Vodafone Pilotentwicklung GmbH

Lecture on Innovation Practices

Overview and Summary of Innovation Methods

1 Service-related software engineering

Michael Maretzke

1.1 Introduction

The birth of a project starts with a business idea. To make results of the project and the way to achieve this goals more predictable a careful project management and a well defined software development process is needed.

The three pillars of an IT project are time, budget and quality. The project have to be in time, in budget and all the required functionality have to be delivered to finish the project successfully. Beside these formal success factors the main factor for success or failure of a project remains the acceptance of the customers.

There are many reasons why software development projects can fail. Project teams need to be competent - the members as well as the team leaders - the goals should be clear in advance and the project planning should be flexible enough for changes. Also, problems between different projects, e.g. regarding rare resources or different departments, can disturb the process. Even the user can be a reason making the project fail. If the input and the briefing is incomplete and the goals aren't clear enough the whole development process takes a wrong direction. Other factors are undocumented changes in requirements, disputes in the team or too much or too less control. Most projects fail because of humans.

The main components of an IT project are document management, change management, configuration management, requirement management, risk management ,quality management and finally the software development process. These are the main topics of project management. The development process itself provides an approach to assign the tasks and responsibilities within the project team (see figure 21). It helps describing who is doing what, how and when. It defines roles, activities, artefacts and workflows.

Specification	Set out the requirements and constraints of the system
Design	Produces a paper model of the system
Manufacture	Build the system
Test	Check the system meets the required specifications
Install	Deliver the system to customer and ensure it is operational
Maintain	Repair faults in the system as they are discovered

Figure 21: Description of a development process in general

Software development without using a specific software process approach shows lacks, like blurred distinctions between specification, design and manufacture, incomplete specifications or no physical realisation of the system for testing. Adopted process are needed.

The waterfall model is one approach to deal with this problem already described in chapter 2.4.3. The process is separated in different phases with a backward relationship between these steps. It is normally used on a higher abstraction level of the real problem, but has an unreal view of the software development process.

Another approach, which determines exact requirements for software projects is the evolutionary development. It is an iterative process and suited for short-time systems. The process starts with the description. Then the specification, building and verifying phases take place, which are completing activities. The specify creates a first, initial version. After this the build phase forms intermediate versions, which are verified later. The result after the verifying is the final version.

A model which takes the risk reduction into account is the spiral model also described in chapter 2.4.3. It starts with creating a prototype. Then the client evaluates the prototype. A risk analysis takes place. Changes and modifications can be added to the project easily. The requirements are defined again and a new prototype is build (see figure 22). This process is repeated again, like in a spiral and in every round, the prototype is redefined again. At least the client gets the project result he expected. There is the danger of a never ending process.

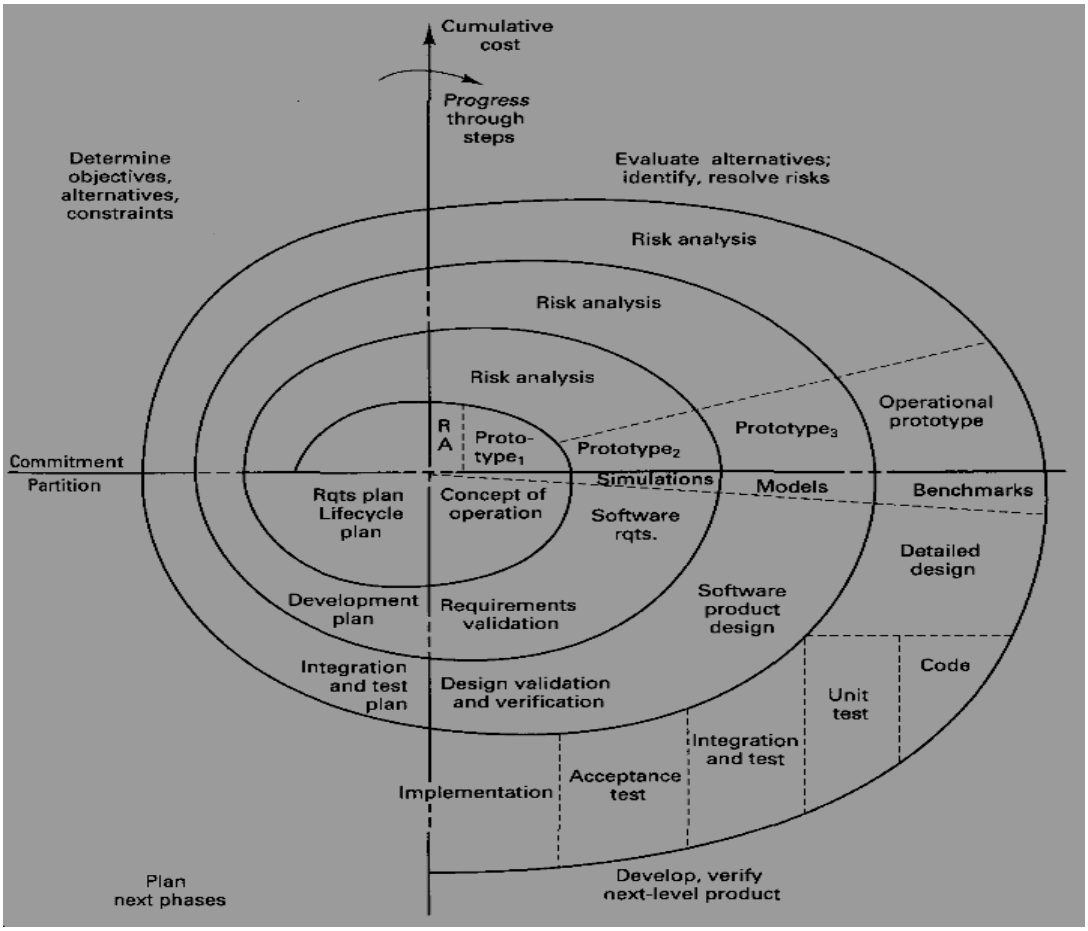

Figure 22: Detailed spiral model of software engineering

1.2 Vodafone prototyping process (VPP)

Main idea of the VPP is to support any team member in any phase of a project. VPP is therefore derived from the daily work done in the Group R&D Germany. The basic idea of the VPP is to define a reliable way from a business idea to a working prototype. During the appliance of VPP several results (so called artefacts) are produced. These artefacts are very well documented to be further used in a production environment. The process itself is based on Rational Unified Process. VPP is an incremental and iterative software development process. VPP uses Unified Modelling Language (UML) for visual modelling.

The Vodafone Prototyping Process is separated into four phases: inception, elaboration, construction and transition (see figure 23). During these different phases several workflows are executed to reach the project goal. The effort for the different workflows differ during the different phases. VPP consists of five main workflows: Requirements, Analysis/Design, Implementation/Integration, Test, User Proof, and three supplementary workflows: Configuration Management, Environment and Project Management.

Each project phase of VPP ends with a milestone. The first milestone after the inception phase contains the formulating of the scope, the planning and preparation of the project, a detailed planning for the next phase and the building of a candidate architecture.

Then the elaboration phase starts. The second milestone deals with the elaborating of the architecture, the specification of the requirements in more detail, the detailed planning for the next phase and the building and testing of the mock-up.

If the test is positive the construction of the software starts. The third milestone means summarizes the complete component development and testing process before the final transition. After the transition the prototype is completed. During the transition phase the prototype is tested again by users in an user trail. Experiences with the service and reactions of users are a valuable input for the business related aspects of the prototype.

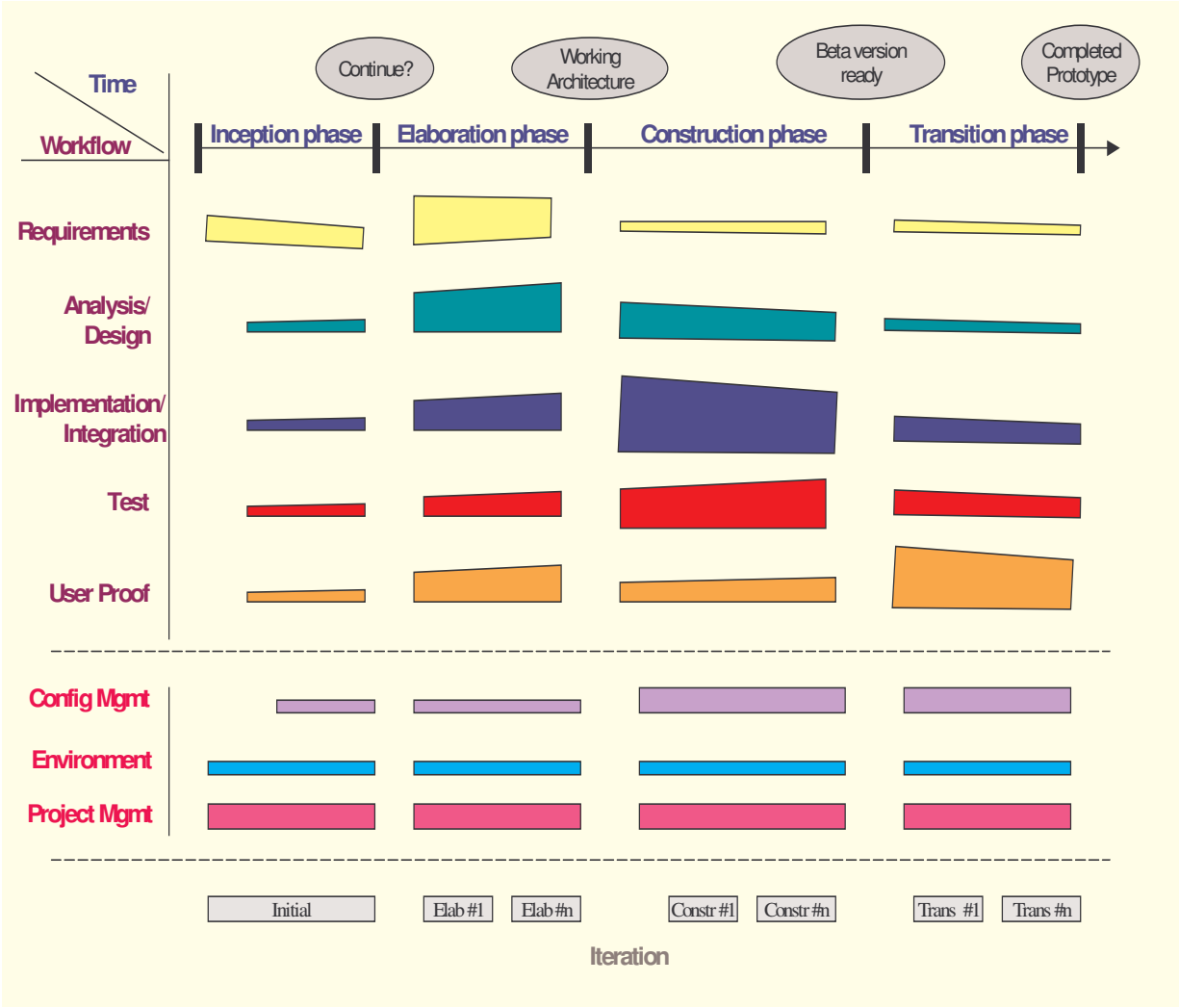

Figure 23: Vodafone Prototyping Process (VPP)

Requirements of the specific project are specified by using use cases. A use case specification consists of a use case diagram, a supplementary specification and an activity diagram. The requirements of the project are fixed incrementally in several rounds to clarify the project goal and to generate a common understanding of the project results between team members and customers.

The system itself and the architecture of the system are described formally using UML during analysis and design phase. This is mostly a task for the designer and the architect. The designer takes care of the creation of the use case realization diagram and the sequence diagrams. The architect creates the design model (the subsystem context class diagram, the class diagram itself and the design description) as well as the deployment model (deployment diagram and description).

Beside the analysis and design, the implementation of needed components and the integration of different technologies must be realised. The whole implementation is done by the implementer. Final product of the implementation is a prototype which could be used for testing and bug fixing. Release and runtime environment description is done by the architect during implementation workflow.

In addition a test designer and a tester exist, who are responsible for the test planning and the results. After the test the user proof is done. The market researcher writes the descriptions and the concept, the user trial conductor the reports and the documentation.

The configuration management controls artefacts within the project team, like documents, source codes and descriptions. Also, the correct environment is one success factor of the workflow. The developer needs special infrastructure. As well, the business analyst and the architect need different workplaces. A description assures that correct environments are provided for project work. The project management includes the iteration plan, the process report and the project plan.

1.3 Vodafone prototyping process in practice

To show how we put the prototyping process into practice I'll give an example of a project at Vodafone Pilotentwicklung (Vodafone Group R&D Germany): the mc²-Projekt. "mc²" means "Mobile Car Community". Objective of the project is to realise a prototype which shows the capabilities of mobile data communication in scope of real mobility – inside cars. The team wanted to implement a prototype which is defined in conjunction with an external partner.

After defining the project scope and formulating the business idea behind the project (we're already in the inception phase) we started to collect ideas about the use cases of the service. Collection of ideas was done by having several face-to-face meetings with our partner. In the end, we defined 17 use cases which formally describe what functionality the prototype finally should provide. Each use case was described using an UML activity diagram and also a textual description of the atomic business actions each use case consists of. The definition and finalizing of the use cases took us about 4 month.

Parallel to the use case definition, we started to look for the ideal computer platform which is capable of fulfilling all requirements, also for in-car usage. During the next phase (elaboration phase) we defined a candidate architecture which maps all functional requirements to a logical architecture diagram. Most important architectural goal was to have a highly modularised architecture. We isolated different functionalities (e.g. map visualization, client-server-communication, persistence, …), described them and formed different work packages which could be implemented in parallel during the construction phase. We extracted 25 different work packages distributed over the client and the server side of this service. The candidate architecture is described using an UML analysis class diagram and a document describing functionalities of different work packages, problems and possible technical solutions. This phase took us about two weeks.

In the construction phase – which is the logical following phase – the responsibility for implementing work packages was distributed between three developers. Regarding the different availabilities and also abilities of these developers the packages were allocated. Each developer itself was responsible for designing the work packages he/she was working on. We followed the goal of creating a modularised architecture, even inside the work packages, having reusability in mind. Finally, after the construction phase, we formed a prototype with more than 220 (!) classes and more than 1,5 MB of source code.

The construction phase itself was one of the most interesting steps from a technical point of view. Several problems during selection of the right pc hardware and integrating external devices with correct drivers occurred. Another problem we had to cope with was the beta stadium of the pc hardware we finally selected as the right platform. Other problems were:

- the GPRS connection we used didn't react as we expected
- the selection of the right software components to integrate was not so easy as thought
- complexity of the whole system was underestimated from the beginning

Finally we managed to implement a fully functional prototype following the requirements defined within the inception phase. The complete implementation was done in about 10 weeks.

Also, the transition phase was very exciting. The complete pc hardware was installed into a real car provided by our partner. It was a real pleasure to test the whole system. After finishing the system test we started presenting the service inside Vodafone and also within the management of our partner. Final step of our transition phase was to define next steps and start a discussion about features of possible next versions.

1.4 Lessons learned

Most important task building of a complex software system is to fix the requirements early and let all stakeholders agree on the system functionality. After having solid requirements, the next important step is to analyse the system and to identify potential problems. The earlier the problems are addressed, the more time is left to think about solutions. After the analysis phase, a solid time for thinking and also designing the final software solution is a good idea. In general it is really very (!) important to follow this truth: “It’s always better to think first and then start doing it”.