Theme 4.2 SE, SE Learning Task 1, M. Kindermans & M. van Mastrigt, group 10b.

Assignment 1

(a) Describe in a few lines what the most important differences and similarities are between RUP, XP and the waterfall-approach.

Opposite to XP and the Waterfall-approach, RUP focuses on well described documents (roles, artifacts, activities) that are written up front. This helps to prevent problems in the further process of the project. XP focus more on reacting to problems/change rather than preventing it. Waterfall has no proper reaction to changes and expects there are no disturbances.

In the Waterfall-approach the cost of change is high, whereas in the RUP it is less, and in XP it is low.

Sources:

- http://www.slideshare.net/maheshpanchal1/comparison-of-rup-agile-method-xp-for-projects
- http://programmers.stackexchange.com/questions/45699/differences-between-a-unified-process-and-an-agile-project-plan

(b) Create your own compact software process aimed at a student project with a size of about 3 weeks and 4 students. Describe:

- The roles which your process contains;
- The disciplines which your process contains (if you decide to leave certain disciplines out then explain why);
- The artifacts that each discipline should produce. Again give your argumentation (‘rationale’). Why do you choose these artifacts?

Roles

- **Stakeholders**
  - Product Owner
  - Steering Group
- **Business modelling**
  - Team Leader
  - Development Team
- **Requirements**
  - Information Analist

    - Use Case Designer
- **Analysis and design**
    - Software Architect
    - User Interface Designer
- **Implementation**
    - Developer
- **Test**
    - Test Manager
    - Tester
- **Deployment**
    - Tool Manager
    - Integrator

Disciplines

- Stakeholders
- Business modelling
- Requirements
- Analysis and design
- Implementation
- Test
- Deployment

Artifacts

Stakeholders

- Acceptation Plan
- Product Backlog
- Release Burndown Chart

Regarding the artifacts for the Stakeholders, we chose to only deliver the documents necessary. The Acceptation Plan gives insight in what acceptable working products are. The Product Backlog contains functional and non-functional requirements. And finally the Release Burndown Chart shows the total amount of progress during the project - and what of it is accomplished. The combination of these three artifacts give good insights on the final state of the iteration to the stakeholders.

Other artifacts (like Business Process Model, DTAP-environment, Training Material, or Change Proposal) are out of the project's scope.

Business modelling

- List of Risks
- List of Bottlenecks
- Iteration Plan (Task Board)
- Iteration Burndown Chart

We chose List of Risks because every project does have certain risks. One of the most common risks is the deadline. It is wise to describe those risks in a document. This way, everybody knows what the risks are and how likely it is they will occur.

We chose List of Bottlenecks because most projects do have certain aspects which could hold up the entire project. If you make a list of those up front, it will be easier to anticipate to these. It maybe will be possible to avoid those bottlenecks.

We chose Iteration Plan (Task Board) because doing this will make it clear for every participator what will be done during this iteration. Questions like "What do I have to do now?" will be less likely to occur.

We chose Iteration Burndown Chart because this will give better insights in the project its progress.

We did not choose for a Software Development Plan because it is a bit of an overkill to create such a document for a small project like ours. Because this is a smaller project you can easily anticipate on dependencies.

We did not choose for the Progress Report artifact because the documents would likely be the same every time. There might be some slight differences between them, but no noteworthy ones.

Requirements

- Vision
- Glossary
- Use Case Model
- Use Case Specification

These four artifacts aim to make communication between the product owner and product group as transparent and clear as possible. Through documented vision, and declaration of technical terms used in describing this Vision, the project goal will be clear between both groups. The Use Case Model and Specification aid in the clarification of the individual elements.

A Navigation Document would make the vision even clearer, yet because of the limited amount of time we deemed it out of scope.

Analysis and design

- Proof of Concept
- Design Model
- Data Model

We chose Proof of Concept because it is wished by most product owners to see that their product can be realized. Although it may be that it's not finished, they do have proof that it can be realized.

We chose Design Model because it useful to see how and where some parts are made. This will improve the reuse of code and is useful for possible switches between task and new group members.

We chose Data Model because every member of the group must have clear how persistent data is stored. Persistent data can be used by every member of the group. It is very time consuming asking one member of the group how data is stored, over and over again.

We did not choose for Architectural Prototype because this project is not big enough that every piece of the project should be checked seperately. We use tests to test seperate parts and seperate functions of the code.

We did not choose for Software Architecture Document because it provides "a complete overview of the architecture", which isn't very big in our case. So it a bit over an overkill.

Implementation

- Working Code

Delivering a working project (and the related Working Code) is an implementation artifact. This includes code documentation, a description of re-usable components, information about used frameworks, and possibly the configuration settings.

We decided not to include the Use Case Realization document as an implementation artifact. Use Case realization itself is very important, yet delivering a document on the technical implementation of each use case, is out of the project's scope - due to limited time and available developers.

Test

- Test Design
- Test Suite

We chose Test Design because it is not necessary to describe which parts will be tested. Doing this, a developer can run the tests. If the tests succeed, he knows that his part of the code is (or is not) tested. This way, everybody knows which parts of the code are covered by tests.

We chose Test Suite because by using a suite, it is certain that everybody is testing in the same and

correct way. Situations like "Oh, but it worked on my computer." are avoided.

We did not choose for Test Plan because in the Test Design and the Test Suite are already described which parts of the code are tested and how they are tested. Creating another single document for this is unnecessary.

We did not choose for Test Report because it is more important that tests are passing. We are writing tests just for the developers. When a test fails, it will be fixed. There is no need for discussions and meetings about why and where tests are failing.

Deployment

- Development Environment
- Delivery Document
- Delivery

We chose that the Deployment discipline should deliver a fully working development and documentation on each fysical part of this environment. Also the Delivery Document will contain the realised Use Cases and (if necessary) the known issues and applied bug fixes. The final Delivery artifacts combines all the previous into one conclusion with additional information about supporting work products (tools, third party dependencies etc.).

These should give the Product Owner a complete picture of what has been accomplished in the iteration, plus the ability to use the product without having to be guided through each part of the project setup process.

Assignment 2

Which artifacts does RUP define concerning testing?

RUP defines the following artifacts:

- Test Plan
- Test Design
- Test Suite
- Test Report

Describe what each does in a few sentences and what the relationships are between them.

The **Test Plan** artifact is a document that describes how and in what environment the testing is done. It defines the test scope, which indicates in how much detail the tests must be written. This level of detail can be measured by code coverage percentage. It describes the type of tests and which tooling is used. Also it gives more information on how the tests are organized and how the test environment is set up.

The **Test Design** artifact is a document describing what elements are tested (for example which Use Case scenarios and non-functional requirements) and what test cases will be used for it. The test design can also include test scripts and test data.

The **Test Suite** artifact is a summary of the tools that are used for testing. These can include:

- Automated tests (for example unit-tests or continuous integration platforms);
- The testing environment (framework);
- Test data and test scripts to install, configure, or prepare the tools;
- Tools used for manual testing.

Maintaining and improving the test suite is a shared responsibility for the developers and the testers.

The **Test Report** artifact is a document that describes test activities, test results, and test summaries per iteration.

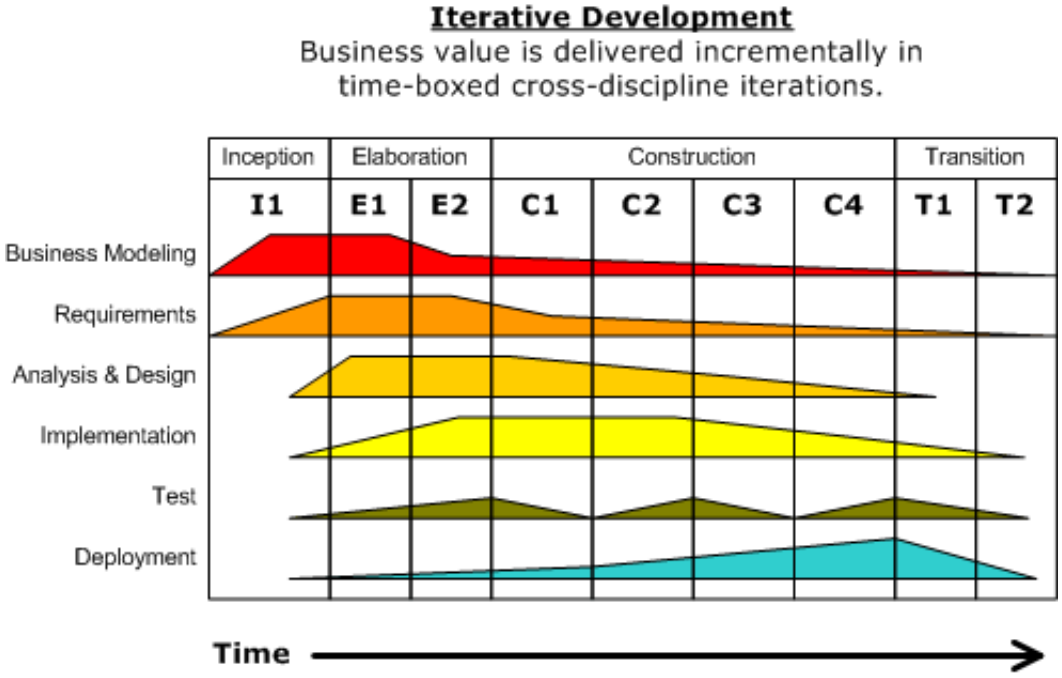
Source: http://rupopmaat.nl/naslagsite2011/index.html

At which point in time are the artifacts produced?

As a best practice to verify the software quality testing should be a major part of the project at any point of time. Testing becomes heavier as the project progresses but should be a constant factor in any software product creation.

This means that the **Test Plan** is writtin at the start of the project. To lay a clear foundation for the

testing enviroment. The other artifacts (**Test Design, Test Suite, and Test Report**) continue to evolve over time. Especially the **Test Report** can be useful to determine certain bottlenecks in the project. It is therefore wise to produce this report after each iteration.



As seen in the RUP hump chart above, testing is done throughout the entire project. Yet it does not have to consume a big chunk of the de project's time (comparing the implementation and testing disciplines in the figure above).

Source: http://en.wikipedia.org/wiki/Rational_Unified_Process