Implementing Systematic Requirements Management in a Large Software Development Programme

Caroline Claus, Michael Freund, Michael Kaiser, Ralf Kneuper[1]
Transport-, Informatik- und Logistik-Consulting (TLC) GmbH

Abstract

This paper describes our experiences in introducing systematic requirements management (RM) in a large development programme. The work was done as part of a general process improvement effort based on the Capability Maturity Model (CMM, see Ch.1.2).

In our work it soon turned out that the main problems of introducing systematic requirements management are those of changing an organisation's way of working, while the technical problems are secondary. In this paper, we describe

- how the requirements management process was defined
- how we went about introducing this process
- the problems we encountered in doing so
- what we did to overcome these problems
- our "lessons learned" and future prospects.

In contrast to papers on requirements engineering such as [RBE98], this paper is mainly concerned with the process of defining and introducing the RM process, with less emphasis on describing the RM process itself.

1 Introduction

1.1 Cargo Projekt Unternehmensmodell (CPU)

The work described in this paper was done as part of a large development programme called Cargo Projekt Unternehmensmodell (CPU) at DB Cargo, the cargo division of the German railway Deutsche Bahn.

CPU is one of the largest software development programmes in Europe, with the goal to reengineer the business processes at DB Cargo. It is led by DB Cargo and Transport-, Informatik- und Logistik-Consulting (TLC) GmbH, the systems house of the Deutsche Bahn, and supported by several large consulting companies.

CPU is divided into 4 releases of 6-9 months each, with each release divided into about 20 projects for design, build and system test, plus projects responsible for integration and acceptance test and for roll-out.

[1] email: {Caroline.Claus, Michael.Freund, Michael.Mi.Kaiser, Ralf.Kneuper}@bku.db.de

Since it was obvious from the start that a programme of this size can only be brought to a successful end if all development processes are kept under strict control, defined processes and continuous process improvement were built into the project from the beginning. After a certain maturity of the processes had been achieved, it was decided to use the CMM as a guideline for further improvement.

1.2 The Capability Maturity Model (CMM)

The CMM [PW*94] was developed by the Software Engineering Institute (SEI) of the Carnegie Mellon University of Pittsburgh. It describes a framework of five maturity levels for developing software (see Fig. 1). Each maturity level comprises a set of process goals that, when satisfied, stabilise an important component of the software process.

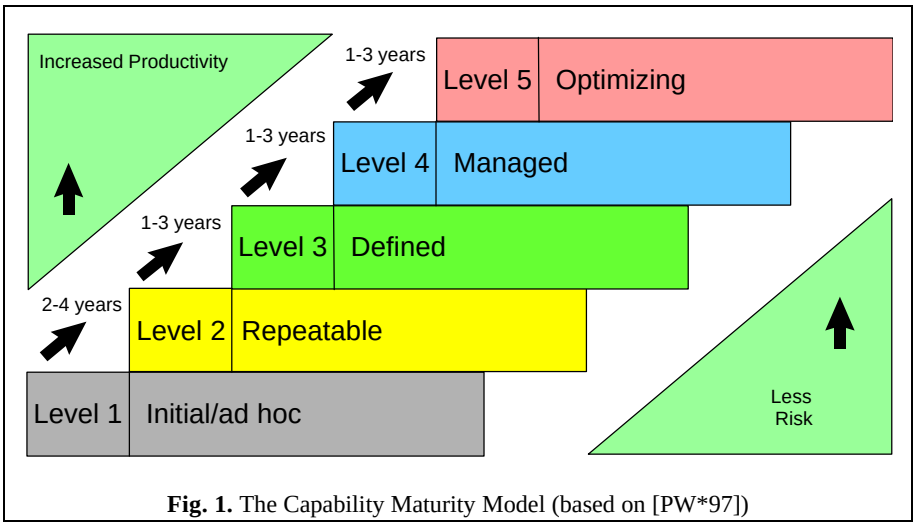

Fig. 1. The Capability Maturity Model (based on [PW*97])

On Level 1 the software process is characterised as ad hoc, and, occasionally, even chaotic. Few processes are defined, and success depends on individual efforts and heroes.

On Level 2 the basic management processes are established to track cost, schedule and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Each maturity level (with exception of Level 1) has been decomposed into key process areas that indicate where an organisation should focus to improve its software process. The key process areas at Level 2 focus on establishing basic project management controls.

The purpose of requirements management is to establish a common understanding between the customer and the software project of the customer's requirements to be

addressed by the software project. This agreement with the customer is the basis for planning and managing the software project.

The other key process areas at Level 2 are Software project planning, Software project tracking and oversight, Software subcontract management, Software quality assurance and Software configuration management

1.3 The Software Development Process

CPU uses a V-shaped development process model described in a Lotus Notes database called Lighthouse, which contains descriptions of the technical and management processes (both optional and mandatory aspects).

In Lighthouse, the software development process is divided into so-called phases, where a phase is a set of logically connected entities (activities, results, rôles, etc.). These entities within a phase do not necessarily apply to the same stage of the development process in time, e.g. requirements management is one phase. Each such entity is described in a separate document, so every phase includes one phase description (document) as well as task descriptions (documents) etc.

These documents are interlinked, so for example, there is a link from every task description to the result type descriptions used as input and output of the task.

An integral part of Lighthouse is the built-in continuous improvement process: every user can suggest new documents or changes to existing ones by just pressing a button. This feature is used intensively, with about 100 improvement suggestions (small and large) per month.

For each individual document, an owner is defined who is responsible for keeping the document up-to-date and incorporating any improvement suggestions. For each phase, a champion is defined who is responsible for the integrity of the documents of this phase. The owners and champions decide whether new or changed documents are approved or not.

Lighthouse serves as a library and is not the repository for project results.

2 Developing the Requirements Management Process

2.1 The Starting Point

Requirements were passed to the projects in several different forms: business process-related requirements were described in the so-called Master-plan, target business processes, and a function tree. Technical requirements like the technical architecture to be used were described in Lighthouse; required performance and required interfaces are part of the project description. Negotiations had to take place not only between the customer and the software projects but also between different projects and between projects and (technical or functional) architecture teams, resulting in new requirements, for example when the technical architecture changed

or when interfaces had to be agreed. Furthermore, the ”customer” was not a well-defined entity but a fairly heterogeneous group of people and organisations within Deutsche Bahn, with different and sometimes conflicting requirements.

Due to the size and complexity of the programme, the projects usually had to deal with a large number of new or changed requirements during their life cycle

A procedure to change requirements had been defined, but was rather cumbersome and therefore only used for major changes. Small changes were handled ad-hoc, with no defined process and little documentation of the changes. Similarly, there was little documentation on the decisions taken, in particular if a requested change had been rejected.

2.2 Developing the Process

As part of the process improvement effort, it was decided to define an RM process for CPU, with the goal of both supporting the projects and satisfying the CMM requirements. A project was set up for this task, staffed by developers interested in the topic.

This project was staffed with a project leader (a member of the senior management working 10% of his time on the project) and two developers with 50% to 60% each of their time.

The following tasks were performed by the project:

- analysing how requirements were currently handled
- developing and describing a new and improved process in co-operation with one pilot development project. This includes the new phase and task description documents and templates for the result-types
- defining process measurements
- co-ordinating and negotiating both the work performed and the results with similar projects responsible for the project planning and tracking and the software quality assurance processes
- developing training materials
- ensuring that the new process is compliant with the requirements of the CMM

Baseline for our work were the requirements of both the CMM and the developers in the pilot project. As a result, the work was based mainly on existing best practices within the organisation, with little emphasis on other literature. We feel that this approach was justified since we wanted to start with a simple process that solves the major problems and could be implemented within a short period of time.

As mentioned above, this work was done in close co-operation with a pilot project guiding the definition of the RM process and applying the resulting process. Al-

though this lengthened the time to develop a stable process, implementation of the process was much easier and faster.

We consider this as one of the main success factors of all new process definitions we introduced: at least one development project must be involved from the start and apply the new processes. This ensures that the defined processes are feasible and actually benefit the project rather than slowing it down.

However, it usually is quite difficult to find a project that volunteers to pilot any new processes. To overcome this resistance, the following must be ensured:

- The project must itself feel that it has a problem and the new process will help solve this problem. In this case, the project found it difficult to keep track of the many changing requirements and was looking for a process to help it cope with this.
- Strong management commitment can also help in this situation. If management makes it very clear that it wants (as in our case) CMM-conformant processes, that the projects' task is not just do deliver a product but also to help improve the processes, and is prepared to reward this behaviour, then projects will try to do so. If, on the other hand, management only asks projects about budget and deadlines, then the projects will act accordingly.

2.3 The Requirements Management Process

The resulting RM process was described as a business process on a high level of abstraction, using ARIS, a process modelling tool based on the EPK formalism (a variant of Petri nets [Kue96]).

This process diagram (see Fig. 2 for an impression of the approach used) was included in the phase description document for requirements management.

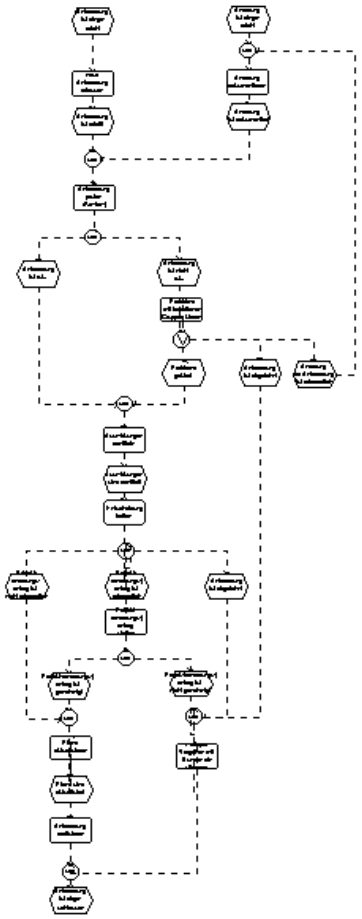

Fig. 2. Impression of the process diagramme for the phase requirements management (roles deleted)

The defined process describes management of the requirements on the level of the individual project (not the whole programme), and involves the following:

- documentation of new or changed requirements
- reviewing the requirements with regard to completeness, consistency, programmability, testability, costs and benefits, and impact on other work
- negotiating problems with all groups involved
- deciding the further steps to be taken (acceptance or refusal of requirements, issuing change request)
- implementation of new or changed requirements

- integration of the various existing requirements representations (processes, functions, etc.) into a consistent picture, and management
- metrics on the process such as number of changed requirements.

Templates are provided to support these tasks.

See Fig. 3 for a screen shot of the documents created in Lighthouse to define the RM process. Both the process and the templates provided are fairly simple, since this was one of the main requirements stated by the development projects.

3 Implementing the Process (Roll-Out)

3.1 Rolling Out the Requirements Management Process

Once the new RM process was defined, including metrics, training material, etc., a new project ECMM (Introduction of CMM) was set up whose task it is to roll out this new process (as well as the processes for the other CMM key process areas) and ensure that it is implemented by the development projects.

ECMM is a project with nine months' duration and with most of its members (6 developers) working part-time on the project, usually 50%.

Four pilot projects were selected where these processes were implemented first. These projects plan to reach CMM Level 2 in the near future.

Together, ECMM and each pilot project analysed where they currently stood with regard to the CMM requirements. For this analysis, the SEI maturity questionnaire [ZH*94] was used. Based on the results, it was decided what still needed to be done, and what kind of training or other support would be required. The responsibility to decide on the support required lay with the pilot projects, while ECMM has the role of a consultant, providing help on demand. This is considered very important to achieve process improvement - leaving responsibility as far as possible with the projects actually doing the work, not with the quality or process improvement group which will then have to fight against the development projects to achieve their goal.

To verify the implementation of the defined RM process in the pilot projects, CMM mini-assessments (a simplified version of a full assessment) and reviews are performed regularly. This helps to ensure compliance of the activities performed with the defined process and to get feedback for process improvement.

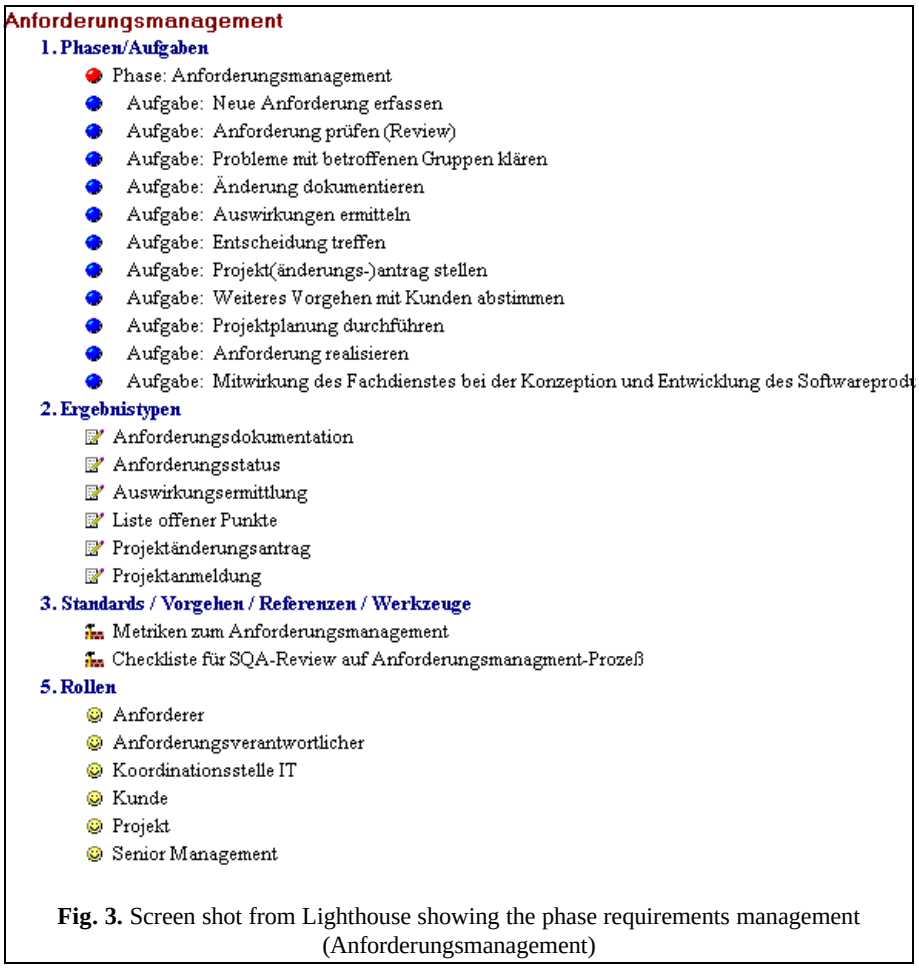

Fig. 3. Screen shot from Lighthouse showing the phase requirements management (Anforderungsmanagement)

3.2 Training and Coaching Developers in the RM Process

Based on the agreement between ECMM and the development project, training is adapted to project requirements and provided for one project at a time. Training was performed in the form of workshops in order to increase integration of the pilot project members, with ECMM in the role of the facilitator. Each of the workshop lasted about four to five hours and had about four to five participants. All roles involved in requirements management were trained, including team leaders, functional architects, and the project manager.

The workshops start off with the participants describing their current requirements handling. In the next step, they themselves evaluate this current process and identify the problems with it, and develop an improved process. This usually leads to an RM process quite similar to the one defined in Lighthouse. As a result, participants find

it easy to accept the process because it is their own. Occasionally, the process defined in the workshop even contained genuine improvements which are then included in the Lighthouse process.

After this, the templates provided for documenting, estimating and tracking the requirements are presented. The projects are free to develop their own templates or to modify these templates provided as long as they adhere to the defined process itself.

Coaching in applying the defined RM process is also offered, for example in the form of regular meetings (jour fixe) between ECMM and the pilot projects. This soon turned out to be a two-way process, with the project members learning to use the defined RM process, and the coach getting feedback and improvement suggestions.

3.3 Process Improvement

Starting with its roll-out the process is subject to continuous improvements mainly based on feedback by the pilot projects. For example, after they have used the defined process and the templates for about eight weeks, joint feedback sessions for several projects are initiated.

In addition, improvement suggestions using the usual Lighthouse mechanisms (cf. Ch. 1.3) are possible at any time. However, in the case of the RM process this is rarely used since most developers prefer to use the feedback sessions or regular coaching sessions.

4 Conclusions

4.1 Lessons Learned and Recommendations

One of the main factors to institutionalise a process is to involve project members and managers in developing the process from the very beginning. This has several effects:

- It reduces the feeling of loss and thus helps to make the transition to a new process [Bri91]. In this case, developers have to give up a way of working they know, and apply a new process that is more work to start with. The claim that they will save a lot of work later is much more convincing if they helped to define that process.
- It helps to define a process that is genuinely helpful to the people that have to execute this process, and identify problems with it early on. Development steps only thought in theory and not used in practise are often unusable and will not be accepted.

In order to increase acceptance and to gather feedback it turned out that sessions in a workshop style are more adequate than formal training. As a result of the workshop a consensus within the project is achieved.

The adaptation of the formal process takes place in the workshops, including the assignment of persons to defined roles, templates and tools. This arrangement was well accepted.

Not everything worked as we want it to when developing and rolling-out the process, and we would change some things in the future. We recommend to give attention to the following:

- Try to minimise the time between starting the work and rolling out the results. Because the expectations of the development projects were very high, letting them wait for results reduced their motivation.
- Give regular information about your activities to both management and development projects to ensure they acknowledge your work. You have to be present in their mind.
- Involve more than one pilot project when developing the process
- Ensure high involvement of both management and future process users from the start, to support the cultural change and increase the acceptance of your work

4.2 Future Prospects

We currently consider developing a Lotus Notes database for documenting, estimating and tracking the requirements. This database could be used as a workflow system (from the customer to the project, to the test team, to management, and back to the customer) and a groupware system (for the project leader, the team leaders and the team members). This way, we want to include the whole organisation, not only individual projects.

Existing (semi-) formal representations of requirements will have to be integrated better into the RM process, increasing the number of documented requirements and the granularity of description. We need to be able to track the requirements from documentation to confirmation of fulfilment by the customer on a very fine level.

4.3 Final words

We found that the real problems were those of change management (or managing transition, the internal, psychological aspect of change [Bri91]). Even though this came as no surprise, we still at first underestimated the difficulty of making the change happen and then stick, not just for a few developers but on a large scale.

The technical aspects of requirements engineering, on the other hand, were comparatively trivial. The RM process and the templates defined are fairly simple and we could easily think of more complex, more formal solutions that would be better from a purely technical point of view (e.g. such as described in [RBE98]). However, the comparatively small advantages of even a very formal approach to requirements engineering [Kne97] would not be sufficient to convince a large number of developers that this is worth the effort of learning and applying a new method in addition to all the other tools, techniques, etc. they have to learn and apply.

5 References

[Bri91] W. Bridges: *Managing Transitions: Making the Most of Change*. Addison-Wesley, 1991

[Kne97] R. Kneuper: *Limits of Formal Methods*. Formal Aspects of Computing, 9 (1997), pp. 379-394

[Kue96] P. Kueng: *Entwurf von Geschäftsprozessen mittels Petri-Netzen*. Informatik - Informatique, 5/1996, pp. 24-31.

[PW*94] M. Paulk, C. Weber, B. Curtis, M. Chrissies (eds.): *The Capability Maturity Model*. Addison-Wesley, 1994.

[RBE98] Regnell, B., Beremark, P., Eklundh, O.: *A Market-Driven Requirements Engineering Process: Results from an Industrial Process Improvement Programme*, Requirements Engineering, 3 (2), 1998.

[ZH*94] Zubrow, D., Hayes, W., Siegel, J., Goldenson, D.: *Maturity questionnaire*. Special Report CMU/SEI-94-SR-007, Software Engineering Institute, Carnegie-Mellon University (SEI/CMU), 1994.