Available online at www.enterprise-systems.net

AIS Transactions on Enterprise Systems 1 (2009) 1, 36-43

Enterprise Architecture Design as an Engineering Discipline

Stephan Aier, Stephan Kurpjuweit, Jan Saat, Robert Winter

Abstract

Enterprise architecture can provide systematic support to organizational change, when requirements of respective stakeholders of business and IT are met. This article focuses on the design of enterprise architecture and proposes a "business-to-IT" approach that considers lessons from classical engineering disciplines. A framework for engineering driven enterprise architecture design is presented. Since such an approach creates specific requirements for tool support, an appropriate software implementation is presented.
Keywords: enterprise architecture, business engineering navigator, tool support

1. Introduction

Organizations are subject to constant evolution. Due to the different impact, organizational change can be distinguished into incremental change (optimization) and fundamental change. While most functional methods of business administration, such as marketing, finance and human resources provide support for optimization (e.g. six sigma) [10], the structured design of innovative and fundamental change requires a holistic approach to systematically support organizational transformation [21]. Complex changes require a thorough understanding and therefore a targeted documentation of the artifacts to be designed, their relationships to each other as well as a clear structuring of the transformation procedure. Therefore, architectural as-is documentation, to-be planning, and support of necessary changes are core challenges for enterprise architecture (EA) analysis and design [13]. To meet these challenges, design objects of EA such as strategic aspects, organizational structure, business processes, software components and data structures as well as IT infrastructure components are modeled to enable communication and analysis of the EA [11, 22]. While there is a broad variety of EA literature focusing on evaluation [18] and generalization [12] of EA frameworks or discussing EA modeling [1], only few publications address EA application and its benefits [14, 17]. In particular an engineering approach is missing which deploys EA to systematically support innovative and fundamental change.

In this contribution we analyze mature engineering disciplines to derive characteristics for a framework to systematically support consistent "business-to-IT" transformation. We propose the business engineering navigator (BEN) concept to support construction, navigation and analysis functionalities for artifacts and relationships of all architectural layers – from strategic aspects down to IT infrastructure. BEN therefore provides a framework on how engineering methods can be applied to organizations. BEN delivers insights on how complex design and transformation challenges can be broken down to manageable projects. We therefore discuss how BEN can be used to systematically support to EA design in this article.

The next section identifies core concepts of mature engineering disciplines. Following lessons learnt from classical engineering, section 3 derives requirements for an engineering based approach to EA. Section 4 introduces the BEN concept to support a stakeholder-oriented EA management (EAM) as one of multiple possible applications. Section 5 discusses a "business-to-IT" EAM tool support and proposes ADOben as an appropriate solution. The findings are summarized, and future research is outlined in section 6.

2. Lessons from Mature Engineering Disciplines

Mary Shaw analyzed the development of classical engineering disciplines [20]. She found that engineering disciplines produce cost efficient solutions for relevant problems by using scientific knowledge in the artifact design process in service to society. These aspects are now further characterized:

1. "Cost efficient solutions": Engineering does not only imply the construction of suitable solutions, but also emphasizes reasonable handling of given resources and conditions.
2. "For relevant problems": The constructed solution addresses practically relevant problems.
3. "By using scientific knowledge": The construction process is comprehensible and traceable based on scientific construction languages, methods, and frameworks so that the solutions will most likely fit the requirements.
4. "In service to society": The engineer acts in a responsible way by providing useful innovations to society and environment.

The following subsections give an idea of addressing these aspects by analyzing classical engineering.

2.1. Engineering Knowledge Patterns

Classical engineering disciplines distinguish between innovative construction and construction routine. Innovative constructions have to address new solutions while construction routine involves reusing existing solution patterns for known problems [23].

Construction routine is the usual design form in classical engineering disciplines, while innovation is rather rare. To make the construction process as efficient as possible, the collection, organization, and conditioning of knowledge is necessary to make this knowledge available to less experienced engineers. All disciplines found appropriate media for this knowledge transfer, e.g. engineering handbooks [2, 6] and tool support for collaborative engineering [15].

2.2. Standardized Construction Plan and Construction Language

Mature engineering disciplines use a high level construction plan (architecture) of the design artifact. This plan depicts the main components and their relationship to each other that is needed in order to achieve the desired behavior. (Some engineering disciplines including civil engineering and software engineering use the "architectural blueprint" or "architectural design" (short architecture) as central construction plan. In the following the term "architecture" is used as synonym for the central construction plan of all engineering disciplines.) All mature engineering disciplines have developed standardized construction languages for architectural description. In mechanical engineering, for example, a dozen standards exist on how to design construction plans [9]. These standards are subject to early stages of mechanical engineering education since they are an essential means of communication.

2.3. Division of Labor

Besides structuring the system to be designed, the construction plan is used to structure the design process: the components of a system are constructed in teams and then assembled in order to become a whole according to the architecture. The division of labor during the construction process is a core feature of classical engineering disciplines, since it is the only way to construct complex systems in large teams.

2.4. Architectural Design

Designing the architecture is the supreme discipline in engineering, which involves the transformation of requirements (problem space) into a high level blueprint of the system to be designed (solution space). Designing the architecture involves fundamental design decisions which have impact on the whole design process. An example can be found in the definition of quality characteristics that the system to be constructed must address (e. g. Which changes to the system can be made easily, which not? What is the system's performance? What is the capacity of the system? How scalable is the system?).

Due to the mentioned responsibilities, great attention is paid to architecture and only experienced and highly qualified engineers are involved in the architectural design. By involving internal and external experts as well as complex analysis frameworks, engineers seek to ensure the quality of the architectural blueprint so that the architecture addresses all the required characteristics of the system to be designed.

3. An Engineering Based Approach to Enterprise Architecture

Following the above introduced characteristics of mature engineering disciplines, requirements for an engineering driven approach to EA can be derived. EA can be regarded as the central

construction plan for organizational transformation in a "business-to-IT" approach. EA describes the main business and IT components as well as their relationships (c.f. "standardized construction plan" in classical engineering). EA is the result of important design decisions and determines fundamental characteristics of the organization, such as strategic positioning, business process efficiency and effectiveness, business/IT alignment, and information systems capabilities. Indirectly, EA therefore implies e.g. an organization's capability to rapidly launch new products, to adapt to new regulations, or to exploit business potentials of IT innovations (c.f. "architectural design" in classical engineering).

Following engineering principles, concrete requirements of internal and external stakeholders build the starting point for EA design. Stakeholders may e.g. contribute model information and also consume information of the EA. As far as designing stakeholders are concerned, conventions (c.f. "standardized construction language") and governance are vital to enable distributed but consistent design (c.f. "division of labor" in classical engineering). Designing EA does not imply to create new models from scratch, but to integrate and aggregate existing knowledge from architectural parts (c.f. "engineering knowledge patterns" in classical engineering). Not all of the stakeholders' concerns and requirements have effects on the fundamental structure of the organization (or EA), but they partially might still have influence as architectural drivers.

There exist different classes of architectural drivers. One class focuses on the functional development of the organization. Examples can be found in the opening of new markets and sales channels or business process outsourcing. Another class of architectural drivers focuses on optimization of organizational structures, e. g. by consolidation of redundant structures or reuse of existing resources to improve flexibility and prepare the organization for possible future changes.

Architectural drivers tend to have tradeoffs which require compromises in the architectural design. Priorities of the architectural drivers are subject to changes which might cause discontinuities in organizational development. A merger, for example, might change any given situation to set the focus on architectural consolidation.

The sketched complexity of the matter often causes difficulties for enterprise architects to choose the appropriate artifacts and relationships for the EA model. From an engineering perspective and taking experiences from EA projects in companies into account, the following heuristics can be derived.

3.1. Criterion of Width

EA models must address the information demand of their stakeholders. Information demands are implied by management tasks (concerns) of the respective stakeholders. EA can for example deliver crucial data for project portfolio management to support decision making, concerning investment decisions for business applications.

A successful method for stakeholder involvement turned out to be the collection and analysis of precise questions that stakeholders have, e. g. "Can investments in applications by justified by additional revenue, gained from the product or service which is supported by this application?" Situational fragments of the EA model (viewpoints) can help to answer such questions by representing the desired information on an aggregate level and in a form of representation which is appropriate for the respective stakeholder.

Following the criterion of width, all artifacts and relationships needed for the creation of view-points must be reflected in EA. The sum of information demands of all stakeholders therefore determines the maximum EA extent.

3.2. Criterion of Depth

When EA is only designed in respect to the criterion of width, chances are high that a huge number of detailed structures of implementations or detailed inventories of single artifacts types are included.

Architecture strategies which are derived from the architectural drivers, and the desired characteristics of the whole system should also be included in EA. These architecture strategies need to be expressed and documented, so that their realization is measurable. Architecture strategies focus on the entire system or on groups of similar artifacts (This heuristic is based on the locality criterion, initially published by [5] and then adapted by [7] This criterion is adapted for enterprise architecture and informally described.) such as all core business processes, all data flows across domains, or all products which are distributed over a certain channel. Structures which only focus on implementation details of one artifact, and which are only relevant for this object, should not be a part of EA. Exceptions might be useful in certain situations, e. g. to support concerns of a key stakeholder.

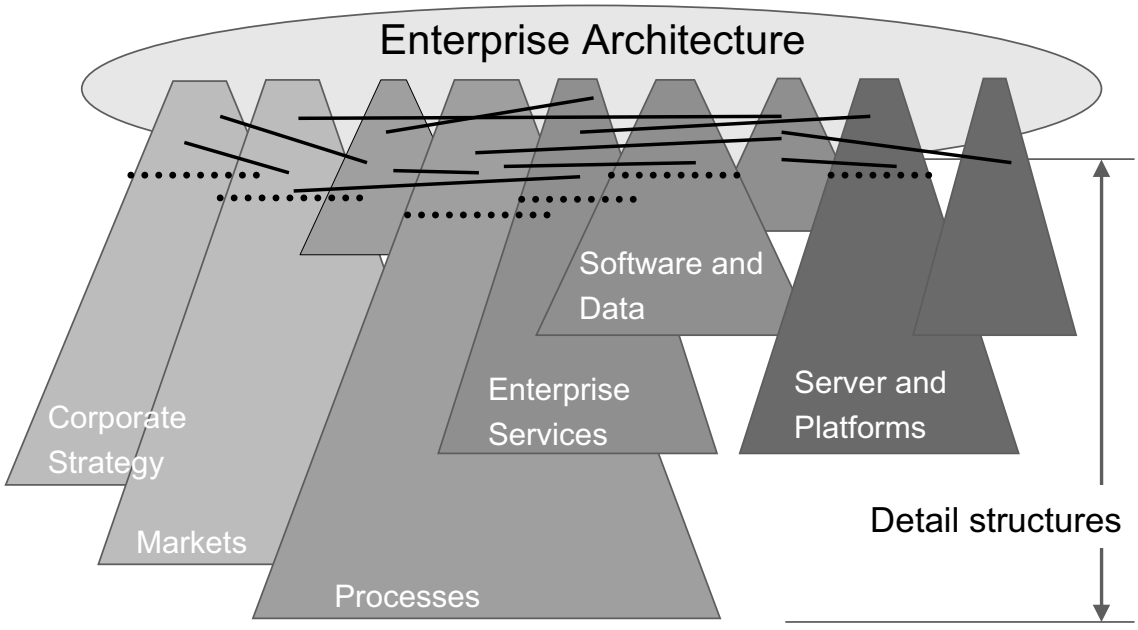
The relevance of an artifact can be indicated by the impact that a change of this artifact has on others (This heuristic is based on encapsulation and information hiding, which originates in object orientation (cf. e.g. [16]).). If a change of an artifact does not influence others at all, it should most likely not be included in EA. Following the idea that EA is the blueprint for change projects, problems can arise from making unnecessary design decisions for the entire architecture which should be better made for individual projects. Therefore, details such as object oriented class structures, detailed data structures, mapping information of network adaptors to servers, structures of teams in individual business units, workflow specifications of business processes, or construction details of products should not be part of EA. Figure 1 illustrates our "broad and aggregate" understanding of EA.

In two cases it can be useful to include detail artifact structures in the EA model. In both cases, changes to the detail structure cause potential changes to other artifacts, which means that the above mentioned heuristic remains valid:

1. Relationships to other detail artifact structures: Examples can be found when deploying single software components on servers or assigning sub-goals to the responsible business units. A relationship on detail level (e. g. application component and server) can always be observed on the respective aggregated level (e. g. respective application and respective server cluster). Detail structures should only be included in EA when they have impact on design decisions with effect on the entire system. This is true for the deployment of application components on servers, since the explicit documentation of this relationship might have considerable impact on the ability of the organization to react in case of a blacked out computing center. An example for a relationship on detailed level without significant impact can be found in the assignment of application functions to detailed activities of a business process. In this case, the aggregate relationship between application and business process delivers sufficient information for EA purposes, while detail documentation can be misleading.
2. Objects on detailed level can be reused in multiple artifacts: Similar to the case above, the detail level should only be taken into account, if reuse has significant impact on the behavior of the entire system. This is the case in examples such as reuse of product components as part of a platform strategy. Contrary, it is not the case when reusing libraries in multiple applications.

Moreover, it cannot be recommended to include many objects of a detail structure which all have similar relationships within the architecture. This

Fig. 1: Enterprise Architecture is Broad and Aggregate



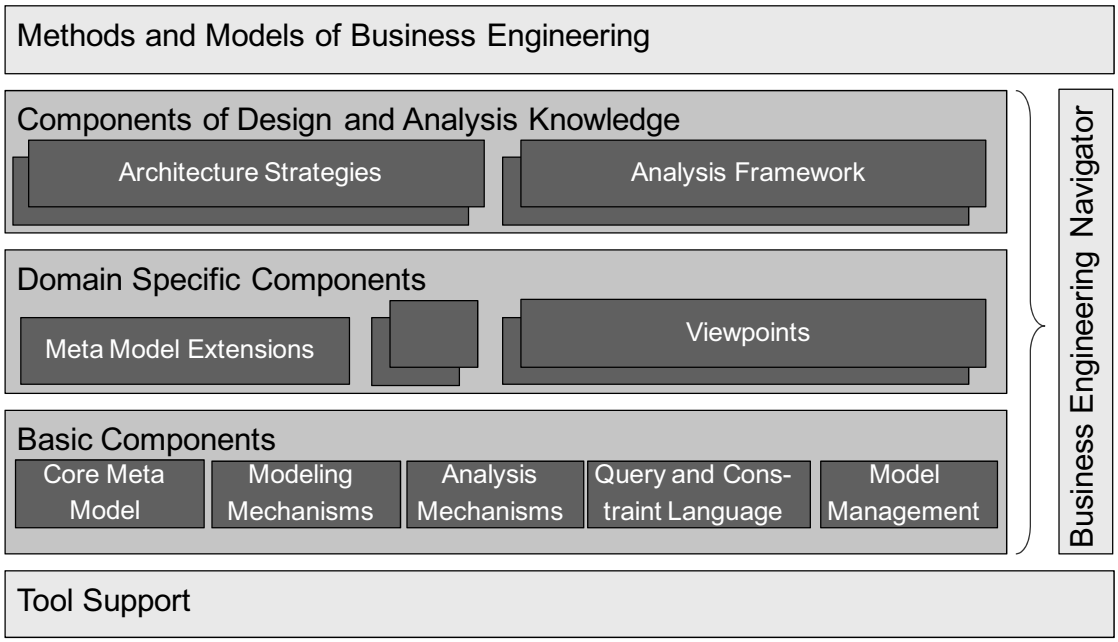


Fig. 2: Architecture of the Business Engineering Navigator Approach

is e.g. the case when considering all client computers (as inventory).

3.3. Pragmatic Criterion

Organizations are subject to constant changes. Therefore EA models need to be updated regularly. Many projects show that continuous maintenance efforts incur high costs. Therefore it needs to be considered if the benefits resulting from covering a stakeholder concern exceed the costs necessary to gather and maintain this information. Not every stakeholder information demand which is claimed by the criterion of width will gain positive revenue. Therefore, the pragmatic criterion proposes to carefully analyze and evaluate the value of artifacts and relationships. No maintenance efforts should be put into artifacts which are not necessary for any concerns [8].

Quantifying costs and benefits of information demand is far from trivial [e.g. 17]: Benefit analysis often results in "reverse" considerations (what if we did not have this information?). Costs arise according to type, origin, necessary conditioning efforts, and frequency of usage. Information demands being served from the same pool of data might realize considerable synergies.

The main feature of the architecture is to provide a high level plan to support long term strategic development of an organization. High frequency in changes of detail information incurs high maintenance costs and can be used as an indicator that the level of aggregation is too low. From our experience, in most cases it is sufficient to use and maintain more aggregate structures (as proposed in the criterion of depth). Usually, high level models can be maintained manually with reasonable efforts, i. e. without having to develop and use automated interfaces to detail repositories (such as configuration management database, process model repository, product configuration system). However, there may be use cases where more detailed model data is needed, automated data imports might be necessary to provide an efficient solution at reasonable maintenance efforts.

4. Business Engineering Navigator

BEN structures the various components of engineering support for EAM. BEN is based on the above mentioned principles of engineering and addresses the main requirements of EAM.

Figure 2 illustrates the components of BEN and their assignment to abstraction layers. This structure can be used as a framework for practical as well as research projects. The components are described in the following subsections.

4.1. Basic Components

Basic components include domain independent functionalities which are used to model, analyze and design EA.

- Core meta model: A common set of vocabulary is a major prerequisite to consistently design the five layers of the business engineering framework. The BEN meta model is based on generic modeling methods and contains artifacts on a strategy layer, organizational layer, integration layer, software layer, and an IT infrastructure layer [22]. This meta model serves as a standardized construction language for organizational transformation.
- Modeling mechanisms: A domain independent description language provides basic mechanisms to create models of the design artifacts. This includes hierarchical refinement of artifacts using "part-of" and "is-in" relationships as well as domain clustering.
- Analysis mechanisms: Generic types of analyses and analysis mechanisms are instantiated for each concrete viewpoint (cf. below). Examples for generic types of analyses include matrix analysis, dependency diagrams, list reports, architecture views, and spider web diagrams [3].
- Query and constraint language: A query language is needed to analyze the models using predefined and ad-hoc queries. Using the constraint language, the architecture strategy and the architectural principles are specified and verified. Both languages are based on formalized modeling mechanisms, e. g. relational algebra.
- Model management: This basic component includes version management functionalities, such variants handling and model history. These aspects are crucial to model life-cycle management

4.2. Domain Specific Components

Domain specific components are instances of generic components for the five different layers listed in section 4.1.

- Meta model extensions: Specific extensions of the core meta model allow the application of the engineering approach in specific contexts (e. g. a certain industry, a certain company size or maturity level) and in specific projects (e. g. business driven changes, IT driven changes, alignment projects).
- Viewpoints: A viewpoint catalogue is comprised of generic analysis mechanisms and types of analyses which are suited to given stakeholder information demands. Queries needed for each viewpoint can be formulated using the above introduced query language [11].

4.3. Components of Design and Analysis Knowledge

Components of design and analysis knowledge help to keep record of the engineers' knowledge.

- Architecture strategies: Generally valid and accepted design patterns and architectural strategies (e.g. handling of redundant master data) and principles can be organized as knowledge repositories [4].
- Analysis framework: An analysis framework implements models of quality and metrics for the design artifacts (e.g. analysis frameworks which help to refine aggregate targets, such as efficiency, into measurable counts, such as scalability, avoidance of redundancies, capability for multi channel usage [19]). Results of the analysis are represented as viewpoints.

The BEN approach proposes to adapt EAM to the respective application scenarios of the respective organization. Therefore, generally valid and accepted components of design and analysis knowledge must be adapted, extended and integrated.

The BEN approach can be understood as interface between methods of business engineering and underlying software tools: On one hand, BEN defines requirements for software systems and gives assistance how to use them in the context of the engineering discipline. On the other hand, BEN is a service layer for different methods, which may give concrete guidance in change and transformation for organizations.

5. Tool Support: A BEN Implementation for Documentation, Analysis and Design of Enterprise Architecture and Learnings from First Applications

Regarding the criterion of width, EA addresses a variety of stakeholders with different information demands and different views on EA. Therefore the implementation of the basic components of BEN (cf. section 4.1) requires a specific tool support where BEN can serve as a foundation for the implementation or configuration of EA software tools. ADOben is such an implementation of BEN requirements based on ADONIS, a commercial modeling tool and meta-modeling platform.

ADOben implements the required model types from a strategy layer down to an IT infrastructure layer as well as the interdependencies between the artifacts and models on these layers. Therefore it is possible to design an architecture plan for the as-is situation. Using means of architecture analysis and a dedicated architecture strategy, a blueprint for the to-be situation can be designed.

To support the application scenarios of potential EA stakeholders, the tool implements

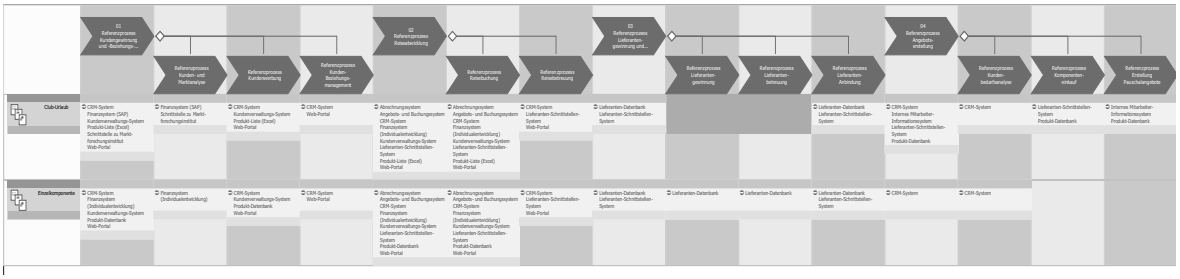


Fig. 3: Three Dimensional Matrix Report in ADOben

the respective queries and visualizes their results. The following example illustrates an application scenario in which a business analyst plans the launch of a new product. Information demands of the business analyst could be: "Do we have adequate application support for the new product?", "Where are potential breaks between applications along the process?" Using the query "Which applications are used in which process for which product?" on the architecture model, a matrix report in three dimensions as shown in Figure 3 is created. The matrix shows the products and processes as well as the underlying applications.

Based on a generic core meta model and generic analysis mechanisms as well as specific extensions for a defined application scenario, every other query could be run on the underlying models and visualized in a report.

Since BEN is not particularly developed for EAM, the generic concepts (as presented in section 4) could also be implemented in different tools and for other business engineering methods. As a first means of feasibility evaluation the BEN approach has been implemented in a German financial service provider using ADOben. The application of the approach verified that EA should be positioned as a planning tool, not as a tool focused on operative tasks (like for example a configurations management database system triggering an alarm when a server hard disk fails). To achieve this, the three criteria defining EA scope have proven to be valuable. The criterion of width requires that the EA meta model and the viewpoints are developed in close collaboration with all stakeholders of the EA. To get the buy-in of the stakeholders, the introduction of EAM should be taken as a chance to revise the planning and documentation processes within the organization in order to ensure that the EAM organization concept is integrated seamlessly and does not cause an overhead work load for the stakeholders. The analysis capabilities of ADOben, especially matrix analyses have turned out to be a valuable tool to foster and rationalize the communication between the IT unit and the business units as well as to systematically address alignment questions between business structures and IT structures.

6. Conclusion

Based on analysis of classical engineering disciplines, this paper presents an engineering approach to EAM which has been generalized as BEN. It is shown how EA models can be constructed based on stakeholder requirements in order to create a pragmatic solution representing a "broad and aggregate", business-to-IT architecture – and not a set of enterprise-wide detail models which will never be completed and soon be outdated. BEN delivers a foundation for efficient EA design and EAM. BEN can be implemented in software tools and applied using business engineering methods to enable structured solution design.

Engineering disciplines in general, BEN and ADOben show that the engineering of complex environments involves a complex 'mechanism'. This mechanism can be evaluated according to its applicability and to its connectivity to other approaches, tools, and methods. The development of this mechanism is aimed at a clear structure so that elements can be arranged according to the respective situation as a best-of-breed solution. This means that ADOben is one solution to implement BEN as an EAM tool. At the same time BEN is not limited for the use in the context of EAM. The core idea is to ensure structured engineering. Further research activities in this area will focus on the methods themselves and their situational character. The ultimate goal is to provide engineering support for the situational development and maintenance of "business-to-IT" solutions – in the context of EAM, but also for integration management, for information

logistics management, for IT/business alignment and other scenarios in information management.

References

[1] Arbab, F., de Boer, F., Bonsangue, M., Lankhorst, M., Proper, E., & van der Torre, L.: (2007). Integrating Architectural Models. Symbolic, Semantic and Subjective Models in Enterprise Architecture, Enterprise Modelling And Information System Architectures, 2(1), 40-56.

[2] Avallone, E.A., Baumeister, T., & Sadegh, A.: (2007). Marks' Standard Handbook For Mechanical Engineers. Mcgraw-Hill Professional.

[3] Bucher, T., Fischer, R., Kurpjuweit, S., & Winter, R.: (2006). Analysis and Application Scenarios of Enterprise Architecture - An Exploratory Study, EDOC Workshop on Trends in Enterprise Architecture Research (TEAR 2006), Tenth IEEE International EDOC Conference (EDOC 2006). Hong Kong: IEEE Computer Society.

[4] Buckl, S., Ernst, A.M., Lankes, J., & Matthes, F.: (2008). Enterprise Architecture Management Pattern Catalog.

[5] DeRemer, F., & Kron, H.H.: (1976). Programming-in-the-large versus programming-in-the-small, IEEE Transactions on Software Engineering, 2(2), 80-86.

[6] Dubbel, H., Kuttner, K.H., & Beitz, W.: (1994). Dubbel. Handbook of Mechanical Engineering. Berlin: Springer.

[7] Eden, A.H., & Kazman, R.: (2003). Architecture, Design, Implementation, International Conference on Software Engineering (ICSE). Portland, OR:

[8] Fischer, R., Aier, S., & Winter, R.: (2007). A Federated Approach to Enterprise Architecture Model Maintenance, Enterprise Modelling and Information Systems Architectures, 2(2), 14-22.

[9] Giesecke, F.E., Mitchell, A., Spencer, H.C., Hill, I.L., Dygdon, J.T., Novak, J.E., & Lockhart, S.D.: (2008). Technical Drawing. Denver, CO: Pearson Education.

[10] Harry, M.J.: (1988). The Nature of Six Sigma Quality. Rolling Meadows, IL: Motorola University Press.

[11] IEEE: (2000). IEEE Recommended Practice for Architectural Description of Software Intensive Systems (IEEE Std 14712000). New York: IEEE Computer Society.

[12] IFIP-IFAC Task Force on Architectures for Enterprise Integration: (2003). GERAM - The Generalised Enterprise Reference Architecture and Methodology. In Bernus, P., Nemes, L., Schmidt, G. (Eds.), Handbook on Enterprise Architecture. (pp. 22-62). Berlin et al.: Springer.

[13] Johnson, P., & Ekstedt, M.: (2007). Enterprise Architecture - Models and Analyses for Information Systems Decision Making. Pozkal: Studentlitteratur.

[14] Kurpjuweit, S., & Winter, R.: (2007). Viewpoint-based Meta Model Engineering, Enterprise Modelling and Information Systems Architectures - Concepts and Applications, Proceedings of the 2nd Int'l Workshop EMISA 2007. Bonn: Gesellschaft für Informatik, Köllen.

[15] McGuire, J.G., Kuokka, D.R., Weber, J.C., Tenenbaum, J.M., Gruber, T.R., & Olsen, G.R.: (1993). SHADE: Technology for Knowledge-based Collaborative Engineering, Concurrent Engineering, 1(3), 137-146.

[16] Parnas, D.L.: (1972). On the criteria to be used in decomposing systems into modules, Communications of the ACM, 15(12), 1053-1058.

[17] Schekkerman, J.: (2005). The Economic Benefits of Enterprise Architecture: How to Quantify and Manage the Economic Value of Enterprise Architecture. Victoria, British Columbia: Trafford Publishing.

[18] Schekkerman, J.: (2004). How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. Victoria, British Columbia: Trafford Publishing.

[19] Schelp, J., & Stutz, M.: (2007). A Balanced Scorecard Approach to Measure the Value of Enterprise Architecture, Journal of Enterprise Architecture, 3(4), 8-14.

[20] Shaw, M.: (1990). Prospects for an Engineering Discipline of Software, IEEE Software, 7(6), 15-24.

[21] Winter, R.: (2008). Business Engineering - Betriebswirtschaftliche Konstruktionslehre und ihre Anwendung in der Informationslogistik. In Dinter, B., Winter, R. (Eds.), Integrierte Informationslogistik. (pp. 17-38). Berlin, Heidelberg: Springer.

[22] Winter, R., & Fischer, R.: (2007). Essential Layers, Artifacts, and Dependencies of Enterprise Architecture, Journal of Enterprise Architecture, 3(2), 7-18.

[23] Zwicky, F.: (1948). Morphological Astronomy, The Observatory, 68(845), 121-143.

Stephan Aier
Institute of Information Management,
University of St. Gallen,
St. Gallen, Switzerland,
E-Mail: stephan.aier@unisg.ch,
Phone: +41 71 224 3360

Stephan Kurpjuweit
Institute of Information Management,
University of St. Gallen,
St. Gallen, Switzerland,
E-Mail: stephan.kurpjuweit@unisg.ch,
Phone: +41 71 224 3316

Jan Saat
Institute of Information Management,
University of St. Gallen,
St. Gallen, Switzerland,
E-Mail: jan.saat@unisg.ch,
Phone: +41 71 224 3782

Robert Winter
Institute of Information Management,
University of St. Gallen,
St. Gallen, Switzerland,
E-Mail: robert.winter@unisg.ch,
Phone: +41 71 224 2190