Enterprise Architecture Management Patterns

Alexander M. Ernst
Technische Universität München
Germany
ernst@in.tum.de

ABSTRACT

This article introduces the concept of enterprise architecture management (EAM) patterns, a pattern based approach for EA management. Three different types of patterns are presented. M-Patterns document proven-practice methodologies to address typical problems in EA management. V-Patterns represent best-practice visualizations, whereas I-Patterns indicate information requirements for EA management. These patterns build up a pattern language for EA management, with an excerpt given in this article.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Patterns*

General Terms

Design

Keywords

Patterns

1. INTRODUCTION AND OVERVIEW

This article gives a short introduction to enterprise architecture (EA) management and the concept of EAM patterns to address typical recurring problems arising in this area.

1.1 Intended Audience

This article and the herein included patterns are intended for people concerned with governing the information technology (IT) of a company, aligning business and IT, standardizing and managing architectures of business applications, or controlling of the infrastructure of a company.

Stakeholders for this article are: enterprise architects, business application owners, solution architects, people concerned with architectural standards.

1.2 Enterprise Architecture Management

EA management is one of the major challenges of modern enterprises. It aims at aligning business and IT in order to optimize their interaction.

Enterprise architectures include everything that is needed to run a business, ranging from strategies (business, as well as IT strategies are of interest), via business processes, representing the value chain of the company, and business applications needed to support the business processes to infrastructure elements, like e.g. application servers or hardware.

Documenting and managing the EA is an advanced topic, as the application landscape, which is part of the EA often includes a few hundreds up to a few thousand business applications and their interconnections in a medium-sized or large company. 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) [1], which determine the information a company has to have available about its EA.

Therefore, typical problems are likely to arise across companies. Whatever preliminary work on EA management exists in a company, there commonly is a demand for a more structured way to manage the evolution of the EA. A variety of approaches to introduce EA management have been proposed by academia and practice (see e.g. [4, 13, 2]), but they all have to cope with at least one of the following problems:

- EA management is **introduced from scratch**, not considering related initiatives already present inside or outside the organization.
- EA management frameworks, like Zachman [28], TOGAF [16], etc., are usually either **too abstract** and therefore **difficult to implement**, or **too extensive** to be used in practice, as they have to be utilized as a whole.
- Lacking an actual **starting point** for an EA management initiative, companies tend to collect requirements from potential EA stakeholders in the organization. Consolidating their demands and integrating their information needs, an **all-embracing EA management approach** is likely to emerge, which requires **a vast amount of data to be gathered**. Leading to a labor-intensive and time-consuming information maintaining process, although only a part of the information would be needed to address the real pain points of the company.

- If an approach has been implemented, it is often **not documented** why certain decisions have been taken, e.g. why a certain concept has been chosen to be documented. This leads to models, which cannot be adapted or extended, due to the fact that **no one knows** what analyses rely on which concepts.
- Approaches proposed, e.g. by organizations or standardization groups, are usually **all or nothing** approaches. They are supposed to be introduced as one single piece and do not support an incrementally developing EA management endeavor. This results in an EA management approach that is not tailored to the company's EA maturity.

In order to address the problems stated above, we propose to apply **patterns**, well known from other disciplines like architecture or software engineering. This entails according to [17] further advantages like enabling 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 pattern exist, see e.g. [3], [8], or [14], but adhere to a common basis.

Patterns **are a general, reusable solution to a common problem** and **are dependent on their context**.

These properties are the basis for the EAM pattern approach, which was initially introduced in [7]. EAM patterns document solutions to typical recurring problems in EA management, based on proven practices.

1.3 EAM Patterns

The EAM patterns follow a template for pattern documentation similar to Buschmann et al. [8] consisting of the sections:

Name, Short Description, Example, Context, Problem, Solution, Implementation, Variants, Known Uses, Consequences, See Also, and Credits [1]. Versioning information and an identifier have been added to this template.

[7] introduced three types of EAM patterns:

Methodology Patterns *(M-Pattern)* specify a methodology to address management problems in a stepwise manner. The procedures defined by the M-Pattern can be very different, ranging from e.g. visualizations and group discussions to more formal techniques as e.g. metrics calculations [22]. M-Patterns have been introduced, because missing methodologies constitute a common issue in current EA management approaches. Frameworks as e.g. TOGAF [16] provide process models (e.g. TOGAF ADM), but leave the details of the methodologies supporting the specific activities in the EA management process relatively open. M-Patterns explicate the methodologies in order to complement activities carried out in an ad-hoc manner or relying on implicit knowledge with activities carried out more systematically.

Viewpoint Patterns *(V-Pattern)* provide visualizations like diagrams, reports, etc., which are practically proven to be adequate to address problems in EAM. The data required to produce the visualization is documented in one or more I-Patterns. Industrial users often specify viewpoints by example, meaning that an exemplary view is provided for the viewpoint, possibly together with some textual explanations. This 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 view according to a viewpoint, a legend should be mandatory. V-Patterns can be used as a utility by one or more M-Patterns.

Information Model Patterns *(I-Pattern)* supply best-practice information model fragments, including definitions and descriptions of the used concepts, which can be used to collect information to address a certain problem in EA management. This information can then be visualized in views according to one or more V-Patterns or be used directly by M-Patterns. [7] shows that different languages are possible for describing an I-Pattern, varying in their degree of formality, including among others textual descriptions in natural language, the Meta Object Facility (MOF), Unified Modeling Language (UML) class diagrams, ontology languages, and mathematical formalizations, or combinations of these languages. Choosing a specific language basically has to consider the needs of the use cases to be supported. While an object-oriented description might be sufficient for creating a visualization or a tabular report, e.g. process simulation may only be reasonably possible on a more formal basis. Therefore, a language adequate to the problem to be addressed should be used, thereby strongly considering UML as the default language, as it is widely understood and has been found by us to be problem-adequate in many practical settings in the context of EA management information models [6]. In case a language different from UML is chosen, complementing its specification with an UML-based description can yield advantages, especially as integrating information model patterns is simplified by them being available in a common language.

It is important to mention that the EAM pattern approach is problem driven. Problems, also known as pain points, are usually the entry point for management activities in EA management and are therefore an integral part of all EAM patterns.

The EAM patterns build up a pattern language, which has been documented in the *EAM Pattern Catalog* [5]. In order to continuously improve and extend the EAM patterns they have been included in a wiki at `\URLEAMPCWiki`. It is advised to look there for the latest version of the EAM patterns as well as for other people interested in this topic. The *EAM Pattern Catalog* Wiki also includes more information on the different usage scenarios of the EAM pattern approach.

As already mentioned before, patterns constitute reusable solutions to common problems observed in practices. In order to identify common problems and patterns for addressing them an extensive survey the *Enterprise Archi-*

[1]The credits section is omitted in this article for the individual EAM patterns but is summarized in Section 5.1 in order to improve readability of this article.

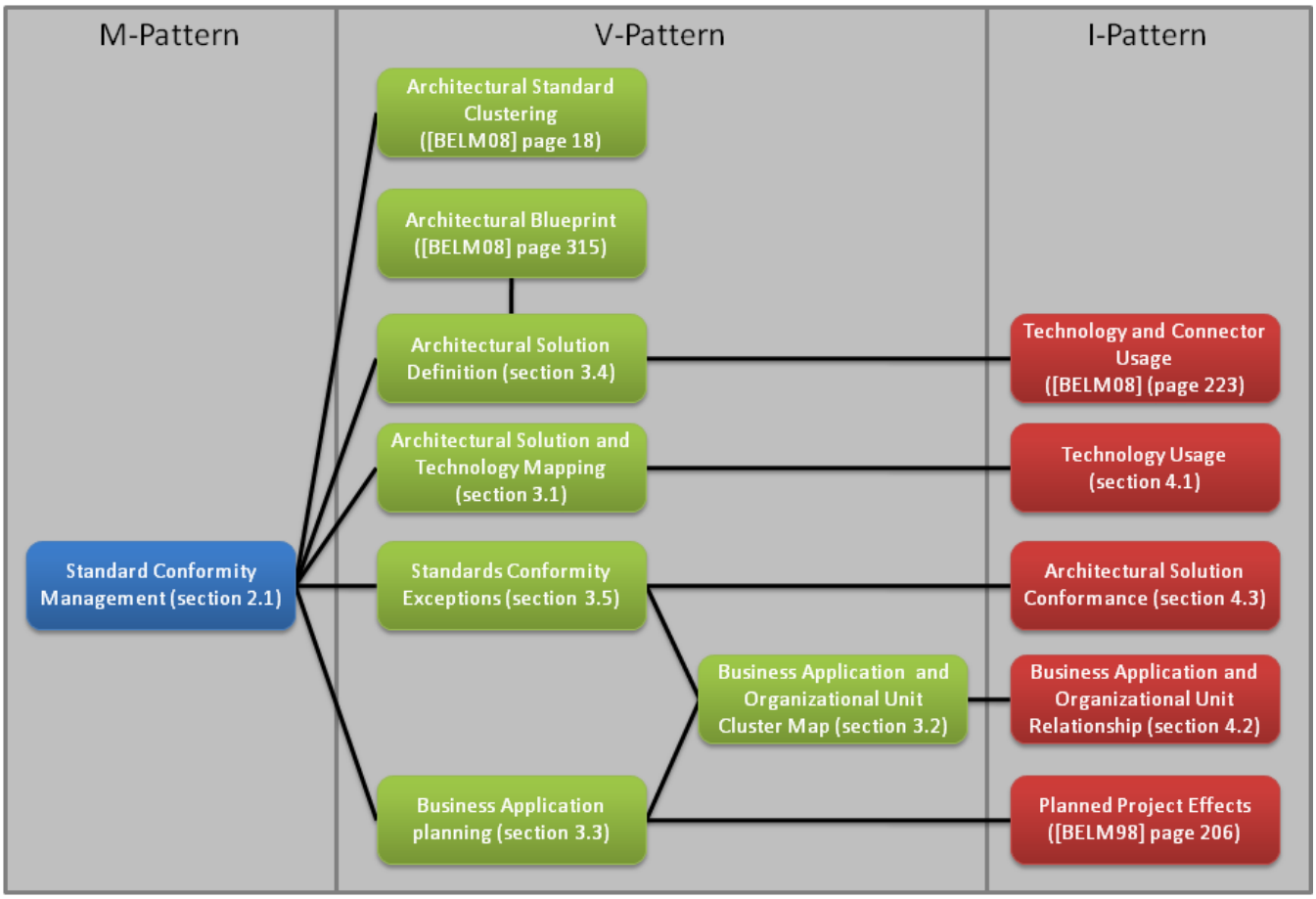

Figure 1: Pattern Map for this Article

tecture Management Viewpoint Survey (EAMVS) has been conducted. Thereby, in a first phase (October 2006 until July 2007), the *EAM Pattern Catalog* was initialized by our group based on input from the following sources:

- Research project Software Cartography, Technische Universität München, Chair for Informatics 19 (e.g. [6, 7, 21, 27])
- Partners of the research project Software Cartography
- EAM Tool Survey 2005 [24] and EAM Tool Survey 2008 [23]
- Enterprise Architecture at Work (ArchiMate) [18]
- Management von IT-Architekturen (Edition CIO) [11]
- IT-Unternehmensarchitektur, 2007 [19]

In a second phase (July 2007 until February 2008), the initial *EAM Pattern Catalog* was evaluated by 30 companies using an extensive online questionnaire to identify methodologies and viewpoints that are considered relevant and useful by practitioners[2].

Based on the evaluation of the questionnaire results, the *EAM Pattern Catalog* in its present form covers

- 43 concerns[3] (48 have been excluded due to the questionnaire evaluation),
- 20 methodologies (10 have been excluded due to the questionnaire evaluation),
- 53 viewpoints (21 have been excluded due to the questionnaire evaluation), and
- 47 information model fragments (19 have been excluded due to the questionnaire evaluation).

1.4 Map of EAM Patterns

The EAM patterns included in this article are part of a larger pattern language and therefore relationships between EAM patterns are an integral part of this approach. Figure 1 shows a pattern map visualizing these relationships. The patterns are referenced by their names, page numbers are included in brackets.

[2]See [5] for details of the selection process as well as relevance and usage statistics for each element.

[3]During the creation of this article the EAM pattern approach has been revised, e.g. the concerns of version 1.0 of the *EAM Pattern Catalog* have been split up into problems and forces, etc. Therefore, concerns are not explicitly covered in the rest of this article. A typical concern in EA management could be *How can licensing costs for business applications and infrastructure be reduced?*

2. METHODOLOGY PATTERNS

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

2.1 Standard Conformity Management

Profile	
Id	M-4
Version	2.0

The M-Pattern 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.

2.1.1 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.

2.1.2 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.

2.1.3 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.) 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?

2.1.4 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.

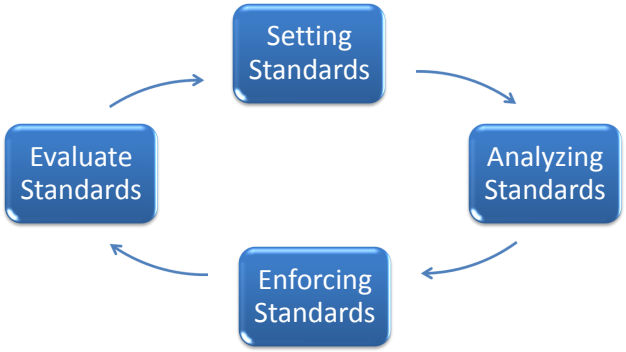

Figure 2: Management process for Standard Conformity Management

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 2), architectural standards which are no longer feasible have to be adapted or new standards have to be created.

The implementation section of this I-Pattern will additionally cover the aspect of involving the right people and establishing the right governance structures.

2.1.5 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 [5] showed that the first two items are most important to practitioners.[4] 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 [9]. This leads to different possible notations for *defining* architectural blueprints:

- We propose V-Pattern *Architectural Solution Definition* (see section 3.4), which is based on the respective UML-notation in [9].
- V-Pattern *Architectural Blueprint* (see page 315 in [5]) 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* [15] 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.

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

Several aspects may influence which and how many architectural standards are offered.

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.

2.1.6 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 section 3.4).

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 section 3.1) 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 section 3.5) and *Architectural Standard Clustering* (see page 101 in [5]). *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 [5]) 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.

2.1.7 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 section 3.3) 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.

2.1.8 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.

2.1.9 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 [10] 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.

2.1.10 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.*

2.1.11 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 [25, 26].

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.

2.1.12 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 [5])
- *Architectural Solution and Technology Mapping* (see section 3.1)
- *Business Application Planning* (see section 3.3)
- *Architectural Solution Definition* (see section 3.4)
- *Standards Conformity Exceptions* (see section 3.5)
- The architectural description language ACME [15]

3. VIEWPOINT PATTERNS

This section contains the following selection of V-Patterns, which are part of the *EAM Pattern Catalog* [5].

- Architectural Solution and Technology Mapping (see section 3.1)
- Business Application and Organizational Unit Cluster Map (see section 3.2)
- Business Application Planning (see section 3.3)
- Architectural Solution Definition (see section 3.4)
- Standards Conformity Exceptions (see section 3.5)

3.1 Architectural Solution and Technology Mapping

Profile	
Id	V-23
Version	2.0

This V-Pattern consists of a table containing the technologies used in architectural solutions.

3.1.1 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.

3.1.2 Context

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

		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 3: Exemplary view for V-Pattern *Architectural Solution and Technology Mapping*

3.1.3 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?

3.1.4 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.

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.

3.1.5 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.

3.1.6 Variants

Different variants for this V-Pattern exist. The information shown in Figure 3 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.

3.1.7 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)

3.1.8 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.

3.1.9 See Also

This V-Pattern may be useful when using M-Pattern *Standard Conformity Management* (see section 2.1). The visualized information is based on I-Pattern *Technology Usage* (see section 4.1).

3.2 Business Application and Organizational Unit Cluster Map

Profile	
Id	V-24
Version	2.0

This V-Pattern visualizes relationships between business applications and organizational units by using the concept of clustering.

3.2.1 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.

3.2.2 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.

3.2.3 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?

3.2.4 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 4 exemplarily visualizes a hosting relationship. This 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.

3.2.5 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.

3.2.6 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.

3.2.7 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*.

3.2.8 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*.

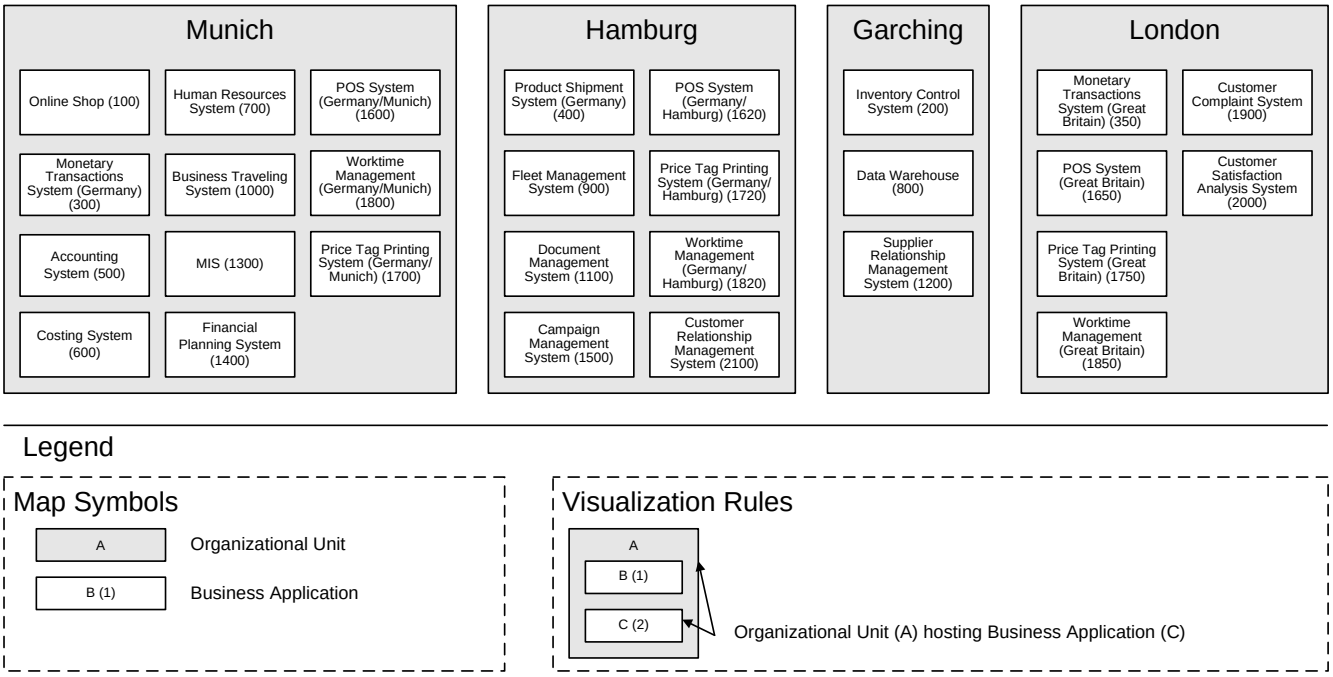

Figure 4: Exemplary view for V-Pattern *Business Application and Organizational Unit Cluster Map*

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.

3.2.9 See Also

The V-Pattern is based on information according to I-Pattern *Business Application and Organizational Unit Relationship* (see section 4.2) 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 section 3.3)
- *Standards Conformity Exceptions* (see section 3.5)

3.3 Business Application Planning

Profile	
Id	V-39
Version	2.0

This V-Pattern visualizes changes to business applications or the introduction of new ones using a color coding. It can be used to perform application landscape planning.

3.3.1 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.

3.3.2 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.

3.3.3 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?

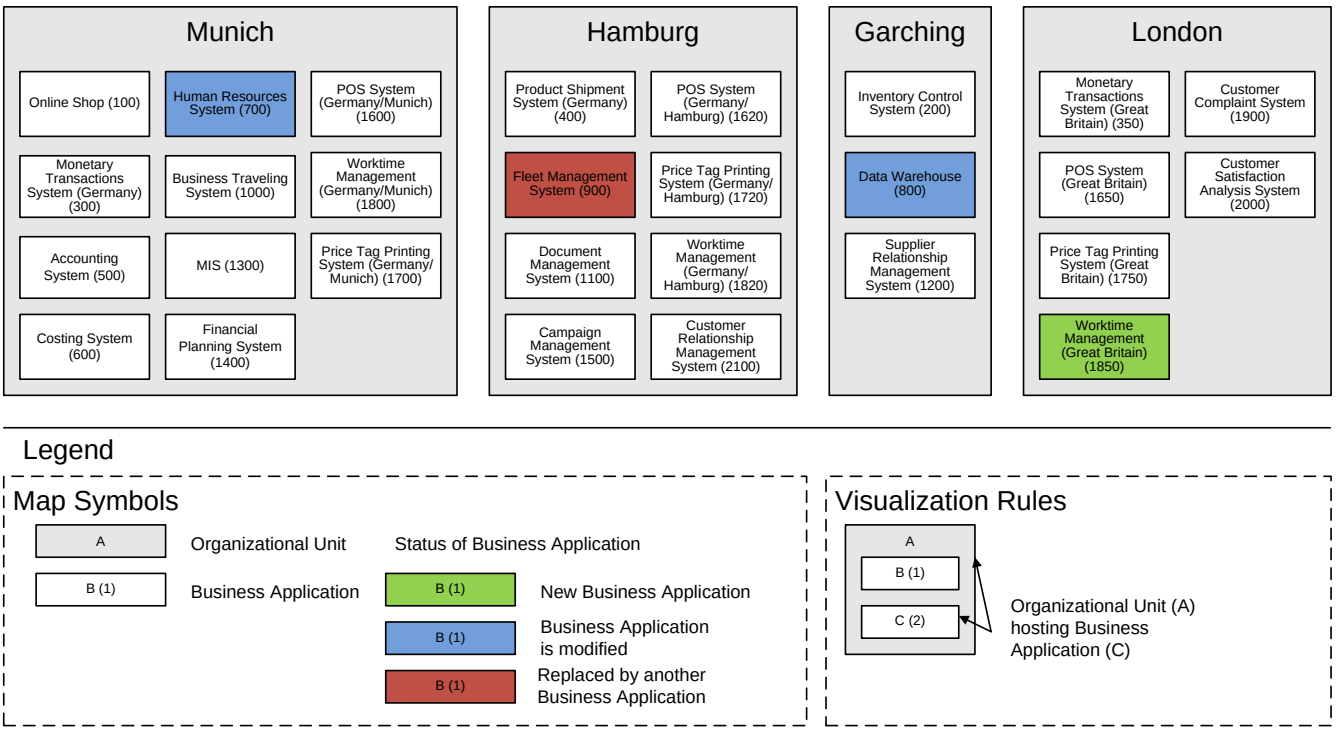

Figure 5: Exemplary view for V-Pattern *Business Application Planning*

3.3.4 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 5 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, 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 3.3.

3.3.5 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[5].

When there is a demand to utilize the layering concept, it is advised to use a tool supporting this functionality.

3.3.6 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 [5]) 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.

3.3.7 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)

3.3.8 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.

[5] See [12] for more details on the layering principle.

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.

3.3.9 See Also

Creating views based on this V-Pattern requires to collect information according to I-Pattern *Planned Project Effects* (see page 206 in [5]) 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 section 4.2) or its alternatives.

3.4 Architectural Solution Definition

Profile	
Id	V-66
Version	2.0

This V-Pattern uses an UML 2.0 object diagram to visualize an architectural solution and the technologies used.

3.4.1 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.

3.4.2 Context

Defining architectural standards and maintaining them is difficult, if various architectural standards and different notations are in use.

3.4.3 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?

3.4.4 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 6.

3.4.5 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.

3.4.6 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.

3.4.7 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*.

3.4.8 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.

3.4.9 See Also

Another V-Pattern, called *Architectural Blueprint* (see page 315 in [5]) 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 [5]) to visualize, the relationships between technologies, connectors, and abstract technologies.

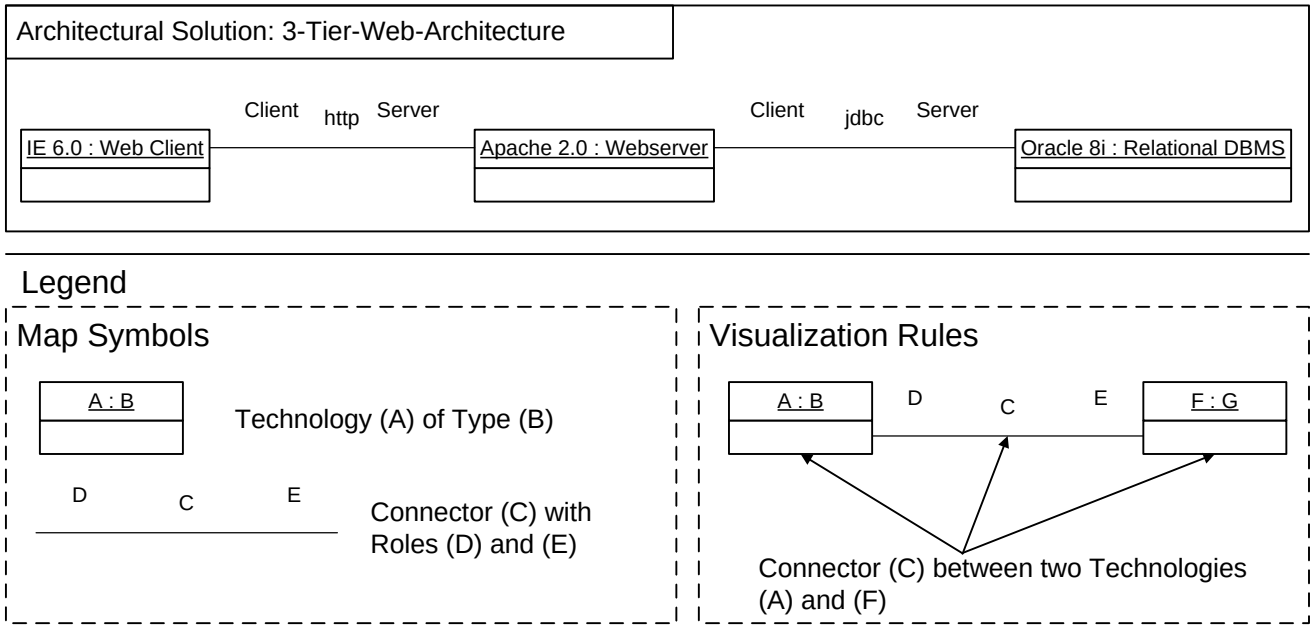

Figure 6: Exemplary view for V-Pattern *Architectural Solution Definition*

3.5 Standards Conformity Exceptions

Profile	
Id	V-67
Version	2.0

This V-Pattern 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.

3.5.1 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.

3.5.2 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.

3.5.3 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?

3.5.4 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 section 3.2), 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 here is favorable as it provides a good and intuitive overview about the relationships described before.

Figure 7 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.

3.5.5 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.

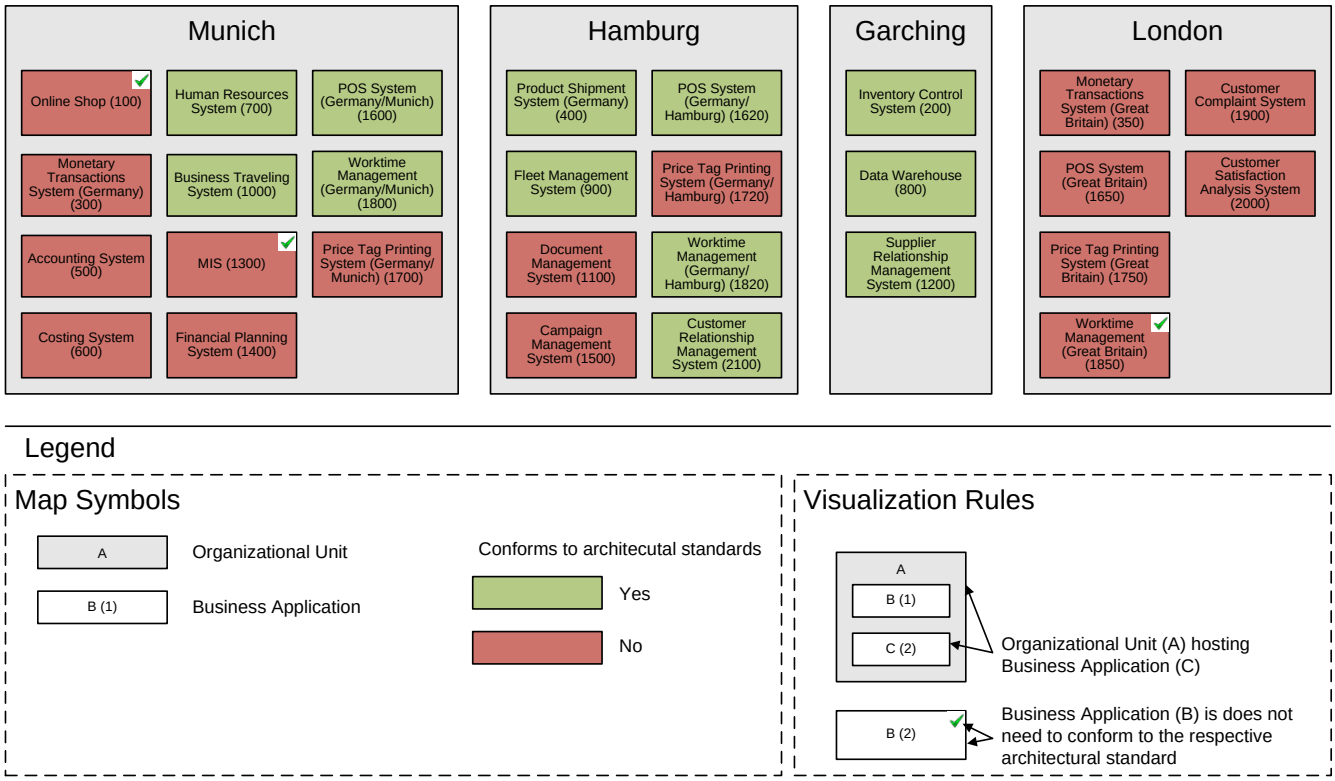

Figure 7: Exemplary view for V-Pattern *Standards Conformity Exceptions*

3.5.6 *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 section 3.2) 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 [5]) additionally offers the possibility to analyze the standardization of business applications in respect to business processes.

In contrast to Figure 7 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.

3.5.7 *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)

3.5.8 *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.

3.5.9 *See Also*

Creating views based on this V-Pattern requires to collect information according to I-Pattern *Architectural Solution Conformance* (see section 4.3) 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 section 4.2) or its alternatives.

4. INFORMATION MODEL PATTERNS

This section contains the following selection of I-Patterns, which are part of the *EAM Pattern Catalog* [5].

- Technology Usage (see section 4.1)
- Business Application and Organizational Unit Relationship (see section 4.2)
- Architectural Solution Conformance (see section 4.3)

4.1 Technology Usage

Profile	
Id	I-23
Version	2.0

This I-Pattern shows how information about the technologies used in an architectural solution, can be stored.

4.1.1 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.

4.1.2 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.

4.1.3 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?

4.1.4 Solution

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. Exemplarly technologies may be "Apache 2.0.53" or "Oracle 9.2i".
- ArchitecturalSolution uses Technology: The association *uses* indicates, which architectural solution uses which technologies.

4.1.5 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.

4.1.6 Variants

The information model fragment shown in Figure 8 may be extended e.g. by additional attributes, like licensing costs for technologies, or more advanced concepts like lifecycles for technologies, as well as for architectural solutions.

4.1.7 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)

4.1.8 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.

4.1.9 See Also

V-Pattern *Architectural Solution and Technology Mapping* (see section 3.1) may be utilized to perform analyses on information stored according to this I-Pattern.

Figure 8: Information model fragment for I-Pattern *Technology Usage*

4.2 Business Application and Organizational Unit Relationship

Profile	
Id	I-24
Version	2.0

This I-Pattern shows how information about business applications and their relationships to organizational units can be stored.

4.2.1 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.

4.2.2 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.

4.2.3 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?

4.2.4 Solution

This I-Pattern consists of two entities *BusinessApplication*, *OrganizationalUnit*, and one relationship *hosts* and is exemplarily visualized in Figure 9. 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 [20] 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.

4.2.5 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.

4.2.6 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

Each of these different semantics results in a variant of this I-Pattern. An example for the *uses* variant is shown in Figure 10.

The following definitions specifiy 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.

Figure 9: Information model fragment for I-Pattern *Business Application and Organizational Unit Relationship*

Figure 10: Information model fragment for variant of I-Pattern *Business Application and Organizational Unit Relationship* based on the uses variant

4.2.7 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)

4.2.8 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 BusinessApplication-Versions for a BusinessApplication, one should consider to change the BusinessApplication in this I-Pattern to the BusinessApplicationVersion.

4.2.9 See Also

This I-Pattern is the basis for V-Pattern *Business Application and Organizational Unit Cluster Map* (see section 3.2) and its variants.

4.3 Architectural Solution Conformance

Profile	
Id	I-67
Version	2.0

This I-Pattern shows how information about business applications and their conformity to architectural solutions, can be stored.

4.3.1 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.

4.3.2 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.

4.3.3 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?

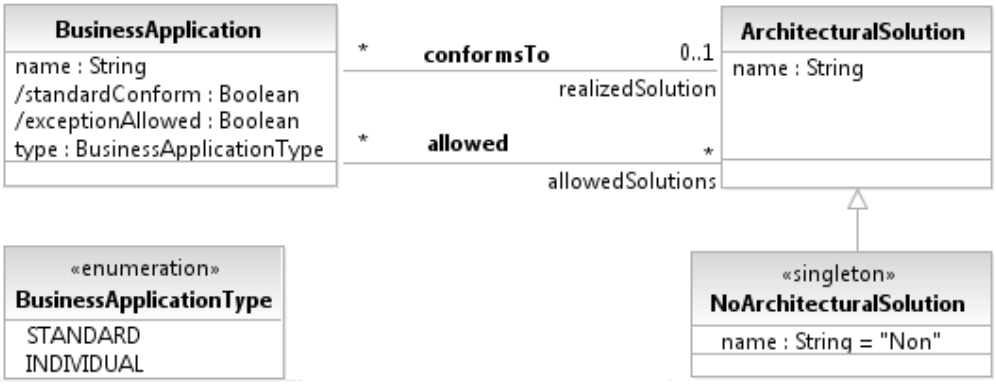

Figure 11: Information model fragment for I-Pattern *Architectural Solution Conformance*

4.3.4 Solution

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 [20] 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 7, the derived attributes *standardConform* and *exceptionAllowed* are used. The values of these attributes are derived by expressions similar to the following[6]:

$$standardConform = \begin{cases} null & \text{for } (realizedSolution = null) \vee \\ & \quad (allowedSolutions = null) \\ true & \text{for } realizedSolution \in allowedSolutions \\ false & \text{for } realizedSolution \notin allowedSolutions \end{cases}$$

respectively

$$exceptionAllowed = \begin{cases} null & \text{for } allowedSolutions = null \\ true & \text{for } NoArchitecturalSolution \in \\ & \quad allowedSolutions \\ false & \text{for } NoArchitecturalSolution \notin \\ & \quad allowedSolutions \end{cases}$$

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.

[6]These expressions might also be realized in the Object Constraint Language (OCL). For reasons of readability, we chose a mathematical notation instead.

- **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.

4.3.5 Implementation

This I-Pattern should be implemented in a repository, which is able to guarantee consistency for the derived attributes *standardConform* and *exceptionAllowed*.

4.3.6 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.

4.3.7 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)

4.3.8 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.

4.3.9 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 [5].

This I-Pattern can be used to manage information for V-Pattern *Standards Conformity Exceptions* (see section 3.5).

5. ACKNOWLEDGMENT AND OUTLOOK

This section includes acknowledgments to the people who supported the creation of this article and gives an outlook to the next steps in the development of the EAM pattern approach.

5.1 Acknowledgments

I want to thank all participants of the "Design & Architecture" workshop of PloP08 (Ademar Aguiar, Filipe Figueiredo Correia, Hugo Sereno Ferreira, Nuno Flores, Ralph Johnson, Trevor Parsons, Srinivas Rao, Peter Sommerlad, Peter Swinburne, Rebecca Wirfs-Brock, and Joseph W. Yoder) and especially my shepherd Peter Sommerlad for the time they spent for reading, commenting, and discussing this article.

Additionally, I want to thank my colleagues of the System Cartography research group (Sabine Buckl, Josef Lankes, Florian Matthes, and Christian M. Schweda) for their help in developing the initial versions of the patterns included in this article.

5.2 Next Steps in EAM Pattern Approach Development

The *EAM Pattern Catalog* is currently available at http://www.systemcartography.info/eampc-wiki as a first version, based on the results of an extensive online survey. In order to improve the current version and to further exploit the advantages of patterns in EA management, an excerpt of the *EAM Pattern Catalog* had been included in this document to be discussed in the pattern community.

During the shepherding phase and the writers' workshop the following changes have been carried out:

- The initial version of EAM patterns followed a uniform structure, which is different than the ones usually used for documenting patterns. Therefore, a template for pattern documentation similar to Buschmann et al. [8] has been used (see Section 1.3).
- The initial version of EAM patterns were highly dependable on each other, especially the references between the different EAM pattern (M-, V-, and I-Patterns) types. In addition a problem section was only included in the M-Patterns. This has been changed to make EAM patterns more self contained.
- The initial version of EAM patterns addressed more than one concern (problem) with one M-Pattern. Analyzing the concerns they have been split up in context, problem and forces.
- Example, context, and problem sections have been revised to be more focused on the content of the EAM patterns.

The changes mentioned before currently only have been carried out for the EAM patterns included in this document. In a next step the remaining patterns of the *EAM Pattern Catalog* will be adapted to reflect these changes.

Certainly, the EAM patterns should continually be revised for readability and understandability and be extended to give more detailed guidance in addressing the problems of EA practitioners, preferably by an *EAM Pattern Catalog* community.

6. REFERENCES

[1] 107th Congress of the United States. Sarbanes-oxley act of 2002 (public law 107–204). http://www.sec.gov/about/laws/soa2002.pdf, 2002.

[2] Aier, S., Kurpjuweit, S., Riege, C., and Saat, J. Stakeholderorientierte dokumentation und analyse der

unternehmensarchitektur. In *GI Jahrestagung (2)* (2008), pp. 559–565.

[3] Alexander, C., Ishikawa, S., and Silverstein, M. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, USA, 1977.

[4] Braun, C., and Winter, R. A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform. In *EMISA 2005* (2005), pp. 64 – 79.

[5] Buckl, S., Ernst, A., Lankes, J., and Matthes, F. Enterprise architecture management pattern catalog (version 1.0, february 2008). Tech. rep., Technische Universität München, Chair for Informatics 19 (sebis), Munich, 2008.

[6] Buckl, S., Ernst, A., Lankes, J., Matthes, F., Schweda, C., and Wittenburg, A. Generating visualizations of enterprise architectures using model transformation (extended version). *Enterprise Modelling and Information Systems Architectures - An International Journal Vol. 2*, 2 (2007).

[7] Buckl, S., Ernst, A., Lankes, J., Schneider, K., and Schweda, C. A pattern based approach for constructing enterprise architecture management information models. In *Wirtschaftsinformatik 2007* (Karlsruhe, Germany, 2007), A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, and Schnizler, Eds., Universitätsverlag Karlsruhe, pp. 145–162.

[8] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. *Pattern-oriented software architecture: a system of patterns.* John Wiley & Sons, Inc., New York, NY, USA, 1996.

[9] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., and Stafford, J. *Documenting Software Architectures: Views and Beyond.* Addison Wesley, Boston, 2002.

[10] Coplien, J., and Harrison, N. *Organizational Patterns of Agile Software Development.* Prentice Hall PTR, Indianapolis, 2004.

[11] Dern, G. *Management von IT-Architekturen (Edition CIO).* Vieweg, Wiesbaden, 2006.

[12] Ernst, A., Lankes, J., Schweda, C., and Wittenburg, A. Using model transformation for generating visualizations from repository contents - an application to software cartography. Tech. rep., Technische Universität München, Chair for Informatics 19 (sebis), Munich, 2006.

[13] Frank, U. Multi-perspective enterprise modeling (memo) - conceptual framework and modeling languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences 35 (2002)* (2002), pp. 1258–1267.

[14] Gamma, E., Helm, R., R., J., and Vlissides, J. M. *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series).* Addison-Wesley Professional, 1994.

[15] Garlan, D., Monroe, R., and Wile, D. Acme: An architecture description interchange language. In *CASCON'97* (Toronto, Ontario, Canada, 1997), IBM Press.

[16] Group, T. O. *TOGAF (The Open Group Architecture Framework). Version 8.1 "Enterprise Edition".* The Open Group, 2003.

[17] Harrison, N. B., and Avgeriou, P. Leveraging architecture patterns to satisfy quality attributes, 2005.

[18] Jonkers, H., Groenewegen, L., Bonsangue, M., and van Buuren, R. A language for enterprise modelling. In *Enterprise Architecture at Work*, L. Marc., Ed. Springer, Berlin, Heidelberg, New York, 2005.

[19] Keller, W. *IT-Unternehmensarchitektur.* dpunkt.verlag, 2007.

[20] Krcmar, H. *Informationsmanagement*, 4 ed. Springer, Berlin et al., 2005.

[21] Lankes, J., Matthes, F., and Wittenburg, A. Softwarekartographie : Systematische darstellungen von anwendungslandschaften. In *Wirtschaftsinformatik 2005* (Bamberg, Germany, 2005), Physica-Verlag, p. 1443–1462.

[22] Lankes, J., and Schweda, C. M. Using metrics to evaluate failure propagation and failure impacts in application landscapes. In *Multikonferenz Wirtschaftsinformatik* (2008), GITO-Verlag, Berlin.

[23] Matthes, F., Buckl, S., Leitel, J., and Schweda, C. *Enterprise Architecture Management Tool Survey 2008.* Technische Universität München, Chair for Informatics 19 (sebis), 2008.

[24] sebis. *Enterprise Architecture Management Tool Survey 2005.* Technische Universität München, Chair for Informatics 19 (sebis), 2005.

[25] Star, S. L., and Griesemer, J. R. Institutional ecology: "translations" and coherence: Amateurs and professionals in berkeley's museum of vertebrate zoology. *Social Studies of Science Vol. 19*, 3 (1989), 387–420.

[26] Strübing, J. Soziale welten – arenen – grenzobjekte. In *29. Kongress der Deutschen Gesellschaft für Soziologie (Sektion Wissenschafts- und Technikforschung)* (Freiburg, 1999).

[27] Wittenburg, A. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften.* PhD thesis, Fakultät für Informatik, Technische Universität München, 2007.

[28] Zachman, J. Extending and formalising the framework for information systems architecture. *IBM Systems Journal Vol. 31*, 3 (1992).