Continuous Requirements Engineering in the FREEDOM Framework: a Position Paper

Marite Kirikova

Riga Technical University, Latvia
marite.kirikova@rtu.lv

Abstract. Continuous change, which, nowadays, is a commonly accepted feature of both business and technical environments, requires continuous change in a business and its supporting systems, including information technology solutions. This, in turn, leads to the need for continuity also in the realm of requirements engineering. It is necessary to be aware whether it is necessary to change the requirements, why and when this should be done; and how to handle the related changes in the environment, business, system, systems development process, and systems maintenance. To find out how to answer at least part of these questions, the FREEDOM framework is established by analyzing different configurations of work systems and also information and knowledge flows in a viable systems model. The paper focuses on propagation and feedback relationships among the requirements engineering function and other constituents of the FREEDOM framework.

Keywords: requirements engineering, continuous engineering, future state model, as-is state model, solution engineering, design.

1 Introduction

In the situation when the business environment changes very rapidly, new approaches to systems development are needed. One of the solutions is agility. However, agility alone can cause chaotic systems growth [1]. Therefore, it is not surprising that more and more attention is currently paid to different variations of systems engineering, such as Enterprise Engineering, Security Engineering, Usability Engineering [2], etc. Engineering is recognized by organized, transparent, and responsible statement and achievement of systems development goals, regardless of whether the system under consideration is a physical one, a technical one, or even an abstract combination of thoughts (idea system). However, traditionally we may understand engineering as a process, which strictly starts with requirements elicitation, and then follows all V model steps down to detailed design and testing and then up to validation [3]. This might be very clear and "one dimensional" if the system is built from scratch. However, nowadays, one of the main challenges is that several evolving systems are working in concert requiring agile and continuous adjustments in organizational strategies, policies, processes, and supporting information technology. As a result, such notions as continuous engineering [4], continuous software engineering [5], DevOps [6], continuous requirements engineering [7] and the like are emerging. Focusing on requirements engineering, the question arises, how continuous

requirements engineering relates to other types of engineering and other phenomena in the contemporary rapidly changing multi-systems environment. To seek answers to this question we propose and then use the FREEDOM framework, which has emerged from related research in worksystems based systems engineering and systems viability [8]. The framework is introduced in Section 2. Afterwards the continuous requirements engineering, as a constituent of the FREEDOM framework, is discussed in Section 3. Brief conclusions are presented in Section 4.

2 FREEDOM Framework

The FREEDOM framework has the following 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.

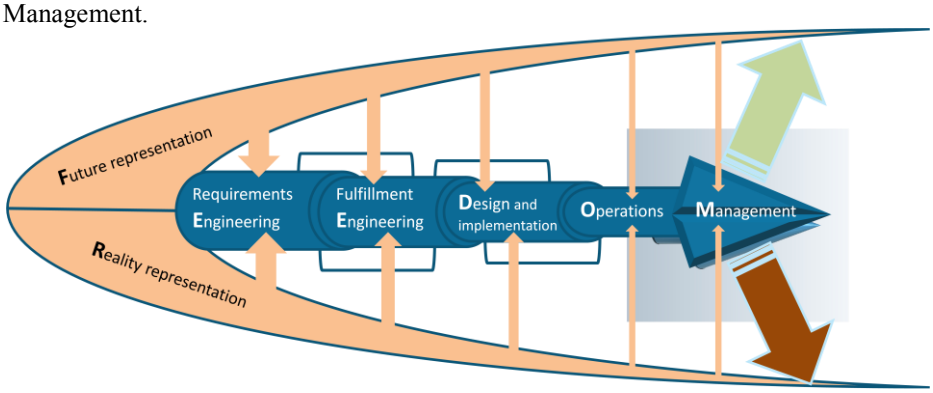

Fig. 1. FREEDOM framework

The constituents of the 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*; 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 [9, 10], enterprise architecture development artifacts [11], project plans, design documents, and even results of predictive analytics [12], 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 Figure 2 and green arrow in Figure 1).

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. 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 - this is one of the challenges of contemporary enterprises.

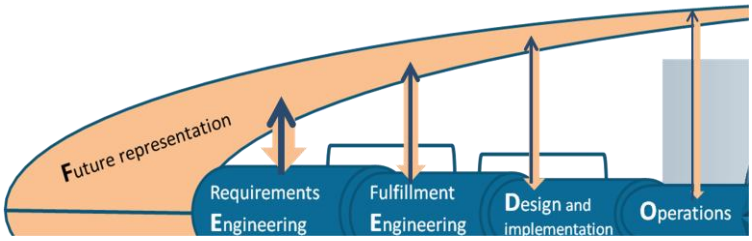

Fig. 2. Contribution of different FREEDOM constituents to the Future representation

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. It is common to put the design next to the requirements engineering [2]. However, we have to take into consideration that the requirements engineering, design, and implementation are often distributed and overlapping nowadays and include cross-cutting concerns, e.g., security [13, 14]; therefore, there should be engineered process(es) that take care of their continuous alignment, flexibility, and quality. In simpler cases E_2 can be included in (merged with) E_1 or it can include (be merged with) D.

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 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. All functions can have the capability to acquire information from the wider external environment beyond the reach of F and R. In the next section we will look more closely at how E_1 deals with information in the case of continuous requirements engineering.

3 Continuous E_1

From the functional point of view requirements engineering is an information processing function. Therefore, embedment of continuous requirements engineering in the FREEDOM framework will be discussed from the point of view of "information relationships" between *requirements Engineering* (E_1), which in this case is continuous requirements engineering; and other constituents of the framework. The "information relationships" are represented in Figure 3, however, 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" are not shown for the sake of clarity of representation. These flows are shown in Figure 1 and Figure 2 (Figure 2 represents only flows with respect to F, but the representation of the flows for R is very similar).

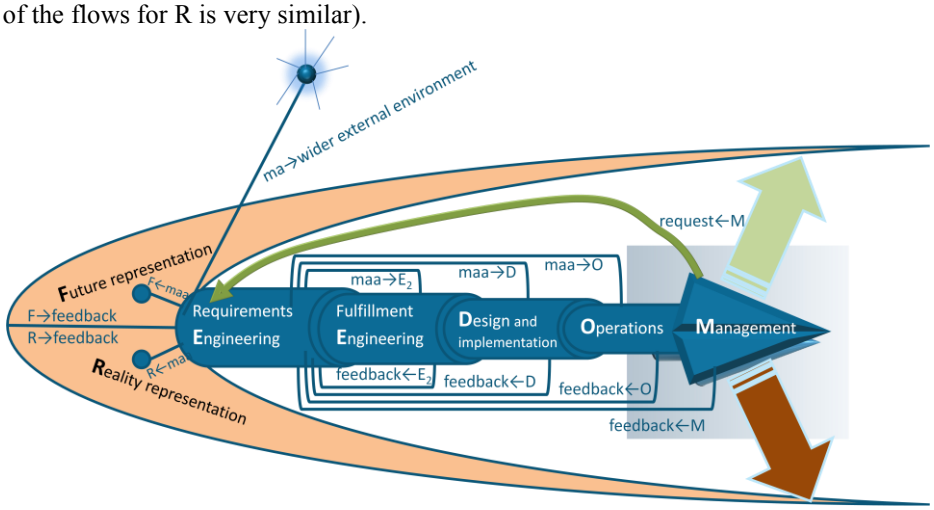

Fig. 3. Continuous requirements engineering in FREEDOM framework (ma - *m*onitoring and *a*nalytics, maa - *m*onitoring, *a*nalytics, and *a*udit)

The following information relationships must exist in the framework to ensure continuity of requirements engineering:

- Knowledge forward propagation from requirements Engineering to other constituents of the model: $E_1 \rightarrow E_2$, $E_1 \rightarrow D$, $E_1 \rightarrow O$, $E_1 \rightarrow M$, $E_1 \rightarrow R$ (these relationships are not shown in Figures 1-3), 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 to, and auditing other constituents of the framework, namely, F, R, E_2, D, and O, as well as by monitoring and applying analytics 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.

The above list of "information relationships" shows the spectrum of information handling variability in continuous requirements engineering. Taking into consideration this spectrum, it is clear, first, that continuous requirements 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.

Another issue to be taken into consideration is the fact that the structure (granularity of constituents - see Section 2) 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.

4 Conclusions

With the purpose to better understand the phenomenon of continuous requirements engineering, this paper analyzed the continuous requirements engineering function in the context of the FREEDOM framework, which has emerged from related research in worksystems based systems engineering and systems viability. The use of the framework helped to identify main information units to be handled by continuous requirements engineering. To some extent, it also allows assessment of the complexity and variability of the tasks of continuous requirements engineering. We can conclude that, besides the regular "duties" of requirements engineering [15], the following issues have to be considered in continuous requirements engineering:

- Continuous requirements engineering has to have such capabilities as knowledge propagation, monitoring, analytics, and auditing.
- As can be derived from the above, continuous requirements engineering must be both reactive and predictive concerning user needs, possible innovative solutions, and possible fulfillment, design, implementation, and operation problems.
- Continuous requirements engineering has to be able to handle a wide variety of knowledge and information flows related to other functions of systems development, operations (including maintenance), and management.
- Continuous requirements engineering has to be flexible with respect to the number and granularity of other functions belonging to the same functional ecosystem, as well as with respect to its own modes of functionality (manual, semi-automatic, automatic).
- The complexity of the tasks requires appropriate support tools for continuous requirements engineering.

In this position paper only "What?" with respect to continuous requirements engineering was considered. The detailed proposals of how to integrate all issues discussed in this paper in continuous requirements engineering processes is the subject of further research.

Acknowledgment

This work is supported in part by the Latvian National research program SOPHIS under grant agreement Nr.10-4/VPP-4/11.

References

1. Haunts, S.: Advantages and disadvantages of agile software development (2014). Available at: https://stephenhaunts.com/2014/12/19/advantages-and-disadvantages-of-agile-software-development/
2. Richter, M., Flückiger, M.: User-Centred Engineering, Springer (2014)
3. Clark, J.O.: System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective, Systems Conference, 2009 3rd Annual IEEE, pp. 381–387. Available at: http://dx.doi.org/10.1109/SYSTEMS.2009.4815831
4. The competitive advantage of continuous engineering, IBM white paper. Available at: http://public.dhe.ibm.com/common/ssi/ecm/ra/en/raw14358usen/RAW14358USEN.PDF
5. Continuous Software Engineering, Bosch, J. (ed.), Springer (2014)
6. Virmani, M.: Understanding DevOps & bridging the gap from continuous integration to continuous delivery. Proceedings of INTECH 2015, IEEE (2015)
7. Kirikova, M.: Enterprise Architecture and Knowledge Perspectives on Continuous Requirements Engineering. Proceedings of REFSQ-2015 Workshops, Research Method Track, and Poster Track co-located with REFSQ 2015, Essen, Germany, March 23, 2015. CEUR-WS.org, Vol. 1342, ISSN 1613-0073, pp. 44–51, CEUR (2015)
8. Kirikova, M.: Work Systems Paradigm and Frames for Fractal Architecture of Information Systems. CAiSE Forum 2014, Thessaloniki, Greece, June 16–20, 2014, Selected Extended Papers. Information Systems Engineering in Complex Environments, Vol. 204, Lecture Notes in Business Information Processing, pp. 165–180, Springer (2015). Available at: http://dx.doi.org/10.1007/978-3-319-19270-3_11
9. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: Enterprise Modeling Tackling Business Challenges with the 4EM Method, Springer (2014)
10. Seigerroth, U.: The Diversity of Enterprise Modeling – a Taxonomy for Enterprise Modeling Actions. Complex Systems Informatics and Modeling Quarterly, CSIMQ, No. 4, pp. 12–31 (2015). Available at: http://dx.doi.org/10.7250/csimq.2015-4.02
11. TOGAF® 9.1: Part II: Architecture Development Method (ADM). Introduction to the ADM, 1999–2011. Available at: http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap05.html
12. Finlay, S.: Predictive Analytics, Data Mining and Big Data: Myths, Misconceptions and Methods, 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. Complex Systems Informatics and Modeling Quarterly, CSIMQ, No. 4, pp. 66–91, ISSN 2255-9922 (2015). Available at: 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. Complex Systems Informatics and Modeling Quarterly, CSIMQ, No. 3, pp. 15–34, ISSN 2255-9922 (2015). Available at: http://dx.doi.org/10.7250/csimq.2015-3.02
15. Pohl, K.: Requirements Engineering - Fundamentals, Principles, and Techniques, Springer (2010)