The REMIS Approach for Rationale-Driven Process Model Evolution

Alexis Ocampo and Jürgen Münch

Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{ocampo, muench}@iese.fraunhofer.de

Abstract. In dynamic and constantly changing business environments, the need to rapidly modify and extend the software process arises as an important issue. Reasons include redistribution of tasks, technology changes, or required adherence to new standards. Changing processes ad-hoc without considering the underlying rationales of the process design can lead to various risks. Therefore, software organizations need suitable mechanisms for storing and visualizing the rationale behind process model design decisions in order to optimally introduce future changes into their processes. This paper presents REMIS (Rationale-driven Evolution and Management Information System), a prototype tool we have developed for providing support to process engineers during the task of collecting the reasons for process changes, introducing the changes, and storing them together in a process model evolution repository. Additionally, we present lessons learned with REMIS during the evolution of a reference process model for developing service-oriented applications.

Keywords: Process evolution, rationale, process management, prototype tool, resource description framework.

1 Introduction

Process models can be used to guide developers, automate and improve processes, support management and execution, and store experience [8]. Changing these models in organizations is typically a complex and expensive task [25]. Process engineers are faced mainly with the following challenges: a) to rapidly update the process model so that the organization can keep up with its business environment; b) to introduce changes that are realizable and acceptable for practitioners; c) to introduce changes that are consistent or do not affect the process model consistency. Achieving a compromise that satisfies such challenges usually depends on the information available for rapidly judging if a change is consistent and can be easily adopted by practitioners. Having information about the rationales of the process design at hand can be of great help to process engineers for overcoming the previously mentioned challenges. Currently, the common situation is that there is a lack of support for systematically evolving process models. Combined with other facts such as budget and time pres-sure, process engineers often take shortcuts and therefore introduce unsuitable or inconsistent changes or go through a long, painful update process.

It has been shown that systematically describing the relationships between an existing process and its previous version(s) is very helpful for efficient software process model evolution [2]. Such relationships should denote differences between versions due to distinguishable modifications. One can distinguish the purpose of such modifications if one can understand the rationale behind them.

In the product domain, rationale has been defined as the justification for a decision by software product designers, who have done extensive research on capturing, organizing, and analyzing design rationales [9]. By making rationale information explicit, decision elements such as criteria, priorities, and arguments can improve the quality of software development decisions. Additionally, once new functionality is added to a system, the rationale models enable developers to track those decisions that should be revisited and those alternatives that have already been evaluated.

The situation is not much different in the process modeling domain, where this topic seems to be less developed, or not yet considered relevant by software process engineers. We are currently working on transferring rationale concepts into the process modeling domain. We do this based on the assumption that the rationale for process changes can be used for understanding the history of software process changes, for comprehensive learning, and for supporting the systematic evolution of software processes. The research roadmap we are following consists of the following steps: a) identification of a taxonomy of reasons for process change (documented in [26]); b) definition of a structured conceptual model of rationale; c) definition of a method that provides guidance on how to perform systematic process evolution supported by rationale; d) implementation of a prototype; e) validation of the concepts and the approach in process evolution projects. The steps are performed iteratively so that the experience acquired in the evolution projects can be used for fine tuning the concepts and the approach.

The research work described in this article consists of the definition of concepts and the implementation of the REMIS prototype that can be used for collecting information about the rationale underlying process changes and as tool support for systematically evolving a software process model. This is one of very few attempts per-formed so far whose goal is to connect the reasons for changes to the actual history of a process model. Section 2 presents a retrospective of work performed on rationale concepts and methods as well as on tools suitable for collecting the rationale of processes and products. Section 3 presents a characterization of the rationale support tools. Section 4 presents the current conceptual model. Section 5 describes the REMIS prototype and its most relevant features. Section 6 presents the experience and lessons learned from a practical application in industry where we used REMIS. Section 6 presents a summary and future research work.

2 A Retrospective of Rationale-Driven Approaches and Tools

The design rationale research community has invested much effort into developing concepts, methods, and techniques for capturing, retrieving, and analyzing the reasoning behind design decisions. The initiators of the field were Kunz and Rittel [15], who developed the IBIS method based on the principle that the design process for complex problems is basically a conversation among stakeholders (e.g., designers, customers, implementers). They started looking at the conversational process of arguing about complex problems during the design of buildings and cities. Each stakeholder contributed with his experience to the resolution of design issues. This approach was oriented to promote debate as a mechanism to provide a means for understanding the other's view and, in consequence, to obtain a more comprehensive view of the complex problem. IBIS led the way for approaches such as the Design Space Analysis proposed by McLean et al. [20] and better known as QOC, the Procedural Hierarchy of Issues (PHI) approach [22], and the Decision Representation Language (DRL) [17].

These approaches are called argumentation-based approaches because they focused on the activity of reasoning about a design problem and its solution [9]. Afterwards, these argumentation-based approaches were transferred to the mechanical engineering and software engineering fields and applied to design problems. Tools were considered an important factor for successfully managing rationale information. Most tools supporting argumentation-based approaches are hypertext-based systems that connect all pieces of information through hyperlinks, e.g., gIBIS [7], SYBIL [18], and the recently developed Compendium [6].

2.1 Software Engineering and Rationale

Dutoit et al. [9] introduce the term Software Engineering Rationale, claiming that this term is more useful for discussing rationale management in software engineering. They emphasize that the software development life cycle contains several activities where important decisions are taken, and where rationale plays an important role. In software engineering, most approaches have contributed to the rationale domain by providing new ideas and mechanisms to reduce the risk associated with rationale capture. Such approaches were conceived having in mind the goal of providing short-term incentives for those stakeholders who create and use the rationale. For example, SCRAM [34], an approach for requirements elicitation, integrates rationale into fictitious scenarios that are presented to users or customers so that they understand the reason for them and provide extra information. They can immediately see the use and benefit of rationale. Something similar happens in the inquiry cycle [27], which is an iterative process whose goal is to allow stakeholders and developers to work together towards a comprehensive set of requirements.

Most of the approaches developed for Software Engineering Rationale offer tool support provided as either adaptations or extensions of specific requirements and development tools, e.g., SEURAT [5], Sysiphus [10], DRIMER [29], or the Win-Win Negotiation Tool [38]. SEURAT integrates into a development environment a sort of plug-in for rationale capturing especially enhanced with an ontology of rationale terms and a rationale checking mechanism that guides developers in efficiently collecting rationale information and showing them at once the benefits of it. A similar short-term incentive strategy is adopted by Sysiphus, but in a collaborative modeling environment. DRIMER [29] is a software development process and tool for applying design patterns. It provides storage and retrieval of patterns application examples and their rationale. Designers who are looking for a pattern can better understand how to use it by looking at the rationale. Finally, the Win-Win negotiation tool, which supports the corresponding model, is an example of rationale as a driver of a software development project [4]. The set of requirements to be implemented in each iteration is decided by following the Win-Win model, where issues, i.e., disagreements between parties, with different win conditions are discussed. Options are proposed and an agreement is taken. Win conditions are prioritized and scheduled to iterations based on risks. Other examples of tools are REMAP [31] and C-ReCS [13].

2.2 Process Modeling and Rationale

Little work has been done in other areas apart from design and requirements. One of them is the process modeling area. Here, the need and value have been identified, and a couple of research initiatives have been followed with the goal of generating rationale information from project-specific process models. One approach developed by Dellen et al. [11] is Como-Kit. Como-Kit allows automatically deducing causal dependencies from specified process models. Such dependencies could be used for assessing process model changes. Additionally, Como-Kit provides a mechanism for adding justifications to a change. The Como-Kit system consists of a modeling component and a process engine. Como-kit was later integrated with the MVP approach [3]. The MVP approach consists of the MVP-L language and the MVP-E system, which supports the modeling and enactment of software processes. The result of such an integration effort was the Minimally Invasive Long-Term Organizational Support platform (MILOS) [37], [21]. MILOS enables the modeling of both algorithmic and creative processes, the collection of data for the purpose of process guidance, and experience management. Sauer presented a procedure for extracting information from the MILOS project log and for justifying project development decisions [33]. According to Sauer, rationale information could be semi-automatically generated. However, the approach does not capture information about alternatives that were taken into account for a decision.

Weber et al. [39] introduce an agile process mining framework that supports the whole process life cycle as well as continuous adaptation to change. The framework combines three different domains, namely process mining [36], adaptive process management (PM) [32], and conversational case-based reasoning (CCBR) [39]. Changes are registered in change logs during project execution. Changes can be referenced to cases in a case-base. A case represents a concrete ad-hoc modification of one or more process instances. A case consists of a textual problem description, a set of question-answer pairs, and the solution. The process engineer can provide information on the case so that future analysis for understanding the context of and the reasons for discrepancies between process models and related instances are possible.

3 Characterization of Rationale Tool Support

Table 1 provides an overview of the diversity of tools implemented for providing rationale support to a given approach.

Table 1. Tool Support

Approach	Tool/Prototype Support
Category 1	
IBIS [15]	gIBIS[7], Compendium [6]
Design Space Analysis (QOC) [20]	Compendium [6]
The Decision Representation Language (DRL) [17]	SYBIL [18]
Inquiry Cycle Potts et al.- [27]	Active HyperText Prototype [28]
Category 2	
Contribution Structures- Gotel and Finkelstein - [16]	Contribution Manager Prototype [16]
Como-Kit [11]	Como-Kit System [11]
Agile Process Mining - Weber et al. - [39]	ADEPT [32] + CBRFlow [45]
Category 3	
Hierarchy of Issues (PHI) [22]	JANUS [12], PHIDIAS [23]
REMAP - Ramesh and Dhar- [31]	REMAP System [31]
C-ReCS - Klein [13]	C-ReCS System [13]
SEURAT- Burge and Brown - [5]	SEURAT System [5]
Sysiphus- Dutoit and Paech - [10]	Sysiphus [10]
WinWin - Boehm et al.- [4]	WinWin Negotiation Tool [38]
DRIMER - Pena-Mora and Vadhavkar - [29]	SHARED-DRIMS [29]

These tools can be classified into three major categories: tools that support debate/argumentation; tools that support editing work and rationale documentation; and tools that support integrated editing work and debate/argumentation.

Category 1 - Support for debate/argumentation: The main feature of the tools in this group is to support the collaborative debate of complex problems. Rationale capture, management, and visualization are important functionalities of these tools. Visualization is implemented with graphical browsers that connect each rationale piece of information as hypertext. Usually, these tools provide a linking mechanism to reference the external artifact being discussed. Examples are: gIBIS [7], SYBIL [18], Compendium [6], and the Active Hypertext Prototype [28].

Category 2 - Support for editing work and rationale documentation: This group consists of tools that incorporate rationale as important additional information, but whose main feature is to provide support for users on the task they are performing. The front end in these tools is the specialized task editor. Possibilities for capturing, generating, visualizing, or retrieving rationale information for a given task or task element are provided. Examples are: Contribution Manager Prototype [16], Como-Kit System [11], and the integration of ADEPT [32] and CBRFlow [39].

Category 3 - Support for integrated editing work and debate/argumentation: The main rationale behind these tools is to avoid the criticism of the costs involved with capturing rationale and its intrusiveness by seamlessly integrating debate/argumentation into the collaborative work. Usually, these tools provide mechanisms to easily switch from the task editor to the rationale editor and to visualize both the task and its rationale in one place. The set of tasks and its rationale is conceived as a whole and therefore, changes to each task propagate to all users.

Some task editors are specialized according to the activity as in the case of requirements or design, while some others have attempted to provide a more “generic” task editor. Examples of such tools are: JANUS [12], PHIDIAS [23], REMAP [31], C-ReCS [13], SEURAT [5], Sysiphus [10], WinWin Negotiation Tool [38], and SHARED-DRIMS [29].

4 Process Rationale Concepts and Prototype

The following is a conceptual model that can be considered a second version of our attempt to understand the information needs for capturing the rationale behind process changes (see Figure 2). The results of our first attempt have been documented in [26]. We decided to start with a small set of concepts that will be refined in time. The reason for keeping the model as simple as possible comes from the criticism regarding the high costs of capturing rationale information. We wanted to avoid these high costs and find those appropriate concepts needed to describe the rationale for process changes.

4.1 Concepts

We decided to take the basic concepts of the argumentation-based approaches and connect them to three entities that were relevant for us, i.e., event, changes, and process element (the non-shadowed classes in Fig 1).

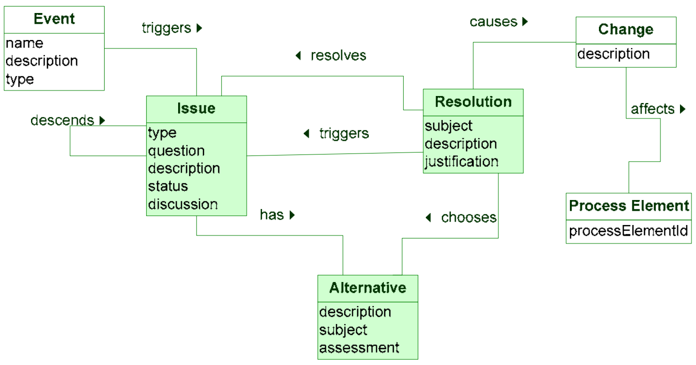


Fig. 1. Rationale Model (UML static structure diagram)

An *event* is considered to be the trigger of issues. Events can happen inside (internal) or outside (external) a given organization. Examples of different types of internal events are: new/updated process engineering technology (e.g., a new process modeling technique); new/updated regulatory constraints; Examples of different types of external events are: responses to failures to pass internal or external appraisals, assessments or reviews (e.g., changes needed to address a failure in passing an FDA audit); new/updated best practices emerging from "lessons learned" in just-completed projects (e.g., a new "best practice" approach to handling design reviews).

Issues are problems that are related to a (part of a) process and that need to be solved. Issues are stated usually as questions in product-oriented approaches. In this work, the question has the purpose of forcing process engineers to reason about the situation they are facing. Additionally, an issue also contains a long 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 can be selected from a classification of issues that needs to be developed or customized for an organization [26]. The classification can be used as a basis, which should be refined continuously based on experience gained from process evolution projects.

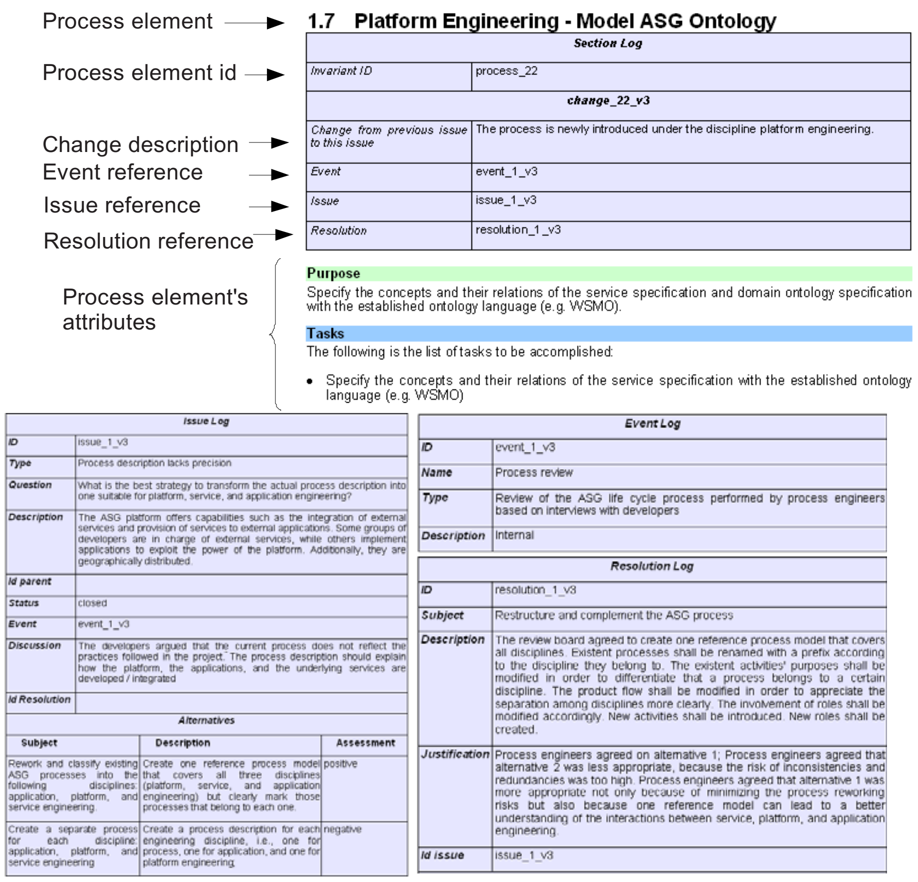
Alternatives are proposals for resolving the issue. Alternatives can be captured with subject (short description) or long descriptions. Alternatives are evaluated and assessed regarding their impact and viability by process engineers.

Finally, a *resolution* chooses an alternative whose implementation causes changes to the process models. 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 pro-posed changes. *Changes* are the result of implementing the decision captured in the resolution. They are performed on process elements. Some examples of changes performed to 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.*

4.2 Prototype and Technical Infrastructure

This section presents the current state of the REMIS prototype, which we developed with the goal of supporting our work concerning the systematic evolution of a process model. It is important to mention that ideas behind REMIS were taken from previous research work, where we developed an approach for evolving a text-based process description within the aerospace domain [2]. Our solution relies on the fact that modern word processing programs increasingly support the Extensible Markup Language (XML) as a document format [24]. 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 [19], we developed a parser that is able to navigate through the XML-specific version of the ASG process model description, identifying the section headings and rationale information tables, and moving information to and from the process evolution repository as necessary. This functionality allowed us to update a database (i.e., the process evolution repository) automatically after a set of changes, and to check the data for consistency before doing any further editing.

We have used the Resource Description Framework (RDF) as a basis for representing both process and rationale information in the process evolution repository. 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. We see the


Process element
Process element id
Change description
Event reference
Issue reference
Resolution reference
Process element's attributes

1.7 Platform Engineering - Model ASG Ontology

Section Log	
Invariant ID	process_22
change_22_v3	
Change from previous issue to this issue	The process is newly introduced under the discipline platform engineering.
Event	event_1_v3
Issue	issue_1_v3
Resolution	resolution_1_v3

Purpose

Specify the concepts and their relations of the service specification and domain ontology specification with the established ontology language (e.g. WSMO).

Tasks

The following is the list of tasks to be accomplished:

- Specify the concepts and their relations of the service specification with the established ontology language (e.g. WSMO)

Issue Log		
ID	issue_1_v3	
Type	Process description lacks precision	
Question	What is the best strategy to transform the actual process description into one suitable for platform, service, and application engineering?	
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.	
Id parent		
Status	closed	
Event	event_1_v3	
Discussion	The developers argued that the current process does not reflect the practices followed in the project. The process description should explain how the platform, the applications, and the underlying services are developed / integrated	
Id Resolution		
Alternatives		
Subject	**Description**	**Assessment**
Rework and classify existing ASG processes into the following disciplines: application, platform, and service engineering.	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.	positive
Create a separate process for each discipline: application, platform, and service engineering	Create a process description for each engineering discipline, i.e., one for process, one for application, and one for platform engineering.	negative

Event Log	
ID	event_1_v3
Name	Process review
Type	Review of the ASG life cycle process performed by process engineers based on interviews with developers
Description	Internal

Resolution Log	
ID	resolution_1_v3
Subject	Restructure and complement the ASG process
Description	The review board agreed to create one reference process model that covers all disciplines. Existent processes shall be renamed with a prefix according to the discipline they belong to. The existent activities' purposes shall be modified in order to differentiate that a process belongs to a certain discipline. The product flow shall be modified in order to appreciate the separation among disciplines more clearly. The involvement of roles shall be modified accordingly. New activities shall be introduced. New roles shall be created.
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.
Id issue	issue_1_v3

Fig. 2. Rationale Information Integrated into the Process Model

rationale as metadata about processes. Fig. 2 shows an excerpt of the ASG process model description as seen by the process engineer. It can be observed that the document contains a section for rationale information references and a section for the actual process description of attributes. The actual rationale information can be documented in special tables at the end of the document. The process engineer can then introduce the rationale information, perform the changes in the respective parts of the document, and then establish a reference to the corresponding rationale.

Such metadata can be queried for describing the evolution of processes. 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 syntax [14] for storing and querying RDF graphs in the database.

Fig. 3 presents the REMIS' user interface and the main functionality offered. This functionality supports the method we have designed so far for systematically changing a process model. Let us assume that the process engineer(s) or person(s)

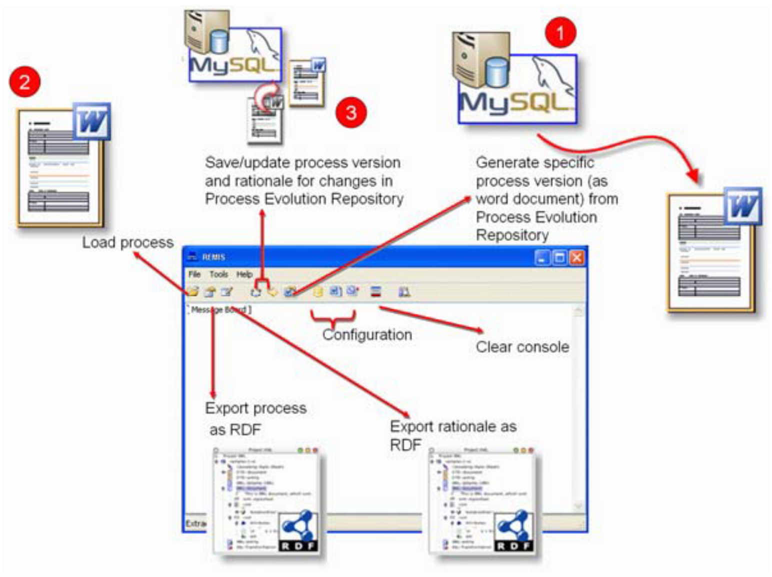

1
2
3
Save/update process version and rationale for changes in Process Evolution Repository
Generate specific process version (as word document) from Process Evolution Repository
Load process
REMIS
File Tools Help
Configuration
Clear console
Export process as RDF
Export rationale as RDF

Fig. 3. Prototype Tool Support

responsible have identified an issue. Let us assume that the process engineer(s) or person(s) responsible have identified an issue. First, he/they should extract the current version from the process evolution repository (step 1 in Fig. 3).

This can be done using the REMIS function, which, given a parameter with the desired version, generates a word document from the process evolution repository.

The process engineer can document the event, the identified issue, the alternatives proposed to solve the issue, and the resolution taken (see Fig. 2). Once this is done, the process engineer can change the process model, and reference the changes to the just documented rationale. After having performed all changes, he can use the load functionality from REMIS (step 2 in Fig. 3). Once this is done, the process engineer can update the process in the process evolution repository with the new changes. He can do this by using the REMIS functionality that updates the process information and the corresponding rationale (step 3 in Fig. 3). This functionality is implemented in such a way that REMIS compares the current version in the repository with the just loaded and changed version, identifies the changes, and stores the new version with its corresponding rationale information in the process evolution repository. We have included as additional functionality the possibility to export the process model or the rationale information to RDF models. This function was implemented for giving the user the possibility to visualize the information in RDF-capable tools such as Protegé [30] or for making queries to the RDF models with languages such as SPARQL [35].

5 Experience and Lessons Learned

The environment where we applied our conceptual model and tool corresponds to the Adaptive Services Grid (ASG) project [1]. The ASG project was intended to develop a software infrastructure that enables design, implementation, and use of applications based on adaptive services, namely the ASG platform. We were in charge of defining, establishing, evaluating, and systematically evolving the development process applied in the project to develop the platform. Development activities were performed, for instance, within the ASG project 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 collocated and team members spoke different native languages.

The software process was described in terms of activities, artifacts, roles, and tools, which are concepts that correspond to process element shown previously in Fig. 1. The resulting process model includes 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).

The ASG reference process model was developed mainly in 5 iterations. This means that there are 5 versions of the model. At certain points in time, we interviewed developers about the current process model version. Such interviews were taken as a basis for performing changes to the process model. We discussed the interviews, decided on the changes, and documented their rationale. Then we proceeded to perform the approved changes. These activities were supported by the REMIS prototype.

The final version contained 26 processes, 31 artifacts, 10 roles, and 11 tools. 353 changes to the model were performed in total from version 1 to version 5. These changes correspond to 15 issues identified from the interviews. The interviews were designed to elicit from developers important aspects such as the current problems and improvement suggestions, which could be directly mapped to the issues and alternatives. One major concern we had was the difficulty involved in first discussing and then performing the changes to the model. At the beginning, it was hard for us, acting as process engineers, to get accustomed to this way of work. However, after having discussed a couple of issues, we felt more comfortable and saw the advantages of it. REMIS assured that all relationships established in the Word document between process changes and rationale were kept and stored in the process evolution repository. REMIS also assured the consistent storage of different versions of the process model. The information stored in the process evolution repository allowed us to answer questions such as: Which process elements were affected by a change? Which process element was affected by the highest number of changes? Which issue had the largest impact on a process? Which are still unresolved issues? Which type of issues demand the highest number of changes?

Concerning the visualization, it was important for us that before changing a process model, we could see previous changes, where they were introduced, and why. This way, we could better justify our new changes. REMIS supported us by generating a given version of the process together with its rationale information as a Word document. Another mechanism to visualize the information was querying the RDF models exported from the tool, or the RDF models stored in the process evolution repository. We used internally developed tools for that purpose. We observed that the amount of information in one visualization graph or table can become overwhelming and difficult to read. We are currently investigating how to minimize this problem.

6 Summary and Outlook

This article presented the current results of our research work towards a systematic mechanism for rationale supported process evolution. In particular, we described the REMIS prototype developed for proving our conceptual model.

Despite certain difficulties arising from applying our concepts and prototype for the first time to a practical, real world project, we consider our results quite satisfactory. REMIS proved to be suitable for the set of problems at hand, and showed the potential for being applicable to future similar problems. REMIS helped process engineers in connecting the reasons for changes to the process model and in consistently storing both in one place.

We think that our technology choice played a central role for the success of this initial prototype trial. First of all, being able to produce easily processable XML documents directly from a standard Word processing application was instrumental to many of the tasks we performed in the project. On the one hand, using a standard word processor (as opposed to a specialized process modeling application) not only made our work easier and more comfortable, but also allowed us to interact with other stakeholders in a straightforward way. On the other hand, having such documents readily available in XML form provided us with a wide choice of technologies to analyze and process the data.

During our work, we identified several open research questions. One of them deals with the visualization of the large amount of information stored in the process evolution repository. We are currently investigating mechanisms that facilitate such visualization, e.g., we are trying to identify a set of "most wanted queries" based on the special interests of organizations interested in managing process evolution. Such queries can be deduced from the goals of the organization and reduce the scope of the information to be analyzed.

Acknowledgments. We would like to thank all ASG project members who participated in the interviews as well as Sonnhild Namingha from Fraunhofer IESE for preparing the English editing of this paper. This work was partially supported by the German Federal Ministry of Education and Research (V-Bench Project, No.01I SE 11 A).

References

1. ASG: Adaptive Services Grid. Integrated Project Supported By the European Commision. Available at: http://asg-platform.org/cgi-bin/twiki/view/Public
2. Armbrust O, Ocampo A, Soto M. Tracing Process Model Evolution: A Semi-Formal Process Modeling Approach. In: Oldevik, Jon (Ed.) u.a.: ECMDA Traceability Workshop (ECMDA-TW) 2005 - Proceedings. Trondheim, 2005, 57-66.
3. Bröckers A, Lott, C.M, Rombach H.D, Verlage M. MVP-L Language Report Version 2. Technical Report 265/95, Department of Computer Science, University of Kaiserslautern, Germany, 1995.
4. Bohem B, Egyed A. Kwan J, Port D, Shah A, Madachy R. Using the WinWin Spiral Model: A Case Study. IEEE Computer, vol 31, no 7, pp 33-44, 1998.

5. Burge J, Brown D.C. An Integrated Approach for Software Design Checking Using Rationale. In: Gero, J (ed) Design Computing and Cognition ’04. Kluwer Academic Publishers, Netherlands, pp. 557-576, 2004.
6. Compendium Institute. http://www.compendiuminstitute.org/.
7. Conklin J, Begeman M.L. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Information Systems, vol. 6, no. 4, pp. 303-331. 1988.
8. Curtis, B., Kellner, M. I., Over, J. 1992. Process modeling. Commun. ACM 35, 9 (Sep. 1992), 75-90.
9. Dutoit A. (Ed), McCall R. (Ed.), Mistrík I.(Ed.), Paech B. (Ed.). Rationale Management in Software Engineering. Berlin: Springer-Verlag, 2006.
10. Dutoit A, Paech B. Rationale-Based Use Case Specification. Requirements Engineering Journal. vol. 7, no 1, pp. 3-19, 2002.
11. Dellen B, Kohler K, Maurer F. Integrating Software Process Models and Design Rationales. In. Proceedings of 11th Knowledge-Based Software Engineering Conference (KBSE ’96), Syracuse, NY, pp. 84-93, 1996.
12. Fischer G, Lemke A, McCall R, Morch A. Making Argumentation Serve Design. In Moran TP, Carroll JM (eds) Design Rationale, Concepts, Techniques and Use, Lawrence Erlbaum Associates, Mahwah, NJ, 267-294.
13. Klein M. An Exception Handling Approach to Enhancing Consistency, Completeness, and Correctness in Collabora-tive Requirements Capture. Concurrent Engineering Research and Applications, vol 5, no 1, pp. 37-46. 1997.
14. Klyne, G., Carroll., J eds.: Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recom-mendation 10 February 2004. Available at: http://www.w3.org/TR/rdf-concepts/
15. Kunz W, Rittel H. Issues as Elements of Information Systems. Working Paper No. 131, Institut für Grundlagen der Plannung, Universität Stuttgart, Germany, 1970.
16. Gotel O, Finkelstein A. Contribution Structures. In: Proceedings International Symposium on Requirements Engi-neering, IEEE, York, pp. 100-107, 1995.
17. Lee, J. A Qualitative Decision Management System. In P.H. Winston & S. Shellard (eds.) Artificial Intelligence at MIT: Expanding Frontiers, Vol.1, pp. 104-133, MIT Press, Cambridge, MA, 1990.
18. Lee, J. SIBYL: a Tool for Managing Group Design Rationale. In Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work (Los Angeles, California, United States, October 07 - 10, 1990). CSCW '90. ACM Press, New York, NY, 79-92.
19. Lutz, M.: Programming Python (2nd Edition). O'Reilly & Associates, Sebastopol, California (2001)
20. MacLean A, Young R.M., Belloti V, Moran T. Questions, Options, and Criteria: Elements of Design Space Analy-sis. Human-Computer Interaction, Vol. 6, pp. 201-250, 1991.
21. Maurer F, Dellen B, Bendeck F, Goldmann S, Holz H, Kötting B, Schaaf, M. Merging Project Planning and Web-Enabled Dynamic Workflow Technologies. IEEE Internet Computing (2000), Vol. 4(3), pp.65-74.
22. McCall, R. PHIBIS: Procedural Hierarchical Issue-Based Information Systems. Proceedings of the Conference on Architecture at the International Congress on Planning and Design Theory, American Society of Mechanical Engi-neers, NY, pp. 17-22, 1987.
23. McCall R, Bennett P, D’Oronzio P, Oswald J, Shipman F.M, Wallace N. PHIDIAS: Integrating CAD Graphics into Dynamic Hypertext. In Streitz N, Rizk A, André J (eds), Hypertext: Concepts, Systems and Applications, Cam-bridge University Press, N.Y, pp. 152-165.

24. Merz, D.: XML for Word Processors. IBM Developer Works: 25 February 2004. Available at: http://www-128.ibm.com/developerworks/library/x-matters33/
25. Nejmeh B.A, Riddle W.E. The PERFECT Approach to Experience-based Process Evolution. Advances in Com-puters, M. Zelkowitz (Ed.), Academic Press, 2006
26. Ocampo A, Münch J.: Process Evolution Supported by Rationale: An Empirical Investigation of Process Changes. In: SPW/ProSim 2006 - Proceedings. Berlin Springer-Verlag Lecture Notes in Computer Science 3966, pp.,334-34, 2006.
27. Potts C, Bruns G. Recording the Reasons for Design Decisions. In Proceedings of the 10th International Conference on Software Engineering (ICSE'10). Los Alamitos, CA, pp. 418-427, 1988.
28. Potts, C. and Takahashi, K. An Active Hypertext Model for System Requirements. In Proceedings of the 7th interna-tional Workshop on Software Specification and Design (Redondo Beach, California, December 06 - 07, 1993). IEEE Computer Society Press, Los Alamitos, CA, pp. 62-68, 1993.
29. Pena-Mora F, Vadhavkar S. Augmenting Design Patterns with Design Rationale. Artificial Intelligence for Engi-neering Desgin, Analysis, and Manufacturing, vol 11, pp. 93-108, 1996.
30. Protegé: Ontology Editor. 06 January 2006. Available at: http://protege.stanford.edu/
31. Ramesh B, Dhar V. Supporting Systems Development by Capturing Deliberations During Requirements Engineer-ing. IEEE Transactions on Software Engineering, vol 18, no 6, pp. 498-510, 1992.
32. Reichert M, Dadam P. ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. Journal of Intelligent Information Systems 10, 93- 29, 1998.
33. Sauer T. Project History and Decision Dependencies. Diploma Thesis. Univer-sity of Kaiserslautern 2002.
34. Sutcliffe A, Ryan M. Experience with SCRAM, a Scenario Requirements Analysis Method. In Proceedings of the 3rd International Conference on Requirements Engineering, Colorado Springs, CO, pp. 164-173, 1998.
35. SPARQL: Query Language for RDF. 06 January 2006. Available at: http://www.w3.org/ TR/ rdf-sparql-query/
36. v.d. Aalst W, van Dongen B, Herbst J, Maruster L, Schimm G, Weijters A: Workflow mining: A Survey of Issues and Approaches. Data and Knowledge Engineering 27, 237-267 2003.
37. Verlage M, Dellen B, Maurer F, Münch J, A Synthesis of Two Process Support Approaches, Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE'96), pp. 59-68, Lake Tahoe, Nevada, USA, June 10-12, 1996.
38. WinWin. The Win Win Spiral Model. Center for Software Engineering University of Southern California http://sunset.usc.edu/research/WINWIN/winwinspiral.html
39. Weber B, Rinderle S, Wild W, Reichert, M. CCBR–Driven Business Process Evolution. Int'l Conf. on Case-Based Reasoning (ICCBR'05), pp. 610-624, Chicago, August 2005.