



**Applications in Practical High-End Computing - Group Project**

Assignment - "Workflow"

**Project Management**

Supervisor: Dr Stuart Barnes

Authors: Mateusz Gołąb

Csaba Kerti

Jakub Kiełbasa

Zsolt Kollarits

Course: CSTE / SETC

**Table of contents**

[1. Summary 3](#_Toc318718685)

[2. Literature review 4](#_Toc318718686)

[3. Scrum Methodology 5](#_Toc318718687)

[a) Theory 5](#_Toc318718688)

[b) Why Scrum? 5](#_Toc318718689)

[c) Releases, Sprints 6](#_Toc318718690)

[d) Roles 6](#_Toc318718691)

[e) Meetings 6](#_Toc318718692)

[f) Scrum Summary 6](#_Toc318718693)

[4. Responsibilities/Roles of team members. 7](#_Toc318718694)

[5. Costs estimation 8](#_Toc318718695)

[a) Costs of finished work 8](#_Toc318718696)

[b) Costs forecasting - "how much to finish" 8](#_Toc318718697)

[6. Notations 9](#_Toc318718698)

[a) Use case 9](#_Toc318718699)

[b) Requirements 9](#_Toc318718700)

[7. Used Tools 10](#_Toc318718701)

[a) Enterprise Architect ver. 9.2 Trial Ultimate 10](#_Toc318718702)

[b) Tortoise SVN ver. 1.75 10](#_Toc318718703)

[c) Google Code 10](#_Toc318718704)

[d) Microsoft Project 2010 10](#_Toc318718705)

[e) Eclipse Indigo Service Release 1 11](#_Toc318718706)

1. Summary

Preparation of c

1. Literature review
2. Scrum Methodology
3. Theory

Scrum itself is an example of Agile methodology. It is quite new approach to project management issue. It fulfils all Agile manifest's assumptions. Scrum should be used in case of small and medium size projects because in the largest ones whole idea is being misused because of the size of a team.

Few facts about Scrum:

* Whole life-cycle of project is divided into *Releases*.
* *Releases* are divided into iterations, in this methodology called *Sprints*.
* *Sprint* should usually be not longer than 30 days and not shorter than 5 days.
* *Release* should contain between 4-12 *Sprints*.
* List of tasks in *Release* is called *Product Backlog*
* List of tasks in *Sprint* is called *Sprint Backlog*

Roles in Scrum:

* Scrum Master - person who gives tasks to others and organize work
* Development Team - all persons responsible for performing tasks
* Product Owner - usually external customer

Events:

* Daily Scrums - daily meetings (usually standing up) where each member of Scrum Team answer 3 questions:

- What have I done since last Scrum?

- What impediments have I met?

- What am I going to do next?

* Sprint review/planning meetings - at the beginning and at the end of each Sprint Scrum Team is reviewing results of last works and planning next Sprint Backlog.

1. Why Scrum?

Main motivation for starting using Agile techniques was quite simple - it is very popular in contemporary, developing world and most of companies working with Software Developing are using it. Since none of us was fluent in this matter we decided that it is worth a shot. Other reason was, that Scrum is actually simple and well-defined way of managing small team, and because we were very short in time, we have realized that we do not have weeks for learning more sophisticated methodologies.

1. Releases, Sprints

Because of the duration time of the project there was actually no point of making any division into several *Releases*.

*Release* has been divided into 4, very short Sprints. The reason for such a situation was that there are 4 students in our team, and 4 main responsibilities (requirements, design, implementation, tests), this way we hoped that we could focus on one job at the time. Another reason for that choice is placed in another header - about Roles.

**TODO DIAGRAM FROM MS PROJECT!!**

1. Roles

Development Team - main problem with dividing roles between us four was that (because of time) we have to use all of our human resources for hard work in Development Team. (Which was actually not only developing but also designing, testing etc.)

Scrum Master - natural problem was lack of Scrum Master when all workers are being used as a developers. We have solved this problem by moving position of Scrum Master within us. (That is another reason why we have decided to split *Release* into 4 *Sprints*.) Each of us was holding the Scrum Master position during one Sprint.

Product Owner - we have mocked our customer by sending them mails with questions which appeared during our work.

1. Meetings

Daily Scrums was very convenient way of exchanging information between team members. Ideally, Scrum should last about 15 minutes in a bigger team than ours. We have not achieve this number (record was 4 and half hour), but, as time went we get better with organisational issues. Another problem was the time of meetings. Ideally it should be always the same hour, but of course, it was impossible because of other classes etc. Because of time pressure we have decided to meet each day including Saturdays and Sundays.

1. Scrum Summary

We have tried to apply Scrum methodology to our project and outcome was generally not bad. During these few weeks we get familiar with general rule of whole process and get some experiences about Agile techniques. Unfortunately, we have met many impediments which have complicated our trials in this matter.

1. Responsibilities/Roles of team members.

During work on this project we have to split responsibilities for different aspects of it to different team members. Although, we have assigned each stage of project to separate persons, we all have been working on each of the stages. This mechanism was quite simple - one persons responsible for particular stage was controlling and monitoring it, explaining what else should be done, and all others were helping him in fulfilling this duties. We have, indirectly, connected this with our Scrum meetings (we have tried to achieve situation where Scrum Master would be a person responsible for, currently, most important stage).

One of main responsibilities of a person responsible for given stage was to prepare reliable report. Another, even more important, aspect was that this person should pass very clear information to person responsible for following stage about situation of the project (very careful explanation about all details connected to it).

Below, we will try to briefly describe more precise responsibilities of particular people.

1. Project Manager

Each of us four was Project Managers for some time. We achieved this situation by mixing Scrum methodology and responsibilities division. When we started a stage for which one of us was responsible, he become Project Manager and was making most important decision. This approach was quite risky, because in mean environment one person can overuses their power as Project Manager e.g. because somebody else was giving them too many tasks.

We have avoided this situation by spending a lot of time on communicating among team. There was (hopefully) no single situation when person responsible for some task was (explicitly) unsatisfied because of it. Price for this situation was quite high - we have spent usually not less than 2 hours on daily discussing about project.

Very good thing about this approach was that all of us has chance to meet problems and concerns connected to managing human resources - which was even better solution because of the fact that none of us was assigned as a Project Manager by lecturer - taking this position by any of us would be simply unfair.

1. Requirement Manager - Jakub Kiełbasa

First stage of each project is about collecting requirements, doing feasibility study and form requirements into more formal shape, so that another stages' Managers would be clear about them. Jakub was responsible for that part. He has been controlling process of forming Functional and Non-functional requirements as well as controlling process of creation and designing Use Case Diagrams. Most of this jobs was very strongly affecting Design Stage and Test Stage as well. He was probably most hated person within team.

1. Test Manager - Csaba Kerti

Stage of Testing was probably most complicated one because, contrary to all others, there is no way of chronological placing it strictly between two other stages. It was continuous process which started just after Requirements' part finished and finished just after Implemented part finished. Person responsible for this stage was Csaba. He was responsible for creating acceptance tests - connected to requirements, integration tests - design and unit tests - implementation. He was probably most frustrated person in the team.

1. Design Manager - Mateusz Gołąb

Stage of Design follows the stage of Requirements and precedes stage of Implementation. It was about creating technical, design of system being created. It includes modelling whole workflow system and find out most appropriate design pattern for our case. Person responsible for this stage was Mateusz. He was appropriate person for this function since he was famous because he organises even his clothes into units. He was controlling process of creation of logical diagrams of project and was responsible for system configuration. He has to assured that project Design will fulfil all Requirements gathered in previous stage. This job strongly affects Implementation Stage.

1. Implementation Manager - Zsolt Kollarits

Stage of Implementation may be considered as finalization of our work. If our System was prepared correctly and efficiently than Implementation should be fairly easy otherwise this stage may be a torment. Person responsible for this part was Zsolt, who strongly refuses to accept the fact that previous stages was prepared excellently. Because of the fact that the entire project Implementation would take millennia we had to choose part of it to implement as a Prototype. He was in charge of it. Implementation form was strongly impacting part of Test i.e. unit tests.

1. Costs estimation
2. Costs of finished work
3. Costs forecasting - "how much to finish"
4. Notations

In order to simplify communication within our team we have decided to introduce some notations, which should decrease number of misunderstandings. Another point of this idea was to simplify documentation process - without unique identifiers it would be very ambiguous. Below we describe few notations used within our project.

1. Use case

UC\_XY - typical use case identifier. Where:

X - stands for actor most connected to current use case.

Acronyms are: U - User, A - Administrator, S - Scientist, M - Managers.

Y - index of use case for X actor.

Examples:

UC\_A3, UC\_M1.

1. Requirements

FUX - functional requirement where X is requirement index.

NUX - non-functional requirement where X is requirement index.

**TODO WHAT ELSE??**

1. Used Tools
2. Enterprise Architect ver. 9.2 Trial Ultimate

Main tool, used in creating System Requirements and in further stages was mentioned above program. It is a very advanced, powerful tool which allows software engineer creating all kinds of UML diagrams and assures support and necessary documentation. We agreed that, for so important project we will use Ultimate edition of it - there was no place for risk.

This software is quite expensive (Ultimate edition costs $849) but vendor delivers free Trial 30-day version which was perfect for us, since project's life-cycle was about 20 days.

All diagrams presented in this document are prepared within EA. After few weeks of usage we can recommend it.

1. Tortoise SVN ver. 1.75

Since we have decided to use Version Control in our project we have to choose server where our code would be stored and client for accessing it. As a client we decided to use mentioned above program.

It is free, user-friendly software which gives quite a lot of possibilities. It is very easy to locate source change, compare different files etc. It also supports all authentication mechanisms.

Only problem with this tool was cooperating with Enterprise Architect - unfortunately project prepared in EA is represented as a single physical file. It caused some problems with merging our works.

1. Google Code

As I server where we could store our whole repository we have chosen Google Code. It is free, very simple in usage program which is especially convenient for persons who use Google accounts for other purpose - whole authentication is then automatic.

It can be easily used together with Tortoise SVN so that there are no problems with incompatibilities etc. It can be also accessed straight from web browser so that there is no need for installing new software on your desktop.

1. Microsoft Project 2010

For generating Gantt charts we have been using this, Microsoft software. Generally it offers a lot more but we have used only a small part of its possibilities.

1. Eclipse Indigo Service Release 1

Since we have decided to use Java programming language in our project we have actually choice only between NetBeans and program mentioned above. We have decided to use Eclipse because, in our opinion, is more stable and open for different kinds of plugins. It is also very well documented software which is very important feature of Developers Environment.