Requirements Inheritance in Continuous Requirements Engineering: a Position Paper

Anita Finke

Riga Technical University, Riga, Latvia
anita.finke@rtu.lv

Abstract. Requirements, information about project's history and information about existing situation of enterprise have important role in successful requirement engineering process during the project and in post-project phases. The first appearance of requirements is in the pre-project phase, then in the project and even in the post-project phase. Each idea and requirement has a history that can be a very important aspect of a successful project and the information system development, management and support. By author's observation, one of today's problems in the beginning of an IT project and during the project's phases is historical information and requirements availability and inheritance. There are situations when specialists need to spend time searching and discovering requirement history instead of studying it. The goal of this position paper is to discuss the problem of poor information and requirements inheritance and to point out the importance of it in continuous requirements engineering.

Keywords: Requirements engineering, Continuous requirements engineering, Requirements inheritance.

1 Introduction

If we try to model a situation where a stakeholder or a team member, during the interviews or other kind of sessions, needs to tell the full story from the first idea and stated requirements till the current moment, we can assume that it will not be normally applicable in the most of situations. In this context, we have observed the following problem in practice: poor information inheritance at the start of the project and during the project hinders the productive collaboration of stakeholders and possibly minimizes the analysis work effectiveness. We can only imagine how lengthy the process can be, if each involved person needs first to seek for historical information and then only study it instead of just opening the repository (system, notes, documents etc.) and immediately starting studying the information. Term "requirements inheritance" is understood as a process of transfer of full package of information and requirements (including metadata, requirements versions, related documents etc.) from one involved person to another from one project or system lifecycle phase to another. In future researches this definition will be expanded with explanation of inheritance techniques, tools and scope.

To better understand the concepts of continuity and inheritance, the author will point out the situation in requirements inheritance with the help of the case example. This

case will be used as an illustration of the complexity of the project, flow of information, and continuous requirements analysis and maintenance. Related research is described in the Section 2. The case example is described in Section 3, and Section 4 discusses CRE (Continuous Requirements Engineering), Section 5 – requirements inheritance. In Section 6 the conclusion of this research is presented.

2 Related work

The term "Continuous Requirement Engineering" is quite new [1]. While some issues relevant in continuous requirements engineering have been discussed much earlier [2], still there are only few research papers with keywords "Continuous requirements engineering" available. M. Kirikova describes continuous RE with respect to Enterprise Architecture and Knowledge Perspectives [1]. One of her paper's goals was to list out different challenges in continuous RE, including ideal linkage between knowledge, enterprise architecture, business processes and development projects. Although these are specific topics relevant to CRE, a big picture of continuous requirements engineering and Requirements Inheritance role in this is still not properly addressed.

In SWEBOK [3] it is pointed out that requirements process is "… initiated at the beginning of a project that continues to be refined throughout the life cycle.". In practice the requirements appear before the IT project is started when business ideas come in. Those pre-project requirements can take an important role in successful IT projects and solutions development.

The Requirements Engineering (RE) concept represented in systems life cycle [4], contains activities like identifying the stakeholders, gaining and understanding their needs, identifying requirements, clarifying and restating the requirements, analyzing, defining, specifying, prioritizing, deriving, partitioning, allocating, tracking, managing, testing and verifying, and validating requirements. These are activities for a "development process" and can fully support requirements engineering processes when the development project team members are involved from the first ideas and requirement generation. But in situations when project team members and other involved persons are changing, these activities are not enough to provide effective requirements engineering processes. They do not include questions about information inheritance – activities where the main task is to study, for example, the history of pre-project phase, the history of existing requirements and existing requirements created before current project phase. And these tasks do not include the precondition of accessibility of this information. It is very important in case of change of project team members or even project team.

BABOK 2.0 [5] describes 6 dimensions – Elicitation, Enterprise Analysis, Requirements Analysis, Solution Assessment and Validation, Requirements Management & Communication, Business Analysis Planning and Monitoring. And inputs and outputs for each of these activities. But strict sequence of these activities is not recommended. The new version of BABOK 3.0 [6] consists of new improvements in dimensions/knowledge areas but still do not include continuity and requirements inheritance.

Ruhaya Ab. Aziza, Bernard Wongb [7] talks about requirements relationship knowledge. They describe importance of requirement relationship knowledge, because "…managing these incremental changes in software development is very challenging.

The knowledge and management of requirements changes is crucial in the management of software changes. However, the knowledge of requirements change is not enough where we need to know how the requirements are related to one another. Furthermore, as requirements change, there is a need to understand what happens to existing relationships between those changing requirements…. ". Ebert and De Man [8] state that knowledge and information are important in continuous systems and requirements improvements. Without information and knowledge inheritance or information flow, each new phase or iteration may require starting information gathering from the beginning.

3 Case example – IS Project

To highlight the mentioned problems, the author will use the case example - a real life project in company Star (the author can't divulge the real name of the company), where there is a need to develop a new cooperation relationship management system. It will be a custom made system with specific functionality, introduced by an outsourced development team. It is an appropriate example, because it shows the full set of project phases. The whole situation during the development of the first part of the project was: project managers from client's side changed twice; analysts from client's side changed 3 times; 5 analysts from development side changed; pre-project phase was 2 years long; development of the first part of the project took 2 years; maintenance started form the second development year; 3 platforms were used for captured requirements – 2 platforms in client's side (JIRA and SharePoint) and 1 platform at the developer sides (JIRA); many different tools and many RE methods were used by clients and developers. The systems were launched in production after one year of development. After that, the development of the systems continued. Parallel to the development of the current system analysis for further developments was carried out on the client's side.

The biggest problem during the project was requirements management by two sides. On client's side - in e-mails, documents and information systems like JIRA and SharePoint. And in developer's side - also in e-mails, documents and information systems like JIRA. It was challenging to keep up to date information at both sides. Another problem was information flow from one person to another, because of the existing communication model. The third problem was information and requirements inheritance in moments of new team member's appearance – there were too many questions asked by each new staff member. In the result, there were unsatisfied business stakeholders and delayed project deadlines.

4 Continuity of Requirements Engineering

Regarding this case example we can see that the process of requirements life cycle does not end up with the start of the systems usage in operations. This especially holds for new systems, where in practice there is a common situation when some of the functionality becomes clearer after this functionality is launched in operations and it is being used. Pre-project phase, project, post-project phase, maintenance, change management etc. - it is continuous work on the system. Today technologies, business laws and regulations, market competition, needs and other factors can change so fast, that it almost

is not possible to develop an information system that will not be changed or improved. So, continuity is inevitable.

For this reason we need to consider the following question: what are the preconditions that will best support RE continuity according to a specific system, process and other analyzable elements? In this paper the author chooses to focus on the inheritance. But this is not the only necessary precondition. To list all necessary preconditions for continuous RE, deeper research will be needed.

5 Requirements Inheritance

If we take a look on project phases described and take into account the whole project history of changing staff and involved sides (clients side, development team, etc.), we can make the following assumptions or conclusions (see Fig.1.): each new person, e.g., a new analyst, needs to repeat all RE (or business analysis activities), from the scratch: gather information, analyze it, etc. Gathering each bit of information is time-consuming task; there can be a number of questions raised repeatedly from each analyst to better understand the situation; the involved stakeholders can become intolerant; projects communication can become complicated; some information can be lost over time, etc.

For successful requirements inheritance we need to consider: the way of information transfer from one tool to another; the historical information and requirement availability for all teams; the inheritance effectiveness (well-structured and available information); and the number and the size of involved teams. This consideration is independent from used project management methodologies and RE methods and methodologies, enterprise politics etc.

One of the problems of successful information transfer from one tool to another and from one person to another is the time we need to collect all existing information, prepare it and send to another party. And the next step is to receive, transform or make some formatting work for the received information, to produce correct “import”. Some of the RE tools support import and export functionality, however, in many cases this support is too limited. It is essential to provide information flow from all phases. But it may be impossible because in some phases there are no specific tools for requirements capturing, some phases may require very specific tools, and some phase can be supported by much simpler tools.

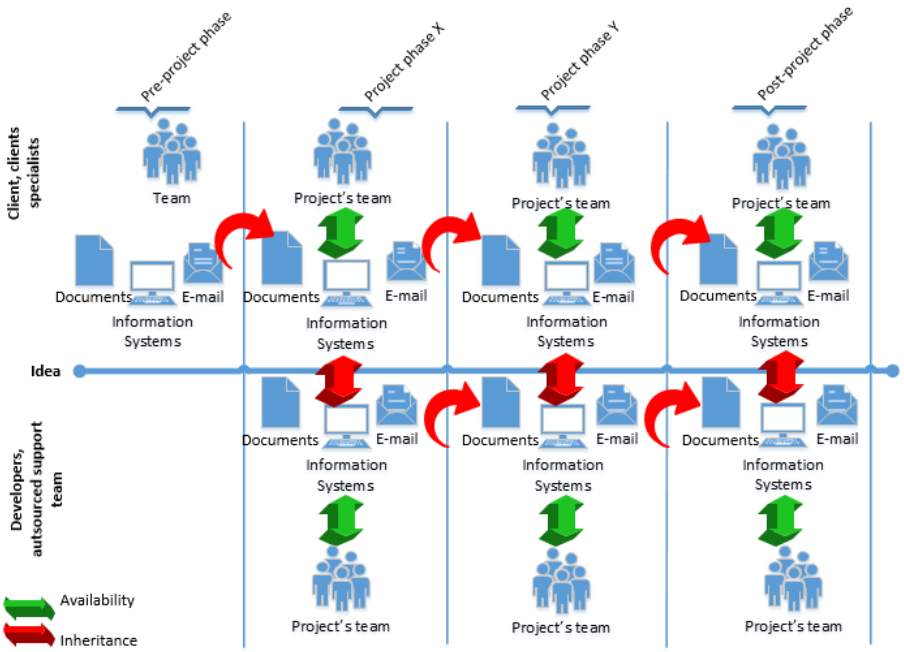

Fig. 1. Requirements inheritance process during the project phases

6 Conclusion

We can say that the start of RE is the first moment when someone starts to consider necessity of system or some solutions of its problems. It is the moment when we start to document some ideas, issues and other information. Of course, not always will it end up with a system or other solution, but in cases when it will, a lot of information will start to be formed and accumulated. The initial idea may grows into a feasibility study, into a project, into development process, into a system. But the system is not the end point of activities; it may be followed by maintenance, change management, and other activities. One system can transform into another, and the life cycle continues. And if we imagine what is happening if the information inheritance from one phase to another is not ensured, we can say that each time someone is trying to understand requirements and the existing situation, he "invents the wheel again", which cannot be considered as profitable and time saving approach. Information inheritance or, in this case, Requirements Inheritance is very important if we want to fluently work with the system, system requirements, systems improvements, maintenance, etc.

Regarding continuity, we know that the system development does not end up with the systems launch in production because the use of system can create bugs, new ideas, and new needs. As IT technologies are changing "by days" not by years, the business needs are also more and more rapidly changing, the environment and laws and regulations are changing. So system changes and the need for requirements engineering are unstoppable. Even disabling or transformation of the system requires RE processes.

This continuity of requirements engineering emphasizes the need for good requirements inheritance.

This paper just introduces the discussion on requirements inheritance. The ideas about continuous RE and Requirements inheritance described in this paper have to be investigated and researched deeper in order to provide a "big picture" of the role of continuous RE and requirements inheritance, and to provide a method or methods and the tool to support requirements inheritance in CRE.

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 M. Kirikova, "Enterprise Architecture and Knowledge Perspectives on Continuous Requirements Engineering," 2015. [Online]. Available: http://ceur-ws.org/Vol-1342/05-CRE.pdf. [Accessed: 10-Jan-2016].
2 J.-P. Finance, Ed., *Fundamental Approaches to Software Engineering*, vol. 1577. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.
3 P. Bourque and R. E. Fairley, *Guide to the Software Engineering - Body of Knowledge*. 2014.
4 R. R. Young, *The Requirements Engineering Handbook*. 2004.
5 IIBA, *A Guide to the Business Analysis Body of Knowledge ® (BABOK ® Guide)*. 2009.
6 "BABOK Guide v3 - Business Analysis Training - Watermark Learning." [Online]. Available: http://www.watermarklearning.com/blog/babok-guide-v3/. [Accessed: 10-Jan-2016].
7 R. A. Aziz and B. Wong, "The Interplay between Requirements Relationships Knowledge and Requirements Change towards Software Project Success: An Assessment Using Partial Least Square (PLS)," *Procedia Comput. Sci.*, vol. 46, pp. 732–741, 2015.
8 C. Ebert and J. De Man, "Requirements uncertainty: influencing factors and concrete improvements," in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pp. 553–560.