Evolution of Requirements

Harri Huuhka

Helsinki 2nd December 2003

Seminar report

UNIVERSITY OF HELSINKI

Department of Computer Science

CONTENTS

Evolution of Requirements

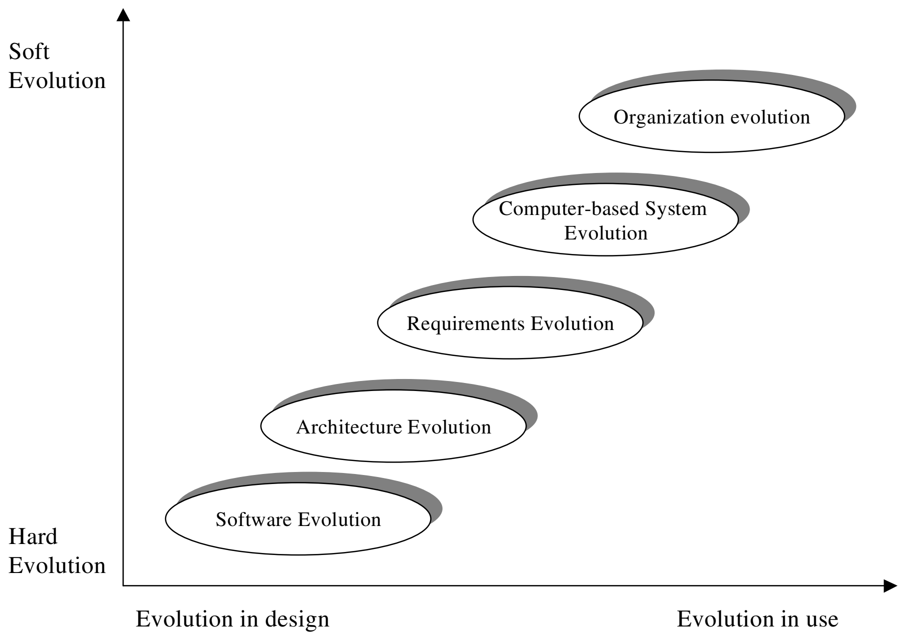
1 Introduction

In this paper requirements engineering is examined from evolutionary perspective. Software systems are bound to adapt or else ”they become progressively less satisfactory in use” [Leh 2001]. There are at least two aspects. The practical one is how to cope with evolving requirements and is closely related to concepts like management and traceability of requirements. These aspects are covered in other papers in the seminar. The subject of this paper is to analyze the nature of evolution in requirements engineering. First some basic concepts of evolution are introduced. In section 3 some research reports that analyze historical data of requirements evolution are discussed. In section 4 problems of evolution are discussed briefly. The paper is concluded in section 5.

2 Concepts of evolution

Evolution of requirements refers to changes that take place in a set of requirements after inital requirements engineering phase [Ant 2001]. Thus changes in requirements that may happen in initial elicitation, analysis, specification and validation phases are not evolutionary. Changes in requirements are additions, omissions or modifications of requirements [Sta 1999].

Evolution of requirements is related to other forms of evolution in computer-based systems. In the following picture a taxonomy of elements in evolutionary space is presented [Fel 2003]:



Picture 1 : Evolutionary space [Fel 2003]

The dimension Evolution in design – Evolution in use refers to temporal dimension. The dimension Hard Evolution – Soft Evolution refers to the place where evolution takes place in the system. Software evolution captures a product viewpoint and architecture evolution captures a design viewpoint. They belong to the hard evolution which emphasizes technical aspects. Computer-based system evolution refers to the human factors and organization evolution refers to the interaction between the computer-based system and the surrounding environment. Requirements evolution stands in the middle position which ties hard and soft aspects of a computer-based system together. Therefore it is ”a natural place where to capture information about the evolution of computer-based systems” [Fel 2003].

A functional requirement is something that enables a user or in the broader sense a stakeholder to achieve her goal(s) [1]. If we want to empirically study existing software systems then requirements establish themselves as services provided by the

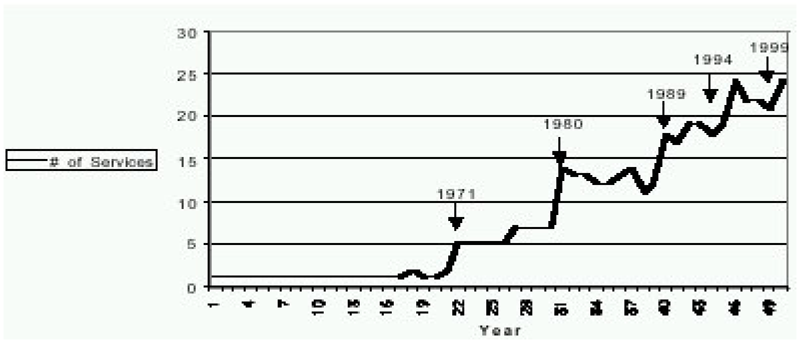
[1] More about goal-based approach to requirements engineering see [Ant 1996].

system. A service that does not support any goals or it does but at an unjustifiable cost is pointless or decorative. Sometimes a new service is a burden to a stakeholder because more harm than benefit is caused. For example if new services are unuseful to a stakeholder he may still have to change the way he interacts with the system which causes him harm. Beneficial services can be divided to core services and second order or modulating services. Core services are directly connected to the application domain whereas modulating services are based on creation or transmission of knowledge about the system. In a telephone system a core service is to be able speak via telephone and a modulating service is to know who is calling without having to answer the phone [Ant 2001][2].

3 Empirical study of requirements

One of the longest records of services in the computer based systems can be found in telephone services. In a study by Antón et al services contained in the call guide of the Atlanta telephone directories for the years 1950-1999 were tabulated. Like in evolutionary biology an interesting problem is if evolution in computer-based systems takes place gradually or rapidly in short periods of time. In the case of Atlanta telephone services there were clear jumps in the number of services in few years (1971, 1980, 1989) (see picture 2). This supports punctuated form of evolution. After a rapid increase a gradual decline in the number of services took place. Also when services were classified according their benefit/burden profiles the results still promoted punctuated evolution [Ant 2001].

[2] More about classification of services see [Ant 2001]



Picture 2. Growth in services over 50 years [Ant 2001]

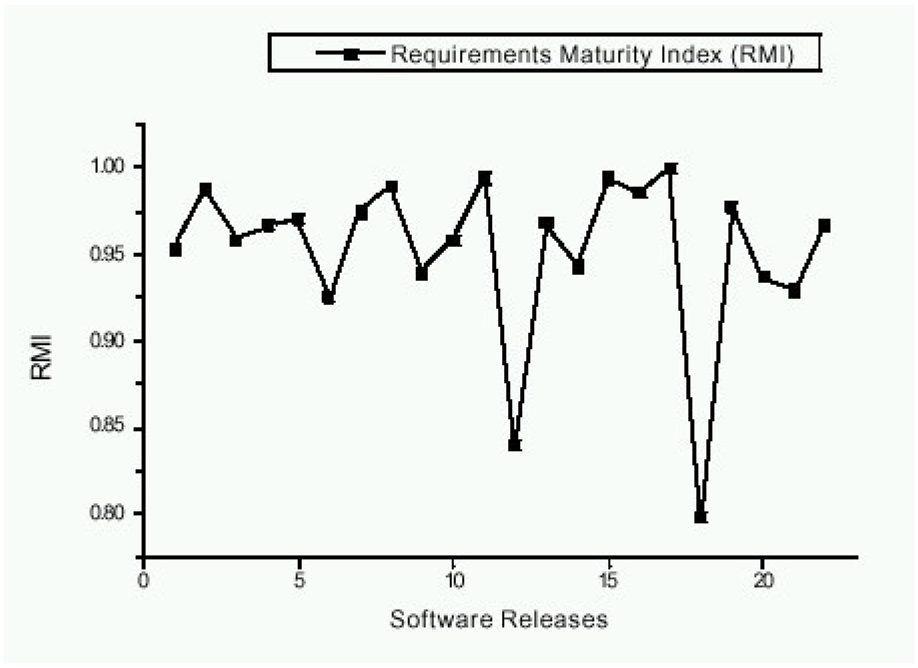
Based on previous results it can be argued that a software product is evolving either steadily longer periods or rapidly in a short period of time. This should affect how the technical and management processes are organized in the development organization. The decline in the number of services after an expansion may be explained by redundancies between new and old services, which result in displacement of old services by new ones [Ant 2001].

Another conclusion of the Atlanta case is that beneficial services for those stakeholders who use the system direcly precede services for other groups of stakeholders. In telephony first services were related to communicating with others. Being accessible to others dominated later phases of development. This suggests that requirements obtained from actor stakeholders shoud be given priority over other stakeholders. Core services that are related to the creation and transfer of information take priority over services that are based on information about core services. These meta-level services deal with information about the state of the system and are intended to help coping with complexities introduced by core services. This indicates the relative unimportance of meta-level services like customization in the early releases of the product [Ant 2001].

In another study by Anderson and Felici an avionics safety-critical system was examined [And 2001]. Like in the previous Atlanta telephone system there was great variation in the number of new services between releases. The Software Maturity Index (RMI) was used to measure the relative change in the number of requirements:

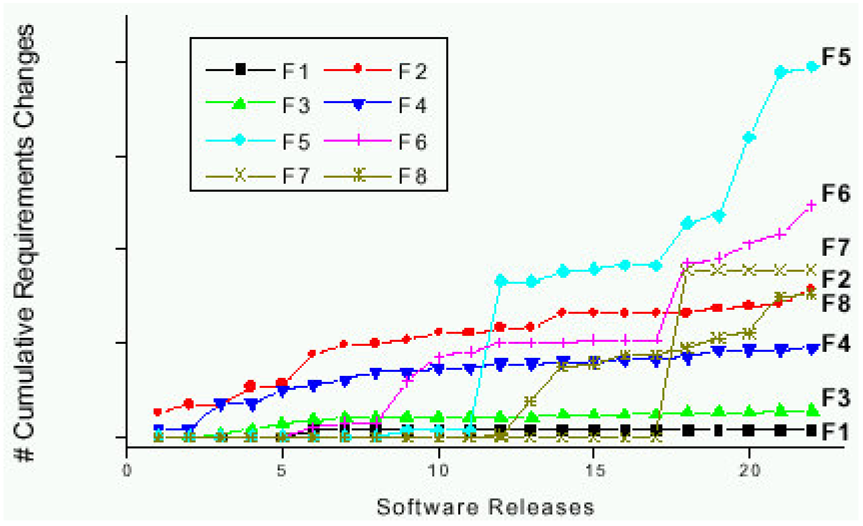
$$RMI = (R_T - R_C) / R_T$$

where R_T is the number of requirements in the current release; R_C is the number of requirements that were added, modified or deleted from a previous release. Figure 3 shows RMI of an avionics system for all the functional requirements.



Picture 3. RMI of an avionics system [And 2001]

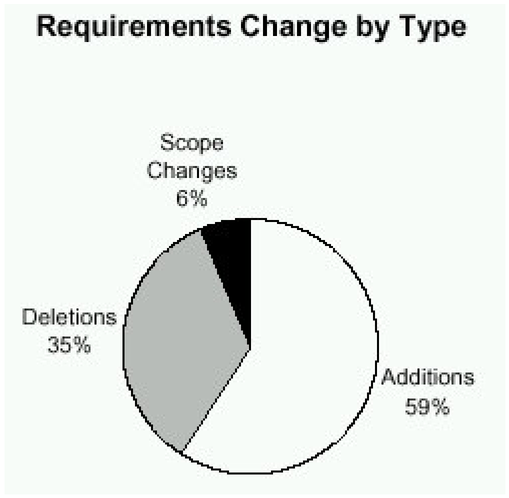
When the number of requirements changes was calculated for different 8 functions of the system the following graph was obtained:



Picture 4. Cumulative number of requirements changes for each function [And 2001]

From Picture 4 it can be observed that function F1 is a stable part of the system. F1 is responsible for interfacing the hardware architecture of the system. Therefore in my opinion even this function can demonstrate punctual evolution when the system is imported to a new hardware. It seems that functions that change in the early releases became stable later and functions that are stable in the early releases fluctuate later [And 2001].

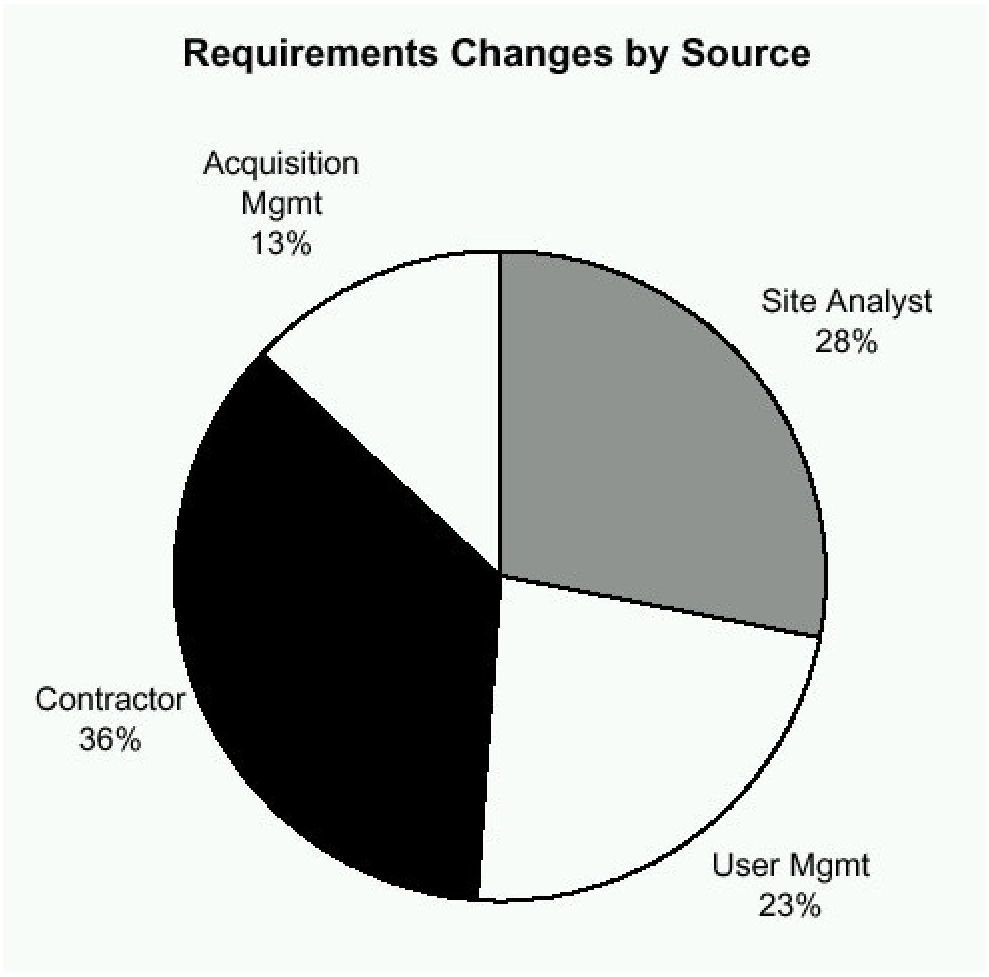
In a study by Stark et al 44 software releases spanning 7 products were examined. Requirements in all releases were listed by change type (addition, deletion and change). Picture 5 shows the distribution of these changes [Sta 1999].



Picture 5. Distribution of requirements change by type [Sta 1999]

Addition is the dominating change type as one might expect followed by deletion and modification

Requirements were also listed by stakeholders. Four groups of people participating the requirements generation process were identified: the contractor development team and the acquisition management team which belong to the maintaining organization and user management personnel and individual site analysts which belong to the client organization. The distribution is shown in picture 6 [Sta 1999].



Picture 6. Distribution of requirements by stakeholder [Sta 1999]

According to this data changes in the requirements originate equally from maintainers and customers. Other conclusions in the study were [Sta 1999]:

- Formal reviews are the most effective means to gather requirements changes
- Error handling and condition tests were the most common types of requirements changes, but new features consume the most resources
- Using data from past releases a regression model can be constructed, which helps estimating the workload associated with different sets of requirements. Since the data and the model can be verified by both maintainers and clients mutual trust can be enhanced.

4 Discussion

An obvious way to critisize empirical case studies is to claim that their results can’t reveal generalities. In evolutionary biology the classical darwinian claim is that evolution is essentialy competition. This claim can be supported by many cases in nature. But equally there is evidence that complicated biological structures like an eye are results of co-operation. So I believe that one can find software systems that have developed steadily or hardly at all after their first release. But it seems obvious that a software in a competitive environment has to evolve rapidly in order to stay in business. Usually there is a major release which contains many new features followed by minor releases that mainly fix the probles contained in the major release.

Personally I think that along with change management and requirements traceabily good initial preparation for likely types of changes is a key in succesfull evolution of a software system. In requirements engineering requirements should be classified by their likelihood of change. Also not all requirements are implemented (usually because of limited resources) and those that are not form a natural basis for evolution. This can be taken into account in the archtecture design phase so that typical nonfunctional requirement ”modifiability” is described in concrete terms. There are also methods to evaluate architectures where different stakeholders test the architecture by describing scenarios which the architecture should be able to handle. Different kinds of change scenarios are useful in testing modifiability of the architecture and in communicating the needs of change experienced by various stakeholders [Bas 2003]

5 Conclusions

According to studies presented in this paper software systems continue evolving long after their initial release. Evolution is punctual; there are long periods of steady progress and short spikes in the number of new features. One can find out if this is the case in a particular system by using simple methods described in section 3: tabulation of the number of new services per release (and possibly per function). More sophisticated regression models can be used to estimate workload associated with different sets of requirements.

References

[And 2001] Anderson S., Felici M., Requirements Evolution From Process to Product Oriented Management, In Proceedings of Profes 2001, 3rd International Conference on Product Focused Software Process Improvement, Kaiserslautern, Germany, September 10-13, 2001, LNCS 2188, Springer-Verlag, pp. 27-41.

[Ant 1996] Antón, A.I. "Goal-Based Requirements Analysis," *International Conference on Requirements Engineering (ICRE `96),* Colorado Springs, Colorado, USA, pp. 136-144, April 1996.

[Ant 2001] A.I. Antón, C. Potts: Functional Paleontology: System Evolution as the User Sees It. In: Proc. 23rd (IEEE) International Conference on Software Engineering, Toronto, Canada, 2001, 421-430.

[Bas 2003] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice Second Edition. Reading, Mass.: Addison-Wesley, 2003.

[Fel 2003] Massimo Felici, ``Taxonomy of Evolution and Dependability' ' . In Proceedings of the Second International Workshop on Unanticipated Software Evolution, USE 2003, Warsaw, Poland, 5-6 April 2003, pp. 95-104.

[Leh 2001] Lehman M.M., Ramil J.: Rules and Tools for Software Evolution Planning and Management. Annals of Software Enginerering 11: 2001, 15-44

[Sta 1999] Stark G., Oman P., Skillicorn A., Ameele R: An Examination of the Effects of Requirements Changes on Software Maintenance Releases. In Journal of Software Maintenance Research and Practice, Vol. 11, 1999 pp. 293-309.