Dealing with User Requirements and Feedback in SOA Projects

Daniel Lübke and Eric Knauss

Leibniz Universität Hannover, FG Software Engineering
Welfengarten 1
D-30167 Hannover, Germany
{daniel.luebke,eric.knauss}@inf.uni-hannover.de
http://www.se.uni-hannover.de

Abstract. SOA projects normally influence the work of many people – especially in large organizations. The software will alter the way people will work in the future and it will hopefully support the accomplishment of their tasks. However, for building a SOA, business processes need to be formalized. Using wrong process descriptions is going to hinder instead of support people's work. Therefore, integrating the future users into the development project is crucial: Requirements need to be gathered and the system needs to be refined over time in order to improve and adapt to new situations. In this paper, we propose a methodology combined of Use Cases and an Experience Forum to better communicate with the system's users. Use Cases are used for elicitating requirements and deriving business processes in the requirements phase. Afterwards, the Experience Forum is used for collecting feedback in order to improve the system over time.

Key words: SOA, Use Case, Business Process, Experience Forum, User Feedback

1 Introduction

Service-oriented Architecture (SOA) is an emerging style for architecting large business applications. They promise to better align the business' IT with the business processes. However, so far SOA has mostly been seen as a way for executing fully automated business processes, e.g. by using the Business Process Execution Language (BPEL) [1]. In reality, users still play a central role in today's business processes: Extensions like BPEL4People [10] and generation of user interfaces [12] try to close the gap on the technical side. Because the requirements for semi-automatic business processes are heavily influenced by the users, SOA projects need to address the users' wishes during the whole software life cycle. This is even more the case in organizations which are formalizing their business processes during the "SOAfication" of their infrastructure: These organizations build up business processes in a very short time. Consequently, the business processes will likely contain errors and unwanted side effects due to their immature nature.

These problems can be addressed by analyzing and improving the project's information flows between the users and the developers. This paper will present a general information flow model for SOA projects in section 2. In this model the communication between the different parties will be enhanced by better integrating Use Cases with Business Processes in the Requirements Engineering phase (see section 3), and an Experience Forum for ongoing refinement of the system by facilitating user feedback (see section 4). Afterwards, an example project is presented in section 5, which utilizes the described techniques. Section 6 discusses how to integrate these measures into different software development processes. Finally, related work is presented and conclusions are given.

2 Information Flow Model

Software projects, and consequently SOA projects as well, are a very communication intensive endeavor. Much of projects' successes is bound to efficient and well-organized communication. In the end, every information item passed through a project can be traced back to some requirement. Consequently, the communication of requirements is essential. In order to illustrate a course-grained requirements flow through a SOA project, the FLOW notation [16] is used.

SOA projects in business environments are based on business processes. Consequently, these are an integral part of the system requirements. However, they alone are not sufficient, because they lack details from the users' points of view. This gap is closed by Use Cases [4]. Business Processes and Use Cases together form the representation of a system's functional requirements. These are passed on to the design, implementation, and testing phase. Finally, the product is used by a large, diverse and often distributed user base.

A simplified and generalized information flow model is illustrated in figure 1.

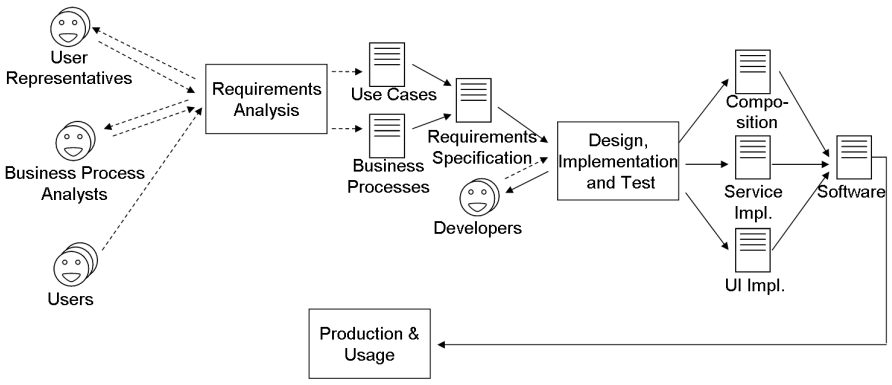

Fig. 1. Simplified FLOW Model for a General SOA Project

Concerning the requirements, there are two imminent problems in projects organized this way:

1. Both formalized business processes and Use Cases contain information about the process to be carried out. While both contain what users of a system must do they are geared towards different target groups: Use Cases are written from the perspective of a single actor while business processes offer an overview about all participants' activities. Managing both models mandates additional effort. If one model could at least be semi-automatically generated, the development team could invest more resources into actual implementation activities.
2. After delivering the first version of an application, it is necessary to collect feedback from users and incorporate the needed changes into the system. The feedback can relate to the implementation or the new business processes in place. Since SOA systems are expected to be a long-term investment, the development should be iterative in order to keep up with occurring changes. Each iteration should not only try to incorporate new functionality, but also learn from the experiences the users made with the previous versions. However, SOA projects normally serve a large user base. Reaching the users and effectively transporting their feedback into the development organization is inherently difficult.

Both problems will be addressed in the following sections.

3 Generation of Use Cases and Business Processes

Requirements Engineering (RE) is one of the core Software Engineering activities. RE aims to identify and document the requirements a system must fulfill. Over the years, many techniques have evolved in order to support RE-related activities. Among them are Use Cases which are well-suited to document functional requirements of user-centric systems. Use Cases partition the system into scenarios related to a main actor. Use Cases are normally written down in a tabular template as used in the example later on (see figure 3).

The Use Case Template is a good help to prepare interviews with users: The interviewers know what information they must elicit from the interviewees. Furthermore, due to their textual nature, Use Cases can be understood by normal users who are not accustomed to UML and other technical notations.

However, SOA projects are normally not based on Use Cases but on business processes, because processes can be easily transformed to service compositions. Therefore, the business processes must be documented as well as part of RE-related activities – if this has not already been done in the organization. Consequently, a SOA project can be initiated under two scenarios:

– The organization wants to introduce software but does not know exactly the underlying business processes. It is important to note that business processes are performed and therefore in place: Every single person knows what she

or he is supposed to do. However, a global overview is missing and no processes have been formalized. Therefore, the organization in contrast to the individuals does not know the business processes.
- The organization has defined business processes in place, and wants to support those by new software systems. This means, management and other responsible persons have the overview, but details from the users' point of view, which are necessary for implementing IT systems, are missing.

In the first scenario, Use Cases can be used to interview users and document the requirements. If those Use Cases are documented with fine-grained pre- and postconditions, the Use Cases can easily be transformed into business processes. It is mandatory to document the Use Cases with literally equal conditions in order to support automatic matching. The following algorithm for achieving a Use Case transformation to business processes in EPC notation has been presented in [11]:.

1. *Preconditions* and *triggers* are realized as events since their conditions fulfill the same role as events do. Because all preconditions must be met and the trigger must occur in order to start the Use Case, all events are joined using an AND join in the EPC if this rule results in more than one event. The first step in the main scenario is inserted after the AND join as an EPC function.
2. All steps of the *main scenario* are mapped to a linear control flow in an EPC. Each step is mapped to an EPC function. Functions are connected by using trivial OK-events. The step's actor becomes the role responsible for the created EPC function. Objects normally cannot be easily extracted from the Use Case templates and are therefore not handled by this algorithm.
3. *Success Guarantees* are – like the preconditions and triggers – conditions concerning a system. These conditions are true after the successful execution of a Use Case. They are mapped to EPC events which are inserted after the last function of the main scenario. Since all guarantees must hold after completion, all events (in case of multiple guarantees) are connected using an AND split.
4. *Minimal Guarantees* are discarded. These guarantees normally represent non-functional requirements which cannot be visualised using EPCs. Since they must be valid before, during and after the Use Case, they do not change the system at all.
5. *Extensions* are introduced using an XOR connector. After the proceeding function, an XOR split is introduced which splits into the OK-event and an start event for each extension. A step with 2 extensions therefore becomes a function followed with an XOR split to 3 paths.
6. All *Extension Steps* are handled like steps in the main scenario. *Extensions to Extension Steps* are handled recursively using these rules.
7. *Jumps* typically occurring from one extension step to a step in the main scenario are realized using XOR joins. A join is introduced before the function representing the step which is the jump target.

For the second start scenario, in which business processes are already available, the business process needs to be partitioned into Use Cases. Such Use Cases

represent sub-processes which are owned by only one actor. Only the scenario with its extensions, the main actor, and the conditions can be extracted from the business process. All other fields must be filled by interviewing the users of the envisioned system.

The main problem is to partition the business process into sub-processes belonging only to one actor. As a starting point, the whole business process can be partitioned into personal views, as presented in [8]. These sub-processes can be transformed to a scenario each. The splits in the business process, i.e. decision points and parallel activities must be mapped to extensions. In [6] an algorithm is presented, which uses logical transformations to change an EPC to a representation, that can be mapped to a Use Case scenario and corresponding extensions:

After this transformation, all Use Cases have been converted to a set of business processes. These sub-processes need to be joined into one large business process. This is done by joining the start events and end events by assuming that events with the same name are actually the same. Only the corresponding boolean connectors (e.g. a parallel and-split) have to be introduced. Two methods for doing this are covered in [11].

In order to practically create Use Cases conforming to the described constraints, tool support is necessary. Intelligent Use Case Editors, like the Use Case DODE [5], can help the Use Case creator to follow these rules by giving instant feedback. Such an editor needs to provide at least (a) the presented template, (b) must warn whenever the conditions do not match the given format, and (c) must detect similar condition names which may indicate typing errors.

4 User Feedback

Development of complex software systems is a knowledge-intensive endeavor. Developers do not only need technical skills, but also domain knowledge, user interface design capabilities, and so forth. Whenever many roles and stakeholders are involved, large projects will rarely meet all customer requirements at the first attempt. There is usually a lengthy period of building, using, and improving the system.

With SOA, such a complex system is structured in a number of independently developed services that need to cooperate as an integrated system. There is an abundance of aspects that developers need to consider when they create SOA systems or services.

However, they typically lack domain knowledge: They do not and cannot know what customers and users really want. At the same time, customers and users do not know what could reasonably be expected from such a system. Fischer has called this the “symmetry of ignorance” [7]. Due to the wicked nature of the problem, and the symmetry of ignorance, several iterations and improvement steps will be needed to produce a satisfactory SOA system [15].

The problem is even harder with most SOA projects, because the software serves a large and often distributed user base.

To connect users and the development organization an Experience Forum, as presented in [13] can be used to facilitate feedback: An Experience Forum is integrated into the client application. At each step, the user is shown the handbook and experiences of other users relevant to the current step. Additionally, the user can submit feedback to the activity she or he is currently performing. Feedback can be of three types:

Bug Reports: Users can instantly submit software defects they find. These can be propagated into the development process. Bug Reports affect only one Use Case as they will address defects in a screen which affects the current user.

Feature Requests: Users observe features which would improve their daily tasks during their work. They can submit these feature requests via the Experience Forum as well. Those requests are fed as well into the development process.

Process Shortcomings and Improvements: Users can leave comments concerning the business process. For example, documents which are forwarded to them but never used can be criticized. Such feedbacks affect the overall process. The development organization together with the business process designers need to take those comments into account.

Process Experiences: The users are actually the ones who perform the business process. They have experiences how to interpret certain rules and how to handle unexpected situations. If they have mastered a new situation, they can enter their newly gained experience into the Experience Forum and let all other users profit from it. This kind of feedback founds a new Community of Practice among the relevant users.

The main advantage of an Experience Forum is the possibility to automatically capture the context of feedback in a SOA: The system knows which function in which process is currently performed by which user because the service composition already has this information. Therefore, the feedback can automatically be assigned to the underlying business function and retrieved whenever another user performs the same business function.

As stated above, the context contains the reference to the business process position it was made in and the submitter. Additionally, the language is stored in the context. Only feedback understandable by a certain user is retrieved and presented.

Figure 2 shows an Experience Forum integrated into a client application. On the right hand side, it is readily accessible and usable with only some mouse clicks. This low threshold is very important in order to improve user acceptance. If the Experience Forum was somewhere hidden in the menus or hard to use, it would not be put into use by the users.

5 Example Project

The example project is taken from a university project. The software and the process for managing student exams and thesis had to be newly designed. Partic-

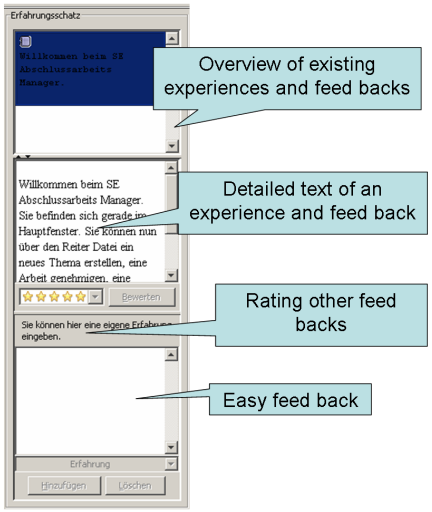

Fig. 2. Example screen shot of an Experience Forum

ipants of this process are the professors, the personnel of the Academic Examination Office, and of course the students. Because no formalized business process was in place, Use Cases have been used to elicit and document the requirements from the point of view of each actor. An extract of the Use Cases are illustrated in figure 3.

Using the described algorithm these Use Cases are transformed into an EPC model, which is illustrated in figure 4. Afterwards, the system has been developed and put into production. Example feedback given via the Experience Forum could have been e.g.:

- The secretary of the Academic Examination Office observes that a mark 2.7 of an exam is incorrectly rounded to 3.0 instead of 2.75 (bug report).
- A professor observes, that she can access the student's name in the application. However, the email address is not visible to her. This feature would be handy for contacting the student. Currently, she has to ask the students for the email addresses which she has to store separately (feature request).
- A student, who already registered for a thesis, had to choose a topic for his thesis first. However, the topic has not been finalized before due to organizational issues. It would be good, if theses could be registered without a topic, which has to be inserted later on (process improvement).
- It took another professor a long time to decide how to grade a problematic thesis. He put his criteria and his reasoning into the Experience Forum. A new professor read the comment, which helped her grading her first thesis at the new university (process experience).

Use Case	#1: Student applies for Thesis		
Primary Actor	Student		
Stakeholders	Student: wants to apply easily Secretary (Academic Examination Office): wants easy to use/read forms for further handling registration		
Minimal Guarantees	none		
Successs Guarantees	Application is submitted		
Preconditions	none		
Triggers	Student wants to write thesis		
Main Success Scenario	1	Student	fills out form with personal data
	2	Student	submits form to Academic Examination Office
Extensions	none		

Use Case	#2: Academic Examination Office approves Thesis		
Primary Actor	Secretary (Academic Examination Office)		
Stakeholders	Secretary (Academic Examination Office): wants easy to use/read forms for further handling registration Manager (Academic Examination Office): wants short handling times		
Minimal Guarantees	Student's data are handled according to regulations		
Successs Guarantees	Student may write Thesis		
Preconditions	none		
Triggers	Application is submitted		
Main Success Scenario	1	Secretary	checks if student has 80% of Credit Points
	2	Secretary	approves application
Extensions	1a If Student has less than 80% of Credit Points then Secretary denies Application		

Fig. 3. Use Cases for describing the new System

The first three feedbacks have been fed into the development cycle. How to proceed from having the feedback to actually using it, is dependent on the development methodology used. In the next section, integration into several development processes is presented.

6 Development Process Integration

While an Experience Forum is a technical mechanism for collecting and facilitating user feedback, it needs to be carefully integrated into the development process used within the project to be successful. The Experience Forum is to be used after the first release used by real users. This will normally be the first production release but can also be a preview version in use by a limited number of users.

Figure 5 shows an overview of the activities in requirements engineering. The following list explains how the Experience Forum can generally be integrated into each activity:

Requirements Analysis (or: system analysis)

Elicitation The Experience Forum supports the activity of requirements elicitation by encouraging users to write down their issues. Nevertheless a requirements engineer has to sort the issues by their different types: requirements that affect the software (real requirements like bug-reports or new requirements), requirements that affect the business model and issues that support the business process with domain knowledge.

Interpretation In this activity the requirements engineer has to refine the raw requirements to make them tangible (ideally making them testable). The benefit of using the Experience Forum is the link between the raw

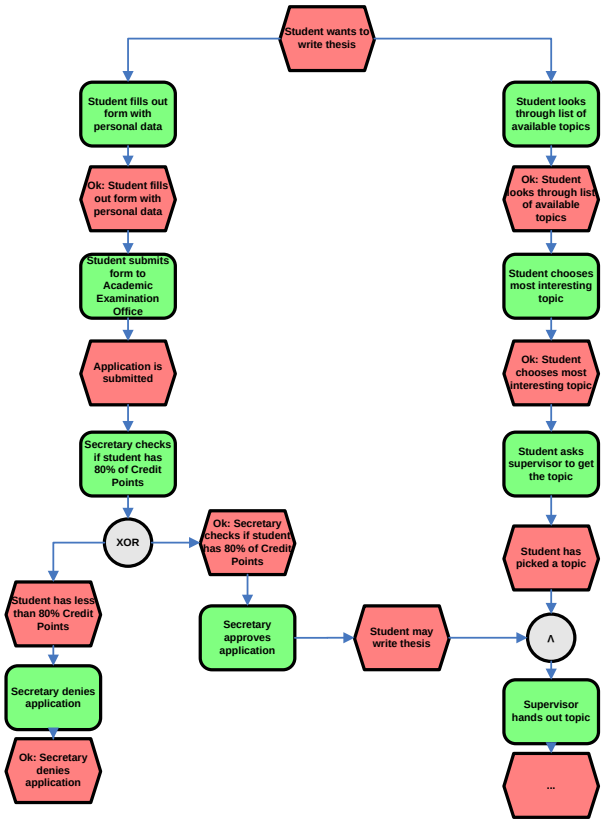

Fig. 4. Business Process extracted from Use Cases

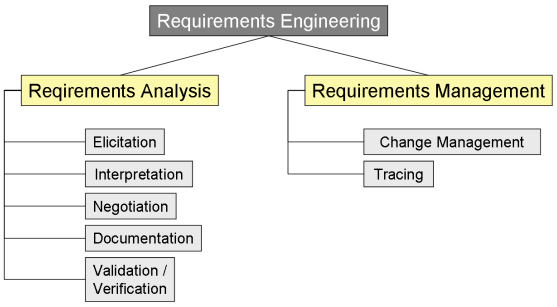

Fig. 5. Overview of requirements engineering and its activities.

requirements and the location in the business process or software application it applies to. This helps the requirements engineer to identify stakeholder. representatives. The ambiguities and inconsistencies in the comments and experiences still have to be clarified by the requirements engineer.

Negotiation The identification and resolution of inconsistencies and conflicts in the requirements as well as the prioritization of requirements define this activity. Obviously this activity demands flair, a fact that is even aggravated by the transparency the Experience Forum introduced (see section 6.3) – especially, if bugs and feature requests are assigned a low priority.

Documentation This activity transforms the raw requirements of the Experience Forum into a form best suited for the software development process.

Verification / validation The inspection of form and content is supported by the Experience Forum's context information.

Requirements Management All activities that deal with requirements management should incorporate the Experience Forum:

Change Management Solved issues have to be labeled or deleted, issues that apply to parts of the software or business process that have been changed must be revised.

Tracing The Experience Forum contains valuable information about the context of requirements. So solutions that aim at traceability should take the Experience Forum in account, too.

The development process itself is supposed to be an iterative development process. Since feedback can only be collected from a first working software version, improvements originating from the Experience Forum can consequently only be added from the second iteration or software version onwards.

6.1 A Document-Based Process

An Experience Forum can enhance the communication flow between the users, the responsible persons for requirements and the development team in document-based development processes. The new communication structure can be seen in figure 6.1.

Our example process defines the role of the *experience and requirements engineer*. This role is responsible for refining the working experiences in the forum into requirements. Another responsibility is to sort the new requirements and give them as *change request* either to the *process designers* or to the *software change control board*.

The *process designers* will adjust the business process which will normally generate additional *change requests* for the software.

The *software change control board* gives the clearance for the *change requests* to the *software developers*. Now the new release of the software is created. Before it can finally be made available for the users, the *experience and requirements*

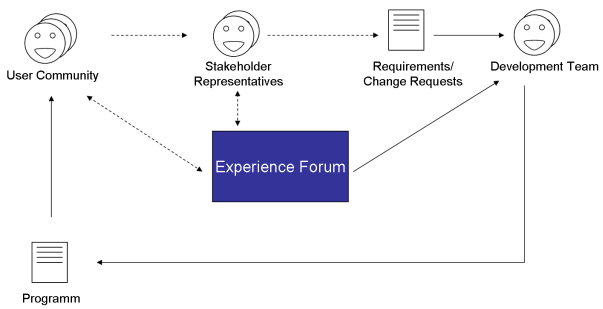

Fig. 6. Experience Base integrated into a SOA project

engineer has to adjust the experiences of the forum to the new release. At this point a refinement of the experiences in the Experience Forum of the old release into best practices (and similar activities proposed in [2]) can be made. For the requirements management process it is important to mark the experiences that lead to the new requirements as closed issues at this point.

Developers will normally have read access to the information necessary to complete their assignments and the users can access their information as well, possibly forming a Community of Practice.

With the introduction of the new software release the next iteration is started. The Experience Forum is seeded with the refined experiences from the last release and the software reflects the latest version of the business process.

6.2 Scrum

Whenever a development process is in place which is not iterative by default, the Experience Forum approach cannot be used because it requires multiple iterations to incorporate the feedback into the later software versions. When organizations have non-iterative processes in use, one way to make these development processes iterative is wrapping them in Scrum.

Scrum [3] is an agile management methodology, partitioning the whole development project into 30 day iterations called *Sprints*. The product-owner maintains a list of requirements ordered by priorities, the *product-backlog*. The development-team takes as much of the requirements from this list as they want to implement in the next 30 days and puts them into the sprint-backlog. Within a sprint the requirements in this sprint-backlog cannot be changed from outside the team. However, the product-owner may append new requirements to the product-backlog during a sprint.

Therefore, the product-owner is responsible for requirements analysis and to incorporate information gathered by the Experience Forum into the product-backlog. Due to the fixation of requirements during a Sprint, information gathered by the Experience Forum can only be implemented during the third or a following Sprint.

6.3 Advantages and Pitfalls

The concept of an Experience Forum has – independent of the development process – many advantages: Users are able to give easily and quickly feedback during their daily work. The threshold is thereby reduced improving the probability of getting feedback. While the user submits information, the application's and process contexts are automatically captured: Issues are linked to the user interface and the current process element. This is especially easy if - as in our prototype system - the user interface is generated from the process description. Therefore, the system can automatically link between user interface elements and the part of the process it supports. Requirements Engineers or similar roles are thereby supported in the requirement interpretation because they can better understand and realize the feedback without needing to recreate the whole situation which normally is not well-described. This also applies to the developers: Whenever they get a defect assigned, they can also see the context to which the defect applies to.

However, Experience Forums - if used wrongly - can have negative effects: If users are willing to give feedback to the software and processes they are using and they are affected by, they expect that things are getting better. If the feedback is not used or seems to be not used due to long response times, the motivation of the user-base may degrade and acceptance may decrease. These risks must be guarded against which in essence means the process integration must be carefully done. Especially, a user notification, what status the user's feedback has, if and when it will be considered, can mitigate that risk.

Furthermore, one should be aware of that an Experience Forum can only assist in making existing processes and software incrementally better but will not lead to radically new processes. If a company wants to completely re-engineer their processes it may not be the tool of choice. However, in this case an already used Experience Forum can be beneficial: Process Designers can use the gathered experiences and information to learn from prior weaknesses and problems.

7 Related Work

Profiting from information contained in Use Cases for other models and development phases has always been a goal in software development projects, which often deal with business processes.

Cockburn himself only mentions the possibility of applying Use Cases for deriving business processes. He offers a template in [4] but no rules or advise how to proceed from there.

The field of model-driven development has tried to combine the concept of Use Cases with its models. Instead of tabular and textual descriptions, UML sequence diagrams or similar models are used [9]. A Use Case is consequently denoted in Use Case diagrams and refined in other models thereby eliminating the textual description. This can pose a problem when communicating with non-technical users. A process for UML-based development of business processes

is given in [14]. Missing from such approaches are capabilities for expressing control flow between Use Cases and therefore the generation of business process models. The only way to achieve business process generation is to explicitly model the control flow between Use Cases in at least one additional model. With the introduction of Use Case Charts and their formalization [17], it is possible to define control flow dependencies between Use Cases and refine them in UML. From such descriptions other models can be generated, e.g. Hierarchical State Machines [18].

The Experience Forum presented in this paper has many similarities to the Experience Base in Basili's Experience Factory [2]. In fact, it offers the same functionality, e.g. it helps users finding applicable experiences by the context of their work. The Experience Base in a Learning Software Organization however supports the software development-team to optimize their development process. Note that the Experience Base is maintained by developers for developers.

In contrast, our Experience Forum serves the users of the software as an annotated online manual and supports them to optimize their daily work. As a by-product it helps with the elicitation of requirements for the next release: the parts of the software that have the most comments are good candidates for the next changes. The fact that the Experience Forum is maintained by the users, but leveraged by users and developers implies new chances and threats.

Change Management is also related to the Experience Forum. When enough comments exist in the Experience Forum, they have to be transformed into new requirements. These requirements have to be introduced into the software development process. Our approach does not strive to replace traditional change management. Instead it supports generating the new requirements that represent the changes in change management.

Closer to the intended purpose of the Experience Forum are feedback facilities nowadays commonly found in many software products (like e.g. in Microsoft Windows, KDE or Mozilla). These facilities also try to collect information about when and how potential bugs and unwanted situations (like maintenance error or malfunction) occurs. Similar to our approach they depend on the help of the users. The Experience Forum is different in that it allows the users to see reports of other users as well. Therefore, users can profit from tips and help each other to avoid certain misconducts of the information system. Of course this transparency makes bugs highly visible throughout the user community. This is certainly an unwanted effect for many software products. In section 6.3 we described how this effect can be used to positively influence the acceptance and maintenance of the software.

The idea of giving users the chance to help themselves is neither surprising (users are experts in their domain) nor new. There is a tendency to introduce this idea for example in online software documentation. In the online documentation of Postgresql or PHP the user can annotate the documentation and enrich it with examples or longer explanations. This approach has many similarities to online communities like forums, wikis or blogs. The difference is that the context of the observation is included in the editable online documentations as well as

in our Experience Forum. When an explanation in an online documentation is wrong, users will notice this fact while working in this chapter. They can quickly annotate the misleading passage. If users have to write their comments into a simple forum, they normally formulate them as general questions or solutions. The problem is that there is no link to the bad documentation that caused the problem. Therefore, the forum will evolve in parallel to the documentation which introduces new efforts to avoid inconsistencies. In contrast editable online documentation present documentation and its user annotations together, preserving the context of the comments. Our approach takes the user commented documentation one step further. Not the documentation is annotated but the application itself.

8 Conclusions and Outlook

Within this paper, a requirements round-trip from the initial requirements for SOA projects to the integration of user feedback has been presented. Since SOA tries to combine the business side and the technical side, SOA projects must integrate business process modeling and traditional requirements engineering. An approach how to derive business processes from Use Cases and vice versa has been presented in order to save effort in such environments which can be useful for more general process-driven projects as well.

However, the initial requirements will not be perfect – despite all the effort to do them "right". Fine-tuning can be done in iterative development processes, when user input is available. However, asking hundreds of users for their opinion and their problems is practically not feasible. Instead, we propose an Experience Forum, which is integrated into the client application. Users can easily enter their feedback and retrieve others' experiences. This way, not only the development team can better understand and fix problems, but also the users can share with each other their experiences related to their daily work, which is carried out using the application.

Currently, there is no other approach known to us, which tries to cover the whole requirements life-cycle for SOA projects. In our opinion, integrating initial requirements and constant feedback are the basis for sustainable successful SOA projects. The initial requirements should be formed of focused requirements in the form of Use Cases and business processes, which offer an overview about the system and are the foundations for easy implementation of service compositions.

The next step is to combine the Experience Forum with existing approaches for generating user interfaces. This would result in a platform which is very flexible and easy to change. That way, changes in business processes, possibly initiated by the Experience Forum, can easily be incorporated into a strong and enduring SOA implementation.

References

1. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana

Trickovic, and Sanjiva Weerawarana. *Business Process Execution Language for Web Services Version 1.1*, May 2003.
2. V.R. Basili, G. Caldiera, and H.D. Rombach. The Experience Factory. In John J. Marciniak, editor, *Encyclopedia of Software Engineering*, volume 1, pages 469–476. John Wiley & Sons, 1994.
3. Mike Beedle and Ken Schwaber. *Agile Software Development with Scrum*. Prentice Hall, 2001.
4. Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 14th edition, August 2005.
5. Christian Crisp. Konzept und Werkzeug zur erfahrungsbasierten Erstellung von Use Cases. Master's thesis, Leibniz Universität Hannover, October 2006.
6. Dimitri Diegel. Konzept zur Verknüpfung von Use Cases mit ereignisgesteuerten Prozessketten. Master's thesis, Gottfried Wilhelm Leibniz Unversität Hannover, September 2006.
7. Gerhard Fischer. Symmetry of ignorance, social creativity, and meta-design. *KBS Special Issues C&C99*, 13(7–8):527–537, 2000.
8. Florian Gottschalk, Michael Rosemann, and Wil M.P. van der Aalst. My own process: Providing dedicated views on EPCs. In Markus Nüttgens and Frank J. Rump, editors, *EPK 2005 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, pages 156–175, 2005.
9. Object Management Group. Unified Modeling Language: Superstructure. WWW: http://www.omg.org/cgi-bin/doc?formal/05-07-04, 2004.
10. Matthias Kloppmann, Dieter Koenig, Frank Leymann, Gerhard Pfau, Alan Rickayzen, Claus von Riegen, Patrick Schmidt, and Ivana Trickovic. *WS-BPEL Extension for People BPEL4People A Joint White Paper by IBM and SAP*. IBM/SAP, August 2005.
11. Daniel Lübke. Transformation of Use Cases to EPC Models. In *Proceedings of the EPK 2006 Workshop, Vienna, Austria*, 2006.
12. Daniel Lübke, Tim Lüecke, Kurt Schneider, and Jorge Marx Gómez. Using EPCs for Model-Driven Development of Business Applications. In Franz Lehner, Holger Nösekabel, and Peter Kleinschmidt, editors, *Multikonferenz Wirtschaftsinformatik 2006*, volume 2, pages 265–280. GITO Verlag, 2006.
13. Daniel Lübke and Kurt Schneider. Leveraging Feedback on Processes in SOA Projects. In *Proceedings of the EuroSPI 2006*, pages 195–206, 2006.
14. Bernd Oestereich, Christian Weiss, Claudia Schröder, Tim Weilkiens, and Alexander Lenhard. *Objektorientierte Geschftsprozessmodellierung mit der UML*. d.punkt Verlag, 2003.
15. W.J. Rittel and M.M. Webber. *Planning Problems are Wicked Problems*, pages 135–144. John Wiley & Sons, 135-144, New York, 1984.
16. Kurt Schneider and Daniel Lübke. Systematic Tailoring of Quality Techniques. In *Proceedings of the World Congress of Software Quality 2005*, 2005.
17. Jon Whittle. A Formal Semantics of Use Case Charts. Technical Report ISE-TR-06-02, George Mason University, http://www.ise.gmu.edu/techrep, 2006.
18. Jon Whittle and Praveen K. Jayaraman. Generating Hierarchical State Machines from Use Cases. In Martin Glinz and Robyn Lutz, editors, *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 19–28. IEEE Computer Society, 2006.