Use of Teams in Game Design and Software Engineering Capstone Project Classes

Bruce R. Maxim[1], Margaret Turton[2]

[1]Computer and Information Science, University of Michigan-Dearborn, Dearborn, Michigan, USA

[2]Information Systems, LaTrobe Melbourne, Australia

[1]bmaxim@umich.edu; [2]mt0003@optusnet.com.au

***Abstract*- This paper describes experiences teaching software engineering project courses at the University of Michigan-Dearborn during the past seventeen years. Modern game development involves significant software engineering effort. Students in these courses are required to 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 or academic clients. Perhaps one of the best ways to teach the importance of managing project resources is to allow students to manage real projects with serious development constraints including concrete deadlines. To improve students' verbal and written communications skills and experience in teamwork and cooperative design projects, students are required to present frequent written and verbal reports as project milestones are completed. Final cumulative written reports and oral presentations are required of all teams at UM-Dearborn.**

Keywords- Software Engineering; Game Design; Project Management; Interdisciplinary Teams

I. INTRODUCTION

The idea of project courses for undergraduate computing majors is not new. Capstone courses in computing have traditionally tried to provide senior students with experiences similar to that encountered in professional practice [1-6]. In several cases, course developers make a case that the purpose of such a course is to help the students integrate theoretical computing concepts with the demands of computing practice. Real-world capstone design projects can be used to meet the expectations of ABET [5, 7].

One approach in recent years has been to involve students in projects that satisfy the needs of real-world clients. 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 course as a two-semester course sequence or have offered the course outside of the normal academic year [8]. However, with careful project selection, some schools have been successful in offering a senior design experience as a one- semester course [9, 10].

Some of the project courses described in earlier literature do not require students to work as part of a development team. More recently, a majority of project courses described in recent years, require students to work as a group as a major course objective as a practical lesson in team dynamics [4, 6, 8, 11, 12]. 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. Peer evaluation is an important aspect of team projects to reduce the likelihood of some students coasting during the project [13].

The development of computer games is labor-intensive [14, 16]. Game developers rarely build computer games on their own, as they did 12 years ago [17]. 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 [18]. It is difficult for students to comprehend the benefits and logistical problems of working on interdisciplinary teams by simply reading textbooks and yet, many game design courses described in the literature emphasize the use of game projects implemented by single developers [3, 16].

There is consensus among members of the Computer and Information Science (CIS) department's professional advisory board that professional practice invariably requires strong verbal and written communication skills. Robillard and Lavallee write that communications breakdowns often result in project failures [15]. Despite this, many computing curricula fail to provide graduates with adequate communication skills [12]. 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 team-based software development encourage students to improve their written and oral communication skills [9, 11].

There are some classic works that address the process of helping teams to jell. DeMarco and Lister write that jelled teams have a common definition of success and an identifiable team spirit. Jelled teams are significantly more productive than non-jelled teams [19]. Tuckman observes that high performing teams go though four phases of development (forming, storming, norming, performing) [20]. It is important to allow student teams to pass through these phases. Jackman [21] lists several conditions to be avoided to prevent team toxicity: frenzied work atmosphere, high levels of frustration, poorly coordinated software process, unclear team role definitions, and repeated exposure to failure. Student teams need to be coached to avoid these conditions.

It is difficult to motivate students to focus on the non-

technical aspects of software development if the final software product is the only artifact assessed by the instructor. By working on team projects students can see the importance of having real-world problem solvers complement their technical skills with knowledge of project management skills. Possibly one of the best ways to teach the importance of managing project resources and team dynamics is to allow students to manage a real-world project with serious development constraints including concrete deadlines [1]. This paper summarizes the content of several CIS project courses and the development professional skills using team-based projects.

II. COURSES

The authors have created and taught a number of project courses at the undergraduate level. This paper will focus on the sequence of project activities taken by game design and software engineering students at the University of Michigan-Dearborn. Students elect these classes after completing a junior level software engineering class where they learn about software process, requirements modelling, software design, cost estimation, and project scheduling. A two semester course sequence (Computer Game Design and Implementation I and II, CIS 487 and CIS 488 respectively) focuses on the application of software engineering principles in the development of computer games are taken by third year students. Some students may take CIS 487 and the junior level software engineering class at the same time. The CIS capstone design experience is organized two semester course sequence (CIS 4961 and CIS 4962) which senior level students complete over two semesters. The majority of students taking the senior design course sequence complete projects for off campus clients as part of this experience. Many of these clients are members of the CIS Professional Advisory Board or CIS Alumni.

A. Game Design Courses

Game Design 1 deals with the study of the technologies involved in the creation of computer games. The focus of this course is on the application of software engineering methods in the hands-on development of computer games. Students study a variety of software technologies relevant to computer game design, including: simulation and modeling, computer graphics, artificial intelligence (AI), game theory, software engineering, human computer interaction, graphic design, game aesthetics and multi-media system design.

The student work for this course includes the completion of two game projects. All projects include design activities and students use existing programming tools. Using existing programming tools and libraries allows students to focus on software engineering design rather writing all source code from scratch. The first project involves using a two person team to create a 2D game. The second project requires a four person team to work through all phases of the software life cycle: specification, design, implementation, testing, and evaluation in the creation of a 3D multimedia game. Several students play test each game product and its design to provide constructive feedback to the authors regarding its overall quality during its development. Play testing is a process where a game is tested using the perspective of the intended game player. The intent is to uncover usability issues, technical issues affecting game play, and places where the game fails to be entertaining.

Game Design 2 focuses on the use of team software engineering process in development of computer games and the use of game development tools (e.g. game engines). Students study a variety of software technologies relevant to computer game design, including: 3D graphics, computer animation, data-driven game design, game AI, game theory, software engineering, and game content development. One important aspect of this course is managing the process of outsourcing game asset creation to art students attending college in another city. Game assets are the artistic elements of a game (graphics, animations, sound effects, music). The term-long project for this course requires each student to participate as a member of a multi-disciplinary team to develop a 3D multi-player computer game.

B. Capstone Courses

Students enroll in Senior Design 1 after they complete their required software engineering courses (Software Engineering 1, Software Engineering 2, Design and Architecture Patterns). Capstone projects generally require approximately 500 hours of student effort to complete. The major activities in Senior Design 1 are requirements gathering and project planning (including risk management and quality assurance efforts). The major activities in Senior Design 2 are product design, implementation, testing, and product delivery. 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 are required to work in four person software development teams for a period of eight months. In exceptional cases and with proper justification, larger group sizes are permissible. Project clients are usually non-profit or for profit groups off campus. Students select their own teammates and determine their own plan for rotating team leadership. Students are free to determine their own team organization. The use of external clients provides students with real-world experience and improves their communications skills.

III. USE OF TEAMS

The use of teams is an integral part of the learning activities in these courses. The authors have observed that it takes time for student management and collaboration skills to develop the level required to complete a capstone project. The details of the team activities are described below.

It is important to note that with each milestone submission in these classes, students evaluate the quality of their own participation during this phase of the project and that of each of their teammates. Students provide numerical ratings (0=none, 5=great) for each team member and provide a list of tasks accomplished to justify their ratings. Students are required to conduct a project post mortem at the end of each project and provide individual summaries of their

perceptions of how things went for their team. Students may submit this information anonymously to the instructor.

A. Game Design

Game Design 1 students self-organize into teams of two and negotiate the approval of a 2D game idea for Project 1. Each team of students prepares a game pitch and conducts a structured walk-through in front of a group of peer reviewers. The team has the opportunity to revise their game pitch before submitting it for grading. The final 2D game produced by each team is play-tested by at least eight students during the Game Design 1 2D project fair.

Game Design 1 students self-organize into teams of four and negotiate the approval of a 3D game idea. Each team of students prepares a game design treatment (initial design document) and conducts a structured walk-through in front of a team of peer reviewers. The team has the opportunity to revise their design treatment before submitting it for grading. The final 3D multimedia game produced by each team is play-tested by at least eight students during the Game Design 1 3D project fair.

The major project in Game Design 2 involves the design and implementation of a 3D multimedia game by an interdisciplinary team. Ideally the student teams are composed of software engineering students from the UM-Dearborn and digital art students from a nearby college.

Before forming teams in Game Design 2, each student participates in a game pitch process. During the one-week pitch process, each student creates a short game pitch document for an original 3D game and presents it to the class. The class assesses each game for feasibility and game play potential. The instructor and students consult with each other in selecting games for further development.

Students organize themselves into four to six member teams consisting of software engineers and game asset developers. Students are free to structure their teams using any model they wish.

Students have three weeks to refine their game requirements and create a design treatment for the first prototype of their 3D game. The game asset creators develop storyboards and concept art for this document. The game teams use feedback from peer reviewers following a structured walk-through to revise their preliminary design documents.

The second milestone activity for each team is to create an alpha release prototype for their game. Teams have four weeks to create UML models for their complete games and create working game prototypes using specialized game development tools. The game asset creators develop placeholder art and develop the initial level design layout. A formal technical walk-through of the UML model is part of the peer review process. Students play test the game prototypes. The teams use reviewer feedback to revise their design documents.

The next milestone activity for each team is to develop a beta release prototype for their game. Teams have four weeks to develop the requirements for an intelligent agent or NPC (non-playing character) to add to their game. This is introduced in the context of a requirements change made by the instructor playing the role of client. The implementation of the intelligent agent becomes part of an incremental release of the game product. A revision of their design document reflects this change and regression testing occurs to ensure that implementation of the intelligent agent has not broken the game. The game asset creators develop the art and audio assets to near production release quality. Students outside the development team play-test the new prototype. A formal technical walk-through of the design document is part of the peer review process. The team uses reviewer feedback to revise the game design.

The final milestone activity is to complete a gold game release prototype. Teams have three weeks to deliver the final design document for a 3D multi-media computer game and the final game software. Several students play test each of the final game products and provide constructive feedback regarding the overall quality of each to the development teams. The gold release prototype requires the creation of a marketing piece to accompany its public debut at the 3D Game launch festival.

B. Capstone

During the first two weeks of Senior Design 1, 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 brief progress reports, a structured walk-through of a work product, or a product demonstration.

In addition to the two hours of class-time each week, students complete several hours of work on their project out-of-class time. 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. Students are expected to share their milestone artifacts with their clients and obtain their feedback frequently as their project evolves. The time spent outside of class is very important as a means of fostering team development and extremely important given the size of the projects.

The role of the instructor in this course is that of a coach or mentor not 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.

Senior Design 2, continues the project work begun in Senior Design 1. Teams continue without change. If team

members fail to elect the course, the teams need to determine how to compensate for the lost student effort. Should this situation occur, the teams will rely on their project management plan and activate their risk management scenarios to create during Senior Design 1. Students must present a letter of acceptance from their client to the instructor, indicating the client's acceptance of the final product, in order to receive a grade for Senior Design 2. 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.

A final presentation is required of all teams at the end of the Senior Design 2 and includes a product demonstration and report. The final project presentation is very important as a vehicle for assessing oral communication skills. The project presentation requires the use of audiovisual support.

IV. LESSONS LEARNED

The anecdotal comments appearing in this section of the paper come from the authors' experiences supervising student teams in the completion of several hundred projects over the past 16 years. Approximately 85% of the senior design projects have off-campus clients. Whereas very few of the game projects completed prior to senior design are undertaken for off-campus clients.

Students are not allowed to do routine maintenance work on existing systems or recreate existing games. Students are expected to have the opportunity to propose design trade-offs to the client during requirements gathering. Potential clients have several concerns about working with students, who are not employees. Clients are concerned with the ownership of the resulting software products and have some privacy concerns that the instructor must address before they are willing to agree to provide project opportunities. Students may be asked to sign non-disclosure agreements (NDAs) with their clients, which is a common industry practice for software engineering contractors.

Clients require assurances that they will receive complete products before students receive grades for the course. It has been our experience that real-world clients have real deadlines and firm product expectations. It is helpful to have students compute budget estimates for their projects as if the work were being done on a for fee basis. This provides students with the process of preparing request for proposal (RFP) and request for quotation (RFQ) documents. Sharing this information with the client is an educational experience for both students and clients.

Requiring students to obtain a letter of acceptance from their client reinforces the importance of client communication and meeting the client's needs. This also allows clients to feel they have some input into the student evaluation process. Many external clients like to feel that they are contributing to the education of future employees.

For courses with one-semester projects, teams need to complete requirements gathering in a timely manner or the whole project gets behind schedule and may fail altogether. Single term projects seem to work best without involving an external client. Similarly teams with uneven commitments to completing the project will have trouble completing projects with short time lines.

We believe that all students need to experience the role of team leader before graduation. In capstone courses, students self-select their projects from a list provided by the instructor and self-organize into teams. Student teams are free to create their own group structures, but their project management plan must include a scheme for rotating the team leadership position. This seems to give students a stronger sense of ownership of the project and helps to build group cohesion. When instructors assign students to teams or appoint team leaders it too easy for students to blame the instructor for team dynamics problems. Students learn very early in our course that it is their project to complete and not the instructor's. We believe that allowing students to take early project ownership can help reduce frustration when things do not go well. Encouraging proactive risk management can also help reduce team member frustrations.

Student team sizes of four seem to work the best for the types of projects offered by our pool of external clients. Game projects sometimes need additional team members to assist with art asset creation. Teams with fewer than four students do not have sufficient software engineering resources to cover the full set of development roles needed to complete the project. Teams having more than four software engineering students may be difficult for novice team leaders to manage and schedule effectively. This can help avoid the problem of having teams with unclear team role definitions.

Modern software engineering practices require document development early in the project life cycle and making changes as the client's needs evolve. Even agile teams are expected to let their documentation evolve as the as the new user stories are implemented and integrated into the build. Waiting until the end of the project, makes it difficult for the students to write a good final report and receive client acceptance for the final system in a timely manner. Evolving the documents as the project proceeds helps to improve the coordination of the software process activities.

Writing and presenting draft documents as project milestones provides valuable opportunities for students to get feedback from their clients, peers, and instructors. This feedback helps to improve the final products as well as the documents themselves. Starting the process of writing documents earlier in the course makes it easier to complete the final report in a timely manner. Creating a formal planning document early in the course, can help reduce the likelihood of frenzied work atmosphere developing, by making sure the project scope is appropriate to the available time and resources.

It is desirable to allow students to control all client contact once the project begins. If the instructor buffers client communication, students will not learn how to manage change requests that can increase the scope of the project. Similarly, students learn how to negotiate reductions in scope with the client and experience its consequences first

hand. This always drives home the importance of satisfying the client's needs as the central goal for the project.

To insure smooth team operation during the project, students evaluate their personal effort and the efforts of their team members as each milestone is completed. Students rate each team member's anonymously effort using the value in the range of 0 to 5 and describe the tasks completed by each. This allows the instructor to detect team problems early. The specter of peer evaluation is often enough to make sure that students complete assigned tasks in a timely manner. Learning to evaluate people's performance objectively is another benefit of this practice.

Students learn to appreciate the importance of using software production tools during this course. Student teams discover that making use of version control repositories is necessary to allow team members to work on project artifacts independently and merging them seamlessly at the appropriate times. Student teams that have access to their client's development environment have the least difficulties in delivering a software product and obtaining a customer acceptance letter.

V. STUDENT POSTMORTEMS

Examination of the student post mortem documents provides some insight into how well teams worked in meeting the goals of each class. Students in Game Design 1 made the following observations:

- Incremental implementation is hard without the existence of a complete code design;
- Proper planning is key to avoid adding new and incompatible ideas to evolving game increment;
- Creating game assets interferes with game programming and design;
- Parallel development of game elements is impossible without the use of teams;
- It takes some time and effort to learn how to collaborate with another programmer;
- Team development is easier if everyone is following the same vision;
- Good coordination and communication is key to the success of any project;
- E-mail and instant messenger communication is not always as good as face to meetings;
- It is hard to use code written by someone else especially if it is poorly documented;

Code design for reuse is an essential part of game development.

The Game Design 2 students made the following observations about teamwork in their post mortem assessments:

- Game design is improved by having a diverse group of people working on the team;
- Role specialization is helpful on large projects to void the "jack of all trades, master of none" phenomenon;
- Roles need to be determined and assigned early in the project;
- Project leaders need to be assertive in delegating tasks and making sure workload is distributed evenly;
- E-mail is not sufficient as the only means of communication among team members;
- It is easier to blow off digital meetings than face to face meetings;
- Weekly team meetings and progress reports are essential;
- It helps to know what other team members are working on to anticipate interfacing issues;
- Software version control repositories are invaluable in building large software products;
- Maintaining multiple copies of all project artifacts (code, art, documents, etc.) in diverse locations is essential;

Maintaining a log of problems and fixes can be a valuable project resource.

The Senior Designs 1 and 2 students made the following observations about teamwork in their post mortem assessments:

- An identifiable team leader is needed;
- Formal source code version control tools need to be used, not Google Groups;
- Important to keep the game requirements up to date during project to ensure smooth negotiation of the client acceptance letter;
- Start early even on things that seem simple;
- Get early client involvement by giving them prototypes to review;
- Game developers need to be serious about the formal technical review process;
- Time management is key to success on large projects;
- Testing takes more time than one thinks;
- Keep requirements specifications up to date at all times and have client review them;
- Keep team mates updated on all work completed, have mandatory weekly status meetings;
- Know your team member's strengths and weakness;
- Trust your team mates to deliver what they promise and accept the result;
- Monitor team members to ensure completion of assigned tasks;
- Write down plans and commitments to avoid misunderstandings;
- Function creep as project the proceeds must be tracked;
- Duplication of game asset work needs to be prevented, better communication is needed.

VI. CONCLUSIONS

The student projects and design activities that result from these courses receive frequent praise from local computing professionals and accrediting agency reviewers. The Game Design students present their work in community showcases sponsored the CIS department. Senior Design 2 projects have earned the prize for best senior design project in the annual College of Engineering and Computer Science senior design competition eight times during the past sixteen years, the latest being 2012. Many students have received job offers from employers after showing their project portfolios during the interview process.

It is interesting that students in each of these courses seem to be learning similar lessons about the importance of communication, planning, and collaboration. Students learn from their mistakes if they are working in a supportive environment. Students look forward to team projects once they have experiences success using teams. Their prospective employers appreciate the fact that the students have learned the importance of teamwork.

REFERENCES

[1] G. Heitman and R. Manseur, "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. 4cl1-4cl5.

[2] B. Maxim and K. Akingbehin, "Contemporary Software Development Trends", Proceedings of the Association of Management 17th International Conference on Computer Science (Vol. 17 August 1999), 1999, pp. 141-146.

[3] I. Parberry, T. Roden, and M. Kazenzadeh, "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.

[4] D. Rover, "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.

[5] G. Altuger and C. Chassapis, "Work in Progress – Preparing Students for Lifelong Learning in a Capstone Environment", Proceedings of 40th Annual Frontiers in Education Conference (Vol. 1, Oct 2010), IEEE Press, Washington, DC, 2010, pp. T2J1-T2J2.

[6] C. Dean, T. Lynch, and R. Ramnath, "Student Perspectives on Learning Through Developing Software for the Real World", Proceedings of 41st Annual Frontiers in Education Conference (Vol. 1, Oct 2011), IEEE Press, Rapid City, SD, 2011, pp. T3F1-T3F6.

[7] J. Ray, "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] R. Bruhn and J. Carp, "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.

[9] B. Maxim, K. Akingbehin, and L. Tsui, "A Capstone Design Course Based on Computing Curricula 1991", Computer Science Education, (Vol. 5 1994), pp. 229-240.

[10] C. Rudd, and V. Delevear, "Developing and Conducting an Industry Based Capstone Design Course", Proceedings of 27th Annual Frontiers in Education Conference (Vol. 1, Nov 1997), IEEE Press, Los Alamitos, CA, 1997, pp. 644-647.

[11] B. Bond, "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.

[12] M. Ardis, S, Chenoweth, and F. Young, "The 'Soft' Topics in Software Engineering Education", Proceedings of 38th Annual Frontiers in Education Conference (Vol. 1, Oct 2008), IEEE Press, Saratoga Springs, NY, 2008, pp. F3H1-F3H6.

[13] J. Wang, P. Imbrie, and J. L, "Work in Progress – A Feedback System for Peer Evaluation of Engineering Student Teams to Enhance Team Effectiveness", Proceedings of 41st Annual Frontiers in Education Conference (Vol. 1, Oct 2011), IEEE Press, Washington, DC, 2011, pp. S4C1-S4C5.

[14] B. Maxim, "Game design: games for and the World Wide Web", The Internet Encyclopaedia, Wiley, Hoboken, NJ, 2004.

[15] P. Robillard and M. Lavallee, "Software Team Processes: A Taxonomy", Proceedings of the International Conference on Software and System Process (ICSSP 2012), IEEE Press, Zurich, Switzerland, 2012, pp. 101-109.

[16] K. Claypool, and M. Claypool, "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.

[17] R. Rouse, Game Design: Theory and Practice, Wordware, Plano, TX, 2001.

[18] B. Maxim, Software Requirements Analysis and Design, NIIT, Atlanta, GA 2004.

[19] T. DeMarco and T. Lister, Peopleware, 2nd ed., Dorsett House, 1998.

[20] B. Tuckman, "Developmental Sequence in Small Groups", Psychological Bulletin, (Vol. 63 No. 6), pp. 384-399.

[21] M. Jackman, "Homeopathic Remedies for Team Toxicity", IEEE Software, (Vol. 15 No. 3, July/August 1998), pp. 43-45.

Bruce R. Maxim received his PhD from the University of Michigan, USA, in 1982. He is currently Associate Professor of Computer and Information Science, University of Michigan-Dearborn, USA. His research interests include software engineering, game design, artificial intelligence, and computer science education. He is the designer of the courses described in this paper.

Margaret Turton received her Master of Business (eBusiness and Communication) from Swinburne University of Technology, Australia in 2004. She is currently Lecturer in Information Systems, La Trobe Melbourne – Melbourne, Australia. Her research interests include, flexible learning in higher education, game simulations for learning, user experience design, and distance education. She has worked with Dr. Maxim and his software engineering students for the past two years.