Requirement Evolution: Towards a Methodology and Framework *

Le Minh Sang Tran
(Co-supervised by Prof. Fabio Massacci and Prof. John Mylopoulos)

DISI, Università degli Studi di Trento,
Via Sommarive 14, I-38123 (POVO), Trento, Italy
tran@disi.unitn.it

Abstract. Software systems are undergoing continuing changes and rapid revolution. As consequence, requirements that were satisfied may no longer be satisfied or new requirements may be introduced. Thus, a challenging aspect is to develop a methodology and tools to model, manage, and analyze the evolution of requirements. In this paper, we describe our work at UNITN which targets a framework and methodology for requirement evolution. As an evidence for the feasibility of our approach, we describe our steps to achieve the goal as well as our preliminary result. In which, we propose a foundation to model requirement evolution, and concepts to support reasoning on evolutionary model.

Keywords: Requirement Engineering, Rule-based Evolution Modeling.

1 Introduction

Evolution is a phenomenon that happens commonly in many domains. It is a fact of life. Environments and natural species living within them evolve. Also, other entities such as societies, theories, concepts, ideas - artificial, or virtual - all evolve over time in their own context. The ability to evolve is prerequisite for survival keypoint. As the term reflects a process of progressive, it is necessary to determine what is admitted as progressive in every context.

The evolution phenomena as observed in diversified domains varies significantly. In most of the cases, evolution results from changes in several or many, of the facets of an evolving entity or set of entities. Individual changes are generally small with respect to an entity, but even then their impact may be critical. In areas such as software, evolution refers to a process of continually updating software systems in responding to changes in their operating environment, their specification and properties. It is inevitable in software systems as they need to continue to satisfy changing business needs, new regulations and standards, and the introduction of new technologies. Such evolution may involve changes that add, remove, or modify features, redesign the system for migration to a new platform, or that integrate with other application. As a consequence, part

* This work is supported by the European Commission under projects EU-FET-IP-SECURECHANGE.

of the software may have to be modified to correct errors that are found in operation to adapt the software to a new platform or to improve its performance.

Numbers of researches have been built for the study of software evolution but still, little attention has been paid for the evolution at requirement level. The fact that requirement evolution is one of the most important evolutions motivates our research. We focus our study on the formalization of evolving requirements, analyzing their affects and management mechanism.

In the rest, we briefly review past work in the field (§2). Then we describe our research objectives (§3) which is followed by a research agenda (§4). Next we present our preliminary results (§5) in which we model requirement evolutions in terms of evolution rules and propose quantitative metrics for reasoning on evolutionary requirement models. Finally we conclude the paper (§6).

2 Related Work

A majority of approaches to software evolution has focused on the evolution of architecture and source code level. However, in recent years, changes at the requirement level have been identified as one of the drivers of software evolution [3, 11, 21]. As a way to understand how requirements evolve, research in PROTEUS [15] classifies changing requirements (that of Harker et al [10]) into five types, which are related to the development environment, stakeholder, development processes, requirement understanding and requirement relation. Later, Lam and Loomes [12] discusses the problem of evolving requirements and presented the EVE framework for characterizing changes. The EVE framework consists of two parts which are a meta-model and an associated process model. The meta model captures key modeling concepts in requirement evolution such as change, impact, risk, and viewpoint. The process model aims to a methodological framework for handling new and changing requirements; however, not much of this was provided.

Several approaches have been proposed for supporting requirements evolution. Madhavji [14] proposed a process model for change management (PRISM) to manage the versions of the changed artifacts, to collect change-related data. Han [9] presented an approach to impact analysis and change propagation to software change. Impact analysis and change propagation are performed on software artifacts and their dependences. Software artifacts are represented using augmented EBNF. The impact analysis concerns the introduction, modification, and deletion of software artifacts and dependences. It is comprised of automatic applying of change patterns and interactive confirmation of potential impacts. The change propagation process is a combination of automatic propagation based on codified rules and interactive user guidance bases on the impact analysis results. Zowgi and Offen [21] work at meta level and view requirement models as theory (or a belief set) in some nonmonotonic logic. Requirements are considered as set of belief about the theory ("machine") going to be built. Requirement evolution are the changes occur when mapping a theory to another theory through a process of rational belief revision. The process begins from an incomplete requirement model, at each step, a new set of requirement changes is brought to bear. After a series of revi-

sions, the requirement model is refined and completed in each step. The limitation of this approach is the overhead in encoding requirement model into logic.

Russo et al.'s [17] propose an analysis and revision approach to restructure requirements to detect inconsistency and manage changes. The main idea is to allow evolutionary changes to occur first and then verify their impact on requirement satisfaction in the next step. Also based on this idea, Garcez et al [3] focus on evolving requirement specifications rather than evolving requirement. This work supports modification while preserving particular requirement goals and properties. It proposed the use of a cycle comprised of two phases: analysis and revision. During the analysis phase, techniques of abductive reasoning are used to check specifications if a number of desirable properties of the system is satisfied. If not, diagnosis information is generated. The revision phase uses the techniques of inductive learning to modify the specifications according to diagnosis information generated. Similar to Garcez et al, Ghose's [7] framework is based on formal default reasoning and belief revision, aiming to address the problem of inconsistencies due to requirement evolution. This approach is supported by automated tools [8]. Also relating to inconsistencies, Fabrinni et al.'s [4] deals with requirement evolution expressed in natural language, which is challenging to capture precisely requirement changes. Their approach employs formal concept analysis to enable a systematic and precise verification of consistency among different stages, hence, control requirement evolution.

Other notable approaches include Brier et al.'s [2] to capturing, analyzing, and understanding how software systems adapt to changing requirements in an organizational context; Felici et al [6] concern with the nature of requirements evolving in the early phase of the system; Stark et al [19] study the information on how change occurs in the software system and attempts to produce a prediction model of changes; Lormans et al [13] use a formal requirement management systems to motivate a more structural approach to requirement evolution.

3 Research Objective

Evolution modeling obviously requires extra efforts in the software development process. The motivation behind this work is to make the software system-to-be more resilient to changes during its lifetime in an efficient way.

Existing requirement modeling languages (*e.g.,* UML, Tropos, KAOS) do not provide any tools or concepts to capture the evolution of the requirement models. Obviously designers cannot express the evolutionary puzzle without additional concepts. Some existing languages may allow users to define new concepts to represent the evolution. For example, to express an idea that a model evolves to another one, UML users can define a new stereotype *e.g.,* 'evolve to', and tag it to the *association relation* between two models. However, such extensions are ad-hoc, and thus are difficult to exchange among designers. Moreover, ad-hoc extensions do not make much sense due to the lack of systematic analyses and reasoning methods. Even if UML extends itself to deal with evolution, this extension would hardly replicate to other languages.

Recent studies on the field focused on how to adapt system configuration due to environment changes in order to satisfy predefined requirements. This kind of system

supports (semi)automatically reconfiguration of system components so that it can guarantee the designed purposes regards less environment changes. This can be considered as internal evolution. In the other side, external evolution refers to the cases that completely new requirements may arrives, then new components have to be implemented. And some existing components become obsoleted. Only a few of past studies covered both internal and external evolutions. However most of them are at high level of abstraction. And there is lack of practical frameworks as well as CASE tools that can be ready for real world projects.

This inspires our research objective. Concretely, we aim to construct:

> *"A comprehensive, practical framework for requirement evolution that can deal with arbitrary changes in requirement models. This framework is general enough so that it can be applied to as much as possible existing modeling languages."*

In which we would like to identify a framework to

- elicit requirement evolutions
- model and represent evolutions in such a way that is consistent with human mental models.
- reason and analyze evolutions that meet stakeholder's desire.

In subsequence sections, we discuss our research agenda to achieve our goals. Also, we sketch a preliminary of our approach which has been published in CAiSE'11

4 Research Plan

Here we describe our research plan in accordance to our objective. Generally speaking, we can separate our working plan into two phases. In the first phase, we work on a generic approach dealing with evolution for requirement model at a high level of abstraction so that our approach will not be limited to any particular requirement modeling language. In the second phase, we instantiate the generic approach to a specific modeling language to prove the applicability of the proposed approach. For validation purpose, we apply our approach to a couple of case studies taken from industrial projects.

In the following we elaborate our moves in order to achieve a framework for requirement evolution. In which we need to construct or support:

An generic approach for modeling requirement evolution that supports uncertainly modeling. We can anticipate the changes but we cannot assure whether changes happen. We can only say these changes may happen with a certain probability.

Reasoning and Analysis on evolutionary model. This is a crucial part of our approach. We identify several kinds of analysis as follows:

- *Coverage Analysis*: This analysis assures that all customer requirements are always satisfied regardless to evolutions.
- *Impact Analysis and Change Propagation*: This analysis addresses research questions such as *"Which is the impact of evolution"* and *"How changes are propagated?"*.

- *Risk Analysis*: This is similar to the risk analysis done in static requirement model [1], but here we put it in the context of evolution.
- *Security Analysis*: We analyze the security properties of requirement models (*e.g.,* confidentiality) to see whether evolutions falsify them.
- *Usefulness Analysis*: This kind of analysis answers research questions such as *"Given anticipated evolutions, what is the best evolution-resilient design for the system?*. The term 'evolution-resilient' means that a system still operates properly even though evolution does actually happen.
- *Robustness Analysis*:This analysis provides another view rather than the usefulness analysis. While usefulness analysis helps to choose optimal design for the system-to-be, robustness analysis takes a design and try to look ahead its evolvability, *e.g.,* how much does it cost for repairing the system once evolutions happens. Together with usefulness analysis, robustness analysis gives a more comprehensive to the evolution of requirements.

The first four anlyses are addressed in existing modeling languages or studies in the literature. However, they should need some modifications in order to apply to an evolutionary requirement model. We might not implement all of these analyses but some of them that meet stakeholder's desires which are discussed in the validation step.

Interaction protocols for the collaboration between stakeholder and engineers to construct evolutionary requirement models. It is motivated by the fact that most of evolutions are uncertain and anticipated. Hence, in order to do any analysis, we need to determine "all" potential evolutions and their likelihoods of occurrences, namely probabilities of evolutions. Intuitively, these probability values should be supported by stakeholder or domain experts who usually are not equipped with a strong background in computer science or mathematic. Thus it is difficult for stakeholder (or domain experts) to give a concrete numbers, say 60%, for the probability of evolution. Instead, using qualifiers such as "more likely", "unlikely" is more familiar. To recover a numerical probabilities from these qualifiers, we might employ the idea of Analytic Hierarchy Process (AHP) [18].

Specialization of the proposed approach . At this step, we will map the idea of the generic approach to a specific modeling language. Also, we might need to develop algorithms for analysis in the context of this modeling language. Notably, when we do specialization, some of generic notions and concepts need to be refined to comply with the modeling language.

Validation of proposed approach. Obviously this is an important part of our work to prove the applicability and usability of the proposed approach. In which we try to answer two questions: *"Could our approach be applicable to real projects?"* and *"Does our approach meet users' desires?"*.

To answer these questions, we analyze case studies taken from industrial projects *e.g.,* ATM [16] and SWIM [5]. We gather their requirement documents and build up requirement models. We directly work with stakeholder to understand their desires to develop our work. Then we apply our approach to model and analyze evolutions. The final outcome will be again validated with stakeholder.

A proof of concept to verify the applicability of our proposed framework. Concretely, we plan to:

- Implement a prototype of a CASE tool for a specialization of our framework. In which we instantiate the approach to a particular language, *e.g.,* i* language, and implements a tool to model evolutions and reasoning on these.
- Support evolution simulation. After designers have chosen a solution (*i.e.* an implementation for the system-to-be), the simulation engine can take this solution and generate random events (evolution) to see how the solution reacts to evolution.

5 Proposed Approach and Preliminary Result

In the following we describe our approach to deal with the requirement evolution. The detail of this approach is published in CAiSE 2011 [20]. Here, we only present the sketch.

5.1 Rule-Based Approach for Modeling Requirement Evolution

To the purpose of generality, we firstly treat a requirement model as a set of elements and relations rather than investigating on any specific requirement model (*e.g.,* goal-based model, UML models). We do not go into details about how many kinds of element and relationship a model would have. Instead, we treat elements at abstract meaning, and are only interested in the satisfaction relationship which determines how to satisfy an element (*i.e.* requirement) by others (*e.g.,* component).

Considering evolutionary requirement models, we classify two kinds of evolution depended on actors (*i.e.* designer, stakeholder, reality) who decide changes. They are *controllable* and *observable* evolutions. In the former, designers can decide which evolution to follow in order to meet some high level requirements from the stakeholder, with low level requirements for components. The later, on the other hand, is not under the control of designers, but it can be somehow detected when it happens ; its likelihood can be estimated with a certain confidence of the stakeholder.

We model these kinds of evolutions in terms of evolution rules: *controllable rule* and *observable rule*. The former basically is pair of *before-* and *after-models*, which means the *before-model* can be substituted by the *after-mode*. In this sense, the *after-model* is a design alternative (or design choice) of the *before-model*. At design time, designers decide which design alternative to implement. The later rule is modeled as a triplet of *before-*, *after-models* and the *likelihood* that this evolution happens, namely evolution probability.

5.2 Reasoning on Evolutionary Model

Among reasonings and analyses listed in the research plan, we have addressed the *usefulness analysis* so far. Here we focus on the question: "Whether or not a model element (or set of element) becomes useless after evolution?". Since the occurrence of evolution is uncertain, so the usefulness of an element set is evaluated in term of probability. We have introduced two metrics based on the evolution probabilities as follows:

Max Belief (MaxB): of an element set X is a function that measures the maximum belief supported by Stakeholder such that X is useful to a set of top requirements after evolution happens. This belief of usefulness for a set of model element is inspired from a game in which Stakeholder play a game together with Designer and Reality to decide which elements are going to implementation phase.

Residual Risk (RRisk): of an element set X is the complement of total belief supported by Stakeholder such that X is useful to set of top requirements after evolution happens. In other words, residual risk of X is the total belief that X is not useful to set of top requirements regard to evolution. Importantly, do not confuse this notion of residual risk with the one in risk analysis studies which are different in nature.

In [20], we have discussed long-tail problem for which we use these two metrics instead of a single one called *Total Belief*.

6 Conclusion

This paper describes our interests on the field of software evolution, particularly requirement evolution. After reviewing many past studies in the field, we focus ourselves on constructing a methodology to deal with requirement evolutions. We are going to perform a series of analyses (*e.g.,* usefulness, robustness, and impact analysis) to obtain the ultimate purpose: support designers in building more evolution-resilient software. So far, we have worked on a possible way to model requirement evolution using evolution rules, and a couple of concepts, Max Belief and Residual Risk, which are the stepping stone for our further analysis. This approach has been published in CAiSE'11.

References

1. Y. Asnar. *Requirements Analysis and Risk Assessment for Critical Information Systems*. PhD thesis, ICT School, University of Trento, 2009.
2. J. Brier, L. Rapanotti, and J. Hall. Problem-based analysis of organisational change: a real-world example. In *Proc. of IWAAPF '06*. ACM, 2006.
3. A. d'Avila Garcez, A. Russo, B. Nuseibeh, and J. Kramer. Combining abductive reasoning and inductive learning to evolve requirements specifications. In *IEE Proceedings - Software*, volume 150(1), pages 25–38, 2003.
4. F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. Controlling requirements evolution: a formal concept analysis-based approach. *ICSEA '07*, 2007.
5. Federal Aviation Administration. System wide information management (swim) segment 2 technical review. Technical report, FAA, October 2009.
6. M. Felici. *Observational Models of Requirements Evolution*. PhD thesis, University of Edinburgh, 2004.
7. A. Ghose. A formal basis for consistency, evolution and rationale management in requirements engineering. *ICTAI '99*, 1999.
8. A. Ghose. Formal tools for managing inconsistency and change in re. In *IWSSD '00*, Washington, DC, USA, 2000. IEEE Computer Society.
9. J. Han. Supporting impact analysis and change propagation in software engineering environments. In *In Proceedings of 8th International Workshop on Software Technology and Engineering Practice (STEP'97/CASE'97)*, 1997.

10. S. Harker, K. Eason, and J. Dobson. The change and evolution of requirements as a challenge to the practice of software engineering. In *RE '93*, pages 266 –272, 1993.
11. J. Hassine, J. Rilling, J. Hewitt, and R. Dssouli. Change impact analysis for requirement evolution using use case maps. In *IWPSE '05*, 2005.
12. W. Lam and M. Loomes. Requirements evolution in the midst of environmental change: a managed approach. In *CSMR '98*, 1998.
13. M. Lormans, H. van Dijk, A. van Deursen, E. Nocker, and A. de Zeeuw. Managing evolving requirements in an outsourcing context: an industrial experience report. In *IWPSE '04*, pages 149 –158, 2004.
14. N. H. Madhavji. Environment evolution: The prism model of changes. *IEEE Trans. Software Eng.*, 18(5):380–392, 1992.
15. Project PROTEUS. Deliverable 1.3: Meeting the challenge of chainging requirements. Technical report, Centre for Software Reliability, University of Newcastle upon Tyne, June 1996.
16. Project SecureChange. Deliverable 1.1: Description of the scenarios and their requirements. Technical report, 2009.
17. A. Russo, B. Nuseibeh, and J. Kramer. Restructuring requirements specifications. In *IEE Proceedings: Software*, volume 146, pages 44 – 53, 1999.
18. T. L. Saaty. Decision making with the analytic hierarchy proce. *Int. J. Services Sciences*, 1(1):83–98, 2008.
19. G. E. Stark, P. Oman, A. Skillicorn, and A. Ameele. An examination of the effects of requirements changes on software maintenance releases. *Journal of Software Maintenance: Research and Practice*, 11(5):293–309, 1999.
20. L. Tran and F. Massacci. Dealing with known unknowns: Towards a game-theoretic foundation for software requirement evolution. In *23rd International Conference on Advanced Information Systems Engineering (CAiSE'11)*, 2011.
21. D. Zowghi and R. Offen. A logical framework for modeling and reasoning about the evolution of requirements. *ICRE '97*, 1997.