

ABSTRACT: Recent research in requirements engineering has identified two strong drivers of requirements evolution: environmental turbulence mediated by system stakeholders and the requirements process that accompanies the design process. Current methods in requirements engineering fail in managing evolving requirements. Most of the methods do not support the different views of the stakeholders involved in the requirements process. Moreover, they do not provide any means for predicting the impact in terms of risk and cost of requirements' changes. The paper shows an empirical investigation of an avionics safety-critical industrial case study pointing out aspects of requirements evolution. In the paper we propose a framework for predictive changes risk/cost analyses relating type of changes into requirements, software life cycle, impact of changes and structure of requirements. The results suggest further works dealing with requirements evolution.

1 INTRODUCTION

Requirements specification is one of the most important phases of the software life cycle (Berry & Lawrence 1998; Davis & Hsia 1994; Siddiqi & Shekaran 1996; Sommerville & Sawyer 1997a). The recovery activity is more cost effective during the requirements specification phase than subsequent phases (i.e., design specification, coding, testing, etc.) (Boehm 1981; Boehm 1984). Recent research has pointed out interesting aspects of changing requirements (Lam 1997; PROTEUS Project 1996; Stark, Skillicorn, & Ameen 1998). Requirements changes are due to several environmental aspects, human factors as well as environmental constraints and aspects (e.g., standards, development processes, business targets, etc.) (Arthur 1992; PROTEUS Project 1996). Moreover, requirements specification must change because

- It is impossible to build requirements specification right the first time
- Requirements evolve during the life cycle due to the environmental turbulence (e.g., stakeholders knowledge evolution, business targets, etc.)
- Requirements are interdependent on one another.

Current methods in requirements engineering emphasise the need for the management and representation of multi-perspectives and evolving requirements (Nuseibeh, Kramer, & Finkelstein 1994; Sawyer, Sommerville, & Viller 1997; Siddiqi & Shekaran 1996; Sommerville & Sawyer 1997b; Sommerville &

Sawyer 1997a). From the requirements process viewpoint there is actually the need for providing tools which help in the management of the process with predictive features. Recent research in requirements engineering (PROTEUS Project 1996) has identified two strong drivers of requirements evolution:

- environmental turbulence mediated by system stakeholders: designers, software specialists, engineers, end-users, project managers
- the process of requirements definition that accompanies the design process.

Moreover, practical experience shows that requirements need to be represented from different viewpoints and with different formalisms/approaches in order to be comprehensible by different system stakeholders. Without this, further turbulence arises due to disagreements between stakeholders.

The paper is organised as follows. Section 2 describes the case studies. Section 3 shows the framework for predictive changes risk/cost analyses. The conclusions and further works are listed in Section 4.

2 A SAFETY-CRITICAL CASE STUDY

The section describes the avionics safety-critical industrial case study that we have analysed focusing on the requirements evolution. The case study consists of software that must satisfy the most stringent level of DO178B (RTCA 1992). The empirical investigation takes into account the software requirements evolution of the case study. There are two main business stakeholders cooperating in the definition of the

provides the engine's requirements, and the *supplier*, who produces the software.

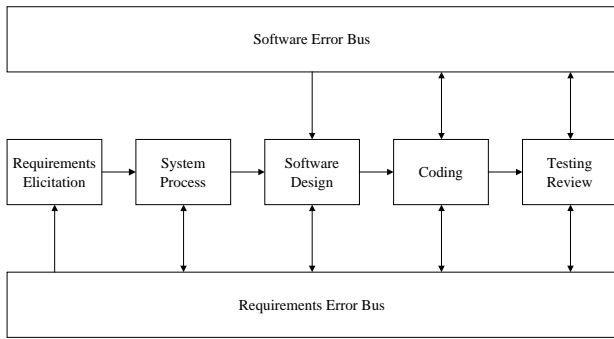


Figure 1: The safety critical software life cycle of the case study

Figure 1 shows the development life cycle. Whenever changes and modifications will arise, they will cause feedback loops into the software life cycle. Notice that as software development progresses from requirements to design and code, problems require items further back in the chain to be modified. The size and scope of these changes will be different for each modification and for each project, thus only general guidelines have been defined from the supplier. For example, while coding a part of the software detailed design a problem may be found with the design or an easier implementation may be constructed which requires the original design to be modified. Also modifications, which are due to problems, hardware changes or even stakeholders, may be introduced from the requirements at any stage of the software project. Another problem is related to the expansion of information down the design chain to the code. As design progresses, good design practice requires that the software requirements are partitioned into smaller more manageable items. This means that the coding phase at the end of the chain is not a single phase but a collection of interrelated phases. The boundaries between phases are not clearly delineated and some instances overlap. For example, as the code consists of many items it may be possible to begin the integration of some of them before the code for the rest is completed. This is often the case, even on small simple projects. Then it is very difficult to determine the end point of a process and to fix the point in which a project transition occurs from one phase to another. Thus, elements within each phase may be in different progress stages. Good configuration management in terms also of traceability, therefore, needs to be maintained on any project to ensure that the final code is complete and consistent with its higher level requirements and that verification of the final code has been

There are two significant milestones that are encountered during a project. One milestone is certification. For modifications to a certified standard of software the full life cycle phase of testing and verification should be complied with. In essence the project should start again, from the beginning, all the phases are repeated. The extent to which all the phases have to be repeated will depend upon the time elapsed since certification. Another milestone occurs around the end of the project life cycle, between the point where the software is being used in flight trials but before certification has been reached. During this stage of the life cycle, new issues to the project specific documents will be required. Also, depending on the size of the modifications, some of the review stages will need to be repeated.

2.1 Evolutionary analysis of the case study

The case study consists of 22 different software releases of the software. Figure 2 shows the evolution of software requirements among releases. The representation of requirements' changes points out three outliers, these are due to the requirements policy of allocating changes into releases. In fact most of the changes are allocated into releases major new ones. The trend of modified requirements (i.e., added, deleted and changed) is simply fitted with two polynomial curves of different degrees.

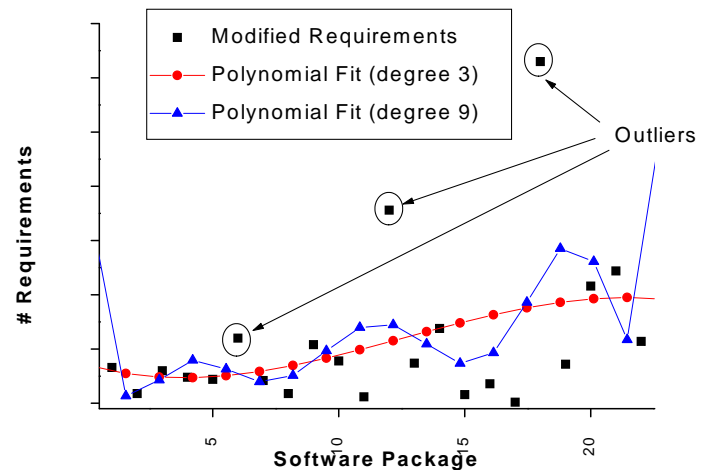


Figure 2: Trends of requirements evolution and identifications of outliers.

These curves (Fig. 3) have a regular trends identifying two different requirements evolution. The first one (circle line) follows a *slow evolution* representing a maturity curve in relation with the requirements understanding of the stakeholders. The second one (up-triangle line) follows a *fast evolution* representing the requirements process adopted in the specific environment, then it is related to the business environment (e.g., changes policy, business targets, etc.). According to their trend we call the two type of evolutions

points between the fast and slow evolution are suitable candidates for control points or milestones of the requirements process. From a general perspectives they represent the intersection between the stakeholders' understandings (slow evolution) and the requirements process (fast evolution).

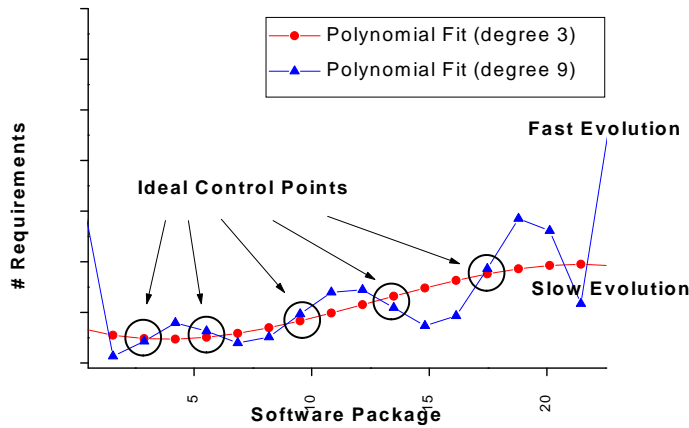


Figure 3: Fast evolution, slow evolution and requirements process control points.

3 PREDICTIVE CHANGES ANALYSES

The case study justifies not only the need for devising evolving requirements, but also the need for defining new methods for assessing the impact and the risk of requirements changes. The evolving nature of requirements points out the need for predictive changes analyses which could be useful tools for decision making during the software life cycle. The risk and cost of requirements changes should be formalised and quantified in a predictive way based on requirements information. The empirical investigation on the case study points out useful information suggesting the idea for predictive changes risk/cost analyses in relation with requirements evolution. The underlying idea is based on information relating type of changes and the structure of requirements. That is, we hypothesise that different changes into requirements can produce different effects which are related to the type of changes, the software life cycle and the structure of requirements. To make clear this idea in what follows there is a description of type of changes and structure of requirements.

Table 1 shows the type of changes that have been identified in the case study. The changes fall into three main classes, namely, *General*, *Domain Specific* and *Product Line*. The general changes are due to the requirements process and they could be associated to the process maturity and knowledge. The domain specific changes are due to aspects which characterise the development domain. The product line changes are due to needs of the specific software product.

attribute of the requirements. Table 2 shows the attributes of requirements that we have identified in the case study. The attributes identify aspects of requirements that are usually defined informally, but they specify useful information for controlling and monitoring requirements changes. Most of the requirements changes affect some of the attributes. The attributes viewpoint helps to evaluate cost of changes, moreover it defines a general initial structure for requirements.

The empirical investigation has identified a general framework which represents a base for white-box and black-box analyses. The underlying idea is devising a black-box predictive changes risk/cost analysis based on to a three-dimensional space with axes in terms of type of changes, life cycle phases and implementation of changes (Fig. 4).

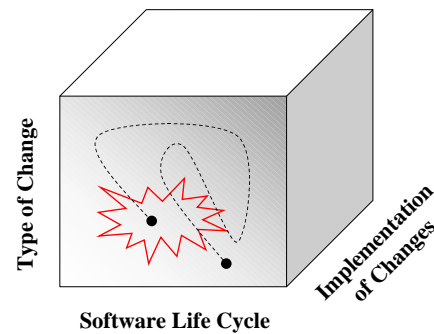


Figure 4: A three dimensional space in relation with requirements evolution.

A point within this three-dimensional space represents the risk/cost of implementing changes. The white-box predictive changes risk/cost analysis is based on to the same three-dimensional space together with traceability information of the process's deliverables (e.g., requirements specifications, design document, programme code, etc.). Experience shows that the traceability information is fundamental for requirements management. A point within the three-dimensional space represents the risk/cost of changing requirements. Requirements engineers have to plan effectively, safely and costly a path within the three-dimensional space. The three-dimensional space could be the base for a "black-box" analysis that joined with requirements attributes and traceability information could be a base for a "white-box" analysis. Both analysis could provide useful support for predicting the risk/cost of changing requirements.

4 CONCLUSIONS AND FURTHER WORK

The paper shows the preliminary results of investigating requirements evolution within an avionics safety-critical industrial case study. These results point out

relation with requirements evolution. The empirical investigation has identified:

- a classification of requirements evolution in terms of type of changes
- a set of outliers in relation with the policy of allocating changes into requirements
- two requirements' evolving trends, namely slow evolution and fast evolution, relating the requirements evolution with the engineering process and environmental constraints respectively
- a candidate set of points for controlling the requirements process
- a rough structure for requirements specification
- a framework for predictive changes risk/cost analyses in terms of type of changes, software life cycle, implementation of changes and the requirements structure.

All the above results suggest further works for

- verifying the hypothesis described in the paper
- modelling the framework
- calibrating the framework.

The results of this work define new directions in requirements engineering for dealing with evolving requirements. The new perspectives represent a step forward in requirements engineering. We know that at this level the results have few impact in practice, but at least they are awareness for practitioners who often forget or underrate the long term perspective of a project.

REFERENCES

- Arthur, L. J. (1992). *Rapid Evolutionary Development: Requirements, Prototyping & Software Creation*. John Wiley & Sons.
- Berry, D. M. & Lawrence, B. (1998, March). Requirements engineering. *IEEE Software*, 26–29.
- Boehm, B. W. (1981). *Software Engineering Economics*. Prentice-Hall.
- Boehm, B. W. (1984, January). Software engineering economics. *IEEE Transaction on Software Engineering* 10(1), 4–21.
- Davis, A. M. & Hsia, P. (1994, March). Giving voice to requirements engineering. *IEEE Software*, 12–16.
- Lam, W. (1997, September). Achieving requirements reuse: A domain-specific approach from avionics. *The Journal of Systems and Software* 38(3), 197–209.
- Nuseibeh, B., Kramer, J., & Finkelstein, A. (1994, October). A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering* 20(10), 760–773.

changing requirements. Deliverable 1.3, M.R. Strens (Ed.), Centre for Software Reliability, University of Newcastle upon Tyne.

RTCA (1992). *DO-178B. Software Considerations in Airborne Systems and Equipment Certification*. RTCA.

Sawyer, P., Sommerville, I., & Viller, S. (1997). Requirements process improvement through the phased introduction of good practice. *Software Process - Improvement and Practice* 3(1), 19–34.

Siddiqi, J. & Shekaran, M. (1996, March). Requirements engineering: The emerging wisdom. *IEEE Software*, 15–19.

Sommerville, I. & Sawyer, P. (1997a). *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons.

Sommerville, I. & Sawyer, P. (1997b). Viewpoints: principles, problems and a practical approach to requirements engineering. *Annals of Software Engineering*, 101–130.

Stark, G., Skillicorn, A., & Ameen, R. (1998, December). An examination of the effects of requirements changes on software releases. *CROSSTAL: Journal of Defence Software Engineering*, 11–16.

Table 1: Type of changes identified in the case study

Type of Change	Change	Description
General	add, delete, modification of requirements	Requirements are modified due to the specification process maturity and knowledge.
	explanation	The paragraphs that refer to a specific requirement are changed for clarity.
	rewording	The requirements itself does not change, but it could be rewrote for clarity.
	traceability	The traces to other deliverables are changed.
Domain Specific	non-compliance	A requirement that is not applicable for a new software package. This is the case when the requirements specification is based on that one of a previous project.
	partial compliance	A requirement that is applicable partially for a new software package. This is the case when the requirements specification is based on that one of a previous project.
Product Line	hardware modification	Several changes are due to hardware modifications. This type of change applies usually to hardware dependent software requirements.
	range modification	The range of the variables within the scope of a specific requirements is modified.
	add, delete, rename parameters/variables	The variables/parameters to which a specific requirement refers can change.

Table 2: Requirements attributes

Attribute	Description
Parameter/Variable	Requirements refer to specific parameters or variable on which the statement is based on.
Function/Task	Requirements define functions or tasks, which have to be performed by the system, that are stated into specification.
Statement	Within the requirements specification there are statements to clarify the meaning of specific requirements.
Track	Each Requirement has tracks to the other project deliverables. These tracks identify parent origins within documents back in the design process and descendents within further documents (e.g., design specification and software code).
Dependence	Requirements can depend on hardware components, software components and other requirements them self. There could also be temporal (in terms of project stages) dependencies.