Towards an Engineering-Based Research Approach for Enterprise Architecture: Lessons Learned from Normalized Systems Theory

Philip Huysmans and Jan Verelst

Normalized Systems Institute, University of Antwerp, Antwerp, Belgium
{philip.huysmans,jan.verelst}@ua.ac.be

Abstract. The emerging field of enterprise engineering provides a promising outlook for positioning relevant research. Enterprise Architecture frameworks which are frequently used in practice, but are often criticized from a research perspective, can be positioned in this field. The challenge for the enterprise engineering field is to provide a framework to improve such frameworks using a rigorous scientific approach. This paper aims to contribute to addressing this challenge by proposing components for a research framework which focuses on applying engineering insights to enterprise architecture. It first explores how current enterprise architecture frameworks handle issues relevant for engineering (i.e., complexity, change and integration). It then introduces additional components which could contribute towards a more systematic approach. These components are derived from the way the Normalized Systems Theory was developed, and successfully introduced engineering standards into the design software architecture.

Keywords: Enterprise Architecture, Enterprise Engineering, Normalized Systems.

1 Introduction

Enterprise engineering is an emerging field which can benefit from many research approaches. One approach is to consider frameworks, methods and analysis techniques in the context of traditional engineering sciences. While certain issues faced in an organizational context will not be addressed by this approach, it limits the scope of issues to a more systematic analysis. Examples of typical issues which are addressed by engineering sciences are complexity, change and integration. These issues are very relevant for enterprises as well.

Integration is an often-recurring theme in many organizational research domains. It refers to the need of an organization to operate as a unified whole, which can be guided in a strategic direction (cf. management research) or is recognized as a distinct brand by customers (cf. marketing research). Within the information systems research domain, the integration issue becomes most apparent in Business/IT alignment research [1]. Even without considering supporting IT functions, organizations need to be able to integrate their product portfolio,

X. Franch and P. Soffer (Eds.): CAiSE 2013 Workshops, LNBIP 148, pp. 58–72, 2013.

the production processes used to create these products, and the organizational structures which execute these processes. These different components are generally considered to increase in *complexity* over time. First, more sophisticated products are needed to avoid a competition based solely on price, while leads for example to the servitization of products (i.e., bundling products and services) [2]. Second, the demand for increased quality (e.g., TQM) and responsiveness (e.g., JIT) requires more complex production processes. Third, globalization and frequent mergers and acquisitions often result in more complex organizational structures, or require a complex reorganization to obtain a simpler structure. Fourth, IT systems has been described to become increasingly complex, as described by Lehman's laws [3]. Additionally, organizations need to be able to respond quickly to *changes* in their environment in order to maintain a competitive advantage [4,5]. Moreover, changes cannot be considered to be individual or isolated. Rather, it is claimed that different changes need to be applied at a steady pace in order to adequately react to contemporary markets [4,5]. This means that it is difficult to describe a stable state of an enterprise to use as a starting point for change, and that different change projects can interact with each other.

Within the enterprise engineering field, enterprise architecture frameworks can be considered to contain the most advanced knowledge to address these issues. The ANSI/IEEE 1471-2000 standard defines architecture as "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution" [6]. This definition distinguishes between descriptive and prescriptive perspectives on enterprise architectures. In a descriptive perspective, the representation of the structure of elements and the relationship between them is the main contribution of an architecture. This perspective is usually associated with a blueprint of an organization as a system. In a prescriptive perspective, the design of organizational artifacts is guided by specifying principles or regulations which limit the design freedom for individual artifacts. By adhering to these principles, a well integrated system will be designed. Various enterprise architecture frameworks are explicitly descriptive (e.g., "enterprise architecture is a set of descriptive representations of an enterprise" [7]) or prescriptive (e.g., "enterprise architecture is a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems and infrastructure." [8]).

Both types of enterprise architecture framework propose to deal with the identified issues of complexity, change and integration in different ways. Certain authors claim that, because of the radical differences between two types of approaches, they are complementary instead of comparable (e.g., [9]). In this paper, we explore how current approaches deal with the issues of integration, complexity and change, and provide an outlook on future research directions to works towards a more deterministic approach. Notwithstanding their differences, similarities between enterprise architecture frameworks can be observed. Therefore, we first explore some general characteristics of enterprise architecture

frameworks in Section 2. Next, we describe more in-depth how the identified issues are addressed in two mainstream enterprise architecture frameworks in Section 3, in order to describe the current state of the field more in-depth. We then discuss in Section 4 how current research initiatives attempt to advance the enterprise engineering field by explicitly applying engineering concepts to software, process and organizational domains. Following this discussion, we propose research directions which are more closely related to a traditional engineering approach to address current research gaps in Section 5.

2 How Complexity, Integration and Change Are Handled in Enterprise Architectures

2.1 Reducing Complexity by Applying Abstraction

Enterprise Architectures comprise a large amount of concepts, entities and variables, which cannot be retained or managed at once. Frameworks aim to reduce complexity by creating abstractions from real-world artifacts by creating models [10]. According to Bernus et al. [8], complexity can be measured using a function of the number of elements and relations in a system. Following this definition, complexity can indeed be reduced by omitting classes of elements and relations. As a result, smaller and simpler models can be created, which focus explicitly on a certain aspect and neglect other aspects. Various abstraction mechanisms, such as perspectives, viewpoints, areas of concern, dimensions, etc. are used to limit the scope of a certain class of models or principles [11]. This approach can aid understandability and enable easier communication between stakeholders in the organization, which is indeed considered to be an important advantage of descriptive enterprise architecture frameworks. However, the application or aspiration of enterprise architecture frameworks is not limited to communication. Enterprise architecture frameworks (especially prescriptive ones) are applied to *design* organizations and supporting systems as well [8,12]. During design, certain aspects of a system cannot be abstracted without considering their impact. Certain authors therefore argue that enterprise architecture design principles should be augmented with standards and guidelines [13]. In this way, each architectural layer can be designed by adhering to a specific set of guidelines.

2.2 Achieving Integration

Enterprise architecture frameworks originated based on the need for integration between the strategic orientation of an organization and its concrete operation. The identification of the need for such an integrated view can be traced back to organizational theory [14]. It was observed that the formal structures, goal orientations and time orientations between sales, research, and production divisions in large industrial organizations were very different from each other. A higher degree of differentiation between divisions was found to be inversely related to the degree of integration of the organization. While the differentiation

of the divisions was explained as a reaction to the variety of requirements from their environment, the effective performance of the organization requires the integration of the outcomes of every division [14]. Various authors have identified organization-wide communication and decision-making as crucial preconditions for this integration [15,16]. Consequently, any organization-wide approach such as the specification of a strategic plan or an architecture can be considered as a contribution to integration and alignment [17].

However, when integration is considered to be more specific (as in: integration between architectural layers), contributions are less frequent. Most frameworks focus on the specification of the different layers which need to be defined, and modeling of the separate layers. As discussed above, this is mainly aimed at the reduction of complexity. Less focus is on the integration between layers. Most publications on enterprise architecture research therefore report on contributions which can be located on a single layer, while few authors address integrating multiple layers [18]. The capability to deal with the relations between model elements on various levels of the enterprise architecture is also referred to as traceability. Wegmann considers this traceability to be essential for enterprise architecture, as it makes the integration between different levels explicit. However, he acknowledges that, while enterprise architecture frameworks should be created to enable this traceability, it is difficult to clearly establish and maintain between the levels [19].

2.3 Dealing with Change

A specific distinction between the handling of change can be observed between descriptive and prescriptive enterprise architecture frameworks. Descriptive frameworks consider an organization as a static system, at a given moment in time. The models created for this static system are referred to as as-is models. When changes need to be introduced, a new version of the framework (i.e., a to-be version) is created, which specifies a future static state. Through gap analysis, change projects can be defined which guide the organization from its as-is to to-be state. However, a method to implement these changes is often out of scope for these frameworks [7].

Prescriptive frameworks, on the other hand, claim to focus more explicitly on providing guidance for change. According to the ANSI/IEEE STD 1470-2000 definition, principles should be defined to govern both the design and *evolution* of enterprise architecture. Other authors consider the very nature of architecture (and hence, principles) to be the *limitation of design freedom* [20]. However, many of the frameworks suggest to create company-specific principles, without providing a way to evaluate the applicability of such principles in other contexts. This limits the generalizability of these principles, and obstructs their systematic application. Nevertheless, this is exactly what is expected from a mature enterprise engineering field.

3 Exploring Specific Enterprise Architecture Frameworks

In this section, we explore whether the Zachman and TOGAF enterprise architecture frameworks provide systematic guidelines to deal with complexity, integration and change.

3.1 Zachman

The Zachman framework [21,7] is an enterprise architecture framework which is often referenced amongst practitioners and researchers. According to several authors, the Zachman framework has the widest adoption in enterprise architecture projects [22,23]. Moreover, it is often used as a basis for evaluating, establishing and customizing other frameworks [23]. Some other frameworks literally position themselves as extensions based on the Zachman framework (e.g., FEAF [24]). Therefore, it is important to understand how the Zachman framework is originally developed. Otherwise, assumptions can be made based on which incorrect solutions for the identified enterprise architecture issues can be suggested. The Zachman framework deals with change and complexity as described in Section 2 for descriptive frameworks: it uses abstraction in different perspectives and dimensions to reduce complexity, and proposes to develop different versions of the framework to deal with changes. In order to achieve integration between models in different cells, different mechanisms are described: constraints, dependencies, and model transformations.

Constraints: The models from each perspective (i.e., each row) have a different set of constraints they need to adhere to [7]. For example, the models in the scope row are subjected to usability constraints (e.g., utility of the artifact), while models in the technology row are subjected to constraints from the state of the art of the used implementation platform. These constraints are additive across the different perspectives: constraints of a lower row also limit the models from higher rows. For example, the technological constraints on the technology models (e.g., only webservices are allowed) will also impact the system model, which will need to structure the system using services. When a constraint in a lower row is inconsistent with a model defined in a higher row, "the designers who are responsible for the two rows must initiate a dialog to determine what must be changed and to ensure that no gap in expectations exists between the different perspectives" [7]. Consequently, it is argued that the issue of conflicting constraints can be handled by communication.

Dependencies: It is acknowledged that the different cells describe abstraction of the same underlying organization, and that dependencies between the cells have an impact. At a minimum, the cells are considered to be related to every other cell in the same row [7]. If a change in the structure of one cell affects the structure of another cell, a dependency between these cells exists. Moreover, such dependencies can occur not only within a row, but also between rows.

The framework however does not provide guidance to determining the possible impacts between models from different cells. It is however acknowledged that the challenge during designing a model "is to design each while understanding the integrity of all others to avoid being surprised by undesirable side effects appearing long after it is possible to contain them" [7, p.595], and that understanding and storing the dependencies would "constitute a very powerful capability for understanding the total impact of a change" [7, p.603].

Model Transformations: While the idea of building IT systems using the Zachman framework is mainly claimed by other authors, Sowa and Zachman already mention the idea of *model transformations* [7]. For example, Pereira and Sousa elaborate on the notion of model transformations by specifying which models are used as input for certain cells [25].

While these mechanisms acknowledge the need for integration, they lack concrete guidelines to achieve such integration. Rather, they either suggest ad-hoc solutions (i.e., handling constraints through communication), admit that no solutions exist (cf. dependencies), or provide no detailed discussion (cf. model transformation). Consequently, the Zachman framework can be considered as a purely descriptive framework, which does not aim to provide prescriptive guidelines.

3.2 TOGAF

Contrary to the Zachman framework, TOGAF [12] explicitly proposes to define principles to guide the development of different architectures. However, these principles are considered to be organization-specific. The provided principles are considered to be examples, which makes them less relevant to evaluate in a broader context. TOGAF is based on four architecture domains: business architecture, applications architecture, data architecture and technical architecture. Instead of focusing on the end products of the architecture, TOGAF focuses on the process to develop the different architectures. In order to develop these architectures, TOGAF suggests an Architecture Development Method (ADM). The ADM is usually considered to be the most important component in TOGAF. Therefore, TOGAF is generally considered to be a process-oriented framework, instead of a product-oriented framework such as Zachman.

TOGAF does specify the "architectural input" which are required for the different phases in the ADM. However, given the absence of a clear approach for integrating different architectures, no concrete way of working can be described. For example, applications and data need to be mapped to physical technological artifacts in phase D. A detailed step in the method description is then: "12.4.6: Resolve Impacts Across the Architecture Landscape" [12]. However, no further details are provided on how to resolve these impacts. Therefore, it is not clear how, for example, conflicting principles need to be resolved or dealt with.

TOGAF explicitly aims to be a starting point for developing an enterprise architecture, but needs to be extended with other methods. For example, it

proposes to use Service Oriented Architecture-related methods to design the IT architecture. Other researchers have reported on research efforts towards this goal as well [26]. Nevertheless, TOGAF itself provides little guidance for the actual design of enterprise architecture artifacts.

4 Suggestions for Prescriptive EA Components

A current lack in enterprise architecture frameworks is the availability of prescriptive guidelines for designing organizational artifacts. Descriptive frameworks do not focus on providing prescriptive guidelines, and prescriptive frameworks either suggest organization-specific guidelines, or focus on defining a process to design architectures, without elaborating on the actual designed artifacts. In order to address this lack, we propose several components in this section which are based on the research initiatives related to Normalized Systems [27,28,29]. The theory on Normalized Systems implies a highly structured approach to software architectures, based on fine-grained modular structures and the systematic reuse of building blocks called elements. The resulting software architectures exhibit behavior that we consider quite different from less structured approaches. More specifically, the result can be considered more *deterministic*. Determinism is related to evolvability (i.e., impacts are easily identifiable as all applications share the same fine-grained modular structure), reuse (i.e., all applications consist of instantiations of the same elements), correctness (i.e., because all applications are based on the same elements, every bug only needs to be corrected once), and reliability and performance (i.e., the investment optimizing an element is economically feasible due to the systematic reuse of the elements). Consider for example how application development is made more deterministic: first, it requires the elements to be built in a certain technology environment, which requires highly advanced expertise in design of software architecture. Next, however, is a phase that is of remarkably lower complexity or expertise levels: building applications. It suffices to identify the required instances of elements, and performing the required manual coding. There is no architectural design phase in building these applications, as the design is already incorporated in the elements. This makes the development process quite similar for all applications, leading to increasing reuse of knowledge about the elements, applications and development process. In this way, Normalized Systems Theory has contributed to resolving wicked issues in software engineering regarding complexity, change and integration. As such, it can be considered a useful candidate to explore for dealing with these issues in the enterprise architecture field.

The components presented here had an essential role in developing the Normalized Systems theory, but are not only applicable to software architecture. They have emerged based on more generally applicable engineering knowledge [27], not on specific software architecture knowledge. Moreover, they have already been applied to the fields of business process design [30] and enterprise architecture [31]. Therefore, we suggest the use of these components as a basis for a research framework for enterprise engineering.

4.1 Modularity Guidelines

Even though originating from systems theory the modularity concept has caught the attention of engineers, managers and researchers in a large variety of fields [32]. Modularity is defined as a property of a complex system, whereby the system is decomposed into several subsystems (i.e., modules). Obviously, each of these modules ultimately must cooperate with other modules in order to ensure the adequate functionality of the system as a whole. The interaction of a module with its external environment should be exhaustively and unambiguously documented in its interface. The interface describes the inputs required by the module to perform its part of the functionality, and the output it will provide to its external environment. As soon as such an interface is designed, one may learn about the intermodular dependencies, i.e., what does a module require from the other modules to perform its own functionality and what is the impact of a change in the module design for other modules.

From a modularity perspective good modular design is characterized by: (a) low intermodular coupling (i.e., few intermodular dependencies), and (b) high intramodular cohesion (i.e., strongly related and dependent elements within a module). From a practical point of view good modular design implies that: (a) changes in the design of one module have no or only a limited impact on the design of other modules, and (b) the function of one module can be studied in more or less isolation from the rest of the system. Consequently, a well-designed modular system enhances the comprehensibility and decreases the complexity of the overall system.

Enterprise architectures can be considered to be modular structures [33,34,10]. On each architectural layer, artefacts are defined which provide functionality required on higher layers, and use functionality from lower layers. Their prescriptive design should then be guided by modularity principles. Modularity has already been proposed by several authors to guide enterprise architectural design [33,34]. Unfortunately, good modular design is far from trivial because many architectural decisions have to be taken. Therefore, more in-depth guidelines are needed which demonstrate how modularity guidelines should be applied in an enterprise architecture context. Hence, a theory which prescribes principles to guide the design of a good modular design is highly desirable. For instance, Normalized Systems theory relies on modularity principles to define more specific software-related principles for achieving an evolvable software structure.

4.2 Preventing Combinatorial Effects

When more insight is gained in the modular structure of enterprise architecture, more specific concepts which build on modularity can be introduced. For example, *combinatorial effects* have been identified as the main obstacle for achieving evolvability in software architectures in Normalized Systems Theory. A combinatorial effect occurs when the effort required to apply a certain change increases with the size of the system. While combinatorial effects have initially been identified at the software level, subsequent research has identified them at

the business process level and enterprise architecture level as well [30,31]. At the business process level, the modular structure has been described as consisting of tasks and their compositions (i.e., process flows). Combinatorial effects have been identified at the level of both the task and process flow levels, and a set of guidelines has been formulated to prevent such combinatorial effects [30]. Consequently, these guidelines can be evaluated and elaborated upon. The resulting set of guidelines can be used to deterministically design business processes without combinatorial effects in any context. Similar to Normalized Systems Theory on the software level, the design can be called deterministic since it needs to adhere to guidelines which have been shown to introduce combinatorial effects when violated, without any assumptions regarding the specific organizational context in which they are applied. At the enterprise architecture level, modular structures have been identified as well, and combinatorial effects have been described [31]. These combinatorial effects have been grouped in two categories. A first category describes so-called *horizontal* combinatorial effects, which affect artifacts within a single enterprise architecture layer. Combinatorial effects at the software level or business process level are examples of such horizontal combinatorial effects. A second category describes *vertical* combinatorial effects. Vertical combinatorial effects can be caused on any layer, but have an impact on other layers. For example, design decisions at the application level have been observed to impact the organizational level [31]. Such combinatorial effect show why the abstraction in current prescriptive enterprise architectures frameworks may be insufficient. As discussed above, prescriptive enterprise architecture frameworks apply abstraction by defining principles on different layers. This implies that issues on each layer can be addressed by principles on that layer. This approach cannot adequately deal with vertical combinatorial effects. As a result, complementary efforts are required.

4.3 Functional—Constructive Gaps

General Systems Theory (GST) studies the general behavior and characteristics of systems. The functional and constructional perspectives on a system are fundamentally different conceptualizations of a system [35]. The functional perspective is concerned with the external behavior of the system [36]. This perspective is adequate for the purpose of using or controlling a system. Consequently, the actual construction of the system is not relevant. Instead, the focus is on how this system interacts with its environment. Therefore, knowledge of the required input variables, transfer function and output variables are key components of this perspective. The input and output variables represent which interactions occur with the environment. The transfer function describes how the input variables are transferred into output variables. This transfer function can be adjusted by control variables, which can alter the behavior of the system. This perspective uses so-called black box models to represent a system.

In contrast, the constructional perspective describes what a system really is [37,38]. In this perspective, knowledge about the composition (i.e., which components constitute the system) and structure (i.e., how these components are

related) is important. Consequently, a fundamentally different kind of model is needed to represent this perspective. This type of model is called a white box model. Such models represent the construction and operation of a system. Analogously to functional decomposition, constructional decomposition can be applied to study the subsystems of a complex system.

Both perspectives need to be integrated: a functional requirement needs to be brought about by a constructional design. However, this relation is not straightforward. Therefore, a so-called *functional—constructive gap* exists, which needs to be bridged by a certain design. Prescribing such designs has proven to be extremely difficult. Nevertheless, further insight in such approaches is valuable, since an enterprise architecture could be considered as a series of functional—constructive gaps. On each layer of an enterprise architecture, different models are created in both a functional and constructive perspective. Insight in how models from both perspectives can be related to each would provide a better way to approach the integration issue in enterprise architectures. Normalized Systems theory has previously been described as a transformation from elementary functional requirements to a constructive software design [28].

4.4 Developing Domain-Specific Patterns

Certain authors argue that a domain-specific approach is required for the specification of prescriptive guidelines, because substantial differences exist between various domains. Such principles can be published in the form of patterns, which describe generic solution for a class of domain problems. This can be considered to be a middle ground between the stance that enterprise architecture principles can only be organization-specific, as discussed above, and that generally applicable principles exist, but have not yet been specified sufficiently. A domain-specific approach is increasingly feasible because business processes are currently being standardized in several sectors. Previously, organizations have developed their own business processes for a very diverse range of domains, including human resources, accounting, finance, order management, logistics and production. The software supporting these business processes was therefore also relatively different between organizations. Many custom-built systems were produced, and software packages needed extensive customizations to make them fit for the organization's processes. These idiosyncratic business processes made it expensive to develop sufficiently generic and flexible building blocks at the level of granularity of services because the cost could not be spread over multiple organizations and/or projects. It has been noted that "domain analysis is invariably conducted within one organization so that transfer of components between domains is difficult" [39]. However, recently a number of indications that organizations are starting to standardize their processes are occurring. Such indications are, amongst others, (a) increased adoption of enterprise software packages such as ERP systems; (b) initiatives to develop reference models of business processes; and (c) standards for communicating business information (ontologies). Domain-specific patterns could make a thorough, engineering-based evaluation

feasible, since they are concrete enough to show design alternatives, and abstract enough to be applied in different organizational contexts.

5 Towards an Enterprise Engineering Research Framework

In this paper, we argued how typical engineering problems such as integration, change and complexity are relevant to organizations as well. Within the field of enterprise engineering, we explored how enterprise architecture frameworks suggest to deal with these issues. We differentiated between descriptive and prescriptive frameworks, and their different approaches to these issues. The current state of the art suggests that many descriptive product-based frameworks are available, together with several prescriptive process-based frameworks. While these frameworks can certainly aid the design of enterprise architectures in practice, they are insufficient for the scientific development of the enterprise engineering field. Instead, an increased focus on prescriptive, product-based components is required.

In the previous section, we discussed several components which could be constituents of a research framework for developing such prescriptive, product-based guidelines. Such a framework would consider an enterprise architecture as a series of functional—constructive gaps, of which each gap needs to be addressed adequately. In order to evaluate the transformation from a functional to a constructive perspective, a theory-based evaluation needs to be conducted. Combinatorial effects, which are based on stability and entropy, are proposed as a candidate for such an evaluation criteria. On the software level, the elimination of combinatorial effects has resulted in software architectures which are evolvable. It is important to note that the adequate identification of combinatorial effects needs to be done on the lowest, most detailed level of software design. Compare this to the tendency of enterprise architecture frameworks to promote high-level overview models. Such models do not contain enough detail to judge whether certain modularity principles are violated. This implies that such high-level overviews may be highly useful in attempts to comprehend an enterprise, but they are not sufficient in order to design an enterprise without combinatorial effects. Nevertheless, many of these enterprise architecture frameworks do make claims that they improve evolvability. This may still be true for example because comprehension is improved, but not from the point of view of a deterministic way to design architectures.

The identification of combinatorial effects requires an explicit statement of how the modular structure of the level which is addressed. Describing the structure of a certain domain requires a constructive perspective. Consequently, a bottom-up, domain-specific approach for developing enterprise architecture patterns should be aimed for. Through iterative refinement of such patterns, reusable building blocks can be created. These reusable building blocks should address all relevant concerns of their level. On the software level, Normalized Systems has shown that this is a requirement for achieving anthropomorphism, as advocated by the Object-Oriented Paradigm. Anthropomorphism can only be achieved when a building

block can be used in a functional, black-box way, without the need to deal with concerns which are related to the construction of that building block.

This research approach suggests that no overarching enterprise architecture principles should be defined up front, since the complexity which is in scope is very large. Rather, a clear positioning of developed artifacts and the way in which they are evaluated allows a consistent maturing of the field. More specifically, this allows for various research initiatives, both in the short and long term. In the short term, such research goals could focus on a further specification of the research framework. First, a more specific scoping of the domains which can be designed and the issues which can be handled by design should be clarified. This could be related to, for example, research related to the identification of modular structures. While much research is already available on the modular structure of IT, products, business processes and organizational structures, other, less obvious, examples of modular structures are emerging, such as in the legal domain [40]. Methodologically, such research could benefit from an explorative case study approach. In order to observe relevant modular structures, in-depth knowledge of the field is required, which assumes a qualitative research approach. Such an approach allows the researcher to study a phenomenon which is not clearly separated from its environment [41]. This is necessary to observe modular dependencies which do not surface within the immediate environment of the phenomenon. Second, a specification of the functional—constructive gaps in an organizational context should be researched. This could related to, for example, architectural layers in enterprise architecture frameworks. Methodologically, a similar research approach as for the previous research goal could be used. However, researchers which actually implement enterprise architecture frameworks in organizations could provide valuable insights for this research goals. Therefore, an action research approach could be attempted as well. Third, a taxonomy of combinatorial effects should be created, which allows classification of empirical observations of such effects. Such a taxonomy could elaborate on the classification of horizontal versus vertical combinatorial effects [31]. This research goal could aggregate the research results of cases performed for the two previous research goals. Fourth, a taxonomy for different types of solutions could be proposed. On the software level, combinatorial effects are prevented by using pattern expansion. However, other solution types can be better suited for organizational designs, such as centralization or bus patterns. These solution types could build on generally applicable engineering knowledge. When exemplary solutions are observed in practice, these could be reported through explanatory case studies. However, the development of new solutions could be presented as genuine design science research.

On the long term, more mature research results can be aimed for. For specific domains, reusable patterns should be researched, which have a well-defined set of functionalities and handle a specific set of concerns. In this way, a more deterministic approach of building enterprise architectures can be achieved. For example, a pattern could be designed for an insurance or a production order. These patterns should be designed to deal with changes from not only their

functional domain (e.g., procurement), but from other domain as well (e.g., legal concerns). Given the current state of the field, a design science approach with multiple iterations seems necessary. An important catalyst for this type of research would be an outlet where concrete artifact designs can be discussed and published.

6 Conclusions and Limitations

In this paper, we examined how enterprise architecture frameworks may improve the emerging field of enterprise engineering. We therefore examined how such frameworks deal with issues such as integration, change and complexity. We then proposed several components to work towards a research framework for performing enterprise engineering research, based on the approach which was used to develop the Normalized Systems theory. This theory has shown that prescriptive, product-based guidelines can be developed to approach software development adhering to engineering standards. We believe that a similar goal should be aimed for by the enterprise engineering community. Of course, not all enterprise architecture issues can be addressed by this approach. For example, social issues during enterprise architecture projects will not be resolved. The scoping of the issues which are relevant for an engineering approach therefore indicate the limitations of such an approach. While such issues are not denied, the issues identified in this paper can be considered to be relevant and unresolved. Focussing on such issues can therefore result in significant contributions for the enterprise architecture field.

References

1. Henderson, J.C., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming organizations. IBM Syst. J. 32, 4–16 (1993)
2. Anderson, J.C., Narks, J.A.: Capturing the value of supplementary services. Harvard Business Review 73(1), 75–83 (1995)
3. Lehman, M.: Programs, life cycles, and laws of software evolution. Proceedings of the IEEE 68, 1060–1076 (1980)
4. Eisenhardt, K.M., Martin, J.A.: Dynamic capabilities: What are they? Strategic Management Journal 21(10/11), 1105–1121 (2000)
5. Teece, D.J., Pisano, G., Shuen, A.: Dynamic capabilities and strategic management. Strategic Management Journal 18(7), 509–533 (1997)
6. Maier, M.W., Emery, D., Hilliard, R.: Ansi/ieee 1471 and systems engineering. Systems Engineering 7(3), 257–270 (2004)
7. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. IBM Systems Journal 31(3), 590–616 (1992)
8. Bernus, P., Nemes, L., Smidth, G.: Handbook on Enterprise Architecture. In: International Handbooks on Information Systems. Springer (2003)
9. Martin, R., Robertson, E.: A comparison of frameworks for enterprise architecture modeling. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 562–564. Springer, Heidelberg (2003)

10. Ross, J.W., Weill, P., Robertson, D.C.: Enterprise Architecture as Strategy – Creating a Foundation for Business Execution. Harvard Business School Press, Boston (2006)
11. Zarvic, N., Wieringa, R.J.: An integrated enterprise architecture framework for business-it alignment. In: Latour, T., Petit, M. (eds.) Proceedings of Workshops and Doctoral Consortium of the 18th International Conference on Advanced Information Systems Engineering (CAiSE 2006), pp. 262–270. Namur University Press, Luxembourg (2006)
12. The Open Group: The open group architecture framework (togaf) version 9 (2009), `http://www.opengroup.org/togaf/`
13. Korhonen, J.J., Hiekkanen, K., Lähteenmäki, J.: Ea and it governance — a systemic approach. In: Proceedings of the European Conference on Management, Leadership and Governance (ECMLG) (2009)
14. Lawrence, P.R., Lorsch, J.W.: Differentiation and integration in complex organizations. Administrative Science Quarterly 12(1), 1–47 (1967)
15. Hicks, H.G., Gullet, C.R.: Organizations: Theory and Behaviors. McGraw-Hill, New York (1975)
16. Gortner, H., Mahler, J., Nicholson, J.: Organization Theory: A Public Perspective. Dorsey Press (1987)
17. Gregor, S., Hart, D., Martin, N.: Enterprise architectures: enablers of business strategy and is/it alignment in government. Information Technology & People 20(2), 96–120 (2007)
18. Schöenherr, M.: Towards a common terminology in the discipline of enterprise architecture. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 400–413. Springer, Heidelberg (2009)
19. Wegmann, A.: The systemic enterprise architecture methodology (seam) - business and it alignment for competitveness. In: International Conference on Enterprise Information Systems 2003 (ICEIS 2003), pp. 483–490 (2003)
20. Op 't Land, M., Proper, E., Waage, M., Cloo, J., Steghuis, C.: Enterprise Architecture: Creating Value by Informed Governance, 1st edn. The Enterprise Engineering Series. Springer (December 2008)
21. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26(3), 276–292 (1987)
22. Schekkerman, J.: Trends in enterprise architecture: How are organizations progressing? Technical report, Institute For Enterprise Architecture Developments (2005)
23. Fatolahi, A., Shams, F.: An investigation into applying uml to the zachman framework. Information Systems Frontiers 8(2), 133–143 (2006)
24. Council, C.: A practical guide to federal enterprise architecture (2001)
25. Pereira, C.M., Sousa, P.: A method to define an enterprise architecture using the zachman framework. In: SAC 2004: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 1366–1371. ACM, New York (2004)
26. Buckl, S., Ernst, A.M., Matthes, F., Ramacher, R., Schweda, C.M.: Using enterprise architecture management patterns to complement togaf. In: IEEE International Enterprise Distributed Object Computing Conference, pp. 34–41. IEEE Computer Society, Los Alamitos (2009)
27. Mannaert, H., Verelst, J.: Normalized Systems—Re-creating Information Technology Based on Laws for Software Evolvability, Koppa, Kermt, Belgium (2009)
28. Mannaert, H., Verelst, J., Ven, K.: The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability. Science of Computer Programming 76(12), 1210–1222 (2011)

29. Mannaert, H., Verelst, J., Ven, K.: Towards evolvable software architectures based on systems theoretic stability. Software: Practice and Experience 42(1), 89–116 (2011)
30. Van Nuffel, D.: Towards Designing Modular and Evolvable Business Processes. PhD thesis, University of Antwerp (2011)
31. Huysmans, P.: On the Feasibility of Normalized Enterprises: Applying Normalized Systems Theory to the High-Level Design of Enterprises. PhD thesis, University of Antwerp (2011)
32. Baldwin, C.Y., Clark, K.B.: Design Rules. The Power of Modularity, vol. 1. MIT Press Books. The MIT Press (January 2000)
33. Richardson, G.L., Jackson, B.M., Dickson, G.W.: A principles-based enterprise architecture: lessons from texaco and star enterprise. MIS Quarterly 14(4), 385–403 (1990)
34. Lindstrom, A.: On the syntax and semantics of architectural principles. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS 2006), vol. 8, p. 178b (2006)
35. Weinberg, G.M.: An Introduction to General Systems Thinking. Wiley-Interscience (1975)
36. Bertalanffy, L.V.: General Systems Theory: Foundations, Development, Applications. George Braziller, New York (1968)
37. Bunge, M.: Treatise on Basic Philosophy. Ontology II: A World of Systems, vol. 4. Reidel, Boston (1979)
38. Gero, J.S., Kannengiesser, U.: The situated function-behaviour-structure framework. Design Studies 25(4), 373–391 (2004)
39. Sutcliffe, A.G., Maiden, N.A.M.: Domain modeling for reuse. In: Frakes, W.B. (ed.) Third International Conference on Software Reuse, pp. 169–177. IEEE Computer Society Press, Los Alamitos (1994)
40. Smith, H.E.: Modularity in contracts: Boilerplate and information flow. American Law & Economics Association Annual Meetings Paper 46 (2006)
41. Yin, R.K.: Case Study Research: Design and Methods, 3rd edn. Sage Publications, Newbury Park (2003)