AC 2008-1760: SERIOUS GAMES AS SOFTWARE ENGINEERING CAPSTONE PROJECTS

Bruce Maxim, University of Michigan - Dearborn

Bruce R. Maxim is an Associate Professor of Computer and Information Science at the University of Michigan-Dearborn where has taught courses on software engineering, game design, and artificial intelligence for 23 years. His current research interests include software usability and accessibility issues, game development, and software quality assurance.

Serious Games as Software Engineering Capstone Projects

Abstract

This paper describes the author's experiences with teaching an industry-based capstone design course. In this course, students work as members of small teams to complete software development projects. These projects proceed from requirements gathering, to analysis, design, implementation, and delivery of products to real-world clients. In recent years, several of these projects have involved the development of serious games for real-world clients. Serious games are games whose purpose is education in its various forms, rather than entertainment. Serious games and simulations can be good candidates for student projects that provide them with opportunities to manage projects with real-world development constraints and deadlines. A final cumulative written report and oral presentation is required of all teams, along with a formal acceptance letter from their real-world client. The student projects and design activities that result from this course receive frequent praise from local computing professionals and accrediting agency reviewers.

Background

The idea of a capstone project course for undergraduate computing majors is not new. The authors proposed creating such a course at the University of Michigan-Dearborn in 1993. Capstone courses in computing have traditionally tried to provide senior students with experiences similar to those encountered in professional practice[3, 4, 6, 8]. In several cases, course developers make argue that the purpose of such a course is to help the students integrate theoretical computing concepts with the demands of computing practice.

One approach in recent years has been to involve students in projects that satisfy the needs of real-world clients. Real-world capstone design projects can satisfy the expectations of ABET[7]. One difficulty with this approach has been the fact that real-world problems frequently require more that one semester to solve. Because of this, some schools have offered the senior design experience as a two-semester course sequence or have offered the course outside of the normal academic year[2].

Several real-world project courses require students to work on teams as a practical lesson in group dynamics[1, 2, 8]. Most instructors organize their courses around the notion of having students follow a computing project from its feasibility study through its design, implementation, documentation, and testing phases.

There is consensus among members of our department's professional advisory board that professional practice invariably requires strong verbal and written communication skills. To develop their oral communications skills, students need opportunities to make presentations and to have opportunities to review other students' presentations. Some instructors believe that the project activities inherent in real-world software development encourage students to improve their written and oral communication skills[1, 5].

It is difficult to motivate students to focus on the non-technical aspects of project management, if the final software product is the only artifact assessed by the instructor. Students can be convinced that real-world problem solvers complement their technical skills with knowledge of project management skills. Perhaps, one of the best ways to teach the importance of managing project resources and fostering good team dynamics is to allow students to manage a real-world project with serious development constraints including concrete deadlines[1]. Development of serious games is no exception. Students working on serious games in our course sequence are required to create the same milestone documents as students working on traditional software engineering projects.

The annual revenue generated from the sales of computer games in the United States alone exceeds $10 billion dollars. Statistics indicate that the revenue generated by the computer game industry will continue to exceed that of the motion picture industry[11]. Game development projects have a reputation for late delivery and being over-budget. The minimum cost incurred by a failed game development project ranges between $150,000 and $750,000[14].

Game developers are beginning to understand that it is important to treat computer game design in the same way that other software engineers approach projects involving large numbers of people and significant investments of time. The process of determining the technical requirements for a game software product is similar to that used to specify any other type of software product. Students seem easily motivated to spend countless hours working on game development projects[2, 6, 11].

The development of computer games is labor-intensive[12]. Game developers rarely build computer games on their own, as they did 15 years ago[13, 14, 15]. Many best-selling computer games contain thousands of lines of code and have multi-million dollar development budgets. Modern game development requires the effort of a team of skilled professionals to integrate multimedia content and complex computer software[16, 17]. Textbooks emphasizing software engineering principles for game development are beginning to appear[13, 14, 17]. It is difficult for students to comprehend the benefits and logistical problems of working on interdisciplinary teams by simply reading textbooks. Development of serious games can allow students to experience the challenges and benefits of working with content developers outside the field of software engineering.

Several recent computing conferences have included descriptions of capstone project courses where students implement games as a means of demonstrating their ability to apply their knowledge of software engineering principles[9, 12]. On our campus, most of the student capstone projects implemented during the past ten years do not involve game development. Most faculty members in the Computer and Information Science Department are impressed with the quality of the recent serious game products created by our capstone design students.

Several recent capstone software engineering projects have involved developing serious games for real-world clients. The primary goals for the requested serious game projects are usually educational in nature. Students in our Computer Science Game Design application track need to create game-based capstone design projects. One purpose of this paper is to examine the lessons learned by students developing serious games as capstone design projects. The expectation is that

students working on serious games will learn many of the same lessons as students working on traditional software projects.

Serious Games

Serious games make use of the artistic medium of games to deliver a message, teach a lesson, or provide an experience. Serious games may be entertaining, but that is not their primary purpose. Video games can teach hand-eye coordination, spatial relationships, and encourage exploratory experiences. Immersion in simulated environments has increased learning speed and retention for some tasks. Video games can engage players for two or more hours, yet these same students may lose interest in classroom activities after only fifteen minutes[18].

Several constraints present in serious game creation provide for interesting design tradeoff considerations during the software development process. Developers often design serious games for wide audiences and not just hardcore game players. One of the design challenges present for serious games is that the games produced often need to run in lower end computing environments without sacrificing the game's core-learning objectives. Reality does not map exactly in a computer game, so software engineers must weigh each simplification assumption carefully to avoid damaging the verisimilitude of an educational experience. Shortcuts from entertainment games (i.e. use of random numbers, time compression, headache removal, perfect communication) may not be appropriate when the learning outcomes for a game may have life and death consequences in the real-world[18].

The intellectual challenges present in making a good game are part of the reason people want to work on the game development process. Game development projects are often large and complex projects. The workflow to complete a large game is non-trivial. Project managers need to coordinate the efforts of programmers and third party game asset creators. Game developers need to manage both technical risk and project risk to ensure the timely completion of a good serious game. Creating and modifying game engine code requires good algorithmic programming knowledge on the part of the developers[20]. In short, serious games require serious software engineering skills to complete them on time and within budget.

Course Description

We organize our software engineering capstone design experience as two, two credit-hour courses (CIS 4961 and CIS 4962) which students complete over two semesters. These courses are required of all software engineering majors. Most students taking these courses do not create serious games as part of their capstone design projects. The educational outcomes for the capstone design experience appear in Table 1.

Students enroll in CIS 4961 after they complete all required software engineering courses. The capstone projects generally require about 500 hours of student effort to complete. The major activities in CIS 4961 are requirements gathering and project planning (including risk management and quality assurance efforts). The major activities in CIS 4962 are product design, implementation, and testing. Serious game projects usually make use of a rapid prototyping process, so a clear distinction between the analysis and design phases of a project may not exist.

Students work in three or four person teams. In exceptional cases and with proper justification, smaller or larger group sizes are permissible. For serious game projects, teams often find it necessary to add the efforts of creative people from outside their team to obtain the art and audio game assets needed.

Students select their own teammates and determine their own plan for rotating team leadership. We have observed that students tend to organize themselves so that one person is the hardware expert, one person is the software expert, and a third team member takes charge of documentation and coordination of activities. The asset creators similarly tend to specialize in the areas of 2D art creation, 3D animation, and audio design.

Table 1: CIS 4961 and CIS 4962 Educational Outcomes

1. The ability to lead a software development team in the successful completion of a software project for an external customer
2. The ability to manage the successful completion of a software project for an external customer
3. The ability to write a management plan for a software project that involves time and resource estimates, personnel scheduling detail, and its production costs
4. The ability to create a risk monitoring, mitigation, and management plan for a real-world software development project
5. The ability to create and execute a software quality assurance plan for a real-world software project
6. The ability to conduct a project post-mortem to determine the effectiveness of a project plan
7. The ability to write a complete analysis document and create an analysis model for a real-world software system
8. The ability to write a complete design document for a real-world software system
9. The ability to participate on a team to design and implement a software system to meet the needs of a real-world customer
10. The ability to make use of appropriate software engineering tools in the development of a software product
11. The ability to create and execute a test plan for a real-world software system, including test case creation based on the specified requirements
12. The ability to conduct two thirty-minute seminar discussions on ethics or professional issues papers requiring independent library and/or Internet research.

Classes meet for two hours each week for 56 semester contact hours over a period of 8 months. The ACM/IEEE Computing Curricula 2004 recommendations suggest that 11 lecture hours be devoted to social, ethical and professional issues. We include this material in our capstone design experience. The recommended topics associated with these knowledge units come from four broad groups of topics:

- *Historical and social context of computing* this includes: definition of computing subject matter, comparison with other disciplines and computing technology uses/limitations.
- Topics associated with *responsibilities of the computing professional* such as: professional societies, social responsibility, professional growth and ethical choices.
- *Risks and liabilities* focusing on: risk types, loss types and liability.

- *Intellectual property* including: definition of intellectual property, protection of intellectual property and infringement penalties.

During the first two weeks of CIS 4961, class time is devoted to course introduction and project organization issues. After project teams assemble, class meetings consist of seminar-type class discussions on professional issues or team presentations of significant project milestone artifacts. These presentations might consist of a brief progress report, a structured walk-through of a work product, or a product demonstration. Students select discussion papers from a textbook containing a collection of contemporary papers on computing ethics and professional issues[10].

In addition to the two hours of class-time each week, students put in an additional six hours per week out-of-class on their project. Six hours of out-of-class work is typical for a two-credit hour course. The out-of-class time in the capstone course consists of team interaction, project planning, software design, product implementation, presentation preparation, report writing, meeting with clients, and consultation with instructor. The six hours of outside work is very important as a means of fostering team development.

The role of the instructor in our course is that of a coach or mentor not a project manager. The students handle routine client contact. Project scheduling and progress tracking is also handled by the student teams. The instructor is available to help student teams resolve unusual problems with the project and the client. The instructor provides feedback on the milestone documents and presentations. Students revise their milestone documents based on the feedback from the instructor and their classmates following the presentation of their documents. The instructor participates in the paper discussions, but does not control their direction or content.

A final presentation is required of all teams at the end of the second semester and includes a product demonstration and report. The sections in this report appear in Table 2. Students leave copies of their final report with their client and the instructors. Copies of the final reports reside in the department library.

The final project presentation is very important as a vehicle for assessing oral communication skills. The project presentation requires the use of audiovisual support such as a computer projector and PowerPoint slides. Students are required to be professionally dressed for the presentation. The entire department and other interested individuals receive invitations to the final presentation.

Students must present a letter of acceptance from their client to the instructor in order to receive a grade for CIS 4962. The use of the client acceptance letter is a very important element of our course to drive home to students the importance of satisfying their clients' needs.

The instructor's assessment of the student work products, tempered by input from the client acceptance letters, determines the student course grade in CIS 4962. The final presentation and report account for 40% of the student's course grade. The student discussions on professional practice and ethics papers account for 20% of the course grade. The milestone documents and presentations account for the remaining 40% of the course grade.

Table 2: Final report Section Headings

Section	Heading
1	Cover Page
2	Acknowledgments
3	Table of Contents
4	Introduction/Overview
5	Requirements/Analysis Model
6	Hardware/Software Design
7	Implementation Details
8	Testing/SQA
9	Future Maintenance Suggestions
10	Client Acceptance Letter
11	References & Bibliography
12	Appendix A - User Manual
13	Appendix B - Program Listing, Sample Output, Diskette
14	Appendix C - Team Member Resumes
15	Appendix D - Project Plan & Log Book
16	Appendix E - Project Demo Notes
17	Appendix F – Final Presentation Slides

Projects

We have had several student teams create serious games as part of their capstone design coursework. We describe three of these projects in this section of the paper. It is interesting to note that in each case, students were motivated to continue work on these projects for several months after receiving the client acceptance letter and final course grade.

One serious game student project titled the Wood Badge Game System is an imitation of the popular TV game show Jeopardy. This Detroit Area Council Boy Scouts of America uses this game as part of their premier adult training course known as Wood Badge. The intention of the Jeopardy Game System is to provide an entertaining and versatile question-and-answer type of game that is at the same time robust and easy to use.

This game system is a portable standalone system for use in outdoor classroom environments. The hardware for the game system consists of a laptop computer, a computer projector, and a set of eight distributed buttons used by teams to signal their responses. The students manufactured the button system by dismantling and rewiring a Gravis game pad so that groups several feet away form one another could use the buttons.

The user interface allows non-computing experts to make use of the program with minimal training. The system allows the client to create a database of questions and answers and use the database to create several different question sets. Because this game is an instructional game, random question sets are not acceptable. Another design constraint is that every judging and

scoring decision made during game play must be reversible. In the heat of game play, judging and input errors may occur and they must easily correctable.

Students have created this system twice. The first time in 1999, the implementation environment relied on Visual Basic and Microsoft Access. The student group creating this product received designation of best Computer and Information Science project in the annual College of Engineering and Computer Science 2000 Senior Design Competition. The judges of this competition are professional engineers from several fields. This project was completely reengineered in 2007 and implemented using C# and MySQL to improve real time performance issues. This was not merely an exercise in platform translation or routine maintenance. The second student team began with a review of the use cases and worked to devise a more robust system with greatly improved error handling. The Boy Scouts regard the Jeopardy Game System is as a valuable resource to the Wood Badge course.

A second serious game project, SimTraffic3D, required the creation of a real-time, 3D graphical simulation tool for vehicle traffic research and education. The system user designs and configures various experiments to provide input to the system. The system inputs include vehicle size, performance specifications, and environment settings. The system outputs are the results of running a simulation experiment, including locations of the vehicles, steering wheel positions, velocities, and time traveled (in addition to the graphical output). The simulation models inter-vehicle communication by creating virtual wireless, ad-hoc networks among the vehicles. The communication networks allow simulated vehicles to share traffic load costs with each other in the environment. The broad goal is to allow the vehicles to mange traffic so that no major congestion points occur.

The system features fully autonomous vehicles that respond to traffic lights, buildings, roadways, and yield to other vehicles. Inter-vehicle communication allows vehicles to avoid high-traffic areas and accidents using a dynamic A* path-finding algorithm. A construction tool allows users to create customized urban environments and roadways. Users are able to model graphically any road system layout. Users can modify the acceleration, distraction, braking characteristics, and dimensions of any vehicle. The user can also customize weather, time of day settings, AI behavior, and simulator content. These settings are stored in an editable configuration file. Users can introduce vehicle accidents and obstructions while the simulation is running. This project won designation of the best senior design project from any department in the College of Engineering and Computer Science 2006 Senior Design Competition.

The title of a third serious game project is Street Smart Detroit. This project involved the development of a single level for a first person action game using the Torque 3D Game Engine. The main premise of the game level is to allow a player to experience the life of a homeless person trying to survive one night on the streets of Detroit. The player interacts with several non-player characters and attempts to find food, warmth, documentation, and shelter to win the game. The player's challenge is to see if he or she is smart enough and tough enough to live (and eventually get off) the streets of Detroit. The client for this game is a project manager from City Connect Detroit.

The educational goal for this project is to design game play to raise the player's awareness of the homeless problem. Through the medium of a game, the player views the issues of homelessness through the eyes of a homeless person. The game includes some facts about homeless people with the goal of educating the player. The client's expectation for the game is to change player attitudes and make people see the homeless as human beings.

One of the software engineering challenges in this project involves ensuring that the delivered game runs on the relatively modest hardware specifications of client's laptop. The students needed to locate and negotiate the voluntary participation of content producers (artists and writers). Managing a large repository of game assets as well as program code presented the team with new project management experiences. Trying to keep the client from expanding the scope of the project with each delivered prototype was also a challenge. The developers also needed to keep in check their desires to add fun features to the game when they came in conflict with the client's desires to have an accurate portrayal of life on the streets.

Lessons Learned

Examination of the student post mortem documents from these three projects provides some insight into how successfully the teams met the course goals. The list of lessons learned by students working on serious game projects is similar to the lessons learned by the author's students working on other types of software development projects[19].

- Getting early requirements approval by the client is essential to avoid scope creep.
- Get early client involvement by giving the client prototypes to review.
- Iterative/incremental process models are practical and easy to adhere to.
- Incremental design processes work well for developing applications that require the integration of many concurrently developed components.
- Learning is good but progress is better, using cool technologies for their own sake is not a good practice.
- The reflexive nature of designing and coding is acceptable, but only if developers can maintain the quality of the project documentation.
- Keep the requirements specification up to date at all times and have client review all changes.
- Make use of computerized version control early in the project and use it throughout.
- Time management is the key to success in large projects.
- Begin implementation during the first semester.
- Do not procrastinate. Work on the project during the semester break.
- Testing requires more time than one thinks.
- Risk mitigation must begin as soon as a risk becomes a threat.
- Game developers need to be serious about the formal technical review process.
- Game play and level design are as important to the presentation of a game as the technical details of the implementation.
- Identify source for art resources early in the project.
- Identify and verify content resources before completing the system design.
- Pay attention to the advice of people who have built similar projects.
- Know your teams strengths and weaknesses.

- Identify a team member to lead the group, even if the group has a democratic team structure.
- Gain an early understanding of the development tools and delivery environment.
- Team cooperation is important and everyone needs to same level of commitment.
- Monitor team members to ensure the completion of assigned tasks.

From the instructor's perspective, one important aspect of our approach to using serious games is to insist that students receive an acceptance letter from the client to receive credit for the course. This reinforces the importance of agreeing early on the project scope and creating testable requirements for the serious game. Delivery of milestone documents (requirements, project plan, software quality, risk management, design, and testing) at the same time or before game prototypes are delivered also helps prevent students from coding first and documenting later. Students should justify technology decisions and game feature decisions by considering (and documenting) the cost and benefit analysis for each alternative, rather than just including a feature that seems cool.

Traditional software engineering documents are similar in structure to those used in the game design industry. Our students find that it difficult to use a design document template that might be useful for a project involving the creation of a form fill-in database application for a serious game containing a real-time graphical user interface. Part of the reason for this difficulty, is that game play, user motivations, and game asset specifications do not usually appear in a traditional software engineering design document. Some game developers do UML modeling as part of their analysis model, but many do not. Our students are required to create a UML model for their first game prototype but rarely revise it as new prototypes evolve.

Instructors wishing to make use of serious games in their teaching need to identify sources for art assets and game programming environments for their students early in the design process. If the goal of the capstone course is to allow students to practice software engineering and project management, then it is not reasonable to have software engineering students create art. The art development time on most projects is at least sizable as the programming time. One strategy to avoid this is to form alliances with instructors in college level art programs, as we have done.

Having students develop their game projects from scratch will teach them a lot about low level C++ or C# graphics programming, but will not leave much time for game design or software engineering activities. Using an existing game engine such as the Torque Game Engine will reduce the level of programming to that which students can accomplish during a typical eight-month capstone design experience.

Summary

Table 3 contains the average scores for course evaluation items completed by all students for the past three fall semester offerings of the senior design experience (4 = completely agree, 0= completely disagree) and the overall course rating (4=highest, 0=lowest). The course evaluation forms for each course indicate above average levels of satisfaction on the part of the students.

Table 3: Student Course Evaluations for CIS 4962

	Fall 2005	Fall 2006	Fall 2007
Course fulfilled my needs	3.5	4.0	3.8
Course objectives were clear	3.7	3.7	3.8
Course prerequisites are adequate	3.8	3.7	3.6
Course was challenging and interesting	3.7	3.7	3.8
Overall course rating	3.5	3.8	3.6

The number of serious game projects that our senior design students have completed is too small in number to allow for meaningful statistical analyses. Students submit course evaluation forms anonymously so it is not possible to separate them by project type. It is the author's observation that students are willing to spend many hours engaged in both the thoughtful design and careful implementation of components for serious game projects. It is interesting to note that students working on serious game projects at our institution have often continued working on them for several months after delivering them to their clients. The author has supervised over two hundred traditional senior design projects and it is extremely rare when students elect to continue working on these projects after receiving their client acceptance letter.

The number of requests for this type of senior design project is increasing as more people gain knowledge of our game design courses and research activities. It is our experience that the types of entertainment games, that our students produce during our game design courses, do not always lend themselves to the deliberate and controlled development that software engineering demands. In part, this may be true because the requirements for these games in our courses are too soft or may not written down at all. Serious game requests seem to come from clients who have specific agendas for the use of the games within their organizations. These clients often have very firm delivery deadlines and computing resource constraints. The same may be true for entertainment games requested by external clients, but we have not done any project work like this in our capstone design courses yet.

Bibliography

1. Bond B. The Difficult Part of Capstone Design Courses, *Proceedings of 25th Annual Frontiers in Education Conference*, (Vol. 1, Nov 1995), IEEE Press, Los Alamitos, CA, 1995, pp. 2c31-2c34.
2. Bruhn, R. and Carp, J. Capstone Courses Creates Useful Business Products and Corporate Ready Students, *SIGCSE Bulletin* (Vol. 36 No. 2), ACM Press, New York, NY, June 2004, pp. 260-264.
3. Heitman G. and Manseur, R. Organization of a Capstone Design Course, *Proceedings of 30th Annual Frontiers in Education Conference* (Vol. 1, Oct 2000), IEEE Press, Los Alamitos, CA, 2000, pp. 4c11-4c15.
4. Maxim, B. R. and Akingbehin, K. Contemporary Software Development Trends, *Proceedings of the Association of Management 17th International Conference on Computer Science* (Vol. 17 August 1999), 1999, pp. 141-146.
5. Maxim, B. R.; Akingbehin, K.; and Tsui, L. A Capstone Design Course Based on Computing Curricula 1991, *Computer Science Education*, (Vol. 5 1994), pp. 229-240.

6. Parberry, I., Roden, T., and Kazenzadeh, M. Experience with an Industry-Driven Capstone Course on Game Programming, an Extended Abstract, *Proceedings of 36th SIGCSE Technical Symposium* (St. Louis, MO, February, 2005), ACM Press, New York, NY, 2005, pp. 91-96.

7. Ray J. Industry Academic Partnerships for Successful Capstone Projects, *Proceedings of 33rd Annual Frontiers in Education Conference* (Vol. 3, Nov 2003), IEEE Press, Los Alamitos, CA, 2003, pp. 3s2b24-3s2b29.

8. Rover D. Perspectives on Learning in a Capstone Design Course, *Proceedings of 30th Annual Frontiers in Education Conference* (Vol 2. 1, Oct 2000), IEEE Press, Los Alamitos, CA, 2000, pp. f4c14-f4c19.

9. Jones, R. Design and implementation of a computer games: a capstone course for undergraduate computer science education. In *Proceedings of 31st SIGCSE Technical Symposium* (Austin, TX, March, 2000), ACM Press, New York, NY, 2000, 260-264.

10. Spinello, R. and Tavini, T. *Readings in Cyber Ethics*, Jones and Bartlett, Sudbury, MA, 2004.

11. Maxim, B. R. Game design: games for and the World Wide Web, *The Internet Encyclopedia*, Wiley, Hoboken, NJ, 2004.

12. Claypool, K. and Claypool, M. Software engineering design: teaching software engineering through game design, *Proceedings of 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Capprica, Portugal, June, 2005), ACM Press, New York, NY, 2005, 123-127.

13. Rabin, S. *Introduction to Game Development*, Charles Rivers Media, Hingham, MA, 2005.

14. Rollings, A. and Morris, D. *Game Architecture and Design*, New Riders, Indianapolis, IN, 2004.

15. Rouse, R. *Game Design: Theory and Practice*, Wordware, Plano, TX, 2001.

16. *International Game Developers Association Curriculum Framework*, <http://www.igda.org/academia/curriculum_framework.php> (6 September 2005)

17. Maxim, B. R., *Software Requirements Analysis and Design*, NIIT, Atlanta, GA 2004.

18. Michael, D. and Chen, S. *Serious Games: Games that Educate, Train, and Inform*, Thomson Course Technology, Indianapolis, IN, 2006.

19. Maxim, B. R. and Akingbehin, K. Experiences in Teaching Senior Design Using Real-World Clients, *Proceedings of the 2006 IEEE Frontiers in Education Conference* (vol. 1, 2007), San Jose, CA, October 2006: T2H13-T2h17.

20. Blow, J. Game development is harder than you think, *Queue* (Vol. 1, No. 10, Feb 2004), ACM Press, NY, pp.29-37.