ETSF01

Software Project Improvement

Group 8

Final Report

Robin Larsson (ic08rl3@student.lth.se),
Erik Lindstén(ic08el2@student.lth.se),
Pétur G. Hjartarson(ic08ph0@student.lth.se)
and Hlöðver Tómasson (hlodver.tomasson.656@student.lu.se)

May 20, 2011

Contents

1 Introduction

This document describes software process improvements for a fresh year students software engineering class project. We suggest simple changes on current software model. This document covers description on the current process and the new process. Furthermore a description is provided on how the new process should be implemented and ways to measure its performance.

1.1 Context Description

This software process improvement is aimed at fresh year students that studies Computer Science- and Information- and Communication technique at LTH. The students are using the well-known, easy to understand yet hard to master, Waterfall model. The students have fairly little coding experience, consisting of an introduction course in Java programming and an advanced course that addresses abstract data structures.

The project itself is not hard to implement if clear requirements are available, but if they are not it will be a lot harder to finish the project in time. The main focus of the project is not the implementation (coding) part, rather it is to get a clear understanding of how software is developed by following a software development process, from requirements analysis to the launch of the software.

1.2 Method

We base our method on four elements: *Baseline process description*, *target process description*, *implementation of target process*, and finally *measurement and control.*

We use the PROFES [2] improvement methodology as a base and guidance to our work, e.g. the PROFES *Characterize* phase steps are used as input for our baseline process description and the *Plan* phase for the target process. This work has input from the first three phases of PROFES because we do not execute the process changes which we only speculate about in the discussion section at the end of this report. We however identify risks of executing the process changes and make speculations of the results of our improvement suggestions.

1.3 Issues

When writing requirement and design specifications used for implementation it is difficult for the inexperienced programmer to know exactly how to write an useful specification. Even with basic knowledge from lectures it can be difficult to know when the task is finished due to lack of experience. Some documents may have to be handed in several times before finally getting approved by the customer (teacher). As the waterfall model is used, this results in inability to move forward to the next step while the customer approves or disapproves of the documents, and inability to move forward in turn leads to the developers not fully understanding the problems and *why* the documents were not approved.

1.4 Goals

The proposed improvement plan introduces an iterative process model, which the baseline process does not have. The goal is to increase the team's knowledge and understanding of the whole development process, resulting in clear requirements and test cases with higher quality.

Experience shows that the requirement specification is what takes the most time getting qualified. Rough estimates for the current process indicate that 100% of the groups need to hand in the requirement specification at least twice, while around 15% of the groups need to hand it in third time. The goal is to reduce the average number of hand-ins of the requirement specification to get approved by 19%, which would mean an average of 1.75 hand-ins per group. Since current data is unavailable, hour measurement for the requirement specification process and the number of SRS hand-ins should be recorded in the future so that improvement can be tracked.

Another goal is increasing the final grade of the groups to 4.3 from current 3.9 average. This increase will be due to a higher level of understanding of the software engineering process.

2 Baseline process

This section covers an description of the baseline process. Included are list of elements (Section 2.1), descriptive mode (Section 2.2), and performance (Section 2.3) of the baseline process.

2.1 Elements of the baseline process

This following sections list the elements of the baseline process and their relation. The elements and sections are respectively: Roles (2.1.1), Methods and techniques (2.1.2), artefacts (2.1.3) and activities (2.1.4).

2.1.1 Roles

The team (ETSA01, Group 8) defines four roles in their project plan. The roles are listed in table 1.

Role	**Description**
Software engineer (SE)	Responsible for analyzing, designing and implementing the software. (All members of the team are included in this role)
Customer	The software stakeholder (the teacher).
Project leader	Responsible of the creation the project plan and monitors the project progress. (One student elected by his group).
Software tester	The team members responsible of integration and system testing. Unit tests are performed by software engineers.

Table 1: Baseline process roles

2.1.2 Methods

In table 2 the baseline process methods (and techniques) are listed. The methods relate to the Waterfall activities *Requirement specification*, *design*, *implementation* and *testing* (unit- and system testing). Other methods relate to project planning and communication with the customer (*evaluation* and *information exchange*).

Method	**Description**
Project planning	A time table with milestones is used with a simple Gantt diagram for the major activities. Project manager is responsible of monitoring the progress
Evaluation	An evaluation meeting between students and teacher is held regularly, where feedback is provided.
Information exchange	Information between students and teacher is exchanged via Wiki web. All document deliverables (artefacts) are available on the Wiki. Information exchange between team members is informal, using face-to-face communication or email.
Requirement specification (analysis)	Requirements are listed and use cases created. The whole team is responsible.
Design	High level design is made from the requirement work. Class diagram is used. The whole team is responsible. The test planning is also included in the design phase.
Implementation	The high level design and the use cases are used for implementation of the system. Java is used for programming. Manual writing is included in the implementation phase.
Unit Testing	Developers are responsible for unit testing their own units.
Integration Testing	Black-box integration testing is executed when individual software modules have been integrated.
System Testing	Black-box functional system testing is made at the end of development. Features are tested for conformance to the requirement specification.

Table 2: Baseline process methods

2.1.3 Artefacts

The artefacts currently created by the team's process are listed in table 3. These same artifacts are to be produced with the new process. The artefacts are the output of activities described in the following section (2.1.4). Furthermore the artefacts are used as input information to other activities.

Artefact	**Description**
Requirement specification	A document containing the requirements for the system.
Requirement review	A meeting where all the requirements will be examined and assessed by a teacher.
Project plan	A document containing details of how the project will be executed.
Project plan review	A document with results from the project plan review meeting.
User manual for interface	A description of how to use the interface designed for the operator.
Test plan	A document containing description of design and execution of test cases.
Test plan review	A document containing results from the test plan review.
Design document	A document containing the design of the software.
Design review	A document containing results from the design review meeting.
Manual for bicycle owner	A description for end users who use the garage part of the system.
Test report	A document containing test results.
Program source code	The source code of the software.
Executable version of the software	The compiled code of the software.

Table 3: Baseline process artefacts

2.1.4 Activities

Table 4 lists the activities of the baseline process. Section 2.2 describes how the baseline activities map to artefacts, methods and roles previously described.

Activity	**Description**
Requirement specification	Collection and analysis of requirements for the software.
Create uses cases	Creating uses cases from requirement specification.
Writing the project plan	Project manager is responsible of the PP creation. All members contribute to the creation.

Project plan review meeting	A team meeting where the project plan is discussed
Project plan rework	Updates on the project plan after project plan review meeting.
Creating the design	Software design activities.
Design review meeting	A meeting where the design is discussed. As many meetings as needed.
Design rework	Any design rework needed after results of the design review meeting.
Writing test plan and test cases	Writing the test cases that covers all the requirements.
Testing review meeting	A meeting where the test plan and test cases are discussed.
Implementation	Programming activities (Coding)
Unit Testing	Each programmer is responsible of testing its own code.Test reports are not necessarily used.
Compiling the program	Compiling, building and packaging the software.
System Testing	System testing activities.
Writing system test report	Test report is written after system testing.
Writing operation user manual	Writing documentation targeting system administrators
Writing user guide	End user documentation writing, targeted at bicycle owners

Table 4: Baseline process activities

2.2 Descriptive Model of the Baseline Process

Table 5 shows a descriptive model of the baseline process. Each line in the table maps the relation between activities, artefacts, roles and methods of the baseline process . The structure of the table is based on the list of activities in table 4.

Activity	**Input Artefact**	**Output Artefact**	**Roles**	**Methods**
Gather requirements		Requirement list	Team, Customer	Requirements spec.
Create use cases	Requirement list	Requirement Specification	Software Engineering (SE) Team	Requirements spec.
Create Project Plan	Requirement Specific.	Project Plan	Project Manager, SE Team	Project planning
Project Plan Review	Project Plan v0.99	Project Plan Comments	Project Manager, SE Team	Project planning
Project Plan Rework	PP v.99, PP comments	Project Plan v1.0	Project Manager	Project planning
Create Design	Requirement Specification	Design Specification	SE Team	Designing
Design Review	Design v0.99	Design Comments	SE Team	Designing
Design Rework	Design v.99, Design comments	Design Specification v1.0	SE Team	Designing
Create Test Plan	Req. Spec, Design Spec.	Test Plan	SE Team (Software Testers)	System testing
Test Plan Review	Design v0.99	Design Comments	SE Team (Software Testers)	System testing
Test Plan Rework	TP v.99, TP comments	Test Plan v1.0	SE Team (Software Testers)	System testing
Programming	Req. Spec, Design Spec.	Code	SE Team	Implementation and unit testing
Compile/Build	Code	Executable program	SE	Implementation
Software Testing	Executable code, Code, Design Spec., Req. Spec.	Test Report	SE Team (Software Testers)	System testing
Create Operating Manual	Executable program, Req. (use cases), design	Operating Manual	SE	Implementation
Create User Guide	Executable program, Req. (use cases), design	User Guide	SE	Implementation

Table 5: Descriptive model of the baseline process

2.3 Performance of the Baseline Process

2.3.1 Average Project Grades

The average project grade for the students 2010 was **3.9**, where 5.0 is the maximum grade. Project grade data for earlier years is unavailable.

2.3.2 Number of SRS hand-ins

We had a discussion with the current ETSA01 group after they had received comments on their first requirement specification hand-in. Their first version of the requirements specification contained a lot of correction to be made, e.g. the name of the signals and furthermore notes on requirements that the team did not know were necessary to include. The amount of feedback with corrections to be made for the team was large and the specification must be handed in at least once more. Experience from the same class earlier years indicate that the SRS iterations are generally too many. An approximation made by the Professor responsible for the course, ETSA01, is that 100% need a second iteration, while around 15% also need a third. This puts the average number of required hand-ins at 2.15.

3 Target Process Description

We want to change the way the project is done. We have chosen to take the waterfall model currently being used and adjust it in a simple manner by adding a *pre-phase*, a *finalization phase* and one additional iteration. Table 6 shows the proposed phases and their proposed duration in weeks.

Phase	*Description*	*Duration (weeks)*
1	**Prephase**. High-level requirement analysis. High-level design. Rough project plan.	1
2	**Iteration 1**. Detailed requirement work, design, implementation and testing for selected use cases.	3
3	**Iteration 2**. Improvements after feedback from iteration 1. Detailed requirement work, design, implementation and testing for the rest of the use cases.	2
4	**Finalization phase**. System testing, rework (if needed). Final hand-in.	1

Table 6: The proposed model phases

Instead of doing the steps in the waterfall once, we want to do them twice. Since we are going to change the actual model we need to change everything in it to fit into two iterations instead of one. In this case, changing the model will be fairly simple, and not too risky. There is an existing time plan with deadlines for all artefacts. What we need to do is adjusting the deadlines to fit an additional iteration in the seven week timespan of the project. Since the main part of the project should be finished before the second iteration starts, more time will also be allotted to the first iteration.

In the baseline process, one task must be completed before the team can move on to the next task, which might lead to difficulties in defining things such as use cases, non-functional requirements and so on.

There will be a phase before the first iteration, where requirements are gathered and all use cases are defined roughly. In this phase a high level design of the system is also made. This phase should take one week.

In the second phase the team will go through the whole waterfall process by starting with selecting what use cases are of most importance. The team makes a detailed description and a design of these main use cases, followed by implementation and acceptance testing. In the improved process, the team will make a ”rough draft” in the first iteration, listing all the main use cases and requirements. Detailed use cases will be made, and the first iteration of the system will only be designed to cover those use cases. Having all the relevant use cases defined, the team will move on to implementing the first iteration version of the system, during which new or improved requirements can be added to in the second iteration.

When the first iteration implementation is done, the team will have a meeting with the customer, so that the involved parties can discuss the implemented functionality. During this meeting, the customer will provide the team with valuable feedback, ensuring that the team is on the right track. The customer will also have a chance to update the requirements, if necessary. This meeting is likely to be efficient because the customer can see what has been implemented and face-to-face communication with the customer is used while the software and other artefacts are evaluated.

After the meeting, the team will start a new iteration, and will update documents as required. The new requirements acquired during the initial implementation of the system, as well as those added by the customer, will be written into the second version of the requirement specification. If something is still missing, the customer (i.e. the teacher) can still return it to the team, requiring them to improve or redo some parts of it.

The project plan for the second iteration will be composed, and if the team noticed during the first implementation that more or less time is needed for certain tasks they can distribute it differently (with consent from the customer) during the second iteration. By doing another iteration it gives the development team chance of improving their way of work. The manual will be updated to contain the new functionality as specified by the new requirements and (if applicable) use cases. The test document will also have to be updated to contain the newly added requirements.

3.1 Detailed Description of the New Process

Table 7 shows a detailed description of the new process where each of the proposed phase and its main activities are described. The project week numbers are listed for each process step with information on who is responsible and how the steps should be executed.

What	**When**	**Who**	**How**
Prephase, where the team is supposed to do some requirements and begin the high level design.	Week 1	The whole team	The group should prepare the first couple of weeks with a prephase where some requirements are done, and design is started.
First iteration, Begin the first iteration of the software process model.	Week 2 and 3	The whole team	The group should decide what use cases are most important and start the implementation and begin testing.
First iteration: Done, Present 1st iteration work.	Week 4	The whole team and the customer	The group should have a beta version of the application based on the rough draft to present to the customer.
Evaluate feedback, after the meeting some feedback will be given that needs to be evaluated.	Week 5	The whole team	With the feedback from the customer the group moves into the second iteration.
Improve the requirements, improve and rework requirements based on customer's feedback.	Week 5	The whole team	With some feedback in hand the step where the requirements are made will be a lot easier to consider.
Second feedback of the requirements, control that the improved requirements are satisfying.	Week 6	The customer and the team	Control with the customer if the improved version of the requirements plan is satisfying, if not iterate through the requirements plan again.
Project plan of the second iteration, since the group moves into a new iteration they need to update their project plan	Week 6	The project leader (with the team)	Since the group have gone through an iteration already the project plan with the time schedule can me improved to meet the new requirements.
Design and test documents are improved.	Week6	Software Engineers and testers	The second iteration will make it easier to do a precise plan of all the steps including the design and test documents.
Final implementation and testing, with all the documents done for the second iteration the group can do the final implementation	Week 7	Software Engineers and testers	The group should now be in the last phase of the project and the final implementation done with respect to the design and test plan.

Table 7: Detailed description of the new process

4 Target Process Implementation

This section describes the steps needed in order to implement the new process. The implementation involves informing the course representatives of the proposed software process (improvements) and updating teaching material accordingly. Table 8 lists the required steps.

What	**When**	**Who**	**How**
Inform tutors that act as customers	Before course starts	Professor responsible for course	Email a description of the new process.
Update exercise sessions to reflect the changed process	Before course starts	Professor responsible for course	Edit the material used by both students and teachers.
Update course desciption, where the goals of the course are described	Before course starts	Institutional secretary	Open the course description in a text editor and edit all parts relevant to the project.
Update lecture slides, all parts about the project	Before course starts	Professor responsible for course	Open powerpoint and change all slides with information relevant to the project.
Update course website	At least two weeks before course starts	Professor responsible for course	Open HTML editor and change all parts of the websites relevant to the project.
Update course wiki	Before course starts	Professor responsible for course	Log in to the course wiki and edit all pages relevant to how the project is carried out (*i.e.* timeline, etc).
Update course material, specifically the compendium with project information	Before course starts	Institutional secretary	Open LaTeX editor and edit the parts of the compendium relevant to how the project is carried out (*i.e.* timeline, etc).

Table 8: Target Process Implementation Steps

5 Measurement and Control

5.1 Measurement plan

Performance measures [1] will be collected for the two goals defined in Section 1.4. Table 9 shows how we set up the measurement plan for the goals and their corresponding metrics.

Goal ID	**Metric ID**	**Metric Name**	**Data Creation Event**	**Data Col. Time**	**Data Col. Resource**	**Data Provider**	**Data Collector**	**Form ID**
G1	M1	Number of SRS hand-ins	SRS hand-in	End of last hand-in.	Excel (Teacher)	SE Team	The customer (teacher)	SRS-X
G2	M2	Average Project Grade	Project delivered	End of course. Final delivery of project.	Ladok	SE Team	The customer (teacher)	Student grades

Table 9: Measurement Plan

The *Number of SRS hand-ins* (M1) will be measured in a number of times the SRS is hand in. The teacher should collect this data in an Excel sheet. The *average project grade* (M2) needs to be calculated. The grades are already measured and registered to the Ladok system. The statistics of the course are presented at the course evaluation meeting after the course has finished. The measurement is taken and the quality assured by the relevant institution. To evaluate if the changes of the process have had any effect they should be compared with statistics from previous years.

After each year the grades need to be compared to the number of SRS hand-ins to see if there is a relationship between the two metrics.

5.2 Action Plan

Based on the statistics and information from the responsible professor for the course we will come to a conclusion whether or not our SPI suggestions have made improvements. If the goals are reached after implementing the revamped process, it will be deemed a success. A small increase in performance that still doesn't make the goals will not be deemed a failure, but will still require additional improvement work. If no improvement is seen, or if results get worse than initially, it will be deemed a failure and appropriate action will have to be taken to get a better result than the baseline process has. Then a step needs to be taken back and improvement work started on the baseline process again with the new information provided. If the results however show that performance increased a bit, i.e. the new process performs better than the baseline process but still did not make the goals, the new process will likely be established and a new round of SPI will be made on the new process. Continuous improvements work is the key to success.

If the changed process is deemed a failure, the SPI needs to be refined and further improved.

6 Discussion

6.1 Underlying Rationale of Proposed Changes

The focus of our improvement is the requirement specification. Coding is usually not a problem for the students and they generally have the required programming skills. When we did this project two years ago we did not quite understand how the software development process works, nor did we have a clear idea or knowledge of what should be in the requirement specification. Our rationale for changing the model of the software development process is mainly to increase the learning experience for the young engineers being educated. We think that by increasing the number of iterations, the student will be able to make qualified decisions based on newly aquired experience rather than just ”guessing”. When writing the requirement plan the student has little or no understanding of the importance and use of making requirement specification, nor any experience in making it.

The reasons for implementing the modified process model are:

- Better clarity in what is required in a SRS.
- Early feedback will be provided on the SRS
- Ability to start the developing earlier while getting continuous feedback
- Better learning and understanding provided via continuous feedback.
- Reducing risks of a failure.

We believe the students will get better feeling for the SRS work if they can go through a ”full circle” of requirements design, implementation and testing without it having to be the final hand-in.

The ultimate goal is to increase students' understanding of the whole software development process which will result in better analysis and requirement specification and therefore minimum number of SRS hand-ins (Goal 1) and rework. Furthermore with better understanding, test cases will be of higher quality resulting in more quality of the product leading to a higher grade (Goal 2). We conclude that an increase of the average grade from 3.9 to 4.2 would be an acceptable improvement goal.

6.2 Risks of Proposed Changes

The risks of our proposed changes are not very high, since we do not make any drastic changes to the baseline process. We design the SPI in a way that it is more likely to reduce risks than adding ones. The development team needs to understand that careful time planning is necessary in order to deliver the finished project on time. The work load will not be the same during the whole period, instead some iterations will take more time then others. The risks can be reduced by introducing them to the original waterfall model first and then add the iteration step. Another risk is that the students may do "too much" the first iteration resulting in a too high workload and a stressed team. This can be reduced by adding an extra lecture about iteration planning where a rough outline is given of what should be contained in each iteration. Also feedback on the project plan will be provided by the tutor before the first iteration starts.

References

[1] Bob Hughes, Mike Cotterell, *Software Project Management, fifth ed.* McGraw-Hill Education (UK) Limited, 2009.

[2] PROFES, *User Manual.* PROFES, 1999.