations of a pattern.
Known Uses	Examples where the pattern was used, e.g. usage in companies, tools, books, etc. Due to the restricted maturity of the field of EA management, some patterns are known under different names in different companies. A list of these synonyms can be given in this section.
Consequences	The benefits the pattern provides, and any potential liabilities.
See Also	References to other patterns solving similar problems, and to patterns that help to refine the pattern under consideration.
Credits	Credits to other authors, reviewers and shepherds of the pattern. This section is important, because only a community process guarantees that patterns constitute proven solutions.

Table 4.1.: Template for EAM pattern documentation

EAM patterns follow a template for pattern documentation similar to Buschmann et al. [Bu96], which is closely related to the canonical form (see Section 3.4) and has proven to work in the context of EA management. Five articles [Er08, Bu09a, Bu09e, Bu09g, La09a] using this form were discussed in five writer's workshops at three pattern conferences (Pattern Languages of Programs (PLoP) 2008, European Conference on Pattern

Languages of Programs (EuroPLoP) 2009, and PLoP 2009) and the Patterns in Enterprise Architecture Management (PEAM) 2009 Workshop (see Section 6.7). The form consists of the sections shown in Figure 4.1 and in more detail in Table 4.1.
Contrary to Buschmann et al. [Bu96] the sections *Structure*, *Dynamics*, and *Example Resolved* are omitted and the additional section *Identifier*, *Version*, and *Status* are included. The next paragraphs justify these adaptations.
In [Bu96] the structure section is used to give a detailed specification of the structural aspects of the pattern (e.g. using UML). Such a representation is only used in the information model patterns (see Section 4.1.2.3) and in this case this representation is put in the solution section as they are the solution for a problem concerning the construction of an information model.
The dynamics section in [Bu96] presents typical scenarios describing the run-time behavior of the pattern. Although this might be of interest in the future development of M-Patterns this section is currently not included in the EAM pattern form.
Example resolved is also not needed in the EAM pattern form, because the example in an EAM pattern is usually described in a textual way and the solution section already presents a solution. Therefore, an example resolved section can be omitted.
The sections identifier, version, and status were included to support the future development and maintenance of the currently available EAM patterns, as a pattern based approach demands for a continuous improvement of the included patterns (see Section 5.5).
The selected template (see Table 4.1) has the additional advantage that references between different EAM patterns are presented in a concise way, which helps to keep them consistent. This is essential in a larger pattern language. For an in depth discussion of this topic see Section 5.
Buschmann et al. [Bu96] advise to use a consistent template for documenting a pattern language due to the following reasons.

Readability and Usability: Patterns become easier to understand and to use, if they are documented in the same way.

Comparability: Patterns can easier be compared, especially when looking for alternative solutions to a problem.

For these reasons the EAM Pattern Catalog uses the same template for all EAM patterns.
Figure 4.1 in the previous section indicates that there are three specializations for an EAM pattern: Methodology pattern, viewpoint pattern, and information model pattern. Reasons for the distinction between those three types of EAM patterns are given in the following section.

4.1.2. Three types of EAM Patterns

Gamma et al. [Ga95] distinguish between *creational patterns*, *structural patterns*, and *behavioral patterns*. Creational patterns like *abstract factory* care about class instantiation, structural patterns like *adapter* focus on class and object composition, whereas behavioral patterns like *visitor* are concerned with communication between objects.
The EAM pattern approach also distinguishes between different pattern types (see Section 4.1.1). This separation reflects the constituents of a method (metamodel, language

(or modeling techniques), and defined steps) introduced in Section 2.1. This is complemented by guidelines for applying the method, the fourth part demanded by Kronlöf et al. [KSH93]. The result is a pattern-based approach to EA management which is aligned to typical characteristics of an engineering discipline
Additional reasons for the different EAM pattern types have emerged during the application of the EAM pattern approach. *Usability* is the first reason. Different stakeholders have different concerns, e.g. someone, who has to visualize the processes to be used for EA management in a company in the future, is not interested in details about the information to be used, or a stakeholder, who has to specify the information model to be used in an EA management approach, is not interested in the management processes of the approach, etc. Separating EAM patterns in different types therefore supports stakeholders in finding the right patterns for their concerns, without having to consider the whole collection of EAM patterns.
The case studies, which will be presented in Section 6 showed that this is an important aspect for the acceptance of the EAM pattern approach. They also showed that the EAM patterns should retain a manageable size. Holschke et al. [HRL09] also emphasize that the granularity of models has an important influence on the reuse of these models. Integrating processes and roles with one or more viewpoints and one or more information model fragments in one pattern would result in the opposite, patterns that become to large to be usable and manageable.
Separation between information and its representation is another reason for the distinction between I-Patterns (instantiated by semantic models) and V-Patterns (instantiated by symbolic models).

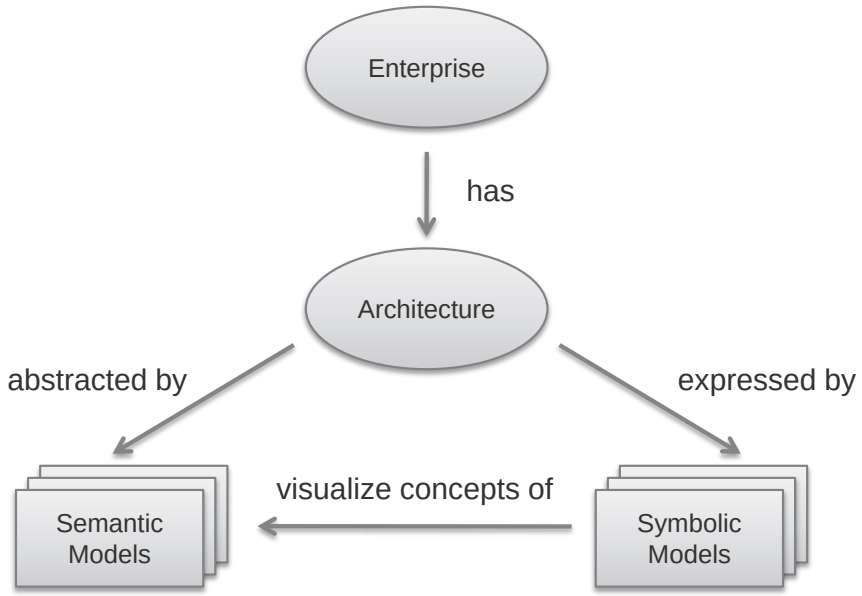

Figure 4.2.: Separation between semantic and symbolic models (based on [Ar07])

Figure 4.2 shows that the *enterprise* has an *architecture*, which is abstracted by *semantic models*. The semantic models include the actual information objects, which describe the

EA irrespective of their representation. The *symbolic models* express the architecture and describes how a visualization of concepts of the semantic model can look like [Ar07]. This concept is called a *view* in [Ie00], which uses a similar distinction between a model (semantic model) and a view (symbolic model).
Semantic and symbolic model can be further detailed by an information and a visualizations model. See [Er06, Bu07b] and Figure 4.3 for more details.

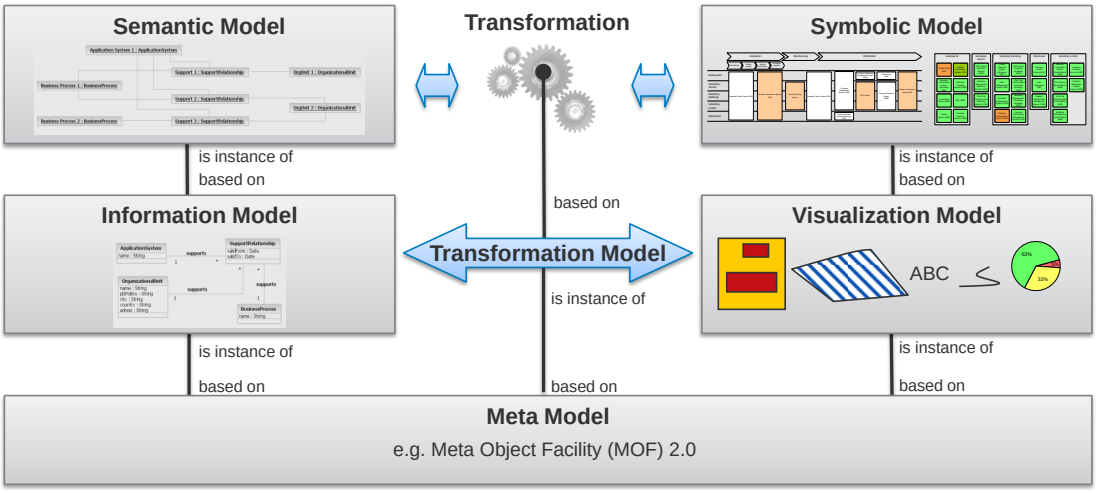

Figure 4.3.: Relationship between semantic, symbolic, information, and visualization model [Bu07b]

Ernst [Er06], Buckl et al. [Bu07b], and Arbab et al. [Ar07] suggest such a separation, although there is a tight relationship between those two aspects. A visualization always relies on some information which is visualized. The separation offers a higher flexibility, e.g. in cases where the same information is to be visualized in different ways. Furthermore, this emphasizes the building block-like approach of EAM patterns.
The three introduced EAM pattern types are defined and detailed in the following sections.

4.1.2.1. Methodology Patterns

The first EAM pattern type introduced in this thesis is the methodology pattern (M-Pattern).

> **Methodology Pattern (M-Pattern):** A methodology pattern documents a proven practice solution to a recurring problem for a specific context in form of a process for the management of an enterprise architecture. It also documents roles, the steps to be taken in the process, inputs and outputs of the process, as well as known variants, and consequences related to its usage. The documented process can use one or more viewpoint and information model patterns during its execution.

The processes defined by an M-Pattern comply with *defined steps and ordering of these steps* in the definition of a method by Kronlöf et al. [KSH93]. They can be very different,

ranging from coarse grained ones like EA Documention to more fine grained ones like Centralized Manual Data Acquisition/Maintenance. They may thereby suggest to use different instruments like visualizations, group discussions, or more formal techniques as e.g. metrics calculations [LS08], etc.
M-Patterns have been introduced, because missing processes and roles constitute a common issue in current EA management approaches [Ma09]. Frameworks as e.g. TOGAF [Op09a] provide a process model (TOGAF ADM) (see Section 2) but leave the details of the processes and roles supporting the specific activities in the EA management process relatively open. M-Patterns explicate the processes and roles in order to complement activities carried out in an ad-hoc manner or relying on implicit knowledge with activities carried out more systematically [Bu09f].
Examples for M-Patterns of the EAM pattern approach can be found in Section A.1 and in [se09a].

4.1.2.2. Viewpoint Patterns

Viewpoint patterns *(V-Patterns)* provide a modeling technique used by one or more M-Patterns and thus proposes a way to present data stored according to one or more *I-Patterns*. The modeling technique corresponds to the *language* as one constituent of a method, defined by Kronlöf et al. [KSH93].

> **Viewpoint Pattern (V-Pattern):** A viewpoint pattern documents a proven practice solution to a recurring problem for a specific context in form of a viewpoint for the creation of views. It also documents techniques for view creation and usage, as well as known variants, and consequences related to its usage. The documented viewpoint is a representation or input method for information, which can be stored according to one or more information model patterns.

This definition for a V-Pattern relies on the definition of a viewpoint in [Ie00]. Even though, a viewpoint is usually interpreted as a graphical representation this need not be the case. A viewpoint can be graphical, but also tabular or text based ones are known and documented. Even a survey form could be a viewpoint but with a special focus on collecting and not on presenting information. Such an information collection approach could also be implemented by editing a graphical visualization, which is one functionality analyzed in [se05, Ma08a].
The research project *Software Cartography* (see e.g. [LMW05, Bu07a, Bu07b, Wi07a]), found out that industrial users often specify viewpoints by example. This means that an exemplary view is provided for the viewpoint, possibly together with some textual explanations. Such an approach may be sufficient in certain use cases, e.g. sketching concepts in presentations, but problems may arise, when the goal is to provide *official* information to a wider audience for an extended period. In order to ensure the understandability of a graphical representation, a legend is regarded to be mandatory.
Examples for V-Patterns of the EAM pattern approach can be found in Section A.2 and in [se09a].

4.1.2.3. Information Model Patterns

Information model patterns *(I-Patterns)* supply an underlying model for the data visualized in one or more V-Patterns or used in one or more M-Patterns. Like for the two previously introduced EAM pattern types this one can also be mapped to one constituent of a method in the definition by Kronlöf et al. [KSH93]. In this case it is the *underlying model*.

> **Information Model Pattern (I-Pattern):** An information model pattern documents a proven practice solution to a recurring problem for a specific context in form of an information model fragment for the creation of an information model. It also includes definitions and descriptions of the used information objects, documents techniques for information model fragment implementation and usage, as well as known variants, and consequences related to its usage.

Documenting an information model fragment always relies on a certain language, in which the elements of the fragment are expressed. Basically a lot of different languages suitable for that kind of conceptual modeling are known. The following list is intended to show some of the more prominent examples, outlining both advantages and disadvantages of the languages.

Textual description in natural language: The model elements and their relationships are described in natural language. This seems to be a good choice as it produces easily understandable and adaptable descriptions. The main disadvantage is that this kind of documentation easily leads to mistakable constructs, insufficient for exactly defining information model fragments.

MOF and UML class models: The model elements and their relationships are described via object-oriented concepts, captured e.g. in the UML or MOF 2.0 metamodel. This description can rely on UML class models, which should be understandable by most developers. A disadvantage of these languages is that they lack a formal basis and hence possibly limit the domain in which these kind of information model fragments can be used.

Ontology languages: The model elements and their relationships are described in terms of an ontology language. Such a language might provide more expressiveness than the approach of MOF or UML and is based on a formal foundation including the possibility to use automated reasoning in the model. Disadvantageous is, that ontology languages are not so widespread in the area of conceptual modeling.

Mathematical formalization: The model elements and their relationships are described in mathematical or logical terms, providing a strong formal background. The main disadvantage is, that mathematical or logical models are usually difficult to use.

Of course, the possibility to combine two or more of the variants described above could unleash the advantages of the languages combined, possibly without having to consider the individual disadvantages. Nevertheless, one disadvantage seems inevitably related

to such a combined approach, namely the effort for describing an information model fragment in both languages and keeping these descriptions consistent.
With different possible languages for describing information model fragments and considering their individual advantages and disadvantages, some languages may be more or less adequate for a specific pattern. This is especially obvious, if the information model fragment under consideration is only used for creating a visualization of the application landscape or a tabular report, where an object-oriented description should be sufficient. For other use cases, e.g. the calculation of metrics or the simulation of processes, the situation looks different, as this may only be possible in a reasonable way, if the information model has a more formal basis.
Therefore, a language adequate to the problem addressed should be used. In most cases it is suggested to use UML as the language of choice, as this language is widely understood and was found to be problem-adequate in many situations [Bu07a].
There is also a second reason for suggesting *one* language. Utilizing the pattern-based approach for information modeling, a crucial point is the *integration* of information model patterns to form a complete information model (see Section 5.4.2). This integration can be regarded more simple, if the different patterns are specified using one language, which cuts the effort of translating the models between languages.
Nevertheless, there might be situations in which it seems advantageous to use a language different from the default one, as e.g. UML would lead to a documentation far too extensive or not expressive enough for the problem at hand and therefore would not be problem adequate. In such cases, it can be advisable to support the information model fragment modeled in a problem adequate language with a corresponding information model fragment documented in the default language, as this simplifies the integration of the patterns.
Examples for I-Patterns of the EAM pattern approach can be found in Section A.3 and in [se09a].

4.1.3. Form of EAM Anti Patterns

EAM anti patterns are the counterpart of EAM patterns (see Section 4.1). Their form is similar to the form of EAM patterns with slight modifications because of the different function of EAM anti patterns. Changes to the form presented in Table 4.1 are inspired by Brown et al. [Br98]. The EAM anti patterns form is shown in detail in Table 4.2.
Compared to the EAM pattern form the solution section was renamed to *general form* to prevent misinterpretations. The *implementation section* was omitted because it shows how an EAM pattern should be implemented, which is not required in an anti pattern. Another section – known uses – was removed, due to the reason that it is hard to find companies, which agree to be named for a not working practical example. In contrast one section was added to the EAM pattern form. The *revised solution* section proposes a working solution to the problems revealed by the anti pattern.

4.1.4. EAM Anti Patterns

EAM anti patterns are the latest extension to the EAM pattern approach and were introduced in [Bu09g]. Section 4.1 defined EAM anti patterns as a way to document

Name of Section	Content of Section
Description	Short summary of the pattern to get a first look at its content.
Identifier	Unique identifier of the pattern to simplify referencing.
Version	Versioning information to keep track of changes on the pattern.
Status	Status information about the pattern: Operational, deprecated, in development.
Example	An example illustrating the problem to be addressed by the pattern. This example should be used by the other parts of the pattern.
Context	The situations in which the pattern may apply.
Problem	The problem a pattern addresses, including a discussion about its associated forces. Only one problem per pattern. Forces are *goals* and *constraints*, which occur in the context.
General form	The recurring, not working solution found in practice.
Variants	A brief description of variants or specializations of a pattern if available.
Consequences	The benefits and liabilities of the not working solution documented in the general form.
Revised solution	A revised solution to the problems presented in the general form section.
See also	References to other patterns solving similar problems, and to patterns that help to refine the pattern under consideration.
Credits	Credits to other authors, reviewers and shepherds of the pattern. This section is important, because only a community process guarantees that patterns constitute proven solutions.

Table 4.2.: EAM anti pattern form

solutions, which have proven not to work in practice in order to prevent common mistakes in EA management. At first sight it may look unnecessary to document solutions, which do not work. But it may be valuable for people, who are new to EA management and are therefore not experienced, to have a resource available, which documents such solutions. Similar approaches can e.g. be found in *incident management* in the *IT Infrastructure Library (ITIL)* [CW07]. In this case incidents and their solutions are documented and used for future processing of incident reports.
Ambler [Am08] sketches anti patterns for documenting bad-practices in EA management, similar to EAM anti patterns. The following list gives an excerpt of these anti patterns:

Detailed Enterprise Modeling The enterprise model(s) are overly detailed, often in an attempt to comprehensive define what the enterprise does (or should do).

Ivory Tower Architecture Your enterprise architecture model(s) reflect a wishful, perfect world scenario instead of the realities of your actual environment.

Real World Disconnect The enterprise models reflect the vision, and the misun-

derstandings, of the modelers but do not reflect what your business stakeholders actually require.

Technology Above All The architects make technology a business driver instead of a business enabler.

Examples for EAM anti patterns can be found in Section A.4 and in [se09a, Bu09g]. For simplicity reasons EAM anti patterns and EAM patterns are subsumed under the term EAM pattern in the rest of this thesis, unless otherwise noted.

4.2. Using EAM Patterns

There are currently four different usage scenarios documented for the pattern-based approach to EA management, which are introduced and discussed in this section. These scenarios were evaluated in practice in six case studies, which are presented in Section 6.

4.2.1. Establish an EAM approach

The pattern-based approach to EA management supports introducing a light-weight, enterprise- and situation-specific approach to EA management based on best practices. It is assumed here that EA management is introduced in a green field approach. In this case, first of all the pain points of the company, the so called problems or concerns have to be identified. This is supported by a list of concerns supplementing the EAM patterns. One or more collections of EAM patterns, supplied by pattern authors, serve as a basis. From these collections, the developers for an EA management choose EAM patterns that are perceived as adequate for addressing specific concerns of the respective enterprise, preferably under participation of the prospective users. In this selection process M-Patterns, V-Patterns, and I-Patterns have to be considered. Hints, which EAM patterns to include additionally can be found in the *see also section* of the patterns. After integrating EAM patterns, thereby creating a coherent, light-weight, enterprise-specific *EA management method*, the respective concepts can be implemented, e.g. in an EA management tool or a suite of tools as an enterprise-specific EA management approach. This process is shown in Figure 4.4.
The process offers the possibility to incrementally implement an EA management approach, starting with an initial set of M-Patterns, V-Patterns, and I-Patterns, which on the one hand includes rationale for the decisions made, e.g. why certain elements of the information model have been selected and on the other hand can later be extended, e.g. when a higher maturity level has been reached. In this case, a map of EAM patterns (see Figure 5.10 in section 5.2), which supplements the EAM patterns, can be used to e.g. identify EAM patterns, which easily fit to the already selected EAM patterns due to being closely related.
For example, it can be possible to create additional visualizations using the information already collected. In this case the I-Patterns, which are already in use have to be revisited and then further V-Patterns have to be found, which use the same information.
The same is true for M-Patterns, as they use V-Patterns in their documented processes. Therefore, it may be possible that V-Patterns, which are already in use, can be utilized to address additional concerns with additional M-Patterns.

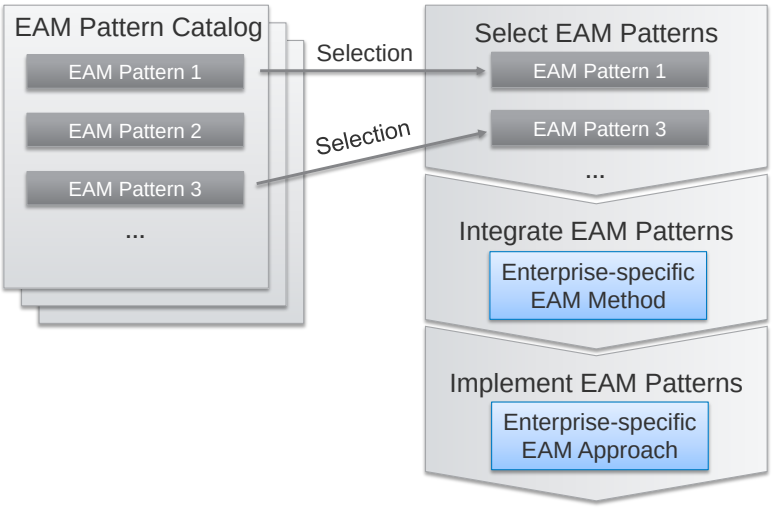

Figure 4.4.: The process for establishing an EAM approach based on EAM patterns

4.2.2. Inspire and Assess an existing EAM approach

The second usage scenario for the EAM pattern approach is to take it as a reference book for suggestions concerning the EA management approach currently in use in a company. This offers the possibility to compare the own EA management approach with best practices in other companies experienced in EA management.
The collection of EAM patterns may e.g. be used to look for typical concerns, which occur in other companies. This case may best be addressed by simply paging through the EAM pattern language.
Additionally the collection of EAM patterns may suggest visualizations that can be found in academia and practice, which may be helpful in the currently selected EA management approach.
In these cases the EAM pattern map can be used to find EAM patterns to address these concerns.

4.2.3. Specify Goals and Requirements for EAM

Specifying goals and requirements is always a difficult task. The same is true in the context of EA management.
Consider a company thinking about introducing EA management as a way to improve the alignment between business and IT. In this case a company typically has to decide between working with a consulting company in order to define concrete goals for the initiative or to try to define these goals by itself. Using the pattern-based approach to EA management a company has the possibility to use the list of concerns, which can typically be found within companies in the context of EA management, and use these to define its goals.
The situation is similar during requirements elicitation, e.g. for an EA management tool. One approach to find the existing requirements is to interview the affected stakeholders (if they are known or have previously been identified) what their requirements are. Usually

this results in a "make a wish" contest creating a long list of requirements, which does not reflect the real requirements of the company. Using the pattern-based approach to EA management this situation can be changed due to the following reasons. Selecting requirements from an existing list is always easier then creating an own list and selecting from this list. Another advantage is that the list of concerns is based on proven practices and can therefore easier be communicated to decision makers. In addition it is easier to express requirements, e.g. it is easier to talk to a tool vendor and tell him that one requirement is to use a visualization according to Business Application and Organizational Unit Cluster Map or that there is a requirement to document information according to Architectural Solution Conformance, compared to a document textually describing these requirements.
In both cases it is better to use the documented concerns or the documented EAM patterns, compared to using a newly developed, incomplete, and unclear textual description. Especially as concerns and EAM patterns are based on proven practices and not invented in an ad-hoc manner.

4.2.4. Academic research

In addition to the application of the EAM pattern approach in practice, it may also be used as a basis for future academic research. Currently, there is no common ground for research on EA management, meaning that there is no approach for EA management, which may be iteratively enhanced and extended. There are punctiform approaches for specific EA management topics, but these lack the integration into a holistic EA management approach, and the acceptance in the EA management community.
The pattern-based approach to EA management addresses this deficiency as it offers the possibility to improve single EAM patterns without having to create a completely new approach. Furthermore, the existing collection of EAM patterns can easily be extended due to the openness of the pattern-based approach.
To establish a community for the EA management pattern approach, articles were published, workshops organized (see Section 6.7), and presentations were given. This community will govern the future development, by e.g. performing reviews of patterns, improve existing patterns, extend the collection of available patterns, etc. See Section 5.5 for more details on these activities.

4.3. Summary

This chapter presented the application of patterns and anti patterns, introduced in the previous chapter, to EA management. Thereby, the terms EAM pattern and EAM anti pattern were defined and their form was detailed.
EAM patterns were additionally distinguished in M-Patterns, V-Patterns, and I-Patterns. M-Patterns document solutions to address problems in EA management in a step wise manner. V-Patterns document visualization, which have proven to work in practice and present information stored according to one or more I-Patterns.
Instead of proven practice solutions documented in EAM patterns, EAM anti patterns document solutions, which have proven not to work in practice, in order to prevent blind alleys.

Afterwards four different usage scenarios were introduced: Establish an EA management approach, inspire and assess an existing EA management approach, specify goals and requirements for EA management, and academic research. For each of these scenarios, the benefits and guidance for utilizing the approaches was given.

CHAPTER 5

EAM Pattern Catalog

Contents

The *EAM Pattern Catalog* is a collection of EAM patterns and has been created in two phases as part of the *Enterprise Architecture Management Viewpoint Survey*, in order to fulfill the requirement that a pattern should be a reusable solution to a common problem observed in practice. This is often referred to as *the rule of three* [Co96] in the pattern community which means that a solution documented as a pattern is an abstraction of at least *three* occurrences.
During *phase one* – Initiation – (October 2006 until July 2007) the initial version of the EAM Pattern Catalog had been created by the Chair for Informatics 19 (sebis) using the following resources:

- Research project Software Cartography, Technische Universität München, Chair for Informatics 19 (sebis) (e.g. [LMW05], [Bu07b], [Bu07a], [Wi07a])
- Partners of the research project Software Cartography
- EAM Tool Survey 2005 [se05]
- Enterprise Architecture at Work (ArchiMate), 2005, Marc Lankhorst et al. (Telematica Instituut) [Jo05]
- Management von IT-Architekturen (Edition CIO), 2006, Gernot Dern [De06]
- IT-Unternehmensarchitektur, 2007, Wolfgang Keller [Ke07]

In this phase the above resources were analyzed and consolidated, e.g. the identified visualizations were grouped according to their layout and the EA management topic area they are used in. Afterwards, the identified and abstracted solutions were document in form of patterns.
Following, the information model fragments were derived from the previously documented V-Patterns and were documented in the I-Patterns. This is possible as it is according to Buckl et al. [Bu07b] possible and advantageous to generate visualizations for EA management based on a defined information model. The approach was inverted in this case to reengineer the information model of a certain visualization. This is similar to the approach proposed by Kurpjuweit and Winter [KW07]. An additional consolidation of the I-Patterns was required for the information model fragments because multiple V-Patterns were identified, which needed a similar kind of information model for its generation. Therefore, I-Patterns documenting similar information were integrated. This is one of the cases where the building block-like approach of the EAM Pattern Catalog can be observed, which emphasizes the reuse of patterns to build up a new EA management approach. In a last step the respective M-Patterns were derived based on the consolidated V-Pattern and the sources listed above. Supplementary, the list of concerns, which are addressed by the EAM patterns was compiled by extracting the concerns from the problem statement of the patterns. The result of this first phase was a set of of 170 EAM patterns.
In *phase two* – Evaluation – (July 2007 until February 2008) the initial version of the EAM Pattern Catalog was evaluated in an extensive online survey during the *Enterprise*

Architecture Management Viewpoint Survey by 30 participants from the following 26 European companies: Allianz Group IT, AXA, BSH, Bosch und Siemens Hausgeräte, BMW Group, BTC, cirquent Softlab Group, Credit Suisse, Deutsche Bahn, Deutsche Börse Systems, Deutsche Post, Deutsche Telekom, EWE, FJH, FIDUCIA IT, HVB Information Services, Krones, Kuehne + Nagel, Münchener Rück, Nokia Siemens Networks, O2 Germany, RWE, Siemens CIO, Siemens PG, Siemens IT Solutions and Services, Wacker Chemie, Zollner Elektronik.
In this online survey the participants rated the previously documented EAM Patterns concerning their relevance and usefulness in practical use. The whole survey took around three hours to completion. The reason for this extent is that the survey analyzed EA management in a holistic way without focusing too much on a special topic. A more detailed analysis of the various topics in EA management was conducted in phase three. As a result of this evaluation 50 patterns were excluded from the EAM Pattern Catalog, resulting in version 1.0 [Bu08a], which includes 120 EAM patterns. For detailed results of the evaluation, including the results for each analyzed EAM pattern, see [Bu08a].
Afterwards, the EAM Pattern Catalog approach was presented to the existing pattern community, which is organized by the *Hillside Group* [Hi09]. To achieve this, the approach had been discussed during the PLoP 2008 conference [Hi08], resulting in a revised pattern form, which has already been described in detail in Section 4.1.1. A detailed listing of changes to the EAM pattern template is given in [Er08].
Currently the development of the EAM Pattern Catalog is in *phase three* – Revision and Improvement – (since February 2008). In a first step the EAM Pattern Catalog has been transformed from a PDF file to a Wiki based platform, the so called EAM Pattern Catalog Wiki (http://eampc-wiki.systemcartography.info/), which is detailed in Section 5.6, to better support community aspects and to improve usability. This wiki currently includes 164 (as of 2009-09-05) EAM patterns.
In addition to the previously described tasks, the EAM Pattern Catalog is continually extended and revised by theses and lectures (see e.g. Sections 6.1, 6.2, 6.3, and 6.8), community members (see e.g. Sections 6.4, 6.5, and 6.6), workshops (see Section 6.7). An online survey (see Section 7.3) was conducted to validate the approach and identify potential for future improvements.

5.1. Pattern Language for Enterprise Architecture Management

The pattern community distinguishes three types of pattern collections, which give an indication concerning the maturity of the corresponding pattern approach. These three types are subsequently introduced to allow for a classification of the pattern-based approach to EA management.
The first type is a *Pattern Catalog*, for which the following definition by Buschmann et al. [Bu96] is used in this thesis.

> **Pattern Catalog:** A pattern catalog is a collection of related patterns, where patterns are subdivided into a small number of broad categories. It does not describe how patterns are tied together.

To visualize the relationships between patterns typically a simple graph representation is used, where patterns are visualized as nodes and relationships as edges. Figure 5.1 uses this type of representation to show that the different patterns in a pattern catalog are not connected to each other. The arbitrary positioning of nodes has no meaning in such representations.

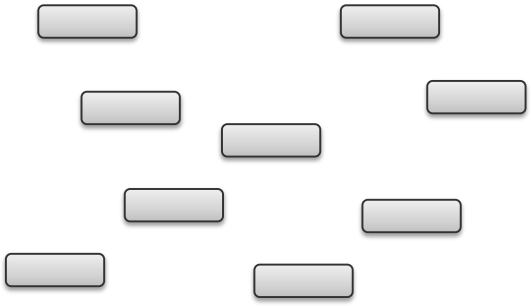

Figure 5.1.: Relationships in a pattern catalog

The second type of a pattern collection is a *Pattern System*. Buschmann et al. [Bu96] defines this term in the context of patterns for software architecture as follows:

> **Pattern System:** A pattern system for software architecture is a collection of patterns for software architecture, together with guidelines for their implementation, combination and practical use.

Based on this definition a pattern system is a pattern catalog, which is additionally enriched by guidelines for the implementation, combination, and practical use of the patterns. The guidelines are thereby part of the patterns themselves. A pattern system may also include additional guidelines for instantiating patterns. An example for patterns which can be used in combination is shown in Figure 5.2.

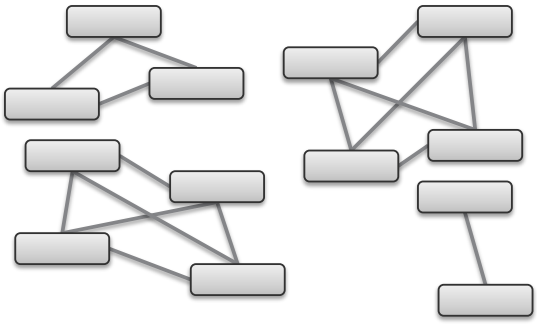

Figure 5.2.: Relationships in a pattern system (based on [Sa00])

The third type is a *Pattern Language*. Alexander [Al79] gives the following definition:

> **Pattern Language:** The elements [of a pattern language] are patterns. There is a structure on the patterns, which describes how each pattern is itself a pattern of other smaller patterns. And there are also rules, embedded in the patterns, which describe the way that they can be created, and the way that they must be arranged with respect to other patterns. However, in this case, the patterns are both elements and rules, so rules and elements are indistinguishable. The patterns are elements. And each pattern is also a rule, which describes the possible arrangements of the elements – themselves again or other patterns.

Summarizing, patterns are both, the vocabulary of the language, and the rules of their implementation and combination which make up its grammar. This implies that the patterns of a pattern language cover every aspect of importance in a particular domain, which is called the *completeness* of a pattern language. The rules by which the patterns are connected are just as important as the patterns themselves [Sa00]. This makes it more difficult to describe a pattern language compared to a pattern catalog as the language tells, which patterns can be combined and in which manner.
According to Hanmer [Ha03] a pattern language description usually includes:

Language intent A short description of the objective of the language, like an abstract for the language.

Language map A diagram that shows an example of how the patterns build upon and relate to each other.

Language description A specification of the pattern language, which is morphologically and functionally complete.

Patterns The Patterns that make up the language.

A pattern language can be seen as a completely connected network of patterns (cf. Figure 5.3).
As a conclusion of this classification it can be said, that the initial version of the EAM Pattern Catalog is a pattern catalog. This explains the name EAM Pattern Catalog, because it was originally created with minor focus on relationships between the EAM patterns. Although relationships between EAM patterns were already included it was not the goal to develop a completely connected network of patterns.
In version 1.0 of the EAM Pattern Catalog the pattern-based approach to EA management evolved to a pattern system because of the included guidelines like the EAM pattern map for using and combining EAM patterns.
Currently, the EAM Pattern Catalog is becoming a pattern language, because the relationships between EAM patterns gain greater importance. An example for the greater importance of the relationships is the adapted pattern template. Since the modifications proposed in [Er08] the relationships are explicitly given in a separate section – see also – of the EAM pattern template. Additionally, the pattern language is becoming more and more complete due to the contributions by community members, and conducted workshops. Although a complete pattern language for EA management is a visionary

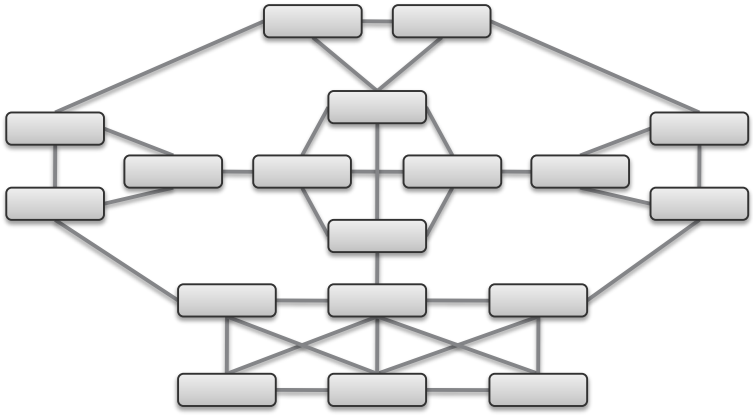

Figure 5.3.: Relationships in a pattern language (based on [Sa00])

goal like for all other pattern languages. For these reasons the EAM Pattern Catalog is considered to be a pattern language.
Measuring the completeness of a pattern language is a difficult task as it is hard to state an objective criterion for the completeness [WV03]. Therefore, Alexander et al. [AIS77] state that "a pattern language is good when it is *morphological complete*", meaning that patterns fit together to form a complete structure without gaps. This leads to the consequence that when new topics become important in the context of EA management a so far complete pattern language for EA management may become incomplete. Another criterion is *functional completeness*. This means that all the internal forces are resolved. When patterns are applied they introduce new forces. A language is functionally complete when these internal forces are covered [Ha03].
Those two criteria may be used to determine the current completeness and topics where no patterns have yet been documented. Such topics may be areas of further academic research (see Section 4.2.4). Such an analysis has not yet been conducted but will be conducted in the future development of the pattern-based approach to EA management. The following section analyzes relationships between EAM patterns in more detail.

5.2. Relationships between EAM Patterns

The previous section emphasized that relationships are an integral part of a pattern language. Therefore, this section details on the kind of relationships existing in a pattern language in general and in the EAM Pattern Catalog specifically.
Different ways to classify relationships between patterns have been presented in literature [No98]. Zimmer [Zi94] introduces three different relationships, which are based on the relationships used in [Ga95]:

- One pattern is *used* in the solution of another pattern.
- One pattern is *similar* to another pattern.
- One pattern can be *combined* with another pattern.

In terms of above the statement that one pattern uses a second pattern means that the second pattern must be applied before the solution described by the former pattern will be complete [No98].
These three relationships are extended by Alpert et al. [ABW98] by a specialization relationship:

- One pattern can be a *specialization* of another pattern.

According to Zimmer [Zi94] three relationships are used in [Bu96]:

- One pattern can *use* another pattern.
- One pattern can be a *variant* of another pattern.
- Two patterns can be used in *combination* to solve a problem.

In a later volume of the *Pattern-Oriented Software Architecture* series [BHS07a] an additional relationship type was added: A pattern can be a *(competing) alternative* to another pattern.
Another different set of relationships is used by Meszaros and Doble [MD98]:

- A pattern can *use* another pattern.
- One pattern is *used by* another pattern.
- A pattern *generalizes* other patterns.
- One pattern *specializes* another pattern.
- A pattern provides an *alternative* to another pattern

The difference in naming and using relationship types in different pattern languages leads to a problem, when trying to organize patterns from different sources. Therefore, a classification for relationships was developed by Noble [No98] and is further detailed in the following section.

5.2.1. Primary and Secondary Relationships

The approach presented in [No98] distinguishes between *primary relationships* and *secondary relationships* based on three criteria: Primary relationships are ubiquitous in the patterns literature, they can be used to describe a large number of other secondary relationships, and their definition is straightforward, because they are similar to the familiar relationships in object-oriented design.
The primary relationships proposed by Noble are given in Table 5.1. Row number one gives the names of the relationship type, followed by a short description, exemplary utilization in literature, and exemplary usage in the EAM Pattern Catalog. Classifying relationships used in the EAM Pattern Catalog to the relationship types proposed by Noble [No98] offers the possibility to organize EAM patterns in the same way as other patterns using the same relationship types. This improves the utilization and comparability of EAM patterns.

Name	Description	Used in Literature	Used in the EAM Pattern Catalog
Uses	One pattern uses another pattern	[AIS77, Ga95, Zi94, WV03]	Business Application and Organizational Unit Cluster Map *uses* Business Application and Organizational Unit Relationship
Refines	A specific pattern refines a general pattern	[ABW98, MD98, WV03, KS08]	Standards Conformity Exceptions *refines* Business Application and Organizational Unit Cluster Map
Conflicts	A pattern addresses the same problem as another pattern	[DA98, Ga95, Be97]	Centralized Manual Data Acquisition/Maintenance *conflicts* Decentralized Manual Data Acquisition/Maintenance

Table 5.1.: Overview about the primary relationship types between patterns (based on [No98])

The *uses* relationship is the most common relationship in the EAM Pattern Catalog. According to Hanmer [Ha03] each pattern builds upon the patterns that came previously and each pattern creates the context for the following patterns. This is what the uses relationship is all about: One pattern uses the results of another pattern as its input. Thereby an ordering between the patterns is created.
Refines is another primary relationship type. It can be compared to the *inheritance* relationship in UML. In this case one pattern can be used to refine another pattern or to identify new abstract patterns by generalizing existing patterns via detaching their commonalities [No98]. This usually results from different levels of abstraction of the patterns. Another name for this relationship, e.g. used by Kohls and Scheiter [KS08] is *is-a relationship*.
The *conflicts* relationship expresses that two patterns present a solution to the same problem, but they cannot be combined and are therefore conflicting.
Secondary relationships are shown in Table 5.2, which has the same structure as Table 5.1. Not all relationship types given in the table are used in the EAM Pattern Catalog. In these cases the respective cell is left empty and an explanation is given in the text. Secondary relationship can be expressed by combining primary relationships. Noble [No98] gives an example for combining primary relationships for the Adaptor pattern variants in [Ga95].
Used by and *refined by* are inversions of the uses and the refines relationship. Hence the examples given in Table 5.2 are also inversions of the examples given in Table 5.1.
The *variant* relationship is used to indicate patterns that solve the same problem in

a different way. These patterns can anyway be combined to a solution, in contrast to patterns, which are in a conflicts relationship. As already detailed in Section 4.1.1 and introduced by Buschmann et al. [Bu96] this kind of relationship can be expressed in two ways. The first way is by listing known variants in the variants section of an EAM pattern. An example for this can be found in Business Application and Organizational Unit Cluster Map (see page 201). The second way is to point to another separate EAM pattern and thereby emphasizing that this pattern is a variant of the referencing pattern.

Name	Description	Used in Literature	Used in the EAM Pattern Catalog
Used by	A smaller pattern is used by a larger pattern	[AIS77]	Business Application and Organizational Unit Relationship *uses* Business Application and Organizational Unit Cluster Map
Refined by	A general pattern is refined by a specific pattern	[MD98]	Business Application and Organizational Unit Cluster Map *refined by* Standards Conformity Exceptions
Variant	A variant pattern refines a more well-known pattern	[Bu96]	Interfaces and Information Flows is a *variant* of Business Object Interface Usage
Variant Uses	A variant of one pattern uses another pattern	[Zi94]	
Similar	A pattern is similar to another pattern	[Zi94]	
Combines	Two patterns combine to solve a single problem	[Bu96, DA98, WV03]	EA Visioning *combines* Business Visioning, IT Visioning, and Target Landscape Definition
Requires	A pattern requires the solution of another pattern	[Dy97]	
Tiling	A pattern uses itself	[Lo97]	
Sequence of Elaboration	A sequence of patterns from the simple to the complex	[SR98, Fo97]	

Table 5.2.: Overview about the secondary relationship types between patterns (based on [No98])

The *variant uses* relationship is similar to the uses relationship but expresses that the usage of another pattern is optional. This means that a pattern can use the results of another pattern but it does not have to. The approach described by Noble [No98] proposes to abstain from this distinction because patterns are ultimately guidelines or rules of thumb, and are meant to be tailored to suit a specific problem and the context of their use. For this reason any part of a pattern may be omitted or varied when the pattern is instantiated [AIS77, Ga95]. The EAM pattern language follows this argumentation and does not use this kind of relationship. Therefore, there is no example given in Table 5.2.
The *similar* relationship is also not used in the EAM Pattern Catalog. Zimmer [Zi94] uses this relationship as a kind of catch-all, if there exists a relationship between two patterns but none of the other relationship types apply. Noble [No98] proposes to use other relationships like variant to clarify the meaning of the relationship. For the same reason this kind of relationship is not used in the EAM Pattern Catalog.
Combines has equivalences in other domains like UML in *aggregation* and *composition* [Hi05] relationships, or related domains like schema theory in *has-a* and *part-of* relationships (see [KS08] and Section 3.2). In [Co96, Ri97, BHS07b] this kind of relationship is further refined to the concept of a *composite pattern*. Riehle [Ri97] defines a composite pattern as follows and uses the Bureaucracy pattern in [Ri97] to exemplify the approach.

> **Composite Pattern:** A composite pattern represents a design theme that keeps recurring in specific contexts. I call it a composite pattern, because it can best be explained as the composition of some other patterns. However, a composite pattern goes beyond a mere composition: It captures the synergy arising from the different roles an object plays in the overall composition structure. As such, composite patterns are more than just the sum of their constituting patterns.

Various other names exist for composite patterns. Riehle [Ri97] provides alternatives like *aggregate patterns*, *combination patterns*, *pattern teams*, or *pattern ensembles*. Coplien [Co96] calls the concept an *umbrella pattern* and Salingaros [Sa00] a *higher-level pattern*. The EAM Pattern Catalog and the rest of this thesis uses the term composite pattern as defined above. Composing patterns to higher level patterns is exemplified in Figure 5.4.
An example for a composite pattern is EA Visioning (see Figure 5.5) presented by Buckl et al. [Bu09e]. EA Visioning comprises three constituent patterns: Business Visioning, IT Visioning, and Target Landscape Definition. These three patterns are documented independently because they are self-contained enough to be valuable to be documented. Nevertheless, if they are used as a part of the composite pattern EA Visioning, their value is higher than just the sum of the constituent patterns. The reason for the added value is that the interaction between the patterns, including additional forces and consequences, etc. is described in the composite pattern and may lead to new insights for the pattern user.

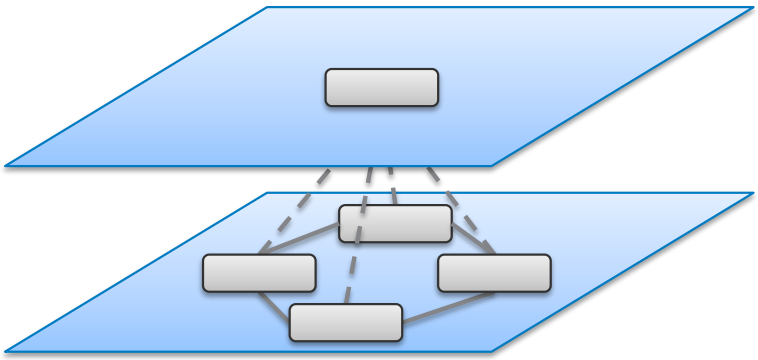

Figure 5.4.: Aggregation of patterns to higher-level patterns (based on [Sa00])

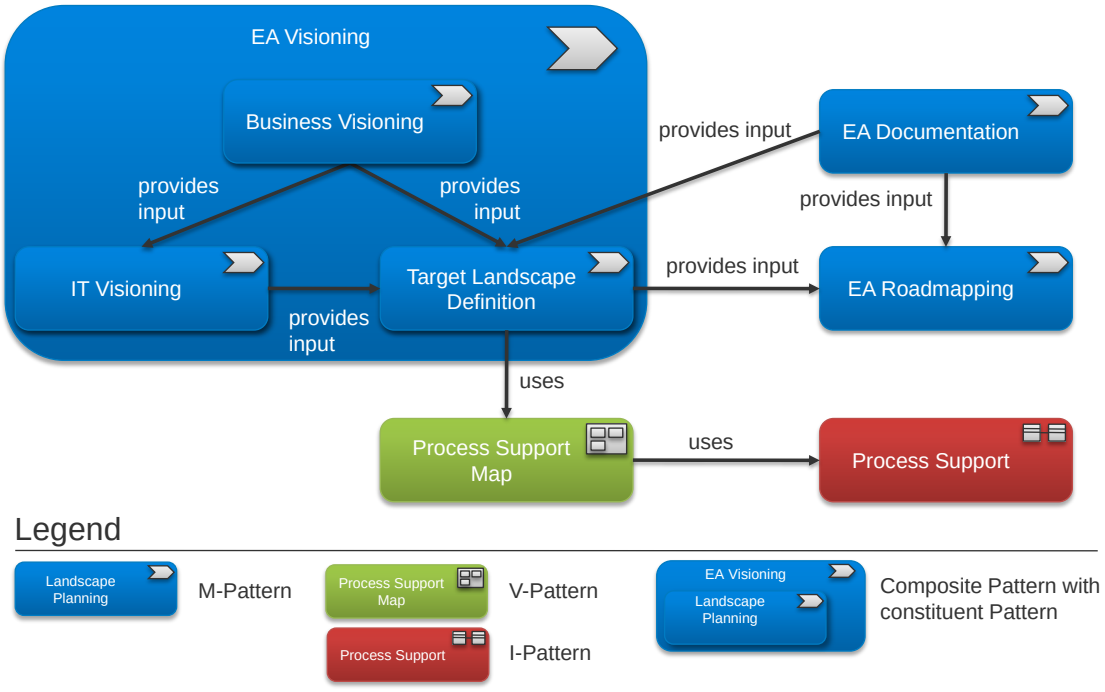

Figure 5.5.: Composite pattern EA Visioning based on [Bu09e]

A composite pattern can be documented in different ways. Buschmann et al. [BHS07b] present the three most used ones.

Document composite pattern on its own, without reference to its constituents patterns:
: This way to document a composite pattern is useful for simple compounds, in which the effort required to document its constituent elements is minimal. In this case the constituents are included as short summaries, but no explicit references are given. A benefit is that the effort for maintaining the relationships is minimized, but this leaves the effort for identifying the elaborated constituents to the pattern user.

Document composite pattern by reference to its constituent patterns: This approach scales better for more complex compounds. A disadvantage is that the knowledge of all patterns is needed by the pattern user.

Document both, composite pattern and its constituent patterns: In this case the composite pattern is documented as a collection of patterns. This requires more effort by the pattern author, but can be utilized more easy by the pattern user because detail is available, if needed.

The *requires* relationship emphasizes that a pattern requires the solution of another pattern. According to Noble [No98] adequately be mapped by the uses relationship and is therefore not used in the EAM Pattern Catalog.
The *tiling* relationship specifies that some patterns can be applied repeatedly to solve a single problem. Like for other relationship types, Noble [No98] proposes to map this to other relationships. In this case it is mapped by introducing additional variants and apply them reflexively. As this relationship type is also not used in the EAM pattern language the example cell is left empty.
Sequence of elaboration is the last relationship type found during the analysis in [No98]. It can be mapped by combining the uses and the refines relationship. Sequence of elaboration is currently not explicitly used, but implicitly when using the combines relationship, e.g. in EA Visioning where the combined patterns are executed in a sequence. Defining pattern sequences for supporting the pattern user in applying the patterns may be a future area of research for the EAM Pattern Catalog.
The different relationship types may e.g. be indicated by different line or arrow styles in pattern maps like Figure 5.10. In addition to these relationship types it is possible to annotate additional information, for example in visualizations detailing the relationships between patterns like in Figure 5.5 where *provides input* is annotated to some relationships.

5.2.2. Abstraction Levels and Hierarchical Relationships

As mentioned before patterns can exist on different levels of abstractions and can therefore exhibit hierarchical relationships. In [AIS77] patterns are ordered by decreasing size. Concerning buildings this goes from Building Complex and Number of Stories to Child Caves and Secret Places. This hierarchical relationship is further detailed

in [Al79] by "Each pattern then, depends both on the smaller patterns it contains, and on the larger patterns within which it is contained".
Coplien [Co96] also emphasizes the importance of abstraction and partitioning: "One of the most important challenges of system design is dealing with complexity. We attack complexity with abstraction. Much of design concerns itself with finding the right abstractions in a system, partitioning the system into mind-size, manageable chunks."

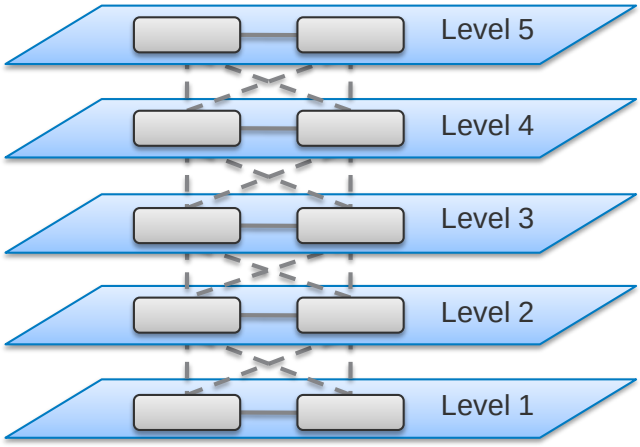

Figure 5.6.: Hierarchical relationship between patterns (based on [Sa00])

Hierarchical relationships between patterns are exemplified in [Sa00] by Figure 5.6. These hierarchical relationships can be of the types described in the previous Section 5.2.1, e.g. uses, refined, combines, etc. A characteristic of a pattern language is that patterns should be connected to both higher and lower levels [Sa00]. If these relationships are missing the pattern language degrades to groups of patterns, which are too far apart in scale to connect effectively. An example for this is given in Figure 5.7.

Figure 5.7.: Hierarchically disconnected patterns (based on [Sa00])

After the description of the different relationship types and the hierarchical character of pattern languages, the following section will go into detail on consistency and connectivity.

5.2.3. Consistency and Connectivity

Relationships between patterns do not solely exist within one pattern language, but there are also external relationships. The internal relationships (see Sections 5.2.1 and 5.2.2) are required for the *consistency* within a pattern language [Sa00], while external relationships are required for the *connectivity* to external pattern languages or sources. The latter one is even more important, because patterns or a group of patterns, which are not connected to other patterns run the risk of getting isolated and will no longer be checked and balanced like the other patterns [Sa00]. This is exemplified in Figure 5.8 where the patterns inside the circle – one pattern language – are internally consistent but fail to connect to external sources.

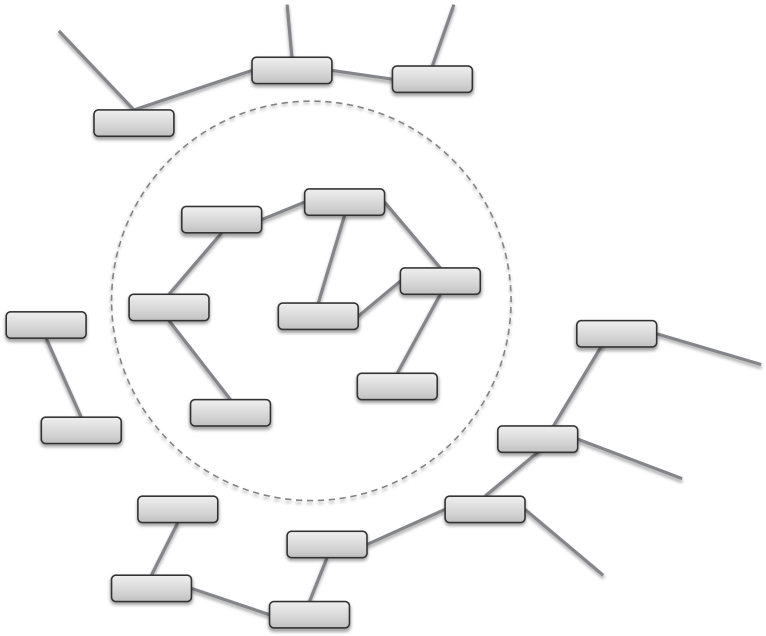

Figure 5.8.: Consistency and connectivity of patterns (based on [Sa00])

Different approaches, like continuous reviews of patterns, help to maintain and increase the connectivity to other pattern languages or other sources but the most promising one is to establish an active community. This community can on the one side identify missing connections to external sources and can on the other side introduce new connections to existing patterns. Identifying missing relationships is also important during the introduction of new patterns (see Section 5.5.1). This can be facilitated by appropriate tool support like the EAM Pattern Catalog Wiki described in Section 5.6. In the EAM Pattern Catalog Wiki a bibliography is included, which fulfills the function of establishing and maintaining connections to external sources.

5.2.4. EAM Pattern Map

The previous sections emphasized the importance of relationships between patterns. To satisfy this demand the internal relationships of the EAM Pattern Catalog are detailed in an EAM pattern map as shown in Figure 5.10. It shows the complete EAM pattern map as of July 1st 2009 and includes all available EAM patterns and EAM anti patterns together with their relationships. Due to size restrictions the relationship types (see Section 5.2.1) are omitted. A legend for the EAM pattern map is given in Figure 5.9.

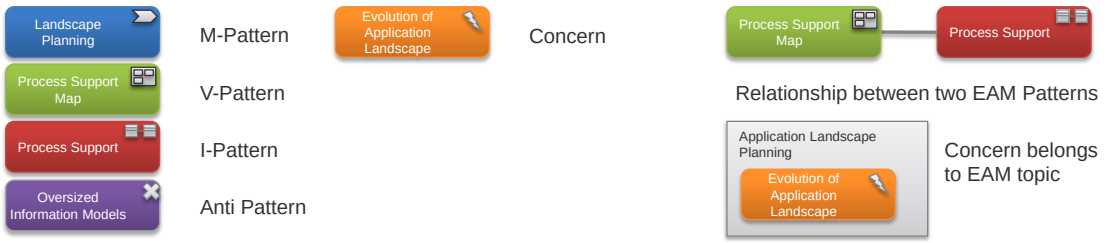

Figure 5.9.: Legend for EAM pattern map

Although, the size restriction limits the readability of the EAM pattern and EAM anti pattern names it can be seen that the patterns are highly interconnected and that they are organized around concerns belonging to EA management topic areas. These topics will be introduced in Section 5.3.1. A zoomable EAM pattern map is available for download in the EAM Pattern Catalog Wiki.
The EAM pattern map may be used for different reasons, with some exemplary ones given in the following. It can be used for identifying patterns related to EAM patterns, which are already in use. These can then be considered for a future extension of the EA management approach currently in use. Another reason for using the EAM pattern map may be to organize the topic EA management. The pattern map can also be used to identify EAM patterns not connected with other EAM patterns. This may be an indication that there is a demand for mining additional patterns or that these unconnected patterns are special for some other reason.

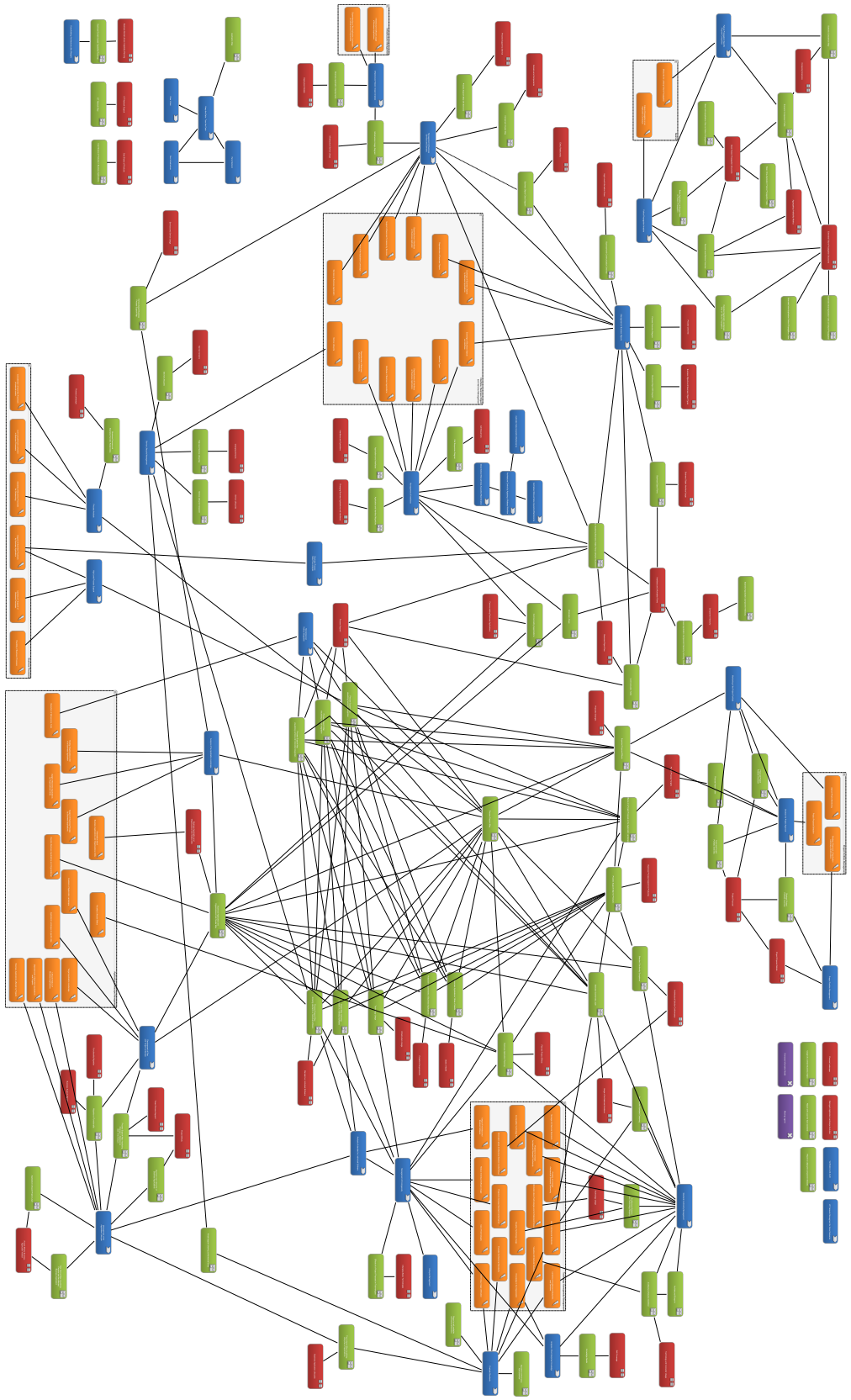

Figure 5.10.: EAM pattern map

5.3. Selection of EAM Patterns

Section 4.2 presented some basic information on how to select EAM patterns using the list of concerns supplementing the EAM patterns and using the EAM pattern map. This section presents some more advanced methods, which may improve the selection of patterns in the future.
The organization of the patterns in the pattern language is tightly connected to the intended EAM pattern selection process and is usually dependent on the intended audience. Therefore, it is desirable to organize EAM patterns in a flexible way, so that every user can easily adopt the organization to his or her usage context. This is supported by the EAM Pattern Catalog Wiki, e.g. by using tags for filtering (see Section 5.6).
Selecting patterns is an important task, when using a collection of patterns, especially if the number of included patterns increases. In [Ga95] Gamma et al. describe this problem as follows. "With more than 20 design patterns in the catalog to choose from, it might be hard to find the one that addresses a particular design problem." In a pattern language like the EAM pattern approach, which includes 164 patterns (as of 2009-09-05) and grows constantly this is also a major issue. Relationships between these patterns also have to be considered to find appropriate solutions, because the solution may consist of a combination of patterns. Patterns of a pattern language can also have relationships to other pattern languages not included in the same collection. These may have a different structure or may be part of a completely different domain of interest. The same problem may occur in cases where different pattern authors contribute to a pattern collection like in the pattern-based approach to EA management.
These are only some problems possibly influencing the selection of patterns. But these already give an indication of the importance of supporting the pattern selection in an appropriate way. Various ways for selecting patterns have been proposed in literature and developed by practitioners, see e.g. [Ga95], [MD01], [Zd07], [Bu08a], etc. This section presents three different approaches and shows how the approaches can be used with the EAM pattern language. A concluding section gives an outlook on additional organization and selection possibilities.

5.3.1. EAM Pattern Selection based on Concerns

The first and most prominent way to select EAM patterns is *based on concerns*. It uses concerns, which are also known as pain points, as a selection criteria. A list of concerns addressed by EAM patterns and EAM anti patterns is included in the EAM Pattern Catalog for easy identification of concerns that could be addressed in an organization.
As mentioned before concerns can be mapped to the problem statement of each EAM pattern and are therefore automatically available and do not have to be additionally assigned to patterns, like e.g. categories. This is one reason why this kind of organization is available since the initial version of the EAM Pattern Catalog. Another reason is that EA management is often executed in a *problem-driven* way, meaning that if an EA problem is identified a search for an adequate solution is started but not that it is tried to anticipate a problem before it occurs.
A list of concerns is one way to organize EAM patterns and to support their selection. If the number of patterns in a pattern language exceeds a number, which is easily manage-

able, it is required to additionally support the selection process. According to Gamma et al. [Ga95] a set of 20 patterns may already require additional guidance for pattern selection. The EAM Pattern Catalog includes 164 EAM patterns (as of 2009-09-05) leading to a similar number of respective concerns requiring additional guidance for selection. As one concern may be addressed by more then one EAM pattern, the number of EAM patterns and concerns is not identical. This requires a secondary organization to be practically usable.
One way to establish such an organization is by using *EA management topic areas*. EA management topic areas were already used in the initial version of the EAM Pattern Catalog [Bu08a] and were only marginally changed up to now. Currently the following EA management topic areas are used [se09a]:

Standardization and Technology Homogeneity Management includes EAM patterns about the management of architectural standards and technologies in an organization, in order to improve homogenization.

Business Processes Support Management is concerned with managing the interaction of business applications, business processes, and related entities relevant to business at a high level of abstraction.

Application Landscape Management targets planning and analyzing the structure and evolution of the application landscape, focusing on current, planned, and target landscapes.

Project Portfolio Management is concerned with managing the portfolio of projects changing the application landscape. For example, technical, financial, and strategic aspects are addressed in selecting projects to be included in the portfolio and in the subsequent monitoring and management of the portfolio.

Infrastructure Management analyzes the technical infrastructure, on which the EA relies, and what impacts this infrastructure can have on the business support provided by business applications.

Interface, Business Object, and Service Management is concerned with analyzing and finding services in the context of service-oriented architectures (SOA). Thereby, the data flows created by communication via services, and the business objects exchanged through service interfaces are important aspects of the analyses.

Metrics includes EAM patterns about defining and applying metrics in the context of EA management.

Other covers every other EAM patterns, which do not belong to any other already described EA management topic area.

EA management topic areas are only one way to organize concerns. Various other criteria could also be used, e.g a mapping to existing EA frameworks, like TOGAF [Op09a] or Zachman [Za87]. Welie and van der Veer [WV03] point in the same direction. "Since it is clear that there are several useful ways to organize patterns, designers should not be forced to choose one particular view".

Due to the decision to use the EAM Pattern Catalog Wiki for the future development of the EAM pattern approach this requirement can adequately be addressed as different organization criteria can be used in parallel e.g. by tagging, etc.
In addition an organization based on a utility rating of EAM Pattern Catalog community members would be imaginable. Currently, no such additional organization criterion is implemented in the EAM Pattern Catalog.
Selecting EAM patterns based on concerns in the EAM Pattern Catalog Wiki would require a four step process:

Select categorization schema in order to organize concerns in a way that best fits ones demands, e.g. based on EA management topic areas.

Select categories uses a categorization separating the concerns in different domains, e.g. like the EA management topic areas previously described, to narrow down the number of concerns that come into consideration.

Select concerns in categories and thereby select EAM patterns addressing the existing concerns. If a concern can be addressed by multiple EAM patterns it is required to look at context, forces, and solution in order to identify the right pattern. Comparing patterns this way may require high effort. Using the summary to compare the patterns may reduce the required effort, but may result in a suboptimal selection.

Select additional EAM patterns is required, because some EAM patterns use additional patterns, which may be belong to a different EA management topic area. Nevertheless, they are required to compose an integrated pattern-based approach to EA management.

A related approach to select EAM patterns based on concerns is to assign stakeholders to identified concerns and to use stakeholders existing or planned to be introduced to select patterns. Such a stakeholder-based approach is proposed in [Be09].
A concern- or stakeholder-based approach is best used for the following three usage scenarios: Establish an EAM approach (see Section 4.2.1), inspire and assess an existing EAM approach (see Section 4.2.2), and specify goals and requirements for EAM (see Section 4.2.3).

5.3.2. EAM Pattern Selection based on Maturity Models

A special variant of selecting EAM patterns based on concerns is to use *maturity models* to organize concerns. Various domains use the concept of maturity levels to organize topics within the domains in order to help organizations to improve their approaches on industry and government best practices. Prominent examples are CMMI for Development [SE06] or CMMI for Services [SE09b] both from the Software Engineering Institute at Carnegie Mellon University. Similar approaches have been developed for EAs and their management, with five of them introduced in the following.
The *NASCIO Enterprise Architecture Maturity Model* [Na03] has been developed by the *National Association of State Chief Information Officers* (NASCIO), which includes the state chief information officers from the 50 states, six U.S. territories and the district of

Columbia, as a tool to objectively review the status of the EA approaches for the state and local governments. The maturity model defines six different maturity levels:

EA LEVEL 0 - No program: At this maturity level no architectural framework is in use. Solutions are developed and implemented with no standards or best practices considered. Individual contributors are the drivers and possess the knowledge.

EA LEVEL 1 - Informal program: A base architectural framework and standards have been defined and are typically performed informally. There is a consensus in the organization that it is reasonable to use such an approach, but its usage is not controlled or tracked. In organizations where an EA framework is used they are still dependent to the knowledge of individuals.

EA LEVEL 2 - Repeatable program: A base architectural framework and standards are in use and are being tracked and verified. Specified processes are repeatable, reusable templates are developed, and metrics are defined and used to track performance. There is consensus that products and components should conform to standards and requirements.

EA LEVEL 3 - Well-defined program: Approved standard and/or customized template versions are in use. Documentation of processes across the organization is available. Defined performance metrics are tracked and monitored in relationship to other general practices and process areas.

EA LEVEL 4 - Managed program: Performance metrics are collected, analyzed and acted upon. The metrics are used to predict performance and provide better understanding of the processes and capabilities.

EA LEVEL 5 - Continuously improving vital program: The defined processes are mature targets have been set for effectiveness and efficiency based on business and IT goals. Processes are refined and improved using the understanding what impact the changes will have on the processes.

For each maturity level detailed statements for the following eight EA categories are given spanning up a matrix including information on what to expect of an organization for a combination of maturity level and category.

Administration: Governance Roles & Responsibilities

Planning: EA program road map and implementation plan

Framework: Processes and templates used for EA

Blueprint: Collection of the actual standards and specifications

Communication: Education and distribution of EA and blueprint detail

Compliance: Adherence to published standards, processes and other EA elements, and the processes to document and track variances from those standards

Integration: Touch-points of management processes to the EA

Involvement: Support of the EA program throughout the organization

The *Extended Enterprise Architecture Maturity Model (E2AMM) v2.0* [IF04] has been developed by the *Institute for Enterprise Architecture Developments* (IFEAD). Like the previously describe maturity model it also defines six levels for EA and procedural improvements:

Level 0: No extended EA

Level 1: Initial

Level 2: Under development

Level 3: Defined

Level 4: Managed

Level 5: Optimized

These levels are detailed concerning what an organization should do in the appropriate level, using eleven different topic categories:

- Business & technology strategy alignment
- Extended enterprise involvement
- Executive-management involvement
- Business units involvement
- Extended EA program office
- Extended EA developments
- Extended EA results
- Strategic governance
- Enterprise program management
- Holistic extended EA
- Enterprise budget & procurement strategy

Like in the previously described maturity model the maturity levels together with topic categories span up a matrix giving an indication what can be expected by a company, which has reached a certain maturity level. According to the Institute for Enterprise Architecture Developments [IF04] the goal of the maturity model is that in organizations, which have reached level 5 enterprise architecture should become an extended-enterprise concept and prescribes the infrastructure for extended-enterprise businesses and provide the conditions and structures.
Similar approaches, using a matrix for maturity assessment, are described e.g. in *A Framework for Assessing and Improving Enterprise Architecture Management* [GA03] and *Maturity Assessment for the Enterprise Architecture (EA) Function* [Ba08].

A different approach is used in *Assessing the Maturity of Information Architectures for Complex Dynamic Enterprise Systems* [My07]. In this thesis the maturity of the *Enterprise System Architecture* (ESA), which is comparable to an EA, is taken as the linear combination of four dimensional maturities:
$ESA_{Maturity} = \alpha_1\ Process_{Maturity} + \alpha_2\ IT_{Maturity} + \alpha_3\ IS_{Maturity} + \alpha_4\ EA_{Maturity}$
Whereas $Process_{Maturity}$ measures the maturity of enterprise processes, $IT_{Maturity}$ measures the maturity of technology-structure and its deployment, $IS_{Maturity}$ measures the maturity of technology-based support of enterprise processes, and $A_{Maturity}$ measures the maturity of enterprise architecture and its deployment.
In contrast Riege and Aier [RA09a] do not present a maturity model but three different EA application realization scenarios – EA engineers, IT architects, EA initiators – which have been identified in an empirical analysis. Riege and Aier [RA09a] define the different scenarios as follows.

EA engineers understand EA as a valuable instrument to develop and thus transform EA in its holistic understanding. They can also rely on a progressive perception of EA within the business and management units. EA engineers in its current state have an intermediate maturity regarding the employment and monitoring of EA data and services.

IT architects are well anchored in the IT domain. This limited architectural understanding is an obstacle in order to really leverage the value of available IT understanding, models and methods.

EA initiators put emphasis on transparency as the necessary precondition to realize benefits from EA application. They are in particular interested in implementing relevant applications to demonstrate these benefits. This also explains the need for more sophisticated analysis techniques which EA initiators lack of. EA initiators often use a tool driven or model driven EA approach.

All these approaches can be used for organizing and selecting concerns and respective EAM patterns based on maturity models. This only requires a mapping of the different maturity levels or EA applications realization scenarios to a set of concerns. A simple way to achieve this is to use tagging, which can be done using the EAM Pattern Catalog Wiki (see Section 5.6).
It is a four step process to select EAM patterns based on maturity models:

Select maturity model, which should be used or which already is in use in the company.

Determine maturity level the company has reached in EA management to narrow down the number of potential concerns.

Select concerns in maturity level and thereby select EAM patterns addressing the existing concerns. If a concern can be addressed by multiple EAM patterns it is required to identify the EAM pattern, which best fits the company context.

Select additional EAM patterns is equal to the same step described in previous approach.

The maturity model-based approach is best used when the EAM Pattern Catalog is used for inspiring and assessing an existing EAM approach (see Section 4.2.2) if a company is looking for what has to be done to reach the next maturity level. And for specifying goals and requirements for EA management (see Section 4.2.3) if a company wants to reach a new maturity level and has to define goals to achieve this.

5.3.3. EAM Pattern Selection based on Pattern Language Grammars and Design Space Analysis

The previous sections introduced different ways on how to select concerns and respective EAM patterns using a certain categorization. This section proposes a more formal way to select patterns. The approach has been developed by Zdun [Zd07] and uses *pattern language grammars and design space analysis* to find the right patterns. [Zd07] states that well-elaborated pattern descriptions contain all relevant information for an informal design decision. This includes:

Functional requirements: A pattern describes, which functional requirements it can potentially fulfill, as the pattern described a proven practice solution to a problem in a context.

Quality goals: Each pattern typically includes two sections, which describe potential influences on quality goals: The problem section with the included forces and the consequence section. Forces are typically formulated as conflicting forces, which influence the quality goals of the patterns solution. Consequences should correspond to the forces of a pattern and should give indications on how to balance the forces and to match the quality goals.

Pattern variants: Variants are described in the solution section or in a special variants section of a pattern (see section 3.4). Latter is implemented in EAM patterns and EAM anti patterns (cf. 4.1.1).

Related patterns: Relationships to other patterns are typically described in a separate related pattern section, which is called *see also* in the EAM pattern approach. These relationships indicate that e.g. another pattern should be considered when using a pattern.

Nevertheless, this information is not sufficient for a systematic design decision. To address this problem, the approach by Zdun formalizes pattern relationships in a grammar and annotates the grammar with effects on quality goals.
In [Zd07] quality goals by Bass et al. [BCK03] and ISO 9126 International Standard for the Evaluation of Software [IS91] are used but it is explicitly mentioned that other quality attributes can also be used, as long as they are well-defined in the context of a pattern language so that all readers share a common understanding of what is meant by each term.
Currently, there is no standardized catalog of quality attributes for EA management, but there are some publications proposing quality attributes for EAs. Lankes and Schweda [LS08] propose *failure propagation*, and [La08a] proposes *cost advantages in*

operation, *installability*, *flexibility*, *functionality*, *maintainability*, *operational risk*, *performance*, *scalability*, and *testability* as quality attributes for applications landscapes, which can be considered to be part of the enterprise architecture. Morris [Mo07] introduces *manageability* as a quality attribute for an EA. Närman et al. [NJN07] use ISO 9126 as a basis and proposes *interoperability*, *information security*, *maintainability*, *performance*, etc. as attributes.
After the quality attributes have been identified within the patterns, the grammar of the pattern language is documented and annotated with the effects on the quality goals. A pattern map like in Figure 5.5 can be used as a basis showing the topology of a pattern language. In the next step such a pattern map has to be enriched by forces in order to visualize possible steps in addressing problems with patterns. As mentioned before forces influence the solution. In addition, they can often be regarded to be quality attributes of the solution. These quality attributes are then added to the relationships in the pattern map.
Zdun [Zd07] proposes the following notation for showing the influence on a quality goal:

- ++, a very positive influence on the quality goal can be expected
- +, a positive influence on the quality goal can be expected
- o, no influence on the quality goal can be expected
- -, a negative influence on the quality goal can be expected
- - -, a very negative influence on the quality goal can be expected

Figure 5.11 shows an exemplary visualization of a pattern map enriched by quality attributes, which is called a *pattern sequence diagram*. The visualization additionally shows if the utilization of one pattern by another one is *optional* or *required* and if patterns are variants.

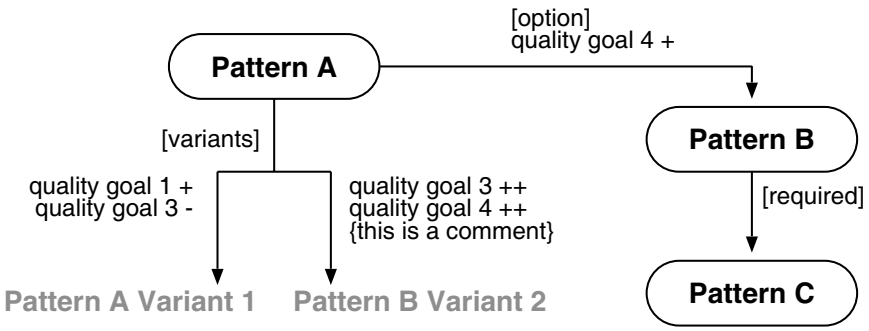

Figure 5.11.: Example of a pattern sequence diagram [Zd07]

As a result it is possible to derive pattern sequences, which drastically reduce the possible pattern combinations. A pattern sequence is an individual path through the web of patterns [Zd07]. It has a positive effect that is repeated and can be recounted [BHS07b]. A pattern sequence diagram shows the effects that using patterns has on the quality attributes.

The pattern sequence diagram, which is a graphical representation of the *pattern language grammar* is sufficient for simple decision. In cases where more complex decisions concerning pattern selection have to be made, a formal grammar has to be derived from pattern sequence diagram, which is then used for a *question–option–criteria* (QOC) [Ma91, MM95] design space analysis, resulting in a detailed decision map.
Zdun [Zd07] defines the grammar G of a language L as the tuple: $G = (V, T, P, S)$, using the following definitions:

- V is a set of syntactic variables (non-terminals).
- T is a set of terminal symbols, which represent a sub-set of the alphabet A of the language L. The terminal symbols in a pattern language grammar are patterns (and pattern variants).
- P is a set of production rules $X \rightarrow Y$, meaning that X can be transformed into Y. Both X and Y can contain syntactic variables and terminal symbols. The production rules are applied until all syntactic variables are transformed into terminal symbols. The result of this transformation is a pattern sequence that conforms to the grammar of the pattern language.
- S is a start symbol, $S \in V$, which starts the application of the production rules.

The language L is the set of all words w that can be derived from $G = (V, T, P, S)$. In a pattern language, the words w are the sequences that can be derived from the pattern language

$$L(G) = \left\{ w \; in \; T^* \;\middle|\; S \overset{*}{\underset{G}{\Longrightarrow}} w \right\}$$

An example on how to derive the grammar from a pattern sequence diagram is given in [Zd07]. The derived grammar can the be used to perform a patterns-based QOC design space analysis. A QOC design space analysis can be used to analyze and decompose patterns to create an organized space of patterns.
Design space analysis is an approach developed and used in human-computer interaction (HCI) [Ma91, MM95] and has also be applied to other contexts like planning [Po98]. The approach uses a semiformal notation based on *questions*, *options*, and *criteria* to represent the design space of an artifact. Question identify key issues to be considered in design decisions, options provide possible answers to the questions and criteria can be used to asses and compare options. QOC can be used to visualize alternatives for design decisions and related design considerations and to capture design rationale.

Zdun [Zd07] mentions one difference in using design space analysis in HCI and in pattern context. In HCI the design space is applied to a concrete design compared to a more abstract design space around patterns, which are themselves an abstraction of concrete designs. To be able to use the design space approach, a mapping of questions, options, and criteria to concepts of patterns has to be established. Such a mapping is given in the following:

Question: One question has to be created for every set of alternative patterns.

Option: Each pattern of the set of alternative patterns is represented by one option.

Criteria: Forces and consequences (quality goals) of each pattern under consideration are mapped to one criteria. There may be cases where one force or consequence has to be mapped to a number of criteria.

Figure 5.13 is based on an activity diagram presented in [Zd07] showing the activities required for mapping patterns to a design space. For reducing complexity of the visualization feedback loops are not included, but have to be considered between all activities, e.g. if the identification of a pattern relationship reveals another alternative in the core patterns.
Up to now variant and alternative relationships have been covered. Required relationships can be simply included in the approach, because there is no extra design decision required. A required pattern has to be included. Relationships leading to optional patterns can be mapped by adding another option and including another question in the design space, which is asking if the option should be taken.
The result of the execution of the activities described in Figure 5.13 is a complete, hierarchical decomposition of a design problem using patterns and their variants as options (solutions). An exemplary result of such a mapping process is shown in Figure 5.12.

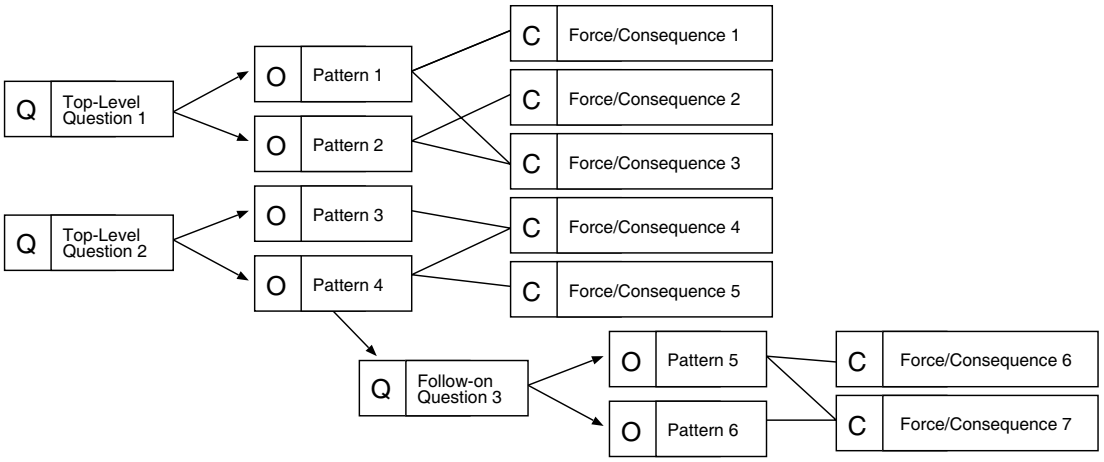

Figure 5.12.: Example for design space visualization [Zd07]

According to Zdun [Zd07] the approach was applied in multiple industrial case studies and research projects. It showed that the approach could be used in an iterative design

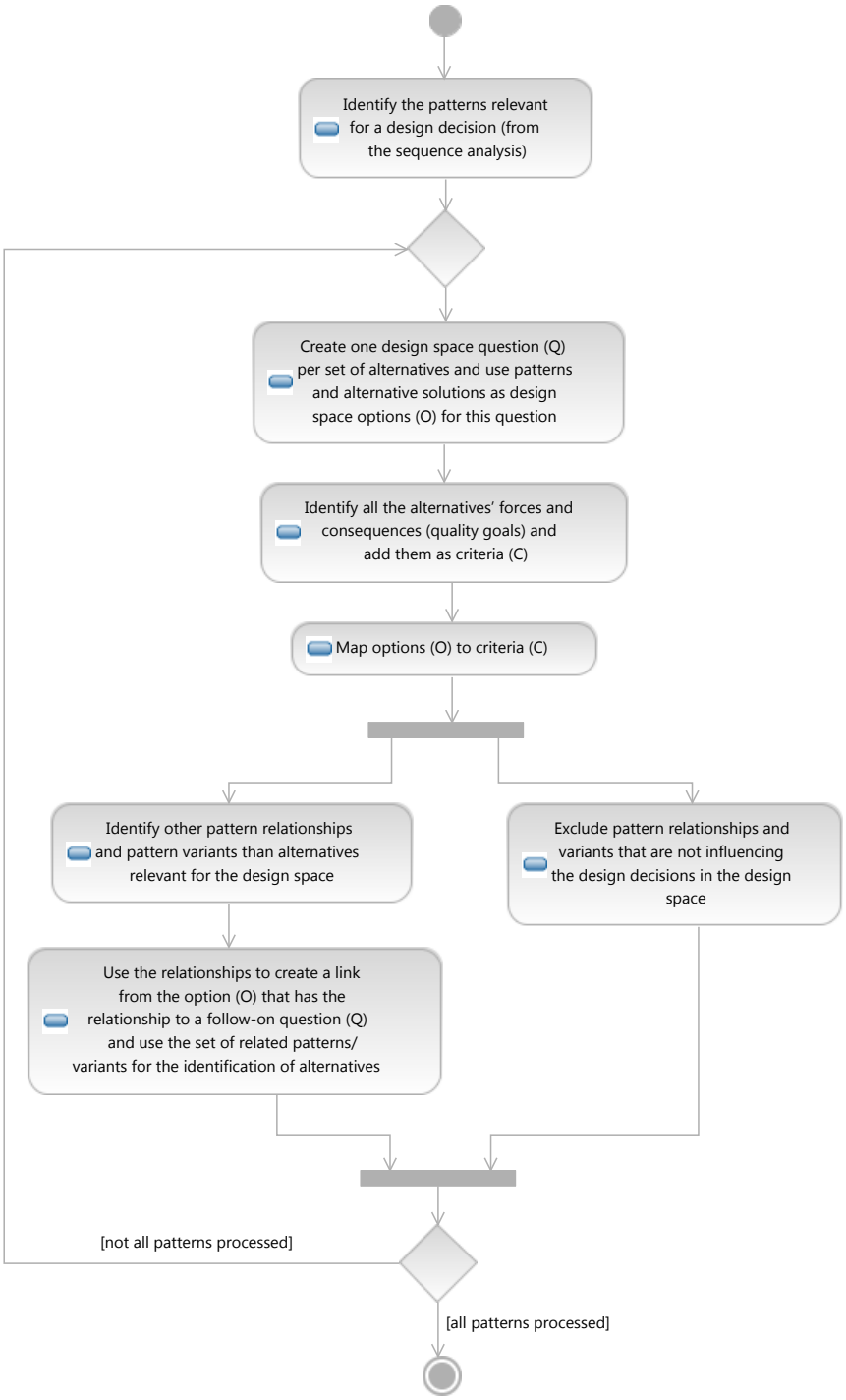

Figure 5.13.: Activities for mapping patterns to a design space (based on [Zd07])

process and that the approach is useful in communicating complex pattern material and design rational to both, technical and non-technical project members. Additionally the approach can be used to organize patterns, reduce the complexity of design decisions, and reduce design and development times. To complement the case studies and the research project a validation of available support for non-experts was executed. The experiment showed that the approach can significantly improve a non-expert's understanding of incomplete pattern material from multiple sources.
The approach of using pattern language grammar and design space analysis was developed for supporting decisions about the selection of design patterns but it can also be applied to EAM patterns. The approach is currently not implemented in the EAM pattern language but can be integrated in the future. This may result in an improved pattern selection support and may also be used as another way to organize patterns to improve understanding of the pattern language by non-experts. An additional benefit of such an extension of the EAM pattern approach would be that documenting design rational for selecting patterns is automatically supported and has not to be supplementary documented.
A disadvantage of the approach is the required effort to create a pattern sequence diagram, to derive the grammar, and in cases with more difficult design decisions to additionally use a design space analysis for choosing between alternative options. In cases where documenting the design rationale is required the approach may be worthwhile the required effort.

5.3.4. Outlook on EAM Pattern Selection

The previous sections presented organization and selection approaches for EAM patterns, which are currently available in the EAM Pattern Catalog or which are planned to be introduced. Certainly, there are various other approaches available. A few of them are presented in this section as an outlook for the future development of the EAM Pattern Catalog. Buschmann et al. [BHS07b] emphasize that it is important to have a method of organization for larger pattern repositories or collections. They propose different ways of organization, which are shortly described in the following.
Ad hoc organization is the simplest way to organize patterns. In this case patterns may be sorted according to their names or by the time of their development. This typically requires a kind of index including the name of the pattern and a short summary to find the right patterns. The Pattern Almanac [Ri00] provides a typical example for such an organization.
Organization by level takes into account that patterns are on different levels of abstraction, granularity, or scale. These different levels are related to the hierarchical relationships described in Section 5.2.2. An example for such an organization can be found in [Bu96]. It differentiates between the most coarse-grained level *architectural patterns*, the medium-grained *design patterns*, and the fined-grained level *idioms*.
Organization by domain is related to the context of the patterns. According to Buschman et al. [BHS07b] a domain defines some or all the context in which a pattern is applied. A domain can be considered a projection of a system or its development with respect to specific criteria like technology, process, business, etc. Examples for such an organization can be found in the previous sections. Quite similar but narrower is the *organization*

by partition. In the context of software architecture this organization may be used to classify patterns with respect to the part of architecture in which they are seen to apply. An analogy in EA management may be different topics, which are relevant in EA management, like the ones presented in Section 5.3.1.
Organization by intent uses the intent of groups of patterns, whereby intent identifies architectural characteristics, common goals, responsibilities, etc. An example for such an organization can be found in [Sc00a] grouping patterns based on four groups: *Service access and configuration*, *event handling*, *synchronization*, and *concurrency*. Another example is [Ga95], which groups patterns according to their purpose to be *creational*, *structural*, or *behavioral*.
Buschmann et al. [BHS07b] describe two more ways to organize patterns, *problem frames* and *pattern semiotics*. Jackson [Ja95] describes a problem frame as "a generalization of a class of problems. If there are many other problems that fit the same frame as the problem your're dealing with, then you can hope to apply the method and techniques that worked for those other problems to the problems you're trying to solve right now." The American Heritage Dictionary of the English Language [Ed04] defines semiotics as "the theory and study of signs and symbols, especially as elements of language or other systems of communication, and comprising semantics, syntactics, and pragmatics". Pattern semiotics tries to apply this concept to organize patterns by considering a pattern and a pattern description to be a sign. According to Buschmann et al. [BHS07b] the benefit of using semiotics is that missing categories and connections are more obvious. Pattern semiotics also emphasizes on potential misunderstandings by making this possibility explicit.
The previously described approaches originate from the pattern community, but it is also possible to use other organization criteria developed in the context of EA management. Schekkerman [Sc04] for example uses four different sets of viewpoints as an organization criteria: *Economic*, *legal*, *ethical*, and *discretionary*. EAM patterns can be assigned to one of these set as an organization criteria.

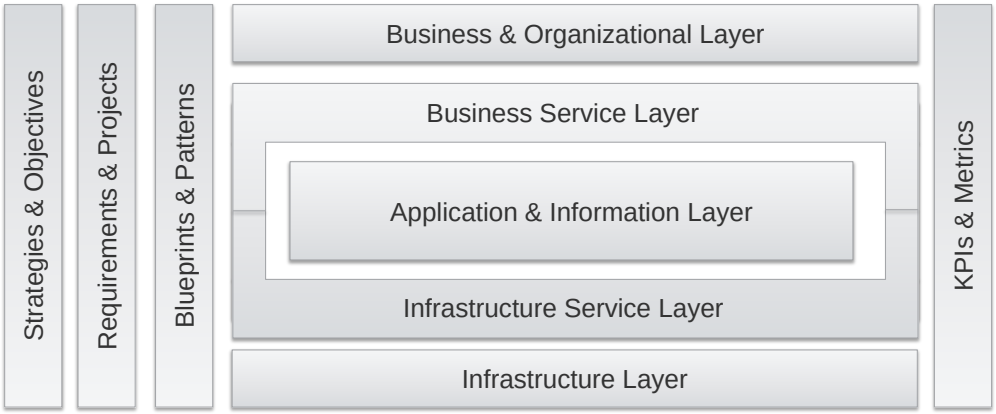

Figure 5.14.: Structure of an information model [Wi07a, Ma08a]

In contrast the next approaches try to divide EA management into different areas of interest. Wittenburg [Wi07a] and Matthes et al. [Ma08a] propose a structure for an

information model (see Figure 5.14). This structure is made up of layers and cross functions, which can be used to organize EAM patterns.
Zachman (see [Za97] and Section 7.1.1) uses a matrix to organize the EA. The Zachman framework is well-known across companies and the EA management community (see Section 7.1.1). Although, it is not widely used in practice it may still be reasonable to organize EAM patterns in a way that matches the matrix proposed in the framework.
The *Enterprise Architecture Desk Reference* [ME02] uses four different key components to subdivide EA (see Figure 5.15): *Enterprise Business Architecture* (EBA), *Enterprise Information Architecture* (EIA), *Enterprise Application Portfolio* (EAP), and *Enterprisewide Technical Architecture* (EWTA). These key components can be found in many companies using EA management to manage their IT. Those companies could therefore be well supported by organizing EAM patterns according to this schema.

Figure 5.15.: Four key components of the Enterprise Architecture Desk Reference [ME02]

As a last example for organizing and selecting EAM patterns another more formal one is presented. The approach was developed by McPhail and Deugo in [MD01]. They proposes to use the **EL**imination **E**t **C**hoix **T**raduisant la **RE**alité II (ELECTRE II) method to select patterns. This approach was developed for multi criteria decision analysis in cases with limited amount of user input in operations research and can be used for ranking alternatives. An advantage of the approach is that it can be extended for specific decision problems, like e.g. for selecting patterns. ELECTRE is available in different variants, which are e.g. described in [Fü05], [Ro68], [Ro91], [FGE05], or [MSE06], which all address specific problems in multi criteria decision analysis.
In the context of patterns the number of criteria used for selecting one or another can be extensive. This has a negative impact on the reliability of the result of a decision.
Therefore, McPhail and Deugo [MD01] try to establish a *hierarchy of criteria* in order to categorize and generalize many criteria. The actor selecting patterns typically is the pattern user, trying to enhance his total satisfaction on quality factors of his design. In the context of the pattern decision problem, the criteria are analogous to the forces that have been organized into a hierarchy, as shown in Figure 5.16, rooted on total satisfaction. Using quality factors is similarly done in the approach by Zdun (see Section 5.3.3). Applying this idea to EAM patterns, such forces can e.g. be flexibility, redundancy, etc. Such forces are the elements of the hierarchy of criteria.
Each node in the force hierarchy is assigned a weight, which is finally used for ranking patterns according to their ability to achieve total satisfaction.

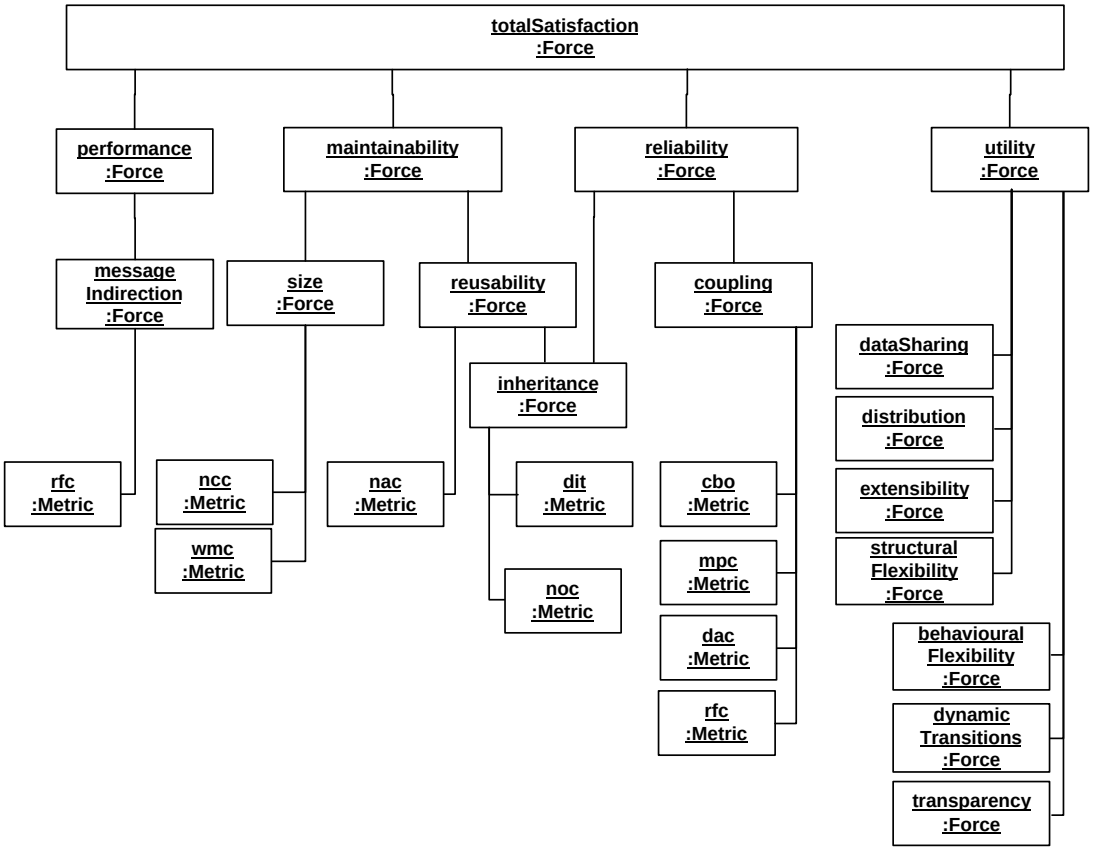

Figure 5.16.: Exemplary pattern force hierarchy [MD01]

McPhail and Deugo[MD01] reports on positive results applying the ELECTRE II approach with the hierarchy of criteria extension to deciding on selecting patterns. The main issue is the creation of the force hierarchy, which has to undergo further analysis and development.
The same is true in the EAM pattern approach. The patterns included in the EAM Pattern Catalog are continuously refined and enriched by forces but the creation of a force hierarchy has not been considered so far. Although this may be helpful for selecting patterns and to obtain an in-depth understanding of the relationships between forces in EA management. This may even lead to a better understanding of EA management itself. A hierarchy of forces for EA management may be another possibility for future extensions of the EAM Pattern Catalog.

5.4. Integration of EAM Patterns

Using a pattern language it is typically not only selecting one pattern to solve a problem but to select a group of interrelated patterns, which then have to be integrated in order to form a solution for the problems at hand. This section shows ways to integrate selected M-Patterns (see Section 5.4.1) and I-Patterns (see Section 5.4.2).

In the currently available EAM pattern approach V-Patterns cannot be integrated. V-Patterns document one or more viewpoints consisting of a base map [Er06] and associated layers and have to be self-contained to fulfill one of the requirements for a pattern. This requirement prohibits that a V-Pattern only documents a layer for a certain base map. This would allow to integrate different V-Pattern by integrating different layers. Nevertheless different V-Patterns can be integrated by looking at the I-Patterns underlying the V-Patterns, because V-Patterns visualize information stored according to one or more I-Patterns. Additionally, they can be used in combination to address certain problems, e.g. in the context of an M-Pattern.

5.4.1. Integration of M-Patterns

M-Patterns can be integrated in different ways. The simplest and the only one currently implemented in the EAM Pattern Catalog is to execute the process steps documented in one M-Pattern one after another and then continue with the process steps of the next M-Pattern. Finding patterns which can be combined in such a way is the difficult part in this approach. In some cases, see e.g. [Bu09e], a composite pattern showing how and in which order to integrate EAM patterns may be available to solve the problem. In other cases manual integration by the pattern user is required. The benefit of this approach is its simplicity, which does not require additional formal extensions.
Besides this, more formal ways to integrate M-Patterns or methods exist. One of them has been described by Kühn [Kü04]. Kühn resolves the problem of integrating different methods by documenting a metamodel for every method fragment. The benefit is that this reduces the integration of methods to the integration of metamodels. As integration of metamodels or in the case of I-Patterns information models is also important in the usage of the EAM Pattern Catalog. This is described in more detail in Section 5.4.2.
Saeki [Sa98] describes a metamodel for method integration. Although this article focuses on integrating software developing methods a similar approach may be feasible for M-Patterns. The core metamodel consists of these concepts: State, event, process (function), data, object, association (relationship among objects). They are used for mapping concepts of the source methods to the common metamodel. As this core metamodel does not cover all concepts of the source methods, it additionally has to be extended by method-specific parts. To prove the applicability of the approach, [Sa98] includes a exemplary integration of object diagrams, data flow diagrams and state transition diagrams in a CASE tool.
Another way to support the integration of M-Patterns would be to complement them with additional information on which input, e.g. previously created views based on V-Patterns or information according to one or more I-Patterns, is required to execute the processes documented in the M-Pattern. This kind of *interface description* could then be used to check whether two M-Patterns can be integrated. Such an extension is currently not available for the M-Patterns included in the EAM Pattern Catalog, but it is planned to include it in the future.
Besides the approaches presented in this section, many more ways to integrate methods are available, see e.g. [Pa97], [Kr93], etc. Those approaches may be evaluated for their applicability in the future development of the EAM pattern approach.

5.4.2. Integration of I-Patterns

The EAM pattern catalog includes various information model fragments, which need to be integrated to one information model in order to be implemented, e.g. in an EA management tool. Many different approaches have been proposed by academia and practice to fulfill this task for example in software engineering or distributed databases. For example Batini et al. [BLN86] introduced a four step methodology for schema integration.

Preintegration analyzes the schemas before the integration starts, to find out where to start the integration, or the number of schemas to be integrated at once, etc.

Comparison of the schemas to find correspondences between the schemas and to detect possible conflicts.

Conforming the schemas in cases where conflicts have been detected these conflicts are resolved.

Merging and restructuring integrates the different schemas after the preliminary steps have been completed. Thereby, four types of semantic relationships are considered: Identical, equivalent, compatible, and incompatible. While integrating the schemas the following qualitative criteria have to be considered: Completeness and correctness, minimality, and understandability. These criteria are similar to the *Guidelines of Modeling* (GoM) introduced by Becker et al. [BRS95]. Both sets of criteria define required and additional principles to foster the development of high quality models.

Another example for schema integration can be found in [Ma05], where an approach for schema integration based on uncertain semantic mappings is introduced. The goal is to support semi-automatic integration of schemas, if all semantic mappings are known. Such an semi-automatic support would also be beneficial for supporting users of I-Patterns. Currently this kind of tool support is not available, but is planned for the future development of the EAM Pattern Catalog (see Section 8.2).
Alter and Goecken [AG09] propose a way to integrate conceptual metamodels of IT governance reference models pursuing a four step approach based on ideas of [Ma05, SPD92, SP94]. These steps are similar to the ones in [BLN86] described earlier, although the domain is different. This way of integration requires that metamodels for reference models like CMMI [SE06], ITIL [CW07], etc. are available or are reverse engineered before the following four steps can be executed.

Preintegration defines which model elements will be consolidated in which order.

Comparison and conflict identification compares the schemas, whereby relationships between concepts of the reference models are analyzed and documented. A special focus in this step is on conflicts and inconsistencies, which have to be resolved in the next step.

Adaptation and overlaying tries to integrate the metamodels by mixing and overlaying. E.g. if similar concepts can be found these can be integrated to one concept in the resulting metamodel. Conflicts or inconsistencies have to be resolved in this step to be able to integrate the remaining parts of the reference models.

Restructuring and optimization uses the resulting metamodel and e.g. tries to remove redundancies, etc. to comply with the GoM criteria [BRS95].

A more recent approach by Lagerström et al. [La08b] is called *probabilistic metamodel merging*. It uses Bayesian networks [Je01] to assess the probability that a merge of two elements is semantically possible. According to Lagerström et al. [La08b] two concerns have to be considered. Firstly, if two classes in the source models represent the same concept. Secondly, when two classes have been merged, do any of the attributes of the merged class represent the same concept. To be able to address those two concerns, the two Bayesian networks shown in Figure 5.17 are constructed by Lagerström et al. More details about the scales of the different nodes can be found in [La08b].

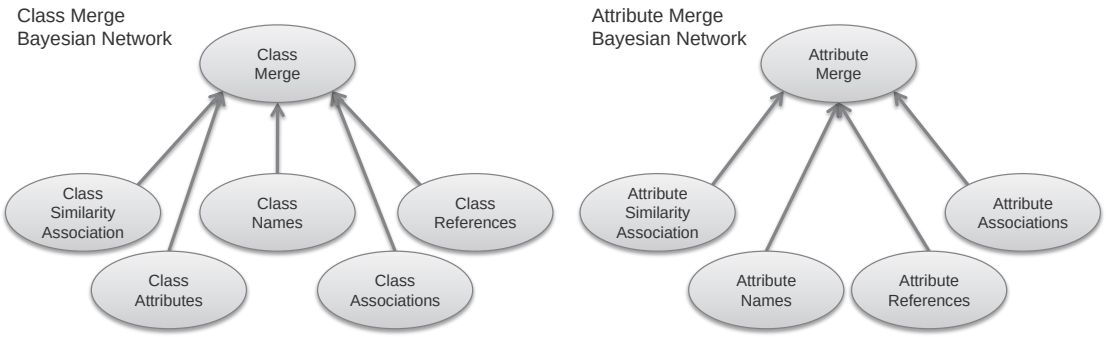

Figure 5.17.: Bayesian networks representing class and attribute merge [La08b]

Figure 5.18 shows an example for a Bayesian network with exemplary values, analyzing a class merge. If the probability for a class merge exceedes a certain threshold two classes can be merged.

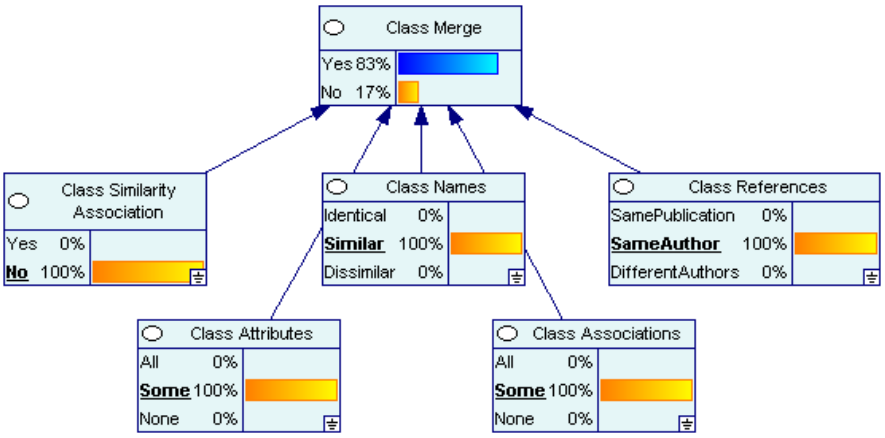

Figure 5.18.: Exemplary Bayesian network for class merge [La08b]

Afterwards a similar analysis has to be performed for the attributes of the merged class. This procedure is iterated until no classes receive probabilities above the defined threshold. The result is an integrated metamodel.

Kühn in [Kü04] describes a similar approach to Alter and Goecken [AG09] using a metamodel based approach for integration. The difference is that Kühn goes into more details and developed a pattern system, which can also be applied to information model fragments, documented in I-Patterns. The following sections present those patterns of the pattern system that can be used for integrating I-Patterns.

5.4.2.1. Integration by Concept Mapping

The first and most intuitive way to integrate two I-Patterns is by mapping two concepts in the information model fragments. If UML is used for documenting the I-Patterns, Buckl et al. [Bu07a] propose to identify one or more identical classes within both patterns. These classes can then be used as points of integration.
Kühn [Kü04] describes a similar approach, which he calls Concept-Mapping-Pattern. This approach is shown in Figure 5.19 in form of a conceptual class diagram.

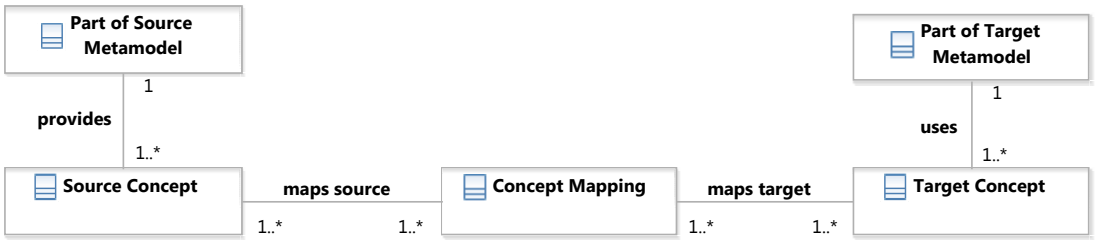

Figure 5.19.: Conceptual class diagram for integration by concept mapping [Kü04]

Kühn [Kü04] uses the term metamodel, which is analogous to information model in this case. Both terms are therefore used in this and the following sections presenting the integration patterns.
In order to integrate one or more *source concepts* from one information model with one or more *target concepts* from another model, a *concept mapping* is defined. Hereby, a source or target concept can be used in one or more concept mappings.
Exemplifying this approach the two I-Patterns Process Support and Interfaces and Information Flows can be integrated by the concept `BusinessApplication`, which can be found in both patterns.

5.4.2.2. Integration by Reference

The second way of integrating I-Patterns is utilizing the Reference-Pattern. In this case a reference between one concepts of the *source information model* and one concept of the *target information model* is introduced.

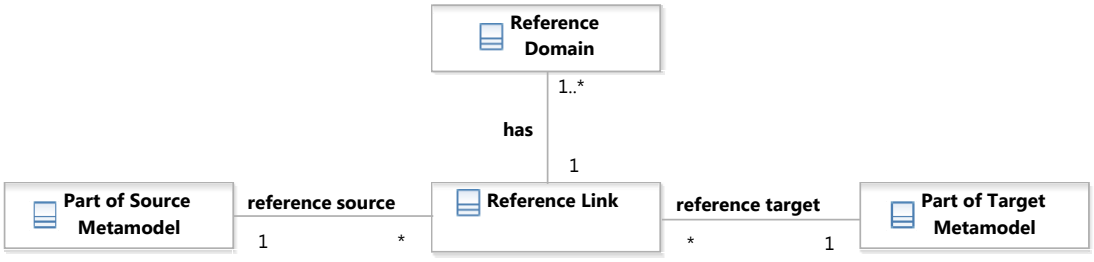

Figure 5.20.: Conceptual class diagram for integration by reference [Kü04]

Figure 5.20 illustrates this kind of integration. The conceptual class diagram from [Kü04] also includes an additional class called `Reference Domain`, which is used to define the cardinalities of the source and the target element.
This approach can be exemplified by integrating Interfaces and Information Flows with Business Application and Organizational Unit Relationship by introducing a reference from `ApplicationComponent` to `BusinessApplication`.

5.4.2.3. Integration by Common Base Class

A similar approach to integrating I-Patterns by using a reference, is to define a *common base class*, which can be used as an integrating element. Kühn [Kü04] calls this pattern Common-Baseclass-Pattern.

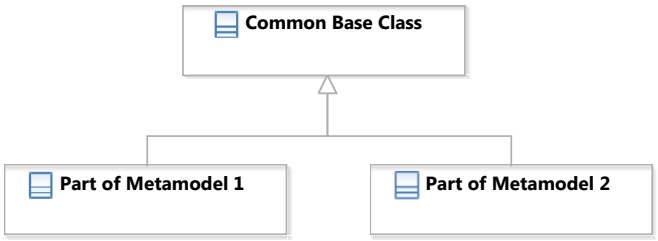

Figure 5.21.: Conceptual class diagram for integration by common base class [Kü04]

The integration approach is visualized in Figure 5.21. Here a common base class is defined for *parts of metamodel 1* and *parts of metamodel 2*.
As an example for integration by common base class, the two I-Patterns Business Application Lifecycles and Service Versions can be integrated by introducing a new base class for `BusinessApplication` and `Service` called `SupportProvider`.

5.4.2.4. Integration by Transformation

In cases where one or more source metamodel exist, which are *used independently* and which should *stay independently*, an *integration by transformation* can be used. This kind of integration is documented in the Transformation-Pattern [Kü04]. The transformation is used to create a new metamodel by using defined *transformation rules*.

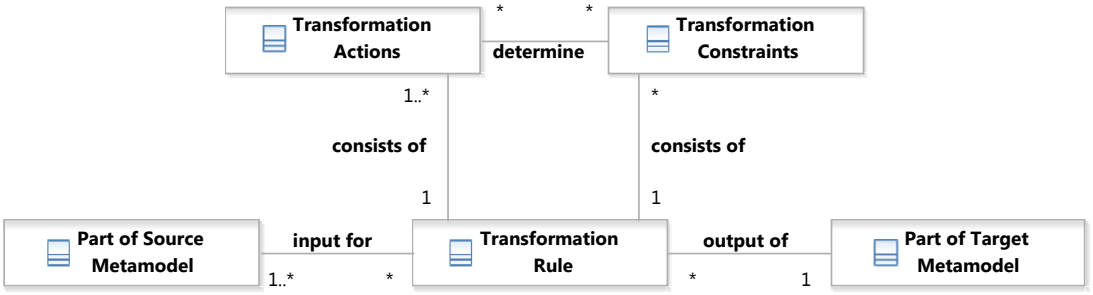

Figure 5.22.: Conceptual class diagram for integration by transformation [Kü04]

Transformation rules thereby consist of at least one *transformation action*, which can be determined by *transformation constraints*.
Figure 5.22 illustrates this integration approach. An example for such an integration may be that one group in an organization cares about managing the business processes of a company, while another group takes care of the business applications. A third group has to care about bringing together information about business processes, business applications, and augment it with additional information about future planning, without manipulating the original information. In such a case integration by transformation can be used to solve the integration problem.

5.4.2.5. Integration by Extension

Integration by extension documented in the EXTENSION-PATTERN [Kü04] is not a way to integrate different I-Patterns, but it shows how existing I-Patterns can be extended to create a deviated variant, by integrating *new concepts*. This can be useful for cases in which e.g. some attributes are missing in an I-Pattern prohibiting to address company specific requirements. For example it may be required to have a *full name* and a *short name* for business applications instead of a name only.
Requirements like this are common when trying to develop an information model based on selected I-Patterns. This can be seen in the case studies described in Section 6. The frequency of occurrence of such requirements is the reason, why this integration pattern by Kühn [Kü04] has been included here.

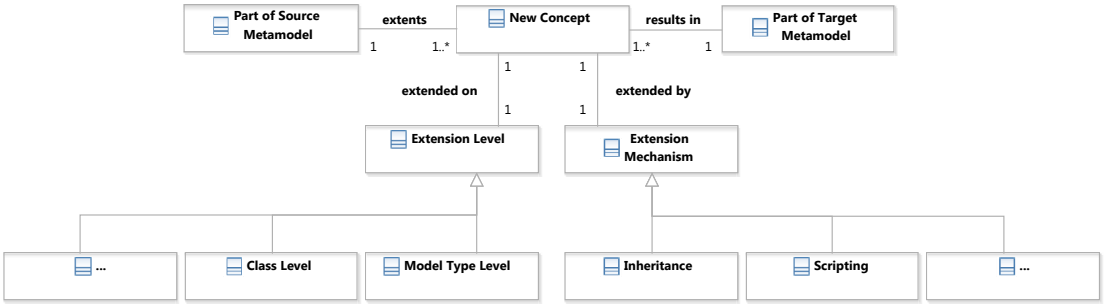

Figure 5.23.: Conceptual class diagram for integration by extension [Kü04]

Kühn [Kü04] illustrates this pattern by Figure 5.23. In this diagram, a *source metamodel* is extended by one or more *new concepts*. New concepts can be integrated on different levels like *class level*, *attribute level*, etc. In order to integrate the new concepts different mechanisms can be used like *inheritance*, *scripting* in cases where the extension is implemented in a tool, etc.

5.4.2.6. Problems in Integrating I-Patterns

Although, integrating I-Patterns, which use UML to document information model fragments, seems to be an easy task using the previously described integration patterns, but some problems may still occur. Some exemplary ones are described in this section.
According to Arbab et al. [Ar07] integrating models is likely to be problematic due to the fact that they have been developed by distinct stakeholders, with their own concerns. Relating different models means relating ideas of those different stakeholders, which are mostly implicit. Therefore, it may be problematic to think that there exists a one-to-one mapping for every model or in the case of the EAM pattern approach for each I-Pattern.
Another problem are attempts to identify classes for integration solely by identifying similar class names, although EAM pattern designers should simplify pattern integration by naming different concepts clearly differently, even across different EAM patterns. This cannot basically be expected, especially, if patterns from different catalogs with distinct authors are to be integrated.
Issues in this respect may originate from the simplifications inherent in the creation of models [St73], which may of course vary in degree of abstraction. A prominent example in this respect can be found in common abstractions of a *business application*. In some cases, a business application signifies an actual system installed in a specific environment and offering specific functionalities. In other cases, the term might specify the software itself, making no statements about specific installations. In addition, the usage of the term might also vary in respect to the versioning information included.
While, as stated above, sensible naming schemes, e.g. *BusinessApplication*, *DeployedBusinessApplication*, *BusinessApplicationVersion*, can help to prevent such problems, one must not rely on this alone. Exact definitions – a glossary – of the used concepts have to be provided by the pattern designers, in order to enable the pattern integrator to find possibly contradictory definitions in used concepts.
Additional integration problems are possible. For example in a case where different I-Patterns (*I-x* and *I-y*) are integrated, of which one (e.g. *I-x*) employs a mandatory, e.g. *1..**, relationship to a class contained in both patterns. In this case, it might be necessary to relax this relationship to an optional (*0..**) one, because otherwise collecting data for *I-y* might cause constraint violations in the model. Nevertheless, it should be possible to use *I-y* independently of *I-x*. More details on such integration problems can be found in [Bu08b].
Another frequent problem is that models describe the architecture at a certain point in time, while others describe architectural changes over time [Ar07]. Trying to integrate those models may lead to problems.
Despite the previously described problems, all integration approaches presented in Sections 5.4.2.1 to 5.4.2.5 have been applied and evaluated in the case studies by

Böhme [Bö08] (see Section 6.1), Dierl [Di08] (see Section 6.2), and Pflügler [Pf08] (see Section 6.3). The case studies showed that these approaches can be applied in practice. Nevertheless, the approaches for integrating I-Patterns described above strongly rely on the integrator's skills in conceptual modeling. Therefore it seems advisable to rely on the help of an expert in conceptual modeling when integrating I-Patterns.

5.5. Operations on the EAM Pattern Catalog

In [Co97b] Coplien states "[...]don't count on being able to mine a pattern language. A pattern language comes together like a jigsaw puzzle from patterns that have been collected over months, years or decades. At some point it becomes publishable, but its evolution doesn't cease at that same point."
This can be summarized as compiling a pattern language for EA management is no one time approach. On the one hand the EAM Pattern Catalog cannot be considered to be complete, meaning that all problems in EA management are yet completely addressed. There are new emerging topics, which have to be considered in EA management in the future, like e.g. *Cloud Computing* [Er09], etc. On the other hand existing solutions should be revised an extended in the future to match the changing demands in EA management. Therefore, it is clear that there exist at least four different operations for changing EAM patterns. These operations should be guided by a steering committee caring about the correct execution of the processes underlying the operations, which are detailed in the following sections

5.5.1. Introduction of new EAM Patterns

The first operation is the *introduction of a new EAM pattern* into the EAM Pattern Catalog. This process is visualized in Figure 5.24. In this case, a pattern author or a group of authors *identify* a new EAM pattern in practice.

Figure 5.24.: Process of introducing a new EAM pattern

After this identification the pattern is *documented* in form of a pattern preferably using the form proposed in Sections 4.1.1 and 4.1.3 to improve usability and comparability within the EAM Pattern Catalog. Other pattern forms were presented in Section 3.4. This pattern then constitutes a *candidate pattern*.
As a next step, a *review* of this EAM pattern is needed, supported by the EAM Pattern Catalog community or by other pattern communities. This is required to improve readability and understandability of the pattern, which is usually achieved by a kind of writers' workshop [CW97]. Another reason is that the reviews can be used to ensure that the pattern really is a pattern, there my be cases where a procedure or an algorithm is documented as a pattern but is no real pattern, and that the pattern really constitutes proven practice. During this review process relationships between EAM patterns should

also be considered, because they are an important aspect of a pattern language (see Section 5.2). The last task that should be addressed in the review phase is to categorize the EAM pattern in order to support the selection of patterns (see Section 5.3) by EAM Pattern Catalog users.
In the last step of the introduction of a new EAM pattern the reviewed pattern should be *approved* by the EAM Pattern Catalog community. After this step the candidate pattern can be considered to be a new EAM pattern, which is part of the EAM Pattern Catalog. The approval is important to guarantee consistency and integrity within the EAM Pattern Catalog.

5.5.2. Revision of EAM Patterns

The second operation is the *revision* of an existing EAM pattern, which requires five steps. A revision may be needed, e.g. because problems in EA management have changed or new technologies offer new possibilities like animated software maps, etc.

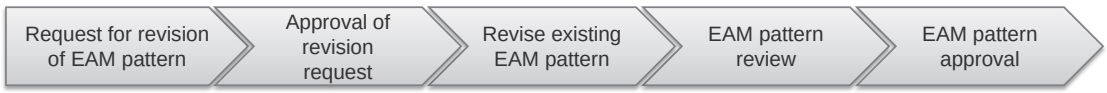

Figure 5.25.: Process of revising and existing EAM pattern

The first step (see Figure 5.25) in the revision of an existing EAM pattern is the *request for revision* by one ore more EAM Pattern Catalog community members.
This request has to be *approved* by the community in order to start the *revision* of the EAM pattern.
The steps EAM pattern *review* and EAM pattern *approval* are identical to the previously described operations to introduce a new pattern, except that the review should additionally consider that the versioning information has been updated.

5.5.3. Merge and Split of EAM Patterns

The third operation is merging or splitting of existing EAM pattern(s). A merging of two or more patterns becomes reasonable, if the usage of those patterns shows that they are too similar concerning problem, forces, solution, and consequences, to be maintained separately.

Figure 5.26.: Process of merging or splitting EAM patterns

In contrast there are two main reasons for the splitting a pattern. The first reason is that a pattern may have become too large to be handled and to be understandable, e.g. due to a revision of the pattern. In this case, it can be split up in two related patterns or a composite pattern with associated sub patterns (see Section 5.2 for more details on relationships between patterns). The second reason is that the variants documented in

one pattern have become so detailed that it is reasonable to split them in two or more related patterns.
The process steps (see Figure 5.26) during the merging or splitting of an EAM pattern correspond with the steps during the revision of a pattern (see section 5.5.2).

5.5.4. Removal and Retirement of EAM Patterns

The *removal* of an EAM pattern is a special operation as a pattern, which may be still utilized by some EAM Pattern Catalog users, is excluded from the catalog. This may e.g. happen if a topic is no longer relevant for EA management. Therefore, it should be decided, if the pattern should really be removed or just *retired*. In this case the EAM pattern can still be used but is marked as *deprecated* in the EAM Pattern Catalog. In both cases relationships to this pattern have to be adapted.

Figure 5.27.: Process of removal or retirement an EAM pattern

Like in the two previously described operations a *request* for removal or retirement of a pattern is required (see Figure 5.27). This request has to be *approved* by the EAM Pattern Catalog community, followed by a *consistency check* of the EAM Pattern Catalog concerning the incoming relationships, which are no longer valid. After these steps the EAM pattern can be *removed or retired*.

5.6. EAM Pattern Catalog Wiki

As already mentioned the community aspect and the editorial aspects are important in the future extension and improvement of the EAM pattern approach (see Section 5.5). To address these, the EAM patterns are available on a web based platform using Tricia [Bü07]. The name for the web based platform change from *Toro* to *Tricia* in 2008. Figure 5.28 shows a screenshot of the *EAM Pattern Catalog Wiki*, which is available at http://eampc-wiki.systemcartography.info/.
As of 2009-09-05, 385 users are registered and the number of visitors of the EAM Pattern Catalog Wiki continually increases (see Figure 5.29) since its initial go-live in July 2008. There is only one decrease of visitors in February 2009 but this is due to changes in the utilized analytics software, which inhibited to collect usage data. The peak in May 2009 can be explained by the EAM Pattern Catalog online survey (see Section 7.3), which resulted in an increased interest in the EAM Pattern Catalog Wiki.
The EAM Pattern Catalog Wiki currently does include 164 patterns (34 M-Patterns, 74 V-Patterns, 54 I-Patterns, and 2 Anti Patterns). These were not only be provided by the System Cartography research group, but also by external authors supporting the EAM pattern approach. Some of them were submitted to and revised during the PEAM 2009 workshop (see Section 6.7).
Tricia was developed to include collaborative aspects, like e.g. user management, multi-user support, file management, etc. Therefore, this platform supports the requirements of the EAM Pattern Catalog community. Additionally, the introspective features of

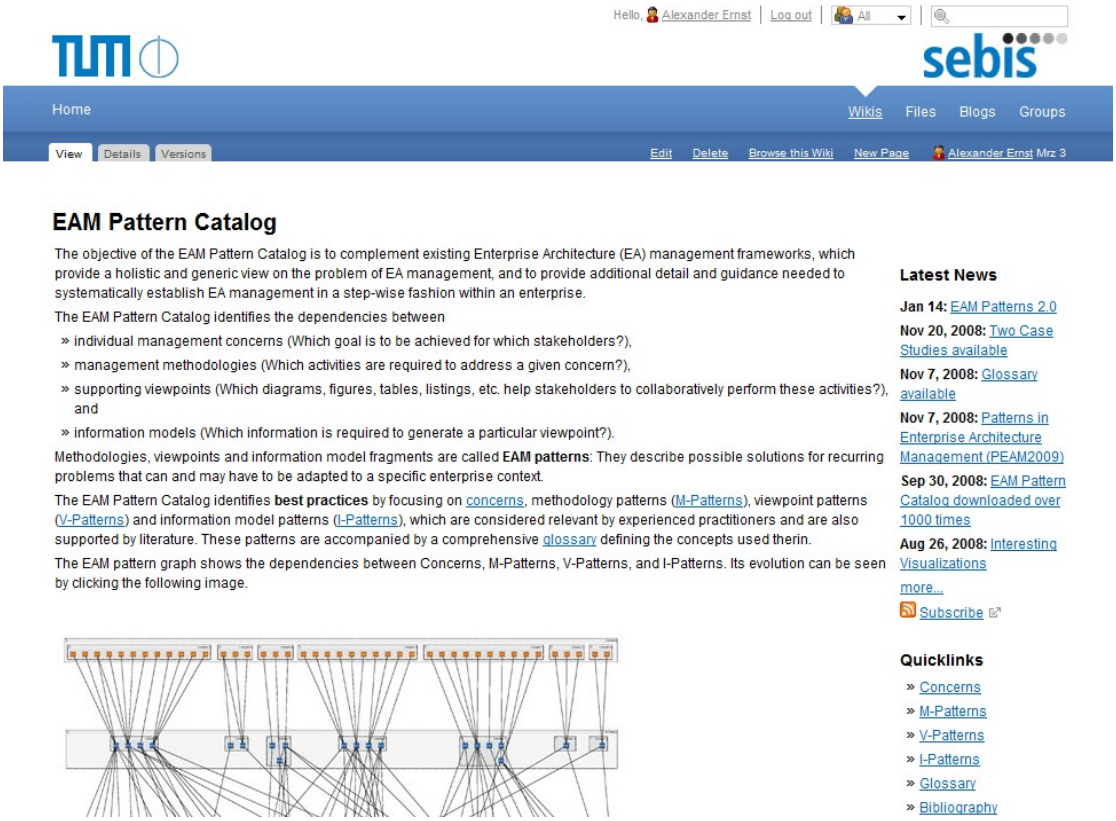

Figure 5.28.: Screenshot of the EAM Pattern Catalog Wiki

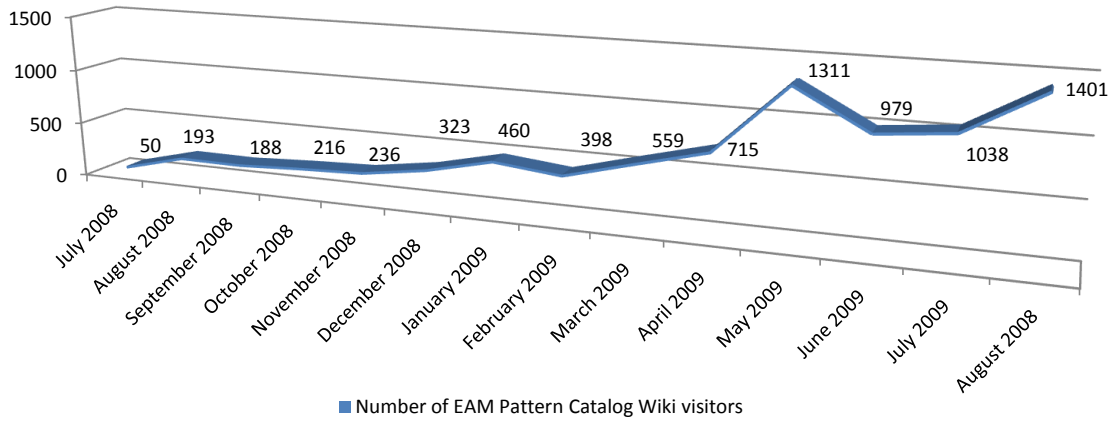

Figure 5.29.: Usage of the EAM Pattern Catalog Wiki (as of 2009-09-03)

Tricia offer the possibility to utilize advanced analysis methods. This can for example be used to dynamically create an EAM pattern map, based on the references between the different wiki pages used to document the EAM patterns. This can e.g. be done using a model transformation based approach presented in [Bu07b], which is implemented in the SyCaTool [Sc06a, Bu07b]. Maintaining an up-to-date map of the relationships between the patterns is important, as it provides one way to select EAM patterns and the EAM pattern map allow to get an overview about the EAM pattern language.
Web 2.0 technologies like tags are also supported and can be utilized in supporting a user of the EAM Pattern Catalog to find the right patterns for his problem, e.g. by assigning EAM patterns to different topics. An RSS feed is also available to keep up-to-date about the latest changes in the EAM Pattern Catalog Wiki.

5.7. Summary

This chapter introduced the EAM Pattern Catalog as a container for EA management pattern language. It was created as part of the *Enterprise Architecture Management Viewpoint Survey* and was validated in an extensive online survey. The analysis, consolidation, and documentation of knowledge on EA management during the creation of the EAM Pattern Catalog is one of the main contributions of this thesis. This thesis analyzed EA management and documented proven practices as EAM patterns and bad practices as EAM anti patterns. It took 16 months and resulted in a document including more than 320 pages. Such an extensive approach has never been pursued before in the domain of EA management. This justifies the importance of this work, which developed a pattern-based approach to EA management.
An important aspect in every pattern language are the relationships between the patterns included in the language. Therefore, an elaboration on the different relationship types used in the EAM Pattern Catalog was presented.
The pattern language currently consists of 164 (as of 2009-09-05) EAM patterns and EAM anti patterns. This requires support for identifying and selecting EAM patterns, which was introduced in this chapter. Three different approaches were presented: Concern-based selection, maturity model-based selection, and selection based on pattern language grammars and design space analysis. These three approaches were complemented by an outlook on additional ways to organize and select patterns.
After selecting the right EAM patterns it is required to integrate them to a customized approach for EA management, which was introduced for M-Patterns and I-Patterns. Thereby, also possible problems in integrating I-Patterns were presented.
EAM patterns, like all patterns, are not documented in a one-time approach, but are documented as an initial version, which is continually extended and revised. To accommodate this pattern life cycle typically operations like introducing new EAM patterns, revising EAM patterns, etc. were presented as a guideline for the future development of the EAM Pattern Catalog.
The last section of this chapter presented the EAM Pattern Catalog Wiki as a platform for the usage and maintenance of the EAM Pattern Catalog. With a continually growing community, supported by the functionality of the utilized wiki platform, everything is prepared for the future development of the EAM Pattern Catalog.

CHAPTER 6

Application

Contents

In order to validate the pattern-based approach to EA management and its different usage scenarios multiple case studies were undertaken with six of them detailed in the following sections. In addition an academic workshop *Patterns in Enterprise Architecture Management (PEAM) 2009* was conducted in conjunction with the *Software Engineering (SE) 2009* conference to evaluate the acceptance in the academic community. Beyond that the EAM Pattern Catalog was also applied in the lecture *Software Engineering for Business Applications - Master Course* [se09c] conducted by the chair for informatics 19 at the Technische Universität München.
Table 6.1 shows an overview about the applications of the EAM Pattern Catalog and their relationships to the four usage scenarios (see Section 4.2) of the EAM pattern approach: *Establish an EAM approach* (Scenario 1), *Inspiring and assessing an existing EAM approach* (Scenario 2), *Specify requirements for EAM* (Scenario 3), and *Academic research* (Scenario 4).

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
University of Munich Hospital (see Section 6.1)	**X**			
Financial Services Provider (see Section 6.2)	**X**			
Insurance Company (see Section 6.3)	**X**		**X**	
Detecon (see Section 6.4)	**X**	**X**		
Christiana Care Health System (see Section 6.5)	**X**	**X**		
E.ON (see Section 6.6)		**X**	**X**	
PEAM 2009 (see Section 6.7)				**X**
Software Engineering for Business Applications - Master Course (see Section 6.8)	**X**	**X**		**X**

Table 6.1.: Case Studies for the application of the EAM Pattern Catalog

The following sections will present an overview and an analysis about the conducted case studies.

6.1. EAM Pattern Catalog at University of Munich Hospital

The first case study presented in this thesis is a bachelor's thesis by Böhme [Bö08], which was conducted in cooperation with the *University of Munich Hospital*. Its main goal was to apply the EAM Pattern Catalog in practice in order *establish a pilot project for introducing an EA management approach* for the hospital and as a first step to document the application landscape.
The thesis pursued a five step process shown in Figure 6.1. In the first step – *Analysis* – the scope of the planned EA management approach as well as the required concerns of the hospital were defined and evaluated. The second step – *Design* – identified the relevant M-Patterns, V-Patterns, and I-Patterns from the EAM Pattern Catalog. In cases where the existing patterns were not sufficient, new ones were documented. Within this step the required data, specified by the selected and created I-Patterns, was collected. During the *Implementation* the views based on the selected and newly documented V-Patterns were created and were validated in the *Test* activity. The last step – *Maintenance and Usage* – defined the process for future update of data and views in the hospital.
Table 6.2 gives an overview about the utilization and the extension of the EAM Pattern Catalog in the bachelor's thesis. Numbers in the second column are based on the concerns and the EAM patterns included in [Bu08a], numbers in the third column give the finally selected EAM patterns, including the newly documented ones, compared to the initially selected EAM patterns, which are given in brackets. The number of initially selected EAM patterns is typically higher then the number of finally selected ones. This is due to the discussions, adaptations, and exclusions of patterns during the design activity.

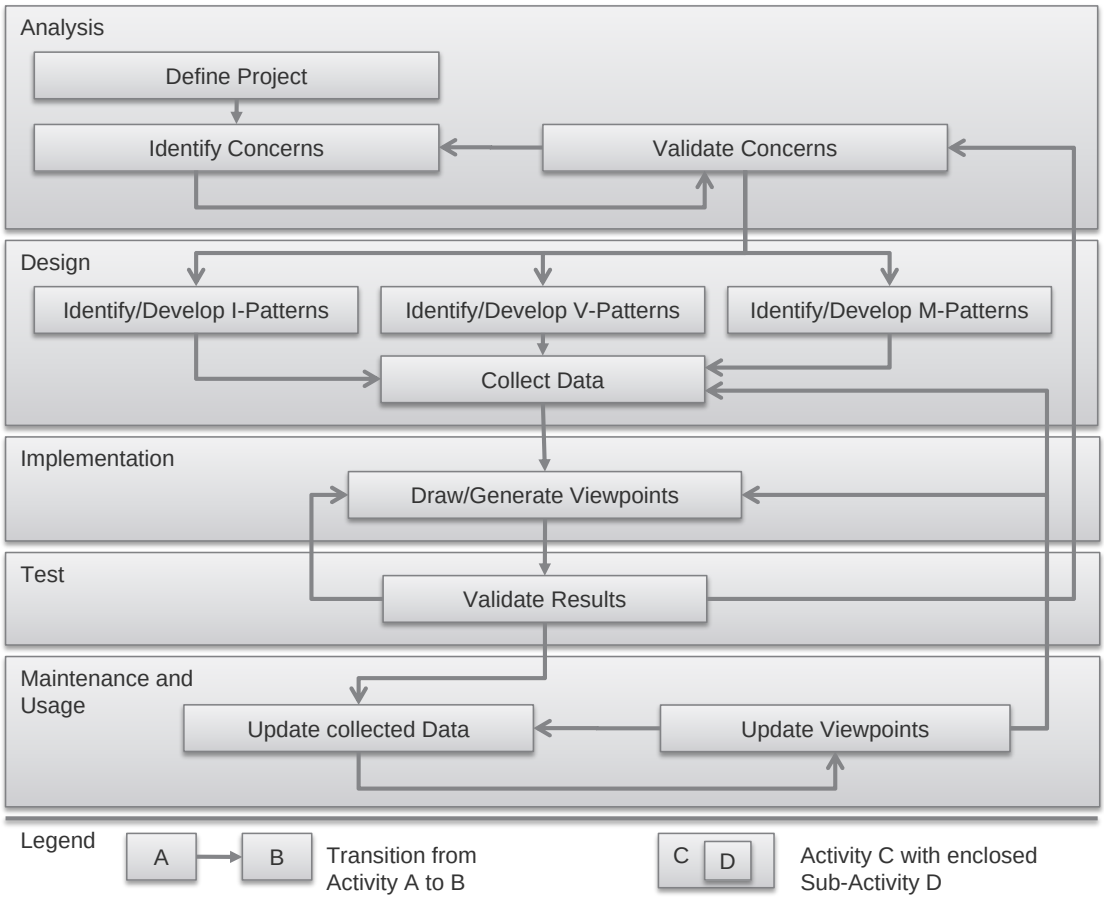

Figure 6.1.: Course of action as pursued in [Bö08] (visualization based on [Bö08])

EAM patterns are excluded because some are variants solving the same problem, require too high efforts to be implemented, etc. The last two columns show the percentage of usage of the EAM Pattern Catalog during the thesis and the number of patterns newly documented during this bachelor's thesis.

	Available	Selected	Percentage of Usage	Newly Documented
Concerns	43	6	14%	1
M-Patterns	20	2 (4)	10%	1
V-Patterns	53	4 (10)	8%	2
I-Patterns	47	4 (10)	9%	2

Table 6.2.: Overview about the usage, extension of the EAM Pattern Catalog in [Bö08]

The percentage of usage for the EAM patterns is rather low due to the following two reasons. At first the goal of the thesis was to establish an initial EA management approach for the hospital, with a narrower focus on the main pain points. The second

reason is that five new EAM patterns were documented to address specific requirements first found in the hospital.
The resulting information model form this thesis, based on the four selected I-Patterns, is shown in Figure 6.2. A color-coding highlights the basic I-Patterns. Additional documentation on the developed information model and how the I-Patterns were integrated can be found in [Bö08].
During the revision of the EAM pattern approach based on [Er08] the two newly documented V-Patterns and the two I-Patterns were altered to become variants of already existing patterns (see [se09a]).
A major result of this bachelor's thesis is that the pattern-based approach to EA management enabled a person with so far no experience in EA management to create an initial company-specific EA management approach based on proven practices. Even though additional guidelines for applying the pattern-based approach would be valuable it was possible to apply the EAM Pattern Catalog in the hospital without such guidelines.
During the bachelor's thesis it showed that in order to established a continuous and efficient EA management, organizational structures have to be created. Refined and new M-Patterns like the ones in [Er08] may be helpful in this context.
The bachelor's thesis further showed that the pattern-based approach to EA management can be applied in domains, which have not been considered during the development of the EAM Pattern Catalog, like e.g. the health care domain. Although some new requirements were identified, the fundamental concept was also applicable in the new domain. A positive result is that those new requirements could also be incorporated in the EAM Pattern Catalog. The adaptations that had to be applied to select patterns stayed on an expectable level, as adapting EAM patterns to the project-specific context is a basic idea of the pattern-based approach to EA management.
In this respect, the EAM Pattern Catalog offers an initial set of patterns, which can be selected, adapted, and extended in an iterative way. The final and most important result is that this does well reflect the typical procedure within a company, e.g. a hospital, usually lacking a proven practice based starting point, which is available in the EAM Pattern Catalog.

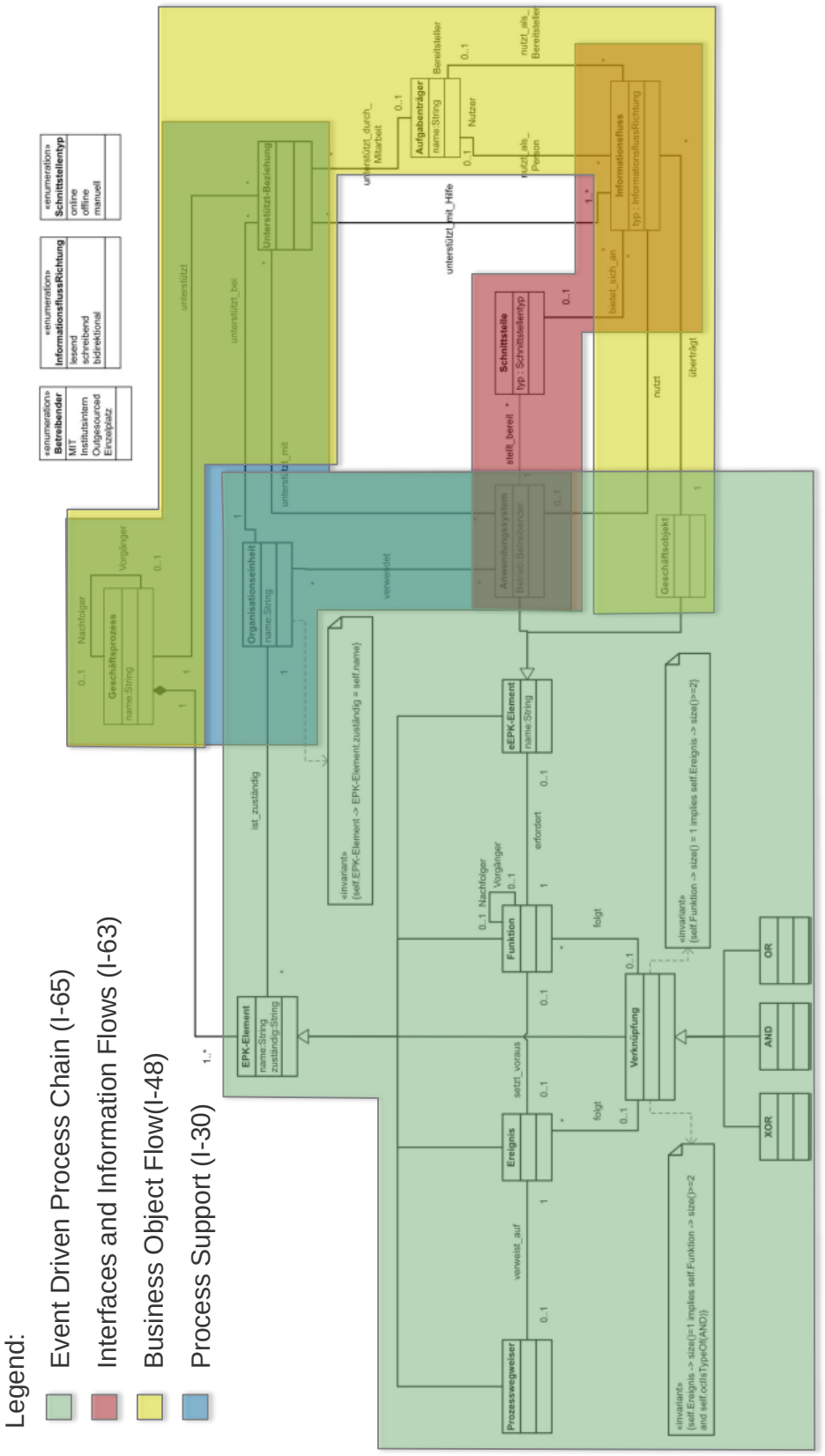

Figure 6.2.: Information model developed in [Bö08]

6.2. EAM Pattern Catalog at a Financial Services Provider

The second case study by Dierl [Di08] was conducted as a bachelor's thesis in cooperation with a large German financial services provider offering insurance, banking and asset management products, and services. This company was founded in 1890 in Berlin and operates world-wide with its headquarter located in Munich. It currently employs 181.000 employees servicing 80 million customers in 70 countries. It is based on total revenue and market capitalization the largest insurance company world-wide and one of the largest financial service providers.

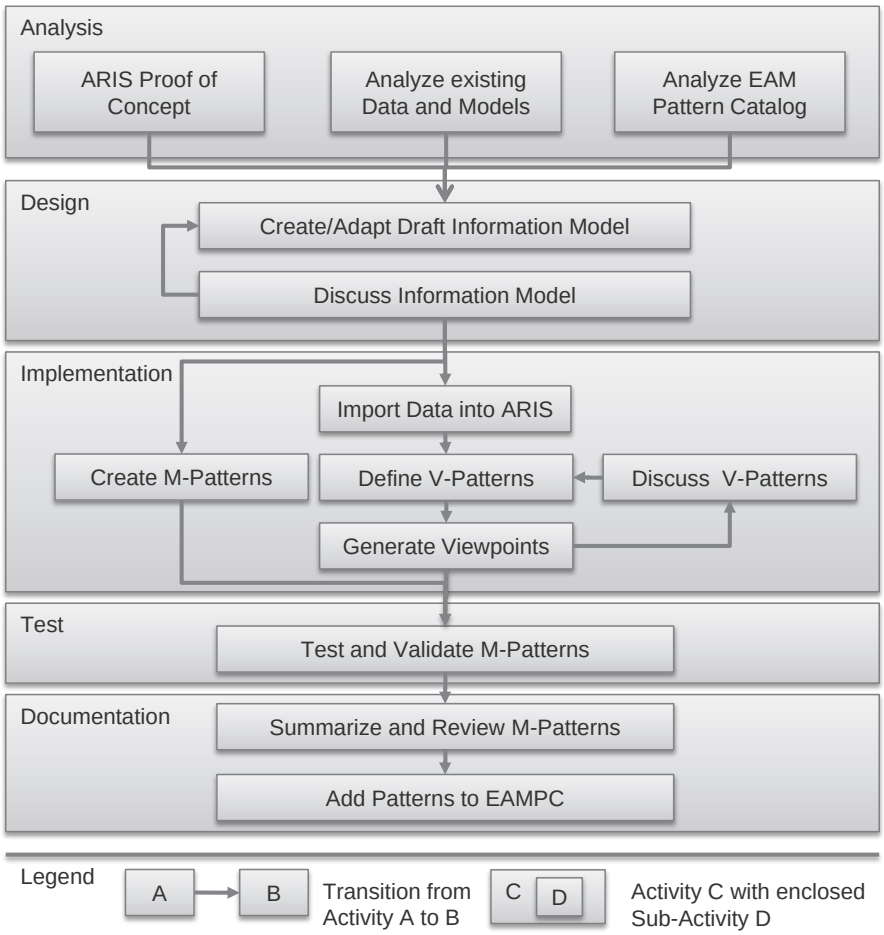

Figure 6.3.: Course of action as pursued in [Di08] (visualization based on [Di08])

The goal of the EAM Pattern Catalog application in this bachelor's thesis was to evaluate and extend the support of *compliance management*, and to establish an initial compliance management function within the company. Therefore, an information model was developed based on I-Patterns, corresponding visualizations were defined using V-Patterns, and methodologies were documented in M-Patterns.
The thesis followed a five step process shown in Figure 6.3. The first step – *Analysis* – was concerned with the initial orientation, including a proof of concept of the EA man-

agement tool *ARIS* [ID09], an analysis of the already existing information models and associated data, as well as the familiarization with the EAM Pattern Catalog. During the second step – *Design* – I-Patterns were selected, revised, and new ones documented in order to create an information model, which could be used for compliance management. As described in Figure 6.3 this information model was developed iteratively, employing several revision cycles. It showed that two iterations were sufficient to achieve an information model accepted by a majority of stakeholders.
The *Implementation* step focused on the documentation of new M-Patterns, because the EAM Pattern Catalog did not include M-Patterns on this topic. Additionally, required V-Patterns were iteratively selected and adapted, and new ones were documented in this step. During this activity, the visualization capabilities of the EA management tool were strongly considered. The fourth step – *Test* – was concerned with validating the M-Patterns, their associated V-Patterns, and I-Patterns. In the *Documentation* step the developed M-Patterns were reviewed and afterwards included in the EAM Pattern Catalog Wiki [se09a], together with the other revised or newly documented EAM patterns.
Table 6.3 gives an overview about the utilization and the extension of the EAM Pattern Catalog in the thesis. The structure of the table is similar to Table 6.2, except that adapting previously existing EAM patterns is differentiated from documenting new ones. An adaptation in this case means, that a pattern e.g. is extended by a variant. but not that for example an information model fragment is extended by a new attribute.

	Available	Selected	Percentage of Usage	Adapted	Newly Documented
Concerns	43	7	16%	0	0
M-Patterns	20	2 (1)	10%	0	1
V-Patterns	53	10 (4)	19%	3	6
I-Patterns	47	6 (5)	12%	0	2

Table 6.3.: Overview about the usage, adaptation, and extension of the EAM Pattern Catalog in [Di08]

The percentage of usage shows that even though the case study focused on compliance management, a topic which had not yet been addressed by the EAM Pattern Catalog at that time, well over 14% of the available EAM patterns could be used to develop an EA management approach for compliance management. This percentage should increase in future projects on the same topic, because the newly documented EAM patterns are now available in the catalog. The positive aspect is that such an iterative extension of the EAM Pattern Catalog will lead to a more and more complete catalog.
The information model developed in the thesis is shown in Figure 6.4. A color coding highlights the underlying I-Patterns.
Establishing a compliance management for the financial services provider using the EAM Pattern Catalog led to additional topics, which should be addressed in the future development of the EAM Pattern Catalog, e.g. by introducing new patterns or by extending existing ones with additional information. Two of those topics are related to metrics in EA management.
How to calculate the benefit of compliance management? Thereby not only the returns,

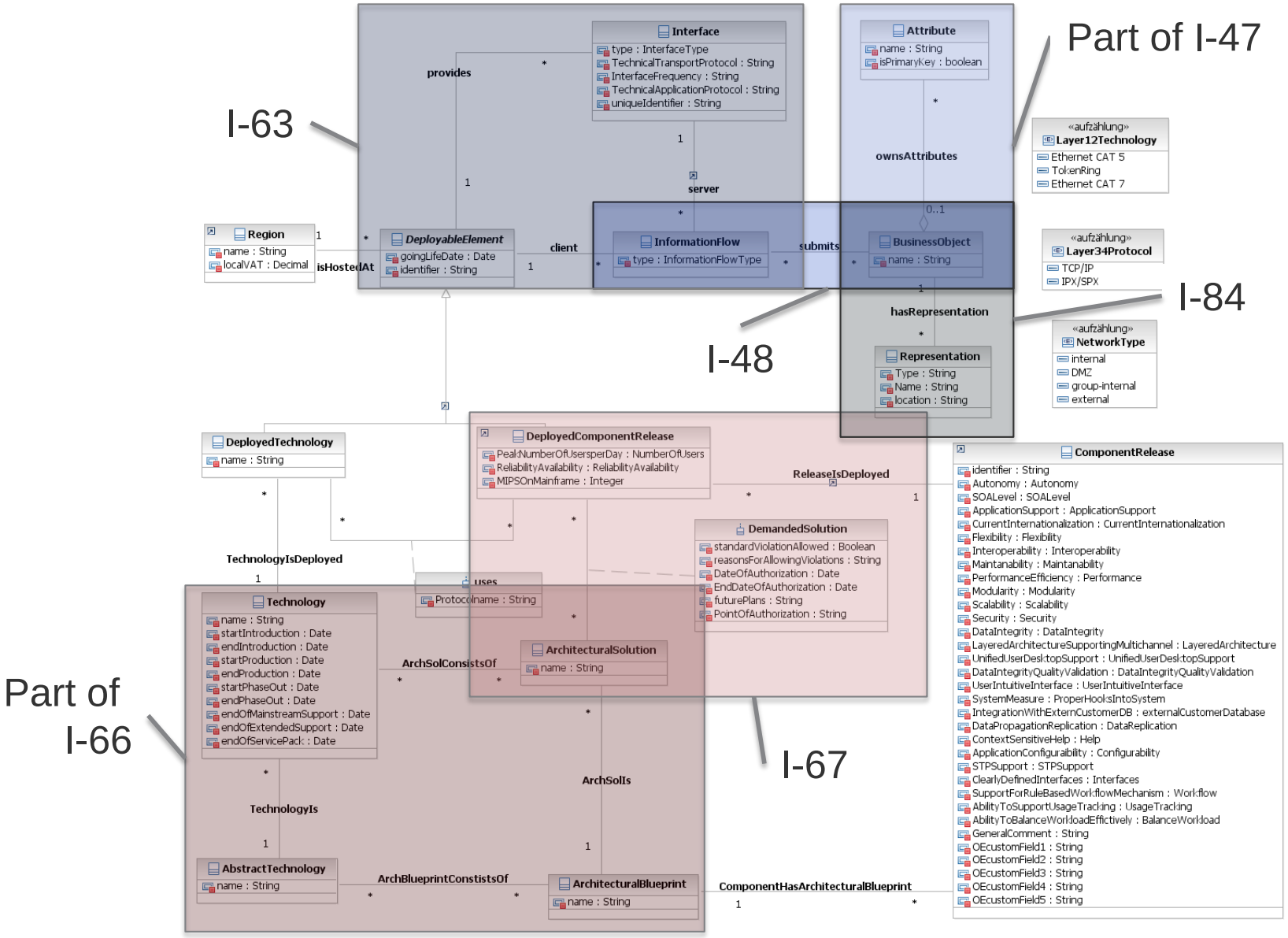

Figure 6.4.: Information model developed in [Di08]

but also the investments have to be considered. Evaluating the benefit is not only limited on compliance management, but can also be applied to various other topics like the benefit of landscape management, etc. This makes it a candidate for one or more EAM patterns.
Another topic concerning metrics is data quality. *Is the data quality getting better after the availability of automatic generated reports?* Like the previous question this can be generalized to data quality in general.
The third topic, which should be considered, is concerned with *social impacts of the achieved transparency.* EA management must always include the people somehow associated with the EA, especially because communication is an important aspect of EA management. This makes the social impacts of managing the EA an important but not yet sufficiently addressed topic. Therefore, it should be considered in the future development of the EAM Pattern Catalog.
Beyond those additional topics the case study shows that the EAM Pattern Catalog provides valuable support for compliance management. Using the iterative procedure to extend and revise the EAM Pattern Catalog this support will even be improved in the future.
The EAM Pattern Catalog is designed to be usable as a reference book. One result of this bachelor's thesis is that the EAM Pattern Catalog can be used in exactly this way. The EAM Pattern Catalog allows fast orientation in the field of EA management and it provides a strong basis to build on. Another important result following from the reference book concept is that the EAM Pattern Catalog offers the possibility for a rapid prototyping and implementation of information models and viewpoints, using an iterative process. This additionally offers the possibility for early consideration of EA management tool related constraints.
At last, one important result of the first case study could be supported in this bachelor's thesis: The pattern-based approach to EA management enables people without in-depth knowledge in EA management to create an initial company-specific EA management approach based on proven practices.

6.3. EAM Pattern Catalog at an Insurance Company

Case study number three was conducted as a bachelor's thesis by Pflügler [Pf08] in cooperation with a German insurance company. The insurance company subsumes 15 individual enterprises and has 6400 employees. It is one of the largest primary insurers of Germany with over 6 million customers.
Compared to the second case study (see Section 6.2) the goal was slightly different. In both cases an *initial EA management should be established*, but in this thesis a second goal was to *specify the requirements for an EA management tool* in order to support the tool selection for the insurance company.
The bachelor's thesis followed a six step process (see Figure 6.5), starting with *Investigating former and current approach.* In this step the previously implemented EA management approach was analyzed for addressed concerns, used visualizations, and information models, as the new approach should include all features of the previous approach. The second step – *Applying the EAM Pattern Catalog* – included the steps to select the ex-

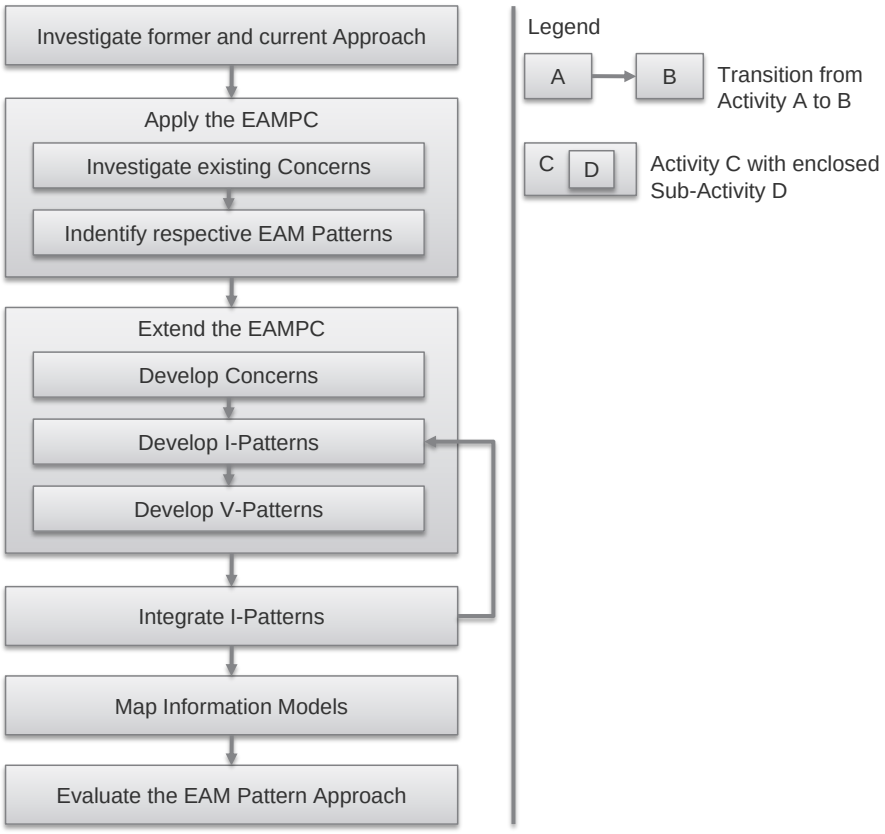

Figure 6.5.: Course of action as pursued in [Pf08] (visualization based on [Pf08])

isting concerns from the EAM Pattern Catalog and to identify the corresponding EAM patterns. Selecting the important concerns was performed in tight cooperation with the involved stakeholders of the insurance company. *Extending the EAM Pattern Catalog* was the third step and included the documentation of new concerns, new I-Patterns, and new V-Patterns. Selecting and developing M-Patterns was not in the focus of this case study, and therefore neglected. The fourth step – Integrating I-Patterns – was the most important step, as the results of this step were used in step *Mapping Information Models* in order to evaluate EA management tools concerning their support for requirements defined by the selected and newly documented EAM patterns. This evaluation was used by the insurance company to select an EA management tool. Integrating, developing, and revising I-Patterns was conducted in an iterative manner similar to the way documented in Section 6.1 and 6.2. In the bachelor's thesis a process of four iterations has proven to result in a stable information model only requiring minor adaptations due to changing concerns of a company. The last step – *Evaluation of the EAM pattern approach* – analyzed the appropriateness of the EAM Pattern Catalog to support the establishment of an initial EA management approach and the specification of requirements for an Enterprise Architecture Management in a company.
Table 6.4 gives an overview about the utilization and the extension of the EAM Pattern Catalog in the thesis and is organized exactly as Table 6.3.

	Available	Selected	Percentage of Usage	Adapted	Newly Documented
Concerns	43	26	60%	0	8
M-Patterns	20	12 (12)	60%	0	0
V-Patterns	53	41 (35)	77%	1	6
I-Patterns	47	34 (29)	72%	2	5

Table 6.4.: Overview about the usage, adaptation, and extension of the EAM Pattern Catalog in [Pf08]

With a usage of 60% or higher of existing EAM patterns this case study shows that the EAM Pattern Catalog is a good starting point for an EA management approach which does not focus on infrequent topics like e.g. infrastructure management. Additionally, the number of adapted and newly documented concerns and EAM patterns together with the limited duration of a Bachelor's thesis indicates that extending the EAM Pattern Catalog does not required disproportional resources.
As previously mentioned, one of the goals of this bachelor's thesis was to develop an information model to specify requirements for an EA management tool. Figure 6.6 shows the result, color coding the used I-Patterns. More details about the construction of this information model and the integration of the I-Patterns can be found in [Pf08]. The description of the iterative process for integrating I-Patterns may be used as a basis for guidelines extending the EAM Pattern Catalog in the future.
Even though, 34 I-Patterns had been selected as a basis for the information model, it showed that due to the way I-Patterns are documented, it is still possible to handle and integrate them. Additional information and real world examples on the integration of I-Patterns can be found in [Pf08]. In this context it turned out that modeling alternatives documented in different I-Patterns should be emphasized more. This issue was addressed by Ernst [Er08] by introducing an explicit variant section where different modeling alternatives may be listed (see Section 4.1.1). Another issue, which became obvious in this thesis, was the linear structure of the initial version of the EAM Pattern Catalog. In this version it is required to select concerns, corresponding M-Patterns, and V-Patterns in order to find the I-Patterns, which help to address the concerns. Ernst [Er08] revised the EAM pattern approach to include the addressed problem in every EAM pattern and not solely in M-Patterns (see Section 4.1.1). This allows for reduced efforts for finding required I-Patterns, as they can be selected directly.
During the development of the information model, a benefit of the pattern-based approach revealed. Although the I-Patterns included in the EAM Pattern Catalog originate from different sources they nevertheless can be integrated to a cohesive information model base on best practices.
This is additionally supported by a consistent and meaningful glossary detailing the concepts used in the EAM patterns, which is included in the EAM Pattern Catalog. Even though the glossary may not perfectly fit all requirements of a using company, it can still be used as a basis for the development of a company-specific glossary.
Another benefit of the EAM Pattern Catalog is the EAM pattern map (see Section 5.2.4),

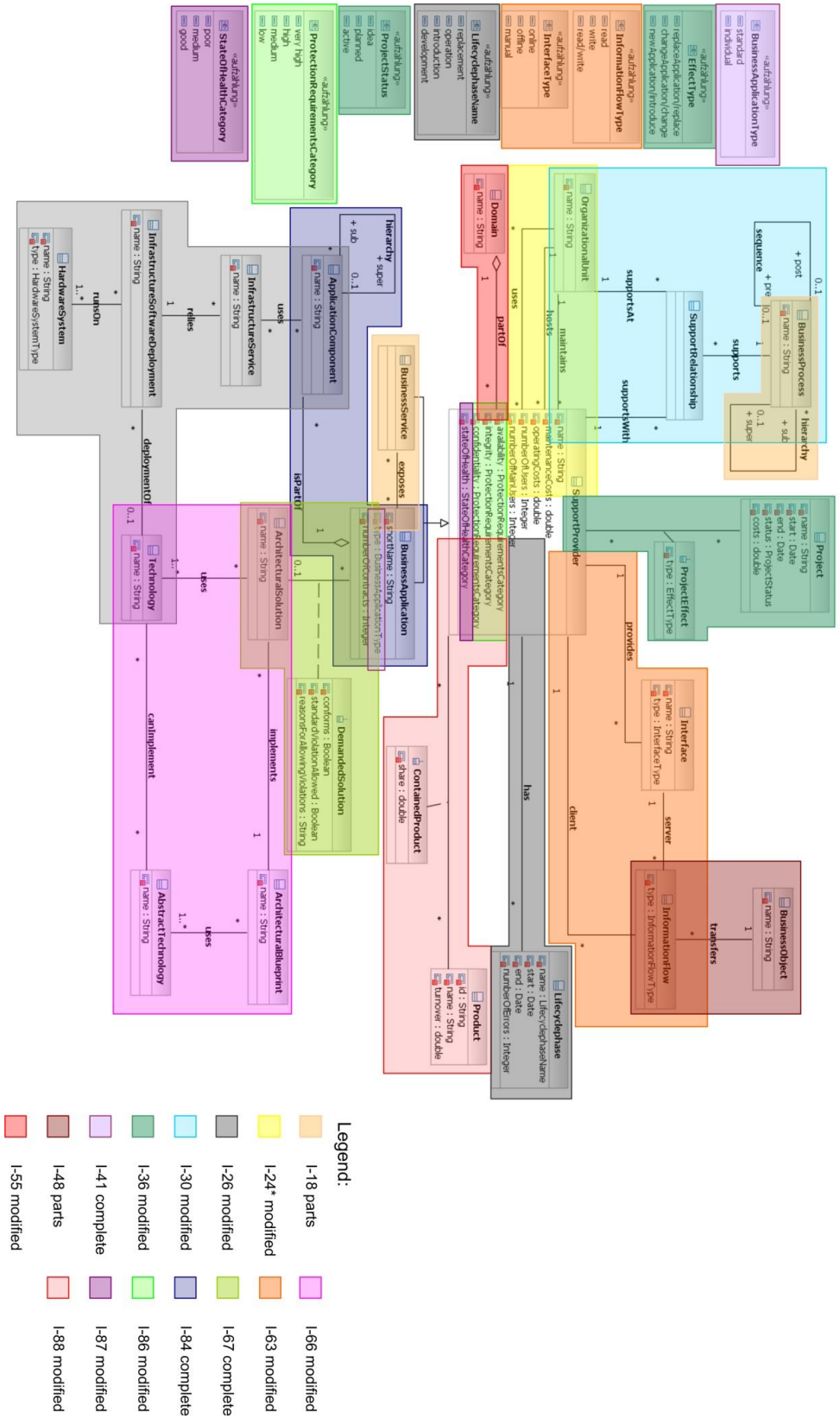

Figure 6.6.: Information model developed in [Pf08]

which can be used to find related patterns and to get an overview about the dependencies of EAM pattern language.
One possible entry point for selecting EAM patterns is the list of concerns included in the EAM Pattern Catalog. In this bachelor's thesis it turned out that this list has a major impact on the way pain points of a company can be discussed. This is because it represents a consolidated list of concerns of all companies, which have participated in the development of the EAM Pattern Catalog. Using this list for example offers the possibility to look for concerns, which have not yet been considered in a company for various reasons, but have occurred in other companies. This statement is additionally supported by the two following case studies (see Section 6.4 and 6.5).
Another important result of this bachelor's thesis is that sole usage of EAM Pattern Catalog was sufficient to establish an EA management in the insurance company, without having to consider additional EA management frameworks. Additionally, it showed that the EAM Pattern Catalog can be used to rapidly generate first results, which may be iteratively enhanced and extended. Using this approach, the created intermediary results can very well be used in discussions and workshops about the future development. One reason for this advantage is that EAM patterns document proven practices and are therefore easier accepted in a working group than custom build prototypes.
This case study, like the ones described in Sections 6.1 and 6.2, was conducted by a student, which at the beginning of the thesis had had no experience with EA management. Nevertheless, the student was able to specify an initial EA management approach and to elicit the requirements for selecting an EA management tool. Summing up, it can be said that the available EAM patterns are practice-oriented and fit the requirements of people and companies wanting to introduce EA management.
A last major result achieved in this case study was a new usage scenario for the EAM pattern approach: Specifying requirements for EA management (see Section 4.2.3). On the one hand these requirements can be used to support the selection of an EA management tool. On the other hand the requirements can be used to define the goals of an EA management approach.

6.4. EAM Pattern Catalog at Detecon

Case study number four is a project at Detecon International GmbH, a German consulting company with around 750 employees, four locations in Germany and ten international locations. Detecon applied the pattern-based approach to EA management in one of its consulting projects with a German telecommunication provider. The goal of the project was to document, revise, and extend the provider's information model in order to address not yet considered EA management concerns.
The project was conducted using the three steps shown in Figure 6.7. The first step *Analysis* was used to determine and document the currently used information model of the telecommunication company. Additionally, the stakeholders involved in EA management were identified, together with their corresponding concerns. These concerns were in a next step used to select I-Patterns from the EAM Pattern Catalog. In the second project step *Design* the selected I-Patterns were integrated with the previously existing information model, followed by a discussion of the integrated model. During the discussion a

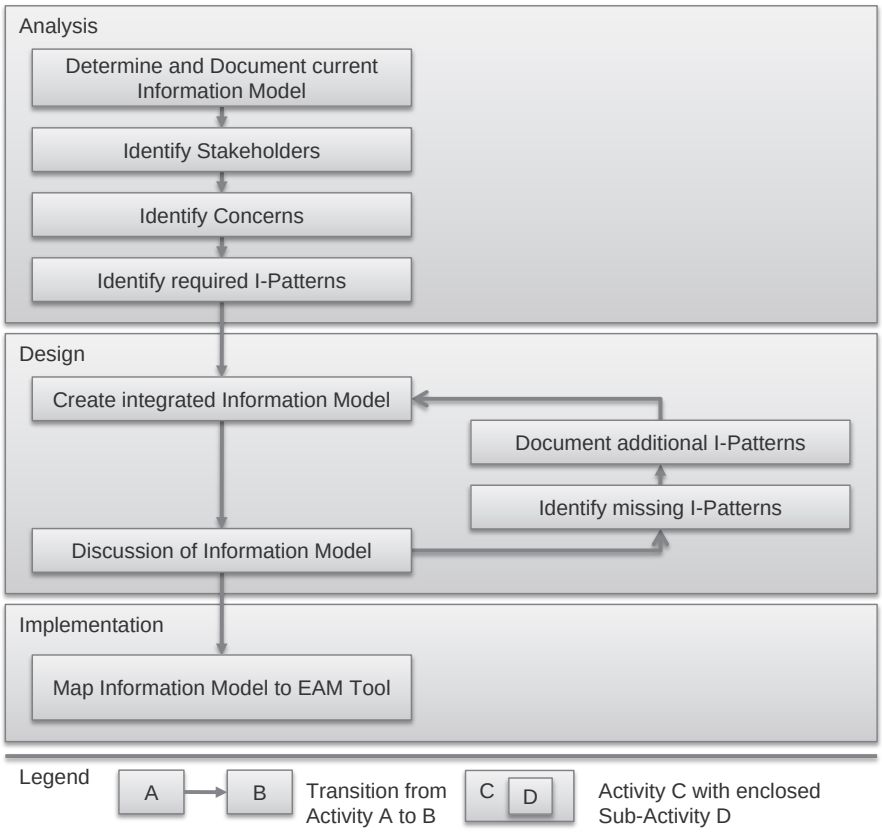

Figure 6.7.: Course of action of the consulting project

gap analysis was performed to find aspects which were not yet adequately addressed by the integrated model. These aspects included the topics security and compliance. The results of this gap analysis were used to document additional I-Patterns, which also were integrated into the information model. The design step was in addition used to analyze the integrated information model for potential areas of improvement, like e.g. information currently included, but not required in the EA management approach, required but missing attributes, etc. Missing attributes may be a problem, because I-Patterns document the minimal set of attributes required to address a certain concern. If for example a short name is required for a business application the corresponding attribute has to be added. After two iterations the extension of the existing model was finished and the resulting model was validated together with the customer.
In the third step of the project, *Implementation*, the extended information model was mapped to the one implemented in the EA management tool utilized in the telecommunication company, in order to use the model in the future.
Currently, there is no detailed information available concerning the utilization and the extension of EAM patterns during the consulting project for information model extension at Detecon, because of confidentiality restrictions. Therefore a table giving an overview about the utilization like in the previous case studies is omitted.
The consulting project led to three major results concerning the utilization of the EAM

Pattern Catalog. During the project it showed that there are some white spots in the EAM Pattern Catalog concerning *security* and *compliance*. These can be explained by the topics analyzed in the *Enterprise Architecture Management Viewpoint Survey*. The results of this survey were the basis for the first version of the EAM Pattern Catalog. Due to the extend of the survey not all topics could be addressed in an equal way. Another reason is that the literature, which could be used for mining patterns, is limited in this area.
Compliance has recently been addressed in [Di08] (see also Section 6.2), extending the support for security should be considered in the future development of the EAM Pattern Catalog. Due to the building block like structure of the pattern-based approach to EA management approach this can be done iteratively.
Besides the consulting project mentioned in this section the EAM Pattern Catalog was also applied in other projects of Detecon, for example as a guideline for interviews with the customers, to provide hints about possible concerns arising in EA management. The projects showed that the EAM Pattern Catalog can provide valuable input in those kind of projects
The major result of this case study is that the EAM Pattern Catalog could be used as a tool in the consulting project and could fulfill the expectations associated with the approach. On the one hand, the catalog provides a structured list of concerns in the EA management context, which can be used to identify the stakeholders and main pain points of a company. On the other hand, the catalog constitutes a valuable reference book for processes and roles, visualizations, and information model fragments, which can be used as a starting point for developing a new EA management approach customized to company-specific demands.

6.5. EAM Pattern Catalog at Christiana Care Health System

The EAM Pattern Catalog is not only used in Europe but worldwide. For in-depth considerations on the EAM Pattern Catalog's distribution see Section 7.3. As an example a case study at *Christiana Care Health System* (CCHS), which is based in Wilmington (United States of America), is presented below. Christiana Care Health System is one of the largest health care providers in the mid-Atlantic region, serving all of Delaware and portions of seven counties bordering the state in Pennsylvania, Maryland and New Jersey. The not-for-profit, privately owned Christiana Care family of services includes two hospitals (Christiana Hospital, on a suburban campus south of the City of Wilmington, and Wilmington Hospital, in the downtown business district of Wilmington) [Ch09].
The EA management program currently in place at CCHS started in 2008 and used the EAM Pattern Catalog to establish a green field EA management approach and to inspire and assess already existing EA management endeavors. The IBM System Architect [IBM09] is in place and is used as a repository to store information about the EA. Some of the information from the EA program is used to make decisions about the future development of the EA. The actual EA management process is not fully developed and documented. Therefore, for some of the areas addressed by the EAM pattern approach

information is available and processes are in place, but for other areas there is potential for improvement.
The elements of the EAM Pattern Catalog were used in various ways. Although an information model had already been created during the introduction of System Architect, I-Patterns were considered in discussions about rebuilding the information model. An exemplary utilization of I-Patterns at CCHS is the utilization of a modified version of Process Support [se09a] to update the existing model to represent applications and their versions together with their relationships to business processes, servers, and roles.
V-Patterns turned out to be very important and very valuable, because the program at CCHS relies on the idea that there are a handful of *standardized* viewpoints which are adapted to fit the needs. Instead of developing these viewpoints existing V-Patterns could be used, which are based on proven-practices. A future goal is to train the users and stakeholders to recognize the defined viewpoints so they can focus on content rather that deciphering the viewpoints' notation. At CCHS, for example, Business Application Planning [se09a] is used to model critical applications that are impacted by the transition to International Catalog of Diseases (ICD) revision ten. ICD-10 was developed by the World Health Organization (WHO) and is the tenth revision of an international classification of diseases, which is used to encode medical diagnoses [WHO09].
When the EA management program at CCHS started a multi-year plan, a capability maturity model [SE06] had been developed to model how the program will advance. This plan was used to establish the goals for each fiscal year. These plans were updated regularly based on changes in the environment in meetings with the CIO. Those goals were mapped to concerns of the EAM Pattern Catalog. A matrix was used to show what key projects are worked on and which goals/concerns will be addressed as a result of that work. Additionally, the concerns along with the program goals were used to communicate with stakeholders and colleagues.
EAM patterns were used in two more ways at CCHS. At first, several of the V-Patterns were used to present information from the repository to address questions from the EA management stakeholders. At second, EAM patterns were used in developing a business process taxonomy and a corresponding inventory.
Finally, one of the goals for the EA management program at CCHS in 2009 was the development of a documented method and associated templates. The EAM patterns were very helpful in doing that, so CCHS actually lowered the priority of this goal to focus on other work.
Currently, CCHS has no formal process established for using EAM patterns. They are more used in an ad-hoc manner: Find an EAM pattern that can be used to address a concern and start using or adapting it to fit the company's needs. This may change, when the EA management team becomes larger, the governance more formal, and the program more mature. In such a case, a process for identifying, approving, and adapting EAM patterns should be established.
Table 6.5 gives an overview about the utilization and the extension of the EAM Pattern Catalog in this case study and is organized as Table 6.3.
The utilization of the EAM Pattern Catalog at CCHS is a real world example and was not part of a thesis or any other academic work. Therefore, the goal was not do adapt or extend the EAM Pattern Catalog. As a result the columns *adapted* and *newly documented* of Table 6.5 only include zero values. The rest of the table shows that

	Available	Selected	Percentage of Usage	Adapted	Newly Documented
Concerns	43	23	53%	0	0
M-Patterns	20	11	55%	0	0
V-Patterns	53	37	70%	0	0
I-Patterns	47	28	60%	0	0

Table 6.5.: Overview about the usage of the EAM Pattern Catalog at Christiana Care Health System

over 50 % of the documented concerns and M-Patterns are used at CCHS. I-Patterns and V-Patterns scored up to 70 %. Those results show the good adoption of the EAM pattern approach in this case study, although the introduction of the approach was not supported or forced by the developers of the EAM Pattern Catalog.
During the application of the EAM pattern approach, two drawbacks were identified, which both are not concerned with the contents of the EAM Pattern Catalog. One problem arose while implementing V-Patterns in System Architect. At this point data from the System Architect is exported into an EA operation data store and SQL queries are used to retrieve the data. This data is then used to draw views by hand with Microsoft Visio. One way to solve this problem is to use model transformation to generate visualizations for EA management. Such an approach was described in [Sc06b, Bu07b] and was implemented in the so called SyCaTool. This tool is able to used arbitrary data, which has to be described by an information model in order to be interpreted. After specifying the transformation, e.g. by using information from V-Patterns, the tool can be used to ad-hoc generate views.
The second problem is that the EAM Pattern Catalog currently is available as a PDF file and as a wiki, which both offer limited support to graphically navigate between the contents of the EAM Pattern Catalog, because the EAM pattern map is separated from the patterns itself. Improving the integration could enable to walk around the catalog online, drill in and out of information, and perceive the catalog easily. Such an integration is planned for the future development of the EAM Pattern Catalog Wiki and uses a variant of the SyCaTool which can be integrated in wikis or any other web-based portals. This is further described in the outlook of this thesis (see Section 8.2).
The EAM patterns was very helpful in the EA management program of CCHS because they did not have to develop a standard methodology to capture and display information. The catalog could further convince the EA management stakeholders at CCMS as the pattern were recognized as proven practices for the field. People within CCHS can relate the EAM patterns to the methods and templates from the Project Management Office which also facilitated adoption.
To sum up the case study, EAM patterns allowed CCHS to focus more on doing the actual work rather than on creating structure. Further, the EAM patterns served as a basis for discussions so that all participants could start from a common ground when discussing a new area of interest.

6.6. EAM Pattern Catalog at E.ON

The EAM Pattern Catalog was successfully applied at E.ON AG. With total revenue of 87 billion EUR and around 93.500 employees, it is one of the world's largest power and gas companies. E.ON is present in 19 different countries with its headquarter located in Düsseldorf, Germany.
The goal of the EA team at E.ON is to define the role and responsibilities of a future EA and IT governance structure to support business agility and reach operational excellence. Key focus is the support for business changes based on mergers and acquisitions, divestments and consolidation.
Beyond the described main goal additional sub goals were defined. One of these goals was to define one common *enterprise architecture operating model* for the whole company. This model should be complemented by an EA management method including the definition of an EA repository, reference architectures, principles, and software engineering. Thereby gaps between the portfolio management, the project management and the service management method should be identified to improve and ensure architectural compliance. The most important task at this point was to define key deliverables and roles to add them to existing methods. Additionally, decision making groups at E.ON should be harmonized.
These goals are to be accomplished by a fast and agile EA unit which is developing an executable architecture method bridging the gap between business and IT. The result of the whole endeavor should be a clear EA method, which is supporting the diversity of the E.ON group and supporting multiple IT sourcing models like software as a service, cloud computing, outsourcing etc.
In this case study the EAM Pattern Catalog was used in different ways. First of all EAM patterns were used to define E.ON's requirements for an EA management tool respectively an EA repository. Additionally, the EAM Pattern Catalog was used to create a common understanding about patterns themselves and the categorization of patterns.
The EAM pattern concept was in a next step used to define the baseline for developing and publishing EAM patterns. For this task E.ON defined three different operations on EAM patterns which correspond to operations in the EAM Pattern Catalog. The relations are shown in Table 6.6.

Operations defined at E.ON	Operations defined for the EAM Pattern Catalog
Add new EAM patterns	Introduce new EAM patterns (see Section 5.5.1)
Modify EAM patterns	Revise EAM patterns (see Section 5.5.2) and merge and split EAM patterns (see Section 5.5.2)
Delete EAM patterns	Remove and retire EAM patterns (see Section 5.5.4)

Table 6.6.: Operations on EAM patterns at E.ON

From the organizational viewpoint, the enterprise architecture and technology management unit at E.ON has a federated architecture community setup, which is defined with

a core team and an extended EA team. The extended EA team covers several architectural domains like technical and application architecture as well as information (data and APIs) and process architecture. They work directly as solution or IT architects at the organizational units in order to maintain close relations to business and to ensure the knowledge transfer.
The EA group at E.ON works as one virtual EA management community. A yearly review of their EA operating model is conducted and main deliverables are managed in a project mode. These deliverables are linked to E.ON's IT governance and IT strategy unit.
A new EAM pattern can be proposed to the EA working group and is approved by the chief enterprise architect. The EA management working group also has to review on a yearly base the currently selected patterns in order to detect non-used or outdated patterns.
The EA management working group creates a change proposal, which is to be approved by the chief enterprise architect and is done in a service project mode for EA. This is similar to the process for approving changes proposed in Section 5.5. The EA management working group consists of IT architects and solution architects on senior level, which are nominated by the chief enterprise architect and IT management in order to ensure enough time to maintain the EAM patterns respectively the EA management service.
Table 6.7 gives an overview about the current utilization and the extension of the EAM Pattern Catalog at E.ON and is organized like in the previous sections.

	Available	Selected	Percentage of Usage	Adapted	Newly Documented
Concerns	43	unavailable	unavailable	0	0
M-Patterns	20	5	25%	0	0
V-Patterns	53	30	57%	0	0
I-Patterns	47	25	53%	0	0

Table 6.7.: Overview about the usage of the EAM Pattern Catalog at E.ON

The result of the EAM pattern utilization shows that the focus currently lies on V-Patterns and I-Patterns with a degree of usage above 50 %. By contrast only 25 % of the catalog's M-Patterns are used. The columns for adapted and newly documented EAM patterns are left empty because the corresponging process is currently established at E.ON. Information about the number of used concerns is unavailable due to confidentiality reasons.
During the application of the EAM Pattern Catalog at E.ON it turned out that some EAM patterns are excerpts of of other patterns leading to a pattern catalog including too many patterns. This problem was already recognized and was addressed by the revised form of EAM patterns, which includes a variant section. This can be used to summarize EAM patterns, which are closely related and constitute variants of each other. An example for this is presented by Ernst [Er08]. Additionally the concept of a composite pattern (see [Bu09e] and Section 5.2.1) was introduced to adequately address that some patterns are used as part of another pattern. Those two extensions have not yet applied on the complete EAM Pattern Catalog, but will be in the near future.

Another interesting drawback of the utilization of the EAM Pattern Catalog occurred at E.ON. It turned out, that it is easy to agree on the EAM pattern concept and its value, but it is hard to adopt it. The implementation requires a lot of internal education and communication. This problem may be addressed in the future development by offering additional guidelines and training material, like e.g. case studies, presentations, etc. in the EAM Pattern Catalog Wiki, which may reduce the required effort for adopting the EAM pattern approach.
Multiple benefits of using the pattern-based approach to EA management could be identified. Two of them support the proposed usage scenarios of the EAM Pattern Catalog. It turned out that the EAM pattern approach is useful as a first check point and reference for new architectural requirements (scenario 2), which means that previously existing work can be adapted resulting in higher speed during the development. A further advantage in this context is the building-block like structure of the EAM Pattern Catalog, enabling to create a step-wise process to EA deliverables and EA service models. Additionally, EAM patterns serve well for defining architectural requirements (scenario 3).
Besides the two previously described scenarios, it turned out that the EAM Pattern Catalog also supports easy leverage of EA content. This means that it is possible to implement an online pattern-matching function asking a few questions, for example about the concerns of the user or the topics he is interested in, to determine the right patterns to start with.
Two more benefits have to be emphasized, which showed in using the EAM Pattern Catalog. The first one is that it can very well be used as a reference in the case of merging different architecture teams and solution designers. In such cases EAM patterns can be used as training material to establish a common understanding between different groups. The second one is that the approach can be used as a reference for the CIO and other management stakeholders, because the approach constitutes proven practice in EA management across different companies and even across industries. Therefore, it is easier to introduce EA management in a company, which is based on proven practice then one which has been developed from scratch, especially in discussions with CxO or other management levels.

6.7. Patterns in Enterprise Architecture Management 2009

In March 2009 a workshop on *Patterns in Enterprise Architecture Management (PEAM)* was conducted as part of the Software Engineering 2009 conference in Kaiserslautern (Germany). 22 participants attended the workshop, which followed the style of typical pattern conferences like, e.g. PLoP, EuroPLoP, etc. Nine articles were submitted from practitioners and researchers from all over Europe. After the first review, the following six articles were accepted for the shepherding phase, during which the articles were revised and improved with the help of a shepherd.

Some Process Patterns for Enterprise Architecture Management

Moser et al. [Mo09a] present six patterns on data acquisition/maintenance, data release workflows, and data life cycle management for enterprise architecture management.

Enterprise Architecture Patterns for Multichannel Management
: Lankhorst and Oude Luttighui [LL09] present one pattern of a larger pattern language on multi channel management. These patterns provide functional structures for designing organizational and technical solutions that help organizations to manage and align the various information channels they use in communicating with their customers.

An Enterprise Architecture Management Pattern for Software Change Project Cost Analysis
: Lagerström et al. [LJH09] present one pattern on software change project cost analysis based utilizing Bayesian networks.

Patterns for Enterprise-wide SOA
: Cace [Ca09a] presents five patterns of a larger pattern language on service-oriented architectures, with a focus on multi-channeling.

EA Management Patterns for Consolidations after Mergers
: Buckl et al. [Bu09a] present three patterns on consolidating two or more companies after a merger utilizing capabilities to identify redundancies or gaps in business support.

EA Management Patterns for Smart Networks
: Lau et al. [La09a] present three patterns on smart networks, which are used for networking between partners not only on an organizational level, but also consider the information technology needed for the networking.

On site, a *writers' workshop* [CW97] was used to discuss a total number of 19 patterns, included in the accepted articles, to get additional feedback by all authors and participants of the workshop.
The workshop received positive feedback by the participants, including the request to conduct a follow-up workshop, e.g. as a focus group in a conference of the PLoP series. Four results could be achieved by the PEAM 2009 workshop. As a first result the EAM Pattern Catalog could be expanded by additional patterns, which are available at the EAM Pattern Catalog Wiki. Secondly the workshop brought together practitioners and researches to discuss and exchange information about patterns in EA management. The main results are that the visibility and acceptance in the academic community as well as in practice could be improve and that the first steps for initiating a community on EAM patterns could be taken.

6.8. Software Engineering for Business Applications - Master Course

In addition to the previously described case studies where the pattern-based approach to EA management was used in various companies it is currently applied in a lecture at the *Chair for Informatics 19 (sebis)* at the Technische Universität München in the summer term 2009. The lecture is called *Software Engineering for Business Applications - Master Course* [se09c] and teaches the challenges arising in the context of complex application landscapes and EAs. Students get to know established methods to control

and reduce the aforementioned complexity and learn how to apply EAM patterns to model and construct application landscapes.
EAM patterns were applied by around 40 students in ten so called *mini projects* in cooperation with seven different project partners of the research project System Cartography: Allianz Deutschland AG, Allianz SE, Bayerischer Rundfunk, UniCredit Global Information Services, Fiducia, Siemens Financial Services, and Wacker Chemie. Each mini project used the concept of EAM patterns to address restricted, well-defined problems at the corresponding project partner.
The application of the EAM pattern approach in a lecture will result in additional information about the advantages and the drawbacks of the approach, as well as on the applicability to real world problems. In addition, the results of the mini projects may be used to formulate extended guidelines for utilizing the EAM Pattern Catalog.
Unfortunately, the results will be available after the completion of this thesis for which reason they cannot be further detailed here. Hence they will be published as a technical report and considered in the future development of the pattern-based approach to EA management.

6.9. Summary

The goal of this chapter was to validate the pattern-based approach to EA management and to confirm the general research hypothesis for this thesis: If a pattern-based approach to EA management is used, the results are superior to results for other EA management approaches.
The PEAM 2009 workshop showed on the one hand side that the EAM pattern approach has been accepted by the academic community, because more than 20 people participated in the workshop, even though it was conducted in form of a writer's workshop, which is uncommon in the information science or software engineering context but popular in the pattern community. On the other hand the workshop showed that the form of EAM patterns can very well be used to document academic ideas for solving problems in EA management.
Similar results became apparent in the lecture using the EAM Pattern Catalog in its laboratory course. First results show that EAM patterns and their application are a good way to exemplify master level students what EA management is and how to address problems occurring in this context. Detailed results will be further analyzed after the completion of the lecture, which unfortunately will be after the submission of this thesis.
The case studies showed that the pattern-based approach to EA management has proven to be valuable in six different real world EA management projects, with three of them being conducted solely by the company, without support of the developers of the EAM Pattern Catalog. In each case study the defined goals could be achieved. Three cases studies even led to extended results, which were not in focus at the beginning of the project, like the utilization of concerns for defining goals of the EA management approach, or the usage of the EAM Pattern Catalog for structuring interviews with customers.
Concluding, this chapter showed that the general research hypothesis for this thesis could be confirmed. The pattern-based approach to EA management is valuable not only for academic research but also for application in real world projects.

CHAPTER 7

Evaluation

Contents

Applying the pattern-based approach to EA management in academia and practice is only way of validating it. For this reason the pattern-based approach will be compared to existing EA management approaches concerning their drawbacks and benefits. Additionally the results of a survey will be presented to prove the utility and validity of the approach.

7.1. Existing Enterprise Architecture Management Approaches

Various approaches for EA management are available and currently in use in academia and practice. This section presents nine of these approaches to compare them to the pattern-based approach to EA management proposed in this thesis in order to show benefits and liabilities of the approaches. For information about additional approaches for EA management see e.g. [Wi07a], [ARW08], [Fr09] etc.

7.1.1. Enterprise Architecture Frameworks

An EA management endeavor typically starts with getting an overview about existing EA frameworks or standards as a basis for an own approach. ISO/IEC 42010 [IS07] defines an architecture framework in the context of systems and software engineering as follows.

> **Architecture Framework:** Set of common practices for architectural description established within a specific domain or stakeholder community.

For a more detailed elaboration on what makes up an EA framework see [Fr09]. [Sc06a] and [Le07] give a comprehensive overview about the major available EA frameworks. Amongst others these are the following:

- Architektur integrierter Informationssysteme (ARIS) [Sc00b, Sc01, ID06, ID09]
- Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance (C4ISR) [C497]
- Computer Integrated Manufacturing Open System Architecture (CIMOSA) [CI09]
- Department of Defense Architecture Framework (DoDAF) [Do07]
- DoD Enterprise Architecture Technical Reference Model (DoD EA TRM) [Do05b]
- Extended Enterprise Architecture Framework (E2AF) [IF06]
- Enterprise Architecture Planning (EAP) [SH93]
- Federal Enterprise Architecture Framework (FEAF) [FC99]
- Generalised Enterprise Reference Architecture and Methodology (GERAM) [IF99]
- Integrated Architecture Framework (IAF) [Ca07a]
- Joint Technical Architecture Version (JTA) [Do05a]
- Ministry of Defence Architectural Framework (MODAF) [Mo08]
- NATO Architecture Framework (NAF) [NA07]
- NATO C3 System Architecture Framework (NC3SAF)

- Purdue Enterprise Reference Architecture (PERA) [PC01]
- Technical Architecture Framework for Information Management (TAFIM) [Do00b]
- Treasury Enterprise Architecture Framework (TEAF) [Do00a]
- Treasury Information Systems Architecture Framework (TISAF) [Do97]
- The Open Group Architecture Framework (TOGAF) [Op09a]
- Zachman [Za87, Za08]

Figure 7.2 shows a time based overview about the previously listed EA frameworks. Blue rectangles show the *current versions* of the frameworks, whereas green rectangles show the *initial version* and yellow rectangles symbolize *intermediate versions*. In contrast to [Sc06a] the figure also shows two kinds of dependencies between the frameworks and their versions. Dashed green arrows show the influence between different frameworks, solid arrows show a predecessor relationship.
Besides the number of available frameworks only a few EA frameworks are considered in practice, typically including Zachman and TOGAF [Bu09d]. Especially during the orientation phase, the Zachman framework is appealing due to its simplicity and its clarity.

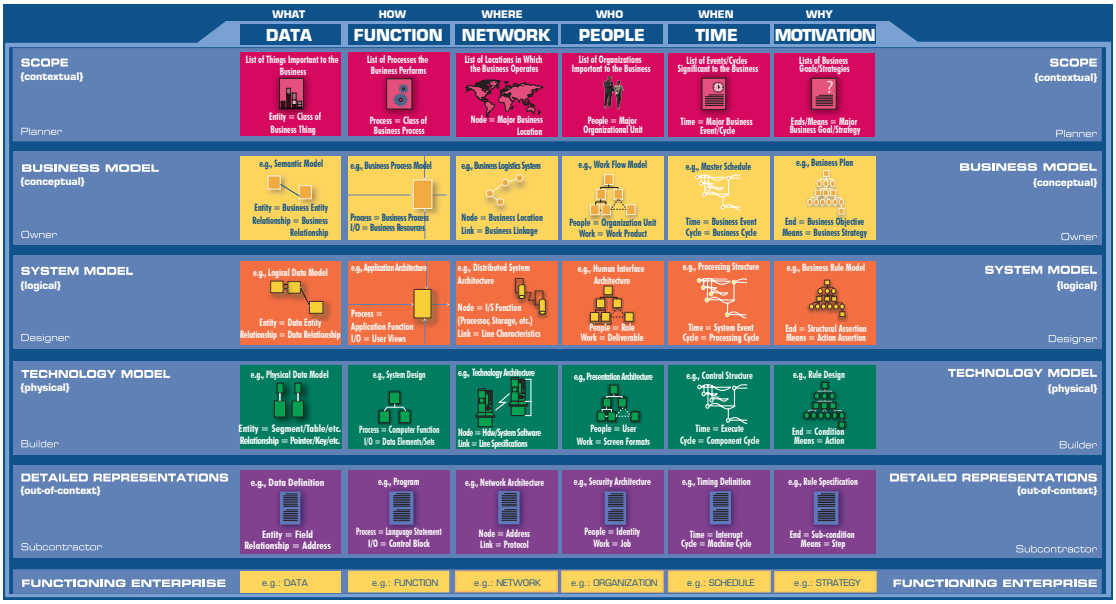

Figure 7.1.: The Zachman framework for enterprise architecture [Za08]

The Zachman framework is the first known framework, which relates to the concept of an EA and was published in 1987 as "a framework for information systems architecture" [Za87]. In [Za97] the Zachman framework is described as being build up on six *abstractions* – What, How, Where, Who, When, and Why – and five *perspectives* – Scope, Owner, Designer, Builder, and Out-of-Context (or, Sub-Contractor) – which have been derived by observations of the descriptive representations of physical products like

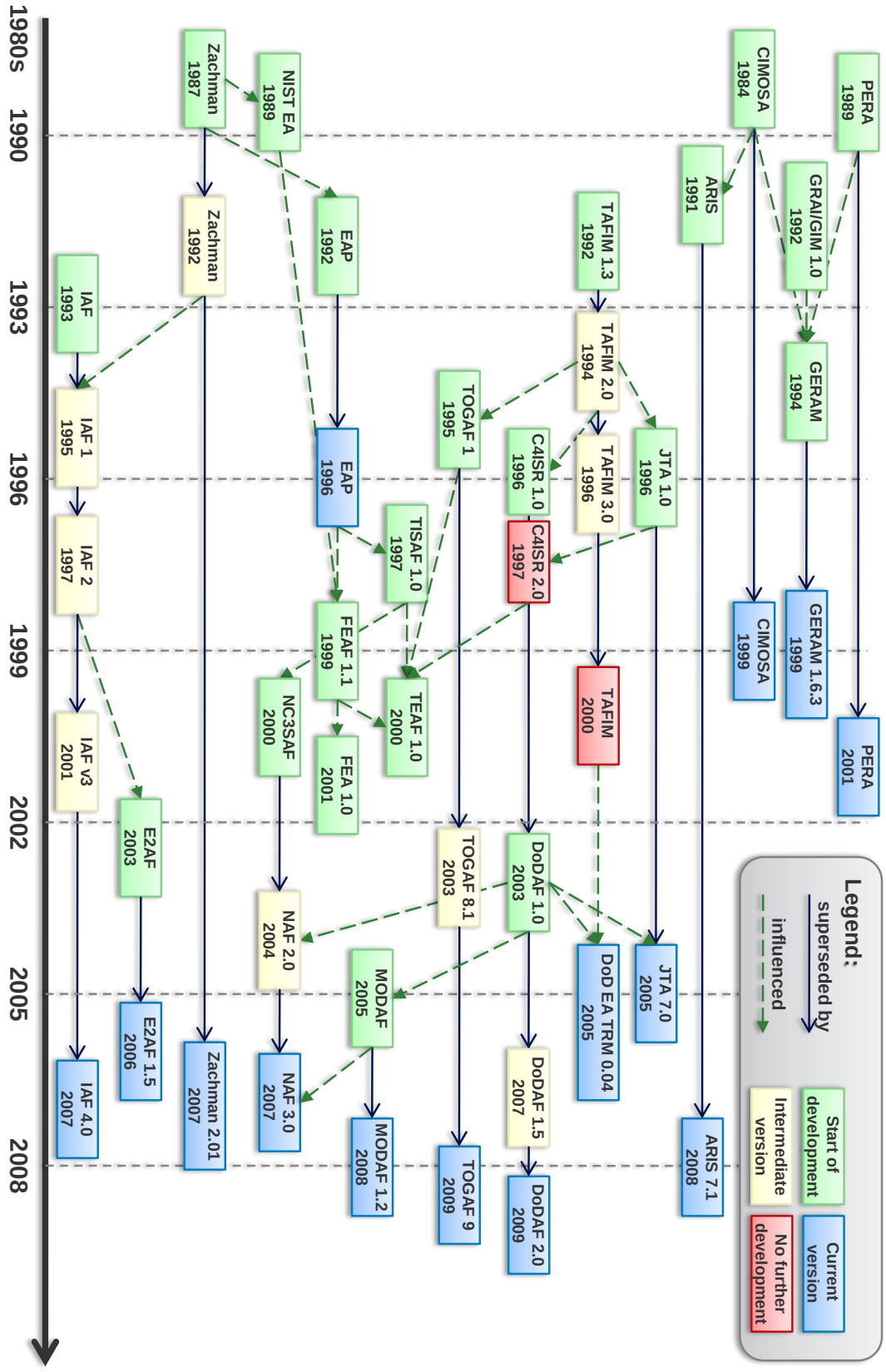

Figure 7.2.: Available EA frameworks as of June 2009 (based on [Sc06a, Le07])

airplanes and buildings. This generic abstraction has then be applied to EA management. The abstractions and perspectives span a matrix of 30 cells (see Figure 7.1) to organize or classify the descriptive representations of the complex elements of an enterprise. Thereby, each cell gives an indication which kind of representation to use, but no specification for the viewpoints or the required information to create the corresponding views is given. The Zachman framework was extended by Sowa and Zachman [SZ92], e.g. by additional information how the information model for the different cells could look like. But these extensions still remained on a level too abstract to be implementable.
The second prominent example for an EA framework is *The Open Group Architecture Framework* (TOGAF) [Op09a] by The Open Group. In February 2009 version 9 of the TOGAF framework [Op09a] superseded version 8.1.1. The new version introduced some additional features to support an enterprise-specific EA management approach:

Modular structure: In TOGAF 9, content that was contained in the resource base in TOGAF 8.1.1, has been modularized to improve usability and incremental adoption.

Content framework: TOGAF 9 includes a content framework with a model of architectural work products for improving consistency throughout the created deliverables.

Extended guidance: TOGAF 9 has been extended by a logical information model and a capability framework with definitions of the organization, skills, roles, and responsibilities.

Architectural styles: This feature provides a set of additional guidelines on how to adapt the process of TOGAF to specific situations like service-oriented architecture (SOA) [KBS05] or how to address a security architecture.

In version 9 TOGAF consists of six main parts. The *Architecture Development Method* (ADM) describes the ten different phases of EA development (see Figure 7.3) as a generic process-oriented method, which is a key strength of TOGAF.
The ADM consists of ten phases: Preliminary, architecture vision (A), business architecture (B), information systems architecture (C), technology architecture (D), opportunities and solutions (E), migration planning (F), implementation governance (G), and architecture change management (H). For more details on these phases see [Jo09].
TOGAF 9 discusses different ways to adapt the method to different *process styles* and to utilization on different *enterprise levels*. These ways are discussed in the *ADM guidelines & techniques* section of the framework. Nevertheless, the adaptation possibilities are limited, as e.g. different ways to organize the enterprise's IT function (cf. [RWR06]) are not alluded to.
The *architecture content framework* is one of the novelties of TOGAF 9. It provides a conceptual metamodel for describing architectural artifacts and deliverables. TOGAF 9 does nevertheless not regard this metamodel to be compulsory and explicitly accounts for the combined usage with other conceptual metamodels, like the Archimate model [La05a].
The *enterprise continuum* provides a model for structuring a "virtual" repository that can be filled with architectural assets and their possible solutions like models, patterns, architectural descriptions, etc. The two main goals of the enterprise continuum are to emphasize reuse and to support communication.

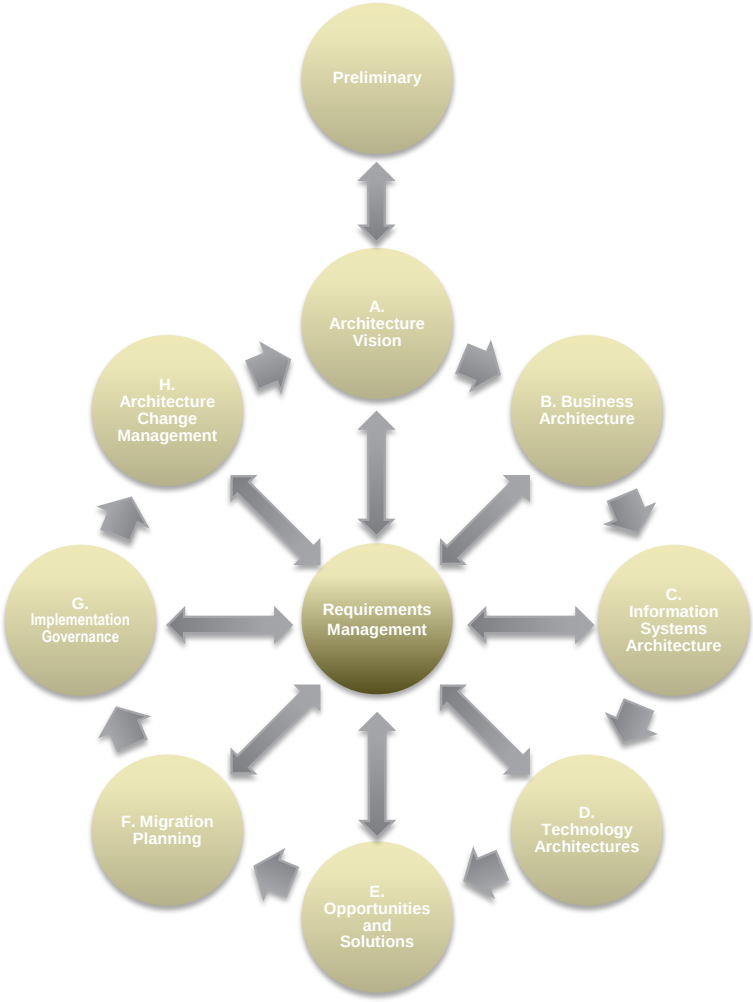

Figure 7.3.: The TOGAF Architecture Development Method (ADM) [Op09a]

The *TOGAF reference models* are divided into the *TOGAF foundation architecture* and the *integrated information infrastructure reference model* (III-RM). The TOGAF foundation architecture comprises a *technical reference model* (TRM) of generic services and functions in order to provide a foundation on which more specific architectures and architectural components can be built. The foundation architecture is embodied in the TRM, which is universally applicable and can be used to build any system architecture. The III-RM helps to address the need to design an integrated information infrastructure with reference designs.

Evaluation of Enterprise Architecture Frameworks

A drawback of the Zachman framework is that it is too abstract to be implemented right away. Although additional information is available further detailing the different cells of the framework, the information is not sufficient to implement an EA management

approach, e.g. because detailed information about the required information model and the utilized viewpoints is not available. In contrast to the mentioned drawbacks, the simplicity is one of the benefits of the framework. It can very well be used to give an introduction to the various aspects of EA management. This is supported by articles and courses provide guidelines on the utilization of the framework.
The main drawback of the TOGAF framework is its extensiveness, which needs a certain amount of familiarization and customization resulting in a reduced applicability. Its extensiveness is a result of the attempt to address all problems with one framework and the attempt to develop a framework which can be used in various industries. Another problem is that EA frameworks typically pursue a monolithic approach. This has been addressed in version 9 of TOGAF by introducing a modular structure. In this version TOGAF also was extended by an information model, which may be used in the implementation of the TOGAF framework. Additionally, the ArchiMate modeling language [Op09b] is more and more incorporated in TOGAF. The defined processes and roles, as well as the available guidelines for utilizing TOGAF are the main benefits of the framework.
Despite the distribution of existing EA frameworks presented in this section their utilization in practice is limited because of either reasons.

Level of abstraction: Some EA frameworks pursue an approach on a high level of abstraction, making it difficult to adopt the framework. An example is the previously described Zachman framework [Za87]. It remains on the level of defining 30 different combinations of abstractions and perspectives, but does not detail those in a way that they can be implemented.

Extensiveness: Trying to establish a generic framework for EA management, which can be used in various companies from different industries easily results in an extensive framework. These are difficult to adapt, to understand, and to implement in a company. Examples for such extensive frameworks are the previously described TOGAF framework, consisting of 778 pages or the NATO Architecture Framework [NA07] with 882 pages.

Thematic focus: Some of the currently available EA frameworks originate from governmental institutions, like military (see e.g. DoDAF [Do07]) or treasury (see e.g. [Do00a]). Due to this, they are focusing on topics, which are of minor importance for non governmental organizations. If such a framework is nevertheless used it is usually not appropriate to address the pain points of the company.

Monolithic approach: EA frameworks tend to follow a monolithic approach (see e.g. [Mo08]) complicating the customization to the demands of a company, as it is not possible to abstain from different parts or modules of the framework. If e.g. a company is not interested in managing their project portfolio in their EA management approach they should be able omit this part of the framework. In a monolithic approach this is not possible with the consequence that a company has to fall back to a "complete or nothing" approach while implementing the framework.

Limited adaptability: This is often tied to the monolithic approach of an EA framework as it is hard to adapt a monolithic approach exceeding a certain size. Additionally often guidelines on how to adapt the framework is missing, see e.g. [Mo08].

Leist and Zellner in [LZ06] present another view on EA frameworks. They use the definition of a method introduced in Section 2.1 to analyze the following EA frameworks: ARIS, C4ISR/DoDAF, FEAF, Model Driven Architecture (MDA), TEAF, TOGAF, and Zachman. Although MDA is not considered to be an EA framework it is still included in this evaluation, because MDA "is an approach to system development, which increases the power of models in that work" [MM03].

	ARIS	C4ISR/DoDAF	FEAF	MDA	TEAF	TOGAF	Zachman
Specification document	●	●	●	●	●	○	●
Information Model	●	◑	○	●	◑	○	◑
Role	○	◑	●	◑	●	○	○
Technique	●	●	○	◑	○	◑	○
Procedure model	○	●	●	●	●	●	◑

Table 7.1.: Evaluation of EA frameworks (based on [LZ06])

The results of the evaluation are shown in Table 7.1. A ● shows that a criterion is *fully accomplished*, ◑ that it is *partly accomplished*, and ○ that it is *not accomplished*. It has to be mentioned that the results are based on the version of the frameworks available in 2006. Therefore, the results may be different analyzing the current versions of the frameworks. TOGAF for example in its latest version 9 includes a metamodel and a description of roles for managing the EA, which should at least result in a *partly accomplished* rating for these constituents. Nevertheless, it shows that none of the evaluated frameworks includes all of the constituents a method is made of. Each of them retains potential for improvement to match the definition of a method.
Besides all recent developments in EA frameworks, it has to be considered that they are only tools for an architect to help him in his daily business. [Mo09b] summarizes this as "architecture frameworks don't make architects". A similar conclusion is given in [Fr09]: The goals that should be achieved by an EA framework are more important then the selection of a framework.

7.1.2. Frameworks in Related Domains

Other domains related to EA management also have developed their own frameworks. Examples for such frameworks are the *IT Infrastructure Library* (ITIL), which includes best practices on IT service management, *Control Objectives for Information and related Technology* (COBIT) [IT07] provides measures, indicators, processes and best practices for the use of IT and for developing appropriate IT governance and control in a company,

and the *Business Process Framework eTOM* [TM09] is an enterprise process framework for the information and communication technologies industry. Even though they have been developed for other domains they may still be a valuable source for inspiration for EA management. As an example for frameworks in related domains this section introduces the ITIL framework.
ITIL is a public framework that describes best practice in IT service management and provides guidelines for the governance of IT. Its primary objective is to ensure that IT services are aligned to business needs, with a focus on the continual measurement and improvement of the quality of IT services delivered, from a business and a customer perspective [Ca07b]. ITIL is made up of five core publications.

Service Strategy [IN07] provides guidance for IT organizations to improve and evolve in a long term perspective. The guidance includes clarification and prioritization of service provider investments in services. Service strategy includes the processes *service portfolio management*, *demand management*, and *IT financial management*.

Service Design [RL07] provides guidance on the design of IT services, processes, and other aspects of the service management effort. Service design includes the processes *supplier management*, *information security management*, *service continuity management*, *availability management*, *capacity management*, *service level management*, and *service catalog management*.

Service Transition [LM07] provides guidance for managing complex changes. Service transition includes the processes *transition planning and support*, *change management*, *service asset and configuration management*, *release and deployment management*, *service validation and testing*, *evaluation*, and *knowledge management*.

Service Operation [CW07] provides guidance for delivery of agreed levels for IT services. Service operation includes the processes *event management*, *incident management*, *request fulfillment*, *problem management*, *access management*, and *service desk*.

Continual Service Improvement [Ca07c] tries to align or realign IT services to changing business needs by identifying and implementing improvements to IT services. Continual service improvement (CSI) includes the processes *7 step improvement process*, *service reporting*, *service measurement*, *return on investment for CSI*, *the business questions for CSI*, and *service level management*.

Those five core components and the enclosed processes form the basis for the ITIL service life cycle shown in Figure 7.4.

Evaluation of Frameworks in Related Domains

Benefits of ITIL are that it is made up of best practices, which have been collected and documented in recent years. The five core documents explicitly mention and detail the concerned stakeholders. In addition all parts of ITIL can be combined to a complete IT service life cycle process, in which some of those processes can be supported by tools, e.g. a configuration management database. ITIL is widely accepted in practice and academia

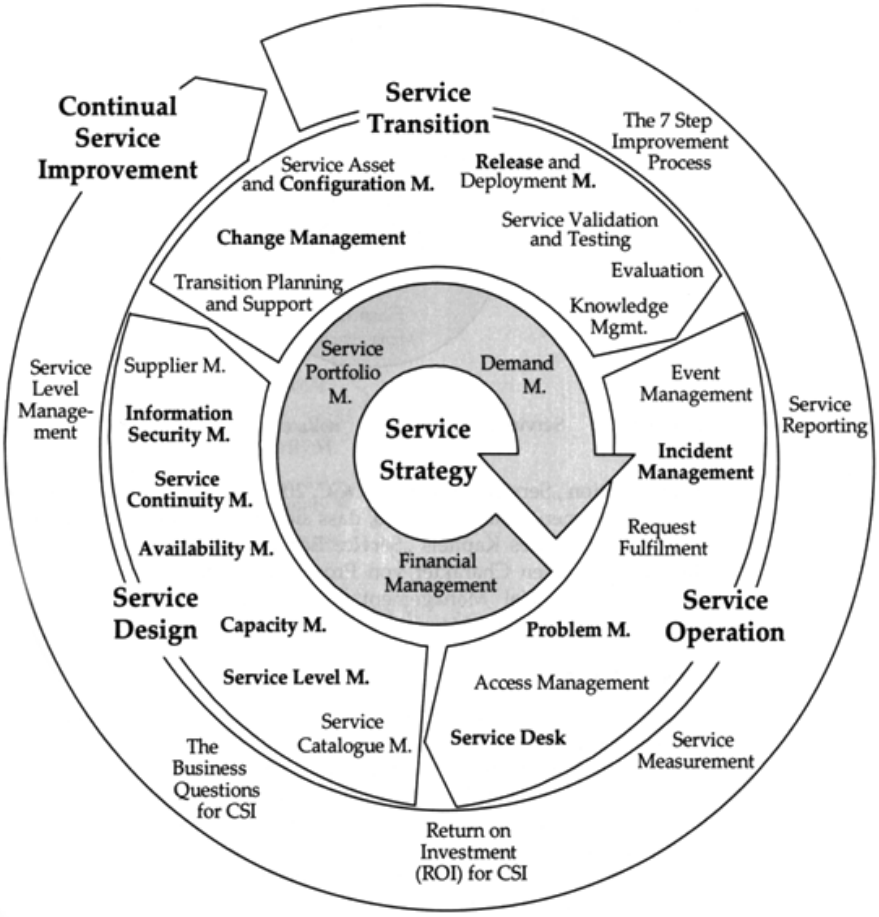

Figure 7.4.: Service life cycle of ITIL v3 [Bu07c]

as an approach to IT service management. Even though ITIL provides a well established process and dedicated roles, it is missing an underlying model and a graphical notation. The major drawback in context of EA management is that the focus of ITIL is not the complete EA but it only focuses on those parts relevant in managing the IT services. Therefore, ITIL has to be complemented by other approaches to constitute a framework for EA management.

7.1.3. Semantic Object Model

The *Semantic Object Model* (SOM) is an object- and process-oriented approach developed for business systems, which can be used for analysis and design. It was proposed by Ferstl and Sinz in [FS90]. SOM is a comprehensive and integrated method for business engineering, based on concepts of systems theory [FS98]. A business system is defined as an open, goal-oriented, and socio-technical system, which uses feedback control and negotiation as its coordination principles. Based on this description SOM can also be applied to EA management.
Typically enterprise models are comprehensive and complex. In order to address this,

the enterprise architecture can be divided in three model layers, which are shown in Figure 7.5.Ferstl and Sinz [FS95] describe theses layers as follows:

Enterprise plan The topmost layer describes an outside view on a business system and focuses on its global tasks and required resources. These global tasks have to be specified. The specification includes the goals and objectives, as well as the products and services to be delivered.

Business process model The middle layer constitutes an inside view on a business system. It specifies main and service processes using two different viewpoints. *Interaction schema* specifies the static structure of a business process and contains objects as well as transactions. The *task-event schema* viewpoint presents the dynamic behavior of a business process by specifying tasks, events, and transactions in their roles as events [Ec04].

Specification of resources The bottom layer is split into three parts, *organization of personnel*, *business application systems*, and *machines and plants*. Personnel, application systems, and machines are considered to be resources, which carry out the business processes. Resources are considered to be independent within the enterprise architecture. Tasks can either be assigned to persons, application systems, or machines, which classifies a task as non-automated or automated. For this layer two viewpoints, *schema of task classes* and *schema of conceptual classes* were specified.

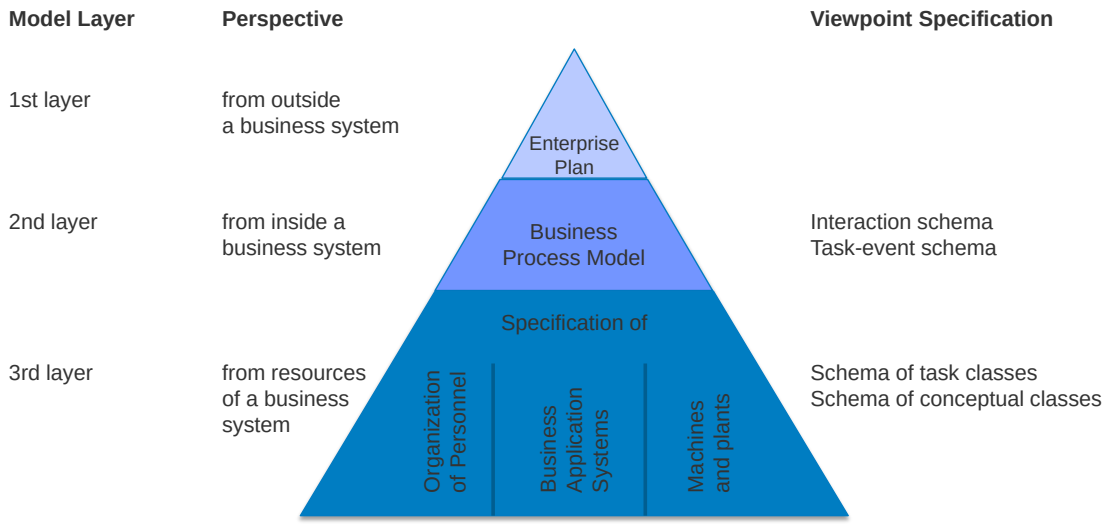

Figure 7.5.: Overview about the enterprise architecture (based on [FS95])

These three model layers describe a business system from a specific viewpoint. Each layer is defined by a metamodel, as well as corresponding viewpoint definitions and patterns, whereby each metamodel is independent of the other metamodels. Figure 7.6 shows the architectural framework of the SOM methodology, which is based on the three models layers of an enterprise architecture shown in Figure 7.5.

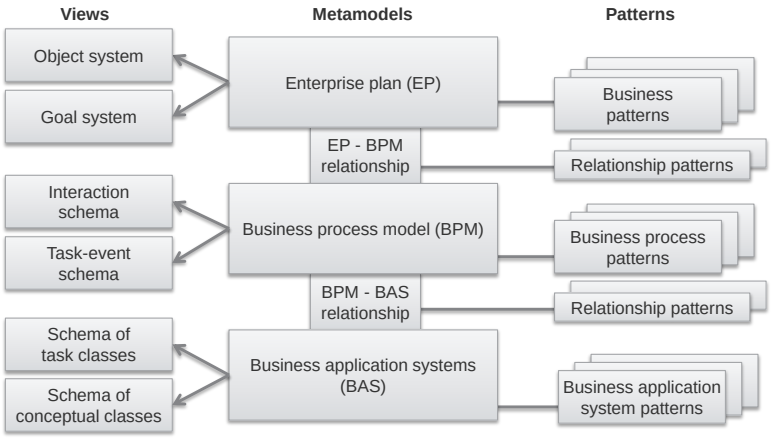

Figure 7.6.: Architectural framework for business system modeling [FS98]

The patterns related to each layer provide "modeling rules and heuristic knowledge applicable to this model layer" [FS98]. Relationship metamodels specify the relationships between different model layers. This may be supported by *relationship patterns* providing transformation rules and additional knowledge. According to Eckert et al. [Ec04] domain-related as well as domain-independent patterns were developed as part of SOM. Examples for such patterns are leasing patterns, the negotiation principle, and the feedback control principle [Fe98, FS01].
In order to offer tool support for keeping the independent models consistent [Fe08] propose an approach for *semi-automated model synchronization in SOM.*

Evaluation of Semantic Object Model

Using SOM typically is a top down approach from the enterprise plan down to business application systems. But other ways to use the SOM architectural framework are possible, e.g. if no elaborated enterprise plan is available it is reasonable to start with the business process model. SOM also defines metamodels and viewpoints for all layers.
Although SOM is available for over ten years now, its influence on practitioners in the context of EA management is limited. Nevertheless, it shows that it is reasonable to use a step-wise hierarchical modeling approach to cope with the complexity of the EA. In addition SOM shows that patterns can be a reasonable approach to document knowledge in managing the EA. In contrast to the pattern-based approach to EA management proposed in this thesis, SOM only uses patterns for documenting modeling guidelines for a certain model layer. Metamodels and viewpoints are not defined in form of patterns.

7.1.4. Systemic Enterprise Architecture Methodology

Wegman in [We03] introduced the concept of *Systemic Enterprise Architecture Methodology* (SEAM), which is based on *general systems thinking* [We75]. SEAM is a systemic, visual approach for systems modeling [Ry07], which allows reasoning about those systems. A system can be any entity that can be seen as a whole or as a composite. As a

result, a market segment, a value network, a company or an IT system can all be modeled as systems. According to [We03], "a system is a set of interacting components. [...] An enterprise is an organization of resources that performs a process. Hence an enterprise is a system in which the components are the enterprise's resources." Therefore, system sciences can be applied to managing the EA.
Based on [We03] the EA can be considered to be a *complex system*, because people are an essential part of the system. The result is a system, which is non deterministic. In contrast, *complicated systems* are systems with a behavior that can be predicted by analyzing the components' interactions.
[We03] claims that, "to address the problems of EA, the architects and the specialists need to make a paradigm shift from the mechanistic paradigm used to understand complicated systems to the systemic paradigm used to understand complex systems".
SEAM is a family of three systemic methods for stepwise design using hierarchical models [BRW03]. *SEAM for business* is a method designed for analyzing the competitive environment of an organization, including its relationships with its customers, partners, and market regulators [We07c]. *SEAM for enterprise architecture* [We07c] adds people, IT systems and software applications to the models of SEAM for business and therefore allows for reasoning about the alignment between business and IT. *SEAM for software* [We05] is a method for analyzing IT systems, software applications, software components, and programming classes. Those methods make up a hierarchy of systems from business to IT resulting in a multi system modeling similar to MEMO (see Section 7.1.5).

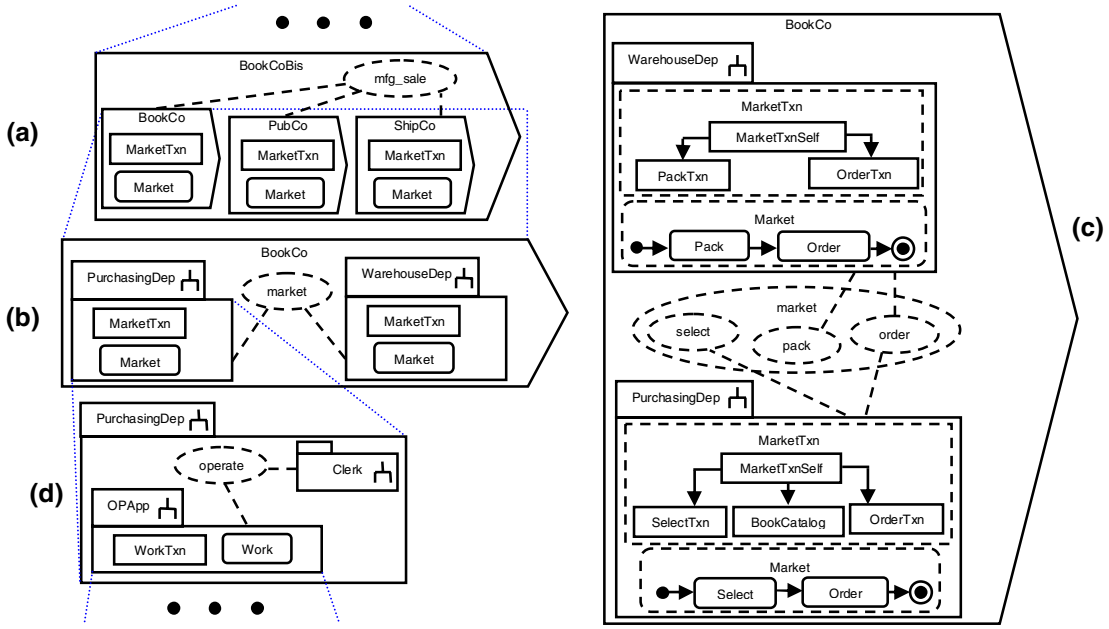

Figure 7.7.: Exemplary SEAM model of an online bookstore [LW06]

As a side effect the approach becomes multi-disciplinary, by using different enterprise models, which are integrated by common concepts. These hierarchical models are defined

by an ontology. The ontology of SEAM provides the concepts required to validate the alignment between the different views of the systems. As these views cover aspects from business and IT, it is also possible to support alignment of business and IT. Specialists from different disciplines can therefore participate in managing the EA. This allows to integrate generic system thinking principles into discipline-specific methods.
According to [We07a] the SEAM modeling ontology is systemic and systematic. Explicating the context in which concepts are defined, the systems boundaries, and the systems life cycles makes the ontology systemic. In addition, it is systematic, because the same concepts are used to represent business systems, as well as IT systems.
The SEAM methods focus on functional analysis and the development of an enterprise model and not the process of building it. They thereby abstains from discussing questions of how to establish and govern the EA management process. As a result any process can be applied for using SEAM.
Figure 7.7 shows an exemplary visualization of a SEAM model of an online bookstore: a) business system organizational level, b) and c) company organizational level at two functional levels and d) department organizational level. SEAM uses a notation similar to UML but includes the capability to represent different kinds of information in a single diagram and provides more contextual information [We07d].
SEAM is based on an hierarchical object-oriented metamodel [LW04b], which is implemented a tool called SeamCAD [We05, LW06]. It can be used to create and verify models complying with the SEAM method [We06]. This tool is also applied in a lecture, which uses SEAM in teaching undergraduate computer science students EA and SOA [We07b].

Evaluation of Systemic Enterprise Architecture Methods

The benefit of SEAM is that different disciplines are related by having common systemic principles. Business, organizational and IT concepts are represented in a systematical way in all disciplines by using a unified notation and the hierarchical modeling ontology. This enables to leverage discipline-specific knowledge by using the vocabulary and the heuristics of each discipline. Offering a tool, which supports to create, simulate, and verify SEAM models is another benefit of the SEAM approach.
A drawback is that SEAM may be too fine grained for EA management, e.g. modeling a persons application for a position as a goal-believe view [We07a] is typically not in the focus of EA management. SEAM is missing a process and dedicated roles to be a method for EA management.

7.1.5. Multi-Perspective Enterprise Modeling

Multi-perspective enterprise modeling (MEMO) [Fr02] is "a method to guide the design and analysis of enterprise models. It is based on a set of modeling languages that allow for creating conceptual models representing various perspectives on an enterprise" [Fr08c].
MEMO offers a framework on a high level of abstraction that allows to structure an enterprise architecture. Figure 7.8 shows this framework made up of three perspectives – strategy, organization and information system – and five aspects – structure, process, resources, goals/success factors, and environment. According to [Fr02] one or more foci

correspond to a particular language, e.g. an organization model serves to represent an organization structure.

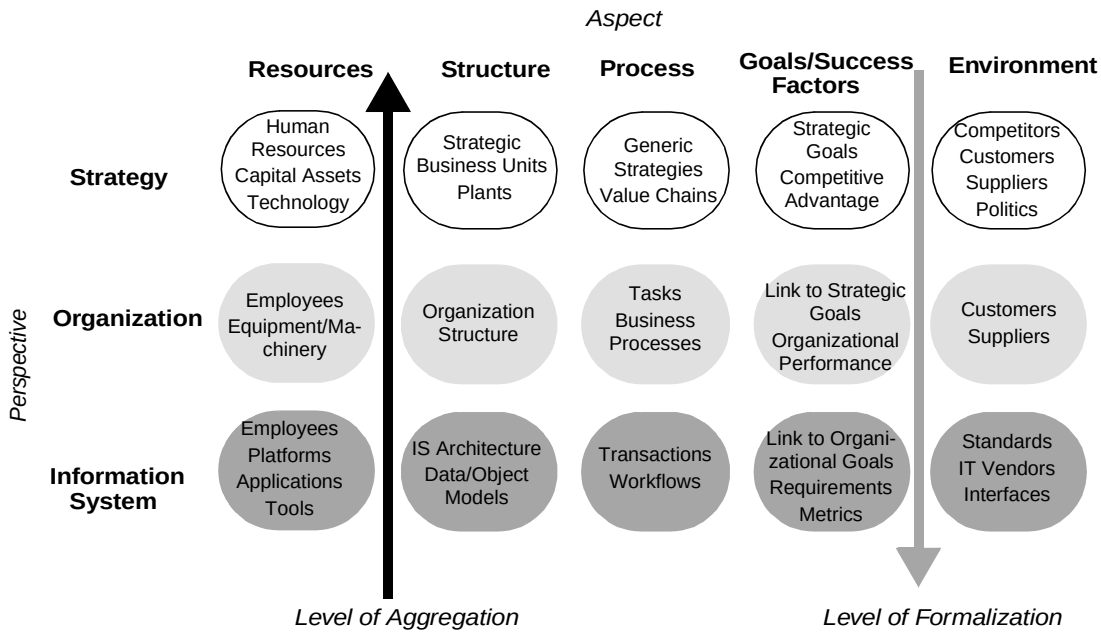

Figure 7.8.: Aspects and perspectives of MEMO [Fr02]

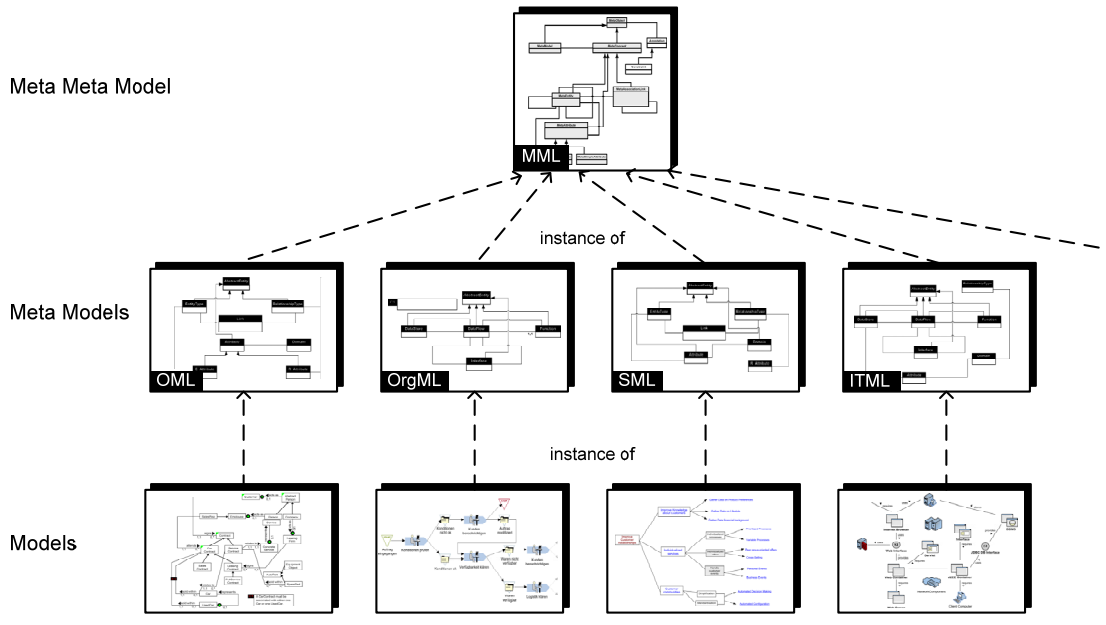

Figure 7.9.: The MEMO language layers [Fr08c]

Each of the languages, e.g. the *IT modeling language* (ITML) [Ki08] for modeling IT related aspects, the SCOREML [Fr08a] for modeling an indicator system, the *Strategy*

Modeling Language (SML) [Fr02] for modeling goals and business strategy of a company, the *Organisation Modelling Language* (OrgML) [Fr02] for modeling a company's organization, etc., includes a metamodel (information model in the terminology of this thesis) and a corresponding model (viewpoint in the terminology of this thesis). Considering specific purposes and requirements for modeling different aspects of the enterprise is supported by offering distinct languages. Another benefit is the reduction of complexity by partitioning EA in different aspects [Fr08d]. A similar approach is pursued by the SEAM (see Section 7.1.4).
In order to integrate the different languages they need to have concepts in common. This can be ensured by using a common meta-metamodel. Relationships between the different languages – with their metamodels and models – and the common meta-metamodel can be seen in Figure 7.9. Based on this common meta-metamodel [Fr08c] proposes a tool support for MEMO, called *MEMO Center*.

Evaluation of Multi-Perspective Enterprise Modeling

The benefit of MEMO is its focus on special languages, for special purposes in managing the EA, which all use a common meta-metamodel facilitating integration. This is an aspect MEMO and SEAM have in common. Graphical notations for the different languages were also documented in MEMO.
A drawback of MEMO is that the languages are not complemented by processes or process steps indicating how to use the languages in order to achieve higher level goals.Frank et al. [Fr08a] indicate that the languages have to be embedded in an existing enterprise modeling method to address this drawback.

7.1.6. Business Engineering

Using principles from well-established engineering disciplines in EA management was already introduced in Section 2.1. This would improve the chance to create cost-effective solutions to practical problems.
Aier et al. [Ai09] propose an engineering based approach to enterprise architecture. Based on [Sh90] four aspects of engineering are introduced. Engineering develops *cost efficient solutions* for *relevant problems* by *using scientific knowledge* in *service to society.*
In the approach, classical engineering disciplines are analyzed for those aspects in order to derive characteristics, which can be used to propose requirements for an engineering based approach to managing the EA. Four characteristics are identified and are given in the following. The derived requirements for EA management are given in italics.

Engineering knowledge patterns Construction routine is more common in engineering then innovative construction. Therefore, it is important to collect and organize knowledge, e.g. in a handbook, etc. to make it available for less experienced engineers. *Designing EA should integrate and aggregate existing knowledge.*

Standardized construction plan and construction language In mature engineering disciplines high level construction plans are common. They show the main components and the relationships required to achieve a desired behavior. Every engineering discipline has developed standardized languages, which are used to create the

construction plans. These languages are even part of engineering education as a means for communication, because communication is regarded to be important.
EA description is the central construction plan. Conventions have to be used in its development.

Division of labor Constructing complex systems with larger teams is only possible, if they are divided into smaller components, which are constructed by separate teams and assembled afterwards.
Governance is vital to enable distributed and consistent design and evolution of the EA.

Architectural design Designing an architecture is based on transforming requirements into high level blueprints of the system. These transformations are very important, because fundamental design decisions have to be made. This is the responsibility of experienced and qualified engineers.
EA implies the organization's capability to quickly fulfill demands and to exploit business potentials of IT innovations.

The proposed solution to these requirements is the so called *Business Engineering Navigator* (BEN), which structures the components of the engineering support of EA management. The basis of BEN are "generally valid and accepted components of design and analysis knowledge" [Ai09], which have to be adapted, extended, and integrated based on the context that they are applied in. Figure 7.10 shows these components and their mapping to abstraction layers.

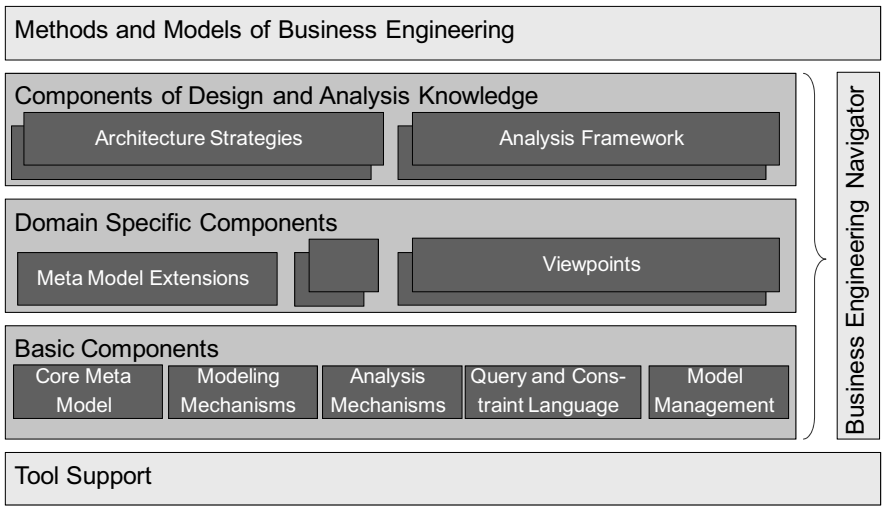

Figure 7.10.: Architecture of the Business Engineering Navigator approach [Ai09]

Tool support is available with *ADOben* [Ai09] which implements the basic and the domain specific components. ADOben is based on the commercial modeling tool and metamodeling platform ADONIS.
The BEN approach includes the following three layers. *Basic components* consist of five elements, which are domain independent and are used to model, analyze and design the

EA. A core metamodel defining the core artifacts and their relationships, which serves as a standardized construction language. The term core metamodel corresponds to the term information model used in this thesis. [Ös07] gives an overview about this core metamodel modeling mechanisms defining a domain independent description language to create models of design artifacts. Analysis mechanisms defines generic analyses and analysis mechanisms like dependency diagrams. A query and constraint language is based on formalized modeling mechanisms like relational algebra and is used to specify and verify architecture strategy and principles. Model management cares about life cycle management issues for models, like model history, etc.
These basic components are complemented by *domain specific components*, which are instances of the basic components. Metamodel extensions augment the core metamodel to specific contexts, like certain industries. Viewpoints use the generic analysis mechanisms and queries introduced before.
Components of design and analysis knowledge is the last part of BEN and includes components for documenting engineers' knowledge. Architecture strategies can e.g. be documented by using the concept of patterns. The analysis framework includes measures and metrics to analyze the design artifacts.
Methods and models of business engineering is the topmost layer and is based on the Business Engineering Framework [Wi03], which defines the different layers of business engineering and introduces methods to manage them.

Evaluation of Business Engineering

A benefit of the business engineering approach is its holistic view on EA management, which can be adapted to context and situation-specific demands. The approach is complemented by a tool customized for supporting the requirements of the approach.
The core metamodel [Ös07] is one drawback of the approach. It includes 46 classes and follows a monolithic approach, which makes it difficult to adapt and extend. In addition the used concepts are not further detailed, e.g. by a glossary. As mentioned before there is no standardized and widely accepted metamodel for EA management. The same is true for the core metamodel. A reason for this may be the requirements for such a model, which vary widely between different companies and are therefore difficult to address with a monolithic metamodel.
Kurpjuweit and Winter [KW07] mention that "innovative engineering approaches will address increasingly complex artifacts (work systems instead of information systems) by means of models that simultaneously express multiple crosscutting stakeholder concerns. Consequently, also the applied modeling concepts and meta models will be more complex and should be constructed systematically". "Furthermore, there is no systematic support for the selection, adaptation, and integration of such meta models" [KW09]. In addition the criteria of width, depth, and pragmatism has to be considered during the construction of a metamodel [Ai09]. This leads to the conclusion that there is no one size fits all metamodel.
Kurpjuweit and Winter in [KW07] propose a possible solution for this problem by systematically eliciting the concerns and information model needs of the concerned stakeholders by partitioning their requirements in viewpoints. They call this approach *concern-oriented business architecture engineering*. For each viewpoint a metamodel fragment is

developed which can be integrated into a comprehensive metamodel. In [KW09] Kurpjuweit and Winter state that the solution they proposed in [KW07] can benefit from reference metamodels that serve as a starting point. Aier et al. [Ai08a] present three case studies, where the approach was validated. The approach is extended in [La09b] by a *causality-based approach* to create a best-of-breed method, which introduces causal relations between attributes. These relationships can be used for analysis and planning. Viewpoints in the context of EA management were specified [Ai08a], but these are not publicly available, which may be another drawback, as this restricts the future development and the usability of the method. The same is true for the documented information models, except the core business metamodel [Ös07].

7.1.7. Quasar Enterprise

In 2004 Siedersleben published *Quasar* [Si04b]. Quasar defines reference architectures, rules and principles used to build business information systems at sd&m, nowadays Capgemini sd&m [Ca09b].
Analogous to Quasar, *Quasar Enterprise* [En08] was developed as a reference for designing application landscapes in an service-oriented way. Quasar Enterprise proposes an approach to derive an application landscape from a business architecture and to identify and design the required services. In addition, Quasar Enterprise shows how to realize these services using appropriate infrastructures. This is done by predefined building block's like methods, rules, patterns, and reference architectures.
In order to achieve the previously described goals Capgemini sd&m [Ca09b] introduced a framework, which is based on the *Integrated Architecture Framework* (IAF) [Ca07a] developed by Capgemini. Figure 7.11 shows the structure of the IAF.

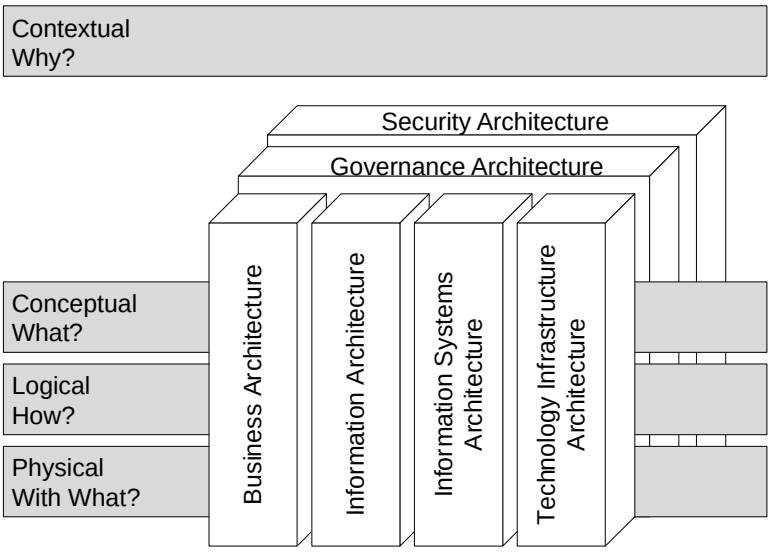

Figure 7.11.: Macro-structure of the Integrated Architecture Framework [Ca07a]

The IAF is made up of six aspect areas. Four of them make up the basic architecture and are structures similar to the key components introduced in [ME02].

Business architecture structures the business processes and business services in order to meet the business goals and reflects the organization of the enterprise.

Information architecture structures the information required in the business architecture.

Information systems architecture structures the application landscape from a business perspective.

Technology infrastructure architecture structures the used technical platform and system software components.

These basic architecture areas are complemented by the following two extended aspects areas.

Security Architecture cares about efficient structures for establishing secure and reliable environments.

Governance Architecture cares about efficient structures for establishing an adaptable environment, which can also be operated.

In addition the architectural layers *contextual*, *conceptual*, *logical*, and *physical* are introduced. The four basic architecture areas and the four architecture layers make up a matrix, which is used to define a roadmap based on the IAF to answer the questions of the enterprise architect.
Quasar Enterprise aggregates business architecture and information architecture to *business architecture* due to its focus to application landscapes. The resulting framework is shown in Figure 7.12.

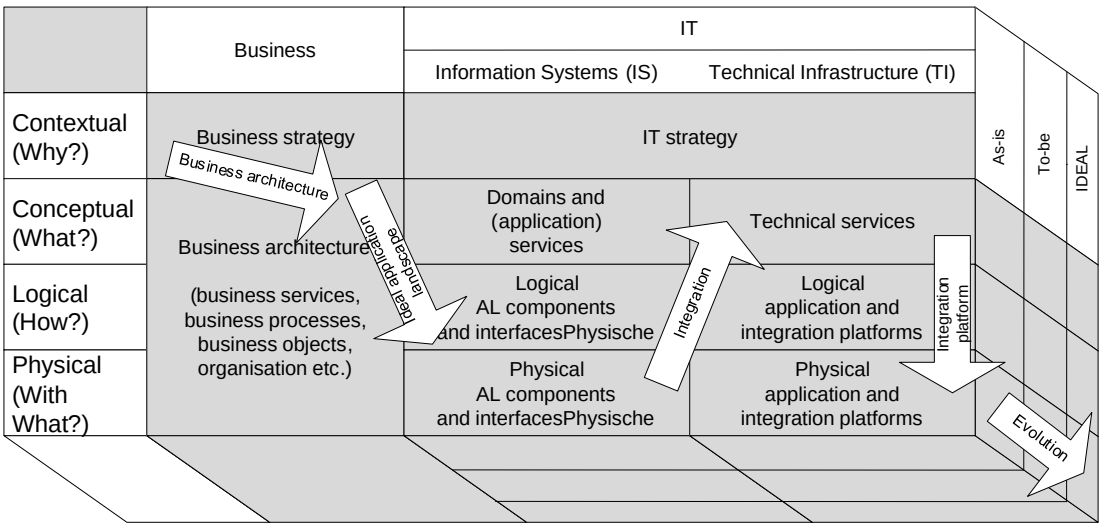

Figure 7.12.: Overview about Quasar Enterprise [En08]

Quasar Enterprise extends the layers and aspects introduced by the IAF by a time dimension used for planning the enterprise architecture. Thereby, the dimensions *as-is*, *to-be*, and *ideal* are distinguished. This is a similar concept like current, planned, and target landscapes as introduced in [Bu09c].
The white arrows in Figure 7.12 are similar to the roadmaps in IAF, except that they are complemented by building block like methods giving guidance on how to pursue a certain goal, e.g. business architecture, ideal application landscape, etc.

Evaluation of Quasar Enterprise

Quasar Enterprise is an approach mainly known in Germany and German-speaking countries, due to the language of the currently available publication. It was developed based on consulting projects by Capgemini sd&m, but it is also getting attention in practice.
The main benefit of Quasar Enterprise is the holistic and service-oriented view on EA management. This is complemented by processes and roles documenting how a service-oriented EA can be accomplished. In addition the building block like structure offers the possibility to only select the required parts of the approach. Another benefit is the use of Quasar Enterprise is exemplified by an introductory story, simplifying the access to the approach.
Drawbacks are the limited support for selecting the required parts in order to customize the approach to own demands and that viewpoints and information models for EA management are not explicated in a way that they can be used without adding further detail. Besides these potentials for improvement, Quasar Enterprise is built upon the knowledge gained in consulting projects. This may lead to a method for EA management, which is not generally applicable.

7.1.8. Ultra Large Scale Systems

Ultra Large Scale Systems (ULS) [No06] pursue a similar approach to EA management like SEAM. Both rely on a system view on EA management. ULS is not yet a research project or a method to manage very large systems of systems. The term ULS was coined by a study conducted by the *Software Engineering Institute* at the *Carnegie Mellon University* for the *U.S. Office of the Assistant Secretary of the Army (Acquisition, Logistics, and Technology)* about the software challenges of the future. The final report [No06] does not present an approach on how to manage such ULS systems, but analyzed the topic for upcoming research areas.
Ultra large scale systems are very large systems, which are built from smaller systems. Examples for such systems are the Internet, the networks of energy providers, etc. This can be seen as a natural continuation of the idea to model and manage the architecture of an enterprise, which by itself is a system of systems (e.g. organizational units are made up of sub units, or business applications are made up of components, etc.) but with a smaller scale.
[Ha05] also states a relationship between systems of systems and enterprise architecture. "The system of systems concept in particular is very important to the emerging discipline of enterprise engineering and related initiatives to develop and maintain enterprise architectures".

The sheer scale of ULS systems is the main problem in their management. Northrop et al. [No06] characterize ULS systems to be typically operated and controlled in a decentralized way. People are an essential part of these systems and the boundary between people and systems is eroded. Such systems have inherently conflicting, unknowable, and feature diverse requirements. ULS systems are in continuous evolution and under deployment. They are made up of heterogeneous systems, which are inconsistent, and have constantly changing elements. In ULS systems failures are a common thing. Additionally new paradigms for acquisition and policy are required.
Based on these characteristic, the report derives three challenges for ULS systems: Design and evolution, orchestration and control, and monitoring and assessment. To address those challenges in the future [No06] propose the following research topics:

Human interaction People are an essential part of ULS systems and are one possible source for failures. This requires to research on user-centered specifications and on modeling users and user communities.

Computational emergence This area of research tries to apply methods and tools based on economics and game theory (e.g., mechanism design) to balance the interests of the participants of the ULS systems in order to ensure global optimal system behavior. Such methods and tools are required as at least some participants of the system will behave opportunistically.

Design Current design theory, methods and tools are not sufficient to design ULS systems effectively. Like in some EA management approaches organizational, social, and economic aspects are not adequately represented. Nevertheless, EA management may be a discipline to contribute experience in this area.

Computational engineering In ULS systems users and developers come from different backgrounds and have different objectives. Therefore, ULS systems will be defined in different languages with different abstractions and semantic structures. This requires new approaches to intellectual control, like research on expressive representation languages, to accommodate this semantic diversity

Adaptive system infrastructure ULS systems require an infrastructure, which allows distributed organizations to develop, deploy, and evolve system components in parallel in distributed locations. Research in this area has to investigate and develop integrated development environments able to cope with this situation.

Adaptable and predictable system quality Failures and attacks are common in ULS systems. In addition the system's subsystems are operated for a long time and are essential for the success for the whole ULS system. Therefore, it is required to measure and maintain quality in ULS systems, especially concerning continuous change, ongoing failures and attacks.

Policy, acquisition, and management Suppliers and supply chains are also intrinsic and essential components of an ULS system. This requires to develop and automate rules and policies to enable fast and effective local action while preserving global capabilities.

As a recommendation to address these research challenges, the report [No06] suggests a roadmap for a future ULS research program. Various discussions, like a panel at OOPSLA 2008 conference [Fr08b] were initialized and are an indication for a community beginning to grow. This shows the importance of this future research topic, which is closely related to EA management and in which the methods developed for EA management may be reused and may contribute to solve problems of ULS systems.

7.2. Comparison to existing Enterprise Architecture Management Approaches

The previous section presented nine existing EA management approaches. They are subsequently compared to the pattern-based approach to EA management proposed in this thesis analogous to the comparison by Leist and Zellner [LZ06] shown in Table 7.1. The assessment criteria are derived from the constituents of a method introduced in Section 2.1: Underlying model, language, processes and roles, and guidance. Like in the comparison of Leist and Zellner [LZ06], a ● shows that a criterion is *fully accomplished*, ◑ that it is *partly accomplished*, and ○ that it is *not accomplished*. Partly accomplished in this case means that some information, e.g. about the underlying model is available but it is not sufficient to start working with it right away.

	Zachman	TOGAF	ITIL	SOM	SEAM	MEMO	Business Engineering	Quasar Enterprise	EAM Pattern Catalog
Underlying model	◑	◑	○	●	●	●	●	◑	●
Language	◑	◑	○	◑	●	●	◑	◑	●
Processes and roles	◑	●	●	○	○	◑	●	●	●
Guidance	●	●	●	●	●	●	●	●	◑

Table 7.2.: Comparison of the pattern-based approach to EA management to existing EA management approaches

The results of the comparison are shown in Table 7.2. ULS systems are not included in this comparison as they constitute no existing EA management approach but give an indication about a future research topic and a possible future application area for EAM patterns and EAM anti patterns.
TOGAF and Zachman are the two EA frameworks introduced in the previous section. The Zachman framework is rated to partly accomplish the underlying model, language, and process and roles criteria. This can be justified by the available information about the required information model and viewpoints. Both are indicated but not available in a detailed form. In addition there is no process specified, except that cells in the

upper level of the matrix are used before cells in the lower levels, which feature a higher level of detail. Roles are also not included in the framework. Guidelines are available in form of articles and training material on the Zachman framework resulting in a fully accomplished rating.
Compared to the Zachman framework, the TOGAF framework is rated better concerning the available information about processes and roles, which are one of the main benefits of the TOGAF framework. Guidelines are available in form of the framework itself and a certification program. Therefore, they are rated to be fully accomplished. An underlying model and a language are introduced in TOGAF 9, although they are not extensive and integrated enough for a fully accomplished rating.
The ITIL framework, as a prominent example from a domain related to EA management, provides a well established process for IT service management, complemented by dedicated roles. Various books, courses, and a certification program is available for providing guidelines for applying the method. Therefore, both criteria are rated fully accomplished in this comparison. Providing a language and an underlying model is not in the focus of ITIL. An underlying model is considered to be important, e.g. in form of a configuration management database, but is not addressed in the framework. Due to these reasons the criteria underlying model and language are rated to be not accomplished at all.
SOM offers an underlying model resulting in a fully accomplished rating and indicates a graphical language resulting in a partially accomplished rating. Processes and roles are not in the focus of this EA management approach and are therefore rated to be not accomplished. In contrast guidance for applying the method is available in form articles and books, fully accomplishing the criteria.
A well documented underlying model and a related modeling language is available for SEAM and are rated to be fully accomplished. The same is true for guidance for applying the method. Various articles are published and the method is taught in a lecture. A process and required roles are not documented in SEAM, resulting in a not accomplished rating.
MEMO's strength are its special purpose languages for managing the EA. They include an information model and a graphical notation. In addition, guidance for applying the method is available by different publications. What is missing in MEMO are processes and roles helping the user implementing the method. For these reasons MEMO is rated fully accomplished for the underlying model, language and guidance. Processes and roles are rated to be not accomplished.
BEN offers the core business metamodel, processes and roles, and guidance for applying the method. This is, why they are rated to fully accomplish this criteria. The language used for creating visualizations is not publicly available resulting in a partly accomplished rating. Similar is true for the models underlying the visualizations, but this is not accounted for, because the core business metamodel is available.
Processes and roles, as well as guidance for applying the method are documented in Quasar Enterprise and are therefore rated to be fully accomplished. The underlying model and the language are indicated but not documented in a way to be used in an implementation of this method. For this reason these two criteria are rated to be partly accomplished.

In contrast to the existing EA management approaches introduced in the previous section, the pattern-based approach to EA management is developed to include all constituents of a method: M-Patterns document processes and roles, V-Patterns providing a graphical language or a technique, and I-Patterns provide a metamodel. Therefore, these criteria are rated as fully accomplished. Guidance for applying the method currently is primarily available in form of case studies. These may be used to develop additional guidelines in the future. This results in a partially accomplished rating.
In order to complement this comparison an online survey is conducted, which is presented in the following Section.

7.3. EAM Pattern Catalog Online Survey

This section presents the EAM Pattern Catalog online survey and its results. It included 20 questions and was divided into three main parts: General questions (see Section 7.3.1), questions about the utilizations of the EAM Pattern Catalog (see Section 7.3.2), and concluding questions (see Section 7.3.3). The survey was conducted between 2009-05-11 and 2009-05-31 resulting in a 20 day response period.
Email invitations to participate in this online survey were sent to known contacts of the System Cartography project (459 contacts) and to the at that time registered users of the EAM Pattern Catalog Wiki (32 members). Additionally, announcements have been posted in the communities *Anything Enterprise Architecture* (`http://enterprisestewards.ning.com/`), *LinkedIn* (`http://www.linkedin.com`), and XING (`https://www.xing.com/`). The announcement of the online survey and the reference to the EAM Pattern Catalog Wiki had the interesting side effect that the number of registered EAM Pattern Catalog community members increased to 236 (as of 2009-06-10) within three months.
72 participants from all over the world responded to the call for the survey. Based on the number of contacts and the registered users this leads to a response rate of around 15%. The answers of the participants will be presented in the following sections.

7.3.1. General Questions

The first group of questions included introductory questions about the personal background of the survey participant and their impression about the support for enterprise architecture management (EAM) currently available in EA frameworks and literature.

7.3.1.1. Question 1: Which industry do you belong to?

Participants could select from the following list of options:

- Aerospace/Defense
- Consulting
- Education (e.g. Professor, Ph.D. Student, Student, etc.)
- Finance (e.g. Banking, Insurance, etc.)
- Government
- Healthcare
- IT Consulting
- IT Products & Tools
- IT Services
- Manufacturing (e.g. Automotive)
- Telecommunication
- Transportation/Logistics
- Other (free text)

The results for this question are shown in Figure 7.13.

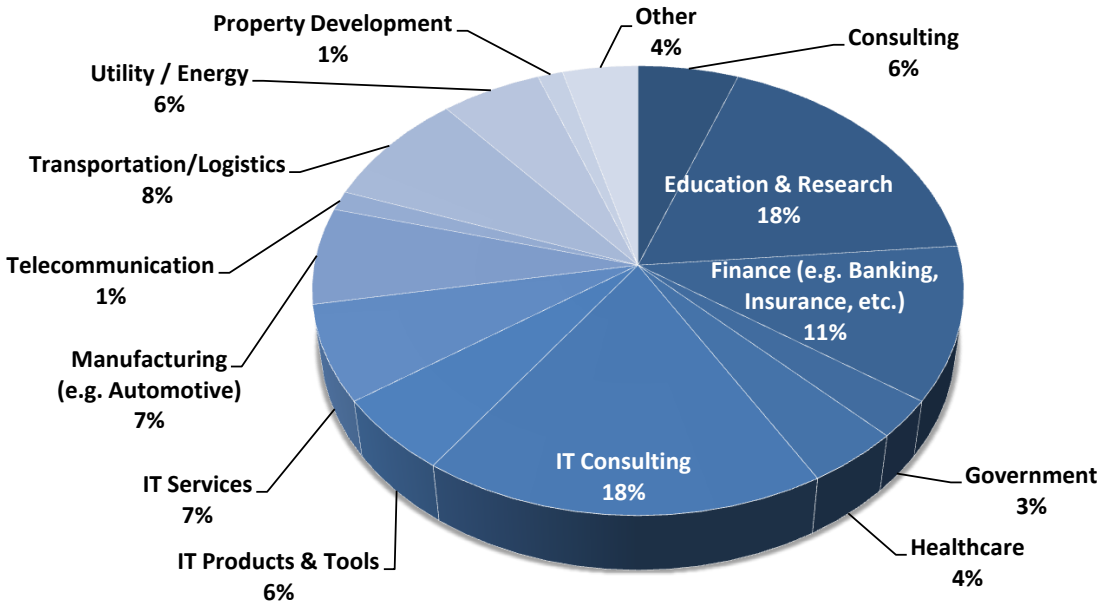

Figure 7.13.: EAM Pattern Catalog pervasion: by industry

The largest groups of participants came from *education and research* (18%) and *IT consulting* (18%). This result matches our experiences concerning requests about the EAM Pattern Catalog. The third largest group of participants is from the *financial industry* (11%), what can be explained by the fact that the financial industry heavily relies on its IT. What is interesting is that the number of participants from the *telecommunication industry* (1%) was low compared to the importance of IT for this industry.
The group of participants, which selected *other* (4%) as their answer gave the following responses: *Travel/hospitalities*, *I do not belong to an industry*, and *no answer*.

Summarizing the results to this question it can be said that the participants came from various industries, with a special focus on industries, which heavily rely on IT or which are into IT consulting.

7.3.1.2. Question 2: Where do you come from?

The second question of the online survey analyzed the world-wide distribution of the survey participants interested in the EAM Pattern Catalog.
Figure 7.14 shows that over 54% of the participants originate from *Germany*, which is not surprising because the EAM pattern approach was developed in cooperation with european companies, with a special focus to Germany. The other countries, which achieved more then 5% of the participants, are the *Netherlands* (10%), the *United States of America* (USA) (10%), and *Switzerland* (7%). The high results for Netherlands and Switzerland can be explained by the same reason like the result for Germany. By contrast the high number of participants from the USA could not be expected. The same is true for countries like *Singapur* or *India*, which have not yet been in focus during the development of the EAM Pattern Catalog.

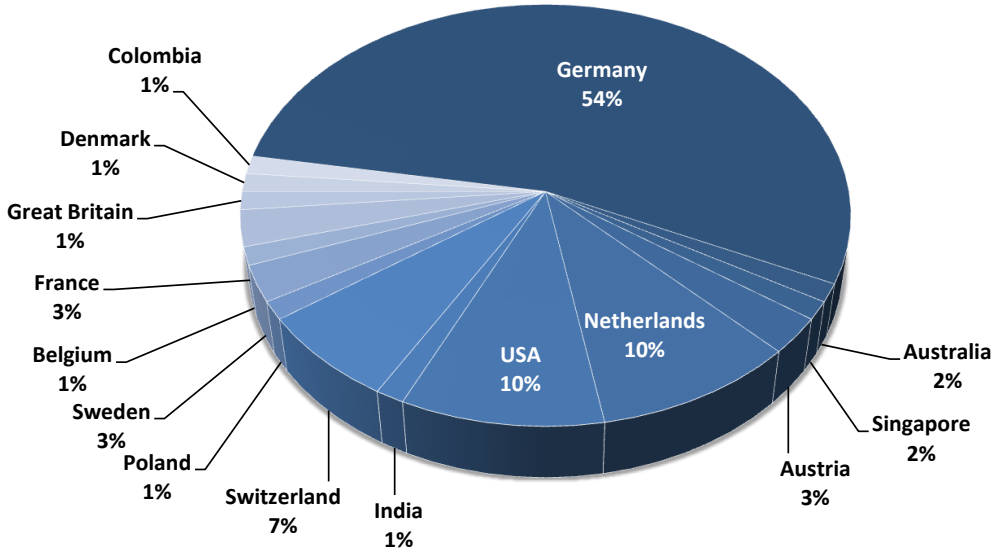

Figure 7.14.: Origin of the survey participants

Figure 7.15 shows the world-wide distribution of survey participants in Google Earth[1].

[1] This image can be accessed at `http://www.29travels.com/getmap.php?j=ATAUBECHCODEDKFRGBINNLPLSESGUS&k=&hm=&f=&c=ccff6633&cf=&w=1020&h=700&pd=none`.

Figure 7.15.: World-wide distribution of participants shown in Google Earth

7.3.1.3. Question 3: Are you using the EAM Pattern Catalog?

Question number three gives indications towards a broad interest, irrespective the fact that the EAM Pattern Catalog was not actively promoted. It evaluated how many people are using the EAM Pattern Catalog. Thereby, one of the following options could be selected by the participants:

- I plan to use it
- I have looked at it briefly
- I am using it
- I passed it on to a colleague who is using it
- Other (free text)

The results for this question are shown in Figure 7.16.
36% of the participants *are using* or *are planning to use* the EAM Pattern Catalog and 53% have *looked at it briefly*. Although promoting of the EAM pattern approach, up to this survey, was limited to presenting at conferences and publishing articles, mainly for the academic community, the results indicate that the EAM Pattern Catalog is a recognized alternative to existing frameworks and books on EA management. The comparison to other approaches on EA management is further analyzed in the following sections.

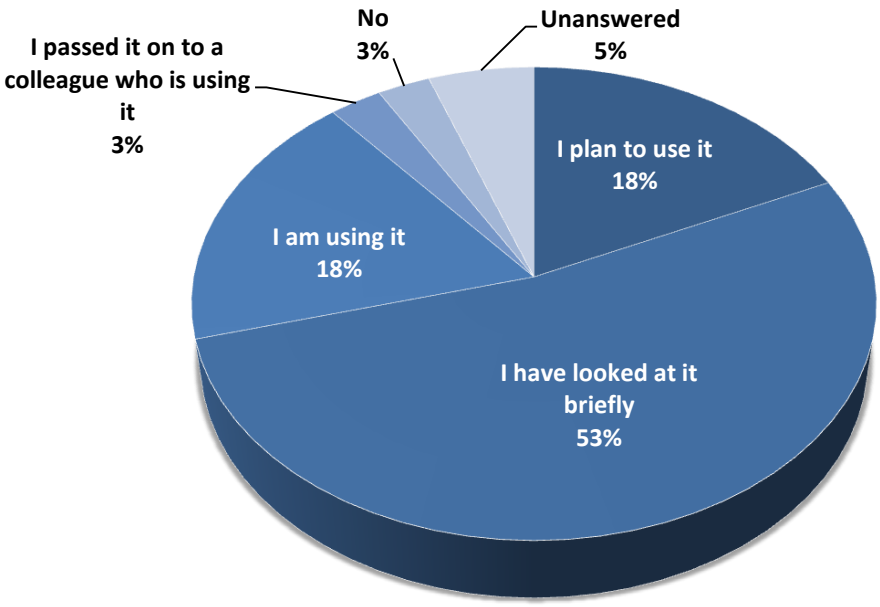

Figure 7.16.: EAM Pattern Catalog usage

7.3.1.4. Question 4: How useful are the following approaches for identifying and supporting stakeholders in the context of EAM in your organization?

Question number four analyzed available EA management approaches concerning their help in identifying and supporting stakeholders.
Participants could rate the following EA management approaches on a scale from *1=No help* to *5=Fulfills all needs*, whereas it was also possible to abstain from rating one or more approaches.

- The Open Group Architecture Framework (TOGAF)
- Zachman Framework
- Department of Defense Architecture Framework (DoDAF)
- Treasury Enterprise Architecture Framework (TEAF)
- NATO Architecture Framework (NAF)
- EAM Pattern Catalog
- EAM Books
- Reference Models (e.g. eTOM)

In addition to the question, participants were given additional information what was meant by EAM books: Examples for books on EAM are Enterprise Architecture at Work: Modelling, Communication and Analysis (Lankhorst 2005) [La05a], Enterprise Architecture As Strategy: Creating a Foundation for Business Execution (Ross, Weill,

Robertson 2006) [RWR06], From Enterprise Architecture to IT Governance (Niemann 2006) [Ni06], and Enterprise Architecture Good Practices Guide: How to Manage the Enterprise Architecture Practice (Schekkerman 2008) [Sc08].
The same choices and hints were also available for the following five questions (see Sections 7.3.1.5, 7.3.1.6, 7.3.1.7, 7.3.1.8, and 7.3.1.9).

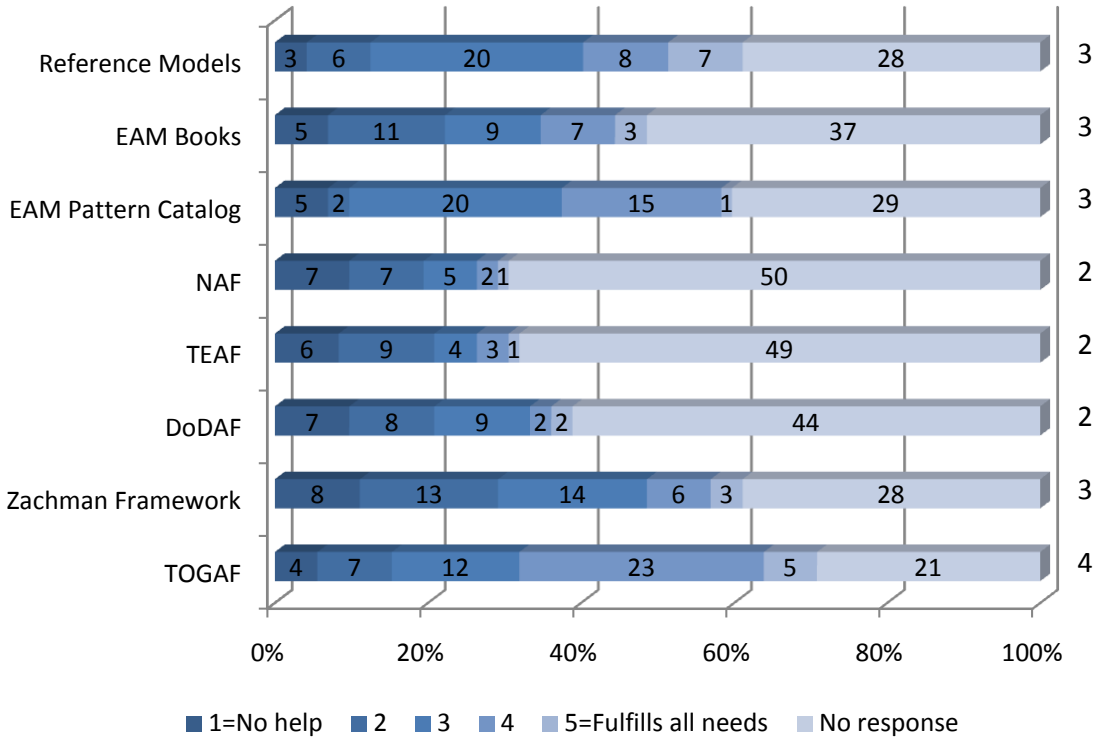

Figure 7.17.: Usefulness for identifying and supporting stakeholders

The diagram in Figure 7.17 shows the results of the question. It shows how many participants selected which rating for which approach, including that some participants did not answer the question for some of the approaches. The rightmost numbers shows the median of all ratings for each approach for easier comparison.
According to the answers of the participants *TOGAF* (median of 4) is the most useful approach for identifying and supporting stakeholders in the context of EA management. It is followed by the *EAM Pattern Catalog*, *reference models*, *EA management books*, and the *Zachman Framework* all with a median of 3.
The good rating of the EAM Pattern Catalog for identifying and supporting stakeholders is interesting, because it does not yet include a list of stakeholders. Such a list is currently developed as part of a master's thesis. Currently, stakeholders are given in the implementation section of the EAM patterns.

7.3.1.5. Question 5: How useful are the following approaches for identifying and organizing concerns for your EAM approach?

Question number five analyzed available EA management approaches concerning their help in identifying and organizing concerns.
Participants could rate the EA management approaches on a scale from *1=No help* to *5=Fulfills all needs*.

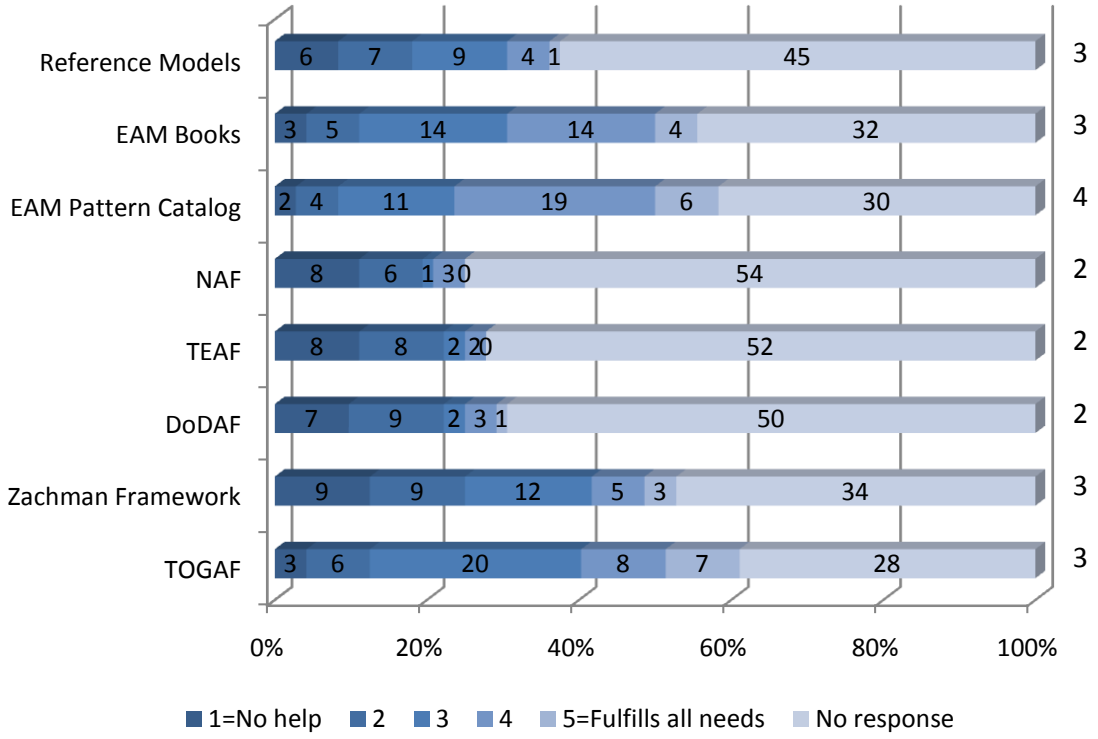

Figure 7.18.: Usefulness for identifying and organizing concerns

Looking at Figure 7.18 it can easily be seen that the *EAM Pattern Catalog* got the highest median (median of 4) of all analyzed EA management approaches. Making it the most valuable approach for identifying and organizing concerns. The EAM Pattern Catalog is followed by *EAM books*, *TOGAF*, the *Zachman framework*, and *reference models* in this order. Far behind are *DoDAF*, *TEAF*, and *NAF*, which like in the previous question received the fewest answers.
A reason for this result may be that the EAM Pattern Catalog is a concern-driven approach, including an explicit list of concerns, which are organized according to typical topics in EA management, as one major starting point. This simplifies the selection of EAM patterns, which are most valuable for addressing existing concerns in a company.

7.3.1.6. Question 6: How useful are the following approaches for establishing governance structures for EAM in your organization?

Question number six analyzed available EA management approaches concerning their help in establishing governance structures.
Participants could rate the EA management approaches on a scale from *1=No help* to *5=Fulfills all needs*.

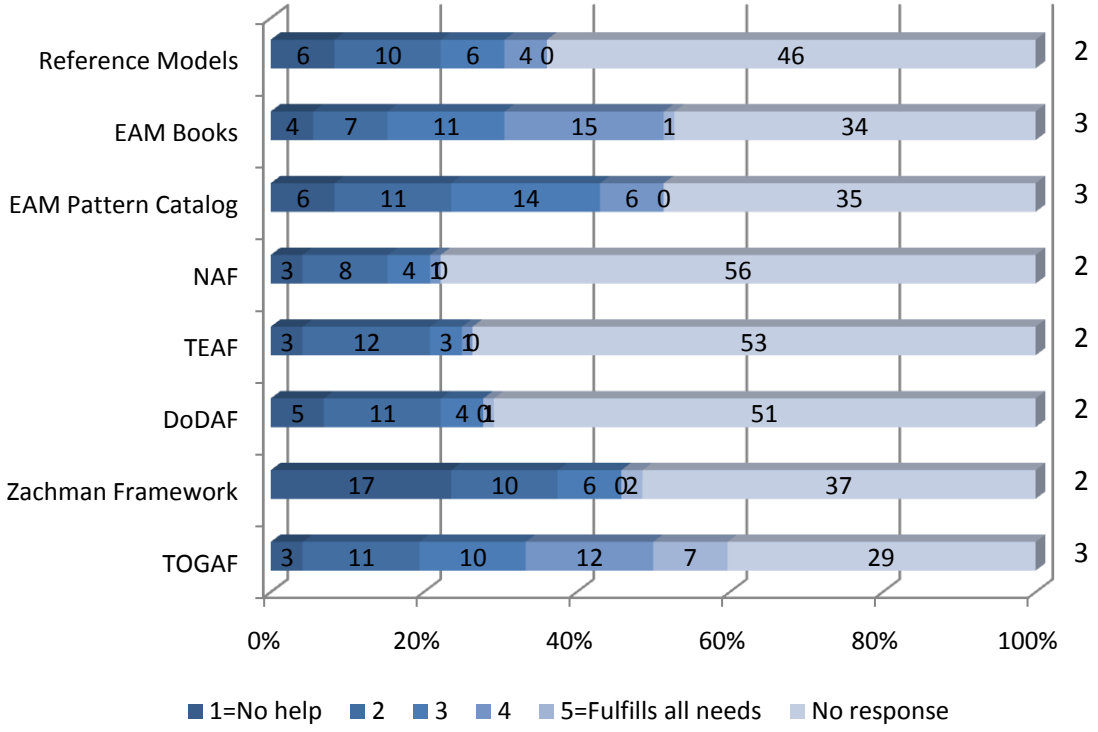

Figure 7.19.: Usefulness for establishing governance structures

Figure 7.19 shows that there is not prominent approach, which had been rated significantly better then the other approaches. *TOGAF*, *EAM books*, and the *EAM Pattern Catalog* received a median value of 3 and can therefore be considered to be the most useful approaches. The other not yet mentioned approaches were rated relatively equal, with *NAF*, *TEAF*, and *DoDAF* again receiving the fewest answers.
In the currently available EAM patterns the information about establishing governance structures is limited. Beginning with [Er08] and the adapted EAM pattern form, the patterns are stepwise extended by such information, e.g. in the implementation section (see Section A.1.1 for an example). Nonetheless this will be one important task in the future development of the EAM Pattern Catalog.

7.3.1.7. Question 7: How useful are the following approaches for defining a holistic and integrated process for EAM in your organization?

Question number seven analyzed available EA management approaches concerning their help in defining a holistic and integrated EA management process.
Participants could rate the EA management approaches on a scale from *1=No help* to *5=Fulfills all needs*.

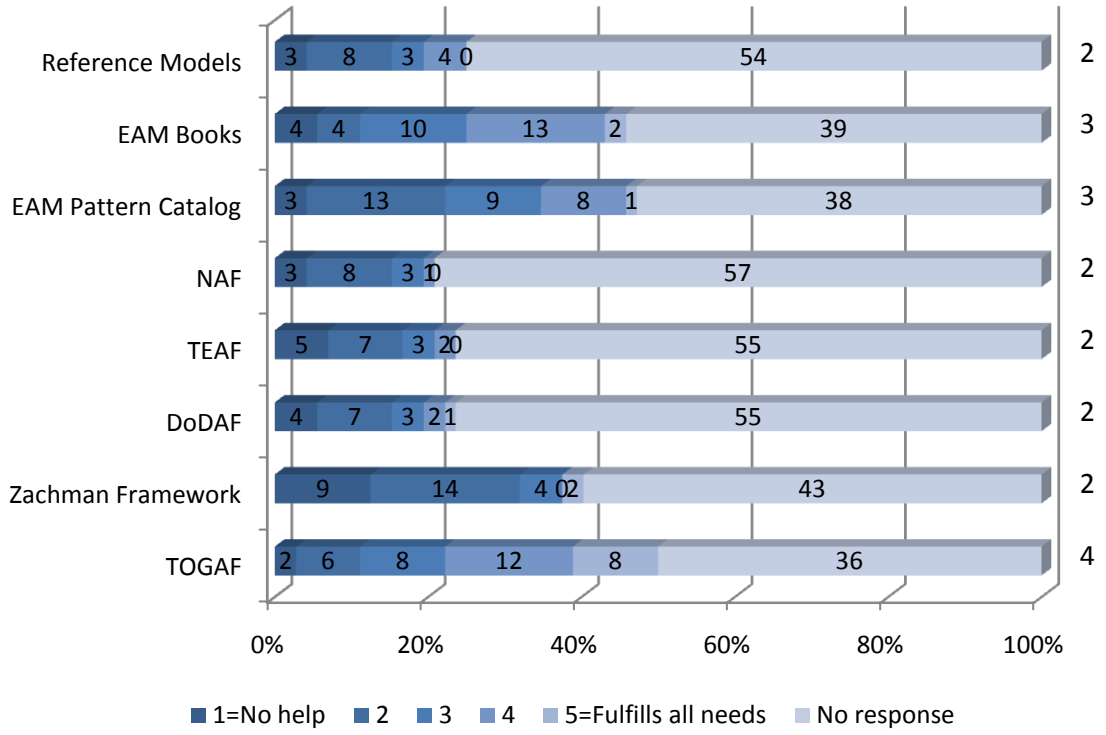

Figure 7.20.: Usefulness for defining a holistic and integrated EA management process

In this question *TOGAF* (median of 4) received the best rating of all approaches (see Figure 7.20). This result is not surprising as a major benefit of TOGAF is its integrated process the ADM (see Section 7.1.1). TOGAF is followed by *EAM books* and the *EAM Pattern Catalog* with a median of 3. All other approaches were rated by a median of 2.
A problem in the results for this question is the high number of *no response* answers. Only TOGAF was evaluated by 50% of the participants. This may limit the information value of this question.
Although the EAM Pattern Catalog has been rated to be one of the top three approaches concerning the definition of a holistic and integrated EA management process, there are nevertheless efforts planned to improve the performance in this area. One possibility is to enrich M-Patterns with a more formal description of the process documented in the M-Patterns.

7.3.1.8. Question 8: How useful are the following approaches for identifying visualizations for your EAM approach?

Question number eight analyzed available EA management approaches concerning their help in identifying useful visualizations for EA management.
Participants could rate the EA management approaches on a scale from *1=No help* to *5=Fulfills all needs*.

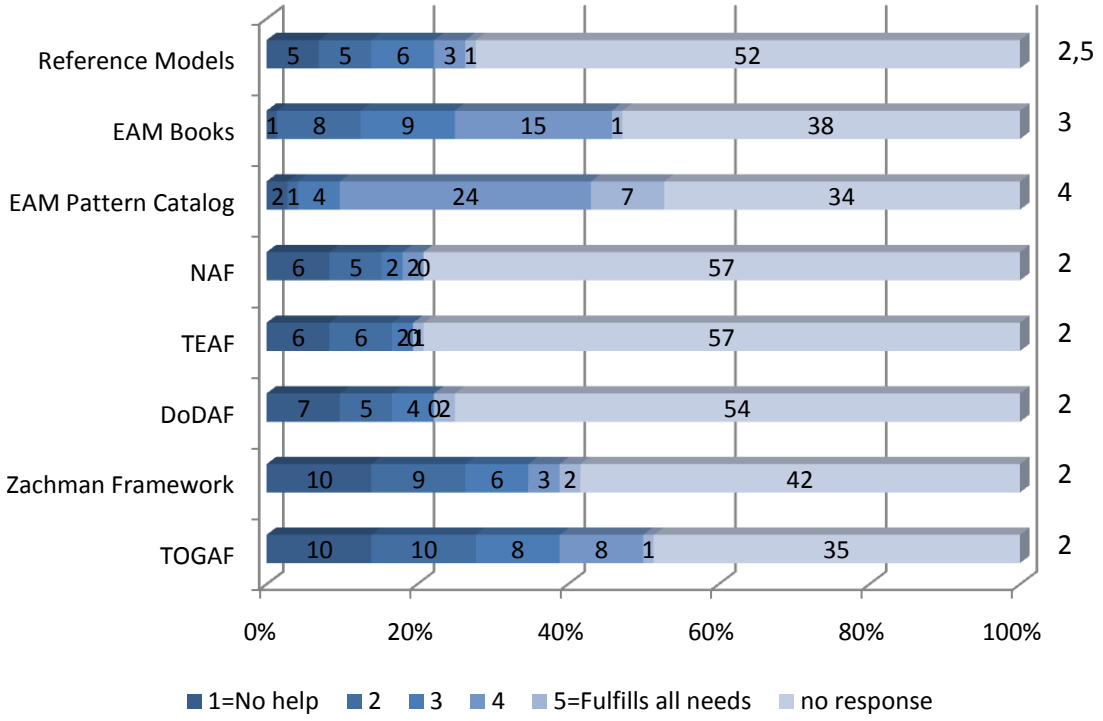

Figure 7.21.: Usefulness for identifying EAM visualizations

The participants of the survey rated the *EAM Pattern Catalog* (median of 4) to be the most useful approach to identify visualizations for EA management (see Figure 7.21). One way to use the EAM Pattern Catalog is to use it like a reference book for visualizations. The result of this question shows that this usage scenario is already well addressed by the currently available EAM patterns.
The EAM Pattern Catalog is followed by *EAM books* with a median of 3 and *reference models* with a rating between 2 and 3. All other approaches received a median of 2. In this question one major shortcoming of TOGAF becomes visible, the support for selecting appropriate visualizations supporting the ADM. An attempt to address this shortcoming can be found in [Bu09f], where TOGAF is complemented with EAM patterns. Providing additional guidance for such tasks will be one goal of the future development of the EAM Pattern Catalog.
The question whether there is a demand for using EAM patterns in conjunction with other EA management approaches is further analyzed in question 17 (see Section 7.3.2.7).

7.3.1.9. Question 9: How useful are the following approaches for defining a metamodel (information model) for your EAM approach?

Question number nine analyzed available EA management approaches concerning their help in defining an information model for EA management.
Participants could rate the EA management approaches on a scale from *1=No help* to *5=Fulfills all needs*.

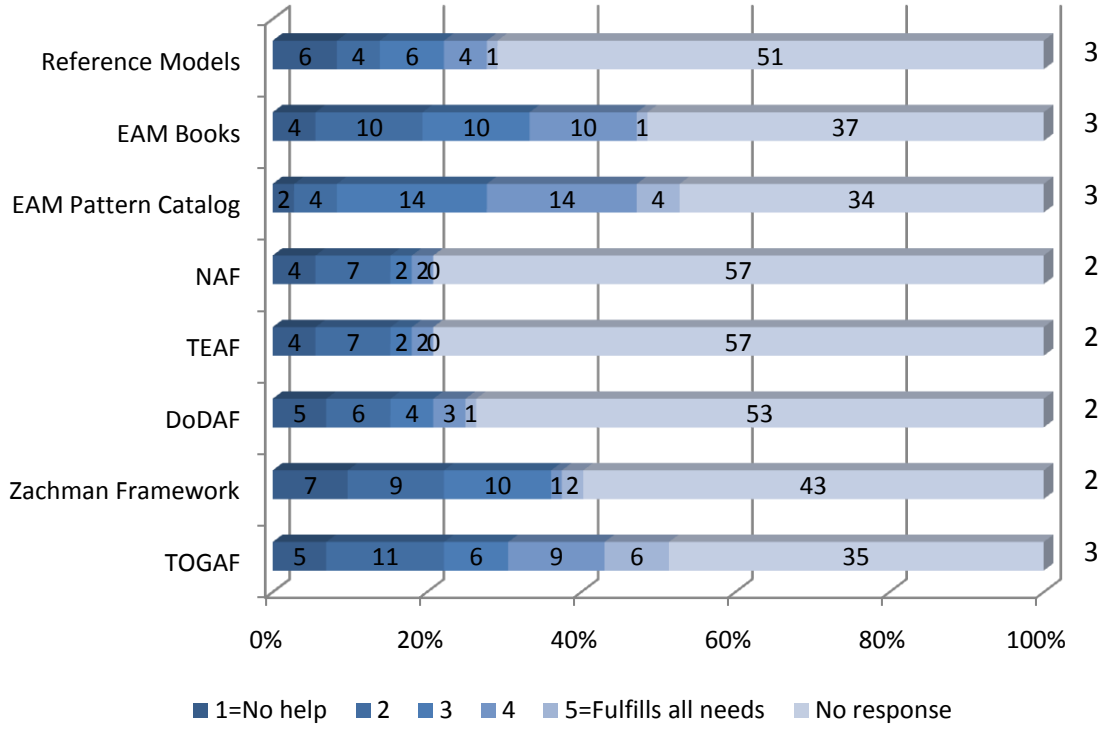

Figure 7.22.: Usefulness for defining an EAM metamodel

The last question where the EAM Pattern Catalog is compared to other approaches did not result in one approach dominating the others. Four approaches received a median of 3: The *EAM Pattern Catalog*, *TOGAF*, *EAM books*, and *reference models*. All other approaches received a median of 2.
Compared to the previous question (see Section 7.3.1.8) this result is surprising. Even though the EAM Pattern Catalog does well in the comparison, it is not dominating the other approaches. The EAM Pattern Catalog includes I-Patterns, which can be used exactly like V-Patterns, in a reference book like manner. The only difference is that I-Patterns document information model fragments, instead of viewpoints. One possible reason for this result is that integrating I-Patterns can in some cases be difficult and that therefore additional guidance is required. This assumption is backed by the results of question 19 (see Section 7.3.3.2) where additional guidance on how to integrate I-Patterns is rated an important topic for the future development.
Another explanation for the result of the EAM Pattern Catalog may be that additional

promotion is required to make the approach of constructing information models based on I-Patterns more popular.

7.3.1.10. Question 10: What topics in EAM are you most interested in?

Question number ten analyzed which EA management topics are most interesting for the participants of the survey. They could select from the following topics:

- Standardization and Technology Homogeneity Management
- Business Processes Support Management
- Application Landscape Management
- Project Portfolio Management
- Infrastructure Management
- Interface, Business Object, and Service Management
- Metrics
- Other (free text)

Figure 7.23 shows the results for this question. The previously given list was extended by the participants by two more topics via the *other* option: EA modeling and IT support of business strategy. These two topics have also been included in the diagram.

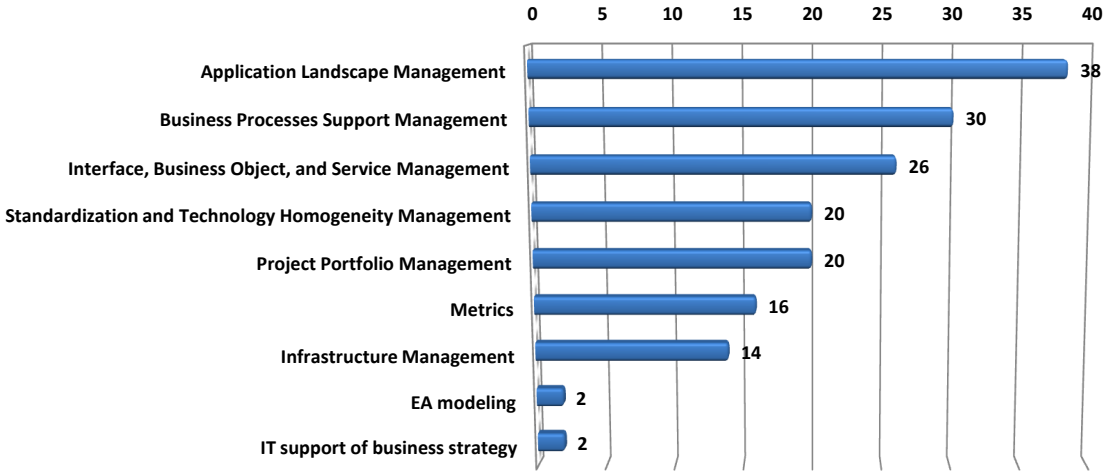

Figure 7.23.: Topics of interest in EA management

The topic with the most votes is *Application Landscape Management* (38 votes), followed by *Business Processes Support Management* (30 votes), and *Interface, Business Object, and Service Management* (26 votes).
The other options roughly got half as many votes or less compared to the top ranked topic: *Standardization and Technology Homogeneity Management* (20 votes), *Project*

Portfolio Management (20 votes), *Metrics* (16 votes), and *Infrastructure Management* (14 votes).
Far behind are the two additional topics, which both received two votes by different participants. This may be due to these two choices have not been listed for selection to everyone but have been added on behalf of the participants via the other option.
Based on this results, it can be said that major interest of the participants is on the management of business applications, their interconnections, their support of business processes, together with planning aspects. This typically matches with the interests during the introduction of EA management. In a subsequent phase topics like standardization, project portfolio, metrics, and infrastructure management are of importance.
These results may be of interest considering a recommendation system for the EAM Pattern Catalog Wiki trying to propose EAM patterns, which should be looked at first. Additionally, the results give hints about the future development of EAM patterns, because EAM patterns belonging to topics with high interest votes should be considered first for revision and enhancements.

7.3.2. Questions about the utilization of the EAM Pattern Catalog

The second group of questions analyzed the utilization of the EAM Pattern Catalog in detail.

7.3.2.1. Question 11: In how many projects have you used the EAM Pattern Catalog?

Question number eleven analyzed in how many projects the participants used the EAM Pattern Catalog. They could select from the following options:

- 1 project
- 2 projects
- 3 projects
- 4 to 10 projects
- More than 10 projects

Participants were offered this hint: If you are a consultant you may use the EAM Pattern Catalog in multiple projects with different companies. If you are working e.g. as an enterprise architect for a company, you typically do one project.
The results for this question are shown in Figure 7.24. The larger part of the participants (39%) used the EAM Pattern Catalog in one project. This is followed by two projects (10%) and three projects (1%). The percentage of people using the EAM Pattern Catalog in four to ten projects is even greater: 4%. The biggest supprise is that there was one participant used the EAM Pattern Catalog in more then 10 projects.
One issue with the result of this question has to be mentioned. While 21% of the participants answered in question 3 (see Section 7.3.1.3) that they are using the EAM Pattern Catalog or that a colleague is using it, 55% gave a similar answer in this question. The discrepancy between the two results indicates that the result to this question should be handled with care and should be further analyzed.

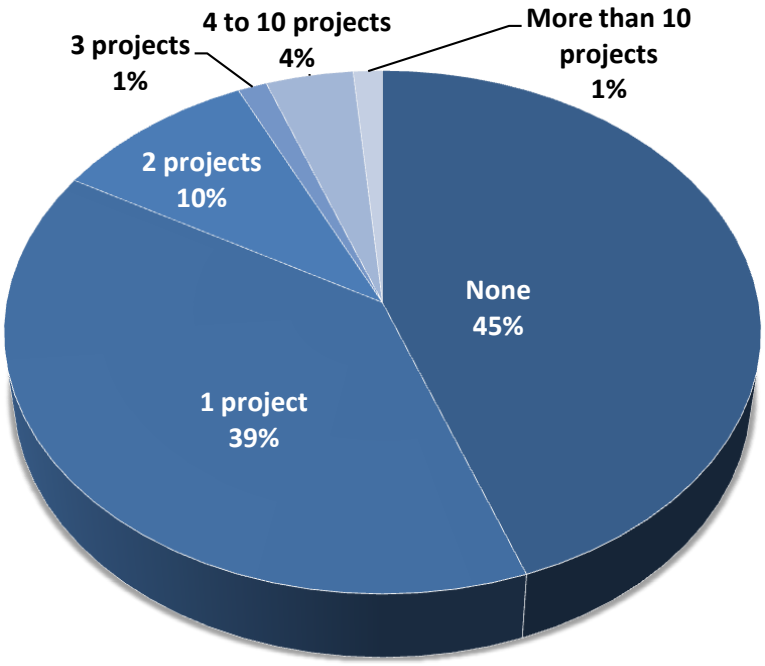

Figure 7.24.: Number of projects using the EAM Pattern Catalog

7.3.2.2. Question 12: Which part of the EAM Pattern Catalog are you using?

Question number twelve analyzed, which of the following artifacts of the EAM Pattern Catalog are used by the participants:

- Printed EAM Pattern Catalog (PDF file)
- EAM Pattern Catalog Wiki
- EAM Pattern Map
- List of concerns
- Other (free text)

The results in Figure 7.25 show that the *EAM Pattern Catalog Wiki* is equally used as the *PDF version* of the EAM Pattern Catalog (23 votes). Although the PDF version has not been updated for over a year now, it is still heavily used. During the last year, the EAM Pattern Catalog Wiki was extended by 44 EAM Patterns even though a few patterns were merged during their revision. This hints towards a need for a printed version of the EAM Pattern Catalog. To be able to provide this the EAM Pattern Catalog Wiki is currently extended by a functionality to export all patterns to a single PDF file.
The *list of concerns* received 12 votes similar to the *EAM pattern map*, which received 13 votes. This shows that the list of concerns is not the only way to select and use EAM patterns. Navigating between patterns using the EAM pattern map is of equal importance. A new release of the EAM pattern map has currently been developed (see

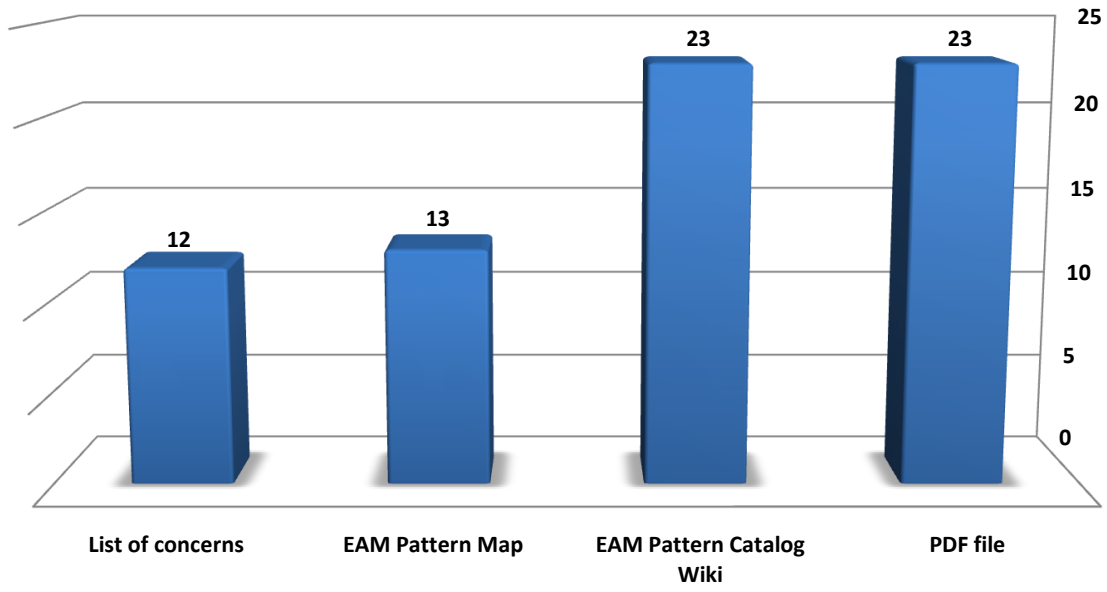

Figure 7.25.: Used EAM Pattern Catalog parts

Section 5.2.4). In a next step the EAM Pattern Catalog Wiki will be extended by the functionality to generate pattern maps on the fly, using the *SyCaService*, a special version of the *SyCaTool* [Sc06a, Bu07b]. This will in the future allow to navigate between EAM patterns in a graphical way, thereby improving the usability of the EAM Pattern Catalog.

7.3.2.3. Question 13: How are you using the EAM Pattern Catalog?

Question number 13 analyzed how the participants are using the EAM Pattern Catalog. They could select from the following options:

- To develop a new EAM approach
- Inspire and assess an existing EAM approach
- To specify goals and requirements for EAM, e.g. for selecting an EAM tool
- For my academic research
- For training new employees in my company
- To document the currently used EAM approach in my company
- As a reference book, e.g. for finding new visualizations
- To organize and classify problems in EAM
- Other (free text)

The results for this question are shown in Figure 7.26. Using the EAM Pattern Catalog *as a reference book, e.g. for finding new visualizations* was by far rated as the most important usage scenario (31 votes). The second most voted choice was *inspire and assess an existing EAM approach* (20 votes). One participant used the *other* choice to document that he or she uses the EAM Pattern Catalog for "general information". This answer has been added to the choice of using the EAM Pattern Catalog as a reference book because it is the idea of a reference book to be used as a general source of information.

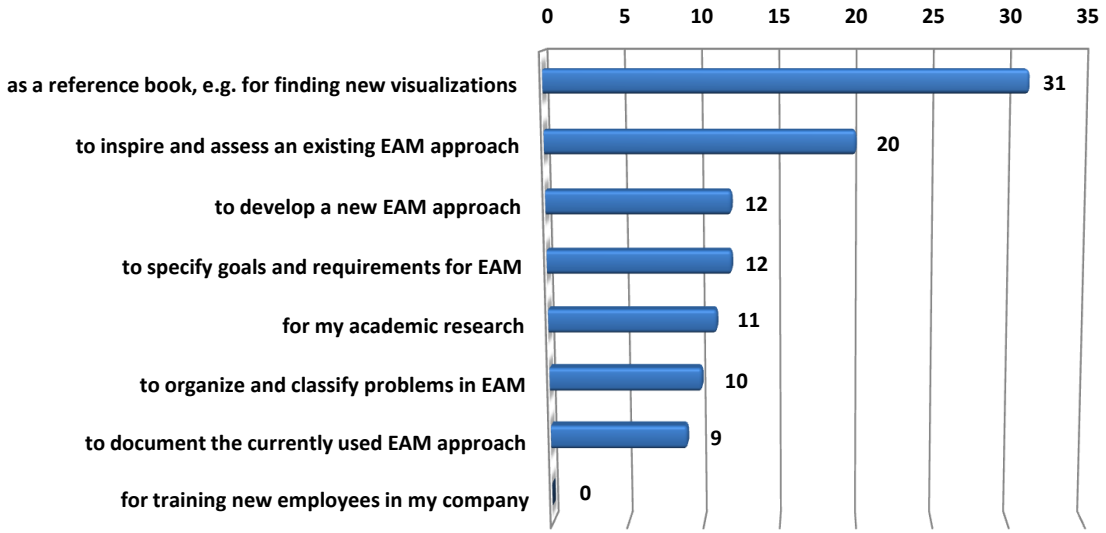

Figure 7.26.: Usage of the EAM Pattern Catalog

To develop a new EAM approach and *to specify goals and requirements for EAM, e.g. for selecting an EAM tool* was rated to be the third important usage scenarios by 12 votes. Specifying goals and requirements is the last usage scenario that was documented in the EAM Pattern Catalog, but the results show that it is as well important as to develop a new EA management approach.
For my academic research is following with 11 votes, showing that the pattern-based approach to EA management is not only accepted by practitioners but also in the academic community. The same was already indicated by the success of the PEAM 2009 workshop (see Section 6.7).
The last two approaches, which were voted for, were *to organize and classify problems in EAM* (10 votes) and *to document the currently used EAM approach in my company* (9 votes). Both approaches have not yet been documented as a usage scenario of the EAM Pattern Catalog. They were included in this survey because they turned out to be usage scenarios at multiple project partners, see e.g. the case study EAM Pattern Catalog at E.ON in Section 6.6. The results for the two approaches show that it is worthwhile to document those usage scenarios in the future development of the EAM Pattern Catalog.
The last choice, using the EAM Pattern Catalog *for training new employees in my company* was not voted for at all, although an indication for such a usage can be found in the case study at E.ON in Section 6.6.

7.3.2.4. Question 14: What EAM Pattern types are you most interested in?

Question number 14 analyzed which of the following EAM pattern types the participants are most interested in:

- M-Patterns
- V-Patterns
- I-Patterns
- Anti Patterns

The four EAM pattern types were explained to the participants as follows: Four types of EAM patterns have been identified in the EAM Pattern Catalog. Methodology patterns (M-Patterns) document processes and governance structures for EA management. Viewpoint patterns (V-Patterns) document viewpoints, which can be used in one or more M-Patterns and use information stored according to one or more information model patterns (I-Patterns). Anti patterns document solutions, which have proven not to work in practice, in order to prevent dead ends.

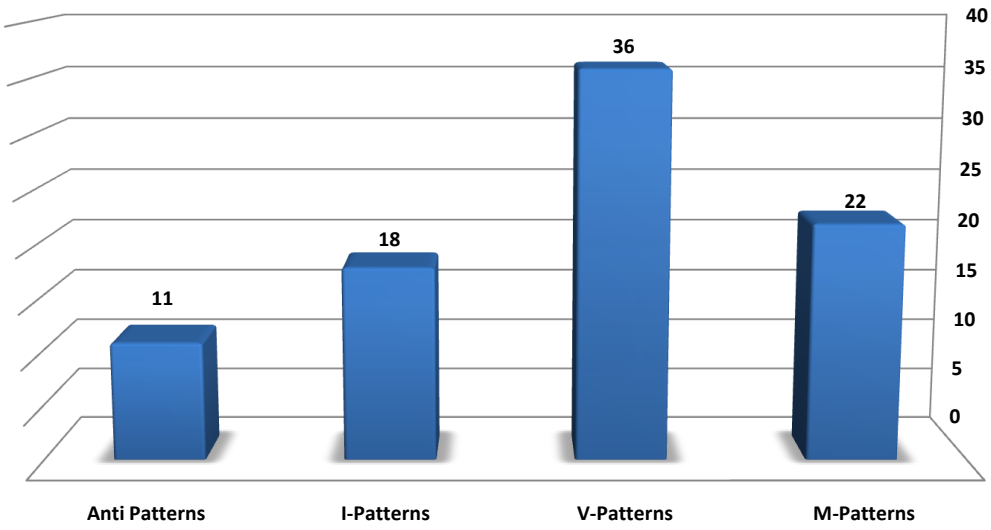

Figure 7.27.: Interest in EAM pattern types

Figure 7.27 shows the question's results. The EAM pattern type participants are most interested in are *V-Patterns* receiving 36 votes. This corresponds to the results of question 8 (see Section 7.3.1.8), where the EAM pattern approach was rated to have the best support for visualizations for EA management.
M-Patterns were rated the second most important EAM pattern type with 22 votes. This matches the results of questions 6 (see Section 7.3.1.6) and 7 (see Section 7.3.1.7). As mentioned in these sections, this is an interesting result, because M-Patterns are currently revised to match the EAM pattern form introduced in [Er08]. They are thereby extended to include stakeholders, governance structures, and detailed information about

the process steps needed to address the concern. This activity will further increase the importance of this EAM pattern type.
The third most important EAM pattern type are I-Patterns (18 votes). As mentioned in Section 7.3.1.9 this result was not anticipated, because I-Patterns are like V-Patterns out of the box solutions that can directly be applied to address a concern.
Although Anti Patterns are the latest addition to the EAM Pattern Catalog, they received 11 votes, showing that there is a demand for documenting additional Anti Patterns. The first two Anti Patterns can be found in Section A.4.

7.3.2.5. Question 15: How many EAM Patterns are you using?

Question number 15 analyzed how many EAM patterns from the EAM Pattern Catalog the participants are using. The following options were offered:

- 1 to 10 EAM patterns
- 11 to 20 EAM patterns
- 21 to 50 EAM patterns
- 51 to 100 EAM patterns
- more than 100 EAM patterns

Figure 7.28 shows the results in form of a pie chart.

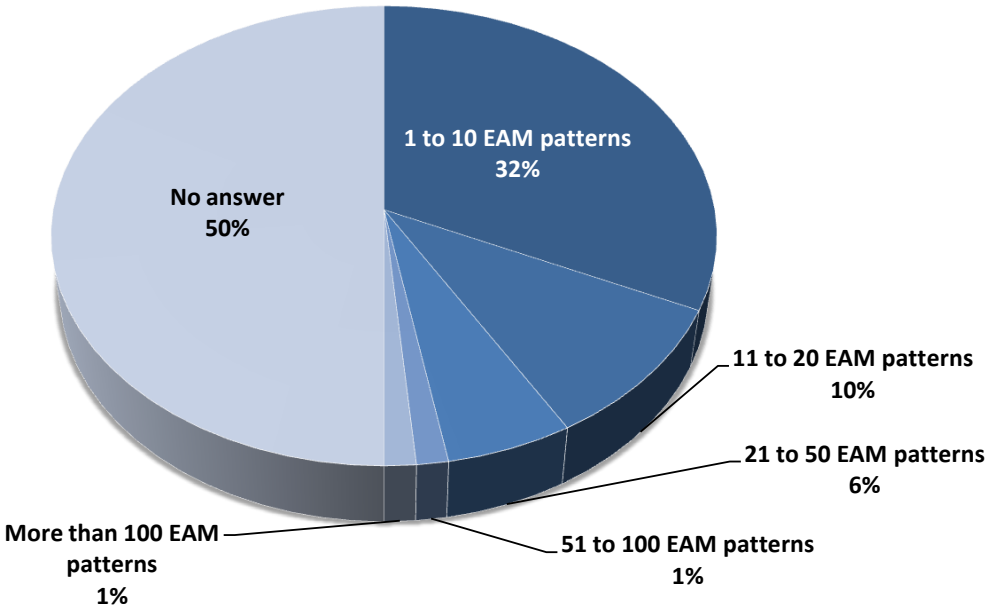

Figure 7.28.: Number of used EAM patterns

32% of the participants, which answered this question, use *1 to 10 EAM patterns* in their work. This corresponds to the results of question 13 (see Section 7.3.2.6) about the utilization of the EAM Pattern Catalog. Most participants use the EAM Pattern

Catalog as a reference book or to inspire existing EA management approaches. Using the EAM pattern approach without any other approach would be expected to result in more patterns to be used.
18% of the participants answered that they are using more then 10 EAM patterns, which reflects a more intensive use of the EAM pattern approach. Whereas the largest part, 10% use between *11 and 20 EAM patterns*, followed by 6% using *21 to 50 EAM patterns*. The least fraction are participants using *51 to 100 EAM patterns* and the ones using *more than 100 EAM patterns* with 1% each. Especially the ones using more then 100 EAM patterns can be considered to be real power users because they use more then half of the 164 EAM patterns included in the EAM Pattern Catalog.

7.3.2.6. Question 16: Which three EAM Patterns are most useful to you?

Question number 16 analyzed which three EAM patterns are most useful to the participants. Names or identifiers of the patterns could be entered into a free text field.
As hints the participants were given web pages with a list of available EAM patterns.

- M-Pattern: `http://eampc-wiki.systemcartography.info/wikis/eam-pattern-catalog/m-pattern`
- V-Pattern: `http://eampc-wiki.systemcartography.info/wikis/eam-pattern-catalog/v-pattern`
- I-Pattern: `http://eampc-wiki.systemcartography.info/wikis/eam-pattern-catalog/i-pattern`
- Anti Pattern: `http://eampc-wiki.systemcartography.info/wikis/eam-pattern-catalog/anti-pattern`

A ranking of the EAM patterns based on their usefulness for the participants of the survey is shown in Figure 7.29. This figure only includes the identifiers of the EAM patterns but the full names are given in the following itemization:

- Business Process and Business Function Relationship (V-12)
- Process Support Map (V-17)
- Service-based Business Process Support Map (V-18)
- Architectural Solution and Technology Mapping (V-23)
- Business Application and Organizational Unit Cluster Map (V-24)
- Application Lifecycle Project Layer (V-27)
- Process Support Map visualizing horizontal Integration (V-28)
- Process Support Map visualizing vertical and horizontal Integration (V-30)

- Process Support Map visualizing Changes in Relation to their Time Horizon (V-32)
- Overview over Lifecycle of Business Applications (V-36)
- Migration of Functionality (V-40)
- Process Support Map, showing standard vs. individual software (V-45)
- Business Object ER Diagram (V-46)
- Process Overview Diagram (V-51)
- Infrastructure Usage Diagram (V-56)
- Strategic Project Portfolio Overview (V-60)
- Technical Project Portfolio Overview (V-61)
- Information Flows (V-63)
- Standard Conformity Management (M-4)
- Process Analysis (M-6)
- Analysis of the Application Landscape (M-13)
- Development of Plan and Target Landscape (M-14)
- Management of Business Objects (M-19)
- Management of Business Services and Domains (M-20)
- Business Process Data Flow Analysis (M-30)

The answers to this question very widely, which can be seen that 25 different EAM patterns were named to be the most useful ones. Hence there are two which received more votes than others. The first one is the Process Support Map *(V-17)* (5 votes) and the second one is Process Analysis *(M-6)* (4 votes). A reason for this may be that Process Analysis uses the Process Support Map for visualizing business processes. The second referenced V-Pattern by Process Analysis is Business Process and Business Function Relationship, which also received one vote.
If the close relationship of Process Support Map to Process Support Map visualizing horizontal Integration (V-28), Process Support Map visualizing vertical Integration (V-29), and Process Support Map visualizing vertical and horizontal Integration (V-30) is also considered – all of them are variants of the Process Support Map – this results in altogether nine votes for Process Support Map.
The results for Business Application and Organizational Unit Cluster Map, only one vote, were rather unexpected, because this visualization was found to be a kind of standard visualization in various projects with practitioners.

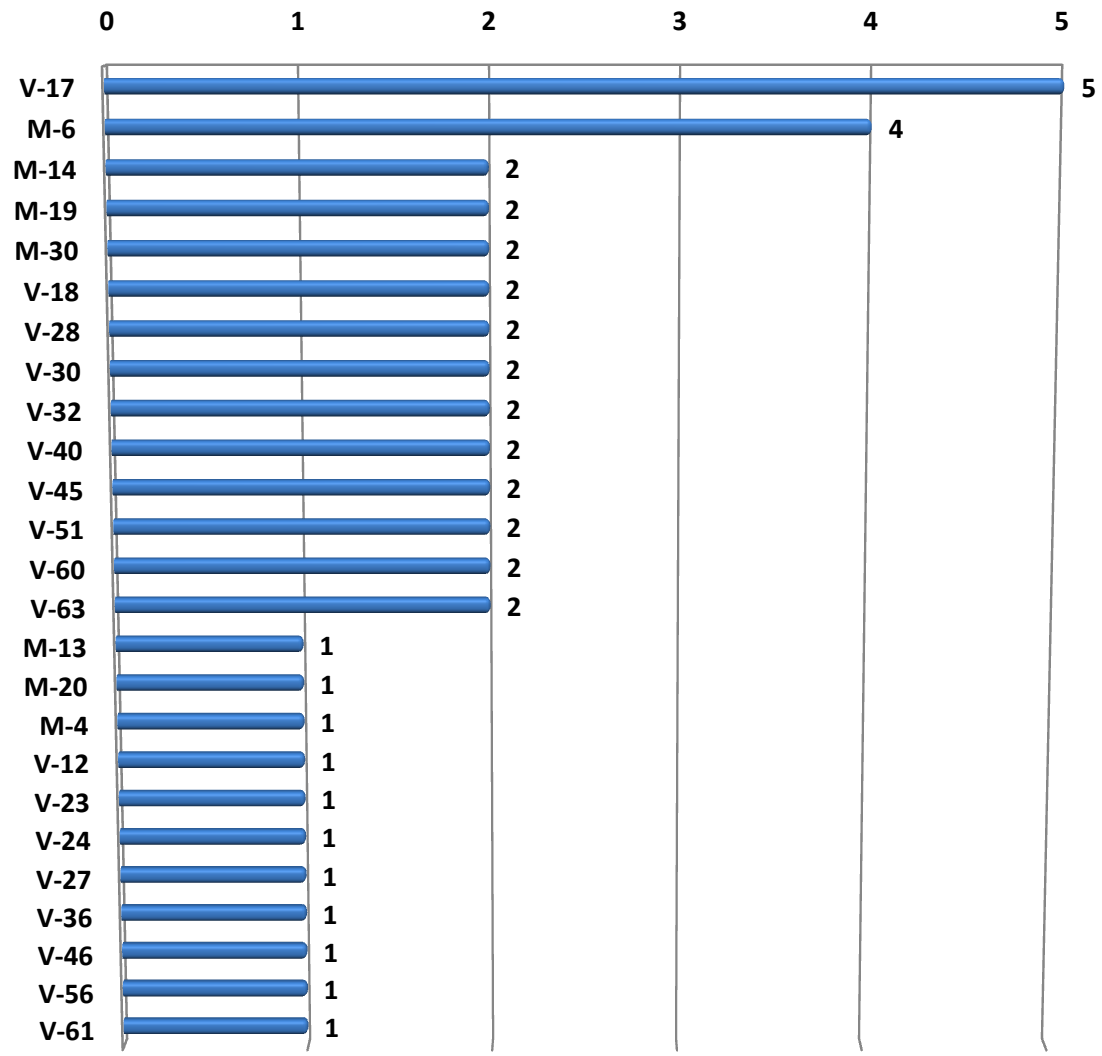

Figure 7.29.: The most useful patterns

All other not yet mentioned EAM patterns received one or two votes. Repeating a survey about the EAM Pattern Catalog with more participants may lead to more significant results concerning, which EAM patterns are most useful. Such an analysis can then be used to give novice EAM Pattern Catalog users an indication which EAM patterns to look at first.

7.3.2.7. Question 17: Which of the following EA frameworks are you using in conjunction with EAM Patterns?

Question number 17 asked, if the participants use EAM patterns in conjunction with one of the following EA frameworks:

- The Open Group Architecture Framework (TOGAF)

- Zachman Framework
- Department of Defense Architecture Framework (DoDAF)
- Treasury Enterprise Architecture Framework (TEAF)
- NATO Architecture Framework (NAF)
- I do not complement an existing framework with EAM Patterns
- Other (free text)

Using an EA framework in conjunction with EAM patterns in this survey means that the framework and the patterns are used in combination to extend and support each other.

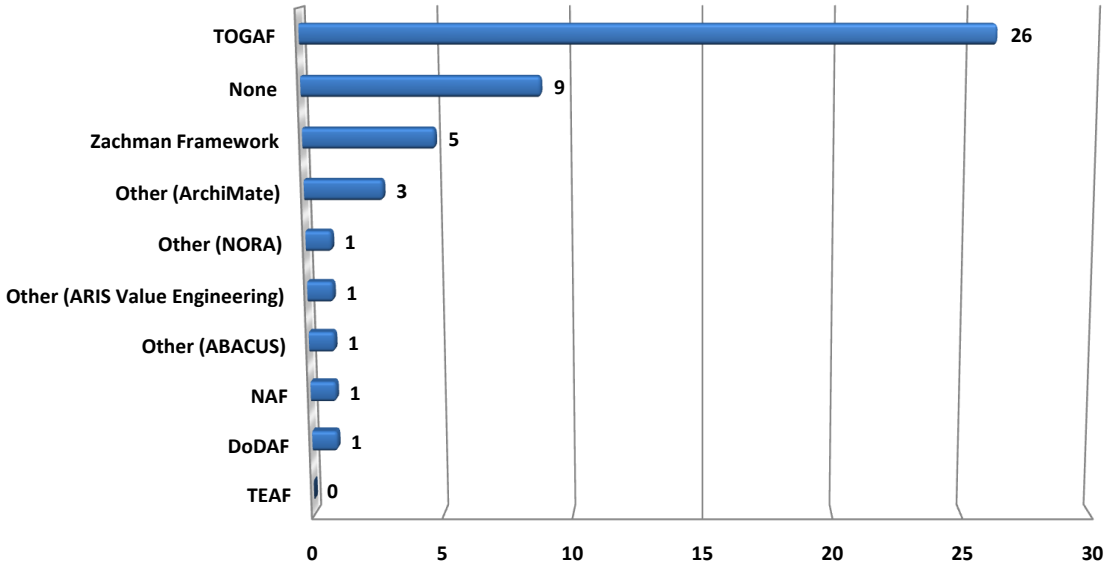

Figure 7.30.: EA frameworks used in conjunction with the EAM Pattern Catalog

Question 13 (see Section 7.3.3.1) already indicated that EAM patterns are often used in conjunction with other approaches or frameworks. This question analyzed this relationship in more detail.
Figure 7.30 shows that *TOGAF* is dominating all other approaches with 26 votes, followed by 9 votes for *no* other approach is used in conjunction. A reason for the dominance of TOGAF in this case may be that it has deficiencies concerning visualizations for EA management (see question 8 in Section 7.3.1.8). This also gives a hint that additional guidance for using the EAM Pattern Catalog in conjunction with TOGAF, like in [Bu09f], would be valuable. The second most votes for a sole usage of the EAM Pattern Catalog allows the conclusion that the EAM Pattern Catalog already is mature enough to be used as a single source of information.
The *Zachman framework* received 5 votes from the participants. On the one hand this shows that the Zachman framework is really used in practice, although it is often said to be too abstract to be used, on the other hand the EAM Pattern Catalog may very well be used to compensate some of the shortcomings of the Zachman framework.

Another approach comes into focus in this question. The *ArchiMate* framework was voted for 3 times, even though it was not part of the initial list of other approaches. This is not too surprising because the EAM Pattern Catalog includes viewpoints and information model fragments from ArchiMate. Anyway this shows that may be some more viewpoints from ArchiMate should be incorporated in the EAM Pattern Catalog. All other approaches only got one vote or none, whereby they can be neglected.

7.3.3. Concluding Questions

The final group of questions included questions about how the EAM Pattern Catalog could be improved in the future and in what way the participants want to take part in the future development.

7.3.3.1. Question 18: What do you think should be improved in the EAM Pattern Catalog?

Question number 18 evaluated how the EAM Pattern Catalog should be improved in the future. Thereby the following options could be selected:

- Additional criteria for organizing EAM Patterns
- Additional search/selection functionalities to identify required EAM Patterns
- That existing EAM Patterns are continually improved/revised
- That additional EAM Patterns are documented and added to the catalog
- A distinction between core and extended EAM Patterns
- Additional information on how to integrate EAM Patterns
- A graphical navigation (EAM Pattern Map) for the EAM Pattern Catalog
- A printed version of the EAM Pattern Catalog
- To have additional information about how to complement existing EA frameworks with EAM Patterns
- Other (free text)

The term *core patterns* means a set of EAM patterns that should be considered first when selecting EAM patterns. This can be compared to a top ten ranking of patterns. Figure 7.31 shows the results, ordered based on the number of votes for each choice. The top-rated choice (18 votes) is *that existing EAM Patterns are continually improved/revised*. On the one hand this leads to the conclusion that at least some of the existing EAM patterns are not yet mature enough, on the other hand it shows that events, like e.g. the PEAM workshop (see Section 6.7) should be repeated and that a continuation of participating in pattern community events like the PLoP conference series is reasonable. *That additional EAM Patterns are documented and added to the catalog* was rated second with 17 votes. There is an ongoing discussion about the growth of the EAM Pattern

Catalog, if it is beneficial to add additional EAM patterns. This result gives an answer to this question and shows that the future development of the EAM Pattern Catalog should also include the documentation of new EAM patterns, because there is a demand for this extension.

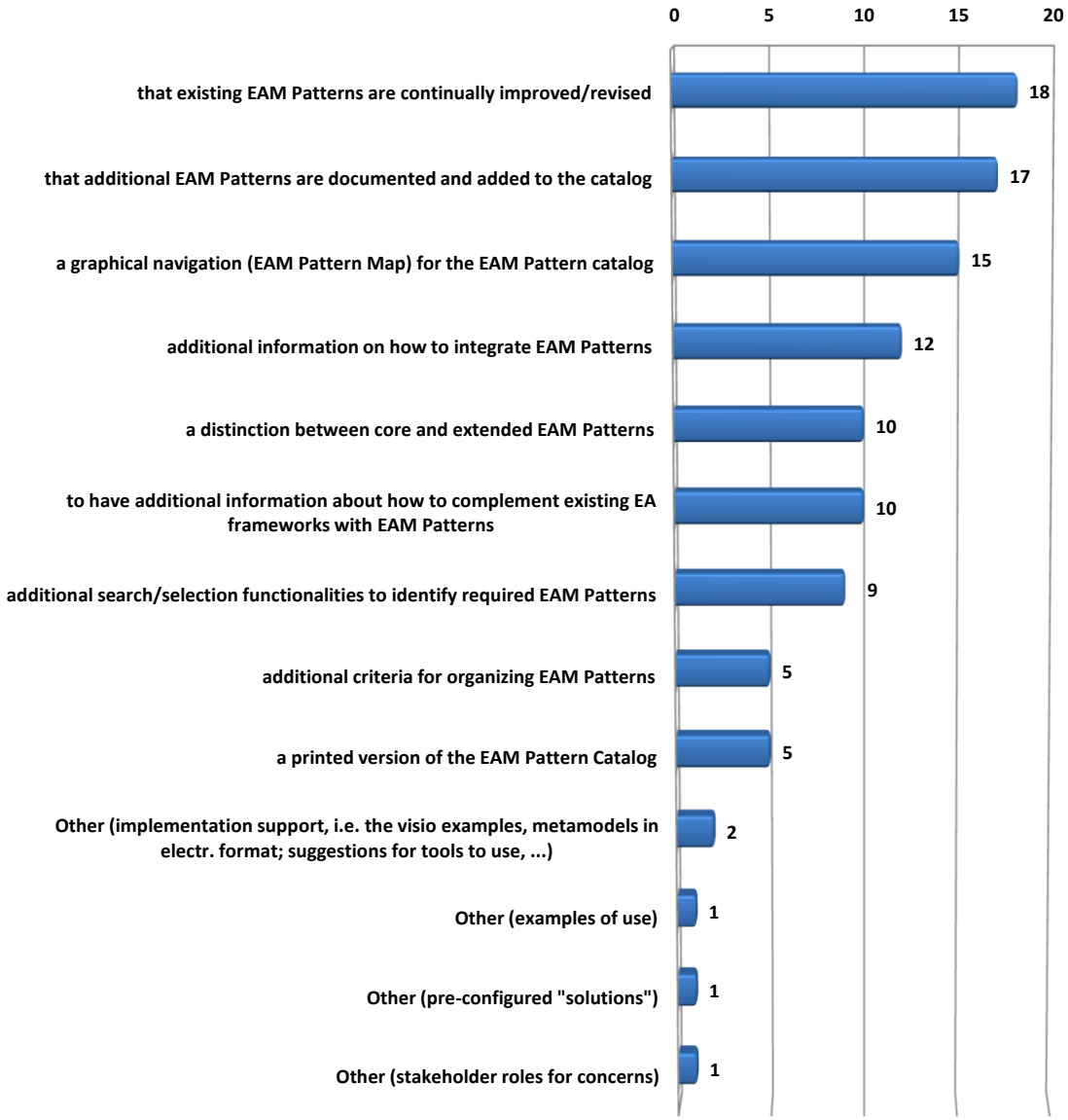

Figure 7.31.: Wishlist how the EAM Pattern Catalog should be improved

The extension of the EAM Pattern Catalog is closely related with improving the usability of the catalog, e.g. by adding *a graphical navigation (EAM Pattern Map) for the EAM Pattern Catalog*. Such an extension was rated as important by 15 participants. Such an extension is currently in development as already described in Section 7.3.2.2.
Another issue already mentioned in previous sections is that it would be desirable to have *additional information on how to integrate EAM Patterns*. The rating for this answer,

12 votes, emphasizes that there is a demand for additional guidelines. This task will be addressed based on the information gained from the various case studies.
A similar demand arises by another choice selected by 10 participants. They want *to have additional information about how to complement existing EA frameworks with EAM patterns*. This issue can also be characterized as a demand to provide additional guidelines for the usage of the EAM Pattern Catalog.
The same number of votes received the choice *a distinction between core and extended EAM Patterns*, which can also be considered to be part of extended guidelines for the usage of the EAM Pattern Catalog. A first attempt to get more information about the most used EAM patterns can be found in question 16 (see Section 7.3.3.2). To be able to give reliable hints about which EAM patterns to consider first a higher number of answers to the question is required, although the results of question 16 already gives some first hints. Another possibility to get more information about the usage of different patterns could be to further analyze the conducted case studies. Additionally, it would be possible to include a functionality for voting for EAM patterns in the EAM Pattern Catalog Wiki. This is further analyzed in the following question in Section 7.3.3.2.
The next answer, *additional search/selection functionalities to identify required EAM Patterns* received 9 votes. This result gives an indication that additional functionality is required in the EAM Pattern Catalog Wiki. 5 votes for *additional criteria for organizing EAM patterns* further supports this demand, as such an extension also requires to introduce new functionality to the EAM Pattern Catalog Wiki. Such extensions are currently developed and will be included in future releases of the wiki software.
Question 12 (see Section 7.3.2.2) already analyzed the usage of different EAM Pattern Catalog elements, with the result that the PDF version is used equally frequent as the EAM Pattern Catalog Wiki, although it has not been updated for about a year. The demand for an update can be seen in the 5 votes for the choice that *a printed version of the EAM Pattern Catalog* should be provided. Maintaining the same information in a wiki and in a PDF is not advisable in a long term perspective, therefore a way to export the content of the EAM Pattern Catalog Wiki directly to PDF is currently developed. This would result in the possibility to create an up to date PDF file, whenever there is a demand for it.
The last four choices were added by participants. Two of them demand for *additional implementation support*, like downloadable information models in electronic format or visio files for creating views corresponding to EAM patterns. One participant voted for providing additional *examples of use* and one for *pre-configured solutions*. Both answers are considered to be part of extended guidelines for using the EAM Pattern Catalog, which are currently under development. One more demand was raised by one participant. He or she demanded for explicit documentation of *stakeholders* for the EAM patterns. Stakeholders are currently mentioned within the EAM patterns, but it would be possible to provide a list of stakeholders in the EAM Pattern Catalog, which shows the relationships between stakeholder, and EAM patterns they may be interested in. Such a list is currently under development as part of a master's thesis.

7.3.3.2. Question 19: Do you plan to contribute to the EAM Pattern Catalog by yourself?

Question number 19 analyzed, if and how the participants would contribute to the EAM Pattern Catalog. They could select from the following options:

- Rating EAM Patterns concerning their usefulness
- Contributing additional patterns
- Peer-reviewing EAM Patterns
- Participating in workshops about EAM Patterns
- Participating in trainings about the EAM Pattern Catalog
- Help writing a book about EAM Patterns
- Other (free text)

Documenting and maintaining patterns is always a community approach. Therefore, the question how the participants would like to contribute to the future development of the EAM Pattern Catalog is of great importance.

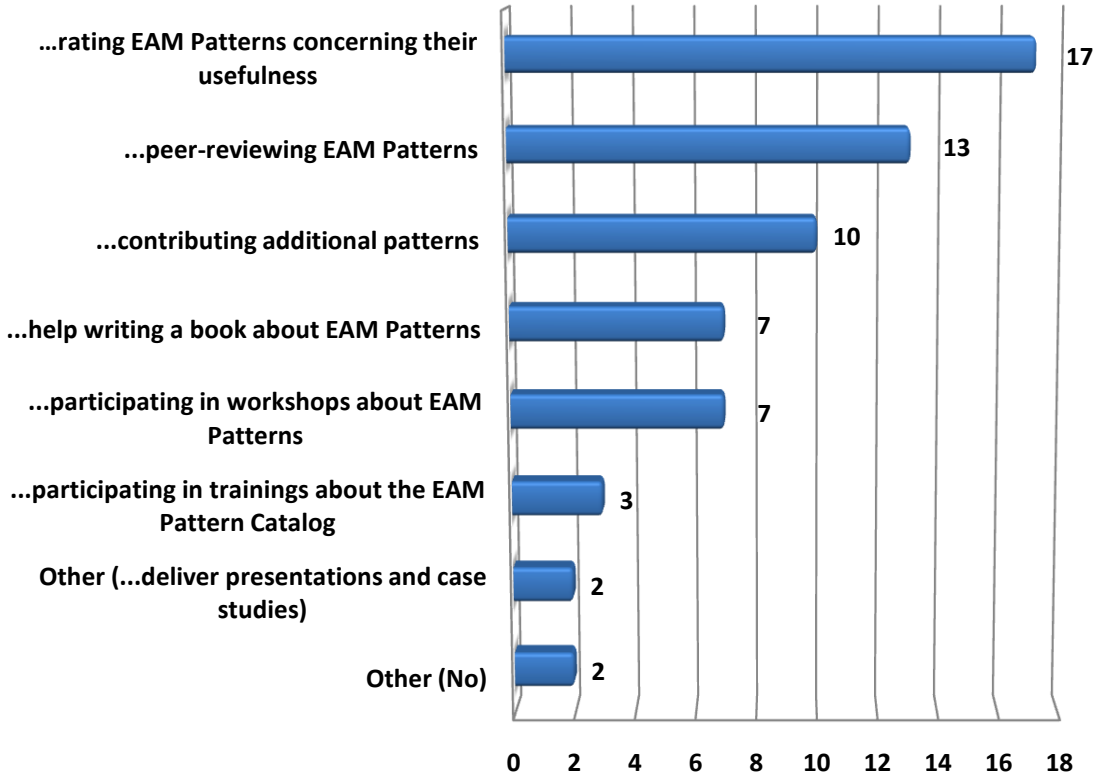

Figure 7.32.: EAM Pattern Catalog contribution

Figure 7.32 shows that 17 participants would participate by *rating EAM patterns concerning their usefulness*. As mentioned in the previous question (see Section 7.3.3.1 this requires to extend the current functionality of the EAM Pattern Catalog Wiki.
Peer-reviewing EAM patterns is also of great importance for the future development of the EAM Pattern Catalog. Therefore, the 13 votes for this choice are very promising for the future development. A way to establish a peer-reviewing process in the EAM Pattern Catalog Wiki could be to point to one EAM pattern each week, which needs to be revised. A similar approach like is pursued in other wikis, e.g. in Wikipedia, which from time to time initiates a quality offensive.
10 participants answered that they would participate by *contributing additional patterns*. This is an especially promising result because it is only possible to try to complete the EAM pattern language if the EAM Pattern Catalog community is working together on this task.
That the choice *help writing a book about EAM patterns* received 7 votes is also an interesting and promising result. This shows that there is a demand and readiness to help to improve the maturity of the EAM pattern approach, which in the end may lead to a book summarizing the collected EAM patterns together with additional guidelines for their usage.
The same number of votes was casted for *participating in workshops about EAM patterns*. This can be seen as a part of the peer-reviewing process, which may be supported by events like the PEAM workshop (see Section 6.7). Under these circumstances it would also make sense to try to initiate a *focus group* at one of the well-established pattern conferences like the PLoP-series.
Using an extensive approach for EA management typically requires a certain amount of training. Nevertheless *participating in trainings about the EAM Pattern Catalog* was only be selected by 3 participants. This may be an indication that, although the EAM Pattern Catalog is extensive, it can still be used without requiring a special training.
Two participants answered that they can *deliver presentations and case studies*. These can be used differently. They may be used to develop guidelines and training material for using the EAM Pattern Catalog, but also to document and provide best practices. Additionally, those case studies can be used to identify core patterns. Another two participants answered, that they do not want to contribute to the EAM Pattern Catalog.

7.3.3.3. Question 20: Is there anything else you want to tell us about the EAM Pattern Catalog?

Question number 20 offered the possibility to the participants to give some concluding remarks about the EAM Pattern approach, the EAM Pattern Catalog Wiki, and about their wishes concerning the future development.
The remarks were organized to different topics addressed by the remarks. Three remarks were given about the future development of the EAM Pattern Catalog.

> You should codify the whole lot using ABACUS like they have done for IEEE 1471 / ISO 42010 - `http://www.iso-architecture.org/ieee-1471/abacus-sbscs/`

Such an organization could be a variant to the current categorization according to topics in EA management. This can be implemented by using the tagging functionality provided in the EAM Pattern Catalog Wiki.

> I think it is important to work more about business requirements, I think it's an intermediate point between organization strategy and EA management, besides traditional frameworks don´t offer enough about it.

The topic *support of business strategy by IT* was already discussed in question 10 (see Section 7.3.1.10). It will be considered in the future development of EAM patterns.

> Keep books coming. More information, case studies, real world usage, etc.

Further case studies are included in this thesis and will be made available in the EAM Pattern Catalog Wiki after the publication of the thesis.
Concerning general remarks on the EAM Pattern approach and the EAM Pattern Catalog the following six answers have been given.

> Overall, I want to thank you for taking this huge step in identifying and documenting these EAM patterns. The EAM Pattern Catalog is indeed a very useful resource and can some day become as well respected and recognized as GoF[2] patterns. As future considerations, I would like to see some enabling tools (e.g. Microsoft Visio stencils for I-Patterns) as well as a comprehensive meta-model (information model). I would imagine that the M-Patterns will be refined over time to be more prescriptive and complete. Also, some direct support to SOA will be useful.

As previously mentioned Microsoft Visio stencils and downloadable information model fragments are currently under development and will be included in the EAM Pattern Catalog Wiki in the future. M-Patterns are extended and revised in the continuous development of the EAM Pattern Catalog and are included in the wiki in a step wise manner. Service-oriented architecture (SOA) is already addressed by some EAM Patterns, e.g. by Service Lifecycle Viewpoint. Additional EAM patterns are continually added to the EAM Pattern Catalog.

> Being relatively new to this discipline I have found the approach in the EAM Pattern Catalog enables quick dives as well as relating concern through approach to deliverable. However, the structure of the catalog is such that it's not always obvious where the M-Patterns (and the supporting I- and V-Patterns) might fit into an overall approach.

This issue will be addressed in the future by providing additional guidelines with the EAM Pattern Catalog, which will help to fit the EAM patterns into an overall approach.

[2] The Gang of Four (GoF) refers to Gamma, Helm, Johnson, and Vlissides and their book on design patterns [Ga95].

> The patterns helped us tremendously and I cannot thank you enough for doing the research and making it available. We are happy do anything we can to contribute and are anxious to be involve to help in any way we can. I know there is work ongoing with a modeling tool and we are anxious to test and contribute to that as well.
>
> Please continue with the work in EAM area.
>
> It groups things usefully and is good for quick lookups. Thanks for you efforts so far!
>
> Great work! Keep at it!

7.4. Summary

This chapter introduced nine different existing EA management approaches and compared them to the pattern-based approach to EA management proposed in this thesis. The comparison showed that none of the approaches fully accomplishes as criteria required for an EA management method. Based on the select comparison criteria the three most complete approaches are MEMO, BEN, and the pattern-based approach to EA management. MEMO lacks full support for processes and roles, whereas the language used in BEN is not available. The EAM Pattern Catalog requires additional guidelines for its application.
Complementing this comparison the results of an online survey evaluating the EAM Pattern Catalog were presented. The survey showed that there is an interest in the pattern-based approach to EA management and it is already applied by practitioners and in academia. Comparing the EAM Pattern Catalog to existing EA management approaches e.g. concerning stakeholder support, visualization, etc. also showed its applicability and utility. Concerning the utilization of the EAM Pattern Catalog the participants indicated a demand for a PDF version and the EAM Pattern Catalog Wiki, whereas the EAM Pattern Catalog is most often used as a reference book or to inspire and asses an existing EA management approach. V-Pattern were rated to the pattern type with the highest interest by the participants. In the concluding questions the participants indicated to be interested in contributing to the future development of the EA management, especially concerning identified potentials for improvement. They also indicated the value of the pattern-based approach to EA management for their work and the demand for its future development.
The evaluation, like the application, confirmed the general research hypothesis for this thesis. The pattern-based approach to EA management is valuable for and applicable in academia and practice.

CHAPTER 8

Conclusion and Outlook

Contents

This chapter concludes the thesis by summarizing the previous chapters and provides an outlook on the future development of the pattern-based approach to EA management.

8.1. Conclusion

This dissertation presented a pattern-based approach to EA management, provided guidelines for its usage and demonstrated its usefulness and applicability in six industrial case studies, one academic workshop, a lecture at the Technische Universität München, and a survey.
Chapter 2 introduced EA as the fundamental conception of the enterprise and EA management as a continuous and iterative process controlling and improving the existing and planned IT support for an organization. Managing the EA was found to be a young discipline with a relatively low maturity, which is more an art than a mature engineering discipline. Consequently, mature engineering disciplines are analyzed for characteristics that can be used for improving the maturity of EA management. The result is that engineering is based on well-known and established methods. Therefore, requirements for an EA management method are proposed. Starting with an historic overview, including available definitions and related approaches, the concept of patterns was introduced in Chapter 3. Patterns are a way to document proven practice solutions for recurring problems in a given context. They are used in various domains like education, project management, architecture, software engineering, etc. and proved to be a valuable approach for documenting knowledge. This knowledge can then be used by novice users

and provides additional insights and alternatives to experts. An excursion to how patterns are documented was followed by an overview about different pattern forms known in the pattern community.
As a core contribution of this thesis patterns were applied to EA management in Chapter 4 as possible solutions to recurring problems. Three types of EAM patterns – M-Patterns, V-Patterns, and I-Patterns – were introduced as ways to document proven practice solutions to recurring problems in a given context. Complementing EAM anti patterns document solutions, which have repeatedly proven not to work, in order to prevent typical mistakes. These patterns, together with additional guidelines for using EAM patterns, make up the constituents of an engineering approach for EA management and therefore provide a contribution to increase the maturity of the EA management discipline.
Chapter 5 showed how 164 EAM patterns and EAM anti patterns were mined, documented, evaluated, and revised in an extensive two and a half year process. The result is the EAM Pattern Catalog, which is available for open access in form of a wiki, in order to support a growing community in documenting additional and improving existing EAM patterns and EAM anti patterns. These patterns make up a pattern language for EA management. As relationships are important in a pattern language these were also further analyzed. In addition guidelines for selecting and integrating EAM patterns, as well as operations on the EAM Pattern Catalog were presented.
Chapter 6 applied the pattern-based approach to EA management in practice and in academia. Its utility and applicability in practice was showed in six industrial case studies. Acceptance within the academic community was likewise shown by the Patterns in Enterprise Architecture Management (PEAM) 2009 Workshop. The pattern-based approach was additionally applied in a lecture at the Technische Universität München. The results of the lecture indicated that novice users were appropriately supported.
Chapter 7 presented related EA management approaches and compared them to the pattern-based approach to EA management proposed in this thesis. This was complemented by a survey, which empirically validated the EAM pattern approach and showed its benefits compared to other approaches to EA management.
The general research hypothesis introduced in Section 1.3 could be validated during the application and validation of the pattern-based approach: Using the pattern-based approach to EA management, the results are superior to results for other EA management approaches.
The thesis contributes two results to scientific research about EA management and to the EA management community in general. A scientific method was developed based on principles from engineering disciplines, which can be used and is valuable for practitioners. This is a first step to establish a mature EA management discipline. In addition existing knowledge about EA management in academia and practice was documented, structured, and presented in a form that supports the developed scientific method and is valuable for academia and practitioners.

8.2. Outlook

During the development of the pattern-based approach to EA management and its validation topics for future development and research have emerged. These are presented in this section.

Refine Organization and Selection of EAM patterns and EAM anti patterns

Identifying the right patterns to address the problems of a pattern user is an important issue for a comprehensive pattern language. The EAM Pattern Catalog currently includes 164 EAM patterns (as of 2009-09-05) and is considered to be such a language. Some approaches for identifying the required patterns were already mentioned in this dissertation (see Section 5.3) but as the topic of EA management itself is a broad one other ways for organizing and selecting EAM patterns might be useful. For example patterns could additionally be organized, e.g. according to industries like automotive or telecommunication, where their usage proved to be reasonable. Using pattern language grammars and design space analysis (see Section 5.3.3) or using the ELECTRE II method (see Section 5.3.4), applying a hierarchy of criteria to the forces of a pattern, would be two more formal approaches. Providing a guiding framework like the layers and cross functions proposed by Wittenburg [Wi07a] is another promising approach.

Extend Tool Support

Organizing and selecting of patterns should be supported by tools. A promising approach to support this is the *HybridWiki*, which is an extension to the Tricia platform used for the EAM Pattern Catalog Wiki, offering the possibility to include and interpret structured meta information.
Improving the EAM pattern approach usage an extended tool support appears to be reasonable. This tool support can be useful on different levels. The EAM Pattern Catalog Wiki offers an intuitive way to document, manage, revise, and select EAM patterns. It could e.g. be extended by a graphical navigation between EAM patterns using automatically generated pattern maps, or by a rating functionality to identify the most relevant EAM patterns, etc.
It would be interesting to integrate the pattern-based approach to EA management in an existing EA management tool, like the ones analyzed in [se05, Ma08a]. Ideally, this would result in an environment, where modifying the features of a tool to support EA management is based on selecting or deselecting EAM patterns. In this case the tool would care about problems like e.g. the integration of different I-Patterns, but still offer the possibility to further adapt the selected EAM patterns to company-specific needs. A promising approach to such a tool support for V-Patterns is the *Viewpoint Definition Language* (VDL) proposed by Ramacher [Ra09b]. It allows to specify viewpoints by using the VDL, which can then be implemented in a tool. Prototypically, such a tool support for the EAM Pattern Catalog has been integrated in the SyCaTool [Bu07b].

Extend Guidance

The EAM Pattern Catalog Wiki currently includes three case studies which may be used as guidelines on how to apply the approach proposed in this thesis. These are especially valuable as they document real use cases of the approach. For this reason the case studies included in this thesis will be made available in the future. Another valuable source of information for using the EAM Pattern Catalog and for possible extensions of the EAM Pattern Catalog are the results of the mini projects of the lecture at the Technische Universität München (see Section 6.8). The results will be made available for the EAM Pattern Catalog community after their publication.

Refine pattern-based Approach to EA management

In the lecture at the Technische Universität München, which used the EAM Pattern Catalog, it showed that it would be beneficial to extend currently existing M-Patterns with a more detailed descriptions of the process conducted as part of the M-Patterns, including a specification of the input and output artifacts. This would allow to use these artifacts for indicating, if two M-Patterns can be integrated. Similar benefits would be achievable by creating additional guidelines for integrating and instantiating patterns. These could be derived from the available case studies.
Explicitly considering stakeholders in the pattern-based approach to EA management is another extension, which would be valuable for the users of the approach. On the one hand this would offer another way for selecting EAM patterns, on the other hand considering stakeholders in EAM patterns would improve the value of the patterns themselves as the relevant stakeholders are an important and valuable information. Such an extension is currently developed by Bender [Be09].
Based on the foundation, which has been established by this dissertation additional EA management topics should be considered in the future extension of the EAM Pattern Catalog. Currently available EAM patterns give an overview about the topic of EA management. There are aspects, like e.g. infrastructure management which should be analyzed in more detail to identify and document additional patterns. This is connected with the basic concept, that a pattern language should be exhaustive and complete. Analyzing this aspect of the EAM pattern approach would be another worthwhile future research topic and may lead to a more complete EAM Pattern Catalog.
Another result of the survey was that the EAM Pattern Catalog is used in conjunction with other approaches to EA management. Buckl et al. [Bu09f] already proposed to complement existing approaches, like e.g. TOGAF with patterns. Additional analyses of this topic may result in additional guidelines, which would be valuable for the future development of the EAM Pattern Catalog.

Revise Printable EAM Pattern Catalog

The EAM Pattern Catalog online survey indicated that the PDF version of the EAM Pattern Catalog is as frequently used as the wiki version. This is remarkable, because the PDF version has not been revised for about a year and is missing 40 newly documented patterns as well as revised ones. There are two possible solutions for this problem. The

first one is to publish the currently available EAM patterns and EAM anti patterns in form of a book. This would further steady the pattern-based approach to Enterprise Architecture Management by extending its sphere of influence. The second one is to revise the available PDF document to reflect the status of the EAM Pattern Catalog Wiki. Ideally this would be supported by an export functionality of the used wiki platform.

Strengthen EAM Pattern Catalog Community

An important aspect of patterns is that they are created and used by a community. The same is true for the EA management pattern language. Therefore, the already established community should be strengthened and extended by regularly meetings like the *Patterns in Enterprise Architecture Management Workshop*, newsletters, and other typical community functions. An convergence to existing pattern communities like the *Hillside Group* may also be beneficially and may lead to broadened basis and an enhanced distribution in academia and practice.

The conclusion and the future research areas indicated in this section show that a valuable foundation was established by this thesis, which paves the way for future developments in EA management.

APPENDIX A

Exemplary EAM Patterns and EAM Anti Patterns

Contents

This chapter presents EAM patterns published by Ernst [Er08] and EAM anti patterns published by Buckl et al. [Bu09g] for exemplifying the pattern-based approach to EA management.
The EAM patterns were taken from version 1.0 of the EAM Pattern Catalog [Bu08a]. During a two month shepherding process they were revised three times until their final acceptance. As a main result the form of EA management patterns was adapted to

the form proposed by Buschmann et al. [Bu96], see Section 4.1.1 for more information. The adaptations regarding the content primarily focused on the extension and revision of the existing patterns, especially of Standard Conformity Management, which was also extended by a graphical description of the process. In addition Business Application and Organizational Unit Cluster Map and Cluster Map for using Relationship, as well as Business Application and Organizational Unit Relationshipand Using Business Applicationswere merged. The main reason is that they constitute variants of each other, which are not independently enough to be documented separately. This illustrates how existing EAM patterns can be consolidated. The EAM anti patterns are the latest extension to the pattern-based approach to EA management. They were documented and accepted at the PLoP 2009 conference. They were revised two times during the shepherding process until their final acceptance.

A.1. Methodology Patterns (M-Patterns)

M-Patterns are grouped according to their membership to typical EA management topic areas, like *Application Landscape Planning*, *Support of Business Processes*, *Interface, Business Object and Service Management*, etc. This section includes one exemplary M-Pattern called Standard Conformity Management, which is part of the question complex *Technology Homogeneity*

A.1.1. Standard Conformity Management

Standard Conformity Management defines and manages architectural standards. Analyses on this information may lead to new guidelines concerning architectural standards, as well as roadmaps to increase or decrease standard conformity.

A.1.1.1. Example

As the department store *SoCaStore* grows it collects a variety of business applications. Many use obsolete architectures and technologies. Some of the systems are retired but often the business support of the systems is too valuable to retire them but too expensive to replace them. Additionally, the high number of different architectures and technologies used in the applications, calls for a high number of experts able to operate and maintain the business applications conforming to them. Licensing and maintenance costs as well as costs for integrating different technologies are also a critical factor.

A.1.1.2. Context

An enterprise with a large number of business applications (typically more than 50), which are part of the application landscape and infrastructure software needed to run these business applications.

A.1.1.3. Problem

You feel the risk of an unmanaged application landscape, with a multitude of technologies, will increase the cost of development of new business applications, operation, evolution and retirement of existing business applications. You do not know, if the business applications follow a common blueprint or architectural style and what the impact of a change to these standards would be. Typically such a situation appears in large organizations with decentralized IT departments, after mergers and acquisitions, or just because the degree of disorder increases over time. You believe architectural standards will help to reduce risks and costs through more homogeneity.
How do you establish and manage conformity to architectural standards?

The following *forces* influence the solution:

- **Standard Conformance:** Do currently used business applications correspond to architectural standards? Are deviation reasons documented, e.g. strategic decisions?
- **Standard Modification:** Which activities or projects have to be started in order to improve conformance to architectural standards? Which modifications to the currently used business applications are necessary to achieve conformity?
- **Standard Usage:** Where are architectural standards used, and are there areas where those standards are breached?
- **Standard Definition:** How is an architectural standard (architectural blueprint, architectural solution made, etc.) set-up?

- **Licensing Costs:** How can licensing costs for business applications and infrastructure be reduced?
- **Incompatible Technology Risks:** How can risks concerning the utilization of incompatible technologies for business applications be reduced?

A.1.1.4. Solution

Set architectural standards, i.e. developing a set of architectural blueprints and architectural solutions, and assigning them to new and existing business applications, in order to increase efficiency in IT operation and development. Thereby, approved deviations to the standards also have to be documented and managed.
Architectural standards are thereby be divided in

- architectural blueprints, which define, which abstract technologies, like e.g. a relational database system, may be used for new business application and in
- architectural solutions, which are like an instantiation of an architectural blueprint with concrete technologies, like e.g. an Oracle 9i database.

Architectural solutions and architectural blueprints consider homogeneity not only on the level of a specific kind of technology e.g. programming languages or middleware, but include architectural solutions and consider technologies at the level of standardized technology *bundles*.
After architectural standards have been set, activities and projects for improving conformance to the standards can be derived, which may then enter project portfolio management as proposals.

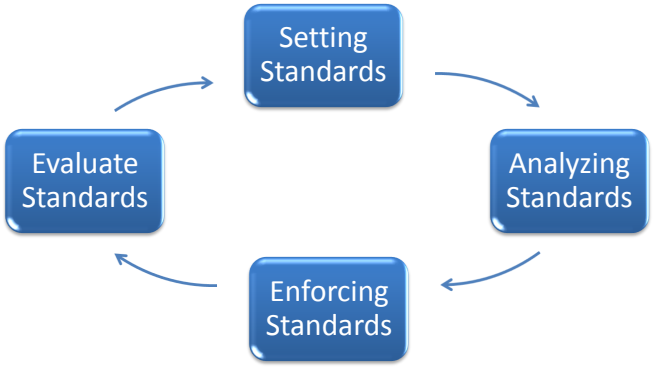

Figure A.1.: Management process for Standard Conformity Management

Subsequently, the four steps of the methodology are described: Firstly setting architectural standards is considered, which afterwards have to be analyzed concerning standard conformity for specific business applications or subsets of the application landscape. This is followed by an enforcement of the defined standards, which at last have to be evaluated if they are still feasible in the company under consideration. To complete the

cyclic process of Standard Conformity Management (see Figure A.1), architectural standards which are no longer feasible have to be adapted or new standards have to be created.
The implementation section of this M-Pattern will additionally cover the aspect of involving the right people and establishing the right governance structures.

A.1.1.5. Setting Standards: Creating Architectural Blueprints and Architectural Solutions

Before setting specific architectural standards, it is necessary to decide, what these standards should encompass. Possibilities here are e.g.:

- The components (deployed and running sub-systems) a business application may consists of, and how these may communicate (connectors).
- The infrastructure software, which the components rely on.
- The hardware running the components.
- Development environments used for developing the respective software.

The EAMVS online survey [Bu08a] showed that the first two items are most important to practitioners.[1] Thereby, the first and the second item can be addressed by architectural blueprints and solutions. Understood this way, an architectural blueprint is an exemplary description of a software architecture in the component-and-connector viewtype according to Clements et al. [Cl02]. This leads to different possible notations for *defining* architectural blueprints:

- We propose V-Pattern Architectural Solution Definition (see page 208), which is based on the respective UML-notation in [Cl02].
- V-Pattern *Architectural Blueprint* (see page 315 in [Bu08a]) is a possible alternative to V-Pattern Architectural Solution Definition, but this pattern was evaluated in the EAMVS to be of minor importance.
- The architectural description language *ACME* [GMW97] is another possibility.

However, the description of the exemplary architecture in an architectural blueprint is technology-neutral. The specific technologies are set when an architectural solution is created based on a specific architectural blueprint, which assigns a *specific technology* to each so called *abstract technology* in the architectural solution. Using this approach is reasonable, because specific technologies change more often then abstract technologies. This distinction offers the possibility to define more stable architectural standards based on architectural blueprints. In this case architectural solutions can be seen as a way to document which specific technologies work well together.
Several aspects may influence which and how many architectural standards are offered.

[1] Ranked by practitioners regarding importance on a 1-5 Likert scale (5 is most important), they received an average rating of 4 or more.

The following arguments are in favor for architectural standards:

- Projects may choose an architecture and technologies they regard to be most suitable for the respective tasks, without having to "reinvent the wheel".
- Architectural standards document proven practices in combining technologies to fulfill certain tasks.
- Architectural standards may be used to reduce the heterogeneity of the application landscape.
- Knowledge about an additional architecture has to be kept available, if the business application does not conform to defined standards, at least as long as it is operated.
- Knowledge about technologies is only needed for allowed technologies.

In contrast the following arguments are against architectural standard:

- It may be easier and faster to develop a business application exactly satisfying its requirements without following the defined standard architecture.

The set of offered standards has to strike a balance between these effects.

A.1.1.6. Analyzing Standards: Analyzing Standard Conformity of Business Applications

First, create an overview of which business application uses which architectural solution and analyze it. For collecting this information, it is important to know that the employees operating a business application might not always be aware of its architecture. Thus, developers might have to be included into the data collection process. Of course, up-to-date architectural blueprint and solution definitions are a prerequisite for this task. Additionally, an understanding of the blueprints should exist among the developers. This can be facilitated by using V-Patterns like Architectural Solution Definition (see page 208).
The collected information should then be verified. Here also different possibilities apply, ranging from automated plausibility checks to manual reviews, which could be tied to visualization creation. If necessary, missing or possibly erroneous information has to be delivered in addition or corrected.
An Architectural Solution and Technology Mapping-diagram (see page 198) can provide background information about the existing architectural blueprints and solutions. It can give a first overview of the technologies included in a standard. This allows a first stage of the analysis: The set of standards might be too small (too restrictive) or too big (too permissive).
Next, analyze the application landscape to find business applications that do not belong to an architectural standard. This can e.g. be done by highlighting such business applications. For example, use Standards Conformity Exceptions (see page 211) and Architectural Standard Clustering (see page 101 in [Bu08a]). Standards Conformity Exceptions can indicate where architectural standards are met, where this is not the case, and where breaking the standard is specifically allowed.

Utilizing these two V-Patterns, the focus is likely to be on the business applications not conforming to the respective architectural standard. On the one hand, such business applications might be looked at specifically, considering e.g.:

- Does it require not to conform with the standard?
- How much costs are thus induced? Who bears these costs?
- Has the wrong standard been prescribed for the business application?

On the other hand, analyses can also focus on the totality of the non-conforming business applications, e.g. looking at:

- What do they have in common?
- Are the standards inadequate for important parts of the application landscape?
- Are there organizational units for which there are no means of enforcing the standards?

Especially an Architectural Standard Clustering-diagram (see page 101 in [Bu08a]) might be helpful in getting an impression of the importance of the different architectural solutions. A standard only existing to serve a small proportion of the business applications might need a special justification.
Breaking standards can e.g. be allowed if significant business success is tied to the possibility to have projects outside the respective standards. However, this introduces the issue of who receives the benefits derived from breaking the standard, and who bears the costs induced thereby.

A.1.1.7. Enforcing Standards: Deriving Measures for Increasing Homogeneity

Once architectural standards are set, measures for improving conformance have to be developed and discussed. Certainly, such measures are described in a detailed, textual way by the architectural standard control group. However, diagrams like V-Pattern Business Application Planning (see page 205) can give an overview of the changes in the application landscape due to a (specific) proposed measure.
Deriving measures involves finding the non-conforming business applications e.g. via analyzes as described above. Based on this, the reasons for the business applications non-conforming to the standards can be determined. This sets the ground for deciding, whether a specific business application currently not conforming to the standards has to be changed. Subsequent points might be important in such a discussion:

- Has the wrong standard been set for a business application? In this case, the standard should be changed.
- If there is excessive cost for standard conformance, an exception could be sensible.
- If the benefit of conforming to the standard cannot be realized in a specific situation, this might also be a reason for an exception.

If it is decided that one or more business applications have to be changed, the respective proposal has to be created, and can then be entered into project portfolio management, if available, or an equivalent management process.

A.1.1.8. Evaluate Standards: Find Standards which have to be Adapted

The steps *setting standards*, *analyzing standards*, and *enforcing standards* are not sufficient for a continuous management approach. As requirements and technologies change over time, the standards, which are currently in use, have to be evaluated concerning their applicability in the future.
There are different ways to achieve such an evaluation. A simple approach would be to count how often a certain standard is in use. If this value is below a certain threshold, an in depth analysis should initiated why the standard is only seldomly used. One reason could be that the standard has been created for specific requirements. In this case the standard need not be revised. Another reason could be that the standard uses a technology which is no longer considered to be state of the art. In this case the standard should be changed or retired.
More sophisticated approaches, like technology roadmaps defining the upgrade paths for technologies that may be used in standard definitions could also be used but require higher efforts to be realized.

A.1.1.9. Implementation

In order to implement this M-Pattern within an organization it is very important to create the required governance structures and to involve the right people, meaning that it is required to establish a group of people, which are able to define the required architectural standards. This group of people is called the *Architectural Standard Group* and usually recruits its members from the software architect and from the enterprise architect group of the company, as knowledge about technologies and their interrelations is required. See *Architect Also Implements* in [CH04] for detailed information about this topic.
Only defining the standards usually is not enough as it is required that these standards are controlled and if necessary are enforced. This should be done by a special group of enterprise architects, the *Architectural Standard Control Group*, which should be incorporated in every project exceeding a certain project cost limit. The limit is depending on the size of the company and the budget available for EA management.
A third group of people is required for escalation. This group, the *Architectural Standard Board*, should be on board level and should incorporate members of the business as well as of the IT part of the company. If no consensus between the project and the architectural standard group is possible, the architectural standard board has to decide if breaking a standard is allowed or not. The enforcement of this decision may also influence the budget of the project under consideration.

A.1.1.10. Known Uses

The approach documented in M-Pattern Standard Conformity Management is in use in the following companies:

- BMW Group
- HVB
- Enterprise Architecture Management Tool Survey 2008 / SoCaStore (sebis)

The approach documented in this M-Pattern can be used in the following EA management tools

- ARIS (IDS Scheer AG)
- planningIT (alfabet AG)

The pattern is also known as *Management of Architectural Standards* and *Blueprint Conformity Management*.

A.1.1.11. Consequences

It is helpful, if not necessary for the M-Pattern, that architectural solutions are *boundary objects* between enterprise architects and software architects. These two domains need an aligned understanding of the architectural standards, enabling them to efficiently communicate in using them.
A boundary object is an object, which allows members of different communities to build a shared understanding in respect to certain things. Boundary objects are interpreted differently by the different communities, and realizing as well as discussing these differences leads to a shared understanding [SG89, St99].
If architectural standards are to be beneficial, there has to be an entity having both power and commitment to enforce the standards as described in the implementation section. This entity is then likely to be also in charge of allowing exceptions from the standards. Thereby, it has to address the problem that the benefit and the costs of conforming to blueprints and solutions occur in different places:

- It is likely that the costs for conforming to an architectural standard occur directly in the development team or operators responsible for the respective application (in the short term). Costs can also occur at users, if a conforming business application is less suitable, e.g. due to decreased performance, which is not improvable without a highly specialized architecture.
- The benefit of increased homogeneity are likely to be of a more long-term nature, and occur primarily with the IT departments responsible for operating and developing business applications. However, if more efficient development can lead to a more swift project execution, business might be able to benefit from a reduced time to market.

If the decision process is not able to balance this on a cross-organizational level, it might happen that decisions are locally optimal for specific organizational units, but suboptimal for the organization as a whole. An example for an approach trying to balance the aspects is allowing deviations from the standard, but estimating the future effort of fixing issues created by this, and imposing a respective fee on the organizational unit that demands breaking the standard.
Another consequence is that defined architectural standards have to be maintained and evolved to keep up with new technologies, developments, etc. On the one hand this has a positive effect as there is a need to continually rethink defined solutions resulting in a potential improvement of the defined standards. On the other hand investments are needed to be able to maintain and evolve the standards, which have to be in balanced with the potential savings.

A.1.1.12. See Also

In order to support the implementation of M-Pattern Standard Conformity Management the following V-Patterns should be considered:

- Architectural Standard Clustering (see page 101 in [Bu08a])
- Architectural Solution and Technology Mapping (see page 198)
- Business Application Planning (see page 205)
- Architectural Solution Definition (see page 208)
- Standards Conformity Exceptions (see page 211)
- The architectural description language ACME [GMW97]

A.2. Viewpoint Patterns (V-Patterns)

This section contains the following selection of V-Patterns.

A.2.1. Architectural Solution and Technology Mapping

Architectural Solution and Technology Mapping consists of a table containing the technologies used in architectural solutions.

A.2.1.1. Example

The application landscape of SoCaStore has evolved over the years to fulfill new business demands as quickly as possible. To achieve the required speed to support these demands new business applications have been developed without caring about selecting technologies for the new business application or defining architectural solutions. This approach resulted in a lack of information and knowledge about the technologies used in the company. In order to change this situation visualizations are needed to give an overview about the dependencies between the operated business applications and the technologies they are built upon.

A.2.1.2. Context

Getting and maintaining an overview about the technologies, which build up architectural solutions is difficult in large companies.

A.2.1.3. Problem

You want to reduce costs and security risks by limiting the number of technologies used to implement business applications. To reduce their number, you first have to know what technologies are in use and where.
How do you visualize technology usage of business applications in a concise manner?

The following *forces* influence the solution:

- **Impact Analysis:** How do you get easy visual feedback of impact analysis?
- **Popular Technologies:** How do you spot popular technologies or trends, as these technologies are candidates for future architectural blueprints?
- **Migration Issues:** How do you detect problems in migrating from one version of a technology to another or in the evolution of a technology?
- **Technology Compability:** How do you get an overview about technologies that may be used in combination to prevent compability problems?

A.2.1.4. Solution

This view consists of a table containing the technologies used in an architectural solution, e.g. a 3-tier architecture. Thereby, an "X" in a table cell symbolizes the usage relationship. It may be used in different ways. At first you may get an overview about the different technologies used within a company, together with the information, in which architectural solution the technologies are utilized.

		Used Technologies						
		Apache 2.0.53	Bea Weblogic	DB2 6.0	IE 6.0	Oracle 9i	Proprietary Fat-Client	Tomcat 5.1
Architectural Solutions	4-tier architecture A	x			x	x		x
	4-tier architecture B	x	x		x	x		
	2-tier architecture A			x			x	
	2-tier architecture B					x	x	
	3-tier architecture		x			x		x

Figure A.2.: Exemplary view for V-Pattern Architectural Solution and Technology Mapping

At second you can use the V-Pattern to perform impact analysis. Therefore, you select a technology, which will e.g. be changed or replaced, and you can then see the affected architectural solutions.
As the number of different technologies and architectural solutions in use within a company may be high it may be useful to filter the information visualized, e.g. to select a technology and fade out all architectural solutions, which do not use it, in order to support the user in performing the impact analysis.

A.2.1.5. Implementation

This V-Pattern can be implemented in a spreadsheet tool or, if a graph representation (see variants section) is chosen, in a graph layout tool. If filtering should be used, than this functionality should be supported by the tool.

A.2.1.6. Variants

Different variants for this V-Pattern exist. The information shown in Figure A.2 could also be shown as a simple textual report, listing the technologies for an architectural solution. Another possible visualization would be a simple graph, where technologies and architectural solutions are represented by nodes and the usage of a technology in an architectural solution is visualized by an edge connecting the respective nodes.
The same kind of viewpoint can be created for architectural blueprints and abstract technologies. In these cases the same alternatives, textual listing, graph visualization, etc. apply.

A.2.1.7. Known Uses

The following companies use this V-Pattern:

- HVB

Views according to this V-Pattern can be created, e.g. using the following EA management tools:

- alphabet (planningIT AG)
- ARIS (IDS Scheer AG)
- System Architect (IBM)
- SoCaTool (sebis)

A.2.1.8. Consequences

The benefit of this V-Pattern is its simplicity. A simple table or graph is sufficient to address the problem described in the problem section. On the one hand these kinds of visualizations can easily be created on the other hand they can very intuitively be used to reduce the number of technologies, if variants or version differences can be eliminated. Additionally, you can get a gist of the impact of such a technology elimination and it is easier to spot the impact of required technology changes, i.e. because of security issues.

A.2.1.9. See Also

This V-Pattern may be useful when using M-Pattern Standard Conformity Management (see page 189). The visualized information is based on I-Pattern Technology Usage (see page 215).

A.2.2. Business Application and Organizational Unit Cluster Map

Business Application and Organizational Unit Cluster Map visualizes relationships between business applications and organizational units by using the concept of clustering.

A.2.2.1. Example

The application landscape of SoCaStore has continually grown since the foundation of SoCaStore. In the next few months a new subsidiary should be established in Hong Kong, which demands for an appropriate IT support. Using already existing business applications is a solution, which is time- and cost-saving. In order to prepare the business applications for their new tasks it is important to know where they are hosted, who uses them, who is responsible for them, etc. Unfortunately, this overview about the application landscape is not available through the continually growth of SoCaStore and now has to be regained.

A.2.2.2. Context

In an enterprise with a large number of business application it is hard to judge who is responsible for running them or who benefits or suffers from changes applied to them.

A.2.2.3. Problem

Relationships between business applications and organization units are of importance, e.g. when trying to analyze and determine responsibilities, utilizations, etc. for business applications.
Which relationships exist between business applications and organizational units and how can you visualize them?

The following *forces* influence the solution:

- **Visualize Responsibilities:** How do you visualize responsibilities for business applications in order to explicate them?
- **Visualize Usage:** How do you visualize the usage of business applications?.
- **Visualize Operation:** What is a distinct and easy to understand visualization to show where business applications are hosted?

A.2.2.4. Solution

This V-Pattern belongs to the software map type *Cluster Map*, which uses the concept of grouping (clustering) of elements in a visualization to express a relationship between them. The positioning of the different clusters is of minor importance as it does not transport any semantic information, but may be used to improve recognition like organizational unit headquarter is always positioned in the top left corner.
In this V-Pattern a cluster map like viewpoint is used to group business applications in organizational units. Figure A.3 exemplarily visualizes a hosting relationship. This

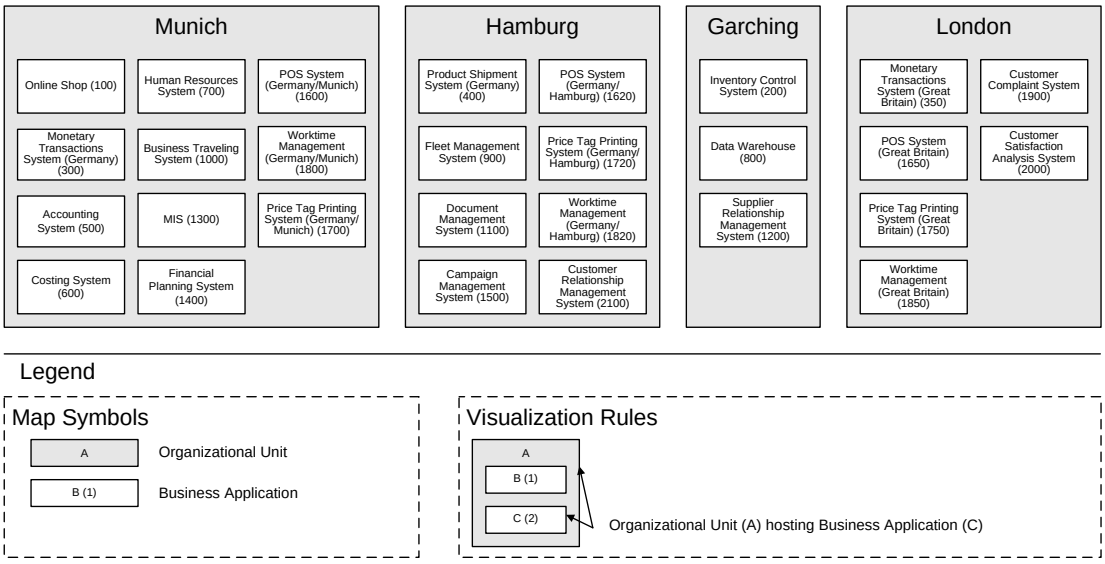

Figure A.3.: Exemplary view for V-Pattern Business Application and Organizational Unit Cluster Map

is only one possible semantic for the relationship between business applications and organizational units, additional variants are described in the variants section of this V-Pattern.
A business application may appear multiple times within a view corresponding to this V-Pattern, e.g. if it is used by multiple organizational units.
Different kinds of usages are supported by this V-Pattern. First of all it is possible to give an overview, about the as-is situation, or about planned and target scenarios of the application landscape, when incorporating the aspect of time. Secondly, it is possible to do extended analyzes, like e.g. impact analysis concerning redundantly hosted business applications, etc.

A.2.2.5. Implementation

Views according to this viewpoint can be created manually by any drawing tool, like e.g. Microsoft PowerPoint. As manual creation is time consuming and error prone it is advised to use a tool, like the ones listed in the implementation section to automatically generate the visualization.

A.2.2.6. Variants

Additional variants for this V-Pattern exist, as different semantics are possible for the relationship between business applications and organizational units. Three exemplary ones are listed below:

- Organizational unit *hosts* business application
- Organizational unit *uses* business application
- Organizational unit *is responsible for* business application

Each of these possibilities results in a variant of the V-Pattern. Thereby, the clustering of elements is used to represent the different relationships.

A.2.2.7. Known Uses

The following uses are known:

- Enterprise Architecture Management Tool Survey 2008 / SoCaStore (sebis)
- Klinikum der Universität München
- Munich Re

Views according to this V-Pattern can automatically be created, e.g. using the following EA management tools:

- alphabet (planningIT AG)
- ARIS (IDS Scheer AG)
- Iteraplan (Iteratec)
- SoCaTool (sebis)

This pattern is also known as *Using Relationship Cluster Map*, *Hosting Relationship Cluster Map*, *Responsibility Relationship Cluster Map*.

A.2.2.8. Consequences

A benefit of this V-Pattern is that it is a good starting point for EA management activities and supports many different analyzes. Two exemplary analyzes are e.g. *find organizational units with(out) intensive relationships to business applications* and *find responsibilities for business applications*.
Views according to this V-Pattern can easily be explained and used, and contain a lot of valuable information about the current situation of the application landscape. In addition they may also be used for planning aspects.
Additionally, the creation of views corresponding to this V-Pattern is simple and can even be done manually in some kind of drawing tool in the last resort. Furthermore, the amount of information that has to be collected is limited.
Another benefit is that this kind of visualization can easily be enriched by additional layers providing additional information, like costs for maintaining the business applications, connections between business applications, etc. Refer to the next section for further details.

A.2.2.9. See Also

The V-Pattern is based on information according to I-Pattern Business Application and Organizational Unit Relationship (see page 218) and its variants.
Additionally, V-Pattern Business Application and Organizational Unit Cluster Map is the basis for all V-Patterns, which rely on visualizing the relationship between business applications and organizational units together with other information. The following list provides an overview about these V-Patterns.

- Business Application Planning (see page 205)
- Standards Conformity Exceptions (see page 211)

A.2.3. Business Application Planning

Business Application Planning visualizes changes to business applications or the introduction of new ones using a color coding. It can be used to perform application landscape planning.

A.2.3.1. Example

SoCaStore wants to start an initiative to consolidate its application landscape. Therefore, existing business applications have to be modified or retired and new ones have to be introduced. This is only possible, if an overview about the application landscape and the planned changes is available.

A.2.3.2. Context

In a large application landscape the future development, e.g. the introduction of a new business application or the phase out of existing one, has to be planned.

A.2.3.3. Problem

You want to plan the evolution of the business applications, which make up the application landscape. To do this, you need to know which business applications have to be introduced, changed, shut down, or are not changed at all. Additionally, the relationships between the business applications and the organizational units are of importance, e.g. to find the responsible person for an organizational unit with a lot of upcoming changes in order to discuss the consequences of these changes.
How do you visualize the life cycle status of the business applications in order to get a quick overview?

The following *forces* influence the solution:

- **Affected Organizational Units:** How do you identify organizational units where a lot of changes take place and which are not at all affected?
- **Planning Conflicts:** What conflicts exist in the current development plan of the application landscape?
- **Explicate Planned Changes:** How can effects of future changes to business applications be explicated in a clear and simple way?

A.2.3.4. Solution

This V-Pattern uses the concept of a cluster map showing a relationship between business applications and organizational units, based on the V-Pattern Business Application and Organizational Unit Cluster Map and its variants.
The exemplary visualization in Figure A.4 depicts the hosting relationship. In addition to this relationship the V-Pattern indicates, which business applications are to be changed, by highlighting these business applications. Normally, these changes can be traced back to a project, offering information about the type of change that has to be performed,

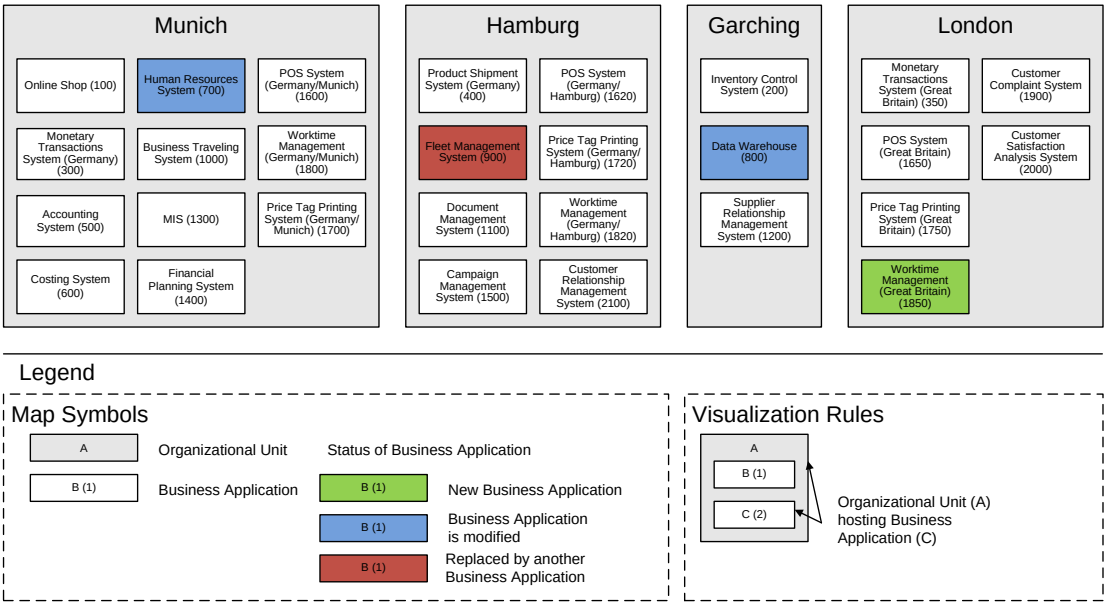

Figure A.4.: Exemplary view for V-Pattern Business Application Planning

e.g. that a business application has to be replaced by another one. The type of change is indicated by different colors, like explicated in the legend of Figure A.2.3.

A.2.3.5. Implementation

The information about the type of change that that has to be performed on the business application should be visualized on a different layer than the relationship between organizational units and business application to be able to profit from the layering principle[2]. When there is a demand to utilize the layering concept, it is advised to use a tool supporting this functionality.

A.2.3.6. Variants

There are many different semantics for the relationship between business applications and organizational units. Each of them constitutes a different variant of this V-Pattern. Additionally, the information, which business applications are affected by changes can be visualized on a different software map type, like a Cartesian map, in particular a process support map. V-Pattern *Process Support Map* (see page 105 in [Bu08a]) gives more information about this kind of software map type. The additional relationship to business processes offers the possibility for extended analyses, like an analysis which business processes are primarily effected by the planned changes and which ones do not have to be considered.
The variants mentioned above may also consider a time aspect, meaning that the visualization of the application landscape will look different, if it e.g. shows the status for today or the status in a year from now.

[2]See [Er06] for more details on the layering principle.

A.2.3.7. Known Uses

The following uses are known:

- Enterprise Architecture Management Tool Survey 2008 / SoCaStore (sebis)

Views according to this V-Pattern can automatically be created, e.g. using the following EA management tools:

- alphabet (planningIT AG)
- ARIS (IDS Scheer AG)
- SoCaTool (sebis)

A.2.3.8. Consequences

Normally, a business application can only be changed by a project, resulting in a need to also collect information about the project itself and not only about the scheduled changes for the business applications.
If tracing back the changes on a business application to a project is needed, it is advisable to have additional information about this project available, e.g. in a textual form, for further analyses.
Considering time aspects demands for additional information, e.g. about the start time and duration of a project changing a business application. Therefore, another I-Pattern is needed to fulfill this additional demand.

A.2.3.9. See Also

Creating views based on this V-Pattern requires to collect information according to I-Pattern Planned Project Effects (see page 206 in [Bu08a]) to visualize which business applications have to be changed due to which projects. Additionally, information about the relationships between the business applications and the organizational units can be gained by I-Pattern Business Application and Organizational Unit Relationship (see page 218) or its alternatives.

A.2.4. Architectural Solution Definition

Architectural Solution Definition uses an UML 2.0 object diagram to visualize an architectural solution and the technologies used.

A.2.4.1. Example

Due to the uncontrolled evolution of SoCaStore's business applications a multitude of different architectures are in use and are planned for future developments. This should be prevented in the future by providing defined and obligatory architectural standards. In order to define this architectural standards in a standardized way a defined notation has to be used.

A.2.4.2. Context

Defining architectural standards and maintaining them is difficult, if various architectural standards and different notations are in use.

A.2.4.3. Problem

You want to increase homogeneity of business applications' architectures by using defined architectural standards. In order to achieve this goal you have to decide for an obligatory notation for defining and maintaining architectural standards.
What visualization should be used to define and manage architectural solutions for business applications?

The following *forces* influence the solution:

- **Standard Definition Overview:** How do you get an overview about defined architectural standards?
- **Plain Notation:** What notation for architectural standards is distinct and easily understandable?

A.2.4.4. Solution

V-Pattern Architectural Solution Definition uses the notation of an UML 2.0 object diagram to visualize the structure of an *architectural solution*. An architectural solution includes the allowed technologies, like e.g. Apache 2.0, Internet Explorer 6.0, etc. and the allowed connectors between these technologies, e.g. an http connection used between the Internet Explorer 6.0 and the Apache 2.0. An example of a view defining an architectural solution is given in Figure A.5.

A.2.4.5. Implementation

Views, which are based on this V-Pattern may be created using an UML modeling tool, in order to use the syntactic checking incorporated in the tools. When using a variant not relying on the UML notation any kind of drawing tool may be used.

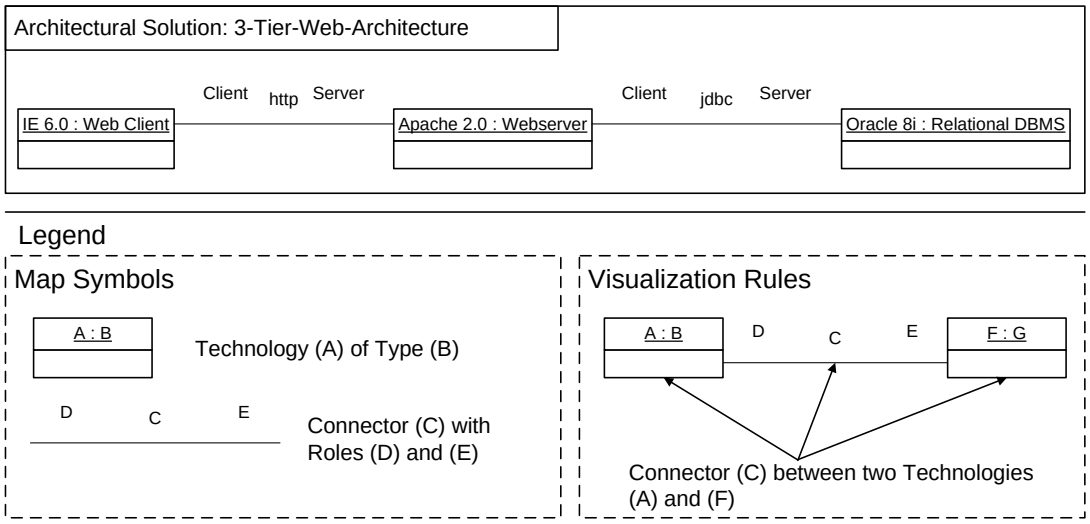

Figure A.5.: Exemplary view for V-Pattern Architectural Solution Definition

A.2.4.6. Variants

A variant of this V-Pattern is concerned about *Architectural Blueprints*. Thereby, the architectural solution is an instantiation of an architectural blueprint, which defines which abstract technologies, e.g. a web client, a web server, etc. may be used and in which combination.
Both variants can also be combined, meaning that information about technologies and about abstract technologies is shown within one visualization. See consequence section for more information.
A second variant would be to abstain from the notation of UML 2.0 object diagram. Whereas, this has the advantage, that the views can be drawn with any visualization tool, this leads to the problem that drawing without defined syntactics and semantics may result in misleading views.

A.2.4.7. Known Uses

The following uses are known:

- BMW

Views according to this V-Pattern can automatically be created, e.g. using the following EA management tools:

- ARIS (IDS Scheer AG)
- Rational Software Architect (IBM)
- System Architect (IBM)

This pattern is also known as *Architectural Blueprint Definition*.

A.2.4.8. Consequences

Visualizing information about an architectural solution and the associated blueprint in one visualization may lead to large and hard to understand views. Therefore, it may be reasonable to omit information about the architectural blueprint or the architectural solution.
A benefit of this V-Pattern is that it is easily understandable by different groups, like software architects, enterprise architects, etc. within the company this improves communication between them. Besides the easy understandability of visualizations according to this V-Pattern, they are extensive enough to avoid misleading interpretation and utilization.

A.2.4.9. See Also

Another V-Pattern, called *Architectural Blueprint* (see page 315 in [Bu08a]) is also focused on defining architectural blueprints utilizing but also incorporates the concept of tiers to separate different layers of the architecture.
Creating views based on this V-Pattern requires to collect information according to I-Pattern Technology and Connector Usage (see page 223 in [Bu08a]) to visualize, the relationships between technologies, connectors, and abstract technologies.

A.2.5. Standards Conformity Exceptions

Standards Conformity Exceptions shows which business applications conform to architectural standards, and where exceptions from these standards have been allowed. This information is combined with information about the relationships between business applications and organizational units.

A.2.5.1. Example

SoCaStore is using the concept of architectural blueprints and architectural solutions for a few months now, but the effects of this concept, like standardization of the application landscape, etc., have not yet been analyzed. To conduct such analyzes visualizations are needed, which not only show the standard conformity of the application landscape, but also the allowed exceptions.

A.2.5.2. Context

It is hard to analyze the standards conformity of business applications if the application landscape exceeds a certain size. Usually this happens if more than 100 business applications have to be considered. It gets even worse if exceptions to defined standards have to be regarded.

A.2.5.3. Problem

You want to reduce costs by increasing the degree of standardization of the application landscape. To achieve this you first have to get an overview of the application landscape and its current use of standards. Before you can begin to adopt the business application not conforming to standards, you have to consider whether there should be exceptions. **How do you visualize an overview about the standardization of the application landscape, also including information about allowed exceptions?**

The following *forces* influence the solution:

- **Exception Overview:** How to get an overview about allowed exceptions to architectural standards?
- **Identify White Spots:** How to identify organizational units where there is no information available about the standardization of business applications?
- **Identify Outstanding Organizational Units:** How to find organizational units with an exceptionally high amount of (not) standardize business applications?

A.2.5.4. Solution

This V-Pattern uses the same concept – a cluster map – as its base, as V-Pattern Business Application and Organizational Unit Cluster Map (see page 201), resulting in the same variety of semantics that can be used. In this case a layer is added to the cluster map showing, which business applications conform to architectural standards, and where exceptions from these standards are tolerated. Using the cluster map concept

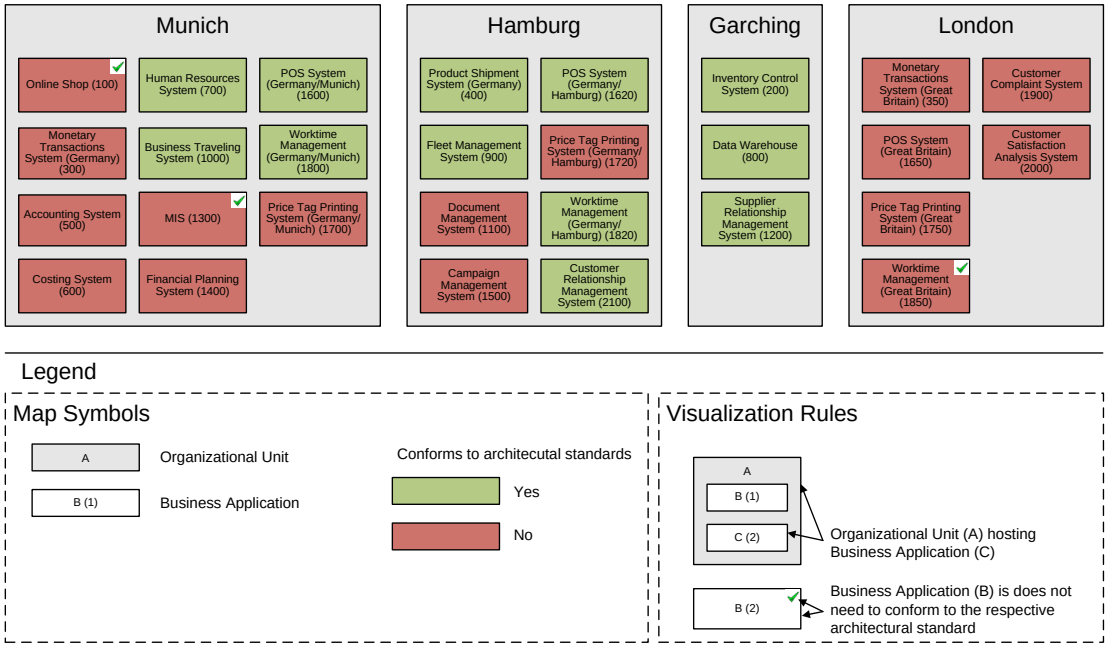

Figure A.6.: Exemplary view for V-Pattern Standards Conformity Exceptions

here is favorable as it provides a good and intuitive overview about the relationships described before.
Figure A.6 shows these relationships via an exemplary cluster map, based on the hosting relationship between business applications and organizational units.
Conformance to architectural standards is visualized by colors, permitted exceptions to these standards are marked by a checkmark.

A.2.5.5. Implementation

You should use a tool supporting layers for implementing this V-Pattern. This offers the possibility to show the information about the type of change that has to be performed on the business application on a different layer than the conformance to architectural standards. In this case the amount of information shown in the view can be adapted to specific requirements by showing or hiding layers.

A.2.5.6. Variants

As already mentioned in the solution section different semantics for the relationship between business applications and organizational units exist. Each of them constitutes a different variant of this V-Pattern. See V-Pattern Business Application and Organizational Unit Cluster Map (see page 201) for more information.
Additionally the information, which business applications are affected by changes can be visualized on a different software map type, like a *Cartesian Map*, in particular a process support map. V-Pattern *Process Support Map* (see page 105 in [Bu08a]) additionally

offers the possibility to analyze the standardization of business applications in respect to business processes.
In contrast to Figure A.6 it would also be possible to visualize the exceptions to architectural standards on an additional layer. This makes it possible to hide this information as long as it is not needed, leading to an easier to interpret view.
If the information about exceptions is not important for analyses within a company then omit it, because it may lead to overwhelming visualizations resulting in misinterpretations.

A.2.5.7. Known Uses

The following uses are known:

- Enterprise Architecture Management Tool Survey 2008 / SoCaStore (sebis)

Views according to this V-Pattern can automatically be created, e.g. using the following EA management tools:

- planningIT (alfabet AG)
- SoCaTool (sebis)

A.2.5.8. Consequences

Documentation for an exception to an architectural standard should explain why the exception is tolerated, e.g. in a separate document, in order to support additional analyses and next steps. This nevertheless comprises the disadvantage that the required information has to be collected and maintained.
If the information about allowed exceptions to architectural standards is not of importance, it should not be visualized, resulting in a reduced amount of information that has to be collected to be able to create the visualization.
A benefit of this V-Pattern is that organizational units, or business processes in case a process support map is used, with a high number of business applications not conforming to architectural standards can easily be found and the additionally included information about the allowed exceptions facilitates the identification of business applications where you should start to increase the standardization.

A.2.5.9. See Also

Creating views based on this V-Pattern requires to collect information according to I-Pattern Architectural Solution Conformance (see page 221) to visualize, which business applications do, or do not conform to architectural standards, together with the information where exceptions are tolerated. Additionally, information about the relationships between the business applications and the organizational units can be gained by I-Pattern Business Application and Organizational Unit Relationship (see page 218) or its alternatives.

A.3. Information Model Patterns (I-Patterns)

This section contains the following selection of I-Patterns.

A.3.1. Technology Usage

Technology Usage shows how information about the technologies used in an architectural solution, can be stored.

A.3.1.1. Example

SoCaStore wants to start an initiative to reduce the number of technologies used within the company. As a first step the technologies, which are currently in use, have to be collected. In order to store this information for future usage, an information model has to be created as an implementation basis for a repository.

A.3.1.2. Context

Getting an overview about which technologies are used by which architectural solution is a good starting point for homogenization of the application landscape. Additionally, this information may be used for documenting the current structure of architectural solutions but also to plan future ones or to document proven practice.

A.3.1.3. Problem

You want to reduce costs (licensing, maintenance, etc.), increase homogenization for the technologies used in a company or to document proven practice, e.g. which combination of technologies work together well.
What is a proven way to store and maintain information about technologies used in architectural solutions?

The following *forces* influence the solution:

- **Minimum Effort:** How can the effort to document the technologies in use be minimized?
- **Usability:** How can impact analysis concerning technologies be supported?

A.3.1.4. Solution

Figure A.7.: Information model fragment for I-Pattern Technology Usage

The solution for the problem described above is based on two entities and one relationship, which are defined as follows:

- ArchitecturalSolution: A concrete stack of corresponding technologies, which are intended to be used together in realizing business applications, together with additional information on how to integrate these technologies into an complex architecture. Combining technologies together to an architectural solution among others indicates, that components created from the technologies are technically suited for interaction and integration.
- Technology: A Technology represents a technical constituent of a business application, ranging from an implementation framework or platform to a database management system or user interface toolkit. Exemplary technologies may be "Apache 2.0.53" or "Oracle 9.2i".
- ArchitecturalSolution uses Technology: The association *uses* indicates, which architectural solution uses which technologies.

A.3.1.5. Implementation

This I-Pattern may be implemented in any tool (e.g. database management system, EA management tool, etc.) or format (e.g. spreadsheet, XML, etc.) able to store multiple entities and a single relationship.

A.3.1.6. Variants

The information model fragment shown in Figure A.7 may be extended e.g. by additional attributes, like licensing costs for technologies, or more advanced concepts like life cycles for technologies, as well as for architectural solutions.

A.3.1.7. Known Uses

The following uses of this I-Pattern are known:

- Enterprise Architecture Management Tool Survey 2008 / SoCaStore (sebis)

An equivalent information model fragment is included in the following EA management tools:

- ARIS (IDS Scheer AG)
- planningIT (alfabet AG)
- SoCaTool (sebis)

A.3.1.8. Consequences

A liability of this I-Pattern is the amount of information that has to be collected to be able to perform reasonable analyses, as an architectural solution is typically built up by four or more technologies. If standard conformity analysis is not part of the selected EA management approach, this I-Pattern should be omitted. Validity of the information is another critical aspect of this I-Pattern. Technologies or architectural solutions may already have changed, e.g. new versions have been introduced, before the information about them can beneficially be used.
A benefit of this pattern is that it presents an easy way to document proven practice about which technologies can be used well together. This information may later be used when planning new business application or defining new architectural standards.
Another benefit is the support for impact analyses concerning technologies, e.g. if a technology has to be changed it is easy to find architectural solutions, which are affected by this change.

A.3.1.9. See Also

V-Pattern Architectural Solution and Technology Mapping (see page 198) may be utilized to perform analyses on information stored according to this I-Pattern.

A.3.2. Business Application and Organizational Unit Relationship

Business Application and Organizational Unit Relationship shows how information about business applications and their relationships to organizational units can be stored.

A.3.2.1. Example

SoCaStore wants to start its EA management initiative with a minimal effort as the benefit of an EA management is unclear. As a first step a minimal information model should be created, which provides sufficient information to perform an EA management show case, but for this purpose only limited information should be collected. An important aspect is, that the information collected should be reusable for the next steps in introducing a more matured EA management.

A.3.2.2. Context

Getting an overview about which business applications are in use in the company, together with their relationships to organizational units is a very important information in an EA management approach.

A.3.2.3. Problem

You want to know more about the relationship between your business applications and organizational units, e.g. to define responsibilities, to document usages or to identify redundancies.
How can information about the relationships between business applications and organization units be collected and stored?

The following *forces* influence the solution:

- **Minimum Effort:** How can the effort to document the relationship between business applications and organizational units be minimized?
- **Business Application Documentation:** How to document which business applications are in use in the company?
- **Initial Showcase:** What is a good starting point for an EA management showcase?

A.3.2.4. Solution

Figure A.8.: Information model fragment for I-Pattern Business Application and Organizational Unit Relationship

This I-Pattern consists of two entities *BusinessApplication*, *OrganizationalUnit*, and one relationship *hosts* and is exemplarily visualized in Figure A.8. The hosts relationship shown in this figure is only one possible semantic for this relationship, additional variants are described in the variants section of this I-Pattern.
The entities and relationships can be defined as follows:

- OrganizationalUnit: An organizational unit represents a subdivision of the organization according to its internal structure. A possible example are the entities showing up in an organigram.
- BusinessApplication: A software system, which is part of an information system within an organization. An information system is therein according to [Kr05] understood as a socio-technological system composed of a software system (i.e. the business application), an infrastructure, and a social component, namely the employees working with the system. An information system is further described as contributing to the business process support demanded by the organization.
- OrganizationalUnit hosts BusinessApplication: The association *hosts* indicates, which organizational unit is hosting a business application.

A.3.2.5. Implementation

This I-Pattern may be implemented in a spreadsheet tool or in some kind of database system. When using different variants of this relationship within an information model for EA management, the semantics of the different relationships should be explicitly defined in order to avoid confusion.
Using only one relationship to implement different variants should be avoided as this leads to problems, when analyzing information stored according to this information model.

A.3.2.6. Variants

Additional variants exist for this I-Pattern. The relationship between business applications and organizational units may thereby have different semantics, with three of them listed below:

- Organizational unit *hosts* business application
- Organizational unit *uses* business application
- Organizational unit *is responsible for* business application

Figure A.9.: Information model fragment for variant of I-Pattern Business Application and Organizational Unit Relationship based on the uses variant

Each of these different semantics results in a variant of this I-Pattern. An example for the *uses* variant is shown in Figure A.9.

The following definitions specify the uses and the responsible for relationship.

- OrganizationalUnit uses BusinessApplication: The association *uses* indicates, which organizational unit uses which business application.
- OrganizationalUnit responsible BusinessApplication: The association *responsible for* indicates, which organizational unit is responsible for which business application.

A.3.2.7. Known Uses

The following uses of this I-Pattern are known:

- Enterprise Architecture Management Tool Survey 2008 / SoCaStore (sebis)
- Klinikum der Universität München

An equivalent information model fragment is included in the following EA management tools:

- ARIS (IDS Scheer AG)
- planningIT (alfabet AG)
- SoCaTool (sebis)

A.3.2.8. Consequences

This I-Pattern may sound trivial, but as the question where to start an EA management approach is not a trivial one, this pattern is of importance.
A benefit of this I-Pattern is that it is a good starting point when building an EA management information model. The amount of information that has to be collected is limited, but it offers extensive possibilities to perform first analyzes of the application landscape, which can then be used to show the benefits, like e.g. documentation of responsibilities, of the selected EA management approach. Additionally, information collected and stored according to this I-Pattern can easily be reused in an extended and a more mature EA management approach.
If this I-Pattern should be integrated in an information model coping with information about BusinessApplicationVersions for a BusinessApplication, one should consider to change the BusinessApplication in this I-Pattern to the BusinessApplicationVersion.

A.3.2.9. See Also

This I-Pattern is the basis for V-Pattern Business Application and Organizational Unit Cluster Map (see page 201) and its variants.

A.3.3. Architectural Solution Conformance

Architectural Solution Conformance shows how information about business applications and their conformity to architectural solutions, can be stored.

A.3.3.1. Example

SoCaStore wants to start an initiative to analyze the status of the application landscape concerning the conformance of business applications to architectural solutions. Only to look for business applications not conforming to defined solutions seems not to be sufficient, as there also exist allowed exceptions to this allegation. Furthermore, in this initiative the challenge to cope with incomplete information regarding the standard conformance of some business applications exists.

A.3.3.2. Context

Managing information about which business application conforms to which architectural solution or why it does not conform to any, is difficult in a large application landscape.

A.3.3.3. Problem

You want to keep track about the status of the business applications concerning their solution conformity.
How should an information model look like, to be able to collect and store information about architectural solution conformance?

The following *forces* influence the solution:

- **Minimum Effort:** How can the effort to document architectural solution conformance be minimized?
- **Unknown Information:** What has to be included in an information model need to be able to differentiate between business applications where no information about its conformance is available and business applications not conforming to architectural solutions?
- **Document Exceptions:** How can exceptions to architectural standards be documented?
- **Document Nonconformity:** What is needed to document business applications not conforming to architectural solutions?

A.3.3.4. Solution

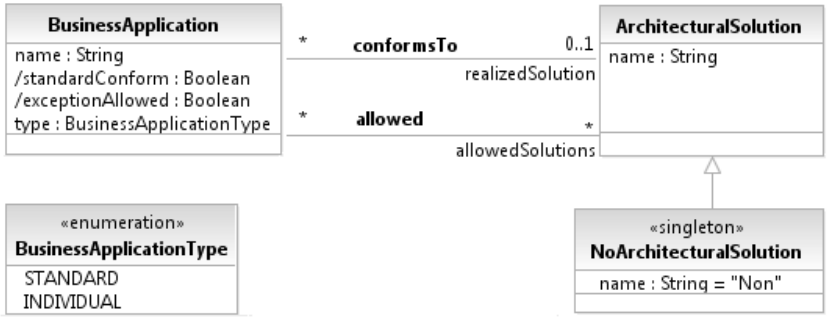

Figure A.10.: Information model fragment for I-Pattern Architectural Solution Conformance

The solution for the problem described above is based on three entities and three relationships, which are defined as follows:

- ArchitecturalSolution: A concrete stack of corresponding technologies, which are intended to be used together in realizing business applications, together with additional information on how to integrate these technologies into an complex architecture. Combining technologies together to an architectural solution among others indicates, that components created from the technologies are technically suited for interaction and integration.
- BusinessApplication: A software system, which is part of an information system within an organization. An information system is therein according to [Kr05] understood as a socio-technological system composed of a software system (i.e. the business application), an infrastructure, and a social component, namely the employees working with the system. An information system is further described as contributing to the business process support demanded by the organization.
- NoArchitecturalSolution: This entity represents the *Non-Solution*, i.e. it means, that an associated business application does not follow or does not need to follow any architectural solution.
- BusinessApplication conformsTo ArchitecturalSolution: The association *conformsTo* indicates, in accordance to which architectural solution a business application is actually realized. Such a solution might be the singleton instance of the *NoArchitecturalSolution*, thereby indicating, that no standard solution has been used. Further, no such information might be present, described by the absence of an associated solution.
- BusinessApplication allowed ArchitecturalSolution: The association *allowed* explicates, which architectural solutions are per standard available for realizing the corresponding business application. Therein, the non-solution, as reflected by the

singleton instance of the class *NoArchitecturalSolution*, is used to represent, that a business application does not need to conform to any architectural solution. This is especially necessary, to distinguish between the prescription of no solution vs. the absence of a prescription of that kind, i.e. missing data.

- BusinessApplicationType: The BusinessApplicationType is used to model, whether a business application has been developed as a piece of individual software or is a bought standard solution.

For determining information about the standard conformance of the business applications, such as displayed in Figure A.6, the derived attributes *standardConform* and *exceptionAllowed* are used. The values of these attributes are derived by expressions similar to the following[3]:

$$standardConform = \begin{cases} null & \text{for} & (realizedSolution = null) \vee \\ & & (allowedSolutions = null) \\ true & \text{for} & realizedSolution \in allowedSolutions \\ false & \text{for} & realizedSolution \notin allowedSolutions \end{cases}$$

respectively

$$exceptionAllowed = \begin{cases} null & \text{for} & allowedSolutions = null \\ true & \text{for} & NoArchitecturalSolution \in \\ & & allowedSolutions \\ false & \text{for} & NoArchitecturalSolution \notin \\ & & allowedSolutions \end{cases}$$

In deriving these values, the result *null* is used to indicate, that based on the current information no valid statements on the respective property can be made. This offers the possibility to differentiate between the following three statements:

- **Conforms to architectural solution:** The business application conforms to one of the defined architectural solutions.
- **Does not conform to architectural solution:** The business application does not conform to one of the defined architectural solutions.
- **Conformance not documented:** There is no information documented about the architectural solution conformance of the business application.

Especially the distinction between the last two statements is of importance as this awareness results in different next steps that have to be taken. If the conformance is not document this information should be collected. If a business application does not conform to an architectural solution detailed analysis has to be conducted to find reasons for this situation.

[3]These expressions might also be realized in the Object Constraint Language (OCL). For reasons of readability, we chose a mathematical notation instead.

A.3.3.5. Implementation

This I-Pattern should be implemented in a repository, which is able to guarantee consistency for the derived attributes *standardConform* and *exceptionAllowed*.

A.3.3.6. Variants

A possible variant of this I-Pattern would be to simply add an attribute to every business application, indicating if the business application under consideration is conforming to defined architectural standards or not. It is not advised to use this simplified variant, as it restricts the possible analyses. The advantage is that the amount of information, which needs to be collected is limited.

A.3.3.7. Known Uses

The following uses of this I-Pattern are known:

- Enterprise Architecture Management Tool Survey 2008/ SoCaStore (sebis)

An equivalent information model fragment is included in the following EA management tools:

- SoCaTool (sebis)

A.3.3.8. Consequences

A liability of this I-Pattern is the amount of data that has to be collected to be able to reasonable analyze the data. Especially the information of conformance to an architectural solution can only be answered by the business application owners. Therefore, every business application owner has to be interviewed, resulting in a certain investment.
A benefit of this I-Pattern is that an explicit distinction between "there is no information about an architectural solution" and "there is an exception from an architectural solution" is possible.

A.3.3.9. See Also

I-Pattern Architectural Solution Conformance is closely related to defining and documenting architectural solutions. This is addressed by I-Pattern *Architectural Solution* (see page 223) in [Bu08a].
This I-Pattern can be used to manage information for V-Pattern Standards Conformity Exceptions (see page 211).

A.4. EAM Anti Pattern

This section contains the following selection of EAM Anti Patterns.

A.4.1. Oversized Information Model

Oversized Information Model shows, why it is not advisable to develop a giant information model for EA management.

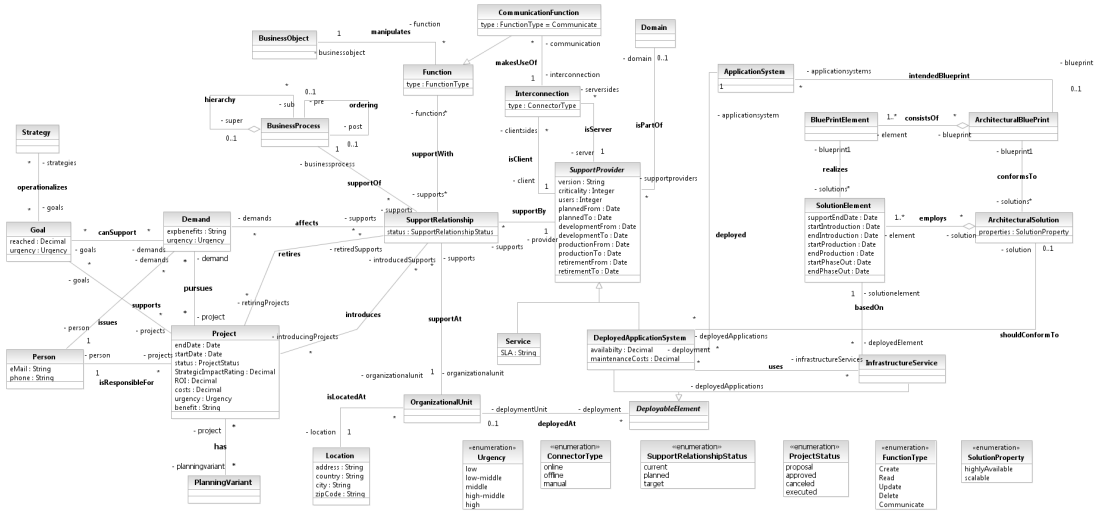

Figure A.11.: Exemplary Oversized Information Model

A.4.1.1. Example

The department store *SoCaStore* wants to start its EA management endeavor. An information model provides the basis for the EA management activities and defines which entities, attributes, and relationships should be documented. Because no commonly accepted standard information model is available, SoCaStore has to develop its own enterprise-specific one.

A.4.1.2. Context

An enterprise, which wants to develop an information model, which fits its specific EA management approach and does not want to rely on predefined ones, as those models do not specifically target the EA management concerns of the enterprise. A predefined information model, as proposed by EA management tools (see [Ma08a] for an overview) or frameworks like TOGAF [Op09a], might contain concepts to address concerns not relevant to the enterprise or might miss concepts needed for specific concerns.

A.4.1.3. Problem

How do you create an information model suitable to fit your enterprise-specific needs, like various kinds of analysis or planning your EA?

The following *forces* influence the solution:

- **Company-wide versus team-wide consultation** The creation of an enterprise-specific information model is a collaborative task, strongly influenced by the participating stakeholder group. Hence, the question on the size of the stakeholder group applies, i.e. the question whether only the EA team should participate in the construction or stakeholders from the remainder of the company.
- **All-embracing versus concern-oriented** Which concerns should be addressed by the information model? Should the information model cover the EA in a holistic manner or should it be developed based on the selected concerns to be addressed?
- **Maximal versus minimal** What is a good size for an information model?

A.4.1.4. General Form

Currently, there is no commonly accepted standard information model for EA management, which satisfies the requirements of all enterprises. As a result most enterprises start to develop their own information model based on their requirements.
In particular, many different stakeholders within the enterprise are asked about their information demands and requirements in respect to EA management. This typically leads to a long list of requirements, resulting in an even larger list of concepts in the respective information model. Due to the length of the list and the high number of stakeholders, which are interested in EA management, this usually results in an Oversized Information Model. An exemplary one is shown in Figure A.11.
The same is true for tools for EA management. Their information models have to fulfill the demands of various companies and interest groups. Table A.1 shows a short overview about the size of information models implemented in typical EA management tools [Ma08a]. How can you maintain an information model including over 200 classes? In order to alter the information model you should at least know, why the classes and attributes have been introduced into the model. It gets even worse, because such models typically include at least twice as many associations and numerous attributes per class.

Vendor	Number of Classes
A	54
B	220
C	470

Table A.1.: Overview about information model size based on [Ma08a]

Oversized Information Model can be observed in various companies. A typical symptom is that a second information model is created, which is way smaller then the initial one.

A.4.1.5. Consequences

Oversized Information Models cannot be maintained in the future, as e.g. the maintaining group neither might be aware of the causes due to which certain concepts were introduced to the model nor might know what the concepts are used for. Another result is that the information stored according to a giant information model cannot be updated in a manner that is required to achieve an adequate information quality. At least if you have to consider the cost benefit ratio for collecting and using the information. The evolution of the information model during the maturation of the EA management endeavor is further hampered by an Oversized Information Model, as the sheer number of concepts and associations in between makes it very difficult to understand the model; thereby, adaptations and changes to the information model are effectively prevented.
When developing an information model, you should therefore execute a prioritization on the collected requirements to concentrate on the most important ones. This initial information model can then be extended in a stepwise manner to keep up with the maturity in EA management of the enterprise. In this case you should document, why the information model has been extended by which concepts or by which I-Patterns [se09a] in order to keep the knowledge on the reasons for the previous extensions for future extensions.

A.4.1.6. Revised Solution

The most important aspect of the revised solution is: Try to avoid Oversized Information Models.
When developing an information model you should first identify the stakeholders, which are relevant for the development. Relevant stakeholders might e.g. be business units as they could serve as future sponsors if the first endeavor is successful or people, who have a current pain, which is addressed in the endeavor.
In a second step try to identify their concerns, which should be addressed by the information model. This can be supported by the EAM pattern approach, which offers I-Patterns documenting information model fragments based on proven-practices [se09a], complemented by a description of the used concepts and relationships. After the required I-Patterns to address your concerns were selected, you can integrate and adapt them, e.g. by introducing additional attributes, to achieve your company-specific information model.
The benefit of this approach is that only those concerns are addressed by the information model, which really have to be addressed, leading to smaller and easier to handle models. This corresponds to the *relevance* criteria, proposed by Becker et al. [BRS95] as part of the *Guidelines of Modeling* (GoM). A similar approach is documented in Detailed Enterprise Model in [Am08].
Another benefit of this approach is that I-Patterns document the problems, which they address including the associated forces. As a result the information model can easier be maintained than an information model, which does not have any documentation why entities, attributes, and associations have been included.

A.4.1.7. See Also

Oversized Information Model should be considered every time when developing a new or changing an existing information model. Ambler [Am08] describes two anti patterns pointing to similar problems. The first one is called Detailed Enterprise Model and described as follows:

> The enterprise model(s) are overly detailed, often in an attempt to comprehensive define what the enterprise does (or should do).

Modeling for Modeling's Sake is the second one.

> Someone thought it would be a good idea to develop an enterprise model but did not have a concrete plan for how to use it in practice. Vague ideas that development teams will be able to use the model for guidance aren't sufficient.

The difference between the anti patterns in [Am08] and Oversized Information Model presented in this paper is basically the extent of its descriptions and that Oversized Information Model subsumes the two anti patterns by Ambler [Am08]. A benefit is that another set of patterns in the context of EA management is referenced and therefore incorporated in the EAM pattern language.

A.4.2. Missing Legend

Missing Legend shows why every visualization should provide a legend.

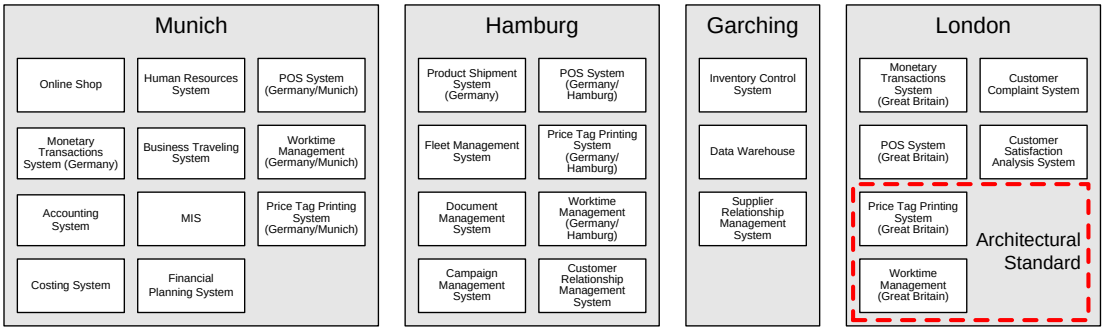

Figure A.12.: Visualization without legend and additional annotation

A.4.2.1. Example

The department store *SoCaStore* wants to get an overview about its application landscape, which has grown unplanned for ten years now. To get this task done the management selects an employee from the IT department to create a visualization of the application landscape, which is understandable by everyone within the company and can be used, even if its creator is not available.

A.4.2.2. Context

Situations where EA visualizations are created or updated to incorporate the new make-up of the enterprise; especially, if the visualizations are used by various people to foster communication about EA management problems.

A.4.2.3. Problem

How do you create a visualization, which is understandable by a wide variety of people, which were not involved the creation process?

The following *forces* influence the solution:

- **Ease of creation versus utility** Creating a visualization with a legend can be considered more complex to do, than simply *drawing* a graphical model. Nevertheless, the omission of a well-defined notation, made explicit in a legend, greatly reduces the visualization's utility, if reuse by people different from the creator is intended.
- **Standardized notation versus free style** Is it required to use a standardized notation to create comprehensible visualizations?
- **Manual versus automatic creation** Is there any influence of the degree of automation on the understandability of created visualizations?

A.4.2.4. General Form

The simplest solution for the employee in the above example is to collect the required information about the application landscape and then to just start to draw using simple symbols, like rectangles, circles, or lines.
This typically results in visualizations like the one shown in Figure A.12. If you have not been part of the team, which created the visualization, you can just speculate about the meaning of the inner and outer boxes. Why are different colors used and why are some boxes placed within other boxes?
In this example it becomes even worse as someone included an annotation, the dotted box, in the visualization. The dotted box includes the text "Architectural Standard". But what does it mean? Do only the two included white boxes satisfy architectural standards? Do all other boxes represent software not corresponding to architectural standards?
Missing Legend can be observed everywhere and is not bound to any company or person.

A.4.2.5. Consequences

A negative consequence of Missing Legend is that it is time consuming and error prone starting to interpret a visualization every time you need it. Additionally, if you come to a wrong interpretation, this may lead you to wrong decisions.
If the semantics is not clear you are unable to communicate the information behind a visualization. Clements et al. [Cl02] present an illustrating example for this situation.

> "These pictures are meant to entertain you. There is no significant meaning to the arrows between the boxes." A speaker at a recent software architecture conference, coming to a complex but ultimately inadequate boxes-and-lines everywhere viewgraph of her system's architecture and deciding that trying to explain it in front of a crowd would not be a good idea.

A.4.2.6. Revised Solution

In cases where a visualization is used by multiple people for discussing or communicating problems it is required to add a legend to the visualization.
Figure A.13 shows an example for a visualization based on Standard Conformity Exceptions [se09a], which includes a complete legend. The legend details about the symbols and colors used within the visualization. Additionally, there is also information available what is meant by the positioning of the used symbols.
For sure it is not possible to answer all questions about a visualization concerning semantics based on the legend, but this is not its goal. It is just meant to clarify the semantics of the visualization.
Even though, the author of the view is not available any more it can still be used as the meaning is clearly described by the legend. For this reason the visualization can be used as a reliable source of information.
Using tools to create visualizations, may help to include an appropriate legend to any visualization, especially if they are generated automatically. In these cases the tool

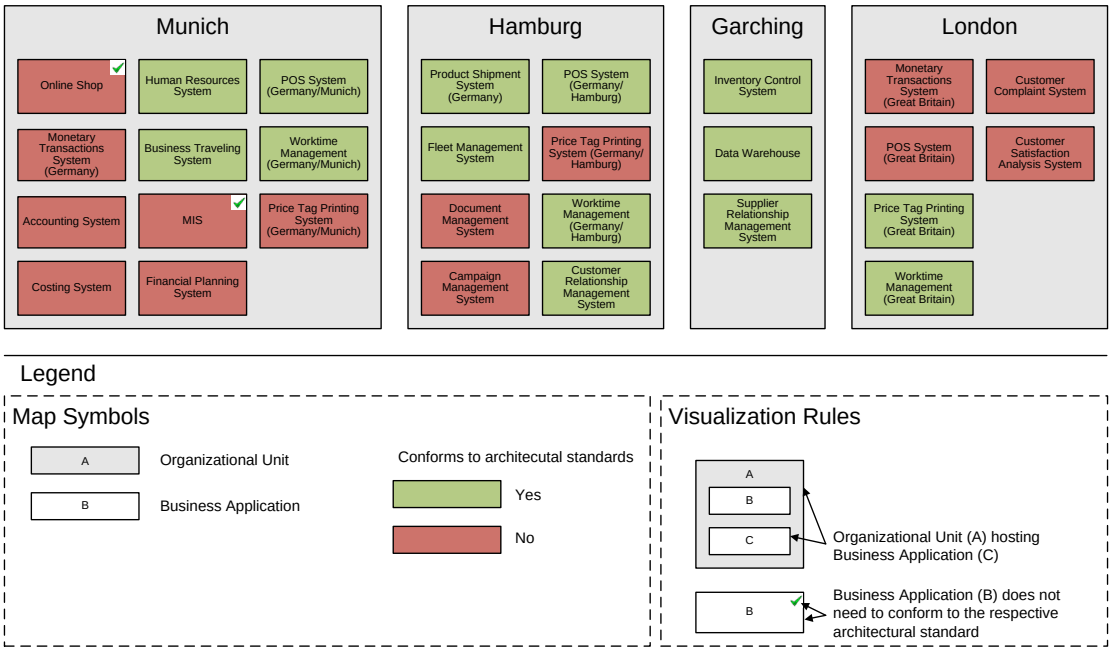

Figure A.13.: Visualization with included legend

already has all required information to generate a legend, because the information is needed for the generation of the visualization itself. Nevertheless, not all EA management tools currently support the creation of a legend [Ma08a].
Another way to resolve the problem of having to include a legend is to use a standardized notation, like e.g. the Unified Modeling Language (UML) [OM05], etc. because the notation should be known or is it least documented.

A.4.2.7. See Also

Missing Legend should be considered whenever a visualization is created, which should be used for communication means between various different people. Especially if the user of the visualization is someone else then the author.

Bibliography

[ABW98] Sherman R. Alpert, Kyle Brown, and Bobby Woolf. *The Design Patterns Smalltalk Companion*. Addison-Wesley Professional, 1998.

[AD06] Antonia Albani and Jan L.G. Dietz. The Benefit of Enterprise Ontology in Identifying Business Components. In *IFIP International Federation for Information Processing: Artificial Intelligence in Theory and Practice IFIP 19th World Computer Congress, TC 12: IFIP AI 2006 Stream, August 21–24, 2006, Santiago, Chile*, pages 243 – 254. Springer Verlag, Boston, 2006.

[Ad09] Jan Stefan Addicks. Enterprise Architecture Dependent Application Evaluations. In *Third IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2009)*, pages 628 – 633, 2009.

[AF09] Stephan Aier and Christian Fischer. Dokumentation und Fortschrittsbestimmung von Methoden zur Gestaltung soziotechnischer Systeme am Beispiel einer Methode zum Service Engineering. In *9. Internationale Tagung Wirtschaftsinformatik (WI 09)*, pages 421 – 429, 2009.

[AG09] Stefanie Alter and Matthias Goeken. Konzeptionelle Metamodelle von IT-Governance-Referenzmodellen als Basis der Kombination und Integration in einer Multi-Modell-Umgebung. In *9. Internationale Tagung Wirtschaftsinformatik (WI 09)*, pages 705 – 714, 2009.

[Ai08a] Stephan Aier, Stephan Kurpjuweit, Christian Riege, and Jan Saat. Stakeholderorientierte Dokumentation und Analyse der Unternehmensarchitektur. In Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, and Christian Scheideler, editors, *GI Jahrestagung (2)*, volume 134 of *LNI*, pages 559 – 565. GI, 2008.

[Ai08b] Stephan Aier, Stephan Kurpjuweit, Otto Schmitz, Jörg Schulz, Andre Thomas, and Robert Winter. An Engineering Approach to Enterprise Architecture Design and its Application at a Financial Service Provider.

In Peter Loos, Markus Nüttgens, Klaus Turowski, and Dirk Werth, editors, *MobIS*, LNI, pages 115 – 130. GI, 2008.

[Ai09] Stephan Aier, Stephan Kurpjuweit, Jan Saat, and Robert Winter. Enterprise Architecture Design as an Engineering Discipline. *AIS Transactions on Enterprise Systems*, 1(1):36 – 43, 2009.

[AIS77] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, USA, 1977.

[Al64] Christopher Alexander. *Notes on the synthesis of form*. Harvard University Press, Cambridge, 1964.

[Al79] Christopher Alexander. *The Timeless Way of Building*. Oxford Univ Press, New York, 1979.

[Am08] Scott W. Ambler. Enterprise Modeling Anti-Patterns. `http://www.agilemodeling.com/essays/enterpriseModelingAntiPatterns.htm` (cited 2009-04-07), 2008.

[Ar07] Farhad Arbab, Frank de Boer, Marcello Bonsangue, Marc Lankhorst, Erik Proper, and Leendert van der Torre. Integrating Architectural Models: Symbolic, Semantic and Subjective Models in Enterprise Architecture. *International Journal of Enterprise Modelling and Information Systems Architectures (EMISA)*, 2(1):44 – 57, 2007.

[ARW08] Stephan Aier, Christian Riege, and Robert Winter. Unternehmensarchitektur - Literaturüberblick und Stand der Praxis. *Wirtschaftsinformatik*, 50(4):292 – 304, 2008.

[Ba08] Daljit Roy Banger. Maturity Assessment for the Enterprise Architecture (EA) Function. `http://www.eaglobals.com/wp-content/uploads/enterprise-architecture-maturity-assessment-10.pdf` (cited 2009-04-28), 2008.

[BC87] Kent Beck and Ward Cunningham. Using Pattern Languages for Object Oriented Programs. Technical report, OOPSLA - Conference on Object-Oriented Programming, Systems, Languages, and Applications, 1987.

[BCK03] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice (SEI Series in Software Engineering)*. Addison-Wesley Longman, Amsterdam, 2nd edition, 2003.

[Be09] Gregor Bender. Designing a Stakeholder-specific EAM based on Patters. Master's thesis, Technische Universität München, Fakultät für Informatik, 2009.

[Be96] Kent Beck, Ron Crocker, Gerard Meszaros, John Vlissides, James O. Coplien, Lutz Dominick, and Frances Paulisch. Industrial experience with

design patterns. In *ICSE '96: Proceedings of the 18th international conference on Software engineering*, pages 103 – 114, Washington, DC, USA, 1996. IEEE Computer Society.

[Be97] Kent Beck. *Smalltalk Best Practice Patterns*. Prentice Hall, 1997.

[BH04] Hans Ulrich Buhl and Bernd Heinrich. Unternehmensarchitekturen in der Praxis – Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Informationssystemen. *Wirtschaftsinformatik*, 46(4):311, 2004.

[BHS07a] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing*. Wiley, 2007.

[BHS07b] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. Wiley, 2007.

[BKL04] Walter R. Bischofberger, Jan Kühl, and Silvio Löffler. Sotograph - A Pragmatic Approach to Source Code Architecture Conformance Checking. In Flávio Oquendo, Brian Warboys, and Ronald Morrison, editors, *EWSA*, volume 3047 of *Lecture Notes in Computer Science*, pages 1 – 9. Springer, 2004.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323 – 364, 1986.

[BM98] Jan O. Borchers and Max Mühlhäuser. Design Patterns for Interactive Musical Systems. *IEEE MultiMedia*, 5(3):36 – 46, 1998.

[Bö08] Martin Böhme. Software cartography: Analysis and graphical visualization of parts of the application landscape of the University of Munich Hospital. Bachelor's Thesis, Technische Universität München, `http://wwwmatthes.in.tum.de/wikis/sebis/BSc-MartinBoehme`, 2008.

[Br05a] Christian Braun, Felix Wortmann, Martin Hafner, and Robert Winter. Method construction - a core approach to organizational engineering. In Hisham Haddad, Lorie M. Liebrock, Andrea Omicini, and Roger L. Wainwright, editors, *SAC*, pages 1295 – 1299. ACM, 2005.

[Br05b] Katharina Brendebach. Integrierte Modelle und Sichten für das IT-Management - Analyse und Entwicklung in Zusammenarbeit mit der HVB Systems. Diploma Thesis, Technische Universität München, 2005.

[Br96] Sjaak Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275 – 280, 1996.

[Br98] William J. Brown, Raphael C. Malveau, Hays W. McCormick, and Thomas J. Mowbray. *AntiPatterns: refactoring software, architectures, and projects in crisis.* John Wiley & Sons, Inc., New York, NY, USA, 1998.

[BRS95] Jörg Becker, Michael Rosemann, and Reinhard Schütte. Grundsätze ordnungsmäßiger Modellierung. *Wirtschaftsinformatik*, 37(5):435 – 445, 1995.

[BRW03] Pavel Balabko, Irina Rychkova, and Alain Wegmann. Operational ASM Semantics behind Graphical SEAM Notation. In *DAIS/FMOODS Ph.D. workshop*, 2003.

[Bu07a] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Kathrin Schneider, and Christian M. Schweda. A pattern based Approach for constructing Enterprise Architecture Management Information Models. In *Wirtschaftsinformatik 2007*, pages 145 – 162, Karlsruhe, Germany, 2007. Universitätsverlag Karlsruhe.

[Bu07b] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Florian Matthes, Christian M. Schweda, and André Wittenburg. Generating Visualizations of Enterprise Architectures using Model Transformations (extended version). *Enterprise Modelling and Information Systems Architectures - An International Journal*, 2(2), 2007.

[Bu07c] Ralf Buchsein, Frank Victor, Holger Günther, and Volker Machmeier. *IT-Management mit ITIL® V3. Strategien, Kennzahlen, Umsetzung.* Vieweg Verlag, 2007.

[Bü07] Thomas Büchner. *Introspektive modellgetriebene Softwareentwicklung.* PhD thesis, Fakultät für Informatik, Technische Universität München, 2007.

[Bu08a] Sabine Buckl, Alexander M. Ernst, Josef Lankes, and Florian Matthes. Enterprise Architecture Management Pattern Catalog (Version 1.0, February 2008). Technical report, Chair for Informatics 19 (sebis), Technische Universität München, Munich, 2008.

[Bu08b] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Florian Matthes, and Christian M. Schweda. Enterprise Architecture Management Patterns - Exemplifying the Approach. In *The 12th IEEE International EDOC Conference (EDOC 2008)*, 2008.

[Bu09a] Sabine Buckl, Alexander Ernst, Harald Kopper, Rolf Marliani, Florian Matthes, Peter Petschownik, and Christian M. Schweda. EA Management Patterns for Consolidations after Mergers. In *SE 2009 - Workshopband*, 2009.

[Bu09b] Sabine Buckl, Florian Matthes, Christian Neubert, and Christian M. Schweda. A Wiki-based Approach to Enterprise Architecture Documentation and Analysis. In *17th European Conference on Information Systems (ECIS)*, 2009.

[Bu09c] Sabine Buckl, Alexander M. Ernst, Florian Matthes, and Christian M. Schweda. Constructing an Information Model for Application Landscape Management. In *21st International Conference on Advanced Information Systems (Caise'09), Amsterdam, The Netherlands*, 2009.

[Bu09d] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Florian Matthes, and Christian M. Schweda. State of the Art in Enterprise Architecture Management - 2009. Technical report, Technische Universität München, Chair for Informatics 19 (sebis), München, Germany, 2009.

[Bu09e] Sabine Buckl, Alexander Ernst, Florian Matthes, and Christian Schweda. Enterprise Architecture Management Pattern for EA Visioning. In *Proceedings of EuroPLoP 2009*, 2009.

[Bu09f] Sabine Buckl, Alexander M. Ernst, Florian Matthes, Rene Ramacher, and Christian M. Schweda. Using Enterprise Architecture Management Patterns to complement TOGAF. In *13th International EDOC conference (EDOC 2009), Auckland, New Zealand*, 2009.

[Bu09g] Sabine Buckl, Alexander M. Ernst, Florian Matthes, and Christian M. Schweda. How to make your enterprise architecture management endeavor fail! In *PLoP 09: Proceedings of the Pattern Languages of Programs Conference 2009*, 2009.

[Bu96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[BS96] Jörg Becker and Reinhard Schütte. *Handelsinformationssysteme*. Verlag Moderne Industrie, Landsberg/Lech, 1996.

[BZ99] Izak Benbasat and Robert W. Zmud. Empirical research in information systems: the practice of relevance. *MIS Quarterly*, 23(1):3 – 16, 1999.

[C497] C4ISR Architecture Working Group. Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance (C4ISR). `http://www.afcea.org/education/courses/archfwk2.pdf` (cited 2009-02-25), 1997.

[Ca07a] Capgemini. Integrated Architecture Framework (IAF). `www.capgemini.com/iaf` (cited 2009-02-25), 2007.

[Ca07b] Alison Cartlidge, Ashley Hanna, Colin Rudd, Ivor Macfarlane, John Windebank, and Stuart Rance. An Introductory Overview of ITIL® V3. `http://www.itsmfi.org/files/itSMF_ITILV3_Intro_Overview_0.pdf` (cited 2009-07-21), 2007.

[Ca07c] Gary Case. *Continual Service Improvement*. The Stationery Office Ltd, 2007.

[Ca09a] Borjan Cace. Patterns for Enterprise-wide SOA. In *SE 2009 - Workshopband*, 2009.

[Ca09b] Capgemini sd&m. Homepage Capgemini sd&m. `http://www.de.capgemini-sdm.com/deutsch/index.html` (cited 2009-07-16), 2009.

[CH04] James O. Coplien and Neil B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[Ch09] Christiana Care Health System. Homepage of Christiana Care Health System. `http://www.christianacare.org/` (cited 2009-05-20), 2009.

[CI09] CIMOSA Association. Computer Integrated Manufacturing Open System Architecture (CIMOSA). `http://www.cimosa.de/` (cited 2009-02-25), 2009.

[Cl02] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. *Documenting Software Architectures: Views and Beyond*. Addison Wesley, Boston, 2002.

[Co02] 107th Congress. Sarbanes-Oxley Act of 2002 (Public Law 107-204). http://www.sec.gov/about/laws/soa2002.pdf, 2002.

[Co92] James O. Coplien. *Advanced C++, Programming Styles and Idioms*. Addison-Wesley, 1992.

[Co96] James O. Coplien. *Software Patterns: Management Briefs*. Cambridge University Press, 1996.

[Co97a] Alistair Cockburn. Project Risk Reduction Patterns. `http://alistair.cockburn.us/Project+risk+reduction+patterns` (cited 2009-04-08), 1997.

[Co97b] James O. Coplien. The Column Without a Name: Pattern Languages. *C++ Report*, 9:15 – 21, 1997.

[Cu09] Ward Cunningham. Portland Pattern Repository. `http://c2.com/ppr/` (cited 2009-04-08), 2009.

[CW07] David Cannon and David Wheeldon. *Service Operation*. The Stationery Office Ltd, May 2007.

[CW97] James O. Coplien and Bobby Woolf. A Pattern Language for Writers' Workshops. `http://www.riehle.org/community-service/hillside-group/europlop-1997/p2final.pdf`, 1997.

[DA98] Paul Dyson and Bruce Anderson. State objects. *Pattern Languages of Program Design*, 3:529 – 574, 1998.

[De06] Gernot Dern. *Management von IT-Architekturen (Edition CIO)*. Vieweg, Wiesbaden, 2006.

[DH08a] Jan L.G. Dietz and Jan A.P. Hoogervorst. Enterprise Engineering - A Manifesto. In *Dietz, Jan L.G.; Albani, Antonia; Barjis, Joseph (Eds.): Advances in Enterprise Engineering I*, pages vii – ix, 2008.

[DH08b] Jan L. G. Dietz and Jan A. P. Hoogervorst. Enterprise ontology in enterprise engineering. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 572 – 579, New York, NY, USA, 2008. ACM.

[Di06] Jan L.G. Dietz. *Enterprise ontology: Theory and methodology.* Springer, 2006.

[Di08] Thomas Dierl. Models, Methods, and Visualizations for Compliance Management. Bachelor's Thesis, Technische Universität München, `http://wwwmatthes.in.tum.de/wikis/sebis/BSc-ThomasDierl`, 2008.

[Do00a] US Department of the Treasury. Treasury Enterprise Architecture Framework (TEAF). `http://www.eaframeworks.com/TEAF/teaf.doc` (cited 2009-02-25), 2000.

[Do00b] US Department of Defense. Technical Architecture Framework for Information Management (TAFIM). `http://en.wikipedia.org/wiki/TAFIM` (cited 2009-02-25), 2000.

[Do05a] US Department of Defense. Joint Technical Architecture Version (JTA). `http://www.acq.osd.mil/osjtf/pdf/jta-vol-I.pdf` (cited 2009-02-25), 2005.

[Do05b] US Department of Defense. DoD Enterprise Architecture Technical Reference Model (DoD EA TRM). `http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf` (cited 2009-02-25), 2005.

[Do07] US Department of Defense. Department of Defense Architecture Framework (DoDAF). `http://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf` (cited 2009-02-25), 2007.

[Do97] US Department of the Treasury. Treasury Information Systems Architecture Framework (TISAF). `http://en.wikipedia.org/wiki/Treasury_Information_System_Architecture_Framework` (cited 2009-02-25), 1997.

[Dy97] Paul Dyson. *Patterns for Abstract Design.* PhD thesis, University of Essex, 1997.

[Ec04] Sven Eckert, Stephan Mantel, Thomas Reeg, Martin Schissler, Otto K. Ferstl, and Elmar J. Sinz. Inter-company integration of application systems - a survey of development methodologies. `http://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/wirtschaftsinformatik_1/dateien/FWN-Bericht-2004-002-Inter-company-Integration.pdf` (cited 2009-08-19), 2004. FORWIN-Bericht Nr. FWN-2004-002.

[Ed04] Editors of The American Heritage Dictionaries. *The American Heritage Dictionary of the English Language*. Houghton Mifflin, 4th edition, 2004.

[En08] Gregor Engels, Andreas Hess, Bernhard Humm, Oliver Juwig, Marc Lohmann, and Jan-Peter Richter. *Quasar Enterprise – Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag, Heidelberg, 2008.

[Er00] Thomas Erickson. Lingua Francas for design: sacred places and pattern languages. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 357 – 368, New York, NY, 2000. ACM Press.

[Er06] Alexander M. Ernst, Josef Lankes, Christian M. Schweda, and André Wittenburg. Using Model Transformation for Generating Visualizations from Repository Contents - An Application to Software Cartography. Technical report, Technische Universität München, Chair for Informatics 19 (sebis), Munich, 2006.

[Er08] Alexander M. Ernst. Enterprise Architecture Management Patterns. In *PLoP 08: Proceedings of the Pattern Languages of Programs Conference 2008*, 2008.

[Er09] Hakan Erdogmus. Cloud Computing: Does Nirvana Hide behind the Nebula? *Software, IEEE*, 26(2):4 – 6, March-April 2009.

[Fa04] Jean-Marie Favre. Foundations of Meta-Pyramids: Languages vs. Metamodels - Episode II: Story of Thotus the Baboon. In Jean Bézivin and Reiko Heckel, editors, *Language Engineering for Model-Driven Software Development*, volume 04101 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2004.

[FC99] US Federal CIO Council. Federal Enterprise Architecture Framework (FEAF). `http://www.cio.gov/documents/fedarch1.pdf` (cited 2009-02-25), 1999.

[Fe08] Christian Flender, Thomas Hettel, Michael Lawley, and Michael Rosemann. Semi-Automated Model Synchronisation in SOM. In Zohra Bellahsène, Carson Woo, Ela Hunt, Xavier Franch, and Remi Coletta, editors, *CAiSE Forum*, volume 344 of *CEUR Workshop Proceedings*, pages 45 – 48. CEUR-WS.org, 2008.

[Fe98] Otto K. Ferstl, Elmar J. Sinz, Christoph Hammel, Michael Schlitt, Stefan Wolf, K. Popp, R. Kehlenbeck, Alexander Pfister, H. Kniep, N. Nielsen, and A. Seitz. WEGA – Wiederverwendbare und erweiterbare Geschäftsprozeß- und Anwendungssystemarchitekturen, 1998. Closing Report.

[FG98] Mark S. Fox and Michael Gruninger. Enterprise Modeling. *AI Magazine*, 19(3):109 – 121, 1998.

[FGE05] José Figueira, Salvatore Greco, and Matthias Ehrogott. Electre Methods. *Multiple Criteria Decision Analysis: State of the Art Surveys*, 78:133 – 153, 2005.

[Fl07] Karl Flieder. Does the Modern World's Design Pattern Concept Have Its Roots in Ancient China? In Marvin J. Dainoff, editor, *Posters*, volume 0000 of *LNCS*, pages 32 – 36. Springer, 2007.

[Fo06] Martin Fowler. Writing Software Patterns. `http://www.martinfowler.com/articles/writingPatterns.html` (cited 2009-04-07), 2006.

[Fo97] Martin Fowler. *Analysis Patterns: Reusable Object Models*. Menlo Park (Calif), Addison Wesley, 1997.

[Fr02] Ulrich Frank. Multi-perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS 3003*, pages 1258 – 1267, 2002.

[Fr08a] Ulrich Frank, David Heise, Heiko Kattenstroth, and Hanno Schauer. Designing and Utilising Business Indicator Systems within Enterprise Models – Outline of a Method. In *Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management 27.-28. November 2008*, Saarbrücken, Germany, 2008.

[Fr08b] Steven Fraser, Ricardo Lopez, Pradeep Kathail, Doug Schmidt, Mary Shaw, Kevin Sullivan, and Dave Thomas. Collaboration and communication: growing and sustaining ultra large scale (ULS) systems. In *OOPSLA Companion '08: Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*, pages 797 – 800, New York, NY, USA, 2008. ACM.

[Fr08c] Ulrich Frank. The MEMO Meta Modelling Language (MML) and Language Architecture (ICB-Research Report). Technical report, Institut für Informatik und Wirtschaftsinformatik, Duisburg-Essen, Germany, 2008.

[Fr08d] Ulrich Frank. Multiperspektivische Unternehmensmodellierung. In *Karl Kurbel, Jörg Becker, Norbert Gronau, Elmar Sinz, Leena Suhl (ed.): Enzyklopädie der Wirtschaftsinformatik : Online-Lexikon*. Oldenbourg, München, 2008.

[Fr09] Ulrik Franke, David Höök, Johan König, Robert Lagerström, Per Närman, Johan Ullberg, Pia Gustafsson, and Mathias Ekstedt. EAF2 - A Framework for Categorizing Enterprise Architecture Frameworks. In *Proc. 10th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 327 – 332, May 2009.

[Fü05] János Fülöp. Introduction to Decision Making Methods. `http://academic.evergreen.edu/projects/bdei/documents/decisionmakingmethods.pdf` (cited 2009-05-02), 2005.

[FS01] Otto K. Ferstl and Elmar J. Sinz. *Grundlagen der Wirtschaftsinformatik Band 1.* Grundlagen der Wirtschaftsinformatik / von Otto K. Ferstl und Elmar J. Sinz. Oldenbourg, München, Wien, 4th edition, 2001.

[FS90] Otto K. Ferstl and Elmar J. Sinz. Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). *WIRTSCHAFTSINFORMATIK*, 32:566 – 581, 1990.

[FS95] Otto K. Ferstl and Elmar J. Sinz. Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. *WIRTSCHAFTSINFORMATIK*, 37:209 – 220, 1995.

[FS98] Otto K. Ferstl and Elmar J. Sinz. SOM Modeling of Business Systems. In *P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbooks on Information Systems*, pages 339 – 358. Springer, 1998.

[GA03] United States General Accounting Office (GAO). A Framework for Assessing and Improving Enterprise Architecture Management. `http://www.gao.gov/new.items/d03584g.pdf` (cited 2009-04-28), 2003.

[Ga04] Wolfgang Gaertner. Ansatz für eine erfolgreiche Enterprise Architecture im Bereich Global Banking Division/Global Transaction Banking IT and Operations der Deutschen Bank. *Wirtschaftsinformatik*, 46(4):311 – 313, 2004.

[Ga91] Erich Gamma. *Objektorientierte Software-Entwicklung am Beispiel von ET++: Klassenbibliothek, Werkzeuge, Design.* PhD thesis, University of Zürich, 1991.

[Ga95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[GF96] Michael Gruninger and Mark S. Fox. The logic of enterprise modelling, 1996.

[GMW97] David Garlan, Robert T. Monroe, and David Wile. ACME: An Architecture Description Interchange Language. In *CASCON'97*, Toronto, Ontario, Canada, 1997. IBM Press.

[GSL07] Magnus Gammelgård, Mårten Simonsson, and Åsa Lindström. An IT management assessment framework: evaluating enterprise architecture scenarios. *Information Systems and e-Business Management*, 5(4):415 – 435, 2007.

[Ha03] Robert Hanmer. Pattern Languages. http://www.cin.ufpe.br/~sugarloafplop/patLang.ppt (cited 2009-04-16), 2003.

[Ha05] Ken Harmon. The "systems" nature of enterprise architecture. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 78 – 85, October 2005.

[HA07] Neil B. Harrison and Paris Avgeriou. Leveraging Architecture Patterns to Satisfy Quality Attributes. In *ECSA*, pages 263 – 270, 2007.

[He04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75 – 105, 2004.

[He93] Michael Heym. *Methoden-Engineering*. Rosch-Buchbinderei, Hallstadt, 1993.

[Hi05] Martin Hitz, Gerti Kappel, Elisabeth Kapsammer, and Werner Retschitzegger. *UML@Work*. dpunkt.verlag, 3rd edition, 2005. (in German).

[Hi08] The Hillside Group. 15th Conference on Pattern Languages of Programms (PLoP) 2008. http://hillside.net/plop/2008/ (cited 2009-07-15), 2008.

[Hi09] The Hillside Group. Hillside.net. http://hillside.net/ (cited 2009-07-03), 2009.

[HRL09] Oliver Holschke, Jannis Rake, and Olga Levina. Granularity as a Cognitive Factor in the Effectiveness of Business Process Model Reuse. In *Business Process Management Conference 2009*, 2009.

[HW08] Martin Hafner and Robert Winter. Processes for Enterprise Application Architecture Management. In *HICSS*, page 396. IEEE Computer Society, 2008.

[IBM09] IBM. Telelogic System Architect. http://www-01.ibm.com/software/awdtools/systemarchitect/ (cited 2009-05-20), 2009.

[ID06] IDS Scheer AG. *ARIS Platform Method ARIS 7.0*. IDS Scheer AG, Saarbrücken, 2006.

[ID09] IDS Scheer. Architektur integrierter Informationssysteme (ARIS). http://www.ids-scheer.com/de/ARIS/ (cited 2009-02-25), 2009.

[Ie00] IEEE Computer Society. IEEE Std 1471-2000 for Recommended Practice for Architectural Description of Software-Intensive Systems, 2000.

[IF04] Institute for Enterprise Architecture Developments (IFEAD). Extended Enterprise Architecture Maturity Model (E2AMM) v2.0. http://www.enterprise-architecture.info/Images/E2AF/E2AMMv2.PDF (cited 2009-04-28), 2004.

[IF06] Institute for Enterprise Architecture Developments (IFEAD). Extended Enterprise Architecture Framework (E2AF). `http://www.enterprise-architecture.info/` (cited 2009-02-25), 2006.

[IF99] IFIP-IFAC Task Force. Generalised Enterprise Reference Architecture and Methodology (GERAM). `http://www.cimosa.de/` (cited 2009-02-25), 1999.

[IN07] Majid Iqbal and Michael Nieves. *Service Strategy*. The Stationery Office Ltd, 2007.

[IS07] International Organization for Standardization. ISO/IEC 42010:2007 Systems and software engineering – Recommended practice for architectural description of software-intensive systems, 2007.

[IS91] International Standards Organisation (ISO). International Standard (ISO/IEC) 9126. Information technology: Software product evaluation: Quality characteristics and guidelines for their use, 1991.

[IT07] IT Governance Institute. *COBIT 4.1*. IT Governance Institute, 2007.

[Ja08] Greta James. Magic Quadrant for Enterprise Architecture Tools, 4Q04, 2008.

[Ja95] Michael A. Jackson. *Software requirements & specifications: a lexicon of practice, principles and prejudices*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[Je01] Finn V. Jensen. *Bayesian Networks and Decision Graphs (Information Science and Statistics)*. Springer, July 2001.

[Jo05] Henk Jonkers, Luuk Groenewegen, Marcello Bonsangue, and René van Buuren. A language for enterprise modelling. In Marc Lankhorst, editor, *Enterprise Architecture at Work*. Springer, Berlin, Heidelberg, New York, 2005.

[Jo09] Andrew Josey et al. *TOGAF Version 9 – A Pocket Guide*. Van Haren Publishing, Wilco, Amersfoort, Netherlands, 2nd edition, 2009.

[Ju04] Elke Jung. Ein unternehmensweites IT-Architekturmodell als erfolgreiches Bindeglied zwischen der Unternehmensstrategie und dem operativen Bankgeschäft. *Wirtschaftsinformatik*, 46(4):313 – 314, 2004.

[KBS05] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA*. Prentice Hall, 2005. Primary.

[Ke07] Wolfgang Keller. *IT-Unternehmensarchitektur*. dpunkt.verlag, Heidelberg, 2007.

[Ki08] Lutz Kirchner. *Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung*. PhD thesis, Universität Duisburg-Essen, Berlin, 2008.

[Kn09] Gerhard Knolmayer. Information Engineering. `http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/Informationsmanagement--Konzepte-des/Information-Engineering` (cited 2009-02-25), 2009.

[Kr05] Helmut Krcmar. *Informationsmanagement*. Springer, Berlin, 4th edition, 2005.

[Kr93] Klaus Kronlöf. *Method Integration: Concepts and Case Studies*. John Wiley & Sons Ltd., West Sussex, England, 1993.

[Kü04] Harald Kühn. *Methodenintegration im Business Engineering*. PhD thesis, University of Vienna, 2004.

[Kü06] Thomas Kühne. Matters of (Meta-) Modeling. *Software and Systems Modeling (SoSyM)*, 5(4):369 – 385, December 2006.

[KS08] Christian Kohls and Katharina Scheiter. The psychology of patterns. In *Pattern Languages of Programs (PLoP) 2008*, 2008.

[KSH93] Klaus Kronlöf, Anne Sheehan, and Matthias Hallmann. The Concept of Method Integration. In Klaus Kronlöf, editor, *Method Integration: Concepts and Case Studies*, pages 1 – 18. John Wiley & Sons Ltd., West Sussex, England, 1993.

[KW07] Stephan Kurpjuweit and Robert Winter. Viewpoint-based Meta Model Engineering. In Manfred Reichert, Stefan Strecker, and Klaus Turowski, editors, *Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA)- Concepts and Applications*, LNI, pages 143 – 161. GI, 2007.

[KW09] Stephan Kurpjuweit and Robert Winter. Concern-oriented business architecture engineering. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 265 – 272, New York, NY, USA, 2009. ACM.

[KW92] Kuldeep Kumar and Richard J. Welke. Methodology Engineering: A proposal for situation-specific methodology construction. In *Challenges and strategies for research in systems development*, pages 257 – 269, New York, NY, USA, 1992. John Wiley & Sons, Inc.

[La05a] Marc Lankhorst. Introduction to Enterprise Architecture. In *Enterprise Architecture at Work*, Berlin, Heidelberg, New York, 2005. Springer.

[La05b] Steffen Lauschke. Softwarekartographie: Analyse und Darstellung der IT-Landschaft eines mittelständischen Unternehmens. Bachelor's Thesis, Technische Universität München, 2005.

[La08a] Josef Lankes. *Metrics for Application Landscapes*. Dissertation, Technische Universität München, München, 2008.

[La08b] Robert Lagerström, Moustafa Chenine, Pontus Johnson, and Ulrik Franke. Probabilistic Metamodel Merging. In Zohra Bellahsène, Carson Woo, Ela Hunt, Xavier Franch, and Remi Coletta, editors, *CAiSE Forum*, pages 25 – 28, 2008.

[La09a] Armin Lau, Thomas Fischer, Sabine Buckl, Alexander M. Ernst, Florian Matthes, and Christian M. Schweda. EA Management Patterns for Smart Networks. In *SE 2009 - Workshopband*, 2009.

[La09b] Robert Lagerström, Jan Saat, Ulrik Franke, Stephan Aier, and Mathias Ekstedt. Enterprise Meta Modeling Methods? Combining a Stakeholder-Oriented and a Causality-Based Approach. In *Proceedings of the International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD)*, June 2009.

[Le07] Jana Leitel. Entwicklung und Anwendung von Beurteilungskriterien für Enterprise Architecture Frameworks. Master's thesis, Technische Universität München, Fakultät für Informatik, 2007.

[LJH09] Robert Lagerström, Pontus Johnson, and David Höök. An Enterprise Architecture Management Pattern for Software Change Project Cost Analysis. In *SE 2009 - Workshopband*, 2009.

[LL09] Marc M. Lankhorst and Paul H.W.M. Oude Luttighui. Enterprise Architecture Patterns for Multichannel Management. In *SE 2009 - Workshopband*, 2009.

[LM07] Shirley Lacy and Ivor Macfarlane. *Service Transition.* The Stationery Office Ltd, 2007.

[LMW05] Josef Lankes, Florian Matthes, and André Wittenburg. Softwarekartographie: Systematische Darstellungen von Anwendungslandschaften. In *Wirtschaftsinformatik 2005*, pages 1443 – 1462, Bamberg, Germany, 2005. Physica-Verlag.

[Lo97] David H. Lorenz. Tiling Design Patterns - A Case Study Using the Interpreter Pattern. In *OOPSLA*, pages 206 – 217, 1997.

[LS08] Josef Lankes and Christian M. Schweda. Using Metrics to Evaluate Failure Propagation and Failure Impacts in Application Landscapes. In *Multikonferenz Wirtschaftsinformatik.* GITO-Verlag, Berlin, 2008.

[LTW08] Kathleen A. Larson, Frances P. Trees, and D. Scott Weaver. Continuous Feedback Pedagogical Patterns. In *PLoP 08: Proceedings of the Pattern Languages of Programs Conference 2008*, 2008.

[LW04a] Kerstin Langenberg and Alain Wegmann. Enterprise Architecture: What Aspect is Current Research Targeting? Technical report, Laboratory of Systemic Modeling, Ecole Polytechnique Fédérale de Lausanne, Lausanne, 2004.

[LW04b] Lam-Son Lê and Alain Wegmann. Meta-model for Object-Oriented Hierarchical Systems. Technical report, Swiss Federal Institute of Technology, Lausanne, 2004.

[LW06] Lam-Son Lê and Alain Wegmann. SeamCAD: Object-Oriented Modeling Tool for Hierarchical Systems in Enterprise Architecture. In *System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 8, Jan. 2006.

[LZ06] Susanne Leist and Gregor Zellner. Evaluation of current architecture frameworks. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1546 – 1553, New York, NY, USA, 2006. ACM.

[Ma05] Matteo Magnani, Nikos Rizopoulos, Peter McBrien, and Danilo Montesi. Schema Integration Based on Uncertain Semantic Mappings. In Lois M. L. Delcambre, Christian Kop, Heinrich C. Mayr, John Mylopoulos, and Oscar Pastor, editors, *ER*, Lecture Notes in Computer Science, pages 31 – 46. Springer, 2005.

[Ma08a] Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M. Schweda. *Enterprise Architecture Management Tool Survey 2008.* Chair for Informatics 19 (sebis), Technische Universität München, Munich, 2008.

[Ma08b] Florian Matthes. Softwarekartographie. *Informatik Spektrum*, 31(6), 2008.

[Ma09] Michael Maicher, Gregor Sauerzapf, Christian Winterhalder, and Felix Palme. Report on Progress of Enterprise-Architecture-Management 2008. `http://www.eam-thinktank.de/index.php?option=com_content&task=view&id=28&Itemid=2` (cited 2009-02-25), 2009.

[Ma91] Allan MacLean, Richard Young, Victoria Bellotti, and Tom Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction*, 6(3):201 – 250, 1991.

[Ma95] Sandra P. Marshall. *Schemas in Problem Solving.* Cambridge University Press, Cambridge, 1995.

[Mc98] Hays W. McCormick. Anti Patterns - Refactoring Software, Architectures, and Projects in Crisis. `http://www.antipatterns.com/briefing/index.htm` (cited 2009-05-08), 1998.

[MD01] Jonathan C. McPhail and Dwight Deugo. Deciding on a Pattern. In *Lecture Notes in Computer Science*, 2001.

[MD98] Gerard Meszaros and Jim Doble. A Pattern Language for Pattern Writing. *Pattern Languages of Program Design*, 3:529 – 574, 1998.

[ME02] META Group. Enterprise Architecture Desk Reference, 2002.

[Me04] Peter Mertens. Diskussionsrunde zum Thema "Unternehmensarchitekturen in der Praxis". *Wirtschaftsinformatik*, 46(4):315, 2004.

[MM03] Joaquin Miller and Jishnu Mukerji. MDA Guide Version 1.0.1. `http://www.omg.org/docs/omg/03-06-01.pdf` (cited 2009-07-16), 2003.

[MM95] Allan Maclean and Diane McKerlie. Design space analysis and use representations. In *Scenario-based design: envisioning work and technology in system development*, pages 183 – 207, New York, NY, USA, 1995. John Wiley & Sons, Inc.

[Mo07] Stephen B. Morris. Enterprise architecture essentials, Part 6: Manageability. `http://www.modaf.org.uk/` (cited 2009-02-25), 2007.

[Mo08] UK Ministry of Defence. Ministry of Defence Architectural Framework (MODAF). `http://www.modaf.org.uk/` (cited 2009-02-25), 2008.

[Mo09a] Christoph Moser, Stefan Junginger, Matthias Brückmann, and Klaus-Manfred Schöne. Some Process Patterns for Enterprise Architecture Management. In *SE 2009 - Workshopband*, 2009.

[Mo09b] JP Morgenthal. Architecture Frameworks Don't Make Architects. `http://www.jpmorgenthal.com/morgenthal/?p=33` (cited 2009-07-15), 2009.

[MS95] Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251 – 266, 1995.

[MSE06] Abbas Sadeghzadeh Milani, Ali Shanian, and Christine El-Lahham. Using different ELECTRE methods in strategic planning in the presence of human behavioral resistance. *Journal of Applied Mathematics and Decision Sciences*, 2006:1 – 19, 2006.

[My07] Mark G. Mykityshyn. *Assessing the Maturity of Information Architectures for Complex Dynamic Enterprise Systems.* PhD thesis, Georgia Institute of Technology, 2007.

[Na03] National Association of State Chief Information Officers (NASCIO). NASCIO Enterprise Architecture Maturity Model Version 1.3. `http://www.nascio.org/publications/documents/NASCIO-EAMM.pdf` (cited 2009-04-28), 2003.

[NA07] NATO C3 Board. NATO Architecture Framework (NAF). `http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf` (cited 2009-02-25), 2007.

[Ni06] Klaus D. Niemann. *From Enterprise Architecture to IT Governance.* Vieweg, Wiesbaden, 2006.

[NJN07] Per Närman, Pontus Johnson, and Lars Nordström. Enterprise Architecture: A Framework Supporting System Quality Analysis. In *EDOC*, pages 130 – 141. IEEE Computer Society, 2007.

[No06] Linda Northrop, Peter Feiler, Richard P. Gabriel, John Goodenough, Rick Linger, Tom Longstaff, Rick Kazman, Mark Klein, Douglas Schmidt, Kevin Sullivan, and Kurt Wallnau. Ultra-Large-Scale Systems - The Software Challenge of the Future. Technical report, Software Engineering Institute, Carnegie Mellon, June 2006.

[No98] James Noble. Classifying Relationships between Object-Oriented Design Patterns. *Australian Software Engineering Conference*, 0:98, 1998.

[Of09] Philipp Offermann, Olga Levina, Marten Schönherr, and Udo Bub. Outline of a design science research process. In *DESRIST '09: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, pages 1 – 11, New York, NY, USA, 2009. ACM.

[OM05] OMG. Unified Modeling Language: Superstructure, version 2.0, formal/05-07-04, 2005.

[OM06] OMG. Meta Object Facility (MOF) Core Specification, version 2.0 (formal/06-01-01), 2006.

[Op04] The Open Group. Business Executive's Guide to IT Architecture. `http://www.opengroup.org/bookstore/catalog/w043.htm` (cited 2009-06-05), 2004.

[Op09a] The Open Group. TOGAF "Enterprise Edition" Version 9, 2009.

[Op09b] The Open Group. *ArchiMate® 1.0 Specification.* Van Haren Publishing, 2009.

[Op09c] Martin Op 't Land, Hans Zwitzer, Paul Ensink, and Quentin Lebel. Towards a fast enterprise ontology based method for post merger integration. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 245 – 252, New York, NY, USA, 2009. ACM.

[Ös07] Hubert Österle, Robert Winter, Frank Hoening, Stephan Kurpjuweit, and Philipp Osl. Der St. Galler Ansatz des Business Engineering: Das Core Business Metamodel. *Wisu - Das Wirtschaftsstudium*, 2(36):191 – 194, 2007.

[Pa72] David Lorge Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12):1053 – 1058, December 1972.

[Pa97] Richard F. Paige. A Meta-Method for Formal Method Integration. In John S. Fitzgerald, Cliff B. Jones, and Peter Lucas, editors, *FME*, volume 1313 of *Lecture Notes in Computer Science*, pages 473 – 494. Springer, 1997.

[PC01] Purdue Consortium. Purdue Enterprise Reference Architecture (PERA). `http://www.cimosa.de/` (cited 2009-02-25), 2001.

[PCV02] Mari Carmen Puerta-Melguizo, Christina Chisalita, and Gerrit C. Van der Veer. Assessing users mental models in designing complex systems. In *IEEE International Conference on Systems, Man and Cybernetics, 2002*, 2002.

[Pe04] Henry Peyret. Getting Value From Enterprise Architecture Tools, 2004.

[Pf08] Katharina Pflügler. Evaluation and Extension of the EAM Pattern Catalog in a German Insurance Company. Bachelor's Thesis, Technische Universität München, `http://wwwmatthes.in.tum.de/wikis/sebis/BSc-KatharinaPfluegler`, 2008.

[Pi09] Alan Pilkington. Enterprise Engineering. `http://personal.rhul.ac.uk/uhtm/001/homepage.html` (cited 2009-07-21), 2009.

[Pl99] Plato. Meno. `http://www.gutenberg.org/ebooks/1643` (cited 2009-01-22), 1999.

[Po98] Stephan T. Polyak. Applying Design Space Analysis To Planning. `http://www.aaai.org/Papers/Workshops/1998/WS-98-03/WS98-03-005.pdf` (cited 2009-05-01), 1998.

[RA09a] Christian Riege and Stephan Aier. A Contingency Approach to Enterprise Architecture Method Engineering (extended version). *Journal Of Enterprise Architecture*, Vol. 5(1), 2009.

[Ra09b] Rene Ramacher. Entwurf und Realisierung einer Viewpoint Description Language (VDL) für die Systemkartographie. Diploma Thesis, Technische Universität München, `http://wwwmatthes.in.tum.de/wikis/sebis/DA-ReneRamacher`, 2009.

[Ri00] Linda Rising. *The Pattern Almanac 2000.* Addison-Wesley, Reading, MA, 2000.

[Ri97] Dirk Riehle. Bureaucracy. In *Pattern languages of program design 3*, pages 163 – 185, Boston, MA, USA, 1997. Addison-Wesley Longman Publishing Co., Inc.

[Ri98] Linda Rising. *The Patterns Handbook: Techniques, Strategies, and Applications.* Trafford Publishing, 1998.

[RL07] Colin Rudd and Vernon Lloyd. *Service Design.* The Stationery Office Ltd, 2007.

[Ro68] Bernard Roy. Classement et choix en présence de points de vue multiples: La méthode ELECTRE. *Revue Francaise d'Informatique et de Recherche Opérationnelle*, 8:57 – 75, 1968.

[Ro91] Bernard Roy. The Outranking Approach and the Foundation of ELECTRE Methods. *Theory and Decision*, 31(1):49 – 73, 1991.

[RWR06] Jeanne W. Ross, Peter Weill, and David C. Robertson. *Enterprise Architecture as Strategy.* Harvard Business School Press, Boston, Massachusetts, 2006.

[Ry07] Irina Rychkova, Gil Regev, Lam-Son Lê, and Alain Wegmann. From Business to IT with SEAM: The J2EE Pet Store Example. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 495 – 495, October 2007.

[Sa00] Nikos A. Salingaros. The Structure of Pattern Languages. *Architectural Research Quarterly*, 4:149 – 161, 2000.

[Sa98] Motoshi Saeki. A meta-model for method integration. *Information & Software Technology*, 39(14-15):925 – 932, 1998.

[SA99] Orsolya Szegheo and Bjørn Andersen. Modeling the Extended Enterprise: A Comparison of Different Modeling Approaches. In *International Conference on Enterprise Modeling*, 1999.

[Sc00a] Douglas Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects.* Wiley & Sons, New York, NY, USA, 2000.

[Sc00b] August-Wilhelm Scheer. *ARIS - Business Process Modeling.* Springer Verlag, 3rd edition, 2000.

[Sc01] August-Wilhelm Scheer. *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen.* Springer, Berlin, 4th edition, 2001.

[Sc04] Jaap Schekkerman. Another View at Extended Enterprise Architecture Viewpoints. `http://www.enterprise-architecture.info/Images/Extended%20Enterprise/E2A-Viewpoints_IFEAD.PDF` (cited 2009-02-25), 2004.

[Sc06a] Jaap Schekkerman. *How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework.* Trafford Publishing, 2006.

[Sc06b] Christian M. Schweda. Architektur eines Visualisierungswerkzeuges für Anwendungslandschaften - Anforderungsanalyse und prototypische Realisierung. Diploma Thesis, Technische Universität München, `http://wwwmatthes.in.tum.de/wikis/sebis/DA-ChristianMSchweda`, 2006.

[Sc08] Jaap Schekkerman. *Enterprise Architecture Good Practices Guide: How to Manage the Enterprise Architecture Practice.* Trafford Publishing, June 2008.

[Sc09] Marten Schöenherr. Towards a Common Terminology in the Discipline of Enterprise Architecture. In *Service-Oriented Computing – ICSOC 2008 Workshops*, pages 400 – 413, 2009.

[se05] Technische Universität München Chair for Informatics 19 (sebis). Enterprise Architecture Management Tool Survey 2005. Technical report, Chair for Informatics 19 (sebis), Technische Universität München, Munich, 2005.

[SE06] CMMI Product Team. CMMI for Development, Version 1.2. Technical report, Carnegie Mellon University, Software Engineering Institute, 2006. CMU/SEI-2006-TR-008.

[se09a] Technische Universität München Chair for Informatics 19 (sebis). EAM Pattern Catalog Wiki. `http://eampc-wiki.systemcartography.info` (cited 2009-05-01), 2009.

[SE09b] CMMI Product Team. CMMI for Services, Version 1.2. Technical report, Carnegie Mellon University, Software Engineering Institute, 2009. CMU/SEI-2009-TR-001.

[se09c] Technische Universität München Chair for Informatics 19 (sebis). Software Engineering for Business Applications - Master Course. `http://wwwmatthes.in.tum.de/wikis/sebis/vorlesung-seba-master` (cited 2009-06-18), 2009.

[se09d] Technische Universität München Chair for Informatics 19 (sebis). Research Project System Cartography. `http://www.systemcartography.info` (cited 2009-07-05), 2009.

[SG89] Susan Leigh Star and James R. Griesemer. Institutional Ecology: "Translations" and Coherence: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology. *Social Studies of Science*, 19(3):387 – 420, 1989.

[Sh90] Mary Shaw. Prospects for an Engineering Discipline of Software. *IEEE Software*, 7(6):15 – 24, 1990.

[SH93] Steven H. Spewak and Steven C. Hill. *Enterprise architecture planning: developing a blueprint for data, applications and technology.* QED Information Sciences, Inc., Wellesley, MA, USA, 1993.

[Si04a] Elmar J. Sinz. Unternehmensarchitekturen in der Praxis - Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Informationssystemen. *WIRTSCHAFTSINFORMATIK*, 46(4):315 – 316, 2004.

[Si04b] Johannes Siedersleben. *Moderne Software-Architektur.* dpunkt.verlag, August 2004.

[Sl03] Terry A. Slocum, Robert B. McMaster, Fritz C. Kessler, and Hugh H. Howard. *Thematic Cartography and Geographic Visualization.* Prentice Hall, 2nd edition, 2003.

[SP94] Stefano Spaccapietra and Christine Parent. View Integration: A Step Forward in Solving Structural Conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258 – 274, 1994.

[SPD92] Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *VLDB Journal*, 1(1):81 – 126, 1992.

[SR98] Peter Sommerlad and Marcel Rüedi. Do-it-yourself reflection. In *EuroPLOP Proceedings*, 1998.

[St73] Herbert Stachowiak. *Allgemeine Modelltheorie.* Springer-Verlag, Wien, 1973.

[St98] Susanne Strahringer. Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips. In *Modellierung*, 1998.

[St99] Jörg Strübing. Soziale Welten - Arenen - Grenzobjekte. In *29. Kongress der Deutschen Gesellschaft für Soziologie (Sektion Wissenschafts- und Technikforschung)*, Freiburg, 1999.

[SZ92] John F. Sowa and John. A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590 – 616, 1992.

[Ta07] Toufik Taibi. *Design Patterns Formalization Techniques.* IGI Global, 2007.

[Te99] Rolf A. Teubner. *Organisations- und Informationssystemgestaltung.* Deutscher Universitäts-Verlag, Wiesbaden, 1999.

[TM09] TeleManagement Forum. Business Process Framework (eTOM). `http://www.tmforum.org/BusinessProcessFramework/1647/home.html` (cited 2009-07-21), 2009.

[To98] Juha-Pekka Tolvanen. *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence.* PhD thesis, University of Jyväskylä, 1998.

[TRL96] Juha-Pekka Tolvanen, Matti Rossi, and Hui Liu. Method engineering: current research directions and implications for future research. In *Proceedings of the IFIP TC8, WG8.1/8.2 working conference on method engineering on Method engineering : principles of method construction and tool support*, pages 296 – 317, London, UK, UK, 1996. Chapman & Hall, Ltd.

[Tu01] Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut, 2nd edition, 2001.

[Uh04] Jürgen Uhl. "Unternehmensarchitekturen" ist ein Dauerthema - aber die Ziele bzw. Motivation und damit Schwerpunkte ändern sich, vor allem mit wirtschaftlichen Randbedingungen. *Wirtschaftsinformatik*, 46(4):317, 2004.

[Vl97] John Vlissides. Patterns: The Top Ten Misconceptions. *Object Magazine*, 7(3), 1997.

[Vö04] Markus Völter. Hope, Believe and Wizardry - Three different perspectives on project management. In *Proceedings on EuroPLoP 2002*, 2004.

[We03] Alain Wegmann. The Systemic Enterprise Architecture Methodology (SEAM) - Business and IT Alignment for Competitveness. In *SEAM). Published at the International Conference on Enterprise Information Systems 2003 (ICEIS 2003*, pages 483 – 490, 2003.

[We05] Alain Wegmann, Pavel Balabko, Lam-Son Lê, Gil Regev, and Irina Rychkova. A Method and Tool for Business-IT Alignment in Enterprise Architecture. In *CAiSE Short Paper Proceedings 2005*, 2005.

[We06] Alain Wegmann, Lam-Son Lê, Lotfi Hussami, and Dirk Beyer. A Tool for Verified Design using Alloy for Specification and CrocoPat for Verification. In D. Jackson and P. Zave, editors, *Proceedings of the First Alloy Workshop (ALLOY 2006), Portland, Oregon*, 2006.

[We07a] Alain Wegmann, Gil Regev, Irina Rychkova, Lam-Son Lê, and Philippe Julia. Business and IT Alignment with SEAM for Enterprise Architecture. In *EDOC*, pages 111 – 121. IEEE Computer Society, 2007.

[We07b] Alain Wegmann, Gil Regev, José Diego de la Cruz, Lam-Son Lê, and Irina Rychkova. Teaching Enterprise Architecture and Servie-Oriented Architecture in Practice. In *Proceedings of the Second Workshop on Trends in Enterprise Architecture Research (TEAR 2007)*, pages 13 – 22, 2007.

[We07c] Alain Wegmann, Philippe Julia, Gil Regev, Olivier Perroud, and Irina Rychkova. Early Requirements and Business-IT Alignment with SEAM for Business. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*, pages 111 – 114, October 2007.

[We07d] Alain Wegmann, Lam-Son Lê, Gil Regev, and Bryan Wood. Enterprise modeling using the foundation concepts of the RM-ODP ISO/ITU standard. *Inf. Syst. E-Business Management*, 5(4):397 – 413, 2007.

[We75] Gerald M. Weinberg. *An Introduction to General System Thinking*. Wiley & Sons. New York, 1975.

[WHO09] World Health Organization. International Classification of Diseases (ICD). `http://www.who.int/classifications/icd/en/` (cited 2009-05-20), 2009.

[Wi03] Robert Winter. *Modelle, Techniken und Werkzeuge im Business Engineering*, pages 87 – 118. Springer Berlin, 2nd edition, 2003.

[Wi04] Robert Winter. Architektur braucht Management. *Wirtschaftsinformatik*, 46(4):317 – 319, 2004.

[Wi07a] André Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. PhD thesis, Fakultät für Informatik, Technische Universität München, 2007.

[Wi07b] Robert Winter, Tobias Bucher, Ronny Fischer, and Stephan Kurpjuweit. Analysis and Application Scenarios of Enterprise Architecture – An Exploratory Study. *Journal of Enterprise Architecture*, 3(3):33 – 43, 2007.

[Wi08] Robert Winter. Virtual Roundtable zu Enterprise Architecture Management (EAM): Ziele und Einsatzperspektiven für Enterprise Architektur-Management in IT-Organisationen. `http://www.competence-site.de/downloads/49/a9/i_file_2585/VR_EAM_IT_Unternehmen_Prof_Winter_080701.pdf` (cited 2008-08-13), 2008.

[Wö04] Friedrich Wöbking. Unternehmensarchitekturen in der Praxis - Architekturdesign am Reißbrett vs. situationsbedingte Realisierung von Informationssystemen. *Wirtschaftsinformatik*, 46(4):319 – 320, 2004.

[WV03] Martijn van Welie and Gerrit C. van der Veer. Pattern Languages in Interaction Design: Structure and Organization. In *Human-Computer Interaction – INTERACT'03*, pages 527 – 534, 2003.

[Za08] Zachman Institute for Framework Advancement. Enterprise Architecture: A Framework. `http://www.zifa.com/framework.pdf` (cited 2008-03-19), 2008.

[Za87] John A. Zachman. A framework for information systems architecture. *IBM Syst. J.*, 26(3):276 – 292, 1987.

[Za97] John A. Zachman. Enterprise Architecture: The Issue of the Century. *DATABASE PROGRAMMING AND DESIGN*, 1997.

[Zd07] Uwe Zdun. Systematic pattern selection using pattern language grammars and design space analysis. *Software - Practice and Experience*, 37(9):983 – 1016, 2007.

[Zi94] Walter Zimmer. Relationships between design patterns. In *In Pattern Languages of Program Design*, pages 345 – 364. Addison-Wesley, 1994.

Index