Project Management and Software Development Processes for Computer Science Capstone Projects

Dean Knudson and Alex Radermacher
North Dakota State University

NDSU has conducted over 50 very successful Computer Science capstone team projects in the past four years, for companies including Microsoft, IBM, Rockwell Collins, Thomson Reuters, 3M, ATK, Polaris, West Corp. and John Deere. This paper will describe our capstone program and briefly relate the evolution of our software development and project management processes. In particular we have implemented processes that follow standard project management phases (initiation, planning, execution, control and closure) as well as built software development processes that follow the SEI's CMMI model. These processes are applicable to any small team, short-duration project environment but are specifically tailored to computer science capstone projects.

Corresponding Author: Dean Knudson, dean.knudson@ndsu.edu

Introduction

The NDSU B.S. program has required undergraduates to complete a capstone course since 2004. Our capstone course is centered on team based software development projects conducted for local or regional businesses.

Sponsoring Companies	
3M	NISC
ATK	Polaris
Honeywell	Rockwell Collins
IBM	Sundog
John Deere	Thomson Reuters
Microsoft	West Corp.

Table 1: Recent Sponsoring Companies

Our students work in teams of three or four for an entire semester. We arrange for development projects centered on applications of immediate practical value to the sponsoring companies so that student learning experiences are similar to activities that they will be responsible for after they graduate. These projects cover the entire software development life-cycle from requirements definition through final delivery to the customer. This requires students to directly confront the need to provide value to their customer and gives students a broad view of how development projects begin and fit into the spectrum of activities and concerns with which a company must deal.

The goals of the capstone course are to provide students with a concrete learning experience in which they develop real-world software for regional companies while being exposed to development and management processes they will encounter in their professional careers. We believe that our methods of applying a rigorous process to a capstone course improve the quality of education students can expect to receive from this type of course[3,5,6].

SEI CMMI Level 2 Model

At NDSU we have been developing a set of processes that follow the Software Development Capability Maturity Model Integration maturity level 2 model promulgated by the Carnegie Mellon Software Engineering Institute[1]. We also have incorporated project management concepts for project initiation, planning, execution, control, and closure as described in the Project Management Body of Knowledge PMBOK Guide[2].

Capability Maturity Model Integration (CMMI) is a process improvement approach that provides organizations with the essential elements of effective processes that ultimately improve their performance. CMMI can be used to guide process improvement across a project, a division, or an entire organization[1]. This model has been used extensively in industry and has been shown repeatedly to provide value to companies that have implemented processes that follow the model[2].

There are five maturity levels defined with level 1 being the lowest and default level and level 5 being the highest. We will be implementing processes to satisfy the first improvement level - maturity level 2 (Managed). At this level many of the most fundamentally important processes (Requirements management, configuration management, etc.) have been defined. Table 2 provides a brief description of the seven different process areas of CMMI Maturity Level 2 model.

One thing to keep in mind is that the CMMI is a model that describes what should be done (i.e. the characteristics of effective processes.) but does not describe how it should be done. We have developed processes along with templates and examples which define how these processes shall be followed.

Process Areas	Purpose
Requirement Management (REQM)	To manage requirements of the project's products and product components, and to identify inconsistencies between those requirements and the project's plans and work products
Project Planning (PP)	To establish and maintain plans that define project activities
Project Monitoring and Control (PMC)	To provide an understanding of project's progress so that appropriate corrective actions can be taken
Supplier Agreement Management (SAM)	To manage the acquisition of products from suppliers
Measurement and Analysis (MA)	To develop and sustain a measurement capability that is used to support management information needs
Process and Product Quality Assurance (PPQA)	To provide staff and management with objective insights into processes and associated work products
Configuration Management (CM)	To establish and maintain the integrity of work products using configuration identification, control, accounting, and audits.

Table 2 – CMMI Maturity Level 2 Process Areas[1]

Capstone Course Structure

The basic structure of our capstone program is as follows. The computer science capstone course is offered every spring semester as a three credit course. (Our processes could be used just as easily in a two semester course.) During the fall semester the instructor works with many potential companies to determine which have potential projects for that spring. These projects are defined so that they are fairly well understood by everyone. It is also important to define projects that have real value to the sponsoring companies but are not on a critical path. Projects are reviewed to make sure they do not require too much special in-house knowledge, are not on a critical path for the company, do not require special equipment or software which is not available and are appropriately sized for a one-semester project. It has always been the case that we have been able to find more than enough companies willing to sponsor the student teams.

The instructor's involvement with the project does not include actual control of the project definition. The sponsor defines the project and owns the detailed requirements. However, the instructor will work with the sponsor to ensure that the project is appropriate for the course. The instructor also grades many of the customer deliverables such as the Project Initiation, Planning, Requirements, Design, and Test Plan documents as well as formal presentations at mid-term and final reviews.

On the first day of class, all potential projects are reviewed with the students. They are then given the opportunity to bid on whichever project is most appealing to them. Students are permitted to request to be on a team with another student, but are not guaranteed that this will happen. The advantage of letting students pick their own teams is that they then often work well together. However, the disadvantage is that they miss the opportunity to learn to work with strangers and experience team building.

Once the students have submitted their bids, the instructor looks over the applications and assigns teams to the various projects based on the preference or experience of the students. Then each team separately meets with the instructor before being allowed to meet with their company sponsor/mentor. The purpose of this meeting is to make sure they understand the project as much as possible from the initial description and are ready with a good set of questions for the sponsor/mentor when they first meet. Having a well thought out set of questions for the initial meeting helps considerably in getting off to a good start with the sponsor/mentor.

Industry standard project management phases are followed (Initiation, Planning, Execution, Control, Closure.) with Execution and Control being performed in parallel. The execution phase includes the standard software development activities of Requirements definition, Design, Coding, and Testing. Templates and multiple examples from previous classes are provided for use during all phases. A software process document describes what is to be done during each of these phases. These processes can be tailored (e.g. using the company coding standard instead of the default coding standard.) or replaced entirely by the team and their industry sponsor with approval from the instructor.

In addition, a website is established for keeping class lecture notes, templates, and general information for the class. Specific project related materials (Project

Initiation Document, Schedule, Weekly Reports, Requirements/Design/Testing documents, Mid-term Presentation, etc.) are kept on team-specific websites with access restricted to only the project instructor, the corresponding student team, and their mentors and sponsors. These websites use the open source project management software Trac[7].

This gives the remote sponsor/mentor the ability to closely follow the project documentation and code development. They also receive weekly progress reports and in almost all cases have weekly conference calls or meetings with their student teams. Teams also schedule occasional meetings with the course instructor to ensure that they are performing well.

Periodic process audits of all teams are conducted by a third party, usually a graduate assistant. The process audits gauge whether or not a team is following the procedures outlined by the course process document. Teams receive reports on their adherence to the process and are told when they are not in compliance with any particular process. Teams are given two weeks to fix and problems and come into compliance with the process document. Code audits are also conducted alongside the process audit to determine whether or not a team is following its coding standard.

It is possible to use our processes in programs where much broader capstone projects are involved; for example, projects involving hardware and software development as well as industrial engineering concerns. On the rare occasion that we have dealt with these kinds of projects we have not had any issues. Also, if a college or university does not have established relationships with external industrial companies, these processes will work without modification on internal projects. Over the past four years, roughly 10% of our projects have been internal.

Evolution of Processes

For the past five years we have asked both students and industry to provide feedback on our processes. Students and sponsors complete surveys at the end of the course and all student teams perform post mortems detailing their project experiences. In addition, we have worked with industry to evaluate our processes in relationship to the SEI CMMI model. A gap analysis was performed two years ago after which we worked to bring our processes in compliance with CMMI Level 2 expectations. These changes were primarily in the areas of configuration management, risk analysis, documentation of processes, and measurement and analysis. We have made multiple updates to our processes and we are currently conducting a new industry led assessment to further improve our processes.

Some specific changes we have implemented as a result of our feedback from industry include:

- Defined specific language usage (e.g. “shall”, “should”, “will”, and “may” have precise definitions.) for use in requirements documents.
- Created a default coding standard (After many suggestions and much feedback on draft documents.) for teams to follow.
- Established requirements for size and effort estimation and how they must be tracked.
- Established the need for a Change Control Board (CCB) and its role (i.e. A CCB is only used for major project changes.) for a capstone project course.
- Established use of Subversion[8] (A content management system.) to store and manage code.
- Defined how to deal with the concept of a software baseline, which is set at delivery to the customer.

Several changes have also been implemented as a result of feedback from faculty and students. Some examples include:

- Established the use of online areas for teams to store and manage proprietary documents. Originally TWikis were used, but we transitioned to using Blackboard and e-Portfolios, and eventually Trac[4].
- Established a capstone project process audit to monitor adherence to the course process and assist teams to correct noncompliance.
- Implemented more reflective opportunities (e.g. at mid-term reviews) than just the final postmortem.

Summary

We have found that the use of well defined processes for both project management and software development has resulted in high quality capstone projects. This has been validated by both student and sponsoring company feedback.

Our results from the past five years have been very positive as indicated in the tables below.

Ranking	**Number of Students**
Very Good	88
Good	23
Marginal	2
Poor	1

Table 3: Student Survey Rating of Overall Capstone Course Value (2004-2009)

Ranking	Number of Projects
Very Good	30
Good	16
Marginal	0
Poor	0

Table 4: Sponsor Satisfaction with Capstone Projects (2004-2009)

Future Work

Agile software development has had many successes and failures during the last nine years. Nevertheless, the great value of some agile development practices is widely recognized. (Agile software development refers to a group of software development methodologies based on iterative development. The main principles are related to early and continuous delivery of valuable software, managing and even welcoming requirements changes, short delivery cycles, involving both business people and developers on a daily basis, building teams around motivated individuals and giving them the environment and support they need, face-to-face communications, making working software the primary measure of progress and creating sustainable systems.) While most software development organizations are not completely agile, most have adopted at least some agile practices, even when their development framework is non-agile. Therefore, it is important that students have the opportunity to learn and practice agile techniques before they graduate into the job market. The capstone project is the logical place to provide this exposure and practice. While the CMMI and agile approaches when used independently have value, we believe that merging the best of these approaches will be even more valuable to capstone project students.

The use of agile software development practices in a student capstone project presents many challenges. Many of these challenges arise from the fact that students have a variety of constraints on their time including other courses, often a part time or full time job, possibly a family, and, for traditional age college students, their individual maturation process. Student schedules vary in how much time they can provide for the capstone project, how scheduled this time can be, and when this time occurs. Other challenges arise from the implicit nature of agile interactions. Agile development depends on informal, unscheduled communication among the development team. Much of what makes agile successful is unplanned (at least in the specific details). Also, challenge arises from the lack of individual ownership of artifacts during agile development. Instead, the team as a whole is responsible for all the artifacts (plan, code, tests).

In the future we plan to address these problems by extending our basic CMMI Level 2 process to incorporate the support of agile methodologies e.g., stories and velocities [9,10].

We also intend to conduct a sequence of empirical studies to validate student learning experiences using these techniques.

References

[1] Capability Maturity Model Integration (CMMI), Version 1.1, Staged Representation, CMU/SEI-2002-TR-029, ESC-TR-2002-029, Software Engineering Institute, 2002.

[2] Project Management Institute, *A Guide to Project Management Body of Knowledge* (PMBOK Guide) Third Edition, Four Campus Boulevard, Newton Square, PA 19073-3299 USA, 2004.

[3] Dean Knudson, Alan Braaten, Kenneth Magel, Kendall Nygard, "Software Engineering Process for Capstone Courses and Projects", *2007 International Conference on Software Engineering Theory and Practice*, Orlando, FL, July 9-12, 2007.

[4] Dean Knudson, Kenneth Magel, "Comments on the Use of TWiki, Blackboard Portfolios, and Trac to Share Proprietary Information in Student Projects", *SITE 2008 – Society for Information Technology & Teacher Education Information Conference*, Las Vegas, NV, March 3-7, 2008.

[5] Dean Knudson, Alan Braaten, "Industry/University Cooperation in Defining Software Processes for Use in Real-world Computer Science Capstone Team Projects", *SEPG 09 North America*, a Software Engineering Institute sponsored conference, San Jose, CA, March 23-26, 2009.

[6] Dean Knudson, Alex Radermacher, "Software Engineering and Project Management in CS Projects vs. 'Real-world' Projects: A Case Study", *IASTED-Software Engineering Applications 2009*, Cambridge, MA, November 2-4, 2009.

[7] The Trac Project http://trac.edgewall.org/

[8] The Subversion Project http://subversion.tigris.org/

[9] Software Development Times (SD Times), "Standing in the way of agile," December 12, 2009, p35-36, www.sdtimes.com

[10] Boehm, B., and Turner, R., Balancing Agility and Discipline, A Guide for Perplexity, Adison-Wesley, ISBN 0-321-18612-5, 2003.