Separation of Modeling Principles and Design Principles in Enterprise Engineering

Tetsuya Suga, Peter Bruyn, Philip Huysmans, Jan Verelst, Herwig Mannaert

Separation of Modeling Principles and Design Principles in Enterprise Engineering

Tetsuya Suga[1], Peter De Bruyn[2], Philip Huysmans[2], Jan Verelst[2], and Herwig Mannaert[2]

[1] Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8550, Japan,
suga.t.ac@m.titech.ac.jp
[2] University of Antwerp, Prinsstraat 13, Antwerp, 2000, Belgium,
{peter.debruyn,philip.huysmans,jan.verelst,herwig.mannaert}@uantwerp.be

Summary. An agile enterprise requires evolvable information systems and organizational structures. Some design theories, e.g. Normalized Systems theory, which were originally developed for designing information systems, have been generalized and extended for the design of organizations. In addition to information systems, Normalized Systems theory has recently expanded its applicability into the organizational level, including business process. This resulted in a set of 25 design guidelines for Normalized Systems Business Processes. On the other hand, Enterprise Ontology provides advantages in understanding the essence of organizations for massive abstraction and complexity reduction. Since these two streams of research apparently have similar goals, i.e. designing enterprises, using different approaches, some early studies tried to compare or combine them. However, most of them achieved limited success. This research investigates the literature, looking for a new way to connect them. It concludes that it may be possible to sequentially utilize those two artifacts in two different phases of enterprise engineering.

Key words: enterprise engineering, design principle, modeling principle, enterprise ontology, normalized systems theory

1 Introduction

Enterprise engineering is an emerging sub-discipline of systems engineering that studies enterprises from an engineering perspective. Loosely summarized, enterprise engineering has attempted to view enterprises from the perspective of engineering and then (re)design and (re)implement them. Many researchers and practitioners have been working on this new challenge and have produced a variety of artifacts, such as ArchiMate [1], Enterprise Ontology (EO) [2], 4EM [3], MEMO [4], Normalized Systems theory (NS) [5], S-BPM [6], and so on. On the other hand, some of those artifacts seem to exhibit some overlap in purpose. However, since they have been developed to achieve their own specific goals, they have their own aptitudes by nature. Instead of pursuing the ultimate and apparently arduous goal of unifying them into a single theory, it seems both feasible and valuable to explore a way to combine some of those artifacts in a beneficial manner.

Therefore, this paper tries to find a possible way to combine two artifacts that have recently raised interest, namely EO and NS, rather than unifying them. Indeed, there are several existing studies which tried to compare and potentially find a clue to the unification of the two artifacts in the past. However, most of them achieved limited success, possibly because the two artifacts were compared in the same class of artifacts. In contrast, the authors believe that they should be located in different classes: **modeling principles** and **design principles**, respectively. Therefore, this paper investigates the literature, keeping this hypothesis in mind. Stated another way,

this study will answer the following research questions: (1)Is EO a modeling principle? (2)Is EO a design principle? (3)Is NS a modeling principle? (4)Is NS a design principle?

The remainder of this article is organized as follows: Section 2 provides brief introductions of the theories of EO and NS, related past works, and the basics of modeling and design principles in the context of model-based systems engineering. Section 3 is the main part of this paper, answering the research questions by collating descriptions of EO and NS with ones of modeling and design principles. After reflection and discussion in Section 4, Section 5 concludes this article with possible future directions for research.

2 Literature Review

2.1 Theoretical Foundation

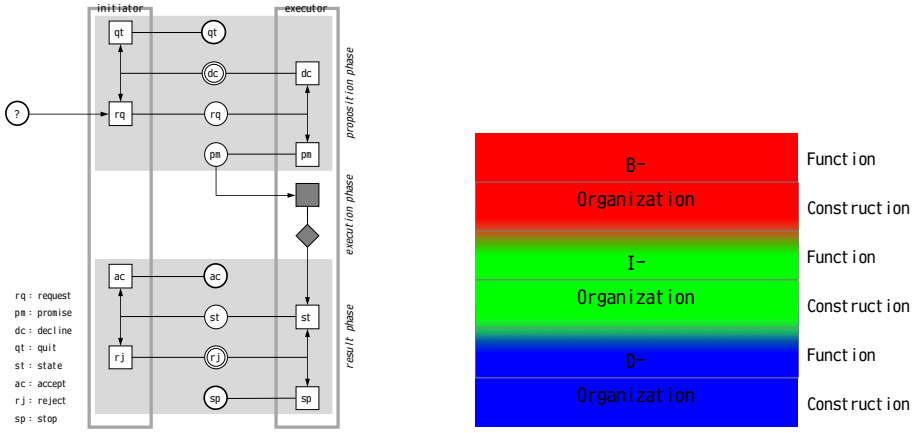
Enterprise Ontology[1] EO can be loosely described as a well-founded distinctive set of notions, theories, and a methodology for grasping and steering the complexity of enterprises. Here, *enterprise* is an overall term to identify an intentionally created entity of human endeavor with a certain purpose, e.g. a company, organization, business, governmental agency, and so forth [7, p. 4]. Although EO may accommodate more than one methodology in principle, there exists only one methodology available at this moment, known as "Design & Engineering Methodology for Organizations (DEMO)". The basic assumption of EO is that enterprises are highly complex, as well as highly organized, entities, and thus, a formal theory and methodology are required to achieve the purpose of an enterprise [7, p. 4]. By collating this comprehension against the definition of a system, enterprises are regarded as systems and are now a subject for systems engineering.

The Four Axioms. EO defines its ways of thinking in four *axioms*[2]. The **operation axiom** states that "[...] the operation of an enterprise is constituted by the activities of actor roles, which are elementary chunks of authority and responsibility, fulfilled by subjects" [2, p. 81]. It also classifies the activities and facts produced by the activities as either production acts/facts or coordination acts/facts (C-acts/facts). The **transaction axiom** states that C-acts are performed sequentially following the steps in the universal pattern, called a *transaction* (Fig. 1a). Some C-acts may be performed implicitly or tacitly. The **composition axiom** defines how transactions are interrelated: every transaction is enclosed in another transaction, or is triggered by actor roles in the environment, or is self-activated. This provides a well-founded definition of a business process: a collection of causally related transaction kinds. The **distinction axiom** defines another classification of human activities into *original*, *informational*, and *documental*. Indeed, this axiom asks modelers to ignore informational and documental activities for a substantialreduction of complexity.

The Organization Theorem. In mathematics, a theorem is a statement that is not self-evidently true but is proven to be true as a logical consequence of axioms. Similarly, as derived from the aforementioned four axioms, the **organization theorem** states that an organization is a heterogeneous system that is constituted as the layered integration of three homogeneous systems: the B-organization for operating *original* activities, the I-organization for operating *informational* activities, and the D-organization for *documental* activities (Fig. 1b).

[1] The contents of this part are mainly based on [2], unless otherwise stated.

[2] An axiom is *a statement or proposition which is regarded as being established, accepted, or self-evidently true* [8]. Although it is possible to discuss whether these axioms are correct or the best for managing complexity in enterprises, those arguments are somewhat beyond the scope of EO.



(a) The Standard Transaction Pattern [2, p. 94] (b) The Organization Theorem [2, p. 118]

Fig. 1: Concepts in Enterprise Ontology

Normalized Systems Theory[3] NS theory aims to design systems pursuing evolvability and observability, originally in the field of information systems development. The basic assumption is that information systems must be able to evolve over time while accommodating future changes which defy Lehman's law of increasing complexity, which is a part of his software evolution laws [9]. To accommodate future changes, information systems must exhibit stability against those changes. It means that the impact of a change should only be dependent on the nature of the change itself [5, pp. 106–107]. Otherwise, if the impact of a change—*ripple effect*—is dependent on the size of the system, thus if it requires more effort as the information system grows, the ripple effect is specifically called a *combinatorial effect*. High evolvability is achieved by removing combinatorial effects.

NS Theorems. NS gets involved in the process of transforming requirements R_i (i.e. functions) into primitives P_j (i.e. construction). In order to achieve a good F-to-C transformation that does not produce any combinatorial effects, NS has proposed a set of four NS *theorems* [10, pp. 88–96][4]:

– SoS; Separation of States	When a primitive uses another primitive as it is executed, both primitives should be separated by a state.
– SoC; Separation of Concerns	A primitive should only contain one concern.
– VT; Version Transparency	A primitive which is used by other primitives should be version transparent.
– IT; Instance Traceability	The input received to execute a particular primitive instance, as well as the output delivered by that primitive instance, should be traceable to the particular version and instance of that primitive.

If the transformation is performed along with the theorems, it should look like Fig. 2b, in contrast to a typical transformation like Fig. 2a, which may contain combinatorial effects. These theorems are independent of application domain.

Guideline and Element. From the theorems, a set of *guidelines* can be derived for a specific application domain, such as information system management, business process reengineering,

[3] The contents of this part are mainly based on [5], unless otherwise stated.
[4] NS theorems in their early stage in software were defined in a different way (see [5, pp. 112–119]).

accounting, enterprise resource planning, etc. The guidelines can be actualized by *elements*, which are reusable sets of primitives. Guidelines and elements are often dependent on specific application domains, in contrast to theorems. In software engineering as a specific application domain, six NS guidelines are derived from the four NS theorems and are implemented by a set of five *elements* (which allow the expansion of software which adheres to these principles): data element, action element, workflow element, trigger element, and connector element. In business process engineering, the theorems are re-interpreted to derive guidelines. The theorems produce 25 specific guidelines named Normalized Systems Business Process (NSBP) guidelines. Currently, no elements implemented by NSBP are available.

2.2 Related Works

[11] studies an alignment of the construct of EO and that of NS by expressing the transaction pattern of EO in the form of NS elements, namely data elements and workflow elements. In order to implement transactions without combinatorial effects, it prescribes two guidelines: (1)additional state transitions need to be created in order to comply with the SoC and SoS theorem, and (2)the cancellation patterns should be implemented with extensions to accommodate their adherence to the NS theorems. The study mainly discusses an emergence of combinatorial effects during the implementation process from the ontological model to the real implementation. Put another way, whether combinatorial effects may occur within the ontological model and even how to remove combinatorial effects remain beyond the scope of the study.

[12] also studies an alignment and combination of EO and NS by establishing a linkage and collaboration between agile enterprises (realized with EO), and agile automated information systems (developed with NS). This research proposes mappings of concepts in EO onto elements of NS with wider coverage than [11]. It also points out that aspects such as user interfaces and non-functional requirements, which are not presented in EO, are considered as aspects in cross-cutting concerns, which should be addressed separately from the functional requirements in NS. Moreover, it reveals that NS does not yet support all concepts of EO, such as a derived fact kind and information link.

[13] is another work with an approach different from the previous two studies [11, 12], in which NS elements for software are directly compared and mapped to EO. Instead of the direct approach, this study compares EO concepts and not NS elements for software but the guidelines of NS Business Processes (NSBP), which are derived from NS theorems for the business process domain. Then, it analyzes to what extent the NSBP guidelines are consistent, complementing, or conflicting with prescriptions from EO. This study concludes that 13 of 25 NSBP guidelines are consistent, 12 of them are ignored and don't appear in EO, and 4 of them are conflicting (there is some overlapping).

In these studies on the relationship between EO and NS, the two artifacts are compared in the same class of artifacts.

$$\begin{bmatrix} P_1 \\ \vdots \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,l} \\ \vdots & \ddots & \iddots & \vdots \\ \vdots & \iddots & \ddots & \vdots \\ x_{n,1} & \cdots & \cdots & x_{n,l} \end{bmatrix} \times \begin{bmatrix} R_1 \\ \vdots \\ \vdots \\ R_n \end{bmatrix} \qquad \begin{bmatrix} P_1 \\ \vdots \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & & 0 \\ & \ddots & \\ & & \ddots \\ 0 & & x_{n,l} \end{bmatrix} \times \begin{bmatrix} R_1 \\ \vdots \\ \vdots \\ R_n \end{bmatrix}$$

(a) A Typical Transformation (b) A Normalized Transformation

Fig. 2: Conceptual Function-to-Construction Transformation in NS [10, p. 76]

2.3 Model-based Systems Engineering Methodology

This section revisits model-based engineering (MBE) to formulate distinct perspectives on modeling principles and design principles to prompt a good understanding of this article.

Modeling and Design MBE is sequentially decomposed into two parts: *modeling* and *design*. Since these terms mean different things to different people, the following paragraphs are cited from the literature to formulate the meaning of them within the scope of this article: Quote 1 for an overview, Quotes 2 and 3 for modeling, and Quote 4 for designing.

Quote 1 ([14, p. 2])**.** *John G. Truxal, former Dean of Engineering at Brooklyn Polytechnic Institute, says, "Systems engineering includes two parts: modeling, in which each element of the system and the criterion for measuring performance are described; and optimization, in which adjustable elements are set at values that give the best possible performance."*

Quote 2 ([15, p. 107])**.** *A conceptual model is a mental image of a system, its components, its interactions. It lays the foundation for more elaborate models, such as physical or numerical models. A conceptual model provides a framework in which to think about the workings of a system or about problem solving in general. An ensuing operational model can be no better than its underlying conceptualization.*

Quote 3 ([16, p. 1])**.** *The role of conceptual modelling in information systems development during all these decades is seen as an approach for capturing fuzzy, ill-defined, informal "real-world" descriptions and user requirements, and then transforming them to formal, in some sense complete, and consistent conceptual specifications.*

Quote 4 ([17, p. 4])**.** *The engineer, and more generally the designer, is concerned with how things ought to be how they ought to be in order to attain goals.*

Principles Let us focus on the role of principles in modeling/design as a normative notion that guides system design. A modeling/design guideline is a guideline or a set of guidelines that assist(s) us to achieve a good model/design and avoid a bad model/design. It also defines what is *good* and *bad*. It is often referred to as design guidelines, patterns, architectures, theories, et cetera.

Modeling Principle. The nature of modeling is characterized by three concepts: **mapping**, i.e. which element in the real world and which element in the model should be mapped with each other; **reduction**, i.e. which element in the real world should be selected or ignored; and **pragmatism**, i.e. models as the outcome of modeling activities should be made to be utilized [18, p. 157]. Thus, a modeling principle should provide guidelines for mapping, reduction, and pragmatism.

Design Principle. The nature of design is difficult to summarize. Nevertheless, an anatomy of design theory proposed by [19] defines the characteristics of information systems design theory (regarded as a design principle) by specifying core components: **purpose and scope**, i.e what the system is for and what the scope of the theory is; **constructs**, i.e. what the entities in the theory are; **principle of form and function**, i.e. what the abstract description of an artifact is (describing either product or method/intervention); **artifact mutability**, i.e. how and to what degree the changes in state of the artifact are anticipated; **testable propositions**, i.e. what the true/false statements are for the design theory; **justificatory knowledge**, i.e. what the kernel theories[5] of the design theory are; **principles of implementation**, i.e. what the processes of implementing the theory (either product or method) in specific contexts are; and **expository instantiation**, i.e. what a physical prototype is. Even though this anatomy was proposed in the context of information system design, it seems able to cover other disciplines to a certain extent.

George Klir (1996) "Review of Model-Based Systems Engineering", in the International Journal of General Systems, Vol 25 (2). p. 179.

[5] kernel theory: underlying knowledge or theory from the natural or social or design sciences [19]

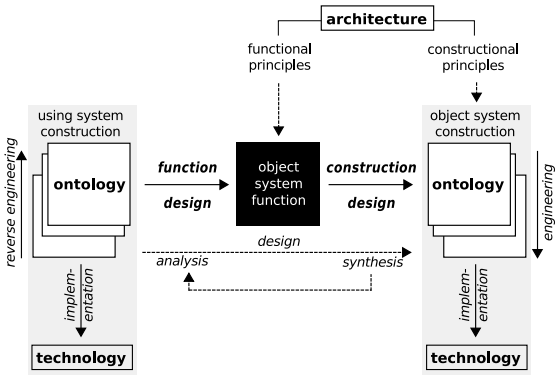


Fig. 3: General System Design Process[6][20]

3 Separation of Modeling Principles and Design Principles

This section examines EO and NS by comparing each of them against the characteristics of modeling principles and design principles, as revisited in Section 2.3.

Is EO a modeling principle? [2, p. 10] states that "Our goal is to understand the essence of the construction and operation of complete systems; more specifically, of enterprises." This adheres to the general characteristics of modeling in the sense of Quote 3. It is also stated that "Firstly, the P[rocess]M[odel] facilitates discussions about the redesign, as well as the re-engineering of business processes [...]." in [2, p. 10], which also adheres to Quote 2. This point is also supported by the fact that the acronym of DEMO, a methodology of EO, was initially Dynamic Essential MOdelling (and later Dynamic Essential Modeling of Organizations). Therefore, EO is along the lines of modeling.

Moreover, EO seems to fully satisfy the conditions as a modeling principle as formulated in Section 2.3. As explained in Section 2.1, the operation axiom defines what and how the **mapping** should be—i.e. what the elements in enterprises are, and how to identify these elements—and the distinction axiom defines the **reduction**—i.e. which among these elements should be included in the model. The general direction of EO also adheres to the concept of **pragmatism**. Therefore, EO can be regarded as a sort of modeling principle.

Is EO a design principle? In EO, it seems that there does not exist any statement regarding design principles except the system design process [2, pp. 73–77], called the Generic System Design Process (GSDP) in [20]. GSDP defines the first step as *function design*—designing the function of the object system (OS) based on the construction of the using system (US)—and the second step as *construction design*—designing the construction of the OS based on the function of the OS (Fig. 3). However, it does not define guidelines for the substantial procedure, i.e. design principles, in the function design or construction design. Those substantial guidelines are called *architectures* in GDSP [20], but the content of the architecture is not specified there. Put differently, GDSP prescribes an ideal design process from a meta-meta-level. Therefore, EO is substantially not a design principle.

This observation is supported by comparing GSDP to the components of design principles prepared in Section 2.3 and finding that they are just not on the same wavelength. For instance, **purpose and scope** might mean to "understand the essence of the construction and operation

[6] The term *principle* used in the figure refers somehow to a different notion than previously mentioned.

of complete enterprises", but they are too abstract in contrast to the ones in NS—to be evolvable. Similarly, although **constructs** might be actor roles and transaction kinds, these are as primitive as attributes and methods, or at the most classes; it seems unknown how to aggregate those primitives into reusable elements, for achieving the purpose (which is also as yet unspecified by EO). The **justificatory knowledge** of EO is social science and systems engineering. However, other components, except the ones mentioned so far, are misted. Therefore, it is hard to assert that EO is a design principle.

Is NS a modeling principle? It seems that NS does not define anything about **mapping** in either of information systems or business processes. Indeed, research in NSBP employs an external artifact, namely Business Process Model and Notation (BPMN), for mapping the real world to the models. The transformation matrix may look like a mapping, but this is not the one specified in Section 2.3, where mappings between objects in the real world and elements in the model are defined, because the transformation matrix represents another kind of mapping from the function (requirements R_i) in the model to the construction (primitives P_j) in the model: $R_i \rightarrow \{P_j\}$. This is also the case for **reduction**. Therefore, it is asserted that NS is far from a modeling principle.

Is NS a design principle? It is almost an established understanding that NS is a design principle. NS was originally a design principle in the domain of information system development, but has been generalized and extended to be a design principle in the domain of business processes. In both the domains, the purpose is the same, i.e. to enhance evolvability and observability. The components of design principles are fully specified in [21]. Therefore, NS is regarded as a design principle.

4 Discussion

Reflection on EO A question then arises: what is the rationale behind the absence of design principles in EO? The answer might be given if we see the motivation of EO: starting from the point that enterprises as a phenomenon existed in the real world, but no models were available, due to the lack of modeling artifacts, to explicitly identify the components and capture interactions and mechanisms. Although EO with DEMO is occasionally considered as a design theory for organizations, the result of this present research implies that EO has accomplished the hard and important work of making enterprises design*able* and engineer*able*, rather than continuing to propose guidelines for better designs of enterprises on the layer of models. This achievement has definitely had an impact, as supported by the fact that EO and DEMO have been used on many occasions.

Strictly speaking, EO includes GSDP, as seen in Section 3. However, GSDP can be positioned in the meta-meta-level and still allow room for design principles at the meta-level to get involved. In other words, EO is noncommittal and neutral regarding design principles. Thus, as long as the model is made to adhere to the standards of EO, EO does not provide any decision criterion to evaluate the model—i.e. whether a model is better or worse than another model for the same requirement.

Another interpretation is that EO as a modeling principle may work with more than one design guideline. The design principles might be brought by NS and/or other design principles (e.g. Service-Orientation systems [22]). Unless the design principles are ones analogically transplanted from other disciplines, they can be directly built on EO, as seen in [23], in which guidelines for splitting and allying enterprises are proposed based on organizational science. "Neither modeling principles nor design guidelines are superior, nor does one include the other. Modeling principles

and design guidelines are at work in different layers. In other words, if either modeling principle or design principle is ignored, the whole process of engineering cannot be completed.

Reflection on NS Another question arises: what is the rationale behind the absence of modeling principles in NS? In part, it is because that computers have been developed with the real world and conceptual models together. Here, models may refer to models represented by modeling languages such as Unified Modeling Language, or even refer to source code written in a programming language. Since the behavior of computers or CPUs is fully understood, we are knowledgeable about the components of software and their internal working mechanisms. That knowledge guarantees the solid and fine-grained alignment between these models and actual behavior including electronic phenomena in semiconductor circuits. Therefore, it requires almost no concern, and we take it for granted. This seems to be a clue which leads to answering why modeling principles are not often discussed in the field of software engineering. Instead, many discussions are diverted to the next step, i.e. how to develop a good design. Put differently, this idea is supported by the fact that when NS was extended into the business process domain, a modeling principle was required, and indeed BPMN was adopted.

5 Conclusion

The primary conclusion is that there is an orthogonality or complementarity between modeling principles and design guidelines. In other words, modeling principles and design principles are at least two different independent variables in enterprise engineering. This implies that EO as a modeling principle may work with design principles such as NS. In general, more than one design guideline can be employed at the same time. For example, the service-orientation design principles can be another design principle for designing enterprises. It may happen that some design guidelines conflict with each other.

The results also show that substantial design guidelines may swing and miss in a certain modeling principle, at least in EO. On the other hand, a substantial number of NSBP guidelines do not emerge in EO models because what the NSBP guidelines state are observed in the informational layer, and thus, are ignored in DEMO models. Moreover, the possibility should not be dismissed that a modeling principle itself might conflict with a design principle. Indeed, [13] pointed out that a very few components of EO conflict with the design guidelines of NS.

The direction of future research is to explore other principles that might be helpful in designing and instantiating enterprises. One approach is to fix a selection of the modeling principles and use design principles as a free variable. For instance, an as-is EO model (DEMO model, not EO or DEMO itself, but a model) can be redesigned for evolvability and observability by deriving a design principle from NS by analogy. The same original as-is DEMO model can also be redesigned differently for a service-oriented structure by deriving a design principle from service-oriented systems by analogy. The other approach is to fix a selection of the design principles and use modeling principles as a free variable. For example, NS may be able to provide a design principle for DEMO models. It may also furnish one for ArchiMate models, for instance. Since the existence of congeniality or uncongeniality between modeling principles and design principles are implied in the previous paragraph, this direction also awaits further investigation.

Acknowledgment

We thank Professor Junichi Iijima at Tokyo Institute of Technology, for assistance and comments that greatly improved the manuscript and presentation.

References

1. The Open Group, *Archimate 3.0 Specification*, Van Haren Publishing, Ed. Van Haren Pub, 2016.
2. J. L. G. Dietz, *Enterprise Ontology: Theory and Methodology*. Springer Berlin Heidelberg, 2006.
3. K. Sandkuhl, J. Stirna, A. Persson, and M. Wißotzki, *Enterprise Modeling: Tackling Business Challenges with the 4EM Method*. Springer-Verlag Berlin Heidelberg, 2014.
4. U. Frank, "Multi-perspective enterprise modeling (MEMO) conceptual framework and modeling languages," *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pp. 1258–1267, 2002.
5. H. Mannaert and J. Verelst, *Normalized Systems: Recreating Information Technology with Laws to Control Its Marginal Cost*. Koppa, 2009.
6. A. Fleischmann, W. Schmidt, C. Stary, S. Obermeier, and E. Börger, *Subject-Oriented Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
7. J. A. P. Hoogervorst, *Enterprise Governance and Enterprise Engineering*, ser. The Enterprise Engineering Series. Springer Berlin Heidelberg, 2009.
8. Oxford University Press. (n.d.) "axiom". Oxford Dictionaries. Accessed on April 15, 2016. [Online]. Available: http://www.oxforddictionaries.com/definition/english/axiom
9. M. M. Lehman, "Programs, Life Cycles, and Laws of Software Evolution," in *Proceedings of the IEEE*, vol. 68, no. 9. IEEE, 1980, pp. 1060–1076.
10. P. De Bruyn, "Generalizing Normalized Systems Theory: Towards a Foundational Theory for Enterprise Engineering," Ph.D. Thesis, University of Antwerp, 2014.
11. P. Huysmans, D. Bellens, D. Van Nuffel, and K. Ven, "Aligning the Constructs of Enterprise Ontology and Normalized Systems," in *Advances in Enterprise Engineering IV*, ser. LNBIP. Springer Berlin Heidelberg, 2010, pp. 1–15.
12. M. R. Krouwel and M. Op 'T Land, "Combining DEMO and normalized systems for developing agile enterprise information systems," in *Advances in Enterprise Engineering V*, ser. LNBIP. Springer Berlin Heidelberg, 2011, pp. 31–45.
13. D. Van Nuffel, P. Huysmans, and P. De Bruyn, "Engineering Business Processes: Comparing Prescriptive Guidelines from EO and NSBP," in *Business Modeling and Software Design*, ser. LNBIP. Springer International Publishing, 2014, pp. 84–105.
14. W. L. Chapman, A. T. Bahill, and A. W. Wymore, *Engineering Modeling and Design*, ser. Systems Engineering (Book 2). CRC Press, 1992.
15. H. N. Pollack, *Uncertain Science ... Uncertain World*. Cambridge University Press, 2005.
16. J. A. Bubenko jr, "From Information Algebra to Enterprise Modelling and Ontologies – a Historical Perspective on Modelling for Information Systems," in *Conceptual Modelling in Information Systems Engineering*. Springer Berlin Heidelberg, 2007, ch. 1, pp. 1 – 18.
17. H. A. Simon, *The Sciences of the Artificial*, 3rd ed., ser. MIT Press. The MIT Press, 1969.
18. H. Stachowiak, *Allgemeine Modelltheorie*. Springer-Verlag, Wien, 1973.
19. S. Gregor and D. Jones, "The anatomy of a design theory," *Journal of the Association for Information Systems*, vol. 8, no. 5, pp. 312–335, 2007.
20. J. L. G. Dietz and J. A. P. Hoogervorst, "Enterprise ontology in enterprise engineering," in *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*. ACM Press, 2008, p. 572.
21. P. Huysmans, G. Oorts, D. Bruyn, H. Mannaert, and J. Verelst, "Positioning the Normalized Systems Theory in a Design Theory Framework," in *Business Modeling and Software Design*, ser. LNBIP. Springer Berlin Heidelberg, 2013, pp. 43–63.
22. A. Lupeikiene and A. Caplinskas. (n.d.) From Needs to Requirements Specification in Service-Oriented Systems Engineering. Accessed on April 15, 2016. [Online]. Available: http://www.mii.lt/paslaugu_internetas/rodikliai/1veikla/1.2p.pdf
23. M. Op 't Land, "Applying Architecture and Ontology to the Splitting and Allying of Enterprises: Problem Definition and Research Approach," in *On the Move to Meaningful Internet Systems 2006*, ser. LNCS. Springer Berlin Heidelberg, 2006, pp. 1419–1428.