From EA management patterns towards a prescriptive theory for designing enterprise-specific EA management functions

Outline of a research stream

Sabine Buckl, Florian Matthes, Christian M. Schweda

Chair for Software Engineering of Business Information Systems,
Technische Universität München
{*sabine.buckl | matthes | schweda*}*@in.tum.de*

1 Motivation

Today's enterprises find themselves confronted with a changing economic, regulatory, and technical environment that they are forced to adapt to (Ross et al. 2006, Wagter et al. 2005). Performing the necessary and beneficial adaptations is aggravated by the intricate and highly interwoven architecture of the overall enterprise. Therein, *local* changes to one enterprise artifact, e.g. a business process or a business application, might have unforeseen *global* consequences at and potentially detrimental impacts on related artifacts. The enterprise architecture (EA) describes the interwoven system of these artifacts and their cause-effect relationships. Therefore, comprehensive knowledge about the EA may support an enterprise to avoid or reduce negative side-effects of adaptations, and may help to leverage the full benefits that can be drawn from an environmental change. Supporting the managed evolution of the enterprise's architecture with a focus on business/IT alignment is the core task of EA management. Driven by the demand from practice, a multitude of approaches to EA management has been developed by practitioners (cf. Niemann 2006, Schekkerman 2008), researchers (cf. Aier et al. 2008a, Ferstl and Sinz 2005, and Frank 2002), and standardization bodies (cf. The Open Group 2009).

The research field of EA management is a relatively new but emerging one. One of the first publications concerning the topic EA management was written by Zachman (1987). Since that time the number of publications targeting the topic is increasing (Langenberg and Wegmann 2004). From there, different approaches rooted in various disciplines as e.g. meta-modeling (cf. Frank 2002) or systems

sciences (cf. Wegmann 2002) have evolved. Based on their perspective, the approaches take differing means to deal with intrinsic challenges of EA management, or even provide different definitions for their object of investigation (cf. Schönherr 2008). This plurality in research reflects the plurality of application scenarios of EA management for which Aier et al. (2008b) give the following non-comprehensive list of examples: process optimization, architectural conformity of projects, quality and security management, and strategic planning. In all of these application scenarios, EA management has to involve a multitude of stakeholders from different parts of the enterprise with various backgrounds (van der Raad et al. 2008).

In this vein, EA management forms an intricate subject for research, which is typically addressed by an engineering approach, aiming at solving problems relevant in practice (cf. Aier et al. 2008a). Although we do not doubt the importance of the thereby achieved results, the contributions remain isolated and a comprehensive understanding of the EA management function has therefore not yet developed. A promising contribution in this direction is the EAM Pattern Language proposed by Ernst (2008), which aims at providing a set of building-blocks for an enterprise-specific EA management function (see Section 2). This language is revisited in Section 3 from a design perspective and interpreted as the basis for a prescriptive theory for designing EA management functions. Thereby, yet not developed elements for a prescriptive theory are elicited. In response, Section 4 proposes a framework guiding future EA management research targeting a prescriptive theory. Final Section 5 provides an outlook on future research streams in this area.

2 EA management pattern language

Patterns have a long history as useful means for structuring and solving problems in complex domains, dating back to Alexander (1979), who introduced patterns as "coherent and modular solutions to specific problems". Subsequent publications (cf. Buschmann et al. 1996 or Gamma et al. 1994) have since then refined the term pattern and put forward structuring guidelines for the description of patterns. A well-known and broadly accepted structure is presented by Buschmann et al. (1996), according to which a pattern comprises the following elements:

- *Context description* is concerned with causes and environmental factors that may have lead to the problem that the pattern solves.
- *Problem description* alludes to the issues and difficulties that occur in many contexts and may be solved with the pattern. Thereby, the description expatiates on conflicting forces that comprise the problem.
- *Solution description* explains the steps to be taken and the concepts to be used in order to solve the corresponding problem.
- *Consequences description* refers to consequences that may be caused by applying the pattern to the given problem.

Building on the idea of patterns, Buckl et al. (2007) coined the term of the *EA management pattern* as a way to structure the domain of EA management. Ernst (2008) further develops this idea towards a *pattern language* for this topic. This pattern language introduces three types of patterns, namely *methodology pattern* (M-Pattern), *viewpoint pattern* (V-Pattern) and *information model pattern* (I-Pattern), that are used to develop an enterprise-specific EA management function. These three types of patterns, subsumed under the term *EAM pattern*, describe constituents of proven-practice solutions for EA management as found in literature but also in partnering enterprises (sebis 2009). The different types of patterns contribute different parts to an EA management function, detailed as follows:

- *M-Patterns* describe management methods (and processes) that solve a specific EA-related problem. Thereby, a pattern provides step-by-step guidance and role-specific information on what and how to do.
- *V-Patterns* describe viewpoints, i.e., types of visualizations that are employed by an M-Pattern in order to communicate solution-relevant information about the EA.
- *I-Patterns* describe conceptual models, whose concepts are instantiated to documentations of solution-relevant parts of the overall EA.

Summarizing, M-, V-, and I-patterns in conjunction provide prescriptions on how to solve EA-related problems with an EA management function. In this respect, a problem-specific EA management function can be designed from corresponding EAM patterns, whose proposed solutions are instantiated into processes, viewpoints, and conceptual models. Nevertheless, not only the solutions described in the EAM patterns play an important role for designing an EA management function. The context descriptions of the EAM patterns can help during design to choose the appropriate patterns that optimally fit the organizational context, in which the management function should be embedded. The problem descriptions are the starting points for selecting the "right" EAM pattern, i.e., those patterns that solve the enterprise-specific problems. If different patterns were applicable to similar problems and hence were to be decided upon during the design process, the context and consequence description can provide additional help to choose the patterns that optimally balance desired outcomes and side-effects. Finally, the interrelationships between the patterns, which are described as part of the pattern language, supports the identification of patterns that might also apply in the given context or may be helpful for solving related problems.

Exemplifying the pattern-based approach towards EA management, we detail the M-Pattern *Standard Conformity Management* (cf. sebis 2009, M-4). This pattern describes methods for defining and managing architectural standards, i.e., prescriptions on the technologies and architectures that ought to be used throughout the enterprise. The pattern is typically applied in contexts, where enterprises operate a larger number of business applications and experience increasing problems in

maintaining the applications due to the plurality of basic technologies. In words of the pattern, the corresponding EA management problem reads as:

You feel the risk of an unmanaged application landscape with a multitude of technologies that will increase the cost of development of new business applications, evolution and retirement of existing business applications. You do not know, if the applications follow a common architectural style and what the impact of a change in these standards would be.(sebis 2009, M-4)

Typically conflicting forces that describe a trade-off in architectural standardization are *technological appropriateness* vs. *technological homogeneity*, or *standard compatibility* vs. *vendor lock-in*. In response to the problem and its intrinsic forces, the pattern defines architectural standards via the concepts of *architectural blueprints* and *architectural solutions*. An architectural solution thereby describes a concrete stack of technologies, e.g. the apache web server and a MySQL database, while the complementing architectural blueprint describes a generalized solution consisting e.g. of a web server and a relational DBMS. Complementing the concepts, the M-Pattern introduces an iterative method for setting, analyzing, enforcing, and evaluating the standards. During the analysis phase, the existing architectures and used technologies are gathered and experts are interviewed to assess the suitability of widely applied "pseudo-standards". From this, standards are derived, committed by the enterprise architects, and communicated, e.g. via an EA management tool. For enforcing the standards different means are proposed, ranging from purely informing the decision makers on the standard's existence to financial penalties for projects that develop applications not conforming to the standards. For the final phase, the pattern proposes to revisit the standards on a regular basis in response to changing environmental factors, as e.g. emerging technologies. Figure 1 summarizes the iterative standard management method proposed in the pattern.

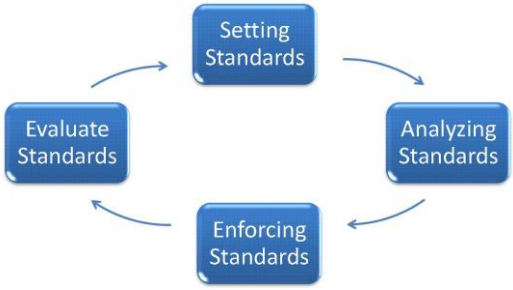

Figure 1: Iterative Standard Management Method (sebis 2009, M-4)

Different consequences of applying the pattern are finally denoted by sebis (2009, M-4), of which two exemplary ones are subsequently summarized. Firstly, the benefits created by architectural and technological homogeneity are mostly middle- to long-term benefits opposed to short-term costs caused by the replacement of a technology or the refactoring of architecture. Secondly, a homogenized set of

technologies can change the distribution of costs between central operations functions and application-using business units, as an infrastructure that is more convenient to operate and hence less expensive may be complemented by a decrease in application performance due to the utilization of standard architectural components.

3 Pattern-based design theories

The pattern-based approach to EA management can be understood as a set of building blocks for constructing an enterprise-specific EA management function. In this vein, it shows characteristics that are typical for *design theories* in the terms of Walls et al. 1992. There, design theories are defined as "prescriptive theories based on theoretical underpinnings which say how a design process can be carried out in a way which is both effective and feasible".

Support for the interpretation of the pattern-based approach as design theory can be found by Schermann et al (2007). They discuss the idea to use pattern languages to describe design theories, and provide a structural framework for pattern-based design theories, which itself was influenced by the work of Gregor (2006) and Gregor and Jones (2007). Below, we present a slightly adapted version of this framework in Figure 2 and utilize it to explore the contribution of the EAM pattern language to a design theory for EA management.

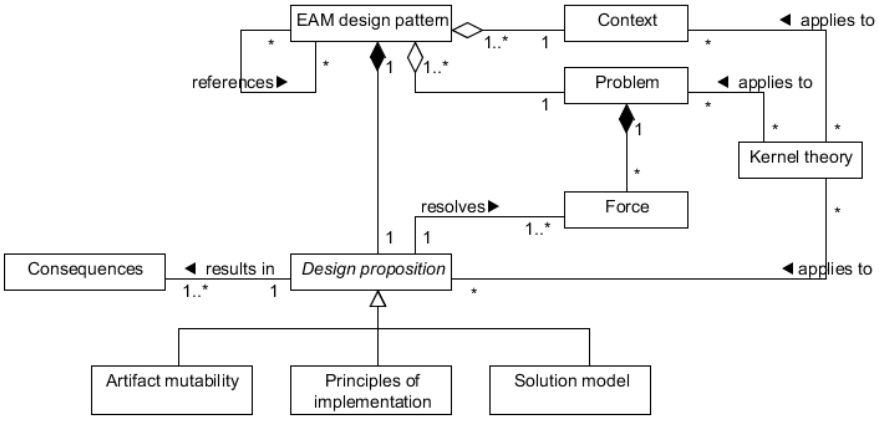

Figure 2: Structural framework for pattern-based design theories

Central constituents of a design theory according to the framework are EAM design patterns that provide design propositions to solve specific EA-problems in given organizational contexts. In accordance to the original work of Walls et al. (1992), two types of design propositions can be distinguished: design artifact-related ones and design method-related ones. Resorting to more common terms

proposed by Schermann et al. (2007) and Gregor and Jones (2007) the corresponding concepts in the framework are named solution model, principles of implementation and artifact mutability, of which the latter two objects are method-related. A design proposition further denotes the problem-inherent forces it is intended to balance and delineates its corresponding consequences. Complementing context, problem, and design proposition, a design theory can supply kernel theories that explain or structure the corresponding context, problem, or solution domains. In accordance to Walls et al. (1992), we regard those theories to be optional and allow for multiple theories, if necessary.

Schermann et al. (2007) focus on patterns providing solution models, i.e., on artifact-related patterns. Design method-related patterns are only mentioned briefly with an exemplary reference to Köhne (2005), who describes design processes with patterns. The solution models provided by Schermann et al. (2007) are described by conceptual models for the problem domain. In this vein, they resemble to I-patterns from the EAM pattern language of Ernst (2008). Complementing, the V-patterns provide viewpoints used in an EA management function, but hence stay to the artifact level, i.e., describe constituents for the management function's structure. The same also applies to the M-patterns that describe methods and processes for conducting EA management, but are not concerned with designing an EA management function. Put in other words, the EAM patterns presented in the EAM pattern language provide solution models for the problem domain. Conversely, the design method for an EA management function is only briefly alluded to in the pattern language, although the following tasks during the design process most likely call for additional guidance:

- **Selection** of appropriate EAM-Patterns
- **Integration** of M-Patterns into a consistent EA management process
- **Integration** of I-Patterns into a conceptual model for the EA

For the first tasks, Ernst provides the notion of "usage scenarios" that provide indications on the utilization of the language (Ernst 2008, sebis 2009). These scenarios may be helpful for understanding the basics of selecting appropriate EAM-Patterns during the design or re-design of an EA management function, but stay on an abstract level. When it comes to the integration tasks, the pattern-based approach can rely on well-established design methods in Situational Method Engineering (see e.g. Brinkkemper 1992) or Schema Integration (see e.g. Batini et al. 1986), respectively.

A possible reason for the aforementioned absence of patterns on design methods for the EA management function is discussed by Buckl et al. (2009). There, they apply the viable system model (VSM) as a framework for structuring EA management functions and analyze prominent EA management approaches alongside the subsystems comprising a system in terms of the VSM. Thereby, they discover that system five, "identity", is not considered by the majority of the ap-

proaches. According to Buckl et al. (2009), this system is in the context of EA management responsible for questions of "scope and reach" of the EA management function . Put in other words, system five is concerned with adapting the EA management function to ensure its purposefulness in respect to intrinsic goals as "alignment of business and IT" and the "managed evolution" of the enterprise. Buckl et al. (2009) ascribe this lack of established methods for adapting, i.e., (re-)designing EA management functions to the novelty of the field and to its interdisciplinary nature. Especially, the nature may hamper the development of consistent and accepted "EA management governance"[1], as it had to align potentially conflicting viewpoints on the topic from management and engineering disciplines.

Preparing our subsequent discussions on design theory research for EA management, we revisit the research approach that was pursued in compiling the EAM pattern catalog, which was later, refined to the pattern language. This approach is outlined by Buckl et al. (2008) as a twofold approach. Firstly, open interviews with EA management practitioners were conducted in a number of enterprises in order to elicit their understanding of EA management, and to collect the methods as well as viewpoints that they are using to perform EA management. Secondly, the visualizations and methods gathered during the interviews or found in literature were presented in a questionnaire, where EA management practitioners could rank their purposefulness. This second step was applied to reduce the number of methods and viewpoints that should be documented as EA management patterns in the pattern catalog. Summarizing, the two steps "interview" and "questionnaire" were applied to collect and document relevant EA-related problems and proven practice solutions. In a subsequent activity, different application cases were conducted to validate the usability of the pattern catalog. of which exemplarily the student theses of Dierl (2008) and Pflügler (2008) can be named. In these cases, the EAM pattern language was applied to design or re-design parts of an enterprise-specific EA management function.

Summarizing the above, the EAM pattern language presented by Ernst (2008) can be regarded a valuable contribution to a design theory for EA management functions. It provides structured descriptions for typical components from the solution domain that can be used to address common EA-related problems. In contrast, when it comes to prescriptions on how to design an EA management function, the pattern language stays on a fairly abstract level. The same is true for questions on how to integrate different solution model patterns into a holistic and comprehensive EA management function. In respect to the research method, the research approach taken for compiling the pattern catalog and language, respectively, seem very promising and sufficiently general to be applicable for researching a design theory.

[1] Buckl et al. (2009) coin this term to denote system five in the EA management context.

4 Conclusion

In the preceding section, some shortcomings of the pattern-based approach for EA management are discussed, especially in respect to design prescriptions, i.e., to principles guiding the implementation of the approach in a specific enterprise. In the following, we outline the research field spanned by the shortcomings and sketch a framework to structure the field in manageable, i.e. "researchable", partitions that retain practical relevance. Justifying the subsequent procedure, we resort to the central factors that contribute to the complexity of the field of EA management research discussed in Section 1:

- EA management is a relatively new field with a multi-disciplinary background.
- EA management has a relatively broad management subject, for which various stakeholders rise concerns.
- The effects of EA management do often not manifest immediately, as measures create company benefit mostly in the long run.

Especially, the heavy involvement of the enterprise stakeholders, the broadness of the subject and the delayed effects make EA management a topic that is not easy to research. The creation of a prescriptive theory that helps enterprises to design their specific EA management function remains a comprehensive and embracing research subject. Nevertheless, the three types of EA management patterns presented in Section 3 give rise to a structuring of the subject. In terms of the pattern-based approach, an EA management function can be described by *methods* and *modeling languages*, which are composed of viewpoints and conceptual models.

Of these high level artifacts (solution models in terms of Figure 2), different concretizations exist in respect to different EA-related problems, as e.g. standardization, and in respect to different *areas of interest* in the EA, as e.g. the application landscape. Further, a design theory does not limit itself to solution models as the management methods or the modeling languages, but also encompasses design methods applying on the artifacts and prescriptions on the artifact evolvability. Especially the latter concepts are of major interest, if the design theory should be applied. Consistently, three dimensions to partition the design domain for theorizing remain. These dimensions and the corresponding options for research are:

- **Solution model**: *Method* and *Modeling Language*
- **Problem:** *Standardization*, *Business Continuity*, *Security*, *Risks*, …
- **Area of Interest**: *Application level*, *Business level*, *Infrastructure level*, …

Limiting the areas of interest, e.g. to solely application level artifacts, seems to be the least promising strategy in this respect, as it would lead to a "degenerate" version of EA management that would not account for the intricate relationships between artifacts on different levels. The two other dimensions give rise to more

promising strategies for overcoming the difficulty with the embracing research subject, namely the strategies of *vertical* and *horizontal* domain decomposition:

- *Vertical* domain decomposition means approaching only a limited number of EA-related problems in an embracing manner, i.e. with appropriate management methods and modeling languages, complemented by suitable design methods and evolution guidelines.
- *Horizontal* domain decomposition means approaching a broad variety of EA-related problems either with appropriate management methods or modeling languages, complemented by suitable design methods and evolution guidelines.

Both decomposition strategies have advantages and disadvantages that are sketched below.

The vertical decomposition should more easily create concise theoretical constructs, as a management method and its corresponding modeling language are developed together. This should also be reflected by the design methods and evolution guidelines that should optimally fit the design artifacts. Nevertheless, vertical decomposition might lead to results of limited generality, i.e., to design artifacts, design methods, and design guidelines that repeat itself for different problems over and over again. In addition, the expository instantiation of the design theory is limited to the small set of problems that the theory covers and a researcher may have difficulties in finding an enterprise, where the theory can be put into action.

Contrasting, the horizontal decomposition may fall for a "mismatch" problem between the management methods and the corresponding modeling languages, as they have been developed separately. In addition, the practical applicability of a management method without a complementing modeling language or vice versa may be limited. This may aggravate the expository instantiation of the theory and its validation, respectively. When it comes to the development of these artifacts, a benefit of this decomposition strategy is quickly discovered. The different artifact types are most likely to rely on different kernel theories that can be more extensively taken into account, if a research endeavor is limited to just one type of artifact. Furthermore, being concerned with only modeling languages or management methods may prove helpful in generalizing the artifacts, and in developing more general design methods and design guidelines.

Having outlined the advantages and disadvantages of the two decomposition strategies above, we propose to apply "horizontal decomposition" to develop a design theory for EA management. This yields two research streams; stream one develops a library of management methods together with a set of integration and evolution guidelines. Stream two is concerned with a language toolset for EA management consisting of a meta-language, which addresses the challenges of a configurable method composition, customization principles, and integration guidelines. These two streams should be executed in tight cooperation and – as a joint-venture – should develop a prototypic tool that incorporates theory components from

both streams. This tool should be used to show the applicability of the theory in a practical setting and provide input for validating the outcomes of the research streams.

5 Outlook

In this article, we revisited a pattern-based approach to EA management (cf. Section 2), more precisely to the construction of an organization-specific EA management function. In doing so, we took a design theory perspective and compared the approach to a pattern-based framework for design theories (cf. Section 3) presented by Schermann et al. (2007). Having identified some shortcomings of the approach, Section 4 presented an overview on open research issues and discussed on ways to address these issue in a researchable form. Thereby, the article provided guidelines for future EA management research that may also be helpful in developing a comprehensive design theory for EA management functions. Some questions are still left open by the paper. Quite a few center around the research method that should be employed by the research streams. Such method must accommodate to the intrinsic complexities associated with EA management research as outlined at the beginning of Section 4. Further, the paper did not account for *epistemological* questions behind the research streams. While such questions are doubtlessly highly relevant, the actual choice is also dependent on the research method chosen and could hence not be discussed here in detail.

References

Aier S, Kurpjuweit S, Schmitz O, Schulz J, Thomas A, Winter R (2008) An engineering approach to enterprise architecture design and its application at a financial service provider. In Loos P, Nüttgens M, Turowski K, and Werth D (edts) Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management 27.-28. November 2008 Saarbrücken, 115-130.

Aier S, Riege C, and Winter R (2008) Unternehmensarchitektur – Literaturüberblick und Stand der Praxis. In WIRTSCHAFTSINFORMATIK 50(4):292-302.

Alexander C (1979) The timeless way of building. Oxford University Press, Oxford.

Batini C, Lenzerini M, and Navathe SB (1986) A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, 18(4):323 – 364.

Brinkkemper, JN (1992) Manipulatable Meta Models for Method Engineering, Research proposal, University of Twente.

Buckl S, Ernst AM, Lankes J, Schneider K, and Schweda CM (2007) A pattern-based approach for constructing enterprise architecture management information models. In Oberweis A et al. (edts) Wirtschaftsinformatik 2007, Universitätsverlag Karlsruhe, Karlsruhe, 145-162.

Buckl S, Ernst AM, Lankes J, Matthes F, and Schweda CM (2008) Enterprise architecture management patterns – exemplifying the approach. In The 12th IEEE International EDOC Conference (EDOC 2008), IEEE Society, Munich.

Buckl S, Matthes F, and Schweda CM (2009) A viable system perspective on enterprise architecture management. In Proceedings of the International Conference on Systems, Man, and Cybernetics 2009, San Antonio. (in publication)

Buschman F, Meunier R, Rohnert H, Sommerlad P, and Stal M (1996) Pattern-oriented software architecture: A system of patterns. Wiley, New York.

Dierl T (2008) Models, methods, and visualizations for compliance management. BSc thesis, Technische Universität München.

Ernst AM (2008) Enterprise Architecture Management Patterns. In: PLoP 08: Proceedings of the Pattern Languages of Programs Conference 2008, Nashville, 2008.

Ferstl OK, Sinz EJ (2005) Modeling of business systems using SOM. In Bernus P et al. (edts) Handbook on architectures of information systems, Springer, Berlin.

Frank U (2002) Multi-perspective Enterprise Modeling (MEMO) – conceptual framework and modeling languages. In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS 2002), Washington, 1258-1267.

Gamma E, Helm R, Johnson R, and Vlissides JM (1994) Design Patterns: Elements of reusable object-oriented software. Addison-Wesley.

Gregor S (2006) The Nature of Theory in Information Sysetms. In MIS Quaterly 30(3):611-642.

Gregor S and Jones D (2007) The Anatomy of a Desgin Theory. In Journal of the Association for Information Systems 8(5):312-335.

Köhne S (2005) Didaktischer Ansatz für das Blended Learning: Konzeption und Anwendung von Educational Patterns. PhD thesis, Universität Hohenheim.

Langenberg K and Wegmann A (2004) Enterprise architecture: What aspect is current research targeting? Laboratory of Systemic Modeling, EPFL, Lausanne.

Niemann KN (2006) From enterprise architecture to IT governance – elements of effective IT management. Vieweg, Wiesbaden.

Pflügler K (2008) Evaluation and extension of the EAM pattern catalog in a German insurance company. BSc thesis, Technische Universität München.

Ross J, Weill P, and Robertson DC (2006) Enterprise architecture as strategy. Harvard Business School Press, Boston.

Schekkerman J (2008) Enterprise architecture good practices guide – how to manage the enterprise architecture practice. Trafford, Victoria.

Schermann M, Böhmann T, and Krcmar H (2007) Fostering the evaluation of reference models: Application and extension of the concept of IS design theories. In Oberweis A et al. (edts) eOrganisation: Service-, Prozess-, Marketing Engineering – 8. Internationale Tagung Wirtschaftsinformatik, Band 2, Universitätsverlag Karlsruhe, Karlsruhe, 181-198.

Schönherr M (2008) Towards a common terminology in the discipline of enterprise architecture. In Service-oriented computing – ICSOC 2008 Workshops, Springer, Berlin, 400-413.

Sebis (2009) EAM Pattern Catalog Wiki. Chair for Informatics 19 (sebis), Technische Universität München, München, http://eampc-wiki.systemcartography.info (cited 2009-10-02)

The Open Group (2009) The Open Group Architecture Framework – "Enterprise Edition" Version 9. The Open Group, San Diego. http://www.togaf.org (cited 2009-10-02).

Van der Raad B, Schouten S, and van Vliet H (2008) Stakeholder perception of enterprise architecture. In Morrison R, Balasubramaniam D, and Falkner KE (edts) Software Architecture, Second European Conference, ECSA 2008. Springer, 19-34.

Wagter R, van den Berg M, Luijpers J, van Steenbergen M (2005) Dynamic Enterprise Architecture: How to make IT work. Wiley, New York.

Walls JG, Widmeyer GR, Sawy E, and Omar A (1992) Building an information system design theory for vigilant EIS. In INFORMATION SYSTEM RESEARCH 3(1):36-59.

Wegmann A (2002) The Systemic Enterprise Architecture Methodology (SEAM). EPFL.

Zachmann J (1987) A framework for information systems architecture. IBM Syst. J. 26(3):276-292.