Improving an Industrial Reference Process by Information Flow Analysis: A Case Study

Kai Stapel, Kurt Schneider, Daniel Lübke, and Thomas Flohr

Software Engineering Group, Leibniz Universität Hannover,
Welfengarten 1, 30167 Hannover, Germany
{Kai.Stapel,Kurt.Schneider,Daniel.Luebke,
Thomas.Flohr}@inf.uni-hannover.de

Abstract. Reference processes are supposed to be the basis for collaboration and mature cooperation in software development. Large business organizations need large and diverse reference processes. However, process conformance is a constant concern. There are many explanations why a project may deviate from its reference process. This is especially true in larger software companies with a lot of different projects and variants modeled in a single reference process. During an industrial cooperation we have identified a phenomenon that adds to the problem: Unclear and incorrect information flows. Process modeling notations and practices in many large organizations nurture information flow anomalies. We improved the information flows in the reference software process by means of information flow analysis and flow patterns. A comprehensible reference process with reasonable information flows is easier to understand and therefore gains acceptance in the project team.

1 Introduction

Mature software organizations define and maintain software development processes based on CMMI [1] or SPICE (ISO 15 504). Modeling a reference process for software projects is a mandatory task in maturing environments. However, reference process models generate new problems. Correctness and conformance have been concerns for many years.

In this contribution, we report on collaboration with a financial institution. We were asked to check and improve their large reference process model. During that work, we identified an interesting class of problems that we traced back to information flow anomalies. We applied information flow analysis to tackle the problems.

The implementation of a large software development reference processes is difficult in itself. Challenges include:

- **Complex, unclear, incomprehensible reference processes:** To cover every intended use many processes tend to be complex. More documents and possible branches are modeled rather than keeping it simple and understandable. Unfortunate name assignments as well as sloppy descriptions make processes and

J. Münch and P. Abrahamsson (Eds.): PROFES 2007, LNCS 4589, pp. 147–159, 2007.

activities unclear. Furthermore the information flows are obscure. Complexity and the lack of clarity as well as badly designed information flows lead to incomprehensibility of the processes.

- **Unrealistic requirements:** Many processes require a lot of documentation work. Almost every activity demands a document as a result. Most of the time this is due to the fact that most process modeling techniques lack the ability to represent the flow of information in other representation media than documents. The reference process analyzed in our case study requires each and every project to produce between 60 and 165 documents.
- **Inflexible reference processes:** Since processes are designed to fit many needs they are often neither suitable for large nor for small projects. That is the reason why tailoring is needed before each project.
- **Faulty reference processes:** Reference processes are faulty in two ways: syntactically and semantically. Syntactical mistakes are caused by not following modeling standards and guidelines, unthoughtful modeling or insufficient support of the notation by the modeling tool. Semantic mistakes like multiple preparations of identical contents or parallel alteration of the same information are caused by e. g. distributed modeling and missing interface coordination. Hence, information flows are not only obscure but also faulty.

At least some of these problems can be found in most reference software development processes. As a consequence, many reference processes are not accepted by project teams. We analyzed such a software development process in a case study at an information technology service provider in the financial sector. We tried to narrow the gap between reference and actual processes by means of information flow analysis. Optimizing the information flows of a process leads to a more comprehensible and sound process which again affects its acceptance positively. The basic concepts of information flow analysis will be described in the following section.

2 Information Flow Analysis Concepts (FLOW Project)

In this section the basic concepts of information flow analysis are presented, as far as they are needed to understand the case study described in section 3. A detailed introduction can be found in [2, 3]. Our FLOW project was initiated at the Leibniz Universität Hannover in 2004. Besides the information flow analysis, FLOW is concerned with active strengthening and coordination of flows. We develop specific techniques and tools that improve information flows [3, 4]. However, these aspects are beyond this paper.

2.1 Goals of Information Flow Modeling

Information that flows in a process or project is modeled to accomplish several goals:

- Reflection and manual analysis by experts helps to remove flow anomalies and to shape information flows more adequate.
- Some anomalies can be described as information flow patterns. Pattern search can then be semi-automated.

- Techniques and tools can be developed that specifically affect information flows.
- Known existing techniques can be reframed and used for information flow improvement by specifying their information flow behavior.

2.2 Postulates of Information Flow Analysis

The approach of information flow analysis is based on some fundamental beliefs and assumptions. A detailed description can be found in [5] and [6]:

- *Information flows link processes and direct communication* and, therefore, also connect conventional and agile approaches.
- *Experience is a special kind of information* which is being modeled explicitly. It often influences activities and acts as a catalyst.
- We introduce the *state of information*: "Fluid" information is verbal or non-objectively reproducible information including e-mails and personal notes third parties cannot access or reproduce. "Solid" information refers to written or recorded information (like videos) which is long-term accessible to third parties.
- *Coarse modeling of information content*. Usually just requirements are modeled on their way through the project. In special cases (a few) different types of information can be used.

The last two points require explanation. Referring to information metaphorically as either fluid or solid states of information is typical for FLOW. It points out that the same information can appear in different shapes. Different states implicate different characteristics similar to fluid and solid matter.

- Solid information can be recalled at any time. It is stored in documents or recordings (video or audio). Access as well as storage cost time and effort, but they are repeatable. Solid experience is a special kind of solid information. It is available through checklists, best practices and (design) patterns.
- Fluid information, on the other side, is bound to people or other volatile media. We call "fluid" whatever someone has in mind and which cannot be obtained or reproduced by third parties. Fluid information can easily be transmitted by conversations and supported by some hand writings. But fluid information can easily be lost or forgotten. Fluid experience slowly grows in a person's mind while the person is doing or observing something.

Self-restriction at modeling is also important: We explicitly do not aim for "as precise as possible" definitions and detailed distinctions between information flows. We rather prefer modeling as coarse as possible. Detailed distinctions in meaning or different forms of representation of information (as tables, texts, pictures) will not be differentiated in our FLOW models. Modeling information flows aims for qualitative optimization and not for exact mapping.

2.3 FLOW Notation

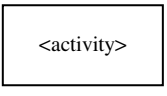
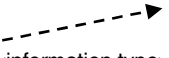
Information flows could be represented as data flow diagrams like in the 1970's [7]. The basic idea was – and still is – *not* to follow the control flow. It is rather important where information flows at all. A visual notation helps to clarify certain situations and

patterns. It ought to transport the basic concepts and still may be open for interpretations at some points. After all, it is a medium of communication for people and not a programming language. It is supposed to comply with the following requirements:

- Very easy understandable even for non computer scientists. Few, easy explainable symbols. Ability to extend existing process notations.
- Means of expression for fluid and solid, respectively for information storages as well as for information flows.
- Means of expression for experience and "other" information.
- The ability to establish relations to existing process notations.

Very, very simple but flexible notations were most appropriate wherever we used information flow analysis so far [3, 8-10]. That was also the case in the following case study. Table 1 depicts the symbols that satisfy the mentioned requirements.

Table 1. Symbols of information flow models; they can be used to extend other process notations

information state	store	information flow	experience flow	activity
solid	 <identifier>	 <information type> (optional)	 <experience> (optional)	 <activity>
fluid	 <identifier>	 <information type> (optional)	 <experience> (optional)	

The symbol for a fluid information store (smiley) is supposed to bring to mind that someone has the information in his or her head. Experience is distinguished from other information through a different color (in this case gray). The distinction between experience and other information is mentioned here for the sake of completeness. It is not needed in the following case study.

Additionally there is a link between FLOW and the used process notation. It is defined via the activity symbol (usually a rectangle) which is then available in both notations (FLOW and process). An activity incorporates incoming information and reissues it as an outgoing flow. The activity symbol adopts an extended meaning. This mechanism can be used to refine and structure combined FLOW-process models into hierarchies. Formally speaking, the respective activity symbol belongs to both notations.

Fig. 1 pictures a combined FLOW-process model. The process part with activities and documents is shaded gray. Below the process, FLOW symbols show requirements and design activities in a certain situation. Both solid (documents) and fluid (people) information flows appear. This composition needs to be optimized considering the characteristics of solid and fluid information. This is the goal of a FLOW software process improvement activity.

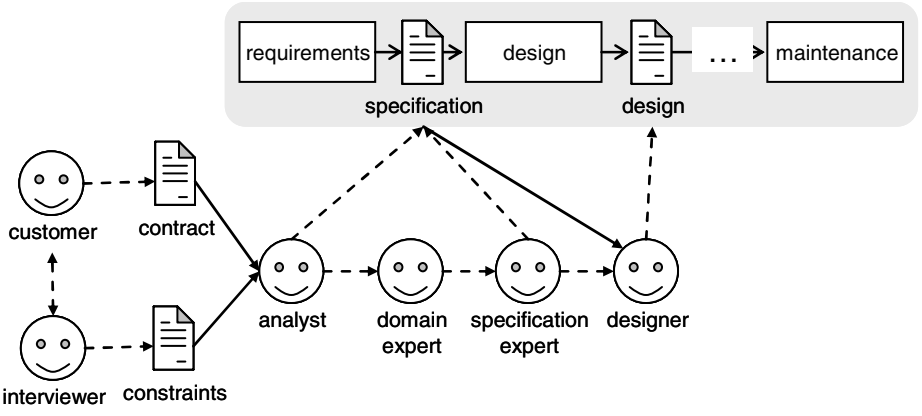


Fig. 1. Part of a combined FLOW-process model

A fairly large financial service provider asked us to help in optimizing their reference software development process. We used this opportunity to test the methods of FLOW in a case study.

3 Case Study

In a master thesis [11] we analyzed a very large existing reference software development process of an information technology service provider in the financial sector. We applied information flow optimization as described above. The respective company has more than 2500 employees. It has a large process model with 5 so called process groups, one of which is the analyzed software development process. It is an iterative incremental process aligned to RUP [12] that has grown historically. The software development process is structured in 6 layers to be clear and concise. All processes are maintained by a dedicated process department. It coordinates several process modelers from different specialist departments, the so called process experts. The process experts perform the actual process modeling. Many process experts and process managers have contributed to the reference process as it stands today.

The analyzed software development process contains most of the problems raised in the introduction, because of its size, its historical development and the many different modelers. It is complex, inflexible, unrealistically demanding and syntactically faulty. However, it is not just a messy and useless process model that the company should or could get rid of – instead, it is a typical representative of reference models in industry. Our goal was to identify causes and propose solutions for some problems that we would find using FLOW optimization. Since the only modeled information carriers in the process are documents and the document flow extraction is fairly easy we started with a document flow analysis.

3.1 Document Flow

Document flows are a special kind of information flows where the information is passed on exclusively by documents. For a detailed differentiation between document and information flows see [11].

In order to analyze document flows they first need to be extracted from the process model. Documents are modeled as prerequisites (inputs) and results (outputs) of activities. Activities are connected through a relationship of dependency or sequence. We say a "document flows" between activities A and B if the document is output of activity A and input to activity B, and if the two activities are connected via the directed sequence relationship. In the notation used, this relationship is denoted by an arrow. Document flow extraction is not as straight-forward as it might appear. In principle, one flow gets created for each document in the model.

The company uses a process modeling tool which is able to export the process in a proprietary XML graph format. We developed a tool that extracts the document flows from the XML representation and stores them in an open source graph file format, namely the Graph eXchange Language [13]. The advantage of using an open format is that any tool capable of reading and illustrating this format can be used to visualize the extracted document flows. Such visualization helps to understand a flow and makes it easier to find faulty flows.

We searched the process for anomalies aided by the extracted and visualized document flows. Identified problems fall into two groups: either syntactic or semantic document flow anomalies. Semantic anomalies are more severe and require manual analysis steps. Syntactic anomalies, however, can be detected automatically during the extraction process. The automatically found problems can be marked in the visualization to guide their tracking. For example, we found 33 documents that are created but never used, 70 documents that are needed in several activities but are never created and even 113 documents that are contained in the model but are never referenced by any activity, neither as input nor output. Altogether, 62% of all documents in the process are affected by one of these anomalies. It is no wonder that many project teams find the process confusing. Fortunately, purely syntactic anomalies are fairly easy to fix. Sometimes the process modelers just forget to connect a document to the according activity or two different names are used for the same document. Both can be fixed during revision. Many not referenced documents can be deleted from the process model, because they are not needed at all.

Semantic anomalies are more demanding, but also more rewarding to resolve. For instance, we found several cases in which the branching after activities that affect the further document flow (conditional activities) was missing. Fig. 2 depicts such a situation in a proprietary process notation used by the analyzed company.

The left side of Fig. 2 shows the quality assurance process as is. A design document is to be improved. A review step is supposed to make sure that the quality of the improved design is okay. A review just "looks" at a document but does not alter it. No changes are made during the review process. The actual problem with the left side process is that no matter what outcome of the review, the improvement process is finished according to the modeled process. Since a review is a conditional activity the process should look like the one on the right side including the condition and the branching. If the quality of the improved design is assured the process can stop and

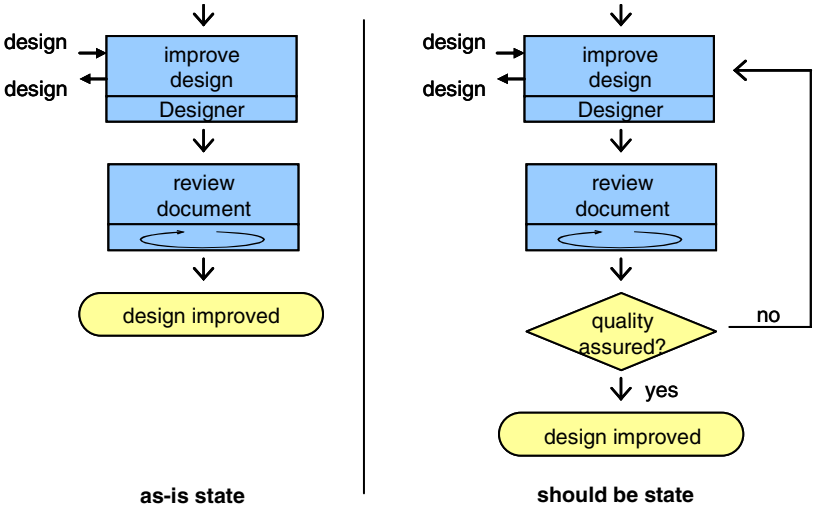


Fig. 2. Left side: as-is problem: no branching after conditional activities. Right side: Fixed process with condition and branching.

the document can be passed on ("yes" branch). But if the design is not good enough yet it has to be improved again ("no" branch).

Multiple occurrence of this problem leads to more confusion with the people who are supposed to work according to the process (the project teams). And finally confusion leads to less acceptance of the process. Teams are asking: "The process does not make sense to us, so why should we follow it"?

Many of the above-mentioned problems may sound ridiculous or avoidable. Nevertheless, we had seen many of them in different settings at different companies before. We decided it was more use to improve those real processes rather than to deny their existence.

What can be done to avoid these problems? It looks like the process modeling did not proceed accurately enough. The company should revise its process model and for instance add branches after each decision where they are not present already. We also proposed some extensions to the modeling guideline, so future revisions will be able to avoid the detected anomalies. The former modeling guideline did not include hints from the document flow point of view. That might be another reason for the many syntactic document flow anomalies.

Document flow analysis identified a lot of anomalies already. Most of them were syntactic problems with minor consequences caused by neglectful design. But there still are problems that cannot be identified by means of pure document flow analysis. What if a document is used to model direct communication, because there is no other way to do this with a given process notation? What if direct communication is not modeled at all?

3.2 Information Flow

Information flows are not limited to documented information being passed on. More effective ways of communication will also be considered, like conversations, phone calls, chats or e-mails (fluid information).

There are several ways to extract information flows from a process. The automatic extraction from the reference process model can be tried. A so-called document dependency graph can be used as a starting point. Such a graph is built by adding an edge from document A to B whenever A is output of an activity of which B is input. Assuming that the information in A depends on B an information flow from B to A can be inferred. With this assumption and the document dependency graph, information flows can automatically be extracted from the reference process. Documents as containers channel information through the project. The flows obtained in that way have two major disadvantages. First, the assumption does not always apply. Second, flows occur only between documents. People and direct communication are left out again.

This is due to a common weakness in most process notations: there is no way to refer to direct communication. To gain information flows including direct communication a process needs to be analyzed in a real project situation. An information flow expert could accompany a project and thereby note all flows. This is often not possible because of secrecy issues. Interviewing project participants is a less demanding approach.

In this case study we observed two extraction methods: The extraction from the reference process model and the extraction via interviews. Although these methods are not as effective and as accurate as e. g. the project monitoring they still produce useful information flows.

Reference Process Extraction

Our first approach of analyzing the information flows was the automatic extraction from the reference process model. To accomplish this we used the method described above. First, we created the document dependency graph from the exported process model. After that, presuming that document dependencies infer information flows, we extracted the flows and even found some anomalies.

One problem we found was a not fully connected document dependency graph. Most likely it was not fully connected because direct communication paths occurring in actual processes could not be modeled in the reference process. Several disjoint document clusters in the document dependency graph raise the question why there should be totally independent information flows in a single development process. Usually a project pursues only *one* goal. A process model should represent that by producing a fully connected document dependency graph. The single goal of a software development process is the creation of software. The main cluster of the document dependency graph correctly contains the corresponding documents. The other smaller clusters stem from some sub processes and should be connected to the main cluster, since the sub processes do not create information independently from the other development activities. However, they are not connected according to the reference process.

Two causes may be responsible for that anomaly: Modeling without due care and attention and shortcomings in the process notation. The first situation usually occurs because the modelers don't pay attention to the information flow perspective or they may not know about the right information flow analysis methods and tools. Even in case they are aware of information flows and breakdowns, they do not have or find a tool to deal with them.

The second situation is the lack of ability to address non documented information flows in most process notations. In *actual* software development processes there *is* a connection between the document clusters. The documents are not connected directly but via non documented information flows (fluid) like e-mails or verbal speech and notes like e. g. in meetings. So the anomaly is caused because fluid flows cannot be depicted in the model. Usually this kind of a problem gives the project teams a bad feeling. They know that something is wrong. But they do not exactly know what and where it is in the process when using the process view alone. Using the information flow perspective clarifies things because it helps to identify the problem and where exactly it occurs.

Other anomalies we identified were cyclic flows. They occur when two or more documents depend on each other in a cyclic way. In the majority of cases this problem can be solved by looking at the documents at the level of paragraphs. Usually paragraphs do not depend on each other anymore. Modeling granularity, thus, is a key concept to effective information flow analysis.

The anomalies derived by means of reference process extraction are a good starting point for more specific information flow analyses, like interviews of the people who perform the affected process. The information flow perspective also helps to explain suspicious parts of the process that were not clear from the process view alone.

Extraction by Interviews

The second information flow extraction method conducted was the extraction through interviews. Two project leaders of two different projects were interviewed. Both were supposed to use the same reference process model. But both described actual processes were much simpler and easier to understand. Hence, the information flows were not as confusing as the ones obtained from the reference process. We even found some information flow patterns. The *document creation process* looked similar in both actual processes and for each document to be created in them. First, the document is created on the basis of a template. Then the document iteratively gets extended or changed accompanied by a quality assurance step. This is the case for all types of documents like specifications, product manuals and even the source code. These patterns seem to be due to the compliance of the projects with the reference process. So there already is a certain level of compliance. The many anomalies usually hide that.

Beside the regular patterns we also found some problematic ones. One of which is the problem of creating a document afterwards. An example of this problem is depicted in Fig. 3.

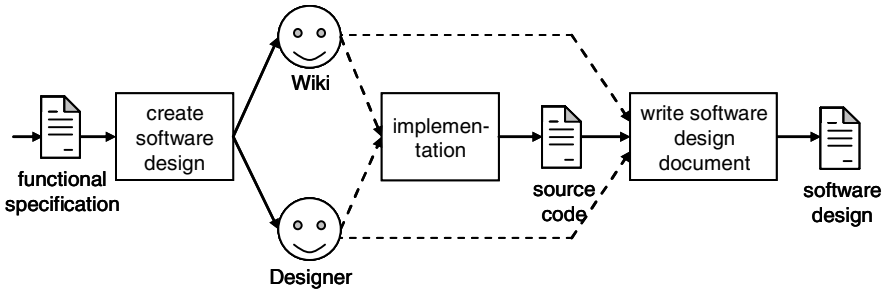


Fig. 3. Problem: Creation of design document afterwards

This FLOW shows a part of the actual software development process as described by one of the interviewed project leaders. Starting from the functional specification a software design is created. The devised design is kept in the minds of the designers, handwritten notes and a system similar to a wiki. This information is then used to implement the software. After that the design gets documented in the software design document although the implementation is already finished. But why did the document get created afterwards? That is because the reference process demands it. The process department supervises the reference process compliance of each project and penalizes non-compliances. To avoid the penalties the project created the document afterwards.

What is the solution to that problem? The process department would answer: "Use the reference process depicted in Fig. 4 where starting from the functional specification the software design gets created first and then based on that document the software gets implemented". The project team would answer: "Use the process depicted in Fig. 3 without the additional creation of the software design document afterwards". Both parties have good reasons for their opinions. The process department wants to make sure that everything gets documented correctly. The project team however wants to successfully finish the project in time and on budget. Too much documentation work slows things down unnecessarily.

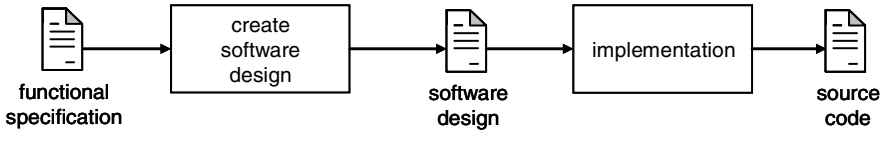


Fig. 4. Software development information flow according to reference process

FLOW provides a solution in terms of decision support rules:

- **Solid information flow:** Use a solid information flow whenever the information needs to be documented permanently. The solid state ensures traceability, third party accessibility and repeatable unmodified accessibility of the information.

- **Fluid information flow:** Use a fluid information flow whenever effective information transport is important. The fluid state (especially verbal communication) ensures the fast and extensive exchange of information.

These rules can be used to tailor a process. The result is a more effective process that produces adequate documentation.

In our case the project team and the process department should get together to decide what is most appropriate in the given situation by means of the FLOW decision support rules. This combined decision making also contributes to the narrowing of the gap between the reference and the actual development process.

In this case study we found most of the reference process problems stated in the introduction. By using FLOW techniques we could identify information flow anomalies and propose solutions. More concise and thus understandable information flows lead to better comprehensibility. The documentation requirements can be cut down by using fluid flows whenever reasonable. All this leads to a better acceptance of the reference process and therefore to better reference process conformance in the projects.

4 Related Work

Zuser et al. [14] requests communication flows to be analyzed and documented well (by models) in order to prevent redundant and superfluous flows. Efficient communication flows neither involve too many roles nor inadequate document forms. In this way loss-free communication can be established. Traceability is important to prevent the loss of already made decisions. That is why especially decisions need to be documented well. A modeling technique is introduced incorporating three kinds of communication flows: verbal communication, communication via written documents and communication via e-mail and talk memos. Furthermore, any flow holds information about the flowing data (e.g. a weekly report). The approach presented by Zuser et al. is fairly basic, because it does not mention patterns or techniques to capture these flows explicitly. Furthermore, it is limited to analyze communication in one work group and not in the whole process.

Hansen and Kautz [15] used so-called Knowledge Maps to identify and analyze knowledge flows. The technique aims to detect and optimize knowledge flows in order to distribute knowledge within a learning organization. Knowledge flows connect roles, individuals or organizational units, if knowledge is exchanged between these entities. These flows can be unidirectional or bidirectional. If necessary information about frequency, intensity, contents, context and importance can be attached to the flow. Hansen and Kautz identified four knowledge flow patterns: hubs, black-holes, springs and missing links. Hubs are connected to many other entities through flows; black-holes only consume knowledge while springs only produce it. Finally, missing links indicate situations where a flow should be established but it is absent. The authors successfully applied the Knowledge Map technique in a medium-size organization. However, the identified patterns are very basic: When using FLOW it is possible to detect other patterns like Chinese Whisper, in which information is passed along many hubs. Thereby, information is lost and gets modified.

5 Conclusions and Outlook

In this paper we presented an approach for optimization of development processes by means of information flow analysis. We started with an introduction of the FLOW project which investigates methods, techniques and tool support for information flow analysis. We then presented our experiences made using FLOW in industry. A large, complex, highly demanding and faulty reference process was analyzed. Problems from the information flow perspective were identified and solutions were proposed. This helped to clarify and optimize the reference process. Such an improved reference process will gain acceptance in the project teams and will therefore be implemented better. The gap between reference and actual development processes will be narrowed.

We first analyzed document flows in the case study because they can be extracted fairly easy from the present reference process. The document flows already contained a lot of anomalies. Most of them were of syntactical nature and hence easy to fix. The elimination of these anomalies already leads to a clearer and easier to understand process. But it still requires a lot of documentation work and some problems could not be identified with pure document flow analysis, yet.

We then performed the actual information flow analysis. In interviews positive information flow patterns were identified. Incorporating them into the reference process helps to improve comprehensibility. Aside from the positive patterns we also identified patterns indicating anomalies. Documents were created after they could have been used purposefully just because the reference process required it. The relevant information had been passed on before without the use of “solid” documents. The FLOW decision support rules help to correct this situation by establishing either fluid (e.g. direct communication) or solid (e.g. documents) information flows. Doing both is unnecessary extra work.

Some anomalies detected were due to the missing ability of most process notations to model fluid information flows in the reference process. Existing process modeling tools need to be extended to incorporate effective communication methods (conversation, e-mail, chat, video conference). One way to accomplish that would be to enable combined FLOW-process models.

References

1. Ahern, D.M., Clouse, A., Turner, R.: CMMI® Distilled: A Practical Introduction to Integrated Process Improvement. Addison-Wesley, Reading, MA (2001)
2. Schneider, K.: Software Process Improvement from a FLOW Perspective. In: Accepted for the Workshop on Learning Software Organizations (LSO, 2005). Kaiserslautern (2005)
3. Schneider, K., Lübke, D.: Systematic Tailoring of Quality Techniques. In: World Congress of Software Quality, Munich, Germany (2005)
4. Schneider, K.: Rationale as a By-Product. In: Dutoit, A.H., et al. (ed.) in Rationale Management in Software Engineering, Springer, Berlin (2006)
5. Schneider, K.: Aggregatzustände von Anforderungen erkennen und nutzen. In: GI-Fachgruppentreffen Requirements Engineering 2005. Hannover, Germany (2006)
6. Schneider, K.: Software-Engineering nach Maß mit FLOW. In: SQMcongress 2006. Düsseldorf, Germany: SQS (2006)

7. DeMarco, T.: Structured Analysis and System Specification. Prentice-Hall, Englewood Cliffs (1979)
8. Allmann, C., Winkler, L., Kölzow, T.: The Requirements Engineering Gap in the OEM-Supplier Relationship. In: LSO+RE 2006. Hannover, Germany (2006)
9. Lübke, D., Schneider, K.: Leveraging Feedback on Processes in SOA Projects. In: Richardson, I., Runeson, P., Messnarz, R. (eds.) Software Process Improvement. LNCS, vol. 4257, Springer, Heidelberg (2006)
10. Schneider, K.: LIDs: A Light-Weight Approach to Experience Elicitation and Reuse. In: Bomarius, F., Oivo, M. (eds.) PROFES 2000. LNCS, vol. 1840, Springer, Heidelberg (2000)
11. Stapel, K.: Informationsflussoptimierung eines Softwareentwicklungsprozesses aus der Bankenbranche, in FG Software Engineering. Leibniz Universität Hannover: Hannover (2006)
12. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley Professional, London (2003)
13. Holt, R., et al.: Graph eXchange Language (2002), http://www.gupro.de/GXL/
14. Zuser, W.: Software-Engineering mit UML und dem Unified Process. München: Pearson Studium, p. 377 (2001)
15. Hansen, B.H., Kautz, K.: Knowledge Mapping: A Technique for Indentifying Knowledge Flows in Software Organisations. In: Dingsøyr, T. (ed.) EuroSPI 2004. LNCS, vol. 3281, pp. 126–137. Springer, Heidelberg (2004)