SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. (2008)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/spip.387

Rationale Modeling for Software Process Evolution

Alexis Ocampo*,† and Jürgen Münch
Fraunhofer Institute for Experimental Software Engineering
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany

Research Section

Evolving a software process without a retrospective on its evolution and, in consequence, without an appropriate understanding, can lead to important problems for a software development organization. Two examples of such problems are inefficient performance as a consequence of the arbitrary introduction of changes or difficulty in demonstrating compliance to a given standard. Capturing information on the rationale underlying process changes provides a means for better understanding process evolution. This article presents two studies aimed at understanding and identifying information needs for describing the rationale for process evolution. Additionally, it presents an approach for incrementally evolving processes supported by this rationale. This approach is derived from the results of the studies and a survey of related work. An application of the approach during the evolution of a reference process for developing service-oriented applications is presented together with experience and open questions for future research work. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: sofware process evolution; rationale; software process models; process changes; resource description framework; process management

1. INTRODUCTION

Software process models support software engineers in systematically performing the engineering processes needed to develop and maintain software products. As these models are enacted, suggestions for process adjustment or refinement arise, which, in turn, demands evolution of the models. Usually, certain events such as the introduction of a new software development technology (e.g. new testing support tools or platform), new/updated standards/guidelines for software development or process engineering, or new/updated best practices emerging from community experience generate situations that must be resolved, i.e. *issues*, by changing the engineering processes and the process models underlying them. Changing these models in organizations is typically a complex and expensive task (Nejmeh and Riddle 2006). In many cases, because of budget and time constraints, arbitrary decisions are made, and process models are evolved without storing or keeping track of the justification behind such changes (Armbrust *et al.* 2005). This, in turn, frequently results in inconsistencies or ambiguity being introduced into the process models.

The justification for a decision has been defined as *rationale* by researchers, who have done extensive investigation on capturing, organizing, and analyzing product design decisions (Dutoit *et al.* 2006). Rationale modeling is considered to be critical in two areas: knowledge capture and decision making. By making rationale information explicit, decision elements such as criteria, priorities, and arguments can improve the quality of software development decisions. Additionally, when a change in a product's functions or an addition, respectively removal,

* Correspondence to: Alexis Ocampo, Fraunhofer Institute for Experimental Software Engineering Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
†E-mail: ocampo@iese.fhg.de

is considered or implemented, rationale models enable developers to track those decisions that should be revisited and those alternatives that have already been evaluated. Dutoit *et al.* (2006) provide a description of potential uses of design rationale and design rationale methods and argue that these can also apply to any other kind of rationale-based software engineering activity apart from software design. Drawing a parallel to process evolution, some of these potential uses are the following: it supports reworking of software process model/standards; it supports identification of the potential impact area of a process change; it encourages self-reflection when making decisions; and it supports the identification and analysis of nonsystematic and rushed decisions. Additionally, we have observed during the execution of a process evolution project (Armbrust *et al.* 2005) that information about the rationale of the process evolution can help process engineers or the person(s) responsible for the process in the organization to effectively and efficiently tailor process models or update them. What matters to real-world organizations is how fit their process is regarding current and future needs. Tailoring a reference process model or a software process engineering standard can be supported by knowing about the issues, alternatives, arguments, and criteria that justified the definition of the current reference process model/standard. The process engineer can define a tailoring strategy that finds a balanced compromise for adequate process changes, taking into account both the rationale behind the current reference process model/standard and the current and future needs of his/her organization. Equally, updating a reference process model/standard whose history is known can lead to changes to process models/standards that reflect actual practices and are oriented towards satisfying current and future needs of development organizations.

The main contribution of this article consists of transferring the concepts defined by the design rationale community to the process modeling community, especially to the problem of process evolution. It is important to notice that large amounts of rationale information could become a straightjacket for the process of evolving a software process. Therefore, we followed an experience-based, bottom-up approach for defining the concepts, the classification of issues, the approach, and a tool (our specific contributions) to be used by organizations for collecting appropriate amounts of information about the rationale underlying process changes and for evolving the process in a systematic way. We have defined a roadmap for moving iteratively forward towards such concepts, classification of issues, approach, and tool. It can be summarized in the following three main steps: The first step is to understand the nature of process changes. This means understanding what is needed for describing a change and the reason for it. Additionally, we assume that by having a predefined classification of the most common situations for process changes, the task of collecting the information related to the rationale can be simplified and become more suitable for use in real process evolution projects. Once this is understood, in a second step, a structured conceptual model of rationale can be produced together with an approach that provides guidance on how to perform systematic process evolution supported by the developed conceptual model. In a final third step, the conceptual model and the approach can be applied in process evolution projects, supported by a tool. The experience acquired in trial projects is used for fine-tuning the conceptual model, the approach, and the tool before using them in new projects. So far, we have derived our current results from the experience acquired in the context of traditional, plan-driven (in Boehm's sense) processes, i.e. development is performed following a requirements/design/build paradigm with standard, well-defined processes (Boehm and Turner 2004).

On the basis of the definition of software engineering validation models provided by Zelkowitz and Wallace (1998), our contributions are the product of three different case studies performed in the context of a large and a small organization: the European Space Agency (ESA) and the Adaptive Services Grid (ASG) project, respectively. The first study was concerned with the problem of capturing the justification of changes to aerospace engineering standards (Armbrust *et al.* 2005). The second study aimed at understanding the most common issues behind process changes (Ocampo and Münch 2006). Both studies were performed in the context of the ESA. The results of these studies and the results of a search on rationale literature provided the input for a new conceptual model tested in the second organization (ASG project), which constitutes our third case study.

Table 1. Comparison of some common rationale concepts

Approach	Rationale				
Kunz and Rittel (1970)	Issue	Position	Argument		
MacLean *et al*. (1991)	Question	Option	Argument	Criterion	
Lee (1990)	Issue	Alternative	Claim	Criteria	Procedure
Bruegge and Dutoit (2004)	Issue	Alternative	Argumentation	Criteria	Resolution
Chung *et al*. (1999)	Goal	Alternative	Claim	Criteria	
Sauer (2002)			Argument prototypes	Assessment function	
Ramaesh and Dhar (1994)	Issue	Position	Argument	Assumption	Decision

This article presents the progress obtained from the research roadmap as follows: Section 2 provides short descriptions of related work, where concepts for understanding process changes and rationale were developed. Section 3 presents the context and enhanced results of our previous studies performed for understanding rationale information needs. Section 4 presents the conceptual model and approach developed incrementally with the findings of the case studies. Section 5 discusses the application of the conceptual model and approach during the evolution of a process for developing service-oriented applications in the context of the ASG project. Section 6 summarizes the most relevant findings with respect to the conceptual model and approach as well as new research questions to be addressed in the future.

2. RELATED WORK

This section discusses how research work from the design domain and the process modeling domain has addressed the problem of understanding the nature of decisions, changes, and their rationale.

Historically, much research about rationale has focused on design (Dutoit *et al*. 2006). Rationale models represent the reasoning that leads to a system, including its functionality and its implementation (Bruegge and Dutoit 2004). Kunz and Rittel (1970) developed the IBIS issue model for design rationale, which has been used as a basis for other work on rationale modeling such as Lee (1990), MacLean *et al*. (1991), Bruegge and Dutoit (2004), and Chung *et al*. (1999). This model contains three types of nodes: issues, positions, and arguments. Issues are used to state the problem. A position is a solution suggested by a developer. Arguments either support or contradict a position. The assumption made for issue modeling is that the design of a system/component is performed as follows: first, an issue is identified; then several alternatives to resolve the issue are proposed; at the same time, such alternatives are evaluated against relevant project criteria; finally, a decision is made by selecting the alternative that best matches the criteria and resolves the issue. The ideas of Kunz and Rittel were implemented by Conklin and Burgess-Yakemovic in a tool called gIBIS (Conklin and Burgess-Yakemovic 1991).

Table 1 presents our classification of some of the approaches for representing rationale information.

This comparison identifies possible commonalities among the approaches. For example, an issue for Kunz and Rittel (1970) is similar to a question or a goal for MacLean *et al.* (1991) and Chung *et al.* (1999), respectively. Kuntz and Rittel's issue modeling is the basis for most of the approaches. The MacLean *et al*. (1991), Bruegge and Dutoit (2004), and Chung *et al*. (1999) approaches propose that the rationale should be the driver for design activities. This way, rationale-related activities become a part of the software development process and not merely a parallel, additional activity. These approaches extend the issue model with concepts such as criteria/criterion/assumption, and resolution/procedure. Criteria are desirable qualities that the selection of a proposal should satisfy. For example, during requirements analysis, criteria include usability or performance, while during project management criteria include timeframe, costs, or risks. Finally, a decision is the resolution of an issue. It contains the selected proposal and the justification of the selection. At the same time, a resolution could trigger more issues. Ramaesh and Dhar (1994) developed a conceptual model and a prototype that relates the use of knowledge about design rationale to the use of knowledge in design tasks. They extended the IBIS model (Kunz and Rittel 1970) with concepts necessary for representing knowledge components involved in system design (e.g. data flow, data storage).

A common drawback of all the previous approaches is that these are prescriptive approaches that call for discussions before creating or updating a design, something that may not be necessary in cases where changes are trivial.

Sauer (2002) presents a procedure for extracting rationale information from the project log and for justifying project decisions. The concepts used are taken from the Event-Based Design Rationale Model (EDRM) developed for the Minimally Invasive Long-Term Organizational Support (MILOS) environment (Maurer *et al*. 2000). Sauer claims that by following his procedure, rationale information can be generated semiautomatically, thereby overcoming one of the major obstacles to capturing rationale, namely the costs and effort required for its elicitation. The concepts are similar to the ones presented by the IBIS issue model, with the major difference being that the information is derived from an analysis of a project trace that contains events and their interdependencies. However, Sauer's approach does not capture information about the positions/alternatives/options that were not taken into account, which is valuable information for future decisions. In other words, semiautomatic generation of possible strategies for making a decision is very helpful, but this must be complemented with the information captured from the experts who actually make the decision (e.g. actual discussion, criteria for selecting the best alternative).

In general, several design rationale concepts and approaches that provide a structured basis for systematically performing and documenting changes to a software product are available. Almost all of the approaches offer tool support (often in the form of hyper-linked information) and put an emphasis on it. Most criticism regarding these approaches is related to the costs of collecting and maintaining such rationale information (Lee 1997, Buckingham Shum *et al*. 2006). Since this prior work is highly relevant to our work, and since we are aware of its pros and cons (Bruegge and Dutoit 2004), we surveyed how process modeling approaches perform and document changes, with a focus on their similarities to the design rationale domain and on determining whether such approaches adequately address the criticism regarding cost.

We found that research work has been directed toward developing approaches (Nguyen and Conradi 1996, Ahmed 2004, Bhuta *et al*. 2005, Nejmeh and Riddle 2006, Madhavji), and process support environments (PSEs)[1] (Jaccheri and Conradi 1993, Kaba and Derniame 1995, Koskinen and Martiin 1998, Maurer *et al*. 2000, Alloui *et al*. 2001, Greenwood *et al*. 2001, Seet *et al*. 2003, Weber *et al*. 2005, Cunin *et al*. 2001), for controlling software process evolution. Some researchers propose encapsulating information about the process, its context, and its changes. Examples are process components proposed by Bhuta *et al*. (2005), dependency and change structures proposed by Madhavji, the change patterns proposed by Nguyen and Conradi (1996), update taxonomies proposed by Ahmed (2004), and the execution logs, change logs, and cases proposed by Weber *et al*. (2005).

Other investigators have proposed conceptual frameworks as in the case of Nejmeh and Riddle (2006).

AU of the approaches recognize the need for concepts and tools that can be used for collecting information about process model changes that help evolve the process in a systematic way. However, most of them have not considered rationale information as an important part of their frameworks. Those approaches that do consider concepts similar to the ones from design rationale provide little evidence (i.e. collected data) on reasons that trigger process evolution in a given context (Madhavji, Nguyen and Conradi 1996, Weber *et al*. 2005). Their research was mostly oriented toward developing structures and patterns to apply in PSEs that evolved both the instance and the model of the process.

One alternative approach called software process redesign (SPR) (Scacchi 2000) is concerned with the identification, application, and refinement of new ways to dramatically transform or improve a process. The main inputs for SPR are the so-called SPR heuristics and knowledge about the use of a process. SPR heuristics can be classified as being of two types: domain-independent SPR heuristics, which can be applied to a large set of processes, and domain-specific SPR heuristics, which can be applied only to certain processes

[1] PSEs are defined as human-oriented systems intended to serve the interaction of computerized tools and humans (Seet *et al*. 2003).

in certain circumstances. The knowledge used to derive the heuristics is obtained from narrative reports on how to dramatically improve the life cycle or prevent defects. Such narrative reports are extracted from case studies, lessons learned, best practices, and experience reports. It is here that rationale concepts can contribute to the SPR approach because they can be used for structuring and therefore improving the sources of knowledge needed for elicitation of redesign heuristics.

In summary, there has been a lot of research in the design rationale domain targeted at providing support to software engineers for making decisions about changes to the software product. There has also been a lot of research in the process modeling domain concerning the evolution of processes, with a focus on how to make changes to processes being enacted, and how to reflect such changes at the process model level. However, less emphasis has been put on the reasons for changing a process model and on how these can be used as input for future decisions that change a process. This motivates our research goal and contribution, which consist of creating a clearer and more systematic link between the domains of design rationale and software process evolution. The next section presents the results of a first step toward our goal, namely understanding rationale information needs. This has been realized by performing a study of the concepts needed for expressing the rationale for process changes and a study of the most common issues for changing a process.

3. STUDIES FOR UNDERSTANDING RATIONALE INFORMATION NEEDS

Two studies that were performed in the context of a project focused on the evolution of aerospace engineering standards are presented in this section. The first study was performed with the objective of identifying which information should be part of the description of a rationale for process changes. Additionally, a second study was performed, with the objective of identifying those most common issues that triggered changes to the process.

3.1. Context

The European Cooperation for Space Standardization (ECSS) is an initiative established to develop a coherent, single set of easy-to-use standards for all European aerospace-related engineering activities, covering all areas, including engineering, quality assurance, and project management. For the ESA, the relevant standards applicable for developing software are the following: ECSS-E-40B Space Engineering – Software (ECSS-E-40 Part 1B Space Engineering: Software – Part 1: Principles and Requirements 2003) (mostly based on the ISO 12207 standard : Information Technology – Software Life Cycle Processes International Organization for Standardization), and ECSS-Q80-B Space Product Assurance – Software (2003). Organizations or projects that are part of ESA are required to develop and use specific tailoring(s) of the ECSS standards suited to their work. This is a particularly complex task because it requires detailed understanding of the whole standard, something that an average software developer or project manager usually does not have (Ponz and Spada 2006). At the ESA Space Operations Center ESOC (the ESA organization where this project took place), this tailoring was called the Software Engineering and Management Guide (SEMG) (Jones *et al*. 2002) and was used for all their major projects.

After some years of experience with the ECSS standards, they were revised by ESA, and a new version was published. This also meant that the SEMG had to be revised, in order to be compliant with the revised ECSS standard. This compliance had to be proven by means of traceability of every ECSS requirement to its implementation and by providing a tailoring justification for every tailored requirement.

Another important task was to improve the usability of the SEMG. For the purposes of this project, process engineers (i.e. the authors of this article) considered that the ease of use of a document is positively influenced by improving (i) internal consistency, i.e. avoiding contradictions between parts of the document, (ii) external consistency, i.e. avoiding contradictions with other documents and assuring that links to external sources are correct, and (iii) conciseness, e.g. ensuring that indexed tables of contents allow people to find important things quickly, that different concepts are explained and marked clearly, and that the document is not larger than necessary.

3.2. First Study

The setting of this project allowed the definition of research goals oriented towards obtaining insights about the rationale for process changes. This section describes the specific research goal defined for the study, the way the study was performed, and its results.

3.2.1. *Goal and Operation*

The goal of the first study was to understand which type of information was needed for describing the rationale for changes to a process model/standard. In order to instrument this goal, process engineers maintained detailed meta-information tables on a per-section basis that allowed introducing the description of the change and the reason for that change. This table was to satisfy the demands of a wide variety of different stakeholders who wanted to keep track of the changes performed to the SEMG and their justifications (Armbrust *et al*. 2005).

Table 2 shows a meta-information table. The table contained not only the section's unique identifier (field invariant ID) but also a change log and a list of traceability relationships. The fields for storing the traceability relationships (ECSS Coverage section) corresponded to the analog fields in a database where this information was stored. The ECSS ID column corresponded to the requirements identifier in the ECSS standard, and the compliance column was used for selecting one out of the following: tailored, tailored out, and compliant. The justification column captured the rationale for the changes. In case the compliance changed, the justification provided the rationale for the new value. Therefore, if a requirement was compliant, i.e. was found implemented in the SEMG, but a decision was made to tailor it out, i.e. to remove its implementation from the SEMG, the description provided in the justification column supported such a decision and the changes performed to the SEMG. The opposite case can also be taken as an example. In case an ECSS requirement was not found in the SEMG, i.e. was tailored out, the rationale for implementing it in the SEMG, i.e. for making it compliant, was described in this justification column.

Table 2. Table for collecting the rationale for changes to the SEMG

Meta-information	
Invariant ID	<ID of the standard's section>
Change log	<description of changes from previous version to this version>
Reviewer's comments	<feedback of reviewers concerning the changes and rationale>

ECSS coverage		
ECSS ID	Compliance	Justification
<ID ECSS requirement>	<compliant; tailored; tailored out>	<rationale for change>
<ID ECSS requirement>	<compliant; tailored; tailored out>	<rationale for change>

The SEMG was modified iteratively and incrementally, resulting in the SETG (Tailoring of ECSS Software Engineering Standards for Ground Segments in ESA) as follows: Process engineers changed the SEMG and delivered a new version for review. Afterwards, reviewers discussed changes performed to the SEMG and accepted or rejected such changes. The reviewers documented their decisions and sent comments and suggestions to the process engineers. Process engineers reworked the SEMG on the basis of the comments and suggestions. This iterative process allowed updating the SEMG in a controlled way and enabled a continual review of the accomplishment of the tasks.

Two versions of the SEMG resulted from the editing–reviewing iterations.

3.2.2. *Study Results*

The collected rationale for changes was validated by means of reviews performed by the project partner members of the review committee and additional process engineers. They studied each rationale and provided feedback accordingly.

The validated rationale stored in a database was used as a basis for automatically generating a part of the standard called 'Part D Traceability Against ECSS' (Tailoring of ECSS Software Engineering Standards for Ground Segments in ESA). This document presents tables with traceability relationships between the SETG (old SEMG) and ECSS standards as well as the justification for changes to affected parts of the SETG standard.

The process engineers documented observations with respect to the use of the meta-information tables and their structure as a means for describing the rationale. Below are some of these observations:

The tables were used by process engineers for describing what changed and why. Sometimes the information about what changed was too detailed, sometimes too abstract. This might be due to the fact that the structure provided did not contemplate a difference between finely granular changes (e.g. grammar errors or misspellings) and larger ones (e.g. wrong control flow). Some SEMG changes collected, such as the correction of misspellings or grammar errors, or format changes, were not helpful for recognizing important decisions. However, the opposite was also observed, i.e. sets of changes that helped to identify important decisions. One example was a set of changes to the product flow, where we observed a clear attempt to separate system and software engineering activities, something that was needed in the organization. For example, many activities (such as system criticality analysis, system requirements review, system partitioning) that were performed by system engineering personnel and their produced artifacts were irrelevant for the software engineers and were still part of the SEMG. This generated a lot of frustration, because software engineers had to show evidence of artifacts they did not need or use.

The lack of structure of text fields (change log and justification) used in the meta-information table influenced the understandability of the collected information. The ESA reviewers commented on confusing justifications that contained what was performed instead of information on why. In other cases, the change log contained information about the change and about the reason in the same place. These findings motivated the need to change the conceptual model and try to use it in a new project. The resulting conceptual model and its application will be presented in Sections 4 and 5.

3.3. Second Study

Once we collected the information about the SEMG changes, we used it as the basis for a second study of the most important and common issues that were resolved by each change. This section presents the goal of the study together with its operation and results.

3.3.1. Goal and Operation

The goal of the second study was to analyze the changes performed to the SEMG for the purpose of characterizing the issue types that triggered changes in the context of the evolution of ESA standards.

We accomplished this by querying the database that contained information on changes to the SEMG and by manually analyzing each change's justification. A process ID uniquely identified the changed entity during both iterations. Each record provided information about the date of the change, the entity's name, and the change justification. First, we assigned each justification a particular situation or problem faced during the SEMG evolution. Then, we proceeded to review in internal meetings the set of situations and problems and grouped them into a list of most common issues faced while doing the SEMG evolution. A group of three process experts outside the project reviewed the list of issues in two iterations and provided feedback. Their comments were used for refining the list.

3.3.2. Study Results

The list of issues was validated by means of reviews performed by three external process experts who provided feedback. The following is the refinement of the list presented in Ocampo and Münch (2006) and an explanation of the issues:

1. Process model is inaccurate
   (a) Process engineers found that the prescribed control flow among activities differed from the one followed in real projects;
   (b) Process engineers found that the prescribed product flow differed from the one observed in real projects.
2. Process model lacks precision
   (a) Process engineers found activity descriptions that could be understood in two or more possible senses or ways;
   (b) Process engineers found examples that could be understood in two or more possible senses or ways;
   (c) Process engineers found names that did not reflect the meaning of the process element as commonly understood by practitioners. A process element can be any of the concepts used to model processes, e.g. activity, tool, role, or artifact.
3. Process model is not concise
   (a) Process engineers found activity descriptions that contained superfluous or unnecessary statements;
   (b) Process engineers found examples that contained superfluous or unnecessary statements;

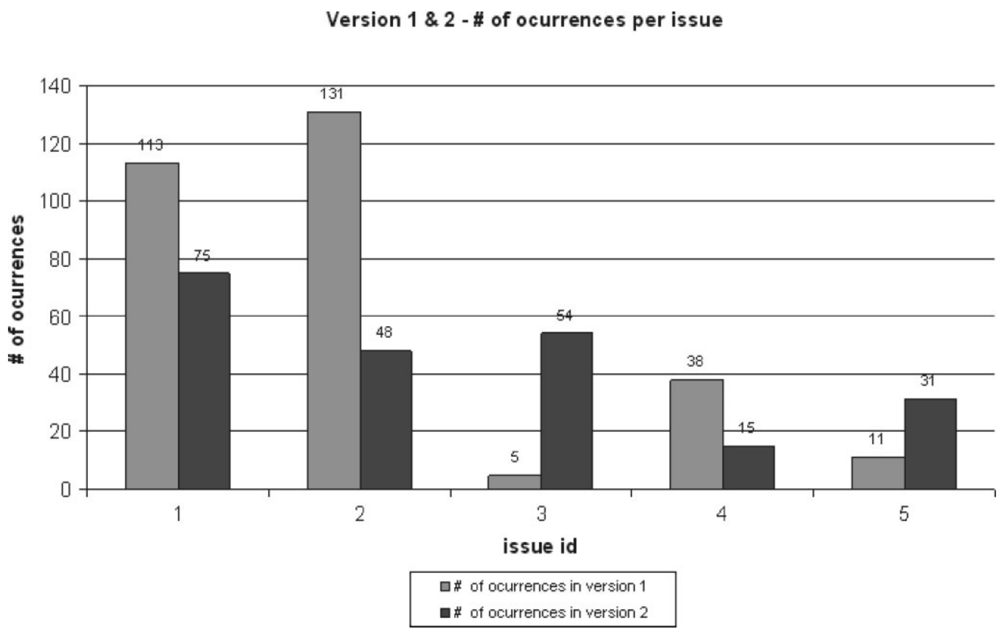

Figure 1. No. of occurrences per issue

(c) Process engineers found duplicate descriptions of activities.

4. The activity is noncompliant
   - Process engineers found cases where activities did not fulfill the requirements stated in the ECSS standards.
5. Process model is inconsistent
   - Process engineers found examples that were incorrectly referenced.

Figure 1 reflects the number of changes caused by the issues listed above when changing the SEMG during the first and second iterations. The list of issues derived from analyzing the database with the information about the evolution of process standards provides an initial insight into the type of changes performed in the context of this type of project. It can be observed that the issues that generated the highest number of occurrences such as 'process model is inaccurate' (1) and 'process model lacks precision' (2) reflect the distance that existed between the process model and the actual understanding of the process by its practitioners. A more detailed description of the results can be found in Ocampo and Münch (2006).

Unfortunately, the analysis performed for extracting the list of most common issues was not repeated afterwards by process engineers not belonging to the project. This constitutes a threat of validity to be further investigated. However, this list of issues was taken into account for the definition of the conceptual model presented in Section 4 and reused during its application described in Section 5.

4. RATIONALE CONCEPTS AND APPROACH

Although the required tracking of changes was supported during the first case study, the project gave clear indications that advanced rationale management could be beneficial. This motivated us to design a new version of the conceptual model and approach.

According to researchers in the design rationale community (Fischer *et al*. 1991, Bratthall *et al*. 2000, Burge and Brown 2004, Dutoit *et al*. 2006), maintainability and controlled evolution of a system are dependant on the understanding of what is currently present, as changes in design are affected by prior design. This means that in the case of software process evolution, one should be able to describe the relationships between an existing process model and its previous version(s). Such relationships denote differences between versions due to distinguishable modifications. The purpose of such modifications can be distinguished if one can understand the rationale behind them. Figure 2

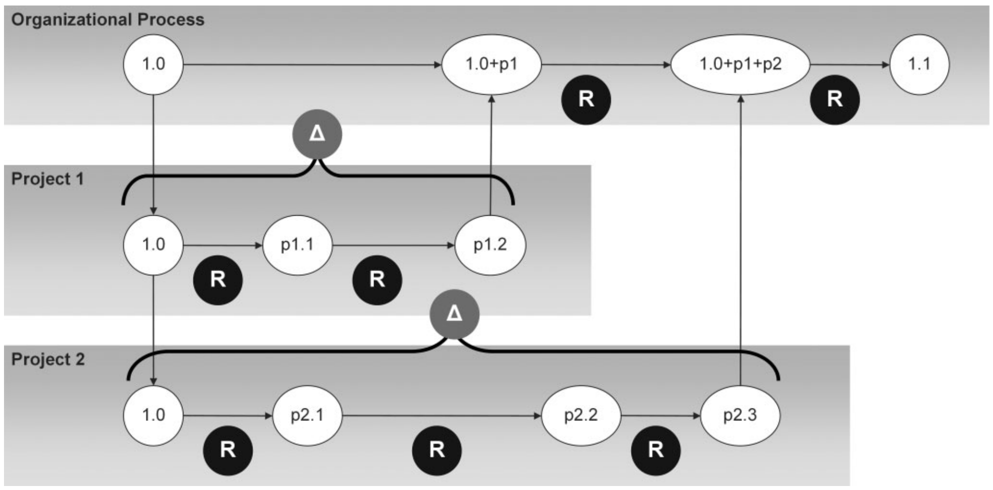

Figure 2. Rationale support for organizational process evolution

shows one scenario where rationale knowledge supports organizational process evolution. Here, an organizational process must be followed by different projects inside the organization. However, owing to the special contextual characteristics and needs, each project changes the organizational process. In this example, project 1 evolves the process and produces two new versions, while project 2 produces three new versions. At certain points in time, the organization's management or roles responsible for the process take a look at the changes performed by each project with the purpose of understanding what is new, what was not suitable, and what could be relevant for other projects. Such questions can be answered better if the reasons for changes *R* (rationale) between project process versions are known. Furthermore, the rationale available can be used as input for changing the organizational process.

Rationale knowledge provides the decision-making elements that lead to the process knowledge at both the project and organizational level (Dutoit *et al*. 2006). It also augments the process knowledge, i.e. the work required to develop the system, including knowledge about roles, resources, tasks, and work products.

According to Kneuper (2002), the knowledge representation schemes must fulfill the following characteristics:

- Expressiveness: refers to the ability to represent the desired knowledge at the appropriate levels of granularity.
- Effectiveness: concerns the means that a representation scheme provides for inferring new knowledge from old.
- Efficiency: means that the user must be able to find the knowledge needed within a reasonable time frame, and be able to find and use it with acceptable effort.
- Explicitness: allows different levels of detail or abstraction.
- Accessibility: is accessible from different technical environments.
- Modifiability: is easy to improve and adapt to changing environments.
- Understandability: is easy to understand and use by users.
- Searchability: refers to appropriate search and selection mechanisms.

One lesson learned from the first study on the evolution of SEMG standards (Armbrust *et al*. 2005) was that the information provided in the respective change logs by the process engineers who changed the standards was not sufficient for understanding the evolution.

Our conceptual model lacked expressiveness, effectiveness, efficiency, explicitness, and understandability. However, the technology that we chose supported us in providing appropriate accessibility, modifiability, and search and selection mechanisms (Armbrust *et al*. 2005).

Before the study, we were convinced that one major value of such meta-information tables is to document information to be used in the future. This was not the case, because our meta-information tables lacked the structure for describing why and where a process was changed, what the changes were, what triggered them, what the alternatives were, and which one was selected as the final decision. We arrived at the conclusion that the lack of a structure of and guidelines for the meta-information tables were the main factors leading to this confusing information. This led us to develop a second version of our conceptual model targeted at understanding the information needs for capturing the rationale behind process changes (Figure 3). We started with a small set of concepts that will be refined with time. The reason for keeping the model as simple as possible comes from the criticism regarding the high costs of capturing rationale information (Bruegge and Dutoit 2004, Burge and Brown 2004, Dutoit *et al*. 2006). We wanted to avoid these high costs and find those appropriate concepts needed to describe the rationale of changes.

We took the most common concepts used in the design rationale domain (Table 1) as a basis, customized them to our needs, and connected them to four entities that were relevant for us, i.e. event, changes, process element, and version (the non-shadowed classes in Figure 3). An *event* is considered to be the trigger of issues. Events can be external or internal. Examples are:

- External
  - New/updated process engineering technology (e.g. a new process modeling technique)
  - New/updated regulatory constraints
- Internal
  - Responses to failures to pass internal or external appraisals, assessments or reviews (e.g. changes needed to address results from a process assessment based on the ISO 15504 standard (International Organization for Standardization. ISO/IEC 15504 : 2003))
  - New/updated best practices emerging from lessons learned in just-completed projects (e.g. a new best practice approach to handling design reviews)

Changes are the result of the decision captured in the resolution. They are performed on *process elements* and are produced in a given *version*. Some examples of changes performed on process elements are: activity x has been inserted; artifact y has been deleted; activity x has been moved to be a sub-activity of activity z. The version is composed of a set of process elements.

Issues are situations that arise as a consequence of an event which need to be solved and are related to a (part of a) process. An issue contains a question, a description, a status (open, closed), and a discussion. The discussion is intended for capturing the emails, memos, letters, etc. where the issue was treated by process engineers. Additionally, an issue can be categorized by a type. This type must be selected from a classification of issues that needs to be developed or customized for an organization. The classification presented as a result of the second case study in Section 3.3 can be used as a basis,

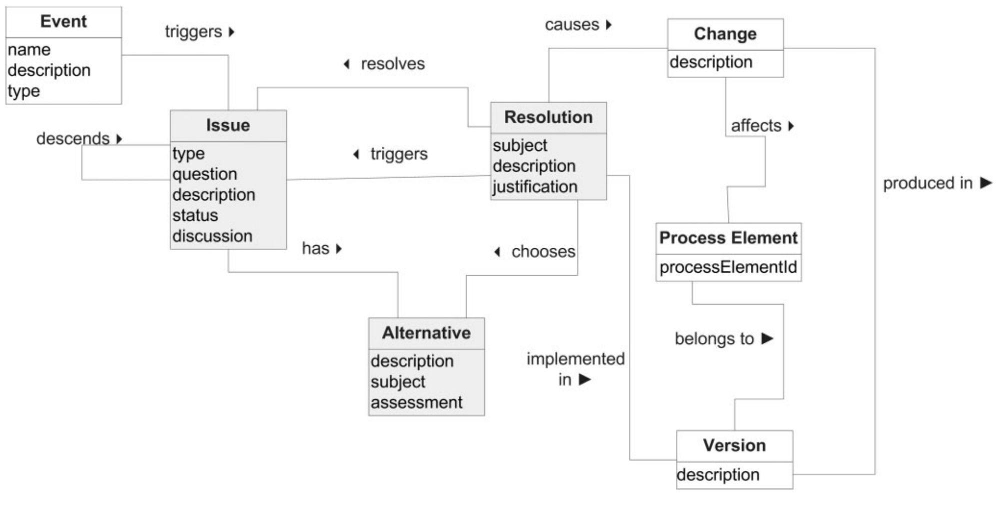

Figure 3. Rationale model (UML static structure diagram)

but it should be refined continually on the basis of experience gained from process evolution projects.

Alternatives are proposals for resolving the issue. Alternatives can be captured with subject (short description) or long descriptions. Alternatives are evaluated and assessed by process engineers regarding their impact and viability.

Finally, a *resolution* chooses an alternative whose implementation causes changes to the process elements. At the same time, one resolution could lead to opening more issues. Note that a resolution has a subject (short description), a long description, and a justification. The justification is intended for capturing a summary of the analysis of the different alternatives, the final decision, and the proposed changes. Table 3 shows an example taken from a real project, where we illustrate the concepts.

The approach we propose for evolving a process based on rationale is illustrated in Figure 4.

This approach is inspired by the existing rationale management approaches presented in Section 2 and traditional change management processes that can be found in several software development standards, e.g. (IEEE Std-828-2005). Despite its similarity to existing approaches, what is new in the approach of Figure 4 is that it attempts to explicitly describe the steps.

Table 3. Example of a rationale for process changes

Concept	Value
Event 1	- name: Process review; - type: Internal; - description: Review performed by process engineers based on interviews with practitioners.
Issue 1	- type: Process model is inaccurate; - description: The processes that describe the engineering of flight software are performed by another team of engineers.
Alternative 1	- description: Remove all processes concerned with the engineering of flight software; - assessment: Positive – It shall be clear that these processes are performed by another special team of engineers and therefore products that are to be checked during the reviews shall not be produced at this team.
Alternative 2	- description: Declare this process optional; - assessment: Negative – The engineers at this team do not follow these processes. This is not their area of competence.
Resolution 1	- description: The review board agreed to remove these processes; - justification: Process engineers agreed that alternative 1 was more appropriate because it reflects better what is being done at the team and avoids confusion during the reviews.
Changes	- Existent processes related to the engineering of flight software must be removed; - The products generated by such processes must be removed along with their templates.

First, an internal or external event that has taken place is communicated to the process engineer in charge of the process via mail, memo, official document, or by the most convenient artifact. In the activity 'identify issues', the process engineer identifies the issue(s) he or she is facing due to the event and documents them using the previously described concepts. The process engineer proceeds to 'identify conflicting issues'. He does this by querying and visualizing previous rationale information in order to find open issues that conflict with this new one. He could also identify issues treated before and how they were resolved. He also analyzes dependencies on issues, and the impact of the issues on the process model. The process engineer 'proposes alternatives' for discussing with other involved roles (e.g. quality assurer, project manager, quality manager, trainer, software developer) to resolve the issue(s). The process engineer then 'resolves the issue(s)' by selecting the most appropriate alternative(s) and defining a strategy of changes to be realized. Then, either the process engineer or the role assigned inside the organization 'extracts the process model version' to be modified and 'updates the process model' using the resolution information as input. Once the changes are finalized, the process engineer 'stores a new process model version' and closes the issue(s).

5. APPLICATION OF CONCEPTS AND APPROACH

The conceptual model and the approach have been applied for the purpose of characterization in a real environment. This has given us insights into the extent to which process engineers used the terminology to capture rationale, and the extent to which such an approach can be introduced successfully in industry.

5.1. Context

The environment where we applied our conceptual model corresponds to the ASG project (ASG:

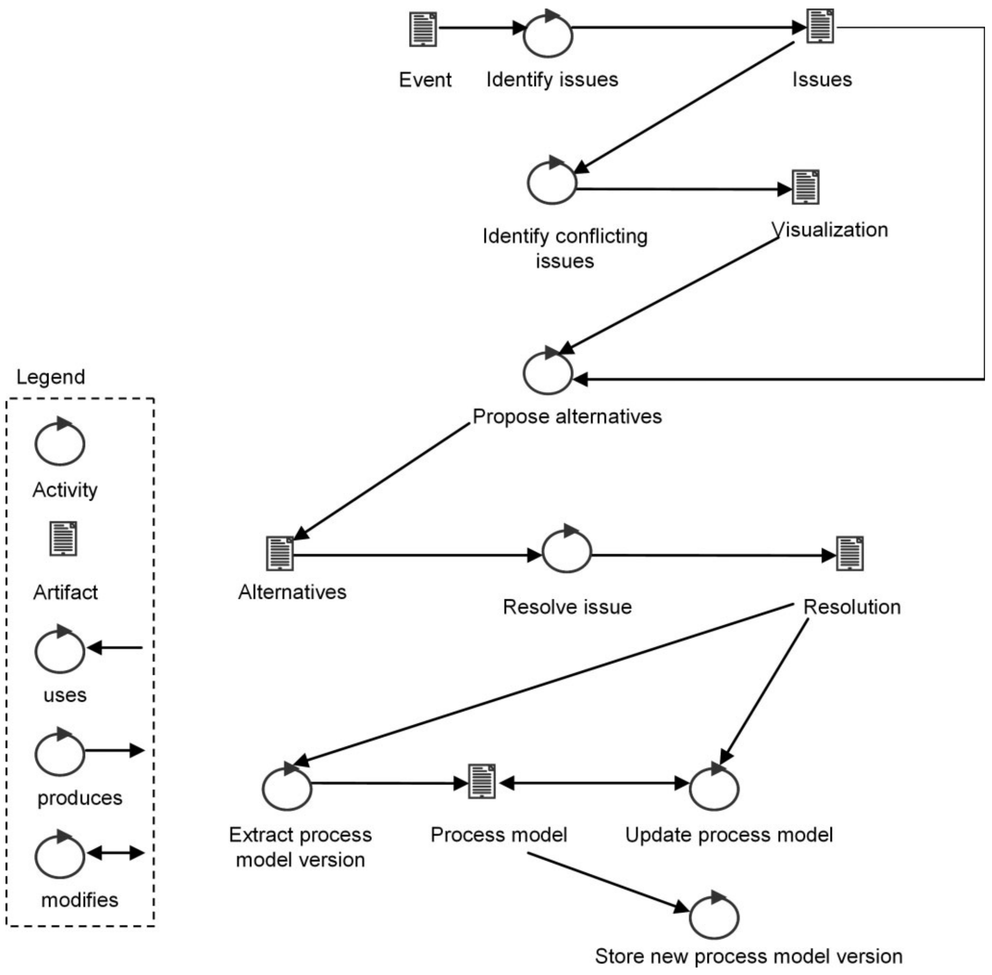

Figure 4. Approach (product flow – Spearmint notation (Becker-Kornstaedt *et al*. 1999))

Adaptive Services Grid). The ASG was intended to develop a software infrastructure that enables the design, implementation, and use of applications based on adaptive services, namely the ASG platform. Although the basic concepts of service-oriented architectures and web services have become very popular in the last few years, much confusion remains concerning strategies and processes suitable for engineering adaptive web services (Bayer *et al*. 2006). Owing to this problem, we were in charge of defining, establishing, evaluating, and systematically evolving the development process applied in the project to develop the platform. Development activities within the ASG project were performed, for instance, by several teams from different companies, universities, and research institutes. Development teams ranged from two-person teams consisting of a PhD student and a master student to ten professional programmers. Development teams were not co-located and team members spoke different native languages.

Software processes were described in terms of activities (e.g. plan project or design module), artifacts (e.g. project plan, module design), roles (e.g. project manager, designer), and assets (e.g. plan template, design guidelines). The difference between artifacts and assets is that artifacts are tangible results whereas assets are resources supporting process performance (Nejmeh and Riddle 2006). The resulting process models include both textual descriptions and diagrams that illustrate the relationships between the entities of the model in a graphical way (e.g. workflows and role-specific views).

5.2. Study Goals and Operation

The goal of this study was to analyze the conceptual model and the approach for the purpose of evaluation in the context of the evolution of the ASG reference process model.

The ASG reference process model was developed mainly in three iterations. This means that there are three versions of the model. At certain project milestones, we interviewed developers about the current process model version. Such interviews were taken as a basis for performing changes to the process model. Mapped to the previously presented conceptual rationale model (Section 4), such interviews can be seen as the event that triggered issues to improve the process. We found these interviews very useful for eliciting issues and alternatives, since developers usually provided information about possible solutions to the problem as observed in Table 4.

We discussed the interviews, decided on the changes, and documented their rationale. Then we proceeded to perform the approved changes.

In general, the main idea behind the systematic evolution approach followed was to start with commonly accepted process knowledge, refine it with information gathered from the development cycles, and therewith improve the textual descriptions and diagrams of the process according to the real project needs. We extended a tool-assisted way of editing and reviewing the process description (Armbrust *et al*. 2005) with a means for editing and visualizing rationale (Figure 5). To achieve this, we created a persistent connection of standard word processor documents containing the process description to a model of the documents in a relational database, which allowed us to keep rationale meta-information as well as automate advanced consistency checks. Our tool called 'Rationale-driven Evolution and Management Information System' (REMIS) (Ocampo and Münch 2007) also allowed us to visualize rationale information in the form of graphs before editing the process.

Our solution relied on the fact that modern word processing programs increasingly support the Extensible Markup Language (XML) as a document format (Merz 2004). As an open format, XML can be processed using a variety of widely available tools, including high-level libraries that can be invoked from most modern programming languages. Using the interpreted, object-oriented Python programming language (Lutz 2001), we developed a parser that was able to navigate through the XML versions

Table 4. Excerpt of interview with software developers

	Interview 3: software developer 1
Interview conducted in	- Tromsø, Wednesday, July 13, 2005, 11 : 00 AM–12 : 00 noon
Interviewed person	- User 1
Organization	- Organization 1
ASG subsystem	- Service Composer (C-2)
Role in prototype development	- Subsystem Manager and Integrator
Team size	- Two persons (incl. Subsystem Manager)
Team experience	- More that 10 years programming experience with several languages (object-oriented, imperative, and functional)
ASG development process	- The ASG process was followed. The ASG process description on ASG Wiki was known and used. Software developer 1 is the author of many of the linked pages for assets and tool descriptions
Process change/improvement proposals	- Owing to the short-termed synchronization points defined for the prototype development, the steps defined within the subsystem-related activities seem to be unnecessarily complex - The process description should explain how the platform, the applications, and the underlying services are developed/integrated - Role-specific views on the overall process should be provided in order to improve readability - The activities Integration and Integration Test should be merged, since they are concurrently performed by the same persons with the aid of the same supporting tools - The process description is too verbose

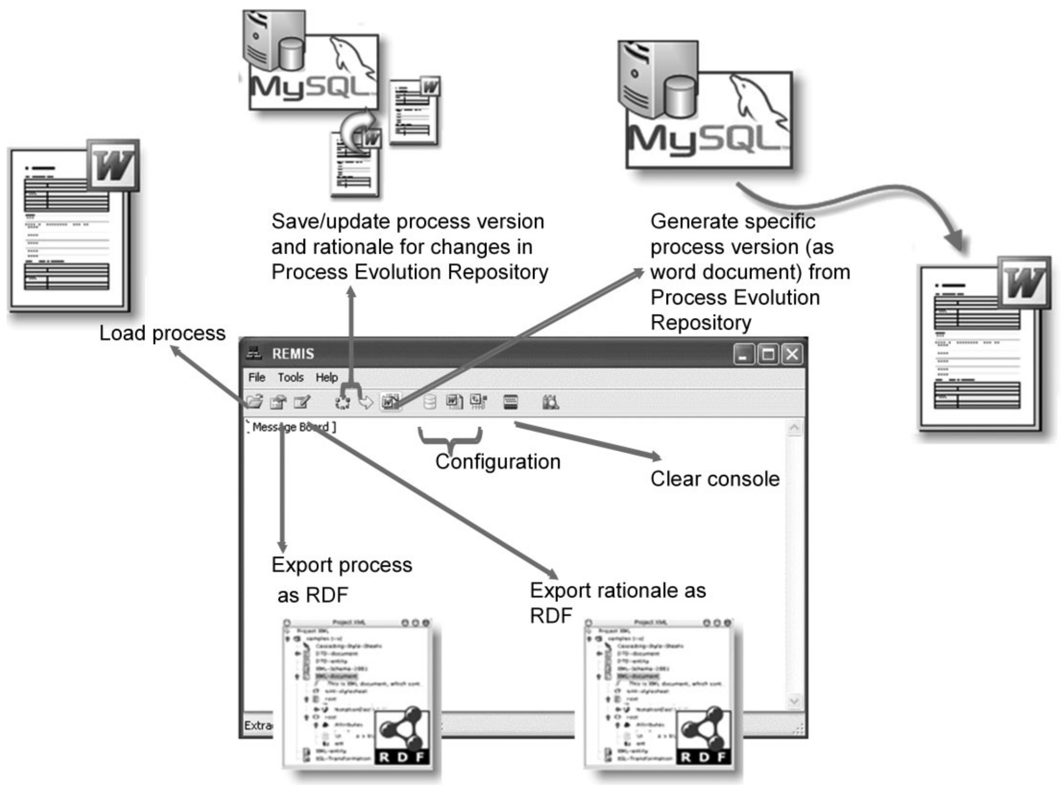
Save/update process version and rationale for changes in Process Evolution Repository
Generate specific process version (as word document) from Process Evolution Repository
Load process
REMIS
File Tools Help
Message Board
Configuration
Clear console
Export process as RDF
Export rationale as RDF
RDF

Figure 5. Tool support for process evolution supported by rationale

of the ASG process model description, identifying the section headings and rationale information tables, and moving information to and from the database as necessary. This functionality allowed us to update the database automatically after a set of changes, and to check the data for consistency before doing any further editing. We used the Resource Description Framework (RDF) as a basis for representing rationale information and process evolution information (Klyne and Carroll 2004). In brief, RDF was originally designed for representing metadata about web resources, such as the title, author, modification date of a web page, and copyright. However, it is possible to generalize the concept of 'Web Resource' and say that RDF can be used to represent 'things' that are identifiable. RDF's conceptual model allows describing 'things' by using statements and models such as nodes and arcs in a graph. We use the RDF/XML (Manola and Miller 2004) syntax for storing and querying RDF graphs in the database. We see rationale as metadata about processes. Such metadata can be queried for describing the evolution of processes. The visualization tools developed so far allow us to display rationale information in the form of graphs. This approach is similar to the one proposed by Scacchi (2000) for representing, querying, and visualizing process model information and redesign heuristics. One key difference between the two approaches lies in the way information is captured. In REMIS, the conceptual model structures the knowledge information to be captured, whereas Scacchi's approach does not provide a structure for it and references different kinds of knowledge, such as narrative reports or case studies. These are analyzed and used for deriving SPR heuristics. Another key difference is that Scacchi's approach uses inference mechanisms for deriving the SPRs, while REMIS does not use them yet for suggesting resolutions to issues that happened in the past.

5.3. Study Results

We observed that attaching the rationale and change information directly to each of the process document sections, and collecting it automatically from there, facilitated the whole evolving activity, in the context of this project. Our conceptual model played an important role, because it allowed structuring the reasoning behind a decision and motivated self-reflection about the need for changing the process. Equally, the structure of the conceptual model allowed reusing this information in a straightforward way, before performing future changes. This improves the quality of the decisions and support traceability from the changed process entity to the issue as well as knowledge transfer.

Table 5 shows an example taken from a real project, where we illustrate the concepts.

We also noted that rationale visualization can be useful for answering different types of questions relevant for process managers or quality assurers before designing the strategy to resolve issues. Examples of such questions are:

- What was the event that triggered certain issues?
- What is the type of issue that causes most changes to process elements?
- Which are the still open issues that may conflict with the resolution of a new issue?
- What was the resolution that caused certain changes to a process element?
- How many processes were affected by a resolution?
- Which are the rejected alternatives for a closed issue? Why?

Figure 6 shows a generated graph-like visualization for answering the question 'How many ASG process elements were affected by resolution 1?' (for

Table 5. Example of rationale for process changes in the ASG project

Concept	Value
Event 1	- name: Process review; - type: Internal; - description: Review of the ASG life cycle process performed by process engineers based on interviews with developers.
Issue 1	- type: Process description lacks precision; - description: The ASG platform offers capabilities such as the integration of external services and provision of services to external applications. Some groups of developers are in charge of external services, while others implement applications to exploit the power of the platform. Additionally, they are geographically distributed. The current ASG life cycle process description guides developers on platform development. However, the ASG life cycle process description does not explain how applications and external services can be developed and integrated into the platform; - question: What is the best strategy for transforming the actual process description into one suitable for platform, service, and application engineering?
Alternative 1	- description: Create one reference process model that covers all three disciplines (platform, service, and application engineering), but clearly mark those processes that belong to each one; - assessment: Positive – Reworking cost of the process description due to redundancy and inconsistency between the different disciplines can be lowered.
Alternative 2	- description: Create a process description for each engineering discipline, i.e. one for process, one for application, and one for platform engineering; - assessment: Negative – The risk of process inconsistencies, and/or redundancies because of maintenance of three similar but at the same time different disciplines can lead to high reworking costs, and low quality products.
Resolution 1	- description: The review board agreed to create one reference process model that covers all disciplines; - justification: Process engineers agreed on alternative 1; process engineers agreed that alternative 2 was less appropriate because the risk of inconsistencies and redundancies was too high. Process engineers agreed that alternative 1 was more appropriate not only because of minimizing the process reworking risks but also because one reference model can lead to a better understanding of the interactions between service, platform, and application engineering.
Changes	- existent processes were renamed with a prefix according to the discipline they belong to; - the existent activities' purposes were modified in order to differentiate that a process belongs to a certain discipline; - the product flow was modified in order to appreciate the separation among disciplines more clearly; - the involvement of roles was modified accordingly; - new activities were introduced; - new roles were created.

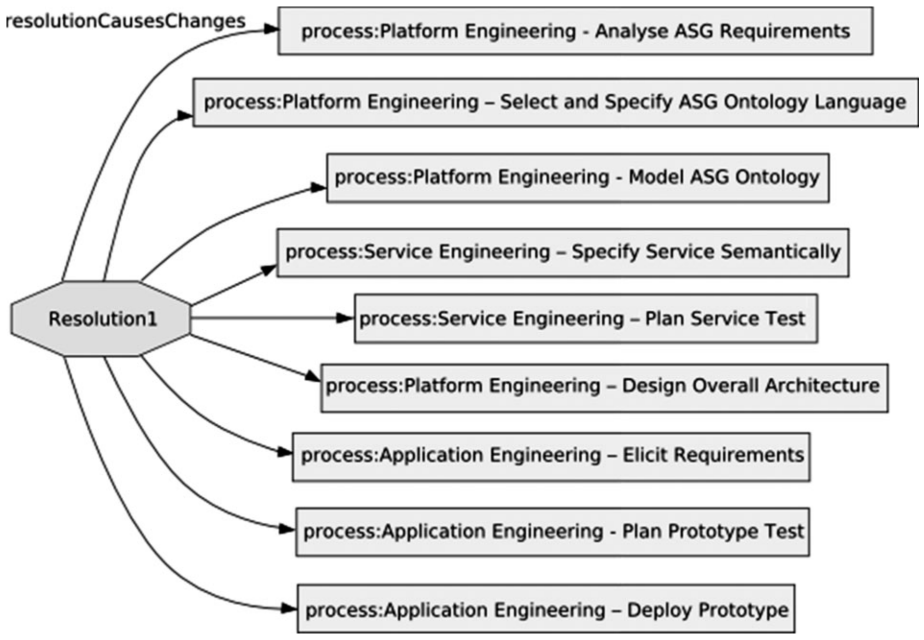
resolutionCausesChanges
Resolution1
process:Platform Engineering - Analyse ASG Requirements
process:Platform Engineering – Select and Specify ASG Ontology Language
process:Platform Engineering - Model ASG Ontology
process:Service Engineering – Specify Service Semantically
process:Service Engineering – Plan Service Test
process:Platform Engineering – Design Overall Architecture
process:Application Engineering – Elicit Requirements
process:Application Engineering - Plan Prototype Test
process:Application Engineering – Deploy Prototype

Figure 6. Impact of a resolution (partial view of a graph generated by the REMIS tool)

a description of resolution 1, see Table 5). It shows the specific name of the processes belonging to three different disciplines (platform, service, and application engineering) present in the ASG reference process, which were affected by several changes performed because of such a resolution. The alternatives, issue, type of issue, and event could also be displayed by using our tool to generate a larger graph, which also includes these concepts and their relationships.

Our tool provided us with functionalities for flexibly querying rationale information and visualizing the results of such queries. Figure 7 shows an example of such a capability with a simple visualization that helped us to answer the question 'Which are the rejected alternatives for an issue?' Having the possibility to get this visualization can be useful for avoiding inconsistencies and rework in cases where process engineers face an issue they believe to have solved in the past. Process engineers can have better means to analyze why other alternatives were rejected at the time and assess if this still continues to be the case. Further research is needed into defining common queries and their most appropriate visualization.

Some observations regarding the collection of rationale gained in relevance. For example, we found that sometimes it was difficult to differentiate between the descriptions of the alternatives and the description of the resolution. We also felt there was redundancy when documenting the justification of the resolution. This led us to the discussion of the need for an additional, lighter rationale approach, because sometimes special types of issues do not require the formulation of alternatives, but rather a direct resolution (Figure 8).

In those cases, the process engineer should be able to document light rationale information after performing the changes. This can be done by using a tool that helps a process engineer to detect changes between two process model versions automatically, such as in the approach presented in Soto and Münch (2006). The comparison tool shows a summary of changes done on the work version, which serves as input for the process engineer for going through each change and documenting a light rationale consisting of short descriptions of the event, the issue, and the resolution. An example of how to collect this rationale information and connect it to the changed process elements after changes have been performed is presented in Ocampo and Soto (2007). Before we start to follow this 'light' variation of the rationale approach, we might further investigate differentiating between those issues that need a very well documented rationale and those that do not. This can be helpful

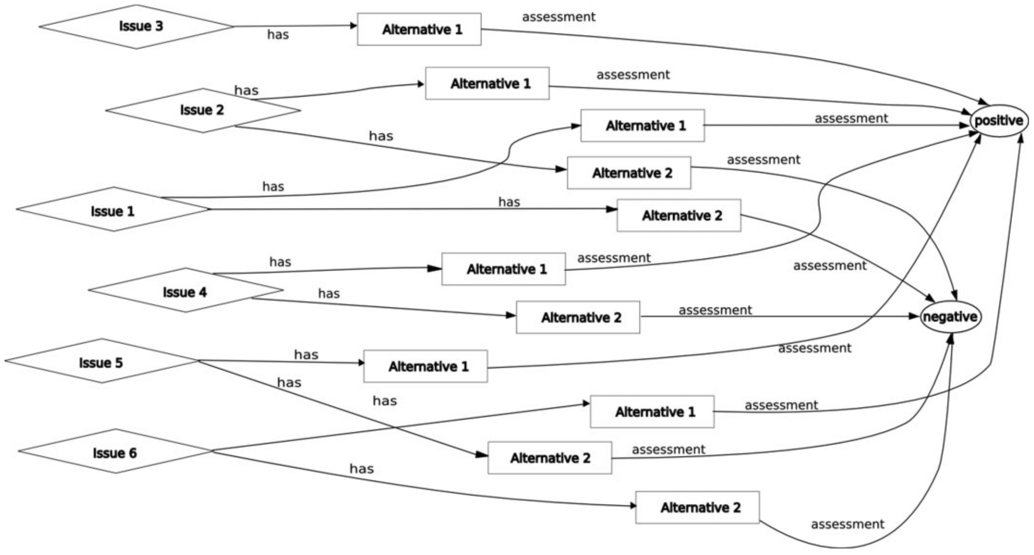

Figure 7. Issues, their alternatives and assessment (graph generated by the REMIS tool)

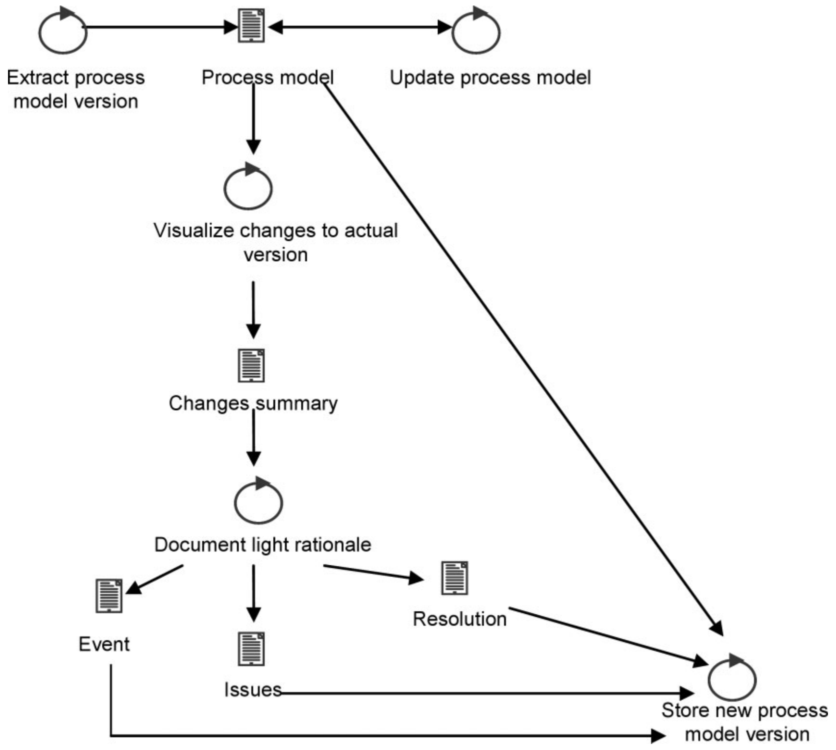
Extract process model version
Process model
Update process model
Visualize changes to actual version
Changes summary
Document light rationale
Event
Issues
Resolution
Store new process model version

Figure 8. Lighter collection of rationale information

for minimizing the risk of over-expenditure and addressing the criticism associated with capturing rationale information (Burge and Brown 2004, Dutoit *et al*. 2006).

6. SUMMARY AND OUTLOOK

Processes may be changed more consistently if the information about the process, its context, and the rationale of its evolution is captured. Existing approaches recognize the need for concepts and tools that can be used for collecting information about process model changes that could help evolve the process in a systematic way. We observed that most of the approaches did not consider rationale information as an important part of their work (Madhavji, Nguyen and Conradi 1996, Weber *et al*. 2005). This is one possible reason for the small amount of evidence available on the rationale of process evolution. With a predefined classification of the issues that trigger process model changes, the task of collecting the information related to such rationale becomes simpler and more suitable for use in real process evolution projects.

In this article, we have presented a classification of issues based on a study performed on data collected in change logs during the evolution of software engineering process standards from the aerospace engineering domain. More research work has to be done on describing this initial list of issues more precisely, so that they are as orthogonal as possible. More empirical data is needed for that purpose. We will analyze in a postmortem study the actual type of issues documented during the evolution of the ASG process, in order to contribute to the significance of the actual list of issues presented in this article. It is important to note that the studies used as a basis for the work have been performed mainly by the authors. This intrinsically biases the results. However, the application in the ASG project was supported by the partners involved, which provided feedback to the conceptual model and method and contributed to diminishing this bias.

We have also presented a conceptual model that can be used as a basis for collecting the rationale for changes. This conceptual model is the result of combining lessons learned from evolving aerospace engineering standards and an extensive literature survey on design rationales and software process evolution. Additionally, the approach that uses the conceptual model and a brief explanation of the tool support developed were introduced. The observations obtained from applying the approach during the evolution of a reference process for developing a software platform for adaptive services motivated us to design an additional light rationale approach to be applied in upcoming projects. This variation of the approach will again be helpful for evaluating the conceptual model and enhance it if necessary. We also decided to define a common set of queries to be displayed through visualization that empowers rationale information. We observed that having the means to visualize this information provides process engineers with powerful analysis tools that can be helpful when designing the strategy to evolve a process model. As mentioned before, not all process model changes are of value for systematic process evolution. Therefore, we need to continue experimenting (during the evolution of processes used in industrial settings) on how to capture and visualize such information in order to more effectively identify those changes (and their rationale) that impact the performance of practitioners and really make a difference on their software processes.

Furthermore, agile development practices and techniques can be used as input for enriching our contributions. According to Boehm and Turner (2004), agile practices are characterized by the following attributes: iterative (several cycles), incremental (the entire product is not delivered at once), self-organizing (teams determine the best way to handle work), and emergence (processes, principles, work structures are recognized during the project rather than predetermined). In the meantime, it can be assumed that in the context of new methodologies like agile (and also open source), which focus on the working code, the likelihood of developers spending effort on carefully evolving their processes is low. However, this remains an open issue for research in future projects. For example, it can be investigated whether our contributions can be used for enhancing self-organization and emergence based on the principle of active communication and discussion behind a rationale. Other questions to investigate could be: Can our conceptual model and approach become an important tool for effective face-to-face communication in agile practices? Could simple schems for capturing and discussing agile practice rationale with good tool support enhance the chances of acceptance?

Although agile practitioners emphasize the informality of their practices, such practices also evolve. We assume that rationale can add value by self-reflecting on the real reasons for such evolution.

Finally, we plan to relate the approach to popular standards such as CMMI (CMMi Product Team 2006) or ISO 15504 (International Organization for Standardization. ISO/IEC 15504 : 2003 2007).

ACKNOWLEDGEMENTS

We would like to thank Dr William E. Riddle for his valuable comments on the list of issues and discussion on process evolution. We thank Michael Jones and Mariella Spada from the ESOC for their reviews, support, and collaboration during the ESOC project. We also would like to thank all ASG project members who participated in the ASG process interviews, and especially Fabio Bella from Fraunhofer IESE, for eliciting and documenting the information. Additionally, we would like to thank Sonnhild Namingha from Fraunhofer IESE for editing the English version of this article. Finally, we thank the anonymous reviewers who have contributed to the improvement of this article. This work was partially supported by the German Federal Ministry of Education and Research (V-Bench Project, No.01| SE 11 A).

REFERENCES

Ahmed NM. 2004. Evolution of software processes and of their models: a multiple strategy. *Journal of Research and Practice in Information Technology* **36**(1): 9–22.

Alloui I, Cîmpan S, Oquendo F. 2001. Monitoring software process interactions: a logic-based approach. In *Proceedings of the 8th European Workshop on Software Process Technology, Lecture Notes in Computer Science*, vol. 2077, Ambriola V (ed.). Springer-Verlag: London, 39–46.

Armbrust O, Ocampo A, Soto M. 2005. Tracing process model evolution: a semi-formal process modeling approach. In *ECMDA Traceability Workshop (ECMDA-TW) 2005 – Proceedings*, Oldevik J, Aagedal J (eds): Trondheim, SINTEF ICT, 2005 (SINTEF Report STF90 AO5133) ISBN: 82-14-03813-8, 57–66.

ASG: Adaptive Services Grid. Integrated Project Supported By the European Commission, Available at: http://asg-platform.org/cgi-bin/twiki/view/Public Last checked 2008-03-05.

Bayer J, Bella F, Ocampo A. 2006. Characterization of semantic grid engineering. In *Workshop on Future Research Challenges for Software and Services*, Margaria T, Fernandez-Villacanas JL, Banti M. (eds). FRCSS 2006: Vienna, Austria, 112–124.

Becker-Kornstaedt U, Hamann D, Kempkens R, Rösch P, Verlage M, Webby R, Zettel J. 1999. Support for the process engineer: the spearmint approach to software process definition and process guidance. *Proceedings of the Eleventh Conference on Advanced Information Systems Engineering (CAISE '99), Lecture Notes in Computer Science*. Springer-Verlag: Berlin, New York, 119–133.

Bhuta J, Boehm B, Meyers S. 2005. Process Elements: Components of Software Process Architectures, Software Process Workshop, China.

Boehm BW, Turner R. 2004. Balancing agility and discipline. *A Guide for the Perplexed*. Addison-Wesley: Boston, MA.

Bratthall L, Johansson E, Regnell B. 2000. Is a design rationale vital when predicting change impact? A controlled experiment on software architecture evolution. In *Proceedings of the Second international Conference on Product Focused Software Process Improvement, Lecture Notes in Computer Science*, vol. 1840, Bomarius F, Oivo M (eds). Springer-Verlag: London, 126–139.

Bruegge B, Dutoit AH. 2004. Rationale management. *Object-oriented Software Engineering*, Using UML, Patterns, and Java, 2nd edn. Pearson Education: Upper Saddle River, NJ.

Buckingham Shum SJ, Selvin AM, Sierhuis M, Conklin J, Haley CB, Nuseibeh B. 2006. Hypermedia support for argumentation-based rationale: 15 Years on from gIBIS and QOC. Rationale management in software engineering: concepts and techniques. In *Rationale Management in Software Engineering*, Dutoit A, McCall R, Mistrík I, Paech B (eds). Springer-Verlag: Berlin, 111–129.

Burge J, Brown DC. 2004. An integrated approach for software design checking using rationale. In *Design Computing and Cognition '04*, Gero J (ed.). Kluwer Academic Publishers: Netherlands, 557–576.

Chung L, Nixon BA, Yu E, Mylopoulos J. 1999. *Non-functional Requirements in Software Engineering*. Kluver Academic: Boston, MA.

CMMi Product Team. 2006. *CMMI® for Development, Version 1.2, Improving Processes for Better Products*. Software Engineering Institute, Carnegie Mellon University, Pittsburg, USA: CMU/SEI-2006-TR-008.

Conklin J, Burgess-Yakemovic KC. 1991. A process-oriented approach to design rationale. *Human-Computer Interaction* **6**: 357–391.

Cunin P, Greenwood RM, Francou L, Robertson I, Warboys B. 2001. The PIE methodology – concept and application. In *Proceedings of the 8th European Workshop on Software Process Technology, Lecture Notes in Computer Science*, vol. 2077, Ambriola V (ed.). Springer-Verlag: London, 3–26.

Dutoit A, McCall R, Mistrík I, Paech B. 2006. Rationale management in software engineering: concepts and techniques. In *Rationale Management in Software*

Engineering, Dutoit A, McCall R, Mistrík I, Paech B (eds). Springer-Verlag: Berlin, 1–43.

ECSS-Q-80B Space Product Assurance. 2003. Standard available at http://www.ecss.nl. Last checked 2007-06-02.

ECSS-E-40 Part 1B Space Engineering: Software – Part 1: Principles and Requirements. 2003. Standard available at http://www.ecss.nl, Last checked 2007-06-02.

European Cooperation for Space Standardization (ECSS). Available at http://www.ecss.nl. Last checked 2008-03-05.

Fischer G, Lemke AC, Mccall R, Morch AI. 1991. Making argumentation serve design. *Human-Computer Interaction* **63&4**: 393–419.

Greenwood RM, Balasubramaniam D, Kirby GN, Mayes K, Morrison R, Seet W, Warboys B, Zirintsis E. 2001. Reflection and reification in process system evolution: experience and opportunity. In *Proceedings of the 8th European Workshop on Software Process Technology*, *Lecture Notes in Computer Science*, vol. 2077, Ambriola V (ed.). Springer-Verlag: London, 27–38.

IEEE Standard for Software Configuration Management Plans, IEEE Std 828 - 2005 (Revision of IEEE Std 828 -1998), New York, USA, 2005, pages 0-1-19.

ISO/IEC 15504-5:2006(E): Information technology – Process assessment – Part 5: An exemplar process assessment model. International Organization for Standardization (ISO). Genf, 2006.

ISO/IEC 12207:1995(E): Information technology – Software Life Cycle Processes. International Organization for Standardization (ISO). Genf, 1995.

Jaccheri LM, Conradi R. 1993. Techniques for process model evolution in EPOS. *IEEE Transactions on Software Engineering (Special Issue on Process Model Evolution* **19**(12): 1145–1156.

Jones M, Gomez E, Mantineo A. 2002. Mortensen U.K. Introducing ECSS Software-Engineering Standards within ESA. Practical approaches for space- and ground-segment software. ESA bulletin 111 – august. Available at: http://www.esa.int/esapub/bulletin/bullet111/chapter21_bul111.pdf.

Kaba BA, Derniame J. 1995. Transients change processes in process centered environments. In *Proceedings of the 4th European Workshop on Software Process Technology*, *Lecture Notes in Computer Science*, vol. 913, Schäfer W (ed.). Springer-Verlag: London, 255–259.

Klyne G, Carroll J (eds). 2004. World Wide Web Consortuim (W3C). *Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recommendation 10 February*, Available at: http://www.w3.org/TR/rdf-concepts/.

Kneuper R. 2002. Supporting software processes using knowledge management. *Handbook of Software Engineering and Knowledge Engineering*, Vol. II. World Scientific Publishing Co Singapore.

Koskinen M, Martiin P. 1998. Developing a customizable process modeling environment: lessons learnt and future prospects. In *Proceedings of the 6th European Workshop on Software Process Technology (September 16–18, 1998)*, *Lecture Notes in Computer Science*, vol. 1487, Gruhn V (ed.). Springer-Verlag: London, 13–27.

Kunz W, Rittel H. 1970. *Issues as Elements of Information Systems*, Working Paper No. 131. Institut für Grundlagen der Planung, Universität Stuttgart: Germany.

Lee J. 1990. A qualitative decision management system. In *Artificial Intelligence at MIT: Expanding Frontiers*, Vol. 1, Winston PH, Shellard S (eds). MIT Press: Cambridge, MA, 104–133.

Lee J. 1997. Design rationale systems: understanding the issues. IEEE Expert: Intelligent Systems and their Applications **12**(3): 78–85.

Lutz M. 2001. *Programming Python*, 2nd edn. O'Reilly & Associates: Sebastopol, CA.

MacLean A, Young RM, Belloti V, Moran T. 1991. Questions, options, and criteria: elements of design space analysis. *Human-Computer Interaction* **6**: 201–250.

Madhavji N. 1992. Environment evolution: the Prism model of changes. *IEEE Transactions on Software Engineering* **18**(5): 380–392.

Manola F, Miller E (eds). 2004. World Wide Web Consortuim (W3C). *RDF Primer W3C Recommendation*, Available at: http://www.w3.org/TR/rdf-primer/.

Maurer F, Dellen B, Bendeck F, Goldmann S, Holz H, Kötting B, Schaaf M. 2000. Merging project planning and web-enabled dynamic workflow technologies. *IEEE Internet Computing* **4**(3): 65–74.

Merz D. 2004. XML for Word Processors, IBM Developer Works: 25 February, Available at: http://www-128.ibm.com/developerworks/library/x-matters33/.

Nejmeh BA, Riddle WE. 2006. The PERFECT approach to experience-based process evolution. In *Advances in Computers*, Zelkowitz M (ed.). Amsterdam, Academic Press.

Nguyen MN, Conradi R. 1996. Towards a rigorous approach for managing process evolution. *Software*

Process Technology: 5th European Workshop, EWSPT '96, Nancy.

Ocampo A, Münch J. 2006. Process evolution supported by rationale: an empirical investigation of process changes. In *Software Process Change: International Software Process Workshop and International Workshop on Software Process Simulation and Modeling, SPW/ProSim 2006 – Proceedings*, Qing W, Wang Q, Pfahl D, Raffo DM, Wernick P. (eds). Springer-Verlag: Berlin, 334–341.

Ocampo A, Münch J. 2007. The REMIS approach for rationale-driven process model evolution. *Proceedings of the International Conference on Software Processes*. Minneapolis.

Ocampo A, Soto M. 2007. Connecting the rationale for changes to the evolution of a process. *Proceedings: PROFES 2007 (Product Focused Software Development and Process Improvement)* Riga, Latvia.

Ponz D, Spada M. 2006. Three Years of ECSS Software Standards: An Appraisal and Outlook. OPS-G Forum, Available at: http://esamultimedia.esa.int/multimedia/esoc/opsg/2006-01-20-OPS-G-Forum_sw_stnds.ppt. Last checked 2007-06-02.

Ramaesh B, Dhar V. 1994. Representing and maintaining process knowledge for large-scale systems development. *IEEE Expert, IEEE* **9**(2): 54–59.

Sauer T. 2002. Project history and decision dependencies. Diploma Thesis. University of Kaiserslautern.

Scacchi W. 2000. Understanding software process redesign using modeling, analysis and simulation. *Software Process Improvement and Practice* **5**: 183–195.

Seet W, Warboys B, Oquendo F. 2003. A compliant environment for enacting evolvable process models. In *Proceedings of the 9th European Workshop on Software Process Technology*, *Lecture Notes in Computer Science*, vol. 2786, Oquendo F (ed.), Springer-Verlag: Berlin 154–163.

Soto M, Münch J. 2006. Process model difference analysis for supporting process evolution. *Proceedings of the 13th European Conference in Software Process Improvement*, EuroSPI 2006. Springer LNCS 4257 Berlin.

Tailoring of ECSS Software Engineering Standards for Ground Segments in ESA. Available at ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/, Last checked 2008-03-05.

Tailoring of ECSS Software Engineering Standards for Ground Segments in ESA. Part D Traceability Against ECSS, Available at: ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/SETG-D1.0.pdf. Last checked 2008-03-05.

Weber B, Reichert M, Rinderle S, Wild W. 2005. Towards a framework for the agile mining of business processes. *Proceedings Workshop on Business Process Intelligence (BPI), in Conjunction with BPM 2005*, Nancy.

Zelkowitz M, Wallace DR. 1998. Experimental models for validating technology. *Computer* **31**(5): 23–31.