AI Projects for Computer Science Capstone Classes (Extended Abstract)

Matthew E. Taylor, Sakire Arslan Ay
{taylorm,arslanay}@eecs.wsu.edu
School of Electrical Engineering and Computer Science
Washington State University

Introduction

Capstone senior design projects provide students with a collaborative software design and development experience to reinforce learned material while allowing students latitude in developing real-world applications. Our two-semester capstone classes are required for all computer science majors. Students must have completed a software engineering course — capstone classes are typically taken during their last two semesters. Project proposals come from a variety of sources, including industry, WSU faculty (from our own and other departments), local agencies, and entrepreneurs. We have recently targeted projects in AI — although students typically have little background, they find the ideas and methods compelling. This paper outlines our instructional approach and reports our experiences with three projects.

Course Description

Several weeks before the start of the semester, the instructor solicits project topics and mentors, favoring projects that 1) involve large scale software development, 2) encourage students to learn about new areas, and 3) are interdisciplinary. Ideally, the project will have an industry mentor.

Initiation (2 weeks): Matching students with projects is done during the first week of the semester and takes into account student preferences, prior coursework, and experience.

Planning (3 weeks): Teams follow an agile software development blended with certain aspects of linear Waterfall model. Software development starts with a short planning phase followed by 2–3 week-long sprints (similar to Scrum), where each sprint encompasses all of the typical stages of software development for a small set of software features.

Development (20 weeks): Teams meet weekly with the instructor and the mentors. Each sprint typically involves some planning, design, implementation, integration, and testing. At the end of each sprint, the teams provide brief demos. The prototypes are demonstrated at the end of the first semester and the middle of the second semester.

Termination (3 weeks): The project terminates with a final report and demonstrations of final prototype to team mentors and to the public in the form of a poster/demo presentation.

Projects are judged in a multi-level competition on originality, compliance with software engineering principles, completion, and their successful deployment.

Assessment

Student performance is assessed on multiple graded items. Writing assignments are graded per team. Teams present their progress weekly to the instructor and project mentor. Meetings are highly effective in enforcing the agile methodology and keep projects on track. At the end of the course, each team prepares and presents a poster. Project posters are judged based on the quality, content, design, and presentation. During the poster session, the teams interact with several different audiences including industry representatives, faculty, and students. This is a valuable experience for students: they must pitch their project in multiple ways and they receive a substantial amount of feedback. During the poster session, the instructor assesses the students' soft skills in presenting their work and their interactions with various audiences. Lastly, students also provide formal *peer review*, in accordance with *cooperative learning* principles.

Example Projects

The *Gamification in Classroom Settings* project built a multiplayer game targeted for educational use. The main novelty of the project was to produce a content independent game — rather than customizing the game to a particular set of skills or class, build a game that could run alongside multiple classes with minimal changes. The team worked with a graduate student to explore multiple ways to leverage games to improve student engagement in introductory classes, giving players in-game bonuses for homework submission, grade improvement, attendance, etc. The team helped to deploy the game in a large introductory computer science class while collecting statistics. The game is currently being improved by a new group of students. Data mining techniques are used to better understand the trends and player preferences associated with the game. The students are incorporating sub-games from genres into the overall framework — goal is to predict which types of games will be played most by different types of students. The collected data will assist with a PhD student's dissertation (Cain *et al.* 2016). Based on the current analysis, there was not a signifi-

cant correlation between player usage and MBTI types (Myers and Briggs 1962). We hope game-specific typologies (e.g., BrainHex (Nacke *et al.* 2014)) will be a better indicators of game preferences.

The *Network Visualization & Anomaly Detection* tool analyzed raw network data to detect suspicious behaviors and attacks. The tool leveraged a virtual network-monitoring platform from the company ExtraHop for acquiring information about network packets. It continuously analyzed the network data and extracted features such as client location, destination IP, time of connection, connection protocol, and the number of similar connections and stored the extracted information in a NoSQL database. The tool then leveraged an SVM classifier, trained over set of data to detect anomalies in the network. The tool provided a web-based interface visualizing the network behavior and highlighting the anomalous connections. The tool allowed users to monitor the network traffic, list the anomalous connections, and display the details of the anomalous behaviors.

The *Semi-Autonomous Wheelchair* project aimed to improve the safety and usability of electric wheelchairs for people with ALS or other neurodegenerative diseases that leave them unable to use typical interfaces. Additionally, traditional wheelchairs have no obstacle avoidance — electric wheelchairs can move quickly but weigh over 300 pounds without an occupant. The multi-year project built a prototype system that assisted the user while driving the chair, autonomously detecting and avoiding obstacles, and moving through doorways (using a Kinect). A hardware interface enabled programmatic control of the wheelchair while receiving inputs from multiple devices, including a keyboard, an eye-gaze sensor, and an Xbox controller. The GUI allowed a user to direct the chair and displayed the environmental map, Kinect's view, and obstacles detected.

One of the many challenges was doorway detection. The first, a Haar cascade classifier (Viola and Jones 2001), is built into OpenCV (Bradski 2000) and while it performed consistently in different lighting conditions, overall it had a high number of false positives. The second, a color based detector, is a more precise approach when door colors and lighting are consistent. The goal of this hand-designed method was to find the contours of all possible targets within a certain color range and worked best on closed doors. The third, a depth-based detector, is most useful to tell if a door is open or closed and could easily be combined with other door detection methods. The fourth, an infrared-based detector, relies on environmental modification and was the most successful. By placing tape that is reflective in the infrared spectrum around the doorways, the Kinect's infrared stream can easily detect all doors. Students reported the project was particularly motivating because they could demo their product to local users with ALS and write a successful workshop paper (Xu *et al.* 2016).

Discussion and Lessons Learned

Most seniors have heavy course loads and they tend to delay senior design work when they get busy. We found that weekly progress reports ensure students make some progress each week and remain motivated. In addition, peer review helps to motivate students to contribute and identify underperformers. Finally, reminding the students that they will have to present their work in a poster session to faculty, mentors, and peers helps to keep the team accountable.

AI projects require more attention from the instructor and mentor, relative to other topics. When the teams are formed, the students who completed (or co-enrolled) in introductory level AI, machine learning, and robotics courses are preferred. In addition, the students who are involved with the robotics club and who have hands-on/programming experience in robotics and/or machine learning are favored. Most students working on the AI projects were familiar with various learning paradigms and SVMs when they started their projects. In addition, the instructor and the mentors made sure that the project's scope and milestones were tightly defined — because there are multiple ways to approach complex AI problems, students may struggle to focus and fall behind while investigating potential solutions. Teams first implemented simple algorithms and only later handled more disparate or challenging use cases. The agile process allowed them to iteratively apply newly learned approaches and validate performance, reducing the risk of failure.

We found testing machine learning algorithms particularly difficult when there was no (well-defined) ground truth. This was a major issue for the Network Visualization & Anomaly Detection project and students struggled extensively in verifying the accuracy of their solution. Interdisciplinary projects have their own challenges and therefore need to be managed carefully. All parts of the systems (hardware, mechanical, and software) need to be unit-tested and integration tests need to be performed continuously — it is very difficult to diagnose software problems without a working hardware platform. Lastly, it is critical for teams to follow strict coding standards and provide documentation, particularly for multi-year projects.

Acknowledgments

The authors would like to thank Eric Eaton and the anonymous reviewers for their suggestions and comments. This research has taken place in part in the Intelligent Robot Learning Lab, which is supported in part by grants from NASA NNX16CD07C, NSF IIS-1149917, NSF IIS-1643614, and USDA 2014-67021-22174.

References

G. Bradski. The OpenCV reference manual. *Dr. Dobb's Journal of Software Tools*, 2000.

C. Cain, A. Anderson, and M. E. Taylor. Content-Independent Classroom Gamification. In *Proc. of the ASEE*, 2016.

I.B. Myers and K.C. Briggs. *The Myers-Briggs Type Indicator: Manual (1962)*. Consulting Psychologists Press, 1962.

L. E. Nacke, C. Bateman, and R. L. Mandryk. Brainhex: A neurobiological gamer typology survey. *Entertain. Comput.*, 5(1), 2014.

K. Seaborn and D. I. Fels. Gamification in theory and action. *Int. J. Human-Computer Studies*, 74(C):14–31, 2015.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of CVPR*, 2001.

R. Xu, R. Hartshorn, R. Huard, J. Irwin, K. Johnson, G. Nelson, J. Campbell, S. Arslan Ay, and M. E. Taylor. Towards a Semi-Autonomous Wheelchair for Users with ALS. In *IJCAI Workshop on Autonomous Mobile Service Robots*, 2016.