

Generating Fast Feedback in Requirements Elicitation

Kurt Schneider

FG Software Engineering, Leibniz Universität Hannover
Welfengarten 1, 30167 Hannover, Germany
Kurt.Schneider@Inf.Uni-Hannover.de

Abstract. Getting feedback fast is essential during early requirements activities. Requirements analysts need to capture interpret and validate raw requirements and information. In larger projects, a series of interviews and workshops is conducted. Stakeholder feedback for validation purposes is often collected in a second series of interviews, which may take weeks to complete. However, this may (1) delay the entire project, (2) cause stakeholders to lose interest and commitment, and (3) result in outdated, invalid requirements. Based on our “By Product-Approach”, we developed the “Fast Feedback” technique to collect additional information during initial interviews. User interface mock-ups are sketched and animated during the first interview and animated using the use case steps as guidance. This shortcut saves one or two interview cycles. A large administrative software project was the trigger for this work.

Keywords: Requirements elicitation, requirements validation, feedback, interview technique, by-product approach, support tool.

1 Introduction: Slow Feedback in Requirements Elicitation

Stakeholder involvement is crucial during requirements elicitation [1, 2]. In software projects that affect numerous individuals and groups of stakeholders, conducting a satisfactory number of interviews for elicitation and validation may take very long. The software engineering group at the Leibniz Universität Hannover was involved in the analysis phase of a large software project for our university’s internal processes. Since those processes affect students, administrators, and faculty of all university departments, there are thousands of affected stakeholders. Different department traditions result in many roles and interest groups – from Computer Science students to Biology professors or the university Chief Information Officer (CIO). The CIO asked our group to analyze the current situation of a number of key processes, and also to collect requirements for a future improved version of a support system. I will call the project *uniPro* in the context of this paper.

During the five months of that phase, different activities were carried out; the requirement analysis led to interviews and meetings. There were long periods during which requirements elicitation made no progress. Analysts could not get appointments with many of the busy stakeholders we needed for elicitation and validation.

This situation is far from unique. In many software projects a large number of busy stakeholders cannot be reached on short notice. This situation occurs in industry, banks, and in the public sector. Usually, it leads to significant project delays. Project

leaders tend to get impatient and declare requirements analysis finished in a premature state – simply because it takes so long. We consider this phenomenon a serious and recurring pattern that deserves research attention.

We wanted to find a way to speed up the elicitation and validation phase – including the idle times between appointments with busy stakeholders. For that purpose, we built an information flow model using our FLOW modelling approach [3, 4, 5]. Based on earlier work, the “By-Product Approach” [6] was proposed to assist in a similar situation (soliciting information from prototype developers [6, 7]). Since we wanted to affect information flows in a similar way, we applied the “By-Product Approach” again: We developed an elicitation technique that allows instant validation of certain elicited aspects. Fig. 1 shows how we identified the problem using information flow analysis in *uniPro*. A desired future situation was also modelled using information flows. We decided to develop a technique that instantiates the “By-Product Approach” in order to reach that goal. We call it “Fast Feedback in RE”.

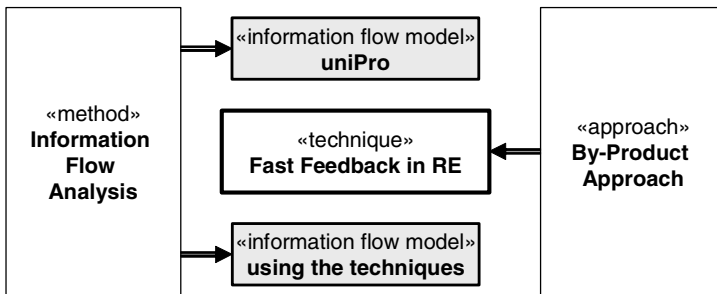


Fig. 1. Instantiating the Fast Feedback technique to improve information flows (this diagram contains type information but is not a UML model)

In this paper, we report on the Fast Feedback technique. At the same time, we describe the process of designing that technique, as we want to encourage others to invent and support their own, tailor-made requirements support techniques. Using information flow analysis and the By-Product Approach can help.

In section 2, we briefly introduce information flow analysis and show how it was applied to our problem. Section 3 presents the By-Product approach which we adopted and tailored to solving the problem. The resulting technique is sketched in section 4. In section 5, we discuss how it affects projects like ours and discuss implications.

2 Analyzing Information Flow

We use information flow analysis for a number of purposes, from tailoring reviews to individual projects [8] to organizational development [9]. I explain the basics of what we mean by “information flow analysis” and why we applied it to *uniPro*.

2.1 The Role of Information Flow in Software Projects

Software development has traditionally been described in terms of process models [10]. Requirements engineering has also been modelled as process [11]. However, in

order to understand the *uniPro* problem better, typical process models are too coarse. They tend to emphasize activities and documents, while roles and oral communication are neglected (like in the V-model www.v-model.iabg.de). However, requirements engineering is a part of a software project. A huge amount of information is generated and transferred through oral communication as well as through written documentation. Interviews and workshops, informal emails and personal notes during a meeting may not appear in the process models – but they shape requirements analysis in the real world. Effective stakeholder involvement is a key to project success [1, 2, 12].

Software projects call for written specifications. That is reasonable, and it would be a bad idea to rely on informal or oral information flows alone. However, the advent of agile approaches [13, 14] has pointed to the problem of over-specification, with useless documents of several thousand pages. They delay information flows in projects and endanger project success [15]. Light-weight practices have increased the awareness for the agile option, even in conventional project environments: It is sometimes advantageous to accept oral communication as an equal carrier of requirement information flow for a specific purpose, e.g. from on-site customer to developers, or during pair programming – or between high-level management and project leaders in a conventional project meeting.

In our FLOW research project, we consider both communication and documentation essential ingredients. We want to optimize the necessary information flows in software projects. Both documentation and communication have strengths and weaknesses, none should be dogmatically ignored. It is in a project's best interest to use the best of both worlds [15]. Most software projects realize they need both: reliable documentation for reference and long-term use; and fast and flexible information flow through well-organized communication. However, it is essential to *coordinate* both aspects, and to *facilitate* the transformation from one to the other. Many consider oral communication “soft”, unreliable, and even sometimes “unscientific”. We do not. We try to support “soft” situations with very concrete techniques and tools.

2.2 Basic Concepts of the FLOW Modelling Technique

A modelling technique was developed in our FLOW research project at Leibniz Universität Hannover. Since FLOW is not the main focus of this contribution, only its core aspects will be briefly mentioned. Observations in industrial projects (like [16, 17, 18]) shaped our view of information flows. We derived a number of resulting convictions and concepts. They are the basis of our information flow analysis. Information flow analysis is a research topic in flux [8, 19].

Assumptions and convictions

- We are convinced of the value of combining communication and documentation.
- Oral communication and short-term storage of information in people (brains) must be taken more seriously. It occurs in all projects, and for some purposes it works. Writing and reading documents cannot fully replace communication.
- We introduced the notions of “solid” and “fluid information” to allude to differences in a metaphorical way. Aggregate states of information share similarities with *aggregate states of matter*.

The metaphor of Aggregate States of Information



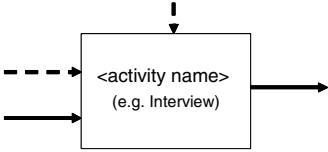



- *Solid information* refers to written or taped or other forms of readily reproducible information. It can be copied and distributed independently of individuals.
- *Fluid information* is stored in the brains of people, on handwritten sketchpads or in personal email. Usually, it comes in smaller units and changes its shape all the time. Only the owner can access and interpret it effectively.
- *Fluid* information flows faster and more painless than solid information. However, there is a limited capacity for fluid information in a brain; it may be spilled or overflow. Information leaks correspond to forgetting pieces of information.
- *Solid* information is less flexible and harder to carry. It takes more effort to bring it into a desired shape. However, it is easier to store over extended periods of time. When someone wants to absorb solid information it needs to be “melted” first (to become fluid).

The aggregate state metaphor of information conveys the idea. It should not be overstretched.

- Experience is a special kind of information flowing in a software organization. It often acts as catalyst: It enables a more efficient and more effective use of requirements or other information [3]. There is a whole body of literature on the role of experience in software projects [20, 21, 22]. However, this aspect is beyond the scope of this paper.
- Tools to feed back experiences to a task at hand were conceptualized by Fischer in his Domain-Oriented Design Environments [23].
- A simple notation for information flows is a core prerequisite for reasoning about information flows. A graphical notation is useful for discussing information flows.

We came up with a somewhat clumsy graphical notation first [24]. We boiled it down to a core of very easy elements [8]. When we use them in companies, many people are not even aware they “use a notation” at all. This contributes to the purpose of developing a common understanding on their information flows. Table 1 shows the basic symbols.

Table 1. Core elements of information flow models. Often used to extend process models.

Aggregate state	Storage	Information flow	Activity/abstraction
Solid	 <Name >	 <kind of information> (optional)	
Fluid	  <Name>	 <kind of information> (optional)	Activity with incoming and outgoing flows (solid and fluid)

For the purpose of this paper, the “activity” or “abstraction” symbol is rather important. It serves three purposes:

- (1) Information flows often follow processes - at least for a while. Therefore, we often attach information flow models to portions of existing process models. Documents and activities are common elements and synchronize both models.
- (2) At the same time, activities are treated as black boxes with an “interface” of incoming and outgoing flows. The box can be refined to show more detailed flows. This mechanism allows us to structure information flows hierarchically, which is important for scaling larger models.
- (3) When an activity box is introduced for a technique or activity that does not yet exist, its *flow interface* specifies the activity. Techniques can be developed to match that specification and interface, as shown in section 4.

2.3 Typical Information Flows During Requirement Analysis

Fig. 2 is an authentic initial sketch of the information flow causing delays in the *uniPro* project. It is presented as a less-than-perfect sketch. It illustrates how flow models are supposed to be used in practice: drawn by hand, not precisely following the notation. This is an appropriate style of information flow modelling, since it serves human discussions and understanding. The model in Fig. 2 was used to discuss what happened in *uniPro* and what we considered the problem. The timeline on the bottom was added a little later when the problem was understood better.

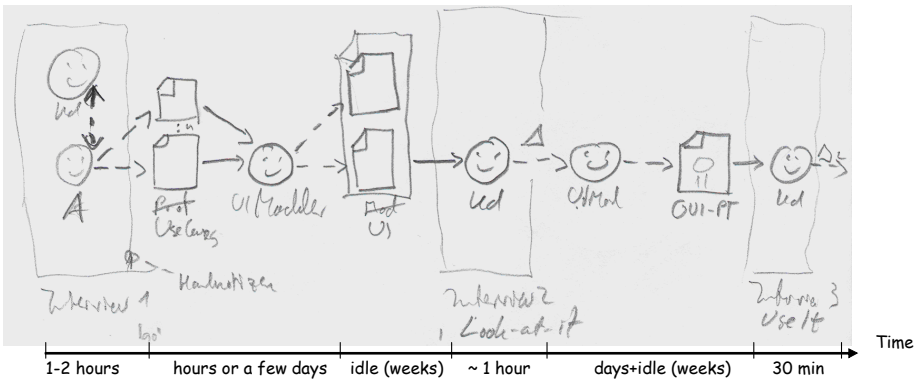


Fig. 2. Sketch of the initial *uniPro* situation during requirements analysis

Fig. 2 starts at the left showing an interview situation between a customer (“Kd” for “Kunde” in German) and an analyst (A). They mainly talk to each other, while A takes some handwritten notes (fluid). The dashed double arrow indicates fluid exchange of information. The box around this interview mixes two abstraction levels on the same diagram, (1) the interview activity box and (2) its details. They conform to the same (outgoing) flows. In the modeled situation, use cases are written as a solid

piece of information (documents). Person A may have sketched use cases during the interview, but rearranging steps and extensions will often leave notes unreadable for others. They need to be cleaned up before they constitute “solid documents” according to the above-mentioned definition of aggregate states.

In the next step, a separate role merges the use cases that refer to different tasks of the same stakeholder and suggest a first draft of the user interface (UI). Each stakeholder might see a different interface, and each interface will usually consist of a number of screens. At this point in time, pencil-and-paper mock-ups are used. According to usability engineering practices [25] the sketchy look of pencil-and-paper prototypes is important. It reminds stakeholders to draw their attention to the pure presence and rough position of interface elements – rather than their colors and sizes and button shapes. Those details are not relevant yet.

In a second series of interviews, customers (Kd) are confronted with the user interface developer, who receives *fluid feedback* on both the user interface and the use cases corresponding to them. The model gets really sketchy and short at this point, but it portrays reality: the UI modeler does not care to update any use case documents, but rather starts to build a first electronic “demonstration” prototype (GUI-PT) based on the feedback of all stakeholders. They can try it during a third interview, and so on.

The added timelines shows: Interviews took only one or two hours each; preparation and analysis of interviews, as well as drawing prototypes took from some hours to a few days. But a follow-up interview could not be scheduled within reasonable time; in many cases, it never took place – with obvious detrimental consequences for requirements validation.

Please note that diagram Fig. 2 does *not* describe a plan or an ideal process or flow: it rather shows the *actual flows* that we reconstructed after we got stuck.

3 Applying the “By-product Approach”

In earlier work, we had captured design rationale [7]. Much like in the interview situations above, there were only a few available time slots to extract knowledge and experience from the experts. The *By-Product Approach* emerged from the desire to use those time slots more effectively. The approach is motivated and described in detail in [6].

3.1 The By-product Approach

The approach can be directly applied to the requirements analysis situation. It emphasizes a clear commitment to shifting effort away from the bearer of information (rationale or requirements). This is essential to making elicitation work [26]. The name “By-Product Approach” comes from the attitude of adding extra value as a by-product of doing something that needs to be done anyway. However, there is no magic: One can add extra value only due to computer support built before. Developing that program ahead of time is the investment that pays back during the interviews.

Definition of the By-product Approach. The following definition was given in [27]. Only a few adaptations needed to be made to apply it to Fast Feedback: Underlined terms and [remarks in brackets] are specific instantiations of more generic terms used in [27]:

“The term **approach** refers to a set of guiding principles for someone to follow in order to achieve a certain goal. The style of describing an ‘approach’ by a list of interconnected principles was successfully used by Beck in his widely-known description of eXtreme Programming [13].

*The **By-Product Approach** is defined by two **goals** and **seven principles**:*

Goals

- *Capture requirements in analysis interviews within software projects.*
- *Be as little intrusive as possible to the person interviewed.*

Principles

1. *Focus on a project task in which requirements are surfacing (interviews)*
2. *Capture additional information during that task (not as a separate activity)*
3. *Put as little extra burden as possible on the person interviewed (but maybe on other people like the analysts)*
4. *Focus on recording during the interviews, defer indexing, structuring etc. to a follow-up activity carried out by others*
5. *Use a computer for recording and for capturing additional task-specific information for structuring purposes*
6. *Analyze recordings, search for patterns and add value. Let the program support you.*
7. *[omitted, not applicable]”*

The principles call for a computer program to record extra information.

3.2 Application to Fast Feedback Interviews: The Vision

In the *Fast Feedback* technique, we use the By-Product principles in order to cover both Use Cases and User Interface issues in one single interview. While only use case information was collected in the initial scenario, we ask for user interface information, too (“additional information”, principle 2). Here is the vision, with a preview of the tool (Fig. 3) that we later derived from that vision:

Both kinds of information are recorded on a low-invasive computer (principles 3 and 5), an A4 tablet-PC with detached keyboard, as in Fig. 3. The screen shows a use case template (left) and a mock-up (right) that can be connected. At first glance, the flat tablet-PC behaves similar to a sheet of paper. However, it also interprets the use case steps (“structuring information”, principle 5) to generate an animation of the pencil-prototypes (exploiting structure, principle 6). Stakeholders can even “interact” with the animation and pretend to enter data.



Fig. 3. Tablet-PC showing a use case and mock-up on a split screen, with optional keyboard

Extending the agenda of an interview by user interface issues extends interview durations (1.5 to 2.5 hours). However, it covers (1) use case elicitation, (2) UI basic decisions, and (3) partial validation (“as a by-product”). Adding a few minutes to an interview is much easier than scheduling another interview. Fig. 4 shows the new situation as an information flow abstraction.

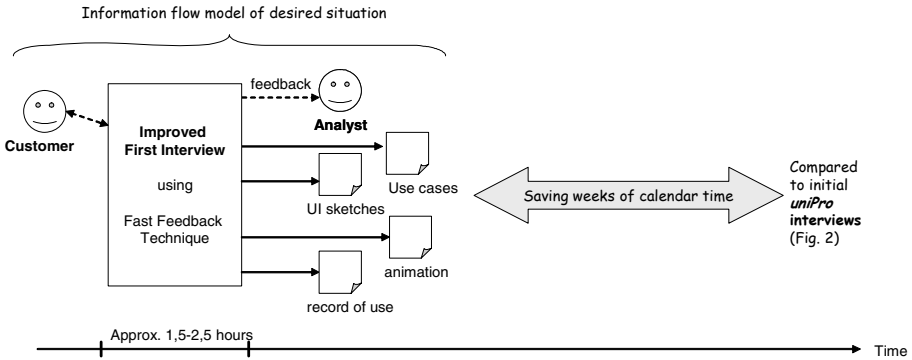


Fig. 4. Specification of the new interview technique as a flow activity

The box specifies a set of flows and pieces of information we want the technique to produce. The left part of Fig. 4 is the information flow model of the desired interview technique. Note that the customer provides input to the technique, and receives some feedback. At this level, we do not know how the technique will work in detail. Constructing a technique that provides that flow interface is the next step. We arranged flows as in Fig. 5 to match the interface.

The gray rectangle marks that parts that belong to the Fast Feedback technique. The flow interface surrounds this technique. There are many flows from and to the customer, which were summarized as a single, two-directional flow in Fig. 4. In terms of dataflow diagrams, one could call this split a “parallel decomposition” of flows [28]. Please note that we did not *exactly* match the customer flows: in Fig. 4, the customer expected to receive fluid information during the interview; in fact, all elements he or she gets in the Fast Feedback technique are solid. This deviation was considered acceptable, or even *an improvement* in interviews.

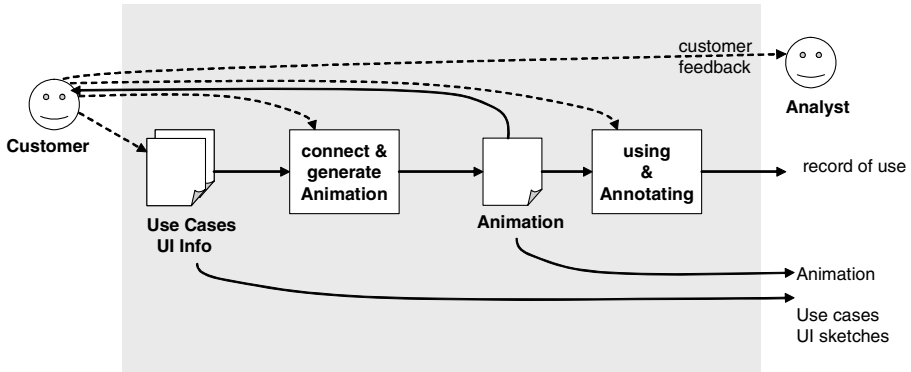


Fig. 5. Implementation of the Fast Feedback technique using a generator tool

Note that there are two “activity boxes” in Fig. 5. One denotes the fine-grained activity of compiling an animation from the use cases, UI sketches, and customer comments. The second activity lets the customer “interact” with the mock-up and records using it. The tool implemented in Carl Volhard’s Bachelor thesis [29] offers both boxes. The time devoted to an interview using this technique is up to 2.5 hours. The key improvement is in shortening the information flow from use cases to animated prototype. Instead of having two phases of manual interpretation and creation, the automated generator shortcuts this portion to a matter of seconds. It can be performed *and iterated* within the first interview. *This opportunity generates more and higher-quality feedback – fast!*

Obviously, it is not the intention of this technique to replace skilled human interface designer by a customer and an analyst scribbling on a tablet-PC. However, when a project tries to elicit requirements and basic interaction sequences from stakeholders, there is no need for professional user interface design; it is all about eliciting requirements. Usability experts receive rich material from the intense new interviews. This can empower their work, too.

4 A Technique to Generate Fast Feedback

By instantiating the By-Product Approach in order to empower interviews, the new technique will improve information flows and, thus, speed up requirements elicitation and validation.

4.1 Fast Feedback Needs Tool Support

The By-Product Approach explicitly calls for specific computer support. It is not just a matter of convenience but a part of the concept to exploit computer power. Therefore, we developed a tool to support the Fast Feedback technique [29]. In that tool, use case templates are completed on the tablet PC (Fig. 6, left). Animated mock-up prototypes are drawn and displayed in the same tool (Fig. 6, right).

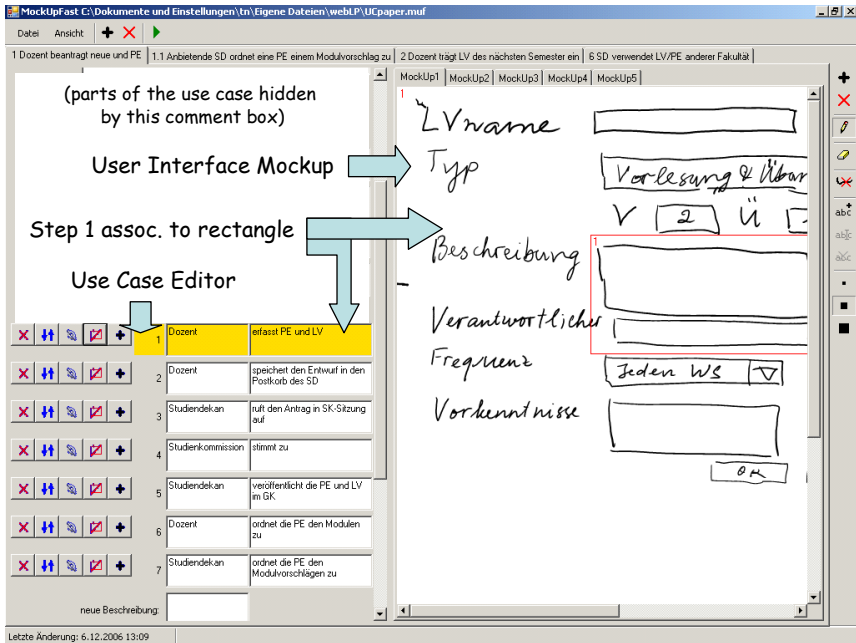


Fig. 6. User interface mock-ups can be animated when connected to use case steps

The tool generates animations from the use case steps and the user interface mock-ups (Fig. 7). Animations display the interface mock-ups in the order they are referenced in the use case. Execution can descend to lower-level used cases, extensions need user decisions. Portions relevant for interaction are highlighted by a thin-line rectangle.

Stakeholders can scribble values into the mock-up input fields during the animation; they may pretend to press buttons, make selections in lists by simply drawing on the mock-ups (in a different color). Again, the tablet-PC is used like a sheet of paper (Figs. 6, 7). User actions appear on the sketch, but the mock-up cannot react to it. It is a pure mock-up. Actions are recorded and provide valuable (solid!) information on how the stakeholders intend to use the system.

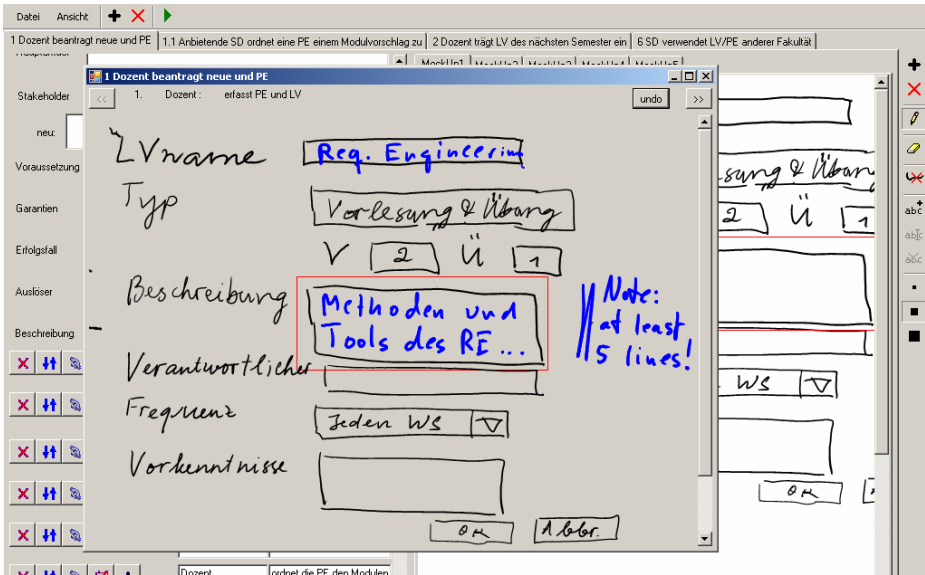


Fig. 7. Animation with “user input” in thin-line rectangle, additional user comments (margin)

4.2 Fine Tuning Is Essential for Tool and Technique

The new interview technique has been explained above. In order to make it work, several subtle adjustments had to be made. In particular, using the program must not distract either partner from the main mission of the interview: eliciting requirements. Carl Volhard, who developed the tool as part of his Bachelor thesis [29] discusses several low- to middle-level usability issues for which he considered different solutions. For example, he compares three options for the mechanism to link use cases to corresponding portions of a mock-up. The thin-line rectangle was chosen. It is often difficult to make an interaction look straight-forward.

The interview technique is supposed to be applied in the following setting:

- One or two interviewers face one (or up to three) stakeholders representing the same stakeholder group.
- They use a tablet-PC with the recording and generator program running. It offers a use case template and an “empty paper” view for drawing the mock-ups.
- If there are two interviewers, one will concentrate on asking and interacting with the stakeholder. The other interviewer will fill the template.
- However, drafting the mock-ups should be done by the main interviewer in tight interaction with the stakeholder. A stakeholder may even grab the pencil and draw a mock-up.
- Since elicitation and validation is folded into one session, both use cases and mock-ups will be revised and updated iteratively. During that part of the interview, the tool must be visible for the stakeholder. A computer projector is an option, but direct interaction with the tablet-PC is preferable.

4.3 Fast Feedback Output

The output of initial *uniPro* interviews consisted of a collection of hand-written notes on paper with several informal sketches. The output of the new brand of interviews includes:

- *A set of use cases*: Using a template on the tablet-PC, analysts may choose to scribble or type what they find out about all aspects of a prototype. In particular, there are fields for scenario and extension steps.
- *A set of user interface mock-ups*: Analysts and users can manually sketch interfaces on the tablet-PC – just like on a piece of paper.
- *The linked animation* of use cases and mock-ups.
- *A recording of the pretended use* of the mock-ups, including the values they scribbled into fields etc.
- *The feedback of users* to all above-mentioned aspects: Immediate modifications can be made to use cases and mock-ups. The animation is automatically updated. Nevertheless, we always take some paper along. When the tool is busy playing the animation, it is better to take notes on paper.

5 Related Work and Discussion

Maiden et al. [30] describe the Mobile Scenario Presenter (MSP), a PDA-based requirements discovery tool that allows users to step through scenarios and add additional requirements or information. They carried out a series of studies to explore the usability and usefulness of such a mobile device. They report users were able to identify events in the real world and relate them to their scenarios. How to deal with limited display and missing keyboard were identified as open research questions.

In the Fast Feedback technique, we avoided the problem of keyboard and small display by using an A4 size Tablet-PC with optional detached keyboard. It was not mainly selected as a mobile device, but as a “discreet” tool that starts out in the background but allows to collect and to interpret additional information (user interfaces). We entered the *uniPro* use cases and UI mock-ups to check for usability. All use cases and mock-ups could be expressed using the tool. In the initial scenario, they had been drawn offline in PowerPoint. Therefore, they looked “more final” than the sketches in our tool. This “final look” may distract customers from their premature status [25]. Usability was a high-priority quality goal and consumed a large percentage of the development effort. As a consequence, no severe usability problems were reported, and one *uniPro* analyst was so enthusiastic about the tool that she demanded to use it on her new project immediately. A range of people from analysts over researchers to school children were able to write steps of “a story” (use case), draw pictures and link them for animation. Although this experience was highly encouraging, it does not constitute a valid empirical result. We plan to conduct a controlled experiment (or rather several case studies) on some aspects of the interviews in the next semester. However, most relevant issues like *amount of information transferred* will be very difficult to trace in a controlled experiment. Case studies are often preferable for software engineering [31].

Davis et al. [32] have reviewed a number of studies on different requirements elicitation techniques. They found interviews more effective than card sorting or thinking aloud, among others. They also found no evidence for prototypes being helpful during elicitation. However, they point to the constraints of their review. Most importantly, elicitation techniques were cut into facets for the purpose of the review. According to Davis et al., the small size of their samples should also be considered.

We are convinced that a *large project with a very large number of stakeholders* like *uniPro* will benefit from intermediate representations. In particular, feedback from a slightly different viewpoint can facilitate interviews (UI mock-up instead of use case). We even saw this effect when working on paper (with PowerPoint mock-ups) before. With a tailored and optimized technique like Fast Forward, we are confident to increase the value.

The hand-writing recognition feature was not unanimously welcome. Correcting errors takes rather long. One should either let them in during the interview, or use the keyboard for completing the template (Fig. 3). However, mock-ups need to be hand-drawn [25]. Fast Feedback sessions provide validated output: use cases and mock-ups were checked for consistency and correctness by the designer and the stakeholder.

Since we were interested in the approach of supporting requirements analysis as such, de Vries [33] developed a quite similar tool. In a similar information flow situation, he applied the By-Product Approach to a slightly different subject: When people discuss their (business) activities, the technique collects add-on information on incoming and outgoing flows. Those are compiled to generate and display processes beyond single interviews: those processes illustrate what happens to information someone else provides. Again, it helps stakeholders to validate what they said before.

6 Conclusions

Requirements elicitation is difficult. Requirements elicitation in a large project with many busy stakeholders and an impatient project leader is very difficult – but it often occurs in reality. When the findings of an interview take weeks to be validated, requirements quality suffers. Due to the long delay, stakeholders cannot remember details.

We tried to improve the situation by analyzing information flow and by applying the By-Product Approach. We emphasized the problem by comparing an initial information flow model with a desired situation model.

In this paper, a technique was introduced that instantiates the By-Product Approach. It requires a tailor-made tool to record and automatically interpret use case steps. At the beginning of the interview, the tool is simply used as electronic paper with recording abilities. As the interview proceeds, additional user interface information is added to generate a different view of the information received so far. This provokes stakeholders to check and validate what was inferred from their input.

In the end, there is more information, highly connected and automatically recorded. The initial investment in tool and technique pays back in fast feedback “as a by-product”. Our tool is a feasibility prototype developed in Volhard’s Bachelor thesis [29]. It is optimized for easy use and discreet behaviour and has been applied to *uniPro* material. We consider this technique a small, but important step forward. In

addition, we want to encourage other requirements engineers to create their own individual techniques where they see bottlenecks. Using information flow modelling and the By-Product Approach puts tailor-made improvements in reach.

References

1. Rupp, C.: Requirements-Engineering und -Management. 3 edn. Hanser Fachbuchverlag (2004)
2. Alexander, I.F., Stevens, R.: Writing Better Requirements, Harlow, Pearson Education Ltd. (2002)
3. Schneider, K., Lübke, D., Flohr, T.: Softwareentwicklung zwischen Disziplin und Schnelligkeit. Tele. Kommunikation Aktuell 59(05-06), 1–21 (2005)
4. Schneider, K.: Aggregatzustände von Anforderungen erkennen und nutzen. In: GI Softwaretechnik-Trends, pp. 22–23 (2006)
5. Schneider, K.: Software Engineering nach Maß mit FLOW. In: SQMcongress 2006. Düsseldorf: SQS (2006)
6. Schneider, K.: Rationale as a By-Product. In: Dutoit, A.H.M.R., Mistrik, I., Paech, B. (eds.) Rationale Management in Software Engineering, pp. 91–109. Springer, Heidelberg (2006)
7. Schneider, K.: Prototypes as Assets, not Toys. Why and How to Extract Knowledge from Prototypes. In: 18th International Conference on Software Engineering (ICSE-18) Berlin, Germany (1996)
8. Schneider, K. and Lübke, D.: Systematic Tailoring of Quality Techniques. In: World Congress of Software Quality 2005. Munich, Germany (2005)
9. Stapel, K.: Informationsflussoptimierung eines Softwareentwicklungsprozesses in der Bankenbranche, Fachgebiet Software Engineering, Gottfried Wilhelm Leibniz Universität Hannover (2006)
10. Curtis, B., Kellner, M.I., Over, J.: Process modelling. Communications of the ACM archive 35(9, Special issue on analysis and modeling in software development), 75–90 (1992)
11. Macaulay, L.A.: Requirements Engineering. Springer, Heidelberg (1995)
12. Arias, E.G., Schneider, K., Thies, S.: A continuum approach: From language of pieces to virtual stakeholders. In: World Conference on Artificial Intelligence in Education (AI-ED 98) (1998)
13. Beck, K.: Extreme Programming Explained. Addison-Wesley, London (2000)
14. Cockburn, A.: Agile Software Development. Addison Wesley, London (2002)
15. Boehm, B., Turner, R.: Balancing Agility and Discipline - A Guide for the Perplexed. Addison-Wesely, London (2003)
16. Schneider, K.: Active Probes: Synergy in Experience-Based Process Improvement. In: Product Focused Software Process Improvement PROFES 2000, Springer, Heidelberg (2000)
17. Houdek, F., Schneider, K.: Software Experience Center. The Evolution of the Experience Factory Concept. In: International NASA-SEL Workshop (1999)
18. Manhart, P., Schneider, K.: Breaking the Ice for Agile Development of Embedded Software - an Industry Experience. In: International Conference on Software Engineering (ICSE 2004) Edinburgh, Scotland (2004)
19. Schneider, K., Stapel, K.: Informationsflussanalyse für angemessene Dokumentation und verbesserte Kommunikation, SE 2007. Hamburg (2007)

20. Basili, V., Caldiera, G.: Improve software quality by using knowledge and experience, Fall: Sloan Management Review (1995)
21. Johansson, C., Hall, P., Coquard, M.: Talk to Paula and Peter - They are Experienced. In: International Conference on Software Engineering and Knowledge Engineering (SEKE'99), Workshop on Learning Software Organizations. Kaiserslautern, Germany, Springer, Heidelberg (1999)
22. Schneider, K.: What to Expect from Software Experience Exploitation. Journal of Universal Computer Science (J.UCS) 8(6), 44–54 (2002), www.jucs.org
23. Fischer, G.: Domain-Oriented Design Environments. Automated Software Engineering 1(2), 177–203 (1994)
24. Sarkisyan, E.: Analyse und Definition von verschiedenen FLOW-Modellen, FG Software Engineering, Leibniz Universität Hannover (2006)
25. Mayhew, D.J.: The Usability Engineering Lifecycle - a practitioner's handbook for user interface design. Morgan Kaufmann Publishers, San Francisco (1999)
26. Grudin, J.: Social evaluation of the user interface: Who does the work and who gets the benefit. In: INTERACT'87. IFIP Conference on Human Computer Interaction. Stuttgart, Germany (1987)
27. Schneider, K.: Aggregatzustände von Anforderungen erkennen und nutzen. GI Softwaretechnik-Trends 26(1), 22–23 (2006)
28. DeMarco, T.: Structured Analysis and System Specification. Prentice-Hall, Englewood Cliffs (1979)
29. Volhard, C.: Unterstützung von Use Cases und Oberflächenprototypen in Interviews zur Prozessmodellierung, Fachgebiet Software Engineering, Gottfried Wilhelm Leibniz Universität Hannover (2006)
30. Maiden, N., et al.: Making Mobile Requirements Engineering Tools Usable and Useful. In: Requirements Engineering (RE 2006), IEEE Computer Society, Minneapolis, USA (2006)
31. Rombach, D., Basili, V.R., Schneider, K.: Experimental Software Engineering Issues: Assessment and Future Directions. Dagstuhl Workshop Proceedings. Springer, Heidelberg (2007)
32. Davis, A., et al.: Effectiveness of Requirements Engineering Techniques: Empirical Results Derived from a Systematic Review. In: Requirements Engineering (RE 2006), IEEE Computer Society, Minneapolis, USA (2006)
33. Vries, L.d.: Konzept und Realisierung eines Werkzeuges zur Unterstützung von Interviews in der Prozessmodellierung, Fachgebiet Software Engineering, Gottfried Wilhelm Leibniz Universität Hannover (2006)