Towards Framing the Continuous Information Systems Engineering

Marite Kirikova

Department of Artificial Intelligence and Systems Engineering, Riga Technical University, Latvia
marite.kirikova@rtu.lv

Abstract. Necessity of continuous information systems engineering has been understood already decades ago. However, up to know there are no clear guidelines on the main constituents that must be present in frameworks that are used in continuous information systems engineering settings. The FREEDOM framework that roots in ideas of Viable Systems Model is one of the candidate frameworks for continuous information systems engineering. By comparing the FREEDOM framework to two other candidate approaches, some essential features for frameworks that can address the continuous information systems engineering peculiarities can be derived.

Keywords: Continuous information systems engineering, solution engineering, design, continuous software engineering, System of Systems, V-model.

1 Introduction

In 1999, Herbert Weber in his extended abstract "Continuous Engineering of Information and Communication Infrastructures" outlined several continuous engineering problems that shall be solved [1]:

- The analysis of existing (legacy) software and documentation of its results
- The (re)integration of existing (legacy) systems into information and communication infrastructures
- The conversion and transformation of existing (legacy) systems into renewed information and communication infrastructure

During the last two decades all of these problems have been extensively addressed. For instance, different approaches of legacy software analysis have been applied in information systems reengineering [2,3], continuous integration has become one of the well known terms of researchers and practitioners [4], and many conversion and transformation approaches have been thought out [5]. However, the continuous engineering challenge still remains. One of the reasons of complexity of the task are dependencies between different objects of interest in the continuous systems change process, which in [1] are described as structures around four categories of invariants of different life times: Category 1 (indefinite life time) that encompasses standards for formats of architectures and components); Category 2 (reduced life time) encompasses standard platforms, interfaces and protocols; Category 3 (further reduced life span) that encompasses data and information structures maintained by information and communication infrastructure, as well as execution structures across components of an information and communication infrastructures; and Category 4 (further reduced life time) that encompasses data and operations themselves. The above-mentioned categories point to the high complexity of a continuous engineering task. To handle this complexity, appropriate frameworks are needed to structure, to simplify (but do not oversimplify), and give means for controlling the systems and their development process still allowing for the decent degree of autonomy in both the target systems performance and the systems development processes.

In the context of continuous requirements engineering the FREEDOM framework was proposed [6]. In this paper the application of the FREEDOM framework for full continuous information systems engineering process is discussed, and the FREEDOM framework is compared to two other candidate approaches for continuous engineering. Thus, the purpose of this position paper is to promote the discussion on essential features of frameworks and reference models that are suitable for handling complexity of continuous information systems engineering. As mentioned already in [1], in continuous information system engineering it is important to follow continuously changing needs of business and also have built-in provision for later changes of information systems (having proper knowledge, information, and data) about different artifacts created during systems development).

As will be discussed further in Section 2, the FREEDOM framework theoretically can help to manage several of above-mentioned challenges. Therefore, in Section 3, it will be taken as the base for comparison with two other approaches that are used in the context of continuous engineering. Essential features of frameworks for continuous information systems engineering are summarized in this section, too. Brief conclusions are presented in Section 4.

2 The FREEDOM Framework for Continuous Information Systems Engineering

The FREEDOM framework was proposed for the purpose of continuous requirements engineering [6]. It has the following main constituents (see Figure 1): F – Future representation, R – Reality representation, E_1 – requirements Engineering, E_2 – fulfillment Engineering, D – Design and implementation, O – Operations, and M – Management. For the framework to be used in the whole information systems engineering context, more relationships should be considered. This issue will be described in more detail at the end of this section.

The constituents of the FREEDOM framework should be viewed as functions with changeable granularity, e.g., E_2 – *fulfillment Engineering* can be fully "moved into (inside of)" E_1 – *requirements Engineering*, and form function EE – *requirements Engineering and fulfillment Engineering* (see the first row in Table 1); or D – *Design and implementation* can be fully "moved into" E – *fulfillment Engineering* and form function ED – *fulfillment Engineering, Design and implementation*; and so forth.

F – *Future representation* is the constituent of the framework that is responsible for representation of the To-Be situation, i.e., the representation of a vision of the target system in its context. Artifacts that are developed by this function are mainly different enterprise models [7,8], enterprise architecture development artifacts [9], project plans, design documents, and even results of predictive analytics [10] that represent and characterize an envisioned future situation. These artifacts may be developed by F itself and also can be contributed by other constituents of the FREEDOM framework (see the light green arrow in Figure 1). Therefore, all blue colored functions are related to the F. It is not shown in Figure 1, however, can be seen in Table 1 and in [11].

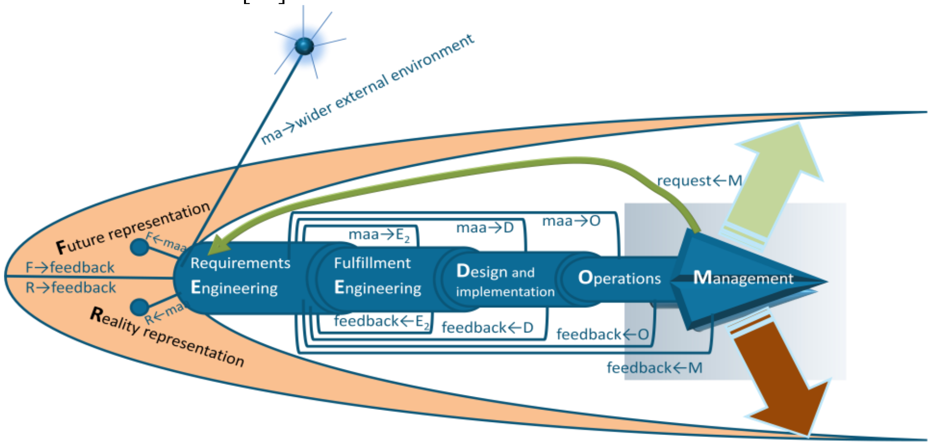


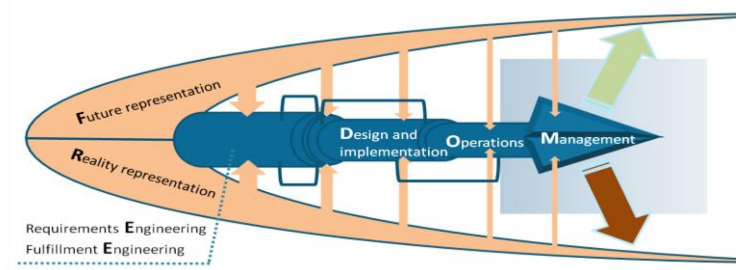
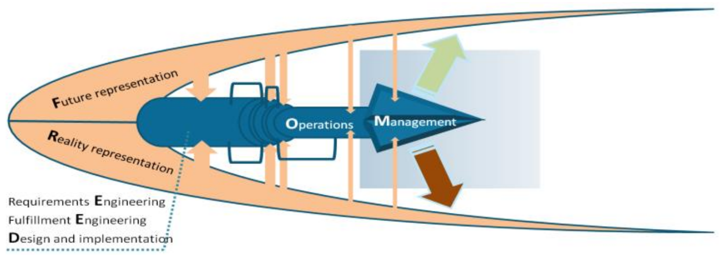
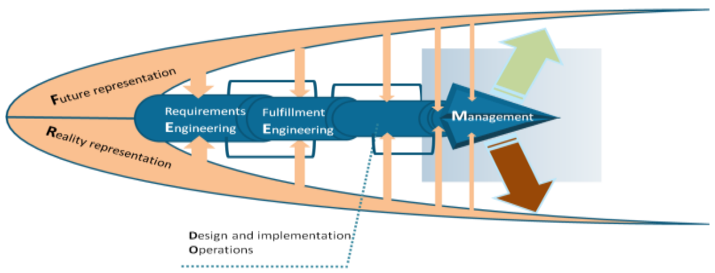
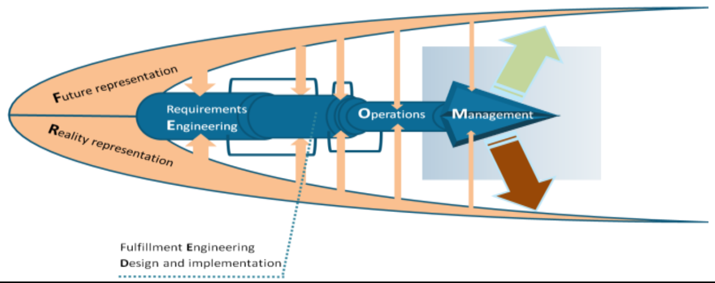
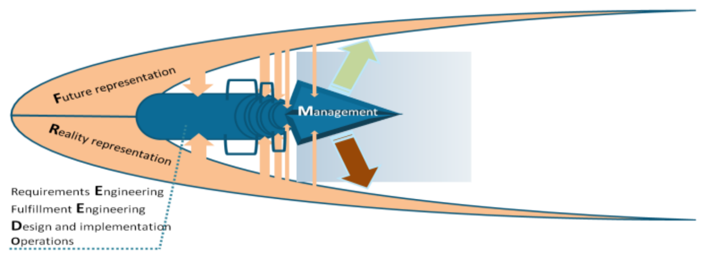
Fig. 1. FREEDOM framework for continuous requirements engineering, proposed in [6]

R – *Reality representation* is responsible for all artifacts that represent the present (As-Is) situation. The types of these artifacts are similar to those of F, with just the difference that here the information is about the current situation. Like in F, the contents may be developed by R itself or by other constituents of the FREEDOM framework. Therefore, all blue colored functions are related to the R. It is not shown in Figure 1, however, can be seen in Table 1 and in [11]. Information available in databases, warehouses, and other IT systems also may belong to R. The mapping and traceability between F and R is to be established and maintained.

E_1 – *requirements Engineering* is the function dedicated to the model and tool based acquisition and management of high quality requirements that can be used by functions on the right from E_1. E_1 to a large extent can help to meet the challenge mentioned in the previous paragraph. It also can richly contribute to F and R.

E_2 – *fulfillment Engineering* is the function that takes care of handling project portfolios that would lead to the fulfillment of stated requirements. This is the function that introduces branching in the model. Details of the branching are outside the scope of this paper. Despite it is common to put the design next to the requirements engineering [12], to take into consideration that the requirements engineering, design, and implementation often are distributed and overlapping, and include cross-cutting concerns, e.g., security [13,14]; the engineered process(es) are needed to ensure their continuous alignment, flexibility, and quality.

Table 1. Some variations of framework granularity

 Future representation Reality representation Design and implementation Operations Management Requirements Engineering Fulfillment Engineering	E_1 and E_2 merged in one process
 Future representation Reality representation Operations Management Requirements Engineering Fulfillment Engineering Design and implementation	E_1, E_2, and D merged in one process
 Future representation Reality representation Requirements Engineering Fulfillment Engineering Management Design and implementation Operations	D and O merged in one process (rarely possible)
 Future representation Reality representation Requirements Engineering Operations Management Fulfillment Engineering Design and implementation	E_2 and D merged in one process
 Future representation Reality representation Management Requirements Engineering Fulfillment Engineering Design and implementation Operations	E_1, E_2, D, and O merged in one process (rarely possible)

In simpler cases E_2 can be included in (merged with) E_1 or it can include (be merged with) D (see Table 1).

D – *Design and implementation* is the function that produces the design and handles implementation of the target system. The border between the design and implementation may be more or less strict depending on the fulfillment strategies, methods, chosen lifecycles, and guidelines established in E_2.

O – *Operations* regard the actual operation of the implemented system, including its maintenance.

M – *Management* refers to all levels of management under which the target system operates. The management can influence both the reality and its representation function R (see brown arrow in Figure 1) and the future vision and its representation function F (see light green arrow in Figure 1).

It is assumed that knowledge continuously propagates from E_1 towards O in a managed and transparent way. It is also assumed that each function can acquire information from other functions and can provide feedback to other functions. The management function can provide direct requests for actions to all other functions (Figure 1 shows it only for E_1). All functions can have the capability to acquire information from the wider external environment beyond the reach of F and R (shown only for E_1 in Figure 1). In the remainder of this section we will look more closely at how E_1 deals with information, however, the same considerations apply to E_2, D, and O.

We use term "information relationships" when referring to linkages between different functions of the FREEDOM framework. For *requirements Engineering* (E_1), the "information relationships" are represented in Figure 1. Here the information and knowledge flows between F and other elements of the framework, R and other elements of the framework, and some other "information relationships" (see Table 1) are not shown for the sake of clarity of representation.

In the framework concerning E_1 the following information relationships must exist to ensure continuous information systems engineering:

- Knowledge forward propagation from requirements Engineering to other constituents of the model: $E_1 \rightarrow E_2$, $E_1 \rightarrow D$, $E_1 \rightarrow O$, $E_1 \rightarrow M$ (these relationships are not shown in Figure 1), $E_1 \rightarrow R$ and $E_1 \rightarrow F$ (shown in Figure 2). In other words, the direct knowledge flow from E_1 to other FREEDOM constituents must be ensured.
- Knowledge supply from F and R: both future representations and reality representations should be available for E_1 (see Figure 1).
- Feedback information from all constituents of the framework: $F \rightarrow E_1$, $R \rightarrow E_1$, $E_2 \rightarrow E_1$, $D \rightarrow E_1$, $O \rightarrow E_1$, $M \rightarrow E_1$. By feedback information we understand here evaluative information about activities or artifacts of E_1.
- Information to be acquired by monitoring, applying analytics (*maa*) to, and auditing other constituents of the framework, namely, F, R, E_2, D, and O, as well as by monitoring and applying analytics (*ma*) to the wider external environment (as requirements engineering should be aware of scientific discoveries, new available technologies, competitive solutions, etc.).
- Requests from management (M), which can directly provide information about necessary deliverables of E_1 (dark green arrow in Figure 1).

The same scope of information relationships applies to other functions of the FREEDOM framework. The above list of these relationships shows the spectrum of information handling variability in continuous information systems engineering. Taking into consideration this spectrum, it is clear that, first, continuous information systems engineering has to deal with complex information handling tasks; second, handling of these tasks requires appropriate IT tool support; and, third, the handling of the information will require manual, semi-automatic, and fully automatic functions. Further discussion on these issues is beyond the scope of this paper.

Another issue to be taken into consideration is the fact that the structure (granularity of constituents – see Table 1) of the FREEDOM framework can change according to particular enterprise and project situations. This may require a different number of constituents with which the "information relationships" are established, but it should not exclude any of the relationships mentioned in the list presented above.

3 A Comparison of Candidate Approaches

In this section we will compare two approaches, which could be considered as candidate ones for continuous information systems engineering, to the FREEDOM framework and try to derive essential features or elements of the continuing information systems engineering by analyzing commonalities and differences between these approaches and the FREEDOM framework. One approach (Section 3.1) is chosen by focusing on “engineering”, another one (Section 3.2) – on focusing on “continuous” as the base concepts of interest.

3.1 The FREEDOM Framework vs. SoS V-Model

The V-model is one of the most known approaches in systems engineering [15]. Taking into consideration that in continuous information systems engineering it is necessary to deal with changes in multiple related systems (subsystems) with different frequency [1], the application of V-model to system of systems (SoS V-Model) is considered [15]. In [15] SoS breakdown structure is represented as the generic triple *Product*, *Processes*, and *People* that is fractally nested for all subsystems of the product, i.e., each product again is represented by Product, Processes and People of a smaller scale. SoS V-Model is an application of V-Model at each level of systems breakdown structure. To compare the FREEDOM framework and SoS V-Model we will use three artifacts from [15], namely, the original V-Model with additions, SoS V-Model, and the representation of technical baselines, documents, reviews, and audits for SoS.

Table 2 in the first column represents constituents of (SoS) V-Model, in the second column it shows corresponding elements of the FREEDOM framework The constituents of the V-Model are artifacts and/or functions: the FREEDOM framework represents just functions. Thus, in case of SoS V-Model artifacts, the functions, in which the FREEDOM framework generates a particular artifact, are represented in the second column. The sequence of V-Model elements is from left-down to up-right with the "Fabricate, Assemble, Code" as the bottom element. In Table 2 only the elements of V-Model are shown. The relationships between elements are considered separately in Table 3. From Table 2 we can see that the FREEDOM framework has corresponding elements to all constituents of SoS V-Model. However, the level of detail is different. Thus, for the integration and verification branch that consists of four elements in V-Model, the FREEDOM framework has just two functions D and R. This may be viewed as non-sufficient distinction between Design and Implementation in the FREEDOM framework. Still this may be compensated by recursive application of the FREEDOM framework for function D (Design and implementation).

Table 2. SoS V-Model and FREEDOM (components)

SoS V-Model constituents	**FREEDOM constituents**
V-Model (Decomposition and definition): User Requirements, System Concept, Validation Plan)	E_1 (requirements Engineering), F (Future representation)
V-Model (Decomposition and definition): System Specification and Verification Plan	E_1 and/or E_2 (fulfillment Engineering) and/or D (Design and implementation) depending on the organizational procedures and systems development methodologies (approaches in use)
V-Model (Decomposition and definition): Configuration item (CI) "Design-To" specifications and Verification Plan	E_1 and/or E_2 and/or D depending on the organizational procedures and systems development methodologies (approaches in use)
V-Model (Decomposition and definition): "Build-To" Specifications and Verification Procedures	E_1 and/or E_2 and/or D depending on the organizational procedures and systems development methodologies (approaches in use)
V-Model: Fabricate, Assemble, Code	D (Design and implementation), to some extent R (Reality representation)
V-Model (Integration and verification): Inspect to "Build-To" Specifications	D (Design and implementation), to some extent R (Reality representation)
V-Model (Integration and verification): Assemble CIs and Verify to Specifications	D (Design and implementation), to some extent R (Reality representation)
V-Model (Integration and verification): Integrate System and Verify to Specifications	D (Design and implementation), to some extent R (Reality representation)
V-Model (Integration and verification): Validate System to User Requirements	D (Design and implementation), to some extent R (Reality representation)
SoS V-Model breakdown structure: (System V-Model (Subsystem 1 V-Model (Sub-subsystem 1.1. V-Model (...)...)...) In SoS V-Model, the V-Model is applied to each level in the system (product) breakdown structure.	FREEDOM breakdown structure: $F(F(F(...)RE_1E_2DOM)R$ $(FRE_1E_2DOM$ $...)E_1...E_2...D...O...M...)$ In the FREEDOM framework, the framework can be applied to any product produced by any functions of the framework at any decomposition level of functions or products produced by them.
	Elements of the FREEDOM framework not addressed by SoS V-Model: O (Operations), M (Management), R (Reality representation) only partly supported

Table 3. SoS V-Model and FREEDOM (linkages)

SoS V-Model linkages	**FREEDOM linkages**
Information propagation *from* User Requirements, System Concept, Validation Plan *to* System Specification and Verification Plan *to* Decomposition and definition to Configuration item (CI) "Design-To" specifications and Verification Plan *to* "Build-To" Specifications and Verification Procedures *to* Fabricate, Assemble, Code	Knowledge forward propagation from requirements Engineering to other constituents of the model: e.g., $E_1 \rightarrow E_2$, $E_2 \rightarrow D$, $E_1 \rightarrow O$, $E_1 \rightarrow M$ $E_1 \rightarrow F$ (shown in Figure 2). In other words, the direct knowledge flow from E_1 through other consequential FREEDOM constituents Knowledge supply from F and R also can be considered.
Information propagation *from* Fabricate, Assemble, Code to Inspect to "Build-To" Specifications *to* Assemble CIs and Verify to Specifications *to* Integrate System and Verify to Specifications *to* Validate System to User Requirements	No corresponding linkages exist in FREEDOM Framework because of larger granularity of the corresponding functionality (see also Table 2). However, if FREEDOM framework is considered as one only branch model (no fulfillment breakdown) the feedback links of FREEDOM framework can be considered here.
Verification links between "Build-To" Specifications and Verification Procedures and Inspect to "Build-To" Specifications; verification links between Configuration item (CI) "Design-To" specifications and Verification Plan and Assemble CIs and Verify to Specifications; verification links between System Specification and Verification Plan and Integrate System and Verify to Specifications; validation links between User Requirements, System Concept, Validation Plan and Validate System to User Requirements	No corresponding linkages exist in FREEDOM Framework because of larger granularity of the corresponding functionality (see also Table 2). However, if FREEDOM framework is considered as one only branch model (no fulfillment breakdown) the feedback links of FREEDOM framework can be considered here.
Audit (SoS breakdown)	Audit (from left to right all linkages between FREEDOM constituents and fractal branching of FREEDOM framework)
Requirements, Functions and Preliminary Design flow Down (SoS breakdown structure)	Requirements flow forward via fulfillment Engineering (E_2)
Detailed design, Verification and Validation Roll Up (SoS breakdown structure)	Feedback links may be regarded in the flat model, otherwise, no corresponding linkages exist
	Links of the FREEDOM framework not addressed by SoS V-Model: Direct information flows between non-sequential elements, e.g., $E_1 \rightarrow D$, links to and from O (Operations) and M (Management) Monitoring links, analytics links

On the other hand, two constituents (O and M) of FREEDOM framework are not considered by SoS V-Model, and R is only partly supported. That shows that the

FREEDOM framework is more related with the business domain than is SoS V-Model.

The main question that arises from Table 2 is the following one "Is it necessary always in continuous information systems engineering to explicitly show the relationship between the developing requirements, developing design, and implementation"? Similar concerns can be derived also from Table 3. Both SoS V-Model and the FREEDOM framework are similar because they can be applied as fractal structures recursively. The main difference is in granularity of some elements. Only the FREEDOM framework is explicitly linked to the operations and management and has means for environment and related functions monitoring and analytics. Also it can be mentioned that the FREEDOM framework is more flexible than SoS V-Model (see Table 1).

3.2 The FREEDOM Framework and Continuous Software Engineering Roadmap and Agenda

In this subsection the FREEDOM framework is compared to the continuous software engineering roadmap and agenda presented in [16] – further named "Continuous*". The comparison is reflected in Table 4.

Table 4. Continuous* and FREEDOM

Continuous*	**FREEDOM functions and linkages**
Business strategy and planning: Continuous planning	O (Operations) and M (Management)
Business strategy and planning: Continuous budgeting	Not directly addressed, but budgeting can be included as sub-function in O, M or other functions.
Development: Continuous integration	D (Design and implementation)
Development: Continuous delivery	D (Design and implementation)
Development: Continuous deployment	D (Design and implementation)
Development: Continuous verification	D (Design and implementation) with supplementary information from E_1 and E_2
Development: Continuous compliance	D (Design and implementation) with supplementary information from E_1 and E_2, O, and M
Development: Continuous security	$E_{2,}$ can be also included as a sub-function in all other functions of the framework
Development: Continuous evolution	E_1 and E_2
Operations: Continuous use	O (Operations) and M (Management)
Operations: Continuous trust	Not addressed in FREEDOM framework
Operations: Continuous run-time monitoring	Monitoring relationships in FREEDOM framework
Improvement and innovation: Continuous improvement	Links between O and E_1 and M and E_1
Improvement and innovation: Continuous experimentation	Not directly addressed, but experimentation can be included as a sub-function of the functions
Linkage between business strategy and development	Forward and backward links between different functions and M (Management)
Linkage between development and operations	Forward and backward links between D and O

According to [16], the "*" in word Continuous* implies that different "continuous activities" can emerge over time. All these activities can be positioned within a holistic view as described in the first column of Table 4. Emerging over time reminds of the categories of invariants of different life times discussed in [1]. Actually, none of approaches explored in this paper provide clear tools for identifying and positioning, and directly handling these invariants.

Continuous* and the FREEDOM framework have several commonalities, however, the abstraction level of Continuous* is higher than that of the FREEDOM framework – so the correspondence, therefore, is rather symbolic than practical. Both frameworks differ from SoS V-Model as they have the monitoring activities or relationships. Continuous* does not directly include Audit function, however, it might be one of the means for ensuring Continuous security. Continuous* does not directly include Analytics, however, it might be one of the means for Continuous strategy. Nesting of Continuous* is quite taxonomy like which is a contrast to possibly fractal nesting of SoS V-Model and the FREEDOM framework.

The comparison of three approaches presented in this section lets to see the following evidences:

- The FREEDOM framework has a potential to be used for framing continuous information systems development because it has means for addressing almost all issues (handling trust is an exception) raised by engineering oriented approach (SoS V-Model) and continuity oriented approach Continuous*. The FREEDOM framework while not prescribing a transparent relationship between the design and implementation can accommodate both strict engineering like V-Model [15] and also widely used agile [17] approaches/methods/tools at arbitrary levels of functional breakdown of the framework. The framework also has a high flexibility for handling different development situations as shown in Table 1.
- The frameworks for handling continuous information systems engineering have to deal with a number of complexities with respect to systems diversity, variability of scope, methods, and change frequencies, differences in scope and project granularity, and other issues. Thus, the frameworks should have both the means for variability handling (branching) and the means for nested (also fractal) functional breakdown maintenance.
- The frameworks for continuous information systems engineering have to incorporate monitoring, audit, and analytics sub-functionality.
- The frameworks have to have explicit means for handling continuous integration, delivery, deployment, verification, and testing (not explicitly addressed in the FREEDOM framework), which most probably have to go hand in hand with continuous representation of a system in scope breakdowns and linkages (can be handled by F and R functions of the FREEDOM framework).
- There should be a transparent linkage between the information systems development and organizational issues in the frameworks to ensure continuity in alignment of business and information systems development.
- None of the approaches presented in this section clearly address time issues for categories of continuous software development relevant invariants of different life times [1]: Category 1 (indefinite life time) that encompasses standards for formats of architectures and components); Category 2 (reduced life time) encompasses

standard platforms, interfaces and protocols; Category 3 (further reduced life span) that encompasses data and information structures maintained by information and communication infrastructure, as well as execution structures across components of an information and communication infrastructures; and Category 4 (further reduced life time) that encompasses data and operations themselves.

4 Conclusions

This paper investigated complexity of framing continuous information systems engineering by describing functions, linkages, and flexible modifications of the FREEDOM framework and comparing this framework with two other candidate approaches. The candidate approaches were selected so that one of them strongly addresses systems engineering issues while another one strongly addresses continuity issues.

The comparison of approaches gave an opportunity to reveal several evidences that are essential in developing frameworks for continuous systems engineering.

The study presented in this paper has the following limitations: (1) only three frameworks were analyzed and compared, while it would be possible to considered a larger variety of frameworks and thus, possibly, find more evidences; (2) the differences in granularity of frameworks were not discussed in detail; (3) the systems development lifecycle perspective was not analyzed in detail; (4) it was not investigated whether there are specific systems engineering situations where strongly the preference should be given to a particular continuous systems engineering framework.

Nevertheless, the presented comparison showed that the FREEDOM framework has a potential to handle many engineering and continuity issues. It also showed that the framework would benefit from the representation of more detailed selectable variations of Design and implementation function and addressing time issues of software development relevant invariants. To see how more detailed continuous software development can be incorporated in the FREEDOM framework, in further research it is intended to integrate some methods engineering techniques into the FREEDOM framework.

Acknowledgment. This work is supported by the Latvian National research program SOPHIS under grant agreement No.10-4/VPP-4/11.

References

1. Weber, H.: Continuous Engineering of Information and Communication Infrastructures. J-P. Finance (ed.) FASE'99, LNCS 1377, pp. 22–29 (1999)
2. Fong, J.S.P.: Information Systems Reengineering, Integration and Normalization. In: Information Systems Reengineering, Integration and Normalization, Springer, pp 1–29 (2015)

3. Von Detten, M., Platenius, M.C., Becker, St.: Reengineering Component-Based Software Systems with Archimetrix. In: Software and Systems Modeling, vol. 13, Issue 4, pp. 1239–1268 (2014)
4. Pouclet, R.: Pro iOS Continuous Integration, Springer (2014)
5. Zimmermann, A., Jugel, D., Sandkuhl, K., Schmidt, R., Schweda, C. Möhring, M.: Architectural Decision Management for Digital Transformation of Products and Services. In: Complex Systems Informatics and Modeling Quarterly, CSIMQ, Issue No. 6, pp. 31–53 (2016)
6. Kirikova, M.: Continuous Requirements Engineering in FREEDOM Framework: a Position Paper. Joint Proceedings of REFSQ-2016 Workshops, Doctoral Symposium, Research Method Track, and Poster Track co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2016), March 14–17, 2016, Gothenburg, Sweden, vol. 1564. CEUR-WS.org (2016)
7. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: Enterprise Modeling Tackling Business Challenges with the 4EM Method, Springer (2014)
8. Seigerroth, U.: The Diversity of Enterprise Modeling – a Taxonomy for Enterprise Modeling Actions. In: Complex Systems Informatics and Modeling Quarterly, CSIMQ, No. 4, pp. 12–31 (2015), http://dx.doi.org/10.7250/csimq.2015-4.02
9. TOGAF® 9.1: Part II: Architecture Development Method (ADM). Introduction to the ADM, 1999–2011, http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap05.html
10. Finlay, S.: Predictive Analytics, Data Mining and Big Data: Myths, Misconceptions and Methods, Springer (2014)
11. Virmani, M.: Understanding DevOps & Bridging the Gap from Continuous Integration to Continuous Delivery. Proceedings of INTECH 2015, IEEE (2015)
12. Richter, M., Flückiger, M.: User-Centred Engineering, Springer (2014)
13. Kaiser, B., Weber, R., Oertel, M., Böde, E., Monajemi Nejad, B., Zander, J.: Contract-Based Design of Embedded Systems Integrating Nominal Behavior and Safety. In: Complex Systems Informatics and Modeling Quarterly, CSIMQ, No. 4, pp. 66–91, ISSN 2255-9922 (2015), http://dx.doi.org/10.7250/csimq.2015-4.05
14. Schmitt, C., Liggesmeyer, P.: Getting Grip on Security Requirements Elicitation by Structuring and Reusing Security Requirements Sources. In: Complex Systems Informatics and Modeling Quarterly, CSIMQ, No. 3, pp. 15–34, ISSN 2255-9922 (2015), http://dx.doi.org/10.7250/csimq.2015-3.02
15. Clark, J.O.: System of Systems Engineering and Family of Systems Engineering from a Standards, V-Model, and Dual-V Model perspective, Systems Conference, 3rd Annual IEEE, pp. 381–387. (2009), http://dx.doi.org/10.1109/SYSTEMS.2009.4815831
16. Fitzgerald, B., Stol, K-J.: Continuous Software Engineering: A Roadmap and Agenda. In: The Journal of Systems and Software (2015), http://dx.doi.org/10.2016/j.jss.2015.06.063
17. Dinsoyr, T., Lassenius, C.: Emerging Themes in Agile Software Development: Introduction to the Special Section on Continuous Value Delivery. In: Information and Software Technology, 77, pp. 56–60 (2016)