Approaching CMM to an Educational CDIO based Software Engineering Process

PhD Daniel Einarson

Computer Science, Kristianstad University, Sweden

ABSTRACT

Software Engineering is probably one of the most recent engineering disciplines. As such there have been major requirements put on that discipline to undergo appropriate transformations from immaturity to higher levels of maturity. This does not only relate to matters of evolution of techniques, and usability, but also the process of developing the software of concern.

The ability to succeed in a large scaled Software Engineering project is probably highly dependent on the ways of working, that is, on the working process that fulfils the project. The Capability Maturity Model aims to assess the maturity of a software process, and serves as a guidance to improve such processes. Maturity here relates to degrees of formality and optimization, from ad hoc driven processes, via defined process structures, to active process optimization.

Seen in a perspective of Software Engineering education we can see that it is of high importance that students get exhaustive practice in project based work. Several working process models of that are proposed in the context of Software Engineering research. Following such models may increase the probability of project success, both in industry and in educational projects. Still, a process with inherent self-reflecting mechanisms may increase such probability even further, with effect on quality of both work and product.

This contribution will shed light on the working process of students, where self-reflection, self-motivation, and process improvement are put in focus. A model for this has been the Capability Maturity Model where reflecting over working processes aims to increase grades of process maturity and quality.

KEYWORDS

Project based learning, Education concepts, Software Engineering, Cooperative work, Process improvement.

INTRODUCTION

At the CDIO 2011 conference, the author of this contribution presented process models of Software Engineering, and showed how those may be applied in Software Engineering education ([1]). Furthermore [1] compared such models with the approach of CDIO. A result of that contribution was an illumination on equalities between Software Engineering process models and CDIO.

Furthermore, [1] provided a case study where work of a project based course was based on a Software Engineering process model. Pointed out problems from [1], included:

- Hard to motivate students to follow the process model, which was needed to make the project groups work as a whole

- Hard to evaluate students on individual levels

A conclusion from the course lecturer (that is, the author) was there is a need to provide ways for students to reflect even more on the process. Reasons for that should be to increase the grade of understanding of the ways of working in itself, and therefore also increase motivations for following the proposed process model. By clarifying that and confront students about that, the extent of individual contributions, or lack of contributions was also made clearer.

The course lecturer has had a chance to test this at the same kind of course, autumn 2011, for three project groups with group sizes of about 20 students. The main project intention is to study how to control interactive house equipments through smart phones/tablets, web-based interfaces, and voice detection, all in all pointing out technology of tomorrow. As pointed out by Nicholas Negroponte:"Computing is not about computers anymore. It is about living" ([5]).

New approaches have been based on meetings with whole project groups every two weeks, where it should be clear what each student should have done. Discussions were performed in full project groups, where individual students had to answer for their individual piece of work. That work was also communicated with other group members, especially those directly dependent on that work.

As a part of the project course the students started with an outlined way of working, but should elaborate on this, and negotiate on alternatives for reasons to improve the working process. Typically, a set of documents representing project artefacts, were used to represent and control work flow.

To be able to communicate work, even in times of being geographically spread, group members had to negotiate over communication tools. All three groups chose facebook, and DropBox as such tools. Facebook was used as an information blackboard/whiteboard, while DropBox was used as a placeholder for all the produced artefacts. Interestingly enough neither facebook nor DropBox were originally developed for educational purposes. However, those were naturally chosen by students belonging to a generation where IT based social networks and Clouds ([3]) are parts of daily living.

This contribution will focus on students' working process and self reflections to improve the working process; here the Capability Maturity Model, CMM, serves as a model. The concept of CMM will be presented and be put in the context of student projects. Moreover, a brief view will be provided on tools for communication and education.

BACKGROUND

At the home department of the author, we have several years of experiences in project based courses. The working process during those projects is very much inspired by process models from Software Engineering research and industry. In [1] the author outlined a number of Software Process models and close correspondences of those with the CDIO initiative ([11]). The sequel will put this contribution in the context of process models of Software Engineering, and show an example of a course (the course of the case study of this contribution) where work flow is based on such a process model. The CMM will also be further explained.

Concepts of Software Engineering

As was pointed out at [1], the relatively short history of IT can unfortunately point out a number of failing projects concerning missing deadlines, functionalities, low quality, etc.

Theories and techniques have been developed, to meet inherent problems and challenges. But also software process models have been emphasized. Typically those mainly relate to prime phases of work, such as *Requirements*, *Design*, *Implementation and Tests*, and *Operation and Maintenance*.

Software Engineering literature, such as [7], exhaustively covers several aspects of Software Engineering. For instance, several process models are presented, such as the Waterfall model, the Rational Unified Process (RUP) model, and the eXtreme Programming (XP) model. A short outline of those:

- The Waterfall model is an early proposed model which emphasizes on the four prime phases mentioned above. Typically those are strictly followed in time, starting with Requirements and fulfilling those before next step, Design take place.
    - Advantages: clear structures, if all requirements are known from start this may be a good choice, in that case also beneficial for especially large scaled projects with perhaps geographically spread development groups.
    - Disadvantages: requirements often change because of lack of understanding both from customer and developer parts. Furthermore change may occur because of changing contexts, technologies, etc.
- The RUP ([6]) typically meets the problems of changing requirements by having an iterative and incremental way of working, more of this will be mentioned below. Disadvantages relate here to exhaustive documentation as a natural part of that process model.
- The XP ([12]) is an example of a so called agile process, an approach from where criticism has been pointed out towards the more heavy processes, such as, RUP and similar, for being too inefficient with too much work put on management and control (including exhaustive documentation).
    - Advantages: Being fast, smooth, and adjustable to sudden changes.
    - Disadvantages: Not suitable for especially large scaled projects.

More on the RUP

While the Waterfall model is a timely linear model idealizing over the main phases *Requirements*, *Design*, *Implement*, *Operate*, the RUP outlines the process through a two dimensional scheme, as shown in Figure 1. Time here moves from left to right through corresponding main phases called *Inception*, *Elaboration*, *Construction*, and *Transition*. In parallel to those disciplines such as Requirements, Analysis and Design, Implementation, etc are performed with varying effort, as outlined in Figure 1 (interesting here is to see that e.g., also project management is considered as a discipline captured in the work flow).

The RUP is considered to be an iterative and incremental process model, with the following meaning:

- *Iterative*: Work in steps (in Figure 1 those are called E1, E2, C1, C2 etc.). Include the customer in those steps for feedback on what has been done so far, and for negotiations on future work. This makes the process adjustable for changing requirements.
- *Incremental*: Work on parts of the system, i.e., parts of the requirements, typically selected through priority, parts of design, implementation, and so on. The system is in that way built in prototypical steps, executable in themselves, and therefore points for feedback and further discussions.

Besides from the system under construction an exhaustive set of artefacts represented by documents of those are being iteratively and incrementally developed. Those have the intention of both illuminating on the state of the project and documenting that for future activities. Please see for instance [6] for more on this. A small subset of those will however be presented in a later section of this contribution.

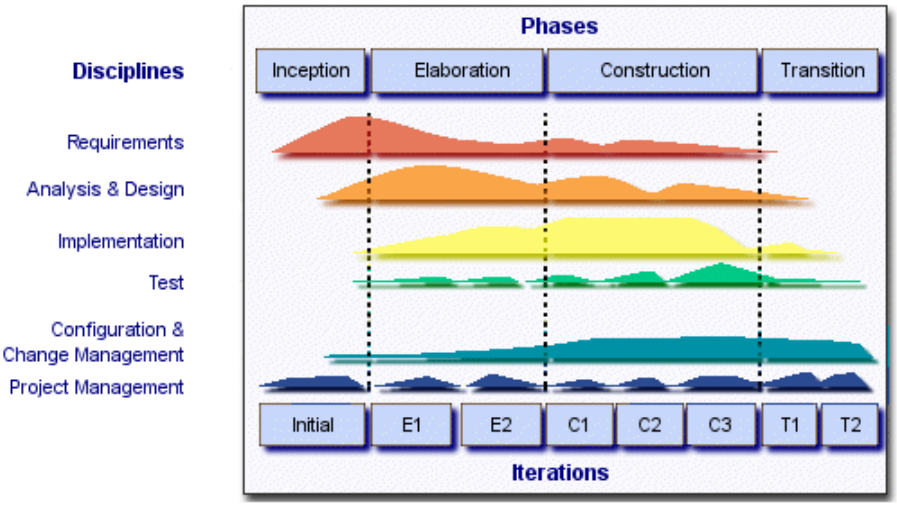

Figure 1. Phases and Disciplines of RUP ([6]).

The Case Study Course

The course of concern is run half time over the fifth semester. It starts early autumn and ends after New Year with final presentations of what has been done. The course has about 60 students divided up into three major equally sized groups. This subsection will cover the core of the project, project organization, and the core of the work flow. More details on this will also be mentioned in later sections.

The Project

The project of the course concerns a so called interactive house. Several devices, physical or simulated, should be controlled from computers or Smart Phones. Figure 2 outlines this. The structure of this system was presented in [1], and will only be briefly covered here to put this contribution in the appropriate context.

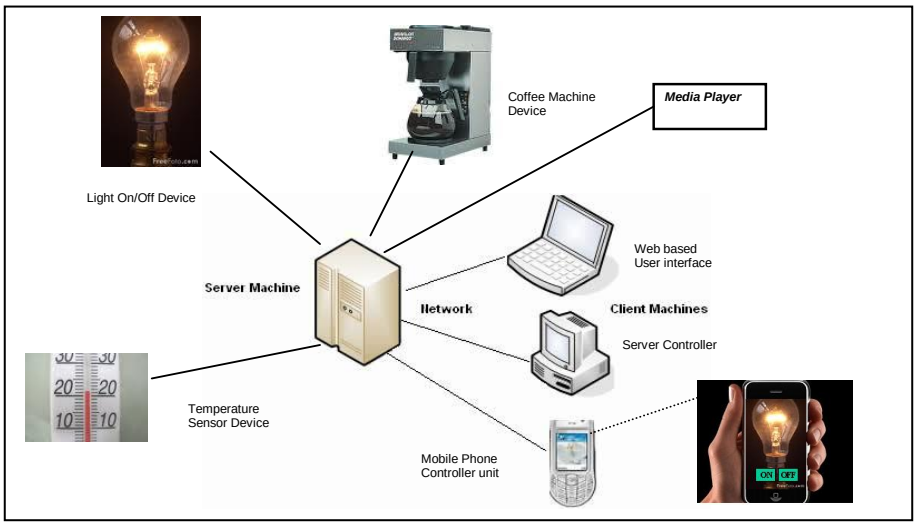

Figure 2. Illumination of system of the Case Study ([8]).

Physical *devices* typically are lamps, fans, and radiators of a small scaled physical house. Software for controlling those has to be developed. Simulated devices may be music media players, a microwave oven, or a coffee machine. Those are typically developed and run at a computer visualizing their behaviour. Software should also be developed for the user *units*, that is, the Smart Phones, tablets (typically android based), or lap tops typically with web

based user interfaces. Furthermore, a server with a database should be put as a mediator between the parts, and finally communication between those parts should be clearly specified.

Project Organization
For one project group with size of about 20 students, there should in first place be one, by the lecturer chosen, Project Manager, PM. The group should decide upon one Requirements Manager, RM, responsible for the main requirements of the project, and also acting as the PM's right hand. The project group should then be divided up into subgroups with sub PMs, with respect to functionality. Figure 3 illuminate this project group structure. Here *devices* relates to equipments of the house, while *units*, relate to user driven equipments (Smart Phones, lap tops). The PM and RM are also part of one of the subgroups. As one of the PMs once expressed it "You are both soldier and commander".

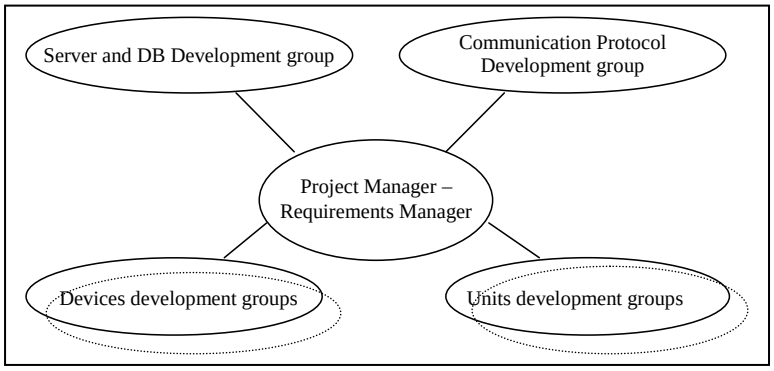

Figure 3. Project group structure.

Work Flow
The former year this course was provided, discussions between course leader and project group were mainly done through PM and RM. A result of this (also presented in [1]) was that it was hard to motivate work, with risks of course failures, there were too many responsibilities put on PM and RM, and it was hard for student of project groups to understand how to work. As a result of that, at the current course there were full project group meetings with the course leader every second week. At those formal meetings everyone should have the possibility to discuss and also were encouraged or forced to do that.

For each formal meeting all subgroups should upload a set of artefacts that should comprise the state of the project/subproject. The artefacts are distributed over the members of the subgroups with responsibility for those to handle those artefacts. The set of artefacts, which is further covered later of this contribution, are also the bases for further reflection on the system and on the process of the project. They are furthermore the prime formal basis for individual grading.

Between the formal meetings the students are also encouraged to have informal meetings in full project groups or subgroups. The intention of those informal meetings should be to check the state of the project, plan future actions, and discuss problems and solutions.

Communication is here especially important, informal as well as more formal. To support communication IT support for that is important. You may use email, but here something more advanced is further required. You need a kind of white board for discussions, and a placeholder for code and documents. As was presented in the Introduction, facebook and DropBox were chosen for those purposes.

The Capability Maturity Model

The Software Engineering Institute (SEI) is a federally funded research and development centre headquarted on the campus of Carnegie Mellon University, Pennsylvania, USA. The SEI operates with major fundings from U.S. Department of Defense, and works closely with industry and academia through research collaborations ([4]). According to [7] the SEI is engaged in a long-term programme of software process improvements. One part of this concerns the Capability Maturity Mode (CMM) for software processes. At a core the CMM is a five level model for reflecting the maturity of software processes. Figure 4 illuminates on this.

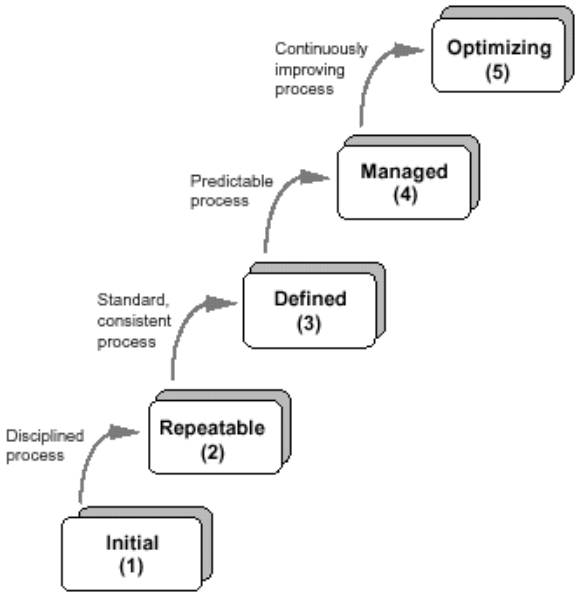

Figure 4. Levels of the CMM ([2]).

Summarized, those steps are (for more on this, see for instance [7]):

- Level 1: Initial or chaotic level. Processes are driven in ad hoc ways, often dependent on personal efforts. If formal methods exist, there are no mechanisms at organizational levels that ensure that those are followed. Software product-, as well as software process qualities will be unpredictable.
- Level 2: Repeatable level. Repeatable here means that an organization may repeat the same kind of project in the same way. This is often dependent on a manager's ability to motivate the project team to work in ways that intuitively may be seen as reasonable. Product qualities and development costs may still be unpredictable.
- Level 3: Defined level. At this level the development process model is integrated within the organization. Formal procedures of that model should be followed for each project. This also means that the defined model serves as a basis for process quality improvements. Product and process qualities may be predicted.
- Level 4: Managed level. Quantitative measurement methods are introduced to assess product and process qualities. Formal procedures for data collection on that are used and fed into process improvement activities.
- Level 5: Optimizing level. Continuous process improvements are here integrated within the process of the organization, and planned and budgeted for. The process may be controlled and adjusted to suit the specific project, with respect to development cost and product qualities.

The CMM may have proved itself to be a useful model for reflecting over process maturity, but has nowadays actually been superseded by the CMMI (I stands for Integration, for more on this, see for instance [10]). The CMM is still used as a theoretical process capability model, and has as such been the prime model of choice for this contribution and the case study of this contribution.

SUGGESTED ARTEFACTS AND WORKFLOW

According to [6], the UPEDU (Unified Process for EDUcation) has been customized from the RUP - an industry-wide process platform - for the educational environment, please see Subsection *Concepts of Software Engineering*, for more on RUP. The process is iterative and incremental and highly driven by the contents of a set of artefacts. Those artefacts are in themselves developed iteratively and incrementally. That is, they are improved during the process, for instance, based upon increased knowledge about the project requirements. Typical points of artefact reflections are those illustrated as the iterative steps by Figure 1.

On Artefacts

By artefacts we mean *documents* (such as a requirement document), *design models*, or *design elements* ([6]). We will here especially regard *documents*, while *design models* or *elements* may be considered as specific parts of design documents.

The UPEDU provides a quite large set of documents, several regarded at each of the disciplines outlined at Figure 1. So for instance, for the Requirement discipline there are documents representing the project's *Vision*, *Glossary*, *Software Requirements Specification*, and *Supplementary Specification*. The Project Management discipline involves the *Software Development Plan*, *Measurement Plan*, *Iteration Plan*, *Risk List*, *Work Order*, *Review Record*, and *Project Measurements*. It is by far no means of this contribution to further explain the more precise meaning behind those documents (please see more at [6] on this), but rather to emphasize on the size of those artefact sets, and more to elaborate on the artefacts used in the case study of this contribution.

The UPEDU suggests furthermore a set of working roles, such as the *Analyst*, or the *Project Manager*, associated with specific disciplines, and therefore also with specific artefacts. So for instance the Analyst typically handles Requirement artefacts, and the Project Manager handles Project Management artefacts.

The case study is clearly inspired from the UPEDU, still without the comprehensive set of artefacts, and without the clear roles of work. For instance, even though we may have the PM and the RM, it is not clear that the PM should handle the Risk Lists, or the RM should handle the Supplementary Specification.

Case Study specific Artefacts

The Case Study Course requires the following set of documents:

- **Vision document**, this explains the main project vision. Typically this relates to an expression of the stakeholder's desires on the system. This also serves as a basis for the Requirement- and Supplementary requirement document.
- **Project risks**, project risks may be categorized as follows ([7]):
    - *Project risks* affect schedule or resources
    - *Product risks* affect the quality or performance of the software being developed
    - *Business risks* affect the organisation developing the software.

- **Requirement document**, system functional requirements. This relates to what the system should do, that is, the functionality it should deliver.
- **Supplementary requirements**, non-functional requirements, or quality attributes. This is typically tree-folded ([7]), relating to:
    - *Product requirements*, Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
    - *Organisational requirements*, Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
    - *External requirements*, Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.
- **Design document**, Software Architecture and Design, may use so called UML diagrams for both analysis and design. This is the main document for guiding further implementation work.
- **Verification, Validation, and Test**, document for different kinds of tests, at different levels, from small scaled unit tests, to integration tests on system as a whole, and user acceptance tests.

Every subgroup of a project group should handle those documents. With six subgroups (see Figure 3 for project group structure), there will then be six such sets of documents. There are normally no specific roles within a subgroup; the documents are rather distributed amongst the members in an arbitrary way. In reality it turned out that once a project member got the responsibility of one specific artefact, she/he continued to have that responsibility also later in the process.

Besides from this the PM and RM should together handle a Vision document, Requirements document, and Supplementary requirements document, representing the project, and the project group as a whole. Those documents should represent the state of the project as well as the work of the project.

Artefacts and Work Flow

Documents should be developed and handed to the course lecturer in time for the formal meetings every twice week. There is then one such meeting per full project group. At such a meeting the artefacts are discussed, and each specific artefact should be defended by its responsible person.

Artefact should not be developed in last minute before delivering time, but rather continuously during one iteration of the process. For instance, it is suggested that work on Risk artefacts, or rather, doing risk management, should be done as follows:

- Risk identification
    - Identify project, product, and business risks
- Risk analysis
    - Assess the likelihood and consequences of these risks
- Risk planning
    - Draw up plans to avoid or minimise the effects of the risk
- Risk monitoring
    - Monitor the risks throughout the project

Similarly, it is suggested that corresponding requirements management should be performed as follows:

- Requirement identification
    - Identify a requirement, typically a customer desire on the system

- Requirement analysis
    - Analyse consequences of the requirement
- Requirement prioritizing
    - Assess the value of the requirement to the project
- Requirement evaluation
    - Evaluate design or even implementation of a requirement, or refine the requirement as a consequence of more understanding

Now, if artefacts are being developed this way, they can also perform a basis for future project plans. High risk elements should be handled as soon as possible, as well as high priority requirements.

Examples:

- A risk may be lack of knowledge in how to communicate with a server from a mobile phone, and this will then make the system pointless. If this is considered a high risk, then plans should be made for solving that, including finding out about the solution and test that out, as soon as possible.
- A requirement may be: 'put on a lamp'. This is probably essential to the whole system, and should therefore be regarded as a high priority task, and therefore be designed for, and implemented as soon as possible. Lessons learned from that task may then also be used to check the state of the lamp, or control the heat of the house.

Again, if done appropriately, this means that those documents serve as a basis for planning the process. As, for instance, the risks and requirements change, as they do when taken care of, then also the corresponding artefacts change. They therefore become 'living' documents developed further during the process of the project.

PROCESS REFLECTION AND IMPROVEMENTS

As previously has been mentioned, a result of the case study of [1] was that the course lecturer had to take more action with full project group meetings, where project group member should answer for their specific effort directly to the course lecturer. However, this was not enough. For further engagement on the process, further self reflection was needed.

In the sequel of this section this is elaborated on further and also serves as the core of this contribution. We will here put the process in the context of the CMM concept. It should however be clearly pointed out that we do not have any ambition to compare this educational process with industrial similarities, but rather use the CMM as a conceptual model to reflect on maturity of the educational process solely.

Examples will be provided below that should be representative for the course. Still, those are modified to keep the examples anonymous. Furthermore, those are of course very simplified excerpts from rather complex educational situations.

Observed Problems

The documents outlined in previous section may be quite meaningless in themselves if not appropriately put in the context of the process. Put this in the context of the CMM, we start at a level 2 process (please see subsection on *The Capability Maturity Model* for more on this), where it is up to a project manager to inspire project members to follow the process appropriately.

Observed problems do here include low quality of artefacts, low grade of responsibility of those, hard to motivate one artefact's contents with respect to other artefacts. At the formal

meetings it was hard for students to answer for their work. There was low connection between members of the same project subgroup.

Below, a number of examples illuminate on those observations. The examples are focused around the lecturer's (=the author) questions to students, based on delivered artefacts.

Example 1
Why is the style of your document different from the other of your subgroup? Do you not have any agreements on how you shall work, and the outlook of the outcome of your work? How is your internal communication actually? This is a risk document, what about the risks of your group? What are the states of the encountered risks? How can you modify your risk document so you make sure you handle the risks of your internal collaboration?

It turned out that the subgroup was quite risky, and there was definitely lack of internal communication. Moreover, risk documents were generally, for many groups, treated without any reflections on their actual state. They often also did not have an actual action plan to handle them (please see Subsection *Artefacts and Work Flow* for more on the risk artefact and corresponding work flow).

Example 2
In your requirement list most of your requirements are not elaborated on well enough. How can you design and implement for a requirement with such a lack of information? It is a high priority requirement so even if we work iteratively and incrementally, you should have come further by now.

This quite typical problem was often due to lack of understanding on how to work, how to use covered techniques for analysis and design.

Example 3
How does this requirement correspond to any of the design items, or test items? How can you see what design item follows from what requirement? How can we see cooperation and coordination of work in your subgroup? How far have you come in developing and implementing this requirement?

This is a typical example of lack of reflection on work. There were no ways of seeing how far they had come with specific tasks, or how to track tasks of the process.

Example 4
This Supplementary requirements list is written in an inappropriate way. You mix product requirements with process requirements. How can you change the artefact to categorize the supplementary requirements clearer?

This is also a typical problem that actually shows a lack of reflection on work, especially and obviously illustrated by the lack of organizational or process requirements. Furthermore, if a high priority demand on the quality of the product is hard to implement and check, then that should probably be regarded as a high priority risk in the risk list, which in turn should be handled as soon as possible.

Solution Approach

An approach to solve some of those typical observed patterns of problems was to find ways for students to handle those within their process models in themselves. Again putting it into the context of the CMM, how can we climb a step towards the third level, the *Defined level*, where the development process is integrated at the level of the project group, including

levels of subgroups? How can this then be done in ways that also catches and handles the observed patterns of problems?

Example
Is the lack of explicit knowledge on connections between requirements, design items, and test items, and development state of those, something that should be regarded at a general level for the whole project? Can PM and RM say something about this? Can this in that case be pointed out in the project overall Supplementary requirement document, part Organizational requirements? What about other things? What can be made clearer regarding ways of working here? How shall this be communicated to all parts of the project group?

Discussions like this (there were several of them) contributed with putting ways of working at more formal levels. It was then up to the students, with most responsibility put on PM, RM, and subgroups' sub PMs to specify how to continue.

Implementation

As problems were pointed out and discussed, the students (especially through initiative of the PMs and RMs) changed the frameworks for their process. This was actually an ongoing activity, performed on the basis of the iterative formal meetings with the lecturer. Moving towards the level 3 of the CMM was, so to say, done iteratively and incrementally as well as the working process in itself.

Following changes to the process framework were observed:

- Templates for the artefacts were changed and customized for each project group.
    - They were here developed with a style that should be project group specific, with intensions of group identity and consistency.
    - Placeholders were added to identify dependences between artefact items (such as one specific requirement).
    - Placeholders were added to identify the state of one specific artefact item, to show if it was still pending, fulfilled, or developed until a certain level, for instance a value representing the percentage of fulfilment.
- The Vision artefact document was used not only to draw the main picture of the product, but also to explicitly express how the group members should work
    - It was clearly pointed out when they should have their group specific informal meetings
    - It was clearly pointed out how they should communicate in between, what tools to use, and need for responsiveness
    - Things, such as being clear about using the Risk artefact as a living document, with continuous monitoring and follow-up, were explicitly expressed
    - Carefully regarding the requirement categories of the Supplementary artefact was also pointed out

Effects could also be observed at the levels of the subgroups. Common artefact template styles were followed, ways of handling artefact items, with respect to, interdependencies, and state, were adopted. The Supplementary artefact was categorized with the effect that product requirements and process requirements, respectively, were more clearly illuminated on, etc. All in all, generally there was a significant improvement on artefact developments.

PROJECT RESULTS

We will here consider the outcome of the project. Furthermore, assessing and grading students will be covered.

Outcome

Compared to the corresponding course autumn 2010 ([1]), there were dramatic changes. It was generally a much higher confidence in fulfilling the project, both from the students' point of view, and the lecture's. According the autumn semester 2011, there were also significant changes during the course period, with many examples on a higher observed maturity in approaching their artefacts. That higher maturity in itself mirrors the maturity concerning their view upon their work as well as on the product they were supposed to develop,

Still, not all students, or all project subgroups managed to make this 'maturity climb'. In those cases it was also observed a low confidence on fulfilment of the product, both from the lecture's side and from the PM, and RM of the corresponding project group. It was also shown that those students or subgroups were 'risky' and led to frustrations and irritations within their project groups, with some significant (but interesting) conflicts at the formal project meetings. Also, there were three subgroups that had to continue after the course ended to improve their work. At today's date almost all of them have now done that and also passed the course.

At the great day of the project presentation all three groups should be ready to present. Group 1 did a superb job. Group 2 was the group that was most affected by low artefact qualities. The PM had to say that they were supposed to fail in presenting the product as a whole, since not all subgroups had fulfilled their tasks. Some subgroups were however ready to present their work individually. Group 3 had claimed before that they were ready to present a month ago, still at the presentation day some technicalities did not go their way. All in all, presentations of work was quite good, however, product show was not. It was decided that all groups should have one more day to prepare, and then do a new product show. At that next presentation all groups, even Group 2, managed to show their systems running! Perhaps that was a kind of surprise not only for the lecturer. Generally, the lecturer (that is, the author of this contribution) was very satisfied and happy about the result!

Assessing and Grading

Assessing the students' work was done mostly from two points of views:

- The process: This means the work that they had done and shown at the formal meetings with the lecturer. Assessing this work is chiefly based on delivered artefacts and discussions in the meetings.
- The project result: This corresponds to the result of the product, that is, the parts of the interactive house project that was developed.

Assessing the process is done on an individual basis, since artefacts mainly are developed per student, and the discussions then are focused on one student's specific work (there were however cases where this was not clear because of obvious cooperation). Maturity and quality is here based on the previous descriptions on approaching the artefacts.

Assessing the project is on one hand done on the basis of a subgroup's work, and presentations. Still, written software code should also be notified with the author. Quality of software and amount of work is therefore possible to detect on an individual basis.

The grade of the course is calculated as a balance between the process and the project result. Those have about equal weight, with a smaller favour in project result. Still, at future implementations of the course there will probably be a change in that relation in favour of the process. This is to emphasize even more on climbing the maturity steps.

TOOLS FOR COMMUNICATION AND LEARNING

For large scaled projects to be successful collaboration and coordination of activities are highly required. As previously has been outlined, all project groups of the case study, were free to solve this, and all of the groups primarily chose two tools, facebook as a common whiteboard, and Dropbox as a placeholder and distributor of artefacts. Here for instance, facebook has been used for spreading information on technical stuff, negotiations on times for personal group meetings, recommendations on how to use, and not use PowerPoint for presentations, Questions_and_Answers, and so on. Still, other tools were also used by some groups, such as, Skype, common mail tools, etc.

On top list of learning tools

Neither facebook nor Dropbox have been developed for educational reasons. Nevertheless, both of them have been chosen for process- and educational purposes. Furthermore, interestingly enough both of those have been elected by Centre for Learning & Performance Technologies (C4LPT, *"A resource site for the use of new technologies for working and learning",* [9]) as being on a top 100 list of tools for learning, 2011. Dropbox was ranked number six, while facebook got position 14. The top three were, 1) Twitter - micro-sharing site, 2) YouTube- video-sharing tool, and 3) Google Docs – collaboration suite (for more on this, please see [9]).

From the list we can also see several interesting examples of famous IT tools, such as, Skype - instant messaging/VoIP tool (number 4 of the list), Wikipedia - collaborative encyclopaedia (11), Google Search - search engine (15), PowerPoint - presentation software (19), LinkedIn - prof social network (21). An interesting reflection here is that several of those IT tools do not originally address education, but have rather been adopted by education because of suitable characteristics of those. It is, however, out the scope of this contribution to further conceptualize over eLearning or tools for educational purposes.

Learning tools of home department

Within the context of learning tools it may also be interesting to point out the main educational tool, Moodle (Modular Object-Oriented Dynamic Learning Environment), of our home department. In the context of the case study course, the Moodle-based course system is used for information sharing and also provides an area for document uploads. Especially interesting is here to see the possibilities for students to upload their artefacts. Typically the workflow here is that all the subgroups of one project group collect their artefacts in folders that are put in the project group's Dropbox. Thereafter the PM or RM zips all the folders down to one file, that is then uploaded at Moodle. That is where the course teacher will get the material for further investigations, discussions, and evaluations. (Moodle positioned itself number 8 of the top list of [9].)

SUMMARY

We have provided a case study on a CDIO based Software Engineering project course. The starting point has been the working process of that course where that has been guided through an iterative and incremental Software Engineering process model. It has been observed that the working process in itself has been driven immaturely without any reflections on what was done or how it was done. This has also shown to be risky for the fulfilment of the project product.

To meet such problems, ways have been introduced where approaching the working process has been done through putting more responsibility on the students to contribute to the model for the working process. A source of inspiration has here been the Capability Maturity Model

where climbing the maturity steps of that, means higher integration of process model at the development organization. It has been shown that there is a correlation between taking such steps and students' reflection on their working process, as well as on their ability to fulfil the project.

ACKNOWLEDGEMENTS

Acknowledgements to colleagues, and students, at my home department for participating and contributing to project based courses. Special thanks to colleagues and students of the project based course of this contribution's case study. Furthermore, great thanks to the anonymous reviewers of the 8th International CDIO Conference 2012, for valuable comments on the abstract, and the peer review versions of this contribution. Also great thanks to Douglas Bush for valuable proposals on text improvements.

REFERENCES

[1] Einarson D., Working- vs. Educational Processes in Software Engineering vs. CDIO, 7th International CDIO Conference 2011.

[2] On Capability Maturity Model. http://ibiblio.org/gferg/ldp/SCM-OpenSource/scm-traditional.html

[3] On Cloud Computing, http://en.wikipedia.org/wiki/Cloud_Computing

[4] On The Software Engineering Institute (SEI), http://en.wikipedia.org/wiki/Software_Engineering_Institute

[5] Quote by Nicholas Negroponte, http://nsb.com/speakers/view/nicholas-negroponte

[6] Rational Unified Process in Education, UPEDU, http://www.upedu.org/

[7] Sommerville I., Software Engineering, Addison Wesley, 2010.

[8] System structure, free pictures and photos from the internet.

[9] Top Tools, C4LPT, http://c4lpt.co.uk/top-100-tools-for-learning-2011/

[10] Wikipedia, on Capability Maturity Model, http://en.wikipedia.org/wiki/Capability_Maturity_Model

[11] Worldwide CDIO Initiative: A Framework for the Education of Engineers, http://www.cdio.org/

[12] XP, on eXtreme Programming, http://xprogramming.com/index.php

Biographical Information

Daniel Einarson has a PhD in Computer Science and has several years of experience in teaching Computer Science and Software Engineering. Furthermore, the author has been experimenting with several different forms of software process models as models for educational forms. Moreover, with inspiration from the CDIO initiative the author will strive after developing the educational forms for Software Engineering even further.

Corresponding author

Dr. Daniel Einarson
Kristianstad University
Norra Stationsgatan 8A
281 48 Hässleholm, Sweden
+46 -44 203177
daniel.einarson@hkr.se