Towards Functional and Constructional Perspectives on Business Process Patterns

Peter De Bruyn, Dieter Van Nuffel, Philip Huysmans, Herwig Mannaert
Department of Management Information Systems
University of Antwerp
Antwerp, Belgium
{peter.debruyn,dieter.vannuffel,philip.huysmans,herwig.mannaert}@ua.ac.be

***Abstract*—Contemporary organizations need to be more agile to keep up with the swiftly changing business environment. The Normalized Systems theory has proven to introduce this required agility within an organization, starting at the software level. However, in order to realize an agile enterprise, also business processes have to exhibit this evolvability. Currently, the relevance of Normalized Systems theory at the business process level has been demonstrated, however no equivalent to the software elements at the organizational level have been developed. Therefore, this paper investigates whether it is possible to base such elements on the available business process patterns in literature. After investigating the usefulness of the MIT Process Handbook with regard to this purpose, this paper emphasizes the importance of recognizing the so-called functional–constructional gap and identifies the need for developing modular and evolvable constructional business process design patterns to further extend Normalized Systems theory on the business level.**

***Index Terms*—Normalized Systems, business process patterns, analysis patterns, evolvability, MIT Process Handbook**

I. Introduction

Contemporary organizations need to be more agile to keep up with the swiftly changing business environment. As a consequence, all constructs of an organization—structure, business processes, information systems—have to evolve at an equivalent pace. The Normalized Systems (NS) theory has proven to introduce this required agility within an organization. First, the theory prescribes how to design and implement information systems that are able to evolve over time, and are thus designed to accommodate change [1]. It is based on the systems theoretic concept of stability and on the prevention of so-called combinatorial effects, i.e., changes of which the impact is not only dependent on the kind of the change but also on the size of the system. As such, NS proposes four design principles that need to be adhered at all times [2]:

- *separation of concerns* requires that every change driver or concern is separated from other concerns;
- *data version transparency* requires that data is communicated in version transparent ways between components;
- *action version transparency* requires that a component can be upgraded without impacting the calling components;
- *separation of states* requires that actions or steps in a workflow are separated from each other in time by keeping state after every action or step.

The design principles show that software constructs, such as functions and classes, by themselves offer no mechanisms to accommodate anticipated changes in a stable manner. The NS theory therefore proposes to encapsulate software constructs in a set of five higher-level software elements: action element, data element, workflow element, trigger element, and connector element [3]. These elements are modular structures that adhere to these design principles, in order to provide the required stability with respect to anticipated changes [2]. As these elements themselves are free of combinatorial effects, also the applications based on them are free of combinatorial effects.

However, it does not suffice to introduce agility within the information systems to realize an agile enterprise. Other organizational artifacts have to evolve in the same way as well. Therefore, the NS theory was extended to other organizational elements, such as business processes and enterprise architectures [4]. Regarding the former, business processes, the applicability of the extension is already demonstrated [5].

Nevertheless, the authors have not yet been able to identify an equivalent of the five software elements at the business process level. Although preliminary research findings indicate that Notification and Payment might classify as such a business process element [5], additional research is required. Therefore, this paper investigates whether it is possible to base such elements on the available business process patterns in literature.

When selecting appropriate business process elements, an important distinction needs to be made between patterns from a functional and the constructional perspective. The functional and constructional view on a system are fundamentally different conceptualizations of a system [6]. The *functional perspective* is concerned with the external behavior of the system [7]. This perspective is adequate for the purpose of using or controlling a system. Therefore, knowledge of the required input variables, transfer function and output variables are key components of this perspective. In contrast, the *constructional perspective* describes what a system really is [8]. In this perspective, knowledge about the composition (i.e., which components constitute the system) and structure (i.e., how these components are related) is focused on. The function of a system is brought about by the operation of its construction. However, the construction cannot be deduced from the functional description, since the two perspectives deal with fundamentally different components. Consistent with the NS elements, we aim to propose process elements using a constructive perspective. Therefore, it is important to consider these perspectives when building on business process patterns found in literature.

The remainder of this paper will be structured as follows: in Section II we will briefly discuss some already existing analysis and design patterns in literature. Next, we will investigate to which extent we can derive corresponding NS conform elements and patterns from those available business process patterns by studying one frequently cited and used framework, being the MIT Process Handbook, in Section III. Afterwards, we will discuss how NS theory extends towards normalized business process design patterns and Section V will end up with some final comments and opportunities for further research.

II. Related Work

The use of patterns in software development and information systems analysis has been increasingly gaining attention during the past decade. One of the first publications on software development patterns that generated considerable interest was probably the so-called Gang of Four (GoF) book of Gamma et al. [9]. Gamma et al. identified patterns as ideas that senior developers have used many times while solving commonly occurring problems [9]. Essentially, the overall meaning or intention of this concept has remained rather unchanged throughout many later publications on design patterns. Further summarizing, patterns are frequently claimed to exhibit the following characteristics:

- starting from a generally occurring problem in the considered problem domain;
- proposing standard and / or best practice solutions to these problems applicable to a myriad of analogous situations;
- incorporating domain knowledge and expertise sometimes requiring multiple years of experience to gather independently;
- exhibiting high-quality and robustness by representing frequently tested solutions;
- increasing pace of the development / modeling process by avoiding to systematically start from scratch and trying to ‘reinvent the wheel’.

Moreover, they are generally claimed to be a sign of a discipline becoming somewhat more mature, in the sense that an accumulation of generally recurring problems and their best-practice solutions becomes identified and documented. As such, existing knowledge from experts can be consolidated, published, and made available for a whole community [10].

After the work of Gamma et al. [9], additional, more analysis-oriented frameworks arised. As such, we will present here a brief illustrative, yet not exhaustive, overview. Given the plethora of available frameworks, a lot of different classification approaches exist as well. Some pattern frameworks for example mainly focus on the data aspects of an organization model, such as Hay [11] and Fowler [12]. Elaborating on these previous two frameworks, an interesting work was also delivered by Silverston providing domain data models for several industries such as manufacturing, telecommunications, health care, insurance, etc. [13], [14]. More process-oriented patterns can be found in, for example, Larman [15]. Furthermore, some claim that the broad area of workflow patterns are to be considered as some form of process-oriented analysis patterns (see e.g., [16]). Scheer also provided domain models for several functional domains of a typical organization, and combined both data and process related aspects [17]. Moving to more abstract levels, some frameworks claiming to state more generic and universal patterns can be noticed. For example, Dietz models every enterprise as an aggregation of instantiations of one universal transaction pattern [18]. Also REA (resources, events, agents) similarly views an enterprise as an aggregation of transactions representing some kind of economic exchange [19], [20]. Finally, one could argue that also some general reference models could be considered to a certain extent as analysis patterns, such as the value chain model of Porter [21], the eTOM model for the telecommunications sector or the SCOR model representing a reference model for supply chain operations [22].

III. Investigating Current Business Process Design and Reference Models

In this section we will discuss the extent to which currently available analysis patterns and reference models can be applied to and serve as a means to extent the previously discussed NS theory to the level of normalized design patterns at the business level (i.e., modular and evolvable business process design patterns). As the number of available frameworks in this regard is rather extensive at first sight (cf. Section II) and due to the limited available space, we chose to focus our attention initially to only one framework that formulates a number of functional patterns at the business process level: the MIT Process Handbook. This approach allowed us to analyze this one specific framework in a rather profound way. The pattern framework was selected mainly because of the fact that it is a generally well known, publicly available framework, extensively discussed and referred to in both academic and practitioners literature. Also, as will be further clarified later on, this framework’s motivation seems to be closely resembling our previously stated purpose in Section I: the formation of a repository consisting out of generally reusable business process patterns. As such, the purpose is to investigate whether these functional patterns can be translated in the required modular and evolvable design patterns.

The MIT Process Handbook initiative originated around 1994 reacting to an identified need of enabling more easily business process redesign and the knowledge management regarding those business processes [23]. As such, the purpose of the project was to identify similarities between and alternatives regarding different business processes at various organizations [23]. This resulted in an online available “process handbook” to exchange ideas regarding organizational practices ending up with a “repository” of knowledge about business processes and featuring more than 5900 entries in July 2002 [24].

In its very essence, the MIT Process Handbook structures its business process repository around two main dimensions: parts and types. Process *parts* represent the fact that a business process can be subdivided into several “sub” business processes or activities as for example the “sell product” business process is broken down into the more detailed processes like “identify potential customers”, “inform potential customers”, “obtain order”, “deliver product”, etc. Process *types* represent different alternatives or “specializations” of a generic activity, as for example the processes “sell by mail order” and “sell in retail store” can be considered as two specializations of the generic process “sell product”. Using these two dimensions to situate the different business processes then results in the so-called “*Process Compass*” where processes are depicted on the vertical axe according to their different parts / subactivities and on the horizontal axe according to their different types / alternatives [23].

Clearly, this way of working already implies a certain amount of modularity: in a top-down way, the general, more “high-level” activities are constantly broken down into more detailed constituent subactivities, representing the respective modular “building blocks”. In addition, the different types / alternatives available for certain processes suggest the possibility of being able to compose new processes in a kind of “plug and play” manner: for each subactivity, frequently some “equivalent” types are proposed, apparently allowing the designer of a new or ameliorated process to choose and trade-off between different alternatives or replace on a later time an existing activity by an alternative “version” of that subactivity.

In order to assess whether the Handbook can provide sufficient support regarding our attempt to extend our framework on modularity and evolvability to patterns on the business level, let us start for example on the overall activity of the Process Handbook (i.e., the most generic and high-level one, claimed as being the basis for most business processes): “produce as a business”. A schematic overview of this process is provided in Figure 1. One can notice that the business process “produce as a business” has five parts (i.e., design product and process, buy, make, sell and manage a business). The figure moreover is partially expanded for the parts “design product and process” and “sell”. When trying to leverage these business processes and parts to our constructional building blocks created at the software level in NS theory, three somewhat related issues arise: under-specification, lack of adherence to prescriptive design principles and an inherent top-down approach.

A. Underspecification

The repository of the Process Handbook regularly seems to be lacking highly specified and detailed descriptions of its different processes and activities. Frequently, some relevant subactivities seem to be omitted and the structure of decomposition already stops before one has attained a very fine-grained modular overview of all the needed, broken-down, activities which are required for exhaustively executing a more generic high-level process. Consider for example the subprocess “identify needs or requirements” as shown in Figure 1. This subprocess has been given the rather vague essential description that it is a process for “identifying the usability parameters of a resource that is managed in a flow dependency” and it is already situated at the bottom of the Process Compass. Therefore, no further subactivities are identified in the Process Handbook. However, one could reasonably argue that this process can still be refined into more fine-grained activities such as, e.g., conducting a market survey, analyzing preceding sales results, etc, which could then be detailed into even more specific subprocesses. As such, the process descriptions frequently leave considerable room for interpretation about the actual specific activities entailed in certain processes.

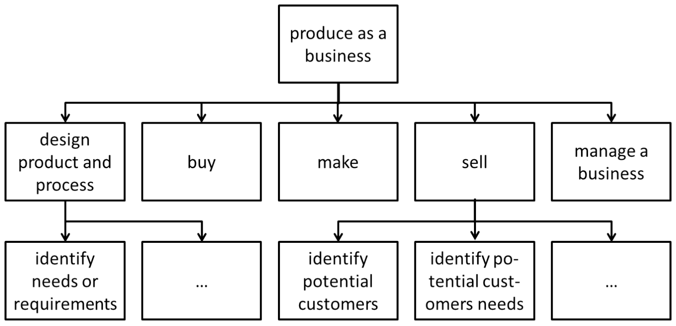
produce as a business
design product and process
buy
make
sell
manage a business
identify needs or requirements
...
identify potential customers
identify potential customers needs
...

Fig. 1. Partial breakdown of the “Produce as a business” business process based on the MIT Process Handbook

Indeed, it has not been the objective of the initiators of the Process Handbook to provide such a very fine-grained overview of each process. While Malone et al. mention in their introduction that their Handbook is expected to be useful in automatically generating software, they argue later on that as their main focus is to support human decision-makers “*there is no requirement that all our process descriptions be detailed or formalized enough to be executable by automated systems*” [23, p. 426]. Explicitly referring to Hammer and Champy’s [25] concept of analysis paralysis, they further claim that it is more important to be able to make a rapid assessment of the basic caracteristics of a process, rather than an elaborate and detailed overview. Proposing an approach to further fill in the Process Handbook, Pentland et al. [26, p. 3] also emphasize that it is “*pointless to spend a lot of energy mapping out [in a detailed way] how a particular activity is accomplished*”. However, in order to directly apply a NS approach, it is necessary to break down the action entities up to the point that it enables the identification of each individual concern which can potentially be considered as a separate “change driver” [1]–[3]. Also applying the NS principles at the business level requires the rather fine-grained identification of such individual change drivers related to individual elementary life cycle information objects, albeit that they can depend on time, context and subjective interpretation [4], [5].

B. Lack of adherence to prescriptive design principles

Also few or no applied prescriptive design guidelines can be retrieved towards the design of the process repository. Previous research regarding evolvable modularity at the software level [1]–[3] and the business level [5] however points out that some very stringent principles should consistently be adhered to with this respect. Consider for instance again the example previously outlined and depicted in Figure 1. “Identify needs or requirements” is a subprocess of “design product and process”, which is at its turn a part of the general process “produce as a business”. “Identify potential customer needs” is a subprocess of “sell” which is at its turn a part of the same general process “produce as a business”. While not completely clear from the descriptions delivered by the Handbook, one could argue that certain functionality is common or repeated within these two distinct processes (each having no ‘lower’ subprocesses / parts in the process compass). Specializing the “sell” process towards “sell via electronic store” has a subprocess “identify customer needs in electronic store”, which again seems to cover at least some common functionality, apart from the employed distribution channel. Surprisingly, the “identify customer needs in electronic store” process is subdivided into several other parts, not reused in the processes “identify needs or requirements” or “identify potential customer needs”. Similar instances of common functionality scattered around in various business processes were noticed regarding the delivery of products, advertising via diverse channels, etc. In terms of evolvability, this would imply that the need for e.g., changing something in the way customer needs are identified, could have a possible impact on all business processes involved with this functionality. This evidence suggests that at least one of the four basic principles of NS theory is not adhered to. The principle of Separation of Concerns namely enforces one to separate each individual change driver in its own distinct construct in order to encapsulate and limit the impact of a change driver in its own construct. More specifically, NS theory demands that during the design of evolve software or business processes, common parts of data or business processes should be kept in a non-redundant form, whereas each variation should be kept separated from the common part.

This lack of unambiguous and stringent prescriptive design guidelines to compose the Process Handbook is not only apparent when analyzing the respective processes, but also while studying the guidelines directed to the users of and contributors to the Handbook. For example, it is stated that: “*we believe that [. . . our structure] is comprehensive and intuitive [. . .] we have, in general, tried to maintain a branching factor of about “7 plus or minus 2” in the specialization hierarchy [. . . and use it] primarily as a rough guideline for editing the Process Handbook*” [24, p. 250]. Later on, one can read: “*wherever possible, we have tried to create groupings that constitute a mutually exclusive and exhaustive partitioning of the possible specializations of that activity*” [24, p. 250]. Finally, Malone et al. [23, p. 439] even explicitly mention that their Handbook is primarily a resource to suggest people what to do rather than proposing “prescriptive rules” in drafting the Handbook and are confident in the “*role of intelligent human “editors” to select, refine, and structure the knowledge represented in the Handbook to tackle the editioral challenge.*”

C. Top-down modeling

Finally, the systematic way of structuring the different business processes in a top-down fashion (i.e., starting from general high-level processes and then refining them into more detailed ones including some possible alternatives) inherently contributes to the functional–constructional gap as mentioned in Section I.

The breakdown of the “produce as a business” business process can illustrate this point. From a functional point of view, one tries to relate the input variables to the output variables through a transfer function. Knowing the transfer function, insight is gained in how the systems responds to various instances of input parameters from the environment. In this case, one tries to understand which resources (input variables) are required to deliver products or services (output variables) to the customers. However, such a transfer function is generally extremely complicated. Therefore, the technique of functional decomposition can be applied to reduce this complexity. Using functional decomposition, the transfer function is replaced by a set of sub-systems of which the transfer function is easier to understand. In the “produce as a business” business process, design, buy, make, sell, and manage are identified as sub-systems. Consequently, one now has to understand the inputs and outputs of, for example, the “sell” transfer function. However, these systems are still considered using a functional perspective: together, they describe in more detail how resources can be converted to products or services. On these functionally decomposed processes, one can again apply functional decomposition, resulting in very detailed descriptions of transfer functions. In traditional architecture literature, this process is referred to as *analysis* [27]. However, this activity is radically different from *designing* a structure which brings about this transfer function. When designing process elements, we aim to describe the structure to bring about the business process functionality. In design studies, this activity is referred to as *synthesis* [28]. Unsurprisingly, the design of a system which brings about the “produce as a business” function will be very complex. Analogously to functional decomposition, constructional decomposition can then be applied. However, the elements which are identified in a constructional decomposition are different in nature than the elements from a functional decomposition. Consequently, it does not make sense to try to relate the elements of a functional decomposition to the elements of a constructional decomposition. In other words, one cannot expect to arrive at essential constructional process building blocks by describing very detailed functionally decomposed elements, as proposed by the MIT Process Handbook.

Indeed, as Kodaganallur [10] seems to be noticing rightfully, few development methodologies seem to be seamlessly integrated with the use of patterns at the analysis level. However, using a systematic integrated bottom-up approach starting from data and action entities, encapsulating them in higher-order elements with proven stability and finally aggregating them into evolvable business process (patterns) is the inherent rationale in the NS theory. In our view, designing organizations towards (proven) evolvability requires such a bottom-up approach where in a first phase some very fine-grained building blocks exhibiting proven evolvability by adhering to the predefined principles (e.g., "send notification") are developed. Later on, these fine-grained building blocks can be reused in a safe and black-box way to construct more coarse-grained blocks, again conforming to the predefined design principles (e.g., "decide on customer creditworthness"). Only in a final phase these coarse-grained blocks could be aggregated towards such general processes as depicted in Figure 1 and enabling the transformation of, for instance, "sell product" to NS compatible patterns. Again, this suggests that the MIT Process Handbook is not directly applicable with respect to evolvable and modular constructional business process patterns.

These three issues are obviously highly related to one another and add up to the conclusion that the MIT Process Handbook primarily gives a *functional* overview of possible business processes in an enterprise. When designing new business processes or trying to ameliorate existing ones, the Process Handbook gives of an overview of some available options regarding the general functional elements such a business process should or could incorporate without giving any further compelling guidance on the way in which they should be aggregated or structured. Additionally, it still leaves (consciously) some room for interpretation regarding the specific activities and change drivers incorporated in each activity or business process. As such, the Process Handbook can (and maybe should) be considered as primarily a kind of reference model containing mainly domain knowledge on broad aspects affecting many enterprises: HR, supply chain management, marketing, etc. It can signify a considerable help and contribution when one is indeed looking for the functionalities common business processes have to perform. However, the Process Handbook does not adhere to specific modularity and evolvability guidelines and was essentially also not established with that main purpose in mind.

IV. The Need for Modular and Evolvable Constructional Business Process Design Patterns

In the previous section, we argued that the MIT Process Handbook can have considerable value regarding the functional analysis and decomposition of business processes, but that it can not be simply translated to existing NS software elements for several reasons. Without explicitly discussing several other existing and possibly relevant reference frameworks or claimed business process patterns, it seems reasonable to argue that some of the aforementioned arguments can be applied to multiple other existing frameworks as well: several of them indeed emphasize the modeling of best-practices and functional requirements in a top-down way. As such, a first important

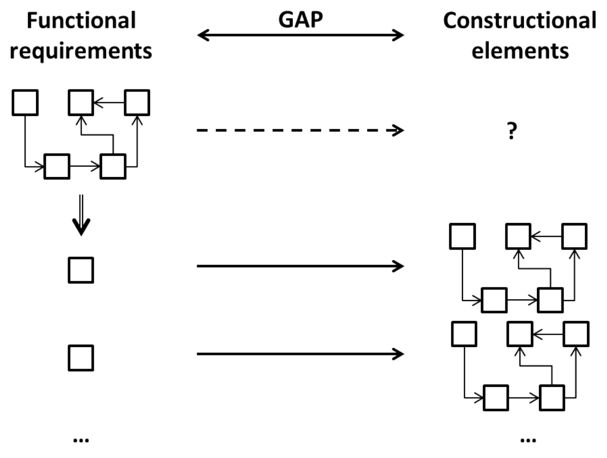

Fig. 2. A visualization of the functional–constructional gap and the need for modular and evolvable constructional business process design patterns

conclusion to be made is that this functional–constructive gap is again clearly been proven to be present and offers considerable challenges to unifying models on the business level with those on the software level. However, the existence of this gap is frequently underestimated or even not mentioned or addressed at all by many current methodologies. Obviously, the question then remains how to indeed develop modular and evolvable business process design patterns, focusing on constructive components instead of functional components.

Based on the NS theory rationale, Mannaert et al. [1] have already emphasized the existence of the functional–constructional gap and the importance of adhering to certain design principles when developing information systems. More specifically they suggest to consider the development of an information system as a *linear transformation* of a set of functional requirements (i.e., data entities, action entities and connectors) into a set of instantiations of software constructs (i.e., data structures and processing functions) at a certain point in time. Also, they show that this transformation can be quite straightforward when one is studying information systems from a static perspective. However, when focusing on the dynamic perspective (i.e., incorporating a marginal transformation of a set of additional functional requirements into a set of additional instantiations of software constructs) it is easy to show that the impact of a single 'extra' functional requirement is not necessarily limited to the addition or modification of a single software primitive. Stated otherwise, the impact of 1 functional change can have impact *n* on the constructional side, thus showing instability.

In order for this instability to be avoided, the NS rationale would require to first decompose the complex (high-level) functional requirements into a kind of more basic requirements. The next step would then be to look for Normalized linear transformations where — at least part of — the basic *functional* requirements can be deterministically transformed into *constructional* business process patterns, thus allowing a 1 to 1 mapping. Ideally, when representing this transformation in matrix form, the transformation matrix should indeed be of a diagonalized form or at least of a Jordan normal form. Finally, this reasoning is also visually depicted in Figure 2: instead of searching for business process patterns on the functional view, NS proposes to try to decompose these functional requirements to very basic ones and subsequently considering the transformation to constructive business process patterns.

These transformations should then again be analyzed from both a static and a dynamic perspective. Ideally, these transformations should be linear and normalized. This would entail from a static perspective that the realization of functional requirements, including the possible insufficiencies, could be located in a bounded and identifiable set of constructional primitives. From a dynamic perspective, this would entail that an increase in an existing functional requirement, or the addition of a functional requirement, would have a bounded impact on the constructional view.

V. Conclusion and Future Work

This paper investigated the extent to which currently available business process patterns can be applied in order to develop 'normalized' elements at the business process level, with the purpose to further extend the NS theory at the business level. Focusing our attention primarily on the MIT Process Handbook, three issues arose as hampering the application of this framework directly to NS elements: under-specification, lack of adherence to prescriptive design principles and an inherent top-down approach. Regarding the latter, it was noted that this aspect is strongly related to the concept of the so-called functional–constructional gap, in that lots of the existing pattern frameworks seem to provide *functional* decomposition. Therefore, they can be useful in managing domain knowledge and expertise, but are not directly transformable to *constructional* primitives at the NS level. Hence, the necessity for future identification of modular and evolvable constructional business process design patterns was called for and a NS theory based approach was proposed for studying the needed transformation between functional and constructional patterns in a dynamic context, emphasizing the need for evolvability.

A limitation of this paper is that it focused its attention into discussing only one existing framework in a rather detailed way. While we expect our proposed reasoning regarding the important, yet frequently underestimated functional–constructional gap to be applicable to many, if not most currently available pattern frameworks, some future research could then obviously investigate the extent to which our conclusions can also be applied to other existing frameworks.

Other related research at our research group will clearly be aimed at trying to find those necessary constructional business process elements. Previous research suggested Notification and Payment as potential business process design patterns, however further research and extension is definitely needed.

Acknowledgments

P.D.B. is supported by a Research Grant of the Agency for Innovation by Science and Technology in Flanders (IWT).

References

[1] H. Mannaert, J. Verelst, and K. Ven, "The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability," *Science of Computer Programming*, vol. 76, no. 12, pp. 1210–1222, 2010.
[2] ——, "Towards evolvable software architectures based on systems theoretic stability," *Software Practice and Experience*, vol. Early View, 2011.
[3] H. Mannaert and J. Verelst, *Normalized systems: re-creating information technology based on laws for software evolvability*. Koppa, 2009.
[4] D. Van Nuffel, H. Mannaert, C. De Backer, and J. Verelst, "Towards a deterministic business process modeling method based on normalized systems theory," *International Journal on Advances in Software*, vol. 3, no. 1-2, pp. 54–69, 2010.
[5] D. Van Nuffel, "Towards designing modular and evolvable business processes," Ph.D. dissertation, University of Antwerp, 2011.
[6] G. M. Weinberg, *An Introduction to General Systems Thinking*. Wiley-Interscience, 1975.
[7] L. Bertalanffy, *General Systems Theory: Foundations, Development, Applications*. New York: George Braziller, 1968.
[8] M. Bunge, *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems*. Boston: Reidel, 1979.
[9] E. R. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns – Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
[10] V. Kodaganallur and S. Shim, "Analysis patterns: A taxonomy and its implications," *Information Systems Management*, vol. 23, no. 3, pp. 52–61, SUM 2006.
[11] D. C. Hay, *Data Model Patterns: Conventions of Thought*. Dorset House, 1995.
[12] M. Fowler, *Analysis Patterns: Reusable Object Models*. Addison-Wesley Professional, 1996.
[13] L. Silverston, *The Data Model Resource Book: v.1: A Library of Universal Data Models for All Enterprises: Vol 1*. John Wiley & Sons, 2001.
[14] ——, *The Data Model Resource Book: A Library of Universal Data Models by Industry Types: v. 2*. John Wiley & Sons, 2001.
[15] C. Larman, *Applying UML and Patterns*. Prentice Hall PTR, 1997.
[16] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, pp. 5–51, 2003.
[17] A. Scheer, *Business Process Engineering – Reference Models for Industrial Enterprises*. Springer-Verlag, 1998.
[18] J. L. G. Dietz, *Enterprise Ontology: Theory and Methodology*. Springer, 2006.
[19] W. McCarthy, "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment," *The Accounting Review*, vol. 57, no. 3, pp. 554–578, 1982.
[20] P. Hruby, J. Kiehn, and C. Scheller, *Model-Driven Design Using Business Patterns*. Springer, 2006.
[21] M. Porter, *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, 1998.
[22] Supply Chain Council (SCC), "Supply Chain Operations Reference Model (SCOR): Version 10.0."
[23] T. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. Osborn, A. Bernstein, G. Herman, M. Klein, and E. O'Donnell, "Tools for inventing organizations: Toward a handbook of organizational processes," *Management Science*, vol. 45, no. 3, pp. 425–443, MAR 1999.
[24] T. Malone, K. Crowston, and G. Herman, *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press, 2003.
[25] M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperCollins, 1993.
[26] B. Pentland, C. Osborn, G. Wyner, and F. Luconi, "Useful descriptions of organizational processes: Collecting data for the process handbook," August 1999, Unpublished Working Paper. Center for Coordination Science, MIT, Cambridge, MA.
[27] C. Alexander, *Notes on the Synthesis of Form*. Harvard University Press, 1964, iSBN: 0674627512.
[28] J. S. Gero and U. Kannengiesser, "The situated function-behaviour-structure framework," *Design Studies*, vol. 25, no. 4, pp. 373–391, 2004.