Introduction

Nothing is more powerful in the world than an idea whose time has come.
Victor Hugo (*)

Introduction to the Topic

Usually, systems developed by humans are not developed for their own sake of existence. Instead, these systems shall help to achieve certain human goals or purposes. Goals or purposes, however, are often very abstract and vague in the same way as the usage situations of these systems are manifold and complex. Correspondingly, a more precise definition of what a system must exactly perform is needed. This leads to the need for defining the exact requirements of a system. Then, such a system must just be designed and constructed to fulfill the defined requirements.

Concerning the development of software-based systems, development experiences of the last decades have been rather disenchanting. Often, five out of six development projects are considered as rather unsuccessful [BMH+98; p.3], [St95], [St01], [Eb05; p.23ff]. One major issue identified through the years is that the developed systems often do not achieve the goals and purposes they were intended for, or if they fulfill them, the resulting system's development project significantly has exceeded planned budget and (resp. or) effort [St95], [St01].

Research on the causes for these problems is ongoing. Among others, three issues can be identified as root causes (cf. ch. I.5): Unclear requirements, often changing requirements and inadequate processes for handling.

One approach to solve the first problem is to spend extra effort on identifying and defining clear and adequate requirements upfront. Today, a whole set of artifacts, heuristics, practices and processes around the topic requirements are available summarized under the theory of *requirements engineering* (*RE*). However, development experiences have shown that even though extra focus and effort is spent upfront on the definition of requirements, changing requirements are still more the norm than the exception. As ch. I.5.6 shows, reasons are manifold.

B. Turban, *Tool-Based Requirement Traceability between Requirement and Design Artifacts,*
DOI 10.1007/978-3-8348-2474-5_1, © Springer Fachmedien Wiesbaden 2013

In the author's opinion, at least two essential causes for the requirements change problem exist:

1. Software (SW) and SW-based systems are abstract and thus essentially difficult to define comprehensively.

2. In addition, SW-based systems themselves with their intercorrelations with other systems and their embedding into processes infer a significant complexity leading to the problem that not all cases and eventualities can be considered beforehand.

These causes – among others described in ch. I.5.6 – significantly challenge the paradigm that the extensive specification and analysis of requirements upfront will tame the requirements change problem. They might rather be a good leverage to mitigate the problem, but changing requirements will still remain a decisive factor for projects. *RE*-theory also seems to have acknowledged this fact in the way that it more and more emphasizes the aspect that requirements must also be adequately managed (see ch. I.5.3). Thus, the author rather prefers to speak of *requirements engineering and management* (*REM*).

In *REM* theory, *requirements traceability* (in the following simply called *traceability*) is considered as central means to manage requirement changes. *Traceability* means "comprehensible documentation of requirements, decisions and their interdependencies to all produced information resp. artifacts from project start to project end" ([RS02; p.407 (*)]). Through recorded *traceability* information, *impact analysis* of changes is possible allowing estimating the *impact* of suggested requirement changes. This information allows project stakeholders to decide, whether the benefits of a requirement change outweigh its costs, thus avoiding disadvantageous changes. Once it is decided to perform a change, *traceability* helps to consistently propagate the change to all *impacted* locations in a project. Thus, consistently inferring the change into the project prevents dangers of forgetting to change affected locations leading to defects or even fatal consequences. In this way, the *traceability* concept is a promising means to improve *REM* and especially change management processes, thus avoiding inconsistencies – introduced during inevitably applied changes – leading to failures in the system, thus leading to significantly improved quality of developed systems.

Even though the *traceability* concept is already known for over 20 years and it always has seemed very promising to be a significant value gain in a project, it is still not very widely spread in development practice except for development projects under certain circumstances. As ch. II.10.5 tries to outline, this seems to

be the case, because it suffers from a general problem of efficiency and of low direct benefit perceived by the project members intended to capture the *traceability* information.

The quality of developed systems generally is a decisive factor. On the other side, ensuring quality involves significant efforts and costs. Even though quality must not necessarily be seen as a cost factor, but should rather be seen as a factor of investment, only finite resources can be spent for quality in order to ensure economic success. For once, this appeals to ensuring a high degree of effectiveness on quality assurance methods in general. For the other, demands for quality may differ concerning the purpose of the system. As an example, it may be an acceptable risk for PC-based SW systems that some minor bugs or other minor flaws remain undiscovered in a delivered system, because applying an update on a PC is acceptable as long as the number of updates is acceptable to the users and it is easy to apply the updates. Concerning embedded systems steering a technical equipment, it is much more difficult to perform SW-updates, as this in most cases implies a product recall to apply the new software update. Besides high costs, this is rather not acceptable for the users and often involves significant image losses for the involved companies. Beyond that, so called *safety-critical systems* exist, where a malfunction can lead to significant damages to values or even impose hazards for persons' health or lives. In these cases, even minimal probabilities of failures involving injury or death of persons must be best possibly eliminated.

Another important means to ensure good product quality is to employ good development processes. In the context of embedded projects and especially for *safety-critical* embedded projects, significant efforts have been undertaken to standardize the processes with their decisive characteristics to be performed in order to achieve high quality outcomes. Ch. I.7 describes these efforts and the demands for these processes. In these process standards, a demand crosscutting through all engineering processes is the demand for *traceability* of every requirement to the influences it imposes on every artifact developed in any engineering process.

The implementation of these demands in practice, however, often makes apparent that these demands themselves are difficult to implement and if they are implemented it is highly questionable whether the effort and resources spent really bring significant benefit to development projects. Instead, *traceability* demands are often rather performed to correspond to the standards' demands.

In this thesis, the author tries to identify several core reasons for these problems. Besides the *benefit problem* mentioned above, an essential problem is that different tools are used for different processes. This, however, implies that the *traceability* concept must somehow cross these tool gaps in order to connect the

information within the different tools. In the author's opinion, this actually is one essential cause for the *benefit problem*, as crossing these gaps generally requires higher efforts, decreases accuracy and significantly increases potentials for inconsistencies.

Unfortunately, the author considers one problem as even more essential: This problem origins from the fact that requirements describe a *problem space* that must be transformed into a solution. This transformation process is usually referred to as design. Usual *traceability* models rather assume that these connections between requirements and design artifacts are rather linear semantic allowing to trace these connections.

The author, however, believes that a semantic gap exists between the *problem space* described by requirements and the solution found. This gap exists, because design is a complex task of performing sequences of complex design decisions leading to the solution. There, the connections being rather nonlinear make it very difficult to record valuable *traceability* information.

As a way to address these problems identified, this thesis also introduces a tool environment called PROVEtech:R2A (R2A) to support *requirements traceability* to design with specific focus on diminishing both mentioned gaps. In this way, the author also hopes to diminish the benefit gap to a degree that collecting *traceability* information provides direct benefit for the designers thus hoping to really achieve the promises of the *traceability* concept.

Context of this Thesis Project

In order to provide a better understanding to the reader how the research results described in this thesis have emerged, this chapter provides a short overview about the history of this research project.

First ideas to some core problems and features addressed by R2A arose as a consequence of the direct development experiences of the author in an automotive ECU development project for lights steering with SPICE level two processes. At that time, the Micron Electronic Devices AG (MEDAG) and the Competence Center for Software Engineering (CC-SE) at the University of Applied Sciences Regensburg have begun a collaboration with the goal to improve the connection of theoretic research with industrial practice.

In the development project, from 2004 to 2005 the author worked as representative of the CC-SE at MEDAG where the author was at first responsible for

introducing *REM*-processes with the *REM-tool* IBM Rational DOORS[1] to be newly introduced into the company's project practice. During further development, the author was responsible for module design and implementation. In this way, the author was also responsible for maintaining the *requirements traceability* to the module design directly experiencing the shortcomings and problems involved.

These experiences have lead to the idea about a tool environment, where designers should directly benefit from gathered *traceability* information by making the influences of requirements on design directly visible to designers (basic ideas of ch. III.13, ch. III.15 and ch. III.18.2.2) and by improving the collaboration of all involved designers (basic ideas of ch. III.18.2.4).

In 2005 the identified key concepts have then been formulated in a theoretic outline with an extended theoretical case study being reviewed by representatives from MEDAG and CC-SE. The concepts proved promising. As the concepts also base on extensive user interaction, where usability is a key factor for success, the project made contact to the Institute for Media, Information and Cultural Studies at the University of Regensburg, where usability is one major research topic.

The three organizations have decided to form a partnership to realize the project. For this goal, the partners decided to develop a prototype tool evaluating the theoretical results by practical feedback and to apply for financial aid at the IUK[2]-program of the Bavarian Ministry of Economic Development.

During the application phase in 2006, the prototype tool implementation has been developed and has been continuously assessed by design practitioners of the partners to achieve immediate feedback of implemented features.

With these granted financial aids, a two years project for six persons could be realized to transfer the achieved theoretical and prototypical research results into a solution relevant for practice. The project has been performed from Feb. 2007 to Feb. 2009 leading to the commercial tool PROVEtech:R2A as it is discussed in this part. Because the tool's features have been considered as very innovative, where good usability at complex user interactions is essential, and because most core features have been extensively analyzed upfront by theoretical discussion and the prototype, the project members decided to develop the project using the evolutionary prototyping concept from agile development methods. Evolutionary prototyping means that the project started with a prototype where all identified features were successively integrated into the prototype so that the prototype

---

1 At that time called Telelogic DOORS

2 The IUK program (In German: Information Und Kommunikation (Information and Communication)) is a research funding program to support transferring newest research results into commercial solutions applicable in practice.

successively evolves to the final product. In this way, new features could at first be realized via a prototype implementation. These features then could be introduced to design practitioners to acquire direct feedback on the prototypical implementation. This feedback could then be used to improve and *refactor* the implementation to fully integrate it into the project's program base. Concerning the tool's architectural design, therefore, only an *architectural skeleton* has been developed sketching the core concepts of the tool environment and leaving details of the architecture open for change.

This proceeding may, at first, seem to contradict principles discussed in this thesis about *REM*, but, as discussed in ch. I.5.6 and ch. I.6.2.2, prototype-based requirement evaluation is a common practice to address the problem that highly innovative projects face a high volatility of requirements.

During the project in the midst of 2008, the MEDAG has been taken over by the MBtech Group GmbH & Co. KGaA (in the further simply called MBtech) a subsidiary company of the Daimler AG specialized on engineering services. The concepts and ideas of the project convinced the MBtech of the innovative potentials of the tool leading to a continued endeavor to develop the results to a commercial solution. In this way, the developed tool has been named PROVEtech:R2A[3] (called R2A in the following) and has been integrated into the PROVEtech tool family.

Currently, R2A is offered as commercial solution of the MBtech to address the *traceability* problems described in this thesis. It is continuously maintained and improved through a half-year release cycle. In this way, the project described here also is an example of how theoretic research results can be successfully brought into commercial project practice.

[3] R2A stands for Requirements 2 Architecture. Further information on PROVEtech:R2A can be found at the company homepage: http://www.mbtech-group.com/eu-en/electronics_solutions/tools_equipment/provetechr2a_traceability_management/traceability_management.html (Access: 2010/09).

General Remarks on this Thesis

Before stepping into the thesis, the reader should note some general remarks.

Registered Trademarks

The reader of this thesis should note that some mentioned techniques and tools referred to in this thesis are registered trademarks or under protection of copyright laws.

Argumentation

The thesis introduced here is not an empirical study, but rather a theoretical work. The work can be considered somewhere between *systems engineering* and *software engineering* theory. As a matter of fact, many of the mentioned theories and 'facts' presented in this thesis have no irrevocable evidence but are to a certain degree a 'fact' of experience, interpretation and believe. When the author collected these 'facts' from different sources, dangers of misinterpretation or selective interpretation by the author cannot be excluded. Facts found in a research paper cannot always be seen on their own. Often, these 'facts' are embedded in a certain context (e.g., a special research theory or project). Now, taking conclusions from these 'facts' should be done with a certain care. To address this problem, the author often considered not only to cite the pure 'fact' concluded somewhere, but also tried to outline the context where these 'facts' have arisen and he also tried to provide available possible alternative interpretations by other authors, or theories to allow the reader to derive his (her) own conclusions about the evidence and how cogent the author's argumentation is. As a matter of fact, however, most theories are not compatible or consistent to each other. Correspondingly, a technique to outline the context of some argumentation may also result in some inconsistency or contradictory statements. The reader should consider these inconsistencies or contradictions as phenomenon of the manifold complexity that research theories produce in their connection to each other and the limited capabilities of humans to completely cope with these complexities. Besides, the author generally doubts the potential existence of one grand unified theory about systems and software development. Rather the author considers inconsistencies and contradictions as spring of new knowledge in research.

For some of the encountered inconsistencies and contradictions the author developed suggestions or assumptions born from the author's own experience and thinking. To highlight these suggestions or assumptions, where the author could not find adequate proof derived from 'facts' basing on evidence, the author uses terms like 'the author feels', 'the author thinks', 'in the author's eyes' and 'the author believes', where these terms have an increasing weight of evidence possibility ascribed by the author.

Citations

During the work on this thesis, the author has developed a slightly individual citation practice. First of all, it is to mention that the author experienced some citation practices of other authors as unsatisfactory to really follow some argumentation. One problem, e.g., often is that some authors simply refer to an extensive text (e.g., a complete book) as an evidence for a single argument. Really retrieving the original statement is then very difficult. The author tried to make the evidence of his thoughts more explicit by referring to the exact page or at least to a collection of pages, when the evidence was rather a synthesis of several paragraphs than just a statement. Only if some more general theoretic discussion has been performed, where the whole book, or article has to be considered the author cited the source without reference to pages.

Furthermore, the author thinks that an evidence found in several sources has a higher potential to be true than originating from a single source. Correspondingly, the author also tried to mention all sources he encountered within a certain argumentation to indicate the potential evidence of the argumentation to the reader.

During writing the thesis, the author often stepped over some wordings of other authors providing a very concise or precise formulation of an argumentation, where any rewording or changes could only lower the quality of the statement or infer a falsification of the original meaning. Correspondingly, in these cases the author decided to cite these wordings verbatim to preserve the conciseness or preciseness of the argumentation for the reader.

Citing verbatim, however, invoked a further problem about quotation marks. The author used the following rules. For verbatim quoting of some other author's argumentation the author has used double quotation marks (“...”). If quotation marks were used in some verbatim quoted text, these quotation marks have been transformed to single quotations marks ('…'). In some cases, the author wanted to refer to a certain jargon-like term generally used by developers or the research

community associated with a discussed topic or to refer to a term having a doubtful connotation[4]. In these cases, the author also used single quotation marks ('…').

It is also to mention that the author is a German native speaker. In many cases, it happened that the author has read German publications with interesting passages to cite. Sometimes, even some books originally published in English have been only available in German translation. This leads to the fact that some citations were translated by the author. Any translation, however, imposes the risk of – hopefully only slightly – changing the meaning of the citation. Therefore, the author decided to mark any citation translated by himself with an asterisk surrounded in brackets ('(*)') indicating the translation by the author to the reader.

General Structure of this Thesis

This thesis is dissected into four parts. Part I tries to outline the connections of this research to other general research topics that must be considered for a tool dealing with *traceability* concerns in the context of processes for *safety-critical* projects. Afterwards, part II discusses the main research topics of interest for this thesis. These are *rationale management* and *requirements traceability*. In part III, the problems surfaced in part I and II are picked up again to outline how these problems can be solved by the innovative concepts of PROVEtech:R2A. Last but not least, part IV provides a synthesis of the results achieved and an outlook, where new ideas about further possible research are outlined.

[4] Above, e.g., the author used the connotation 'facts' to indicate that 'facts' in research are not necessarily absolute facts but are often bound to a certain paradigm. If such an paradigm is replaced by a new research paradigm, a considerable portion of 'facts' previously believed as true becomes invalid, obsolete or at least doubtful (e.g., cf. [Fe86]).