Evolutionary Design Observations

Massimo Felici
LFCS, School of Informatics, The University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
tel. +44-131-6505899, fax. +44-131-6677209
massimo.felici@ed.ac.uk

Abstract

This paper is concerned with software requirements evolution in industrial settings. The discussion provides new insights in requirements engineering. This paper highlights the problem of empirically understanding and modelling requirements evolution.

1 Introduction

Requirements Evolution is one of the main issues that affect development activities as well as system features (e.g., system dependability). Although researchers and practitioners recognise the importance of requirements evolution, research results and experience are still patchy. This points out a lack of methodologies that address requirements evolution. This paper explores new directions in requirements evolution research. Heterogeneous engineering provides a comprehensive account of system requirements. Heterogeneous engineering stresses a holistic viewpoint that allows us to understand the underlying mechanisms of evolution of socio-technical systems. Requirements, as mappings between socio-technical solutions and problems, represent an account of the history of socio-technical issues arising and being solved within industrial settings. The formal extension of a heterogeneous account of requirements provides a framework to model and capture requirements evolution [3]. This paper argues that the better our understanding of socio-technical evolution, the better system dependability. In summary, this paper is concerned with software requirements evolution in industrial settings. This paper highlights the problem of empirically understanding and modelling requirements evolution.

2 Evolutionary Design Observations

Requirements, as mappings between socio-technical solutions and problems, represent an account of the history of socio-technical issues arising and being solved within industrial settings. The characterisation of requirements and requirements changes allows the definition of requirements evolution [3]. Heterogeneous engineering stresses a different role for requirements. The shift from the paradigm of problems searching for solutions to the one of solutions searching for problems points out a new role for requirements with respect to (design) solutions and problems. Most software design processes and organisations rely on the first paradigm (i.e., problems searching for solutions). Thus, software production takes into account a certain relationship between requirements, design and system implementation. This relationship implies a specific order to software production phases. Regardless the adopted development process, each software production complies with the paradigm of problems searching for solutions. This is one of the reasons because most software development processes start with a requirement phase. In contrast, heterogeneous engineering takes into account the second paradigm (i.e., solutions searching for problems). Heterogeneous engineering therefore points out that requirements link (design) solutions and problems observed (by coding, testing, usage, etc.) in the system under consideration. On the one hand requirements map solutions to given problems. On the other hand requirements narrow and browse solution spaces in order to address observed problems.

This section describes how the comprehensive account of heterogeneous requirements evolution supports the refinement of design models. Heterogeneous requirements engineering highlights how design models (that is, solutions) support the observation of requirements (evolution). Moreover, the solution space transformation allows the gathering of requirements (evolution) during design. Consider a simple design scenario using UML [6]. *Use cases* in UML models capture high level system requirements: “A use case describes sequences of actions a system performs that yield an observable result of value to a particular actor” [5]. Use cases in UML therefore represent the starting point with respect to the Rational Unified Process (RUP) [4]. The analy-

sis of use cases provides the basis for the production of class diagrams [1]. The collaborations required between classes to provide the functional capability necessary to support requirements (in the form of use cases) will be examined in more details as the design progresses. The consideration of these collaborations will clarify the specification of the classes in the class model, hence class diagrams. As the specification becomes more and more firm, classes can be organised into subsystems of coherent and cohesive functional capability. Thus, on the one hand use cases capture system requirements, on the other hand system design identifies use cases. It is easy to figure out how the solution space transformation extends development workflows that rely on UML modelling. In this case, system design (in terms of architecture and classes) represents solutions. As development progresses, stakeholders highlight anomalies. According to the solution space transformation, requirements are mappings between solutions and problems. Solutions (i.e., architectural and classes design) contextualise given problems. The solution space transformation therefore resolves the given problems into the proposed future solutions. The future solutions reconcile the initial solutions with the given problems. The solution space transformation highlights how design solutions evolve in order to address observed problems. On the other hand the solution space transformation identifies the requirements, as mappings between solutions and problems. These requirements therefore highlight use cases of the system solutions. Hence, the solution space transformation supports the identification and refinement of use cases [5].

3 Evolution as Dependability

Requirements evolution represents just one aspect of the evolution of socio-technical systems. A taxonomy identifies an evolutionary space, which provides a holistic viewpoint in order to analyse and understand the evolution of socio-technical systems [2]. The taxonomy stresses the relationship between system evolution and dependability. Different models and methodologies take into account to some extent the evolution of socio-technical systems. Unfortunately, these models and methodologies rely on different assumptions about the evolution of socio-technical systems. This can cause misunderstandings and issues about system dependability and evolution. Therefore, a taxonomy of evolution identifies a framework that allows the analysis of how socio-technical systems evolve. On the one hand the resulting framework allows the classification of evolution of socio-technical systems. On the other hand the framework supports the analysis of the relationships between the different evolutionary phenomena with respect to dependability. Unfortunately, the collection and analysis of evolutionary data are very difficult activities, because evolutionary information is usually incomplete, distributed, unrelated and vaguely understood in complex industrial settings.

4 Requirements Evolution Engineering

Requirements evolution modelling allows to tailor development processes and artefacts to development environments. The combination of empirical analyses and requirements evolution models captures environmental features. It identifies a convenient requirements engineering practice that continuously provides feedback while software production progresses. Although these methodologies provide a comprehensive account of requirements evolution, requirements engineering practice little exploits them. On the other hand requirements engineering practice needs to identify how requirements evolution supports software production. Requirements evolution engineering involves analysis, modelling and practice of requirements evolution. Requirements evolution therefore identifies strategies and methodologies that support software production. This paper describes how the comprehensive account of heterogeneous requirements evolution supports the refinement of design models. Heterogeneous requirements engineering highlights how design models (that is, solutions) support the observation of requirements evolution. The solution space transformation allows the gathering of requirements (evolution) during design.

References

[1] Simon Bennett, John Skeleton, and Ken Lunn. *Schaum's Outline of UML*. Schaum's Outline Series. McGraw-Hill, 2001.

[2] Massimo Felici. Taxonomy of evolution and dependability. In *Proceedings of the Second International Workshop on Unanticipated Software Evolution, USE 2003*, pages 95–104, Warsaw, Poland, April 2003.

[3] Massimo Felici. Observational models of requirements evolution, 2004.

[4] John Hunt. *The Unified Process for Practitioners: Object Oriented Design, UML andJava*. Practitioner Series. Springer-Verlag, 2000.

[5] Dean Leffingwell and Don Widrig. *Managing Software Requirements: A Use Case Approach*. Object Technology Series. Addison-Wesley, second edition, 2003.

[6] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.