Designing a Capstone Course to Simulate the Industrial Environment

Greg Speegle
Department of Computer Science
Baylor University
Waco, Texas 76798
Greg_Speegle@baylor.edu

Abstract

The creation of a capstone course for an undergraduate computer science curriculum mirrors traditional software design. Initially, a high-level goal is created. This is refined into lower-level specifications by conducting interviews with the users. The implementation is based on these specifications. In our case, the capstone course is to simulate (as much as possible) the student experiences after graduation. Lower level specifications are created by approximately 20 hours of interviews with developers, testers and project managers in consulting, development and information technology departments. The results of these interviews are synthesized into a capstone course to be offered in the Spring of 2010.

1 Introduction

Capstone courses are popular finishing touches for computer science degrees (see e.g., [1, 4]). There are several motivations for having such a course, such as domain-specific knowledge, teamwork (including communication skills) and real-world experience [9]. Our motivation is to provide the students with a realistic simulation of their future workplace environments. Of course, our students will take jobs in many different areas. From our most recent graduating class, the students went to software development companies, consulting companies, traditional information technology departments, and graduate school. All of these post-graduate endeavors have different expectations. Furthermore, large corporations have different practices from small companies. Thus, no one model will prepare every student for life after graduation.

Therefore, I propose a hybrid course that combines aspects of different organizations, preparing all students for some of their future requirements. In order to create such a hybrid, the current practices of many different organizations are studied. In particular, practitioners from a consulting company, two software development companies and an information technology department are interviewed. This paper presents the interview techniques, summarizes the results, and defines the course that evolved as a result of these interviews. It should be noted that due to non-disclosure agreements, no specifics from any organization are included. However, in many cases there is significant similarity between the organizations, making it clear how a simulation of the work environment should be structured.

2 Related Work

One of the common issues in capstone courses (and an issue raised in the interviews) is the time required to complete a capstone experience [9], with particular problems relating to the learning curve for existing systems [7]. One solution is to extend the capstone course to two semesters[1, 3], but this is not possible in our environment.

Although it is agreed that students must work in teams (see e.g., [4]), the size of teams for capstone projects varies across universities. For some projects, the teams are typically on the order of three to seven students [1, 8, 9]. Larger projects consist of teams of 10-12 in [5] or even one team for the entire class [8]. From the interview process, simulating the industrial environment requires teams of about 12-15 members.

Likewise, the responsibilities of the team members varies in capstone projects. In smaller team projects, all team members are responsible for all activities, but specialization is the norm according to the interviews. The division of labor has a significant impact on deliverables [4]. In [5], teams are divided into three groups, but the groups are all developmental in nature. In [3], quality assurance (testing) along with project management and support are critical components, which is consistent with the interviews.

Finally, it is generally accepted that capstone projects have increased student motivation if they incorporate a real-world situation [1, 9, 4, 3], and the interviews support this claim. Another attempt at student motivation is game programming [7], which was specifically not included for the construction of this capstone course. Our curriculum envisions the creation of a separate gaming capstone course in the near future.

3 Interviews

One of the important contributions of this work, is the interview process used to develop the specific goals for the capstone course. The typical interview transitions through three phases. The first phase consists of getting to know the person being interviewed. This is accomplished by asking the interviewee to describe their current work. This technique has the advantage of allowing the interviewee to relax, as some of the them were understandably nervous about the process. However, since they are the experts about their profession, that question could always be answered easily. Follow-up questions provide the basic understand of the interviewee's occupation. The second phase of the interview process consists of questions specifically tailored to the individual being interviewed. For example, a management type would be asked questions regarding the structure of a project, while a tester might be asked questions about the design of end-to-end tests. The most frequently asked questions are in Figure 1. Information regarding specific questions appears in Section 3.1. The final phase of the interview is wrap-up and appreciation. In this phase, the interviewees ask questions and make additional statements that are valuable and might have been missed. Also, it is important to sincerely thank the interviewee for the time they took out of their busy schedule to help with the project.

In order to initialize the interviews, contacts within the company are needed. These contacts should be of sufficiently high management positions that they are able to organize interviews. Possible sources for such individuals are advisory boards, alumni, and colleagues of faculty. These individuals need to have an interest in the department and its graduates, and a willingness to allocate resources to the interviews (and away from the normal work efforts). Ideally, these individuals should be from diverse areas within computer science, so as to broadly cover the current practices.

Working with the contacts, a series of interviews should be scheduled with as many different people within an organization as possible. However, more important than the number of people is the diversity of the jobs held by the interviewees. There are four broad categories or personnel within a software development organization – developers, testers, managers, and support (which includes databases, networks, business planning, etc). Among these, the managers are the most critical to interview since they represent the equivalent of a professor in a capstone course, but at least one of each type needs to participate.

3.1 Interview Questions

Over twenty hours of interviews were conducted in four distinct organizations. Individual interviews ranged from thirty minutes to two hours in length. The majority of the interviews were conducted in person, with the rest on the telephone, both with individuals and via conference call. The typical interview was a one-on-one session, but there were times when multiple people were interviewed together.

While interviewing developers, my questions focused on the development process, environment and paradigm. In particular, I learned about the

- specifications provided
- code review process
- unit testing process
- development paradigm (agile, waterfall, etc.)
- documentation requirements

The importance of testing was strongly reinforced by the interview process. When interviewing testers, the key issues are

- specifications
- automated test suites
- levels of testing (functional, system, end-to-end)
- feedback

Interviews with management personnel provided insight on the overall structure of projects, including issues such as

- timeline
- resource allocation (specifically personnel)
- "feature creep" management
- installation and training
- maintenance

Support personnel is a broad term I use to represent all people outside of the typical code-test-deploy cycle. They are critical to the success of the enterprise, and have diverse titles across the institutions. Support personnel are concerned with (among other things)

- Security
- Return on investment for the project
- Disaster recovery
- Network capacity
- Database capability

3.2 Responses by Interviewees

The response to the interviews was extremely positive. The upper-management contacts not only made arrangements for the interviews, but followed up to ensure that everything went well. The individuals interviewed seemed to be genuinely interested in the project and provided as much information as possible. The desire to help is quite strong, as in more than one case, individuals sacrificed their lunch time for interviews or follow-up conversations.

It should be noted that after explaining the goal of the capstone course, a common sentiment among many of the interviewees is that such a course would have been beneficial for them as students. Likewise, a desire to see the end results of the interviews lead to this paper. In conclusion, the interview process is very positive, as it allows academics to stay in touch with the real world, and allows practitioners a chance to provide guidance for the next generation.

1. What is your general responsibility on the project?
2. What are your typical daily activities?
3. What is your development environment like?
4. How are the priorities for the project established?
5. How are changes in priorities handled?
6. What is the testing procedure?
7. Do you use off-site or off-shore personnel, and if so, how are they included in the process?
8. How is information shared between different parts of the enterprise?
9. If you have 15 part-time new hires for 15 weeks, what would you try to accomplish with them?

Figure 1: Frequently asked questions during interviews. Not all interviewees were asked all questions.

4 Results

All of the results from the diverse interviews have been synthesized into the following areas. None of the institutions incorporate all of these components, but the commonalities between the organizations yield several interesting results.

4.1 System Integration

Within the consulting and information technology groups, a very high percentage of the work focuses on integration of already existing software systems. In general, it is far more efficient to extend existing or off-the-shelf software than to

generate new code. Even the development companies rarely program in a vacuum, as software must function with outside vendors or other internal projects.

Integration is concurrent with development (and therefore, design and testing). Although the process is quicker for integration, the process must be planned and tested just as rigorously as pure development systems. Likewise, integration with other teams or external products must be carefully designed and tested. As a result, the capstone project should require integration with existing software, or integration of two independently developed pieces of software.

4.2 One Project - Three Parts

All of the companies interviewed have structures for their projects, consisting of developers, testers and project management/support (PM/S) components. Although all environments have roughly equal number of people involved in development and testing, the PM/S component varied greatly at the different organizations. For example, the consulting and IT environments place increased emphasis on training, considering that an integral part of the product development, while the development environments are not as concerned about training. The PM/S team includes design, documentation, management, resource allocation, disaster recovery, ethics and security. "Developer" is used to describe any person who generates code of any sort, from scripts to access data in existing software systems to writing device drivers for an operating system.

Within each environment there exists the notion of a production environment and a development environment. It is crucial that the developers and the PM/S team coordinate their activities so that the production environment can adapt to any changes efficiently. Likewise, the PM/S team should be quick to provide the support needed for the developers in terms of database access or other resources. Integration of the three parts within one project is an fundamental objective of the capstone experience.

4.3 Documentation

Documentation is handled in both a bottom-up and top-down style. The top-down documentation consists of specifications which are generated in the design process and distributed to both the development and testing teams. The bottom-up documentation is generated by the developers and is used in code review and testing. In general, the bottom-up documentation should satisfy the top-down documentation after iterations over both documents. The coupling between the documentation and the code varied between organizations.

4.4 Testing

Although described using different terms, the basic testing mechanism is similar in all environments. The developers are responsible for unit testing. The testers are responsible for checking progressively larger units, eventually culminating in end-to-end testing for the entire system. In some cases, end users are integrated into the testing process for early feedback. In all cases, testing is automated as much as possible. Test cases and test programs are development concurrently with the software.

4.5 Code Review

Although the process varied between companies, external review of written code is a key part of the development process. One aspect of code review is the concept of knowledge transfer. It is important for others within the organization to be familiar with the development, and code review facilitates this transfer.

4.6 Security

In each environment, security plays a role in the development process. It can be directly supported at the developer level, or it can be part of the external testing and review process.

4.7 Feedback

Every development process requires feedback from the users to determine the success of the project. This feedback has different names in different companies, but it is addressed in all of the organizations. The project must have a mechanism for providing feedback from a userbase that is outside the students. PM/S will be responsible for the feedback portion of the project. However, implementation of changes deriving from feedback must be allocated to future semesters.

4.8 Feature Creep

Every enterprise is concerned about adding additional features during the development process. The procedure for handling feature creep varied between the companies, from trying to accommodate the features as much as possible to trying to eliminate the requests by upfront discussion of the requirements and the costs of the changes. Feature creep will need to be an intentional aspect of the course.

4.9 Good Luck with That

A question asked of all managers was "What project would you do with 15 part-time new hires for 15 weeks?" The universal reply was "Nothing." Within each company, there is a learning curve required. The shortest learning curve mentioned for any project was three months. The longest was a full year. Given that a semester is fifteen weeks, the learning curve is a significant barrier to accomplishing useful work in the capstone course.

4.10 Environment

A primary concern for an academic institution is the cost of the computing environment needed to simulate the workplace. Although the computing resources varied between the developer, tester and PM/S teams, none of the environments had excessive capabilities. Dual monitors were common, but not ubiquitous. In general, most work was performed on servers, not on desktop machines. As a result, the individual machines, while powerful, were not prohibitively expensive. On the other hand, the server farm capabilities far exceed my department's resources, requiring a more modest project scope. However, that is consistent with the time limitations for the project. Virtualization is common, and should be encouraged.

Another challenge within the environment is the diversity of software required, particularly in legacy systems. Given that my department is primarily motivated to produce students capable of meeting the demands of the future, it is difficult to simulate the requirements of the past.

4.11 Failings

The limitations of an academic environment prevent simulation of some aspects of the workplace. In particular, we cannot simulate the 24/7 development cycle, including the difficulties in communicating with a different culture. Our institution is fairly homogeneous, and the students in the capstone course consist almost entirely of upper middle income white males, 21-22 years old.

Another limitation is the fifteen week course time limit. Total project development time is typically measured in several months to years. As a result, the project will have a limited scope. Related to this limitation is the business planning typical in all software projects. Whether the project is to support business or the business is the software project, each organization requires a justification for any software, be it projects, bug fixes or upgrades. Clearly, within an academic department, similar motivations do not exist.

Finally, although maintenance is a significant portion of any software project, it is not possible within a fifteen week time frame. Future work is to incorporate maintenance, possibly by adding prerequisites to the capstone experience that allow students to work on bug fixes early in the semester.

5 Course Design

A project team will consist of no more than fifteen students, and ideally no fewer than nine. In order to accommodate the one project - three parts construction commonly found during the interviews, the team will be divided into three groups – development, testing and PM/S.

5.1 Project

In order to facilitate system integration, the project will be a Firefox addon that will allow users to select arbitrary quotations from a database. This will require the group to familiarize themselves with a new API and a new paradigm, thus fulfilling the system integration requirement. Documentation will be produced by all groups. Consistent with the interviews, the development team will conduct unit tests and the testing team larger tests. Review will be carried out across all aspects of the project (design documents, code, and test suites). The PM/S

team will create a web site to gather user feedback and collect bug reports. The web site will also allow downloads of the software. Specification change will occur sometime in the middle of the semester. A dedicated lab will be used to allow the students complete control over their environment. Additionally, one of the responsibilities of the PM/S team will be the initial construction and maintenance of a repository. Everything will be maintained in the repository, including all versions of the source code, documentation, and testing suites.

5.2 Work Structure

Before the course begins, students signed up for the capstone course will receive the high-level description of the project. The students will be expected to come to the first class session with a thorough understanding of these materials. The initial class session will serve as the project kickoff, where the structure of the class, the expectations and the project will be introduced. At the end of the first class, the students should know their groups, their initial goals and their weekly obligations.

The interviews indicate a structure for project management with a lengthy iteration cycle. This process works for large-scale industrial projects, but creates significant downtime for the groups not involved in the initial aspects of the project. As a result, an iterative method with a 3-4 week timebox will be employed. Individual features will be assigned to specific timeboxes by the groups. The teams will work in conjunction with each other to design and document, implement and test each feature concurrently. Constant revision and cross-communication is expected. As such, this schedule more closely matches the Unified Process as opposed to the waterfall model [6].

For the PM/S group, the requirements are to generate human understandable documents for each feature, and map these features to the overall project. The implementation and testing groups are to provide feedback for any aspect of the design which does not allow obvious implementation and testing. Likewise, the implementation and testing groups are to provide feedback on any external documents which are not clear to an end user.

The testing group designs test suites based upon the initial capabilities of the feature, as well as test suites for combinations of features. The test suite designs are vetted by the PM/S and implementation groups. The goal for the testing group is to build the initial test suite by the time the initial implementation is complete. This will allow testing to proceed in parallel with the design and development. Each test run will produce a test report showing all tests applied, and for all failed tests, the expected and produced results. Ideally, all tests will be automated, but given the browser based nature of the project, manual tests may be required. However, all tests must be reproducible. As the project matures, testing with other addons and end users is required.

The implementation group will initially work on understanding the technology used for the project. In this case, XUL (XML User Interface Language) [2]. Initially, this knowledge will be used to refine the feature design and to develop the first working product. Code review will be practiced by the members of the testing and PM/S teams in order to ensure good programming practices as well as additional aspects such as security and efficiency. The implementation group releases each build to the test group, and when all tests are satisfied, the build is released (although not necessarily in its final form).

Each group will meet weekly with the faculty. During the meeting, every student should be able to report on their progress. This progress should be documented with additional material in the repository, and with a documented review of their work.

5.3 Feature Creep

It is anticipated that additional requirements will be added to the project at some time. For the Firefox project, the additional requirement might be a "Next" button which produces another quote, or a "Next by Author" button that produces another quote by that author, or additional subclasses for the data. It is also possible for features to be dropped from the requirements.

5.4 Grading

Assigning a final grade to the project and then determining individual student participation is not reasonable given the team sizes. As a result, grades will be assigned on a bi-weekly basis according to the contribution of each student. The expectations for the following report period should always be clear, and the success or failure

of a student to perform his assigned task should also be obvious. Combining the requirement for everything to be in the repository and for reviews to be submitted each meeting will require the faculty to read a large amount of material. To make this a manageable task, standardized review forms should be developed and followed.

6 Conclusion and Future Work

Developing a capstone course to simulate industrial experience is very similar to a project itself. The initial requirements are refined by gathering additional information. In this case, the additional information is derived from the developers, testers, managers and support personnel in industry to determine the environment to be simulated. The interview process is well received.

Several key components emerged from the interview process. The size of the teams and the specialization of the team members is clearly important. In turn, this leads to the importance of documentation, testing and management of new features, which may be overlooked in a computer science environment. Integrating the development process within another system is also crucial.

With the knowledge from the interviews, the capstone project for the spring will be an addon to the popular Firefox browser. This will require system integration, new technology, crucial testing and significant project management and support. It will also result in a real-world application that can be downloaded by others.

All simulations have limitations, and the capstone course will not be able to reproduce the experience perfectly. For example, it is a common situation for development to be a 24/7 process with team members all around the world. More importantly, the semester provides a hard time limit which cannot be changed. As a result, whatever is completed in fifteen weeks is the end result of the project.

In the future, adding a pre-Capstone course requirement might allow the learning curve to begin earlier, and allow maintenance to be incorporated into the capstone experience. This course would be a 1-hour pass/fail course that requires the students to pass an exam in order to take the Capstone course. Maintenance jobs would then be assigned in the initial phase of the Capstone course.

Likewise, the project can be extended or even totally changed (version 2.0) in future semesters. Support for larger citations – such as literature excerpts – can be added. It is possible to interface with online sources to provide current information such as headlines. As a result, continued improvement of the product will provide course material for several semesters.

References

[1] Russel E. Bruhn and Judy Camp. Capstone course creates useful business products and corporate-ready students. *SIGCSE Bull.*, 36(2):87–92, 2004.

[2] Mozilla Developer Center. XUL Tutorial. https://developer.mozilla.org/en/XUL.

[3] A. T. Chamillard and Kim A. Braun. The software engineering capstone: structure and tradeoffs. In *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 227–231, New York, NY, USA, 2002. ACM.

[4] Tony Clear, Michael Goldweber, Frank H. Young, Paul M. Leidig, and Kirk Scott. Resources for instructors of capstone courses in computing. *SIGCSE Bull.*, 33(4):93–113, 2001.

[5] Annegret Goold. Providing process for projects in capstone courses. In *ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education*, pages 26–29, New York, NY, USA, 2003. ACM.

[6] Craig Larman. *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 3rd edition, 2005.

[7] Ian Parberry, Timothy Roden, and Max B. Kazemzadeh. Experience with an industry-driven capstone course on game programming: extended abstract. In *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 91–95, New York, NY, USA, 2005. ACM.

[8] Michael V. Stein. Student effort in semester-long and condensed capstone project courses. *J. Comput. Small Coll.*, 18(4):200–212, 2003.

[9] Richard C. Thomas and Rebecca Mancy. Use of large databases for group projects at the nexus of teaching and research. In *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 161–165, New York, NY, USA, 2004. ACM.