Designing a Capstone Course to Simulate the Industrial Environment

Greg Speegle
Baylor University

Designing a capstone experience to simulate the industrial environment provides excellent preparation for students' lives after graduation. To determine this environment, current industry practices are modeled from 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 new course.

Corresponding Author: Greg Speegle, Greg_Speegle@baylor.edu

Introduction

Capstone courses are popular finishing touches for computer science degrees.[1,2] There are several motivations for having such a course, such as domain-specific knowledge, teamwork (including communication skills) and real-world experience.[3] 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 schools. 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, this course 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 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 environment should be structured.

Related Work

Capstone courses exist at many universities, and the experiences at these universities are well documented. Although the literature agrees that real-world involvement is highly motivational,[1,2,3,4], and that time constraints are a significant obstacle to learning,[1,3,4,5] on other issues there are diverse approaches. In particular, there is wide variety in the size of the teams and the homogeneity of their responsibilities.

For some projects, the teams are typically on the order of three to seven students.[1,3,6] Larger projects consist of teams of 10-12[7] or even one team for the entire class.[6] Likewise, the responsibilities of the team members varies in capstone projects. In smaller team projects, all team members are responsible for all activities, but division of labor has a significant impact on deliverables.[2] In one case, teams are divided into three groups, but the groups are all developmental in nature.[7] The experience most similar to ours creates separate concerns for quality assurance (testing) along with project management and support.[4]

Interviews

One of the important contributions of this work, is the interview process used to develop the specific goals for the capstone course. The interview process was initiated by contacting representatives from diverse companies associated with Baylor University or personally known to the faculty. Members of the Board of Advocates for the School of Engineering and Computer Science were enthusiastic about this project, and arranged interviews with personnel who regularly interact with new hires. Four companies were selected to complete the interview process – one in consulting, two in software development and one in information technology.

Each company setup a series of interviews with employees with diverse responsibilities. Specifically, four broad categories of personnel were interviewed – developers, testers, managers, and support (which includes databases, networks, business planning, etc). I was able to interview an example of each type with most of the companies and first line managers at every institution. The managers were the most critical to interview since they know the expectations for new hires.

The typical interview transitioned 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 provided the basic understanding of the interviewee's position within the organization and their basic responsibilities for the software development process. The second phase of the interview process consisted 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 final phase of the interview is wrap-up and appreciation. In this phase, the interviewees asked questions and made additional statements that are valuable and would have been missed otherwise.

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 with groups 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, we discussed the specifications provided, code review process, unit testing process, development paradigm (agile, waterfall, etc.) and documentation requirements at each of the cooperating institutions. These discussions reinforced the notion of differences between development environments, but commonalities did emerge. In particular, the importance of documentation to allow collaboration (both in 24/7 development cycles and with other units such as testing) is strongly emphasized.

The importance of testing was reinforced during the interviews. When interviewing testers, the key issues to emerge were specifications, automated test suites. levels of testing (functional, system, end-to-end) and feedback. From these interviews, it became clear that development of automated test suites must be a key component of any industrial simulation.

Interviews with management personnel provided insight on the overall structure of projects for capstone courses, including issues such as a timeline,resource allocation (specifically personnel), "feature creep" management, installation and training and maintenance.

Support personnel is a broad term representing all people outside of the typical code-test-deploy cycle. They are critical to the success of the enterprise, and have diverse titles across institutions. Support personnel are concerned with (among other things) security, return on investment for the project, disaster recovery, network capacity, and database capability.

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.

Also, 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.

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.

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 must require integration with existing software, or integration of two independently developed pieces of software.

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 a fundamental objective of the capstone experience.

Team Interaction

Team interaction is a broad category for handling issues such as documentation, testing and code review. An important aspect of large project building is cooperating with others, either internationally in 24/7 development cycles or across the hall in terms of knowledge transfer. Organized and scheduled reviews of not only code, but also documentation and testing plans are critical to the success of the project. This ensures that all project members understand the various components. One important consequence of knowledge transfer is the ability to rapidly take over for someone who is unable to continue with the project.

Additionally, 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.

Challenges

A primary 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.

One 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.

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.

The limitations of an academic environment prevent simulation of other aspects of the workplace as well. 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.

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. The actual project itself will be determined the semester before the capstone course is offered.

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 deliverables will be introduced.

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 4-5 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.[8]

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.

Meetings with the faculty must be held regularly. These meetings should vary from individual meetings with students to group meetings to the entire class. The purpose of these meetings is to evaluate performance and to mediate any disputes. The faculty has a primary obligation to prevent failures in one area from damaging the other participants.

Grading

One of the requirements for each phase of the project will be a timeline resulting in the completion of the requirements for that phase. The timeline will detail the individual requirements for each student during the phase, including due dates for the tasks. Individual grades will be assigned based on the completion of these tasks, both in terms of quality and punctuality. A student's final grade will be a combination of their individual work and the success of the overall project.

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 interviews with the developers, testers, managers and support personnel in industry to determine the environment to be simulated.

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 under emphasized in an academic environment. Integrating the development process within another system is also crucial.

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.

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. ISSN 0097-8418. doi: http://doi.acm.org/10.1145/1024338.1024379.

[2] 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. ISSN 0097-8418. doi: http://doi.acm.org/10.1145/572139.572179.

[3] 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. ISBN 1-58113-836-9. doi: http://doi.acm.org/10.1145/1007996.1008039.

[4] 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. ISBN 1-58113-473-8. doi: http://doi.acm.org/10.1145/563340.563428.

[5] 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. ISBN 1-58113-997-7. doi: http://doi.acm.org/10.1145/1047344.1047387.

[6] Michael V. Stein. Student effort in semester-long and condensed capstone project courses. *J. Comput. Small Coll.*, 18(4):200–212, 2003. ISSN 1937-4771.

[7] 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. ISBN 1-58113-672-2. doi: http://doi.acm.org/10.1145/961511.961522.

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