Formalizing Organization Implementation

Marien R. Krouwel[1,2(✉)], Martin Op 't Land[1,3], and Tyron Offerman[1,4]

[1] Capgemini Netherlands, P.O. Box 2575, 3500 GN Utrecht, The Netherlands
{Marien.Krouwel,Martin.OptLand,Tyron.Offerman}@capgemini.com
[2] Radboud Universiteit, Comeniuslaan 4, 6525 HP Nijmegen, The Netherlands
[3] Antwerp Management School, Sint-Jacobsmarkt 9-13, 2000 Antwerp, Belgium
[4] LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract. Our research program aims at finding building blocks that are able to deal quickly with the constant change that organizations face. In order to do so, a deeper understanding of possible organization implementation variants is necessary, as well as the implications on the operation and IT support of organizations. In earlier research, we have composed a list of Organization Implementation Variables to informedly decide upon organization implementation, enabling traceability in governing enterprise and IT transformations. This list has been validated and extended by four practical case studies and has been formalized afterwards and validated by prototyping. In this paper the resulting framework is presented which (*a*) is broader and more detailed than before, (*b*) has a sound theoretical basis, and (*c*) contains precise and validated definitions of the variables itself. This paper shows that the framework is not only suitable for organization modeling, but also has possibilities for designing software in which implementation choices can be made explicit and variable. This paper also provides insights in the implications of implementation choices on the operation of an organization.

Keywords: DEMO · Enterprise engineering · Organization implementation · Agile

1 Introduction

As strategic and operating conditions become increasingly turbulent due to factors such as hyper-competition, increasing demands from customers, regulatory changes, and technological advancements, the ability to change, often referred to as 'agility' [1], becomes an important determinant of firm success [2]. Though change occurs in organizational essence, such as products and services delivered, most of the time change deals with different implementations [3]. Our research program [4] aims at finding building blocks that are able to deal quickly with the constant change that organizations face. In order to do so, a deeper understanding of possible organization implementation variants is necessary, as well as the implications for the operation and IT support of organizations.

D. Aveiro et al. (Eds.): EEWC 2016, LNBIP 252, pp. 3–18, 2016.
DOI: 10.1007/978-3-319-39567-8_1

For the process level, several approaches have been proposed for describing different process variants, e.g., [5,6]. Variants of the same process describe different implementations of the same process, differing by e.g. product type or location. These variants may either coexist or exist sequentially. These dimensions for variance can be seen as types of change, each being variable in the implementation of the processes. Others have classified different types and dimensions of process change, e.g. [7,8]. However, these approaches are restricted to the process area, whereas change does not only occur on the process level but also in the organizational structure and the (supporting) means that are available. Also, the implications for the operation and IT support are not described.

For variability in IT, the Normalized Systems theory [9] describes a list of anticipated types of changes and proposes a set of building blocks that can deal with these types of change in order to avoid combinatorial effects in which change becomes harder and harder over time. This list, however, is on a very technical level and not in terms of typical organizational changes. Instead, we are looking for a list of anticipated types of change on the organizational level, including a way in which organizational changes are transparently translated into IT changes, enabling traceability in governing enterprise and IT transformations.

Other researchers have tried to bridge the gap between organization model and IT, while leaving room for different implementation variants, making the implementation variable in some dimensions, e.g. [10–12]. However, only de Jong [12] provides a framework to systematically detail some of the design decisions that are needed to specify an Enterprise Information System. Also, none of the authors make explicit how their organizational models are used to design software; traceablity or completeness of design choices is not possible.

As an alternative, in 2013, Op't Land and Krouwel composed a list of Organization Implementation Variables (OIVs) [13], based on literature and structured according to the Enterprise Engineering Framework (EEF) [14]. The EEF has the same theoretical basis as the Design and Engineering Methodology for Organizations (DEMO) [15] which has shown to offer a quick way for finding the essence of an organization as starting point for identifying local differences [16–18]. Since then, this list of OIVs has been validated and extended by four practical case studies [19–22]. It was noticed by many reviewers that the goals and concepts of this framework were not clear. Also it was concluded that the implications towards operation and IT support should be made more clear in order to avoid implicit design choices.

Within the goals of the research program, in this paper the OIV framework is presented: its goals, requirements, concepts, as well as some example contents and its implications on the operation and IT support of organizations. With this formalized framework, it becomes easier to validate whether the framework really meets its goals. One of these goals is validated by prototyping.

The remainder of this paper is structured as follows. In Sect. 2, we will outline some terms used in this paper. Next, in Sect. 3 the research approach is presented and in Sect. 4 the resulting framework is presented, including some examples. Section 5 provides the conclusions, including the implications on operation and IT, and provides directions for future research.

2 Way of Thinking

Weinberg and Dietz discern two distinct perspectives on any system: function and construction [23,24]. The functional perspective, or black-box model clarifies the behavior of the system in terms of (functional) relationships between input and output of the system. The constructional perspective, or white-box model clarifies the internal construction and operation of the system in terms of collaboration between its elements to deliver products to its environment. The highest level white-box model of a system, completely independent of the way in which it is realized and implemented, is called its *ontology*. The lowest level, most detailed white-box model of a system is called its *implementation model*. By *implementation* is understood the assignment of technological means to the elements in the ontological and implementation model, so that the system can be put into operation. By *technology* we understand the means by which a system is implemented. For organizations, a wide range of technologies is available, including human beings and organizational entities, ICT artifacts (e.g., phone, email, computer programs) and mechanical means.

DEMO is a methodology for the design, engineering, and implementation of organizations [15]. As the highest level white-box model of an enterprise – a goal-oriented cooperative – DEMO defines the *enterprise ontology*: the essence of an organization, fully independent from the way in which it is realized and implemented. The organization of an enterprise is a heterogeneous system, constituted as the layered integration of three aspect systems, namely the Business (B) system, the Informational (I) system and the Documental (D) system [15, p. 115]. The production of these systems concern (B) original acts (material and immaterial), such as deciding, judging and creating, (I) informational acts, such as remembering, recalling and computing and (D) documental acts, such as storing, retrieving, transmitting and copying. The ontology of any organization (B, I or D) can be expressed in a DEMO model which consists of four aspect models:

Construction Model (CM) represents the composition, environment and structure of the organization and consists of transaction kinds, associated (initiating and executing) actor roles, and information banks including the links between these banks and actor roles;
Process Model (PM) details each transaction kind according to the transaction axiom and makes explicit the waiting links between coordination acts;
Action Model(AM) specifies for every agendum kind the action rules to be applied by the actor roles;
Fact Model (FM) contains entity types with their property types, product kinds and their relationships (fact types).

DEMO is grounded in a set of theories, among which are the FI and MU theory. FI [25] is a philosophical theory about knowledge in general and clarifies the notion of (factual) knowledge and information. It also explains how factual knowledge is created from perceptions of concrete things, directed by (fact) types, which operate as conceptual sieves. MU [26] is a theory of models and modeling in general, and of conceptual modeling in particular. It also

presents General Ontology Specification Language (GOSL), a universal language for specifying conceptual complexes, conceptual schemas and meta schemas. As we are building a framework to *model facts* about the implementation of an organization, both theories provide a sound basis to build the framework on.

3 Way of Working

The goal of this research is to design a framework for the understanding and modeling of organization implementation, based on sound theories and validated in practice. As the result is an artifact that needs to be designed, in this research we adopt the design science methodology [27] as main methodology. Where behavioral science seeks to develop and justify theories that explain or predict phenomena related to the identified business need, design science seeks to construct and evaluate artifacts designed to meet the identified business need [28]. However, as Hevner states, these methodologies cannot be separated and should be used complementary [29]. Because design is inherently an iterative and incremental activity, Hevner suggests three cycles for Design Science Research [30] which can be applied in as many iterations as needed (Fig. 1):

- the *relevance cycle* provides the requirements for the research and determines whether the resulting artifact improves the environment;
- the *rigor cycle* provides past knowledge to the project and ensures new contributions are added to the knowledge base;
- the *design cycle* is where the artifact is constructed and evaluated.

In this paper, the result of several iterations is presented. Hevner suggests a checklist for design science research [31]. We will use that checklist (Table 1) to assess progress on this design research project. In Sect. 4 we will discuss the answers to these questions.

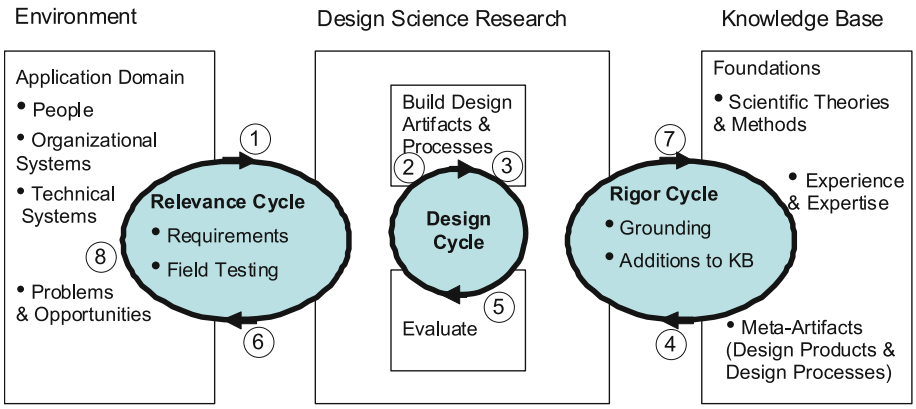

Fig. 1. Design Science Research Cycles [30], including references to the questions of the checklist (Table 1)

Table 1. Checklist for design science research [31] and section in which the question will be addressed

1	What is the research question (design requirements)?	Subsect. 4.1
2	What is the artifact? How is the artifact represented?	Subsect. 4.3
3	What design processes (search heuristics) will be used to build the artifact?	Subsect. 4.4
4	How are the artifact and the design processes grounded by the knowledge base? What, if any, theories support the artifact design and the design process?	Subsect. 4.2
5	What evaluations are performed during the internal design cycles? What design improvements are identified during each design cycle?	Subsect. 4.4
6	How is the artifact introduced into the application environment and how is it field tested? What metrics are used to demonstrate artifact utility and improvement over previous artifacts?	Subsect. 4.4
7	What new knowledge is added to the knowledge base and in what form (e.g., peer-reviewed literature, meta-artifacts, new theory, new method)?	Subsect. 4.5
8	Has the research question been satisfactorily addressed?	Subsect. 4.5

4 Result

In this section the result of several iterations of the OIV framework development is presented, by answering the questions of Hevners checklist (Table 1).

4.1 Goals and Requirements

Answering question 1, the goal of the framework is to gain insight in the implementation of an organization (either B, I or D) in order to

(a) decide informedly upon organizational changes;
(b) enable traceability and completeness in governing enterprise and IT transformations;
(c) assess to what extent IT platforms (can) support organizational implementation variability;
(d) design IT that inherently supports typical organizational changes; and, ultimately
(e) design organizations (or, better, implementations) that support typical organizational changes.

Moreover, the ontology plus implementation choices should capture all design decisions that need to be taken to come to a complete implementation model that can be put into operation. Note that the framework only concerns the 'hard' aspects that really can be chosen and changed on the short term and not the more 'soft' aspects that cannot really be changed or are very hard to change, like culture, beliefs, (shared) values or (management) style [32,33].

4.2 Use of Existing Theories

Answering question 4, the framework is based on existing theories in the field of Enterprise Engineering as outlined in Sect. 2. More specifically, it is built on top of the Enterprise Engineering Framework [14], fed by earlier literature research [13], and structured using DEMO [15] and the FI [25] and MU [26] theories. For the construction and evaluation, design science theories [27,28] are used, as explained in Sect. 3. The resulting framework and its concepts are presented in the next sections.

4.3 Concepts

In this section, the concepts of the framework are explained (question 2).

Organization Implementation. In the framework, by organization implementation is meant all design decisions that are taken to create the implementation model, the lowest level and most detailed white-box model, of an organization, including the assignment of technological means to the elements in the implementation model, so that the system can be put into operation.

Variable. In mathematics, a variable is a placeholder of some element in some set. In general, a variable is any entity that can take on different values.

Organization Implementation Variable. Following the mathematical definition of a variable, we will define an Organization Implementation Variable (OIV) as a placeholder for an element in the set of possible organization implementation design choices in some category. Thus, an Organization Implementation Variable describes the design freedom or restrictions in some organization implementation design category. For example, the functionary types, organizational units, work locations and authorizations, need to be decided upon before an organization can become operational. Moreover, they may be subject to change, and thus are (a) variable in the implementation of an organization. Note that an OIV is not a target variable or KPI, nor does it contain the principles that guide the decisions or the rationale behind it – an OIV belongs to the *construction* of an organization, while target variables and KPI's belong to the *function* of an organization; the principles belong to the *context* or environment in EEF terms.

Two kinds of OIVs can be distinguished: *elementary* and *cross reference* OIVs. An elementary OIV is an OIV that is not dependent on the existence of some other OIV or element in the ontology, e.g. functionary type. On the other hand, a cross reference OIV is an OIV of which the existence depends on the existence of some other OIV or element in the ontology, e.g. authorization. Therefore, a cross reference variable can be compared to a weak entity type in ER modeling [34]. More example OIVs can be found in Subsect. 4.6.

OIV Values. Note that there are two ways to express the value(s) of any variable:

1. by expressing the specific value or set of values by enumerating the/each value, and
2. by setting constraints, implicitly defining the value(s) of the variable.

For elementary OIVs, the first way is recommended as its value can often easily be expressed in an amount or entity. For cross reference OIVs, declaring constraints makes the resulting set more flexible, allowing for future values without having to enumerate every value explicitly.

The assignment of a value to a OIV can be modeled as a set of transaction kinds; the process of implementation design itself can be modeled in a DEMO Construction Model. The choices itself can be considered facts which are the results of transactions. They can be modeled in a DEMO Fact Model. Note that a cross reference OIV can be discerned in the FM by the presence of a *mandatory role constraint* [35, p. 315], denoted by a black dot (see legend in Fig. 2).

Layers. The EEF contains the layers *Parties and People* and *ICT and other means* while Dietz and Hoogervorst propose three categories for the implementation: implementation, installation, and operation [36, p. 43][1]. However, the meaning of the layers and categories was not fully spelled out, and they do not fully encompass the sourcing process as well as the assignment and scheduling of (human) resources. Therefore we propose the following layers or categories in the OIV framework.

Implementation contains the (non-ontological) structure of the organization such as functionary types, organizational units, work locations as well as the relations between them and with the ontological elements (mainly agendum type);
Means contains all technological means, including human beings and ICT artifacts – also known as silicon and carbon servers [37] – as introduced in Sect. 2, needed to operate;
Installation contains the (temporary or more durable) assignment of specific means to elements in the implementation;
Operation contains the assignment of specific agenda to specific means.

4.4 Construction and Evaluation Process

Answering question 3, in the design process, every OIV is defined as an entity type in a DEMO Fact Model, including definitions and examples from the EU-Rent case [38]. The example instances are used for validation by population [39,40][2]. For every entity type, its producing transaction kind is identified and

[1] Note that it might be confusing that implementation itself is a category within the broader meaning of implementation. For the rest of this section, implementation is meant in the narrow definition.

[2] The term 'population' is used instead of 'instantation' as instantiation may imply that one instance is enough, where population implies multiple instances should be used for validation.

modeled in a DEMO Construction Model. In order to enable iterations in the construction and evaluation process (questions 3 and 5), the implementations of four large European public organizations and one academic case have been modeled (for details, see Table 2), by assigning values to each of the (applicable) OIVs. This may also count as field application (question 6).

Table 2. Details of case studies: organization, research question, approach and results

Rijkswaterstaat (RWS) [19]
Question To what extent can OIVs be found in the documentation of RWS?
Approach Analysis of IVS90, the national supporting Information and Monitoring System of the (main) waterways and the Maritime in the Netherlands
Results Clearer definitions; observation instructions; 1 new OIV: Region
Jeugdzorg Nederland [20]
Question Which of the proposed OIVs can be identified in an enterprise?
Approach Analysis of documentation and interviews regarding the implementation of Jeugdzorg and their recently built case management system (WIJZ)
Results Clearer definitions; 1 new OIV: Region
European parking law enforcer [21]
Question How to build a simulation model based on a DEMO model and OIVs?
Approach Proposed method is applied to two cases: one fictional and one real
Results Clearer definitions; 2 new OIVs: agenda cluster, X-ref
Dutch municipal subsidy providers [22]
Question To what extent does the Capgemini MultiSubsidy application support the different implementations of Dutch subsidy providers?
Approach Analysis of documentation and interviews regarding the implementation of Dutch municipal subsidy providers and the MultiSubsidy application
Results Clearer definitions; no new OIVs
EU-rent [41]
Question How to construct a model to assess the support of OIVs by IT?
Approach Construction of DEMO CM and FM of organization implementation, validation by population with examples from EU-Rent case
Results Clearer definitions; CM and FM of implementation; no new OIVs

Additionally, a prototype has been built, based on the DEMO Fact Model (FM) as presented in Subsect. 4.6, in which the implementation, including means, installation and operation, can be configured on top of the identified transaction kinds. This helped in defining the OIVs to the level where they can be instantiated. Also, it helped gain insight in the impact on the operation. The result is a fully functional prototype in which transactions can be started and agenda are routed to authorized persons who can deal with the agenda, creating new agenda, while completely adhering to the organization implementation choices. In this prototype, no

software (programming code) needs to be changed when changing the organizational implementation.

4.5 Additions to Knowledge Base and Practice

Answering question 7, the framework provides a more detailed insight in what organization implementation entails than earlier research does. Also, a set of Organization Implementation Variables is provided, including a method to assign a value to each OIV. This forms the foundations for a sound theory regarding organization implementation.

Answering question 8, it can be concluded from the five cases that the framework provides insight in the implementation of an organization. More specifically, we will elicit to what extent the goals of the framework are met.

(a) **decide informedly upon organizational changes:** this goal is met as a direct consequence of a detailed insight in the implementation of an organization. Although only in the case of the European parking law enforcer organizational change was proposed, the other cases show that it is possible to provide insight in the proposed change, as well as its consequences.
(b) **enable traceability and completeness in governing enterprise and IT transformations:** this goal is almost met; the detailed insight in the implementation of an organization enables traceability. Completeness is hard to claim, but the cases have only brought three new OIVs with respect to the original set of OIVs [13]. It is expected this set will not grow significantly from new case studies.
(c) **assess to what extent IT platforms (can) support organizational implementation variability:** this goal is met as confirmed by the Jeugdzorg and Dutch municipal subsidy case. However, more research is required to come to a complete method for such an IT agility assessment.
(d) **design IT that inherently supports typical organizational changes:** this goal is partly met as a first prototype is built. More research in this area will be needed in order to be able to support *all relevant* OIVs, which requires that it is clear for each OIV whether it is possible, relevant and necessary to support it in IT.
(e) **design organizations (or, better, implementations) that support typical organizational changes:** this goal is not yet met as the framework is not yet used to design implementations without combinatorial effects, i.e., such that change does not become harder over time.

4.6 Examples

In this section some example Organization Implementation Variable are outlined (Table 3), including some definitions and example instances (Tables 4, 5, 6, 7, 8 and 9) as well as a DEMO FM (Fig. 2) of it. This paper does not attempt to be complete as it is impossible to provide definitions for all 25 variables within the restriction of 15 pages.

Table 3. Example organization implementation variables

Category	Example OIVs	
	Elementary	*Cross reference*
Implementation	Competence	Competence requirement
	Functionary type	Logical unit of work
	Organizational unit	Task competence
	Work location	Authorization
	Addressee	Event location restriction
	Juristic person	Order of working
Means	Human resource	Competence validation
	ICT artifacts	
	Mechanical means	
Installation		Installation
Operation	Incidental delegation	Addressee specification

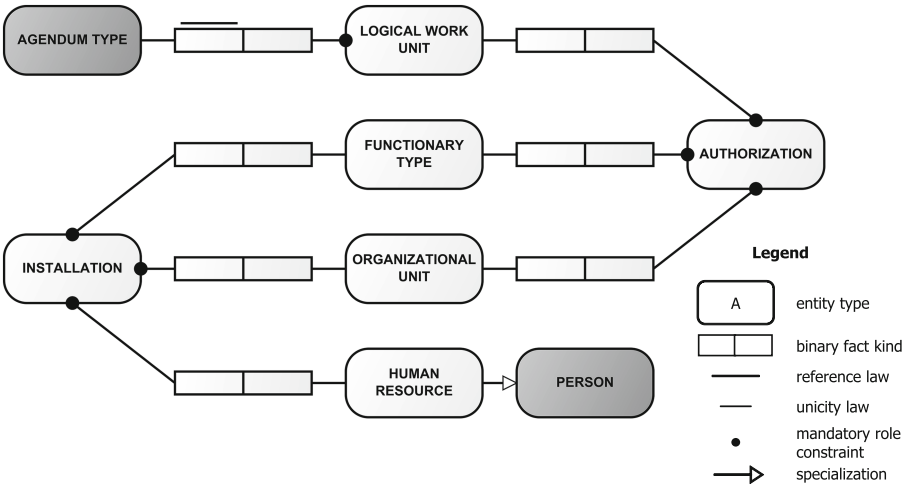

Fig. 2. (part of) DEMO FM for implementation design, in ORM notation [40]

Table 4. Functionary type

Definition	A functionary type is a call sign intended for the assignment of agendum types
Example(s)	1. Desk officer 2. Distributor

Table 5. Organizational unit

Definition	An organizational unit is a named element or segment of an organization, possibly with an hierarchical relation to another organizational unit.
Example(s)	1. Sales 2. Logistics 3. Distribution, placed under Logistics 4. Transportation, placed under Logistics

Table 6. Logical unit of work (LUW)

Definition	A logical unit of work is then union of agendum types of which instances are usually dealt with by a single person as being an inseparable unit of work. Note that it is often separable, therefore the term logical is used.
Example(s)	1. T1/pm, T1/ex and T1/st are combined in LWU 'T1 dealing'

Table 7. Authorization

Definition	An authorization is the assignment of a functionary type to a (set of) LWU(s), a(n) (set of) organizational unit(s) and a (set of) work location(s), with some responsibility (e.g. cf. RACI [42]). Note that authorization includes structural delegation, i.e., it might be possible that one functionary type is Responsible for the work, while another functionary type is Accountable.
Example(s)	1. Desk officer is assigned to deal with T01/rq (Accountable and Responsible), in the sales unit at ABC Street 123, Leiden 2. Distributor is assigned to deal with T07/rq (Accountable and Responsible), in the distribution unit at Air Lane 23, Amsterdam

Table 8. Human resource

Definition	A human resource is a natural person who works under a contract of employ or hire agreement. Note that this includes volunteers.
Example(s)	1. Jane is employed at RAC. 2. Chiara is employed at RAC. 3. Anthony is hired externally at RAC

Table 9. Installation

Definition	An installation is the assignment of a mean, either a human resource or a technological or mechanical mean, to a (set of) functionary type(s), a(n) (set of) organizational unit(s) and a (set of) work locations.
Example(s)	1. Jane is assigned to desk officer in the sales department at ABCstreet 123, Leiden, Netherlands. 2. Anthony is assigned distributor in the distribution department at Air lane 23, Amsterdam, Netherlands

Operation. For the operation layer, additional concepts were identified, that could not be defined as an OIV. For instance, an organization might define *info@company.com* as its general email address. Any agendum that is received on this email address needs to be routed to the right person. Often this is done by some secretary or automated system, for which the actor role *dispatcher* can be identified. This actor role needs knowledge about the possible addressees and the rules for routing general requests to a person in the organization who can deal with such a request. Note that the same reasoning holds for phone numbers and other communication channels, as well as for other agenda types that come from outside the organizations. In conclusion, additional actor roles, necessary for the implementation, installation and operation of an organization, can be identified and should be modeled in a (generic) DEMO Construction Model.

5 Conclusions and Future Research

The goal of this paper was to present a framework for the understanding and modeling of possible organization implementation variants, as well as to understand the implications on the operation and IT support of organizations. Using Hevners checklist, as part of looping through the three cycles of design science, we have elaborated the process, progress and preliminary results of our research;

1. the framework has as goal to gain insight in the implementation of organizations, in order to informedly decide upon organization implementation, enabling traceability in governing enterprise and IT transformations, as well as to design more agile organizations and IT;
2. the framework is built on top of several existing EE theories, fed by literature research, while DEMO is used to model the OIVs in a Fact Model, giving the framework a sound theoretical basis;
3. the framework has been evaluated against and extended by four practical case studies and one academic case study, resulting in additional OIVs and more precise definitions;
4. a prototype has been built in which the OIVs can be instantiated and its effect on the operation and IT support can be studied.

The five case studies show that the framework can indeed be used for organization implementation modeling. The prototype shows that it is possible to design software in which implementation choices can be made explicit, while allowing for future change. However, reflecting on the goals of the framework:

- The case studies are limited to public organizations. Case studies at commercial organizations should be performed in order to investigate whether they reveal new OIVs.
- Although the framework already enables assessing whether software (can) support the OIVs, it is worthwhile to investigate whether a thorough method for the assessment of IT agility can be designed.
- More research is needed to investigate whether it is possible, relevant and worthwhile to support all OIVs in software, preferably in a way that does not create combinatorial effects and consequently allowing future change without growing efforts.
- Also, more research is needed to investigate whether agile organizations without or with little combinatorial effects can be designed, enabling the agile enterprise, based on this framework.

The original framework [13] contained a layer called *ICT and other means*. In subsequent work, OIVs in this layer did not receive much attention. In fact, they are now aggregated in just two OIVs in the *Means* layer. It might be worthwhile to put more effort in finding detailed design choices in this layer, e.g. by looking at methods and modeling languages that are more focused on this layer, such as ArchiMate [43,44].

As shown in Subsect. 4.6, research in the operation layer shows that there might be some generic actor roles that should be implemented in every organization. Extending the prototype might be helpful to find all OIVs and other concepts in the operation layer. Moreover, a generic DEMO Construction Model could be made of the concepts in that layer, or maybe even for every process in the organization that is concerned with designing the organizations ontology, implementation and operation.

The prototype has not yet incorporated revocation patterns [45]. Though other researchers have put effort in translating the transaction pattern, including revocations, to software [11,46], additional research is required to investigate the impact of organization implementation (variables) on revocation, especially when implemented in software. Additionally, the topic of time outs has to be addressed, both in organization and ICT.

References

1. Oosterhout, M.P.A.: Business agility and information technology in service organizations. Ph.D. thesis, Erasmus University Rotterdam, June 2010
2. Overby, E., Bharadwaj, A., Sambamurthy, V.: Enterprise agility and the enabling role of information technology. Eur. J. Inf. Syst. **15**, 120–131 (2006)

3. Dietz, J.L.G., Hoogervorst, J.A.P.: Enterprise ontology and enterprise architecture - how to let them evolve into effective complementary notions. GEAO J. Enterp. Architect. **2007**, 1 (2007)
4. Krouwel, M.: Towards designing modular structures for reducing non-linear effects of organizational change: research proposal. http://www.ciaonetwork.org/uploads/eewc2013/presentations/doctoral_consortium/1_krouwel.ppt. Accessed 3 April 2016
5. Lind, M., Goldkuhl, G.: Designing business process variants – using the BAT framework as a pragmatic lens. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 408–420. Springer, Heidelberg (2006)
6. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the provop approach. J. Softw. Maintenance Evol. Res. Pract. **22**(6–7), 519–546 (2010)
7. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of flexibility in business processes. In: Regev, G., Soffer, P., Schmidt, R. (eds.) BPMDS, CEUR Workshop Proceedings, vol. 236 (2006). CEUR-WS.org
8. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.P.: Towards a taxonomy of process flexibility. In: Bellahsne, Z., Woo, C., Hunt, E., Franch, X., Coletta, R. (eds.) CAiSE Forum, CEUR Workshop Proceedings, vol. 344, pp. 81–84 (2008). www.CEUR-WS.org
9. Mannaert, H., Verelst, J.: Normalized Systems: Re-creating Information Technology Based on Laws for Software Evolvability. Koppa, Kermt, Belgium (2009)
10. Terlouw, L.: Modularization and specification of service-oriented systems. Ph.D. thesis, TU Delft (2011)
11. van Kervel, S.: Ontology driven enterprise information systems engineering. Ph.D. thesis, TU Delft (2012)
12. de Jong, J.: A method for enterprise ontology based design of for enterprise information systems. Ph.D. thesis, TU Delft (2013)
13. Op 't Land, M., Krouwel, M.: Exploring organizational implementation fundamentals. In: Proper, H.A., Aveiro, D., Gaaloul, K. (eds.) EEWC 2013. LNBIP, vol. 146, pp. 28–42. Springer, Heidelberg (2013)
14. Op 't Land, M., Proper, H.A.: Impact of principles on enterprise engineering. In: Österle, H., Schelp, J., Winter, R. (eds.) Proceedings of the 15th European Conference on Information Systems, pp. 1965–1976 (2007)
15. Dietz, J.L.G.: Enterprise Ontology - Theory and methodology. Springer, Heidelberg (2006)
16. Mulder, J.B.F.: Rapid enterprise design. Ph.D. thesis, TU Delft (2006)
17. Op 't Land, M., Zwitzer, H., Ensink, P., Lebel, Q.: Towards a fast enterprise ontology based method for post merger integration. In: Proceedings of the 2009 ACM symposium on Applied Computing (SAC-ACM2009), pp. 245–252. ACM (2009)
18. Krouwel, M.R., Op 't Land, M.: Using enterprise ontology as a basis for requirements for cross-organizationally usable applications. In: Figueiredo, A.D., Ramos, I., Trauth, E. (eds.) Proceedings of the 7^{th} Mediterranean Conference on Information Systems 2012 (MCIS2012), MCIS Proceedings, University of Minho, Portugal, AIS Electronic Library (AISeL) Paper 23(2012)
19. Molly, S.: Exploring organizational implementation fundamentals in a real enterprise. Master's thesis, Antwerp Management School (2014)
20. van Bockhooven, S., Op 't Land, M.: Organization implementation fundamentals: a case study validation in the youthcare sector. In: Complementary Proceedings of the Workshops TEE, CoBI, and XOC-BPM at IEEE-COBI 2015 (2015)

21. de Laat, L., Op 't Land, M., Krouwel, M.: Supporting goal-oriented organizational implementation - combining DEMO and process simulation in a practice-tested method. In: Aveiro, D.G.D. (ed.) EEWC 2016. LNBIP, vol. 252, pp. 19–33. Springer, Heidelberg (2016)
22. Krouwel, M.R., Huysmans, P.: Observing organization implementation variables in practice: a case study on dutch municipal subsidy agents. Forthcoming
23. Weinberg, G.M.: An Introduction to General Systems Thinking. Wiley-Interscience, New York (1975)
24. Dietz, J.L.G.: Enterprise engineering - the manifesto. http://www.ciaonetwork.org/publications/EEManifesto.pdf. Accessed 3 April 2016
25. Dietz, J.L.: The FI theory - understanding information and factual knowledge. Technical report, Czech Technical University in Prague, Delft University of Technology version 2.3 (2015)
26. Dietz, J.L.: The MU theory - understanding models and modelling. Technical report, Czech Technical University in Prague, Delft University of Technology, Antwerp Management School version 1.2 (2015)
27. Simon, H.A.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge (1996)
28. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decis. Support Syst. **15**(4), 251–266 (1995)
29. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
30. Hevner, A.R.: A three cycle view of design science research. Scand. J. Inf. Syst. **19**(2), 87–92 (2007)
31. Hevner, A., Chatterjee, S.: Design Research in Information Systems: Theory and Practice, 1st edn. Springer Publishing Company, Heidelberg (2010)
32. Hayes, J.: The Theory and Practice of Change Management, 4th edn. Palgrave Macmillan, Basingstoke (2014)
33. Schein, E.H.: Organization Culture and Leadership, 3rd edn. Wiley, Hoboken (2006)
34. Chen, P.P.: The Entity Relationship Approach to Logical Database Design. Q.E.D. Information Sciences, Wellesley (1977)
35. Halpin, T.: Information Modeling and Relational Databases, 1st edn. Academic Press, Cambridge (2001)
36. Dietz, J.L.G., Hoogervorst, J.A.: EE theories Overview. Technical report, Czech Technical University in Prague (2015)
37. Tribolet, J.: An engineering approach to natural enterprise dynamics: From top-down purposeful systemic steering to bottom-up adaptive guidance control. In: 2014 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), p. 1, April 2014
38. Object Management Group: Business Motivation Model (BMM) Specification, V1.1. OMG Available Specification OMG Document Number: formal/2010-05-01, Object Management Group http://www.omg.org/spec/BMM/1.1/PDF/. Accessed 3 April 2016
39. Dietz, J.L., Halpin, T.: Using DEMO and ORM in concert - a case study. In: Siau, K. (ed.) Advanced topics in database research, vol. 3, pp. 218–236. Idea Group Publishing, Hershey (2004)
40. Halpin, T.: Object-role modeling version 2. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 1941–1946. Springer, Heidelberg (2009)
41. Offerman, T.: Improving IT Supported organizational change; formalizing organizational implementation fundamentals. Master's thesis, Universiteit Leiden (2014)

42. Mike, J.M., Keller, P.J.: Business Process Mapping: Improving Customer Satisfaction. Wiley, New York (2009)
43. Open Group: Archimate 2.1 Specification. Van Haren Publishing, Zaltbommel (2013)
44. Ettema, R., Dietz, J.L.G.: ArchiMate and DEMO – mates to date? In: Albani, A., Barjis, J., Dietz, J.L.G. (eds.) CIAO! 2009. LNBIP, vol. 34, pp. 172–186. Springer, Heidelberg (2009)
45. Dietz, J.L.: The DELTA theory - understanding systems ontology. Technical report, Czech Technical University in Prague, Delft University of Technology, Antwerp Management School version 3.0 (2015)
46. Gouveia, D., Aveiro, D.: Two Protocols for DEMO Engines: PSI or Tell&Agree. CIAO DC 2015 (2015)