NIMSAD Evaluation of the Rational Unified Process

Introduction

The Rational Unified Process is the information systems methodolgy most widely in use today. The main contributers are the three amigos **Ivar Jacobson**, **Grady Booch** and **James Rumbaugh** who also designed the Unified Modeling Language.

It is mainly based on the *Ericsson Approach*, *Objectory* and the *Rational Approach*, which were combined 1995 to the *Rational Objectory Process*. The Unified Modeling Language together with the expirience of from Rational Inc. acquired software tool companies formed the Rational Unified Process.

The Unified Process is a software development process, that is the set of activities needed to transform a users's requirements into a software system, but it is also seen as process framework, which can be specialised for different purposes.

The three main aspects of the Unified Process are that it is

- use-case driven
- architecture-centric
- iterative and incremental

The basic sequence of an RUP project according to [2]:

- Get the team together.
- Decide what system will be built (there is apparently no other option than to build a system).
- Build a use case model and UI prototype.
- Use the UML process extensions to build an analysis object model.
- Segue into the more conventional UML stuff to do the design - class, state, sequence diagrams and the like.
- Think hard about architecture while you assign the designed classes to modules and packages
- Test against the use case model. RUP provides some excellent guidance on testing.
- Transition to live system and do the post mortem.

Element 1: The Problem Situation

The methodolgy is concered about the context. This is especially seen in the two core workflows *Requirements* and *Analysis* which are mostly important in the first to phases (*Inception*, *Elaboration*), but come all along the process, because of it's iterative nature.

As mentioned above the Unified Process is *use-case driven*. It means that the whole process is controlled by the way the users interact with the system. Every produced model can be traced back to a use case.

That the Unified Process is architecture-centric means that also from the beginning of the process there is a strong emphasis on the architecture of the information systems. This includes hardware and frameworks used, distribution and programming languages.

Furthermore there are two optional concepts in the requirements workflow which support the capturing of the business environment:

- *Domain Model* - a UML class diagram which captures the most important types of objects in the context of the system.
- *Business Model* - technique for understanding the business processes of an organization. It presents a business like a use-case model for a software system from the usage perspective and outlines how it provides value to its users. It also has a *Business Object Model* which depicts the business entities like the domain model.

But apart from that is very little said about the problem situation. It is a positivistic methodology. It assumes that one is only concerned with system specification. RUP has nothing to say about business requirements or business process modelling - exept that use cases are enough. [2]

Element 2: The Methodology User (Intended Problem Solver)

RUP handles the methodology user with the concept of *Workers*.

Evaluation of the *Mental Construct*

The Intended Problem Solvers and their different roles are *Workers*. Each worker represents an abstraction of a human with certain abilities needed in in software engineering. When a project is staffed, a worker represents the knowledge and abilities that someone needs to take on the job as that worker in the project. In the methodology itself however is a worker described primarily through the responsiblities.

Desirability Levels of the *Mental Construct*

What a user should know depends much on the role the user fulfills in the process, i.e. which worker he/she is. Every worker should have a knowledge of the UML, a good picture of the overall process and his/her special responsibility in the process. Commonly identified workers are:

System Analyst and *Use-Case Specifier*: The workers with probably the highest level of skills. They got to have skills in analysing business process and organisations, experience and a good reasoning ability.

User-Interface Designer: Obviously technical and graphical skills are important but also a feeling for usability

Architect: The architect needs also high skills, but more on the technical side but he/she also needs some understanding for the use-cases to fulfill his/her goals.

Use-Case Engineer, *Component Engineer*, *System Integrator*: They need mainly technical skills, because the only design and implement according to the identified use-cases

Test Designer: High technical skills but also a good understanding of the processes.

Integration Tester, *System Tester*: technical skills

Element 3, Stage 1: Understanding of the Situation of Concern 

This stage corresponds very much to the core workflow *Requirements Capture*. It provides different startpoints, as a business model, a domain model or a complete, detailed requirements specification from the client.

Afterwards certain steps are performed. First a feature list is developed which grows and shrinks during the process, because of the iterations. Secondly the user should get an understanding of the system context. To express the context of a system there are the two approaches *Business Model* and *Domain Model*. The naming of the objects is also used to develop a glossary of terms that will help to communicate. The third thing is the capture of the functional requirements by the help of use-cases. Finally the nonfunctional requirements are captured.

It has a strong emphasis on the *reflection-in-action* through the iterative nature of the process. The requirements and boundaries of the system are re-evaluated with every iteration

Investigation Models and Techniques

As mentioned above a feature list is developed, which might include status, estimated cost or priority. This helps managing the requirements during the process.

The in Element 1 explained models *Business Model* and *Domain Model* can be used to understand the system context and capture the requirements.

Still in the *Requirements* workflow there are use-case models used to describe the diagnosis. They describe how the users interact with the system. Each type of user is represented as one or more actors. Each external system that the system interacts with is also represented as actors. The flow of events for each use case can be captured as a separate textual description of the use case's sequence of actions. Also statechart diagrams can be used to describe a use-case.

Element 3, Stage 2: Performing the Diagnosis 

To perform the diagnosis, RUP uses the diagrams developed in the requirements workflow. They are more on a conceptual or logical level and nothing is said about the physical level.

RUP admits that there can be requirements which can't be automated and solved by an information system.

Element 3, Stage 3: Defining the Prognosis Outline 

RUP doesn't really involve the current state apart from the requirements worklflow, but already there the decisions are made what the desired situation is. There is no comparison between the current state and the desired state. There is also no direct questioning about the expectations and requirements of the client, but RUP suggests that they should be worked out in workshops where the analysts and the client participates.

Element 3, Stage 4: Defining Problems

RUP focuses on the domain of information systems. Business modeling is just used to grasp the context of the system, but not to identify problems in the business.

Element 3, Stage 5: Deriving the Notional System

The core worklflows requirements and analysis are used in this stage. From there on the Unified Process is very strong. It tells little about the current state and the problems there but has straight guidelines to create the requirements, as mentioned above, and how these requirements are put into practice.

In the analysis workflow are the use-cases from the requirements ordered in a new way and refined to fit the view of the developers of the system in opposition to the requirement models where the language of the client is used. Furthermore a analysis model is developed, which is a UML class model with some special stereotypes. These are entity classes, boundary classes and control classes.

Element 3, Stage 6: Performing Conceptual/Logical Design

This stages mappes to the *Design* workflow of the Unified Process. The input to this workflow is the analysis model. It produces the design model, that is a blueprint of the implementation. This is mainly done through class diagrams, which are use-case realizations. Additionally there are interaction diagrams which model the sequence of actions in a use case. This can be a collaboration or a sequence diagram, wheras the latter emphasises the order in time.

All of this is accompanied by a textual description called *flow-of-events-design*. The *implementation requirements* are also a textual describtion, but captures nonfunctional requirements which should be minded at implementation.

The system is divided into subsystem and their interfaces are identified. They are placed on different nodes in the *deployment model*. The *architecture description* is a view of the deployment model.

For some objects in the model it is suitable to model the behavior via a statechart diagram. It describes the different state transitation of the corresponding design class.

Element 3, Stage 7: Performing the Physical Design

The physical design is not really separted from the logical design in the Unified Process. This is achieved through the direct mapping of classdiagrams to object-oriented programming languages. Furthermore is the component engineer who designes the subsystems also the one who implements it. So the stage 7 can be seen as included in both workflows design and implementation.

However there is an implementation model which describes how elements in the design model are implemented in terms of components such as source files, executables, and so on. The implementation model also describes how the components are organized according to the structure and modularization mechanisms available in the implementation environment and the programming language in use, and how the components depend on each other.

The important model in the implementation is the component, which is the physical packaging of model elements. That can be executables, files, tables or documents. There's also an architecture description which contains an architectural view of the implementation model, depicting its architecturally significant artifacts.

Element 3, Stage 8: Implementing the Design

This is the part of the implementation workflow where the real implementation is done. The implementation models are put into practice and the subsystems are integrated. Finally the components are deployed on the nodes.

The Unified Process has additionally an emphasis on the test of the system. There's a core workflow called *test*, which is also performed during each iteration. It produces test models which are based on the use-cases from the earlier workflows. Here again the emphasis on the user requirements can be seen. A use-case traces directly to a test-case. For every test-case are one or more test procedure developed. Some test can be automated by test components.

Element 4: Evaluation

RUP has no dedicated evaluation phases but it does evaluation in all phases. The transition phase is for the evaluation of the whole project. At the end of the transition phase, which is also the end of the project in budgetry terms, the project manager convenes a group to review actual schedule time, person-hours, cost, defect rates, and such other metrics as the company may employ, in relation to the planned numbers for the entire project to

- See if the projet attained the planned goals.
- Ascertain reasons why it did not (if that is the case
- Add the project's metrics to the company's metrics database for future use.

[1]

Also the economical success is evaluated using the business plan and the project manager assembles a small group to assess the transition phase and to conduct a postmortem of the development cycle as a whole.

Summary

RUP has several positive features compared to older approaches. It uses UML throughout and includes several specifically OO techniques. Most important among these is its use of use cases to drive specification and testing. It is thoroughly iterative and incremental and, for an existing user of Rational's tools, it is well integrated with them. [2]

RUP is said to be architecture-driven. This too is apositive feature, though it must be said that it takes the restricted view of architecture as mere structures. Again this good providing that one is only concerned with system specification; RUP has nothing to say about business requirements or business process modelling - except that use cases are enough. [2]

One advantage of RUP is also a disadvantage: being tied to the tools of one supplier makes many organizations feel uneasy and makes it harder to take advantage of tool innovations as they arise. There is also nothing in RUP about GUI design. Metrics are not specified in RUP, though they are expected to be collected. [2]

Perhaps the worst feature of RUP, as a modern process, is its sheer size: well over 1,700 pages. That's not really lightweight. [2]

Further Information:

[1] Jacobson, Booch, Rombough: The Unified Software Development Process, 1999, Addison-Wesley
[2] Ian Graham: Object-Oriented Methods, Principals & Practises, 3rd Ed., 2001, Addison-Wesley
[3] Nimal Jayaratna: Understanding and Evaluating Methodologies, 1994, McGraw-Hill
[4] The Rational Unified Process Homepage: http://www.rational.com/rup
[5] Completing the Unified Process with Process Patterns: http://www.ambysoft.com/unifiedProcess.html
[6] Phillipe Kruchten. The Rational Unified Process. An Introduction. Addison-Wesley, 2000.
{7] Nilgün Özek. Evaluation der OOSE-Methode mit Hilfe des NIMSAD-Frameworks. Studienarbeit. Universität Hamburg, 1998.