CubeSat: A Multidisciplinary Senior Design Project

Dr. Adam Kaplan, California State University, Northridge

Adam Kaplan received a BS degree in Computer Science and Engineering from UCLA in the year 2000, and worked in the burgeoning internet software industry for a year and a half before returning to UCLA for a Masters (2003) and PhD (2008) in Computer Science. Since 2008, Adam has lectured at UCLA, UCLA Extension, and California State University, Dominguez Hills, while also working as a software engineer for local startup businesses IEnteractive and Perceptive Development. In the year 2012, he joined California State University, Northridge as an Assistant Professor. His research interests include assistive technology for victims of acute aphasia, the evolving cost models of cloud services, and the development of power and cost-efficient embedded and mobile software.

Mr. James Flynn, California State University, Northridge

James Flynn is a part time faculty member in the Department of Electrical and Computer Engineering at California State University, Northridge (CSUN). He holds a B.S. (1977) degree in Electrical Engineering from the Illinois Institute of Technology and a Master of Fine Arts (1981) degree from Northwestern University. He is owner of a consulting firm specializing in electronics for television and film production. Currently, he is the system architect on a faculty-student team engaged in the development and construction of a CubeSat to be launched in 2015. He has also been active in developing education tools involving software defined radio (SDR).

Dr. Sharlene Katz P.E., California State University, Northridge

Sharlene Katz is a Professor in the Department of Electrical and Computer Engineering at California State University, Northridge (CSUN) where she has been for over 25 years. She graduated from the University of California, Los Angeles with B.S. (1975), M.S. (1976), and Ph.D. (1986) degrees in Electrical Engineering. Currently, she is principle investigator on a faculty-student team engaged in the development and construction of a CubeSat to be launched in 2015. Her other areas of research interest have been in engineering education techniques, software defined radio, and neural networks. Dr. Katz is a licensed professional engineer in the state of California.

CubeSat: A Multidisciplinary Senior Design Project

Abstract

Engineering and computer science programs often require a culminating senior design project. Several of the Accreditation Board for Engineering and Technology (ABET) accreditation outcomes are best demonstrated in the context of such a project. These include the ability to design a system, process or component to meet desired needs and the ability to function on a multidisciplinary team. This paper describes a recent California State University, Northridge senior design project in which engineering (computer, electrical, and mechanical) and computer science students work on a multidisciplinary team to design, build, test, and eventually launch a CubeSat carrying a research experiment. The scope of this project has provided an excellent opportunity for computer science students to collaborate with engineering students. In addition to its value as a motivational multidisciplinary project, the project has given students an opportunity to collaborate with local industry on an actual mission that will be launched into space.

I. Introduction

Engineering and computer science programs often require a culminating senior design project. Several of the Accreditation Board for Engineering and Technology (ABET) accreditation outcomes[1] are best demonstrated in the context of such a project. These include:

- The ability to design a system, process or component to meet desired needs
- The ability to function on a multidisciplinary team
- The ability to communicate effectively
- The understanding of professional and ethical responsibility

At California State University, Northridge (CSUN) engineering and computer science students are assigned to work on group design projects during their senior year. At CSUN, senior design projects are typically offered within the individual engineering departments. Some projects have included engineering students from outside the department to provide a multidisciplinary team experience. However, prior to the project described here, computer science students had not joined any of the engineering project teams.

Projects without a computer science component do not reflect the current engineering approach where software development is an essential part of any real world project and the boundaries between disciplines are increasingly blurred. It is therefore vital that engineering students learn to work effectively on projects that span as wide a discipline spectrum as possible[2 - 4].

This paper describes a recent CSUN multidisciplinary senior design project in which engineering (computer, electrical, and mechanical) students teamed with computer science students to design and build a CubeSat capable of being launched and carrying out a research experiment. The range of tasks required to complete this project make it ideal for a team from multiple departments.

Since the uniqueness of the mission requires custom software, rather than an integration of existing software with an operating system and since previous papers have concentrated on projects across engineering disciplines, this paper will focus on the computer science aspect of the CubeSat project. This CubeSat project is being performed in partnership with the Jet Propulsion Laboratory (JPL), a local employer of CSUN graduates.

Section II of this paper describes the CubeSat project. Section III describes the project team and the challenges in running a large multidisciplinary project. Section IV describes the project management approach of the software team and the relationship between the project and the computer science curriculum. Section V includes some assessment of this approach. Section VI presents the conclusions.

II. Description of the CubeSat Project

A CubeSat is a miniature satellite (20 x 10 x 10 cm) capable of carrying an onboard experiment into space. CubeSats are launched free of charge as part of government and commercial satellite launches around the world and have a potential lifetime of years, orbiting the earth approximately every ninety minutes. In addition to requiring a team of students to design, build, and test the miniature satellite, this project also requires the design and construction of a telecommunications ground station, complete with mission control, to command and control the CubeSat during its mission and retrieve the valuable scientific data it will produce.

The project offers a wealth of educational and technical opportunities ranging from project management, unique design challenges, rigorous constraints, extensive testing and the demand for excellence in execution. Students must overcome daunting real-life problems to carry out a mission that will have tangible consequences in future space missions. They must wrestle with the scale of a project involving several subsystems which have to be brought together seamlessly and where any failure threatens the mission. Student members have to adopt the skills of working with project members outside their discipline, to communicate, coordinate and cooperate through meetings, reports and documentation. They have learned firsthand the reasons for requirements, timelines, interface control documents, power and thermal budgets and test plans.

The fact that this is a real mission, not a simulation or class exercise, motivates students to expand their knowledge, re-orient their thinking and refine their skills. They approach their work with pride knowing that this is something that literally the whole world will see.

III. CubeSat Project Organization

The design, build, test, and launch of the CSUN CubeSat will require approximately two years. In a given semester, 20 - 25 students are actively working on the project. During the first year the team included one mechanical engineering student, six computer science students, four computer engineering students, and ten electrical engineering students. The senior design course is a two-semester sequence. To provide continuity, new students are added every semester so that half of the team is new and half of the team is continuing.

A project this large requires a significant amount of faculty supervision. The current team consists of four faculty advisors who supervise various aspects of the project.

Most students are divided into groups which focus on the various subsystems (power subsystem, satellite communications, satellite processor, software design, and ground station) while others focus on systems level tasks.

In addition to CSUN students and faculty, engineers from JPL are participating in the CubeSat project by designing the payload experiment. JPL provided the initial seed funding for this project while a NASA grant is providing faculty support, equipment, and parts.

IV. CSUN Software Team

The software team of the CSUN CubeSat Project was comprised of senior computer science students whose contributions to the CubeSat comprised their Capstone project (partially fulfilling the requirements of the computer science Senior Design Project course). A professor from the Computer Science Department acted as manager and advisor to these computer science students.

Upholding the requirements of the Senior Design Project course, the Agile Project Management Methodology was employed in managing the software team. In this section, we provide a brief introduction to Agile Project Management[5] with Scrum[6], and provide the specifics of how Agile is employed in our computer science Senior Design Project. We also discuss the aspects of the CubeSat Project that coincide with a traditional computer science undergraduate education. Further, we discuss how the software team was selected from the seniors in the Senior Design Project. Finally, we discuss how the software team organized their tasks into a set of consecutive sprints, and provide an overview of the tools they used in their development of the satellite software.

A. Agile Project Management With Scrum

Traditionally, software projects have been managed using a sequential approach, such as the Waterfall method[7]. In such a method, development proceeds through a series of steps, beginning with a conceptual design document and ending with the deployment of a tested software system. Each of these steps is performed in its entirety before proceeding to the next step. Value is delivered all-at-once, at the very end of the project lifecycle.

This approach to project management has its roots in assembly line manufacturing, in which each product was created by a series of independent steps. Although such an approach to human endeavor is appropriate for predictable and repeatable tasks, in the modern world tasks of this nature are largely automated by computers. In today's software industry, humans are employed to perform tasks that contain an element of ever-present change, ambiguity, and unpredictability. In such unstable conditions, sequential software project management methods like Waterfall tend to suffer, as less is known about the system at the beginning of the design process than will be known later in development. For instance, requirements are fully specified before the system is designed, which tends to impose rigidity on the system implementation. If fundamental design problems are not identified until implementation or testing, they can be difficult to correct. In the presence of evolving or ambiguous requirements, project progress may be stalled as designers scrap partial implementations and go "back to the drawing board."

In Agile Project Management[5] each of the stages of the Waterfall method is performed repeatedly, in small iterations of development. The term "Agile Project Management" is an umbrella to describe a handful of such lightweight methods, including Scrum[6]. Agile methods like Scrum were experimented with in the mid-1990s, and began to see adoption in the software industry in the last decade. Although Scrum is now the leading Agile method, it has yet to be widely adopted outside of the IT industry[8].

In Scrum[6], every iteration of development is called a *sprint,* and a sprint is scheduled over a period of 2-4 weeks. The set of development tasks to be performed in a given sprint are called the *Sprint Backlog*. This set of tasks is a subset of tasks maintained in the *Product Backlog*, which is a list of all tasks to be performed in the development of the final product. Prior to every sprint, a set of tasks from the Product Backlog is selected for inclusion in the current Sprint Backlog. At the end of every sprint, any tasks from the Sprint Backlog which are not complete are re-placed in the Product Backlog. An estimate of the time remaining on each sprint task is maintained alongside the Sprint Backlog, in order to forecast progress.

Every workday, all team-members have a time-boxed 15-minute *Scrum meeting* (often referred to as a *standup meeting* as it is often held with the entire team standing in a circle). At this meeting, each team-member shares their experience since the last Scrum meeting, and their intended progress before the next Scrum meeting.

B. Agile Method of Computer Science Senior Design

In the CSUN Fall 2013 computer science Senior Design course (COMP 490 and 490L, taught as a separate lecture and lab), Scrum was adopted as the project management methodology for each student team. The seniors formed teams of 12 students, with each team independently implementing a separate year-long software development project. The CubeSat software team was an exception, as the team size was limited to 6 students, and the team was selected by the faculty managing ongoing CubeSat development.

The CubeSat project did not employ any Agile project management methodology prior to the Fall 2013 semester. Instead, CubeSat development had relied on a loose sequential methodology. In concordance with the course requirements of the Senior Design course, the CubeSat software team adopted Scrum as their management methodology. This methodology was employed only among the 6 seniors on the software team, in isolation from the rest of the CubeSat engineers.

Sprints were scheduled as two week iterations of development, with standup Scrum meetings scheduled three times a week (twice in Senior Design lab section, and once in a weekly Friday meeting with the CubeSat faculty advisor from computer science). The CubeSat computer science faculty was given the Scrum role of Product Owner, whose responsibility is maintaining and prioritizing the overall Product Backlog. The roles of Scrum Master (a facilitator of Scrum standup meetings) and Scribe (the unofficial role of meeting note-taker) rotated between the software team members every two sprints.

The Agile Scrum methodology employed by the software team provided some unique advantages. First, as per the official rules of Scrum[6], none of the members of the software team were designated “team lead,” nor were any members given any other sort of authority or oversight. Second, Scrum encourages teams to be self-organizing. Indeed after the first sprint, the student team was independently capable of decomposing sprint tasks into subtasks with little-to-no additional guidance from faculty, and self-managed its assignment of these subtasks to team members. (The first sprint's tasks were decomposed and assigned by the computer science faculty, in order to ease the team into the project.) The team was responsible for maintaining its own Sprint Backlog, including documenting daily estimates of remaining work on each task. The frequent standup meetings maintained accountability, as each team member reported their incremental progress (or lack thereof) to the entire team, and discussed any obstacles they encountered in their work. Finally, using an Agile methodology helped the team remain flexible in the face of evolving project requirements. The scope of the CubeSat software, and even some fundamental decisions regarding high-level system organization, were not hardened until the end of the Fall Semester. Regardless, the software team was able to complete many critical development tasks in lieu of complete requirements, a feat that is difficult or impossible using a sequential methodology like Waterfall.

C. CS Education Aspects of Project

The software requirements of the CubeSat project reinforce many important aspects of a traditional computer science education. The software team was provided with hands-on experience programming and debugging a large project. In the course of programming a low-level embedded development platform, students gained valuable practice with bitwise operations, bit-masks, and number system conversion. The limited bandwidth of the communication link between the CubeSat and the ground gave the team an opportunity to explore networking and packet-switching beyond the domain of internet protocols. Naturally, this led to issues of computer security, such as the ability to authenticate transmissions from Mission Control, and the decision of whether or not transmissions should be digitally signed by each side. For these design choices and their subsequent implementation, the software team was able to apply lessons learned in our computer science curriculum, providing a true capstone experience.

However, there were a number of significant differences between the CubeSat project and the types of projects students had prepared for in their undergraduate classes. Whereas the language emphasized in our curriculum is the object-oriented Java, the CubeSat software was written in the C programming language, which computer science students are only required to learn and use during one brief semester, in the Operating Systems course. Moreover, due to early design decisions made in favor of determinism and reliability, the team was faced with the unusual task of developing firmware to run on the "bare-metal" of the CubeSat processor, without any underlying Operating System nor RTOS. Thus, these computer science students found themselves without a heap for the first time in their programming careers, and had to forgo any kind of dynamic memory allocation.

For the majority of the team (five out of six students), this was the first encounter with embedded systems programming, as there is no Embedded Systems course required in our curriculum. Thus, this project marked the students' first practical experience with hardware timers, interrupts, programmable controllers, processor I/O, and in-circuit debuggers. Challenges posed by this unfamiliar platform included the need for a fault-tolerant design, and the imposition of timing, power, and memory constraints. Students had to revisit concepts of system organization, including the behavior of the runtime stack, the functionality of a boot loader, the intricate behavior of linkers and loaders, and the layout of code and data in memory. Such an experience was invaluable, as this project took these concepts beyond the whiteboard and brought them to life as tangible software challenges.

Finally, the software team's integration with other student engineers working on other aspects of the CubeSat provided a true multidisciplinary project experience. Our computer science seniors had to digest information from electrical engineers, and package details about their own work in a form that their electrical engineering peers could understand. This deepened the technical communication ability of the software team, and provided excellent preparation for a professional workplace setting with large, complex projects built by multiple teams of varied backgrounds and specialization.

D. Selecting the Team

At the beginning of the Fall 2013 semester, the CubeSat faculty presented the satellite software as a prospective project to both sections of COMP 490, and interested students were encouraged to email the faculty for more information. The faculty then conducted a round of interviews with all interested students, before selecting the students who would join the CubeSat team.

During the interviews, the faculty got a sense of each student's enthusiasm about the project, and that student's ability to commit ten hours per week to the CubeSat software. Additional technical questions were asked to probe students' familiarity with the C programming language (in particular with pointers), as well as with dynamic memory allocation, and bitwise operations.

E. Sprint Planning

The fifteen weeks of the Fall 2013 semester were decomposed into seven 2-week Scrum sprints (with one sprint scheduled for 3 weeks owing to the Fall holiday schedule). At the beginning of each sprint, the team and the computer science faculty advisor had a three-hour Sprint Planning meeting, wherein a set of high-priority Product Backlog tasks were selected and added to the current Sprint Backlog. The set of tasks in the Product Backlog, and their priority, had been decided on in advance by the CubeSat faculty, and had been added to the Product Backlog by the Product Owner (computer science faculty). Once the Sprint Backlog was assembled, the team collectively estimated the number of hours each task would take using a live Planning Poker session.

During Planning Poker, for each task on the Sprint Backlog, each team member held up a paper card indicating whether they felt the task would be small, medium, large, or extra-large (corresponding to an estimate of 3, 5, 8, or 13 hours of work). For each task where teammates disagreed on the estimates, a brief discussion followed, where the difficulty of the task was debated. Next, each teammate once again held up a card indicating their estimation of difficulty, which was potentially revised after the discussion period. After this, if consensus was not reached, the majority opinion became the official estimated effort of the task. This step differs from the original rules of Planning Poker[9], which indicate that discussion should continue until consensus is achieved, or until the team decides that a task should be deferred until further information is available.

Planning Poker discussions had the useful side-effect of encouraging the team to discuss the details of task implementation, and highlighting some of the subtle difficulties of certain tasks.

Such discussion amounted to a collaborative design of each component of the software, and allowed large tasks to be decomposed into subtasks by the entire team.

F. Software Development Environment

The CubeSat processor is a Microchip dsPIC33 microcontroller. This processor is found in Microchip's Explorer16 Development Board (five of which were shared between the software team and the other engineers on the CubeSat project) as well as the Pumpkin CubeSat Kit™ Development Board (one of which was shared by project members). The Microchip MPLAB X IDE was used in concert with the Microchip XC16 C Compiler. Hardware debugging was performed with a Microchip ICD 3 in-circuit debugger. Subversion was used as our source code management tool.

V. Assessment

In order to assess the educational experience of this project, a survey was created for the six computer science seniors, comprised of three questions. The response rate was 100%. In the first question each team member compared their teamwork, communication, and programming skills before the senior project (i.e. before this semester) to those they have now. In the second question, each team member compared the current effectiveness of the entire CSUN CubeSat Software Team to the team's effectiveness at the beginning of the project. The third question allows students to rate the contribution of the Agile methodology to their own work.

Each of the sub-questions is rated with a five point Likert scale, with responses assigned a score of 1 (poor / strongly disagree), 2 (below average / disagree), 3 (average / neutral), 4 (above average / agree), or 5 (excellent / strongly agree). Student responses are reported with the average score across all six seniors.

Finally, we use artifacts of the Scrum process to track the Software Team's productivity over the first six sprints of the CubeSat project this semester. Specifically we report the team's estimated value provided each sprint, and use this to calculate the team's acceleration over the course of the semester. This measurement provides additional confirmation of the team's success in navigating the learning curve of the CubeSat project. Moreover, team acceleration is an impartial indicator of team capability, untarnished by any personal biases.

A. Self Assessment

In Figure 1, we include the first survey question, which asks computer science students to rate their own abilities before-and-after the first semester of the CubeSat project. As this senior design project is intended to be the culminating experience of the Software Team's undergraduate education, many of the self-assessment survey questions were selected to measure the degree to which this project met ABET Criterion 3 objectives for computing programs[10]. These ABET objectives are student outcomes, and enumerate the skills that students must attain by the time they graduate. All questions related to understanding of memory allocation, the runtime stack, and the C programming language (1.9 - 1.14) are derived from Criterion 3 Student Outcome (a): "An ability to apply knowledge of computing and mathematics appropriate to the discipline." Questions 1.15 and 1.16 directly assess Student Outcome (b): "An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution."

Questions 1.5 and 1.6, as well as questions 1.17 and 1.18, are related to Student Outcomes (c) and (i), which are “An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs” and “An ability to use current techniques, skills, and tools necessary for computing practice” respectively. Questions 1.1 and 1.2 measure the extent to which this project met Student Outcome (d): “An ability to function effectively on teams to accomplish a common goal.” Questions 1.3 and 1.4 measure the ability for computer science students to communicate with non-computer science engineers to exchange ideas, which assesses Student Outcome (f): “An ability to communicate effectively with a range of audiences.” Finally, questions 1.19 and 1.20 assess Student Outcome (h): “Recognition of the need for and an ability to engage in continuing professional development.”

The remaining questions 1.7 and 1.8 ask each student to rate their own interest in embedded system development, and how it has changed over the course of this project. This question was included to determine if the project has attracted students to this computing field, or if it has done the opposite and discouraged student interest in embedded systems.

Question #1: For each sub-question, please provide a rating (from poor to excellent) of yourself and your own work.

1.1) How would you rate your ability to contribute to a team (before this semester)?
1.2) How would you rate your ability to contribute to a team (now)?
1.3) How would you rate your ability to exchange technical ideas with engineers outside of computer science (before this semester)?
1.4) How would you rate your ability to exchange technical ideas with engineers outside of computer science (now)?
1.5) How would your rate your comfort at programming embedded systems (before this semester)?
1.6) How would your rate your comfort at programming embedded systems (now)?
1.7) How would your rate your interest in programming embedded systems (before this semester)?
1.8) How would your rate your interest in programming embedded systems (now)?
1.9) How would you rate your understanding of memory allocation (before this semester)?
1.10) How would you rate your understanding of memory allocation (now)?
1.11) How would you rate your understanding of the runtime stack (before this semester)?
1.12) How would you rate your understanding of the runtime stack (now)?
1.13) How would you rate your understanding of the C programming language (before this semester)?
1.14) How would you rate your understanding of the C programming language (now)?
1.15) How would you rate your ability to analyze a computing problem and decompose it into implementation tasks (before this semester)?
1.16) How would you rate your ability to analyze a computing problem and decompose it into implementation tasks (now)?
1.17) How would you rate your ability to debug source code (before this semester)?
1.18) How would you rate your ability to debug source code (now)?
1.19) How would you rate your desire to learn and continue learning the art of computer science (before this semester)?
1.20) How would you rate your desire to learn and continue learning the art of computer science (now)?

Figure 1: Self Assessment Survey Questions

Figure 2 summarizes student responses to the self-assessment question, combining each pair of sub-questions (1.1 and 1.2, 1.3 and 1.4, 1.5 and 1.6, and so on…) into a pair of bars that show student ability before the project versus now. For instance, the top pair of bars labeled “team contribution" reports responses to sub-questions 1.1 (Before this Semester, the white bar) and 1.2 (Now, the black bar). Therefore, for the 20 sub-questions, there are 10 pairs of bars.

A positive trend can be seen in Figure 2 for each of the 10 student abilities assessed. Students reported that each these abilities became stronger during the first semester of the CubeSat project. By the end of the first semester (i.e. Now), every ability has an average rating of higher than 3 (average), with only one ability reported as less than 3.5 (comfort at programming embedded systems, from questions 1.5 and 1.6).

The computer science students report that they have become above-average (score of 4.0 or higher) in their ability to contribute to a team, their ability to exchange ideas with other disciplines, their interest in embedded programming, their understanding of the runtime stack, and their desire for continuous learning. Despite such a high expressed interest in embedded systems (4.33, the highest of the aggregate scores), students still exhibit an average level of comfort in embedded programming (3.17, the lowest of the post-semester aggregate scores). However, the overall students' growth in comfort at programming embedded systems (from 1.67 to 3.17 in a mere 15 semester weeks) is highly encouraging. It is anticipated that their comfort will continue to grow in the coming semester, as their development experience deepens.

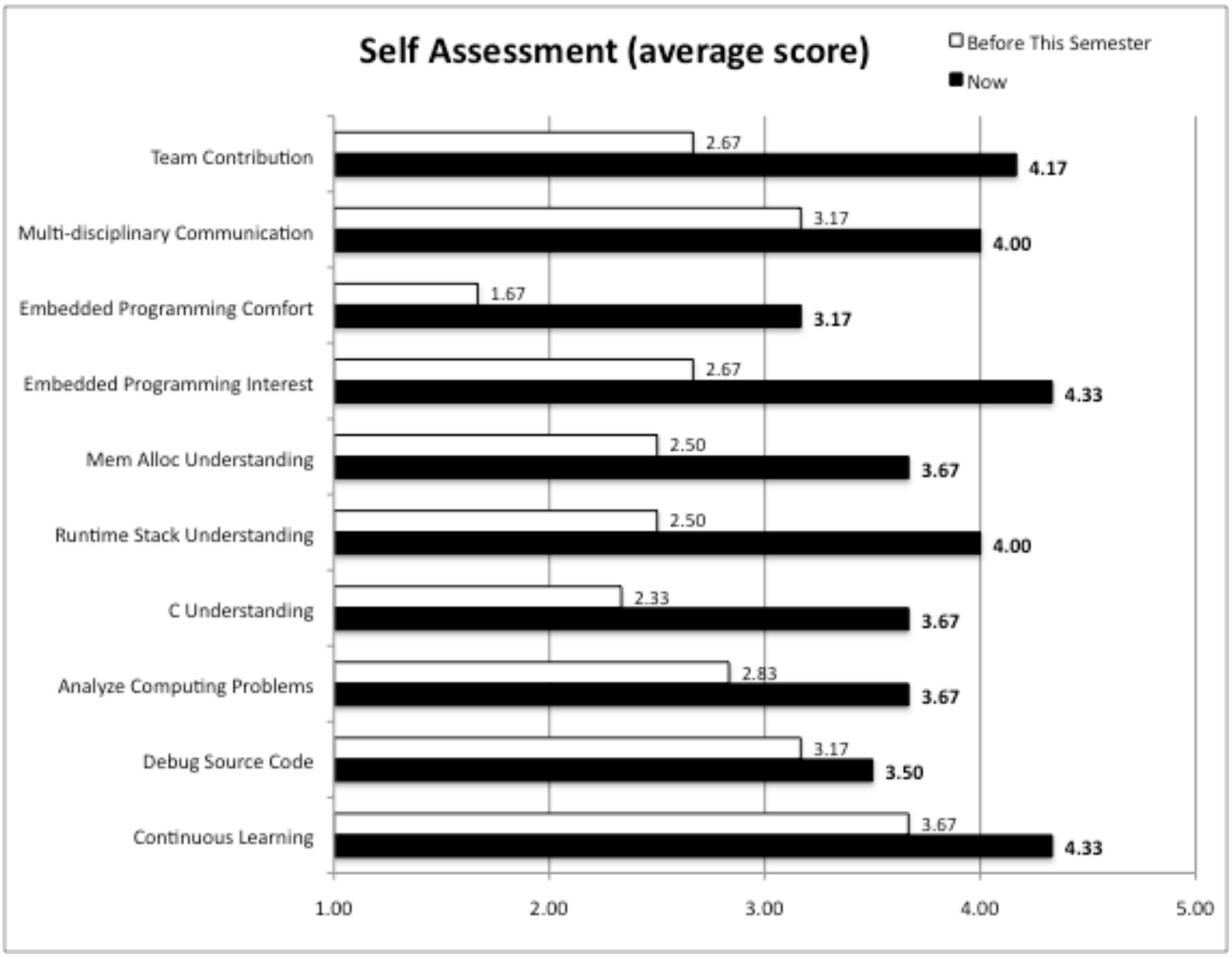


Figure 2: Self Assessment Responses

In Figure 3, we track the growth of the students' self-assessed abilities over the semester, by reporting the difference between their before and after scores (difference of average scores) in each ability. Students experienced the most growth in their interest in programming embedded systems, with 1.67 points gained over the semester. This result indicates that this project has garnered interest in the embedded computing field, rather than scaring students away from such endeavors.

Other areas where students grew significantly (by more than one point) over the semester include their ability to contribute to a team, their comfort in programming embedded systems, and their understanding of memory allocation, the runtime stack, and the C programming language. These results indicate that this project has been most successful in training students to meet ABET student objectives (a), (c), (d), (i).

Students experienced the least growth in their ability to debug source code, from an aggregate score of 3.17 before the semester to 3.50 now. As students primarily develop this skill during the first half of a computer science undergraduate education (in CS1 and CS2 classes, in particular), it is unsurprising that less growth in this area was experienced during the senior year.

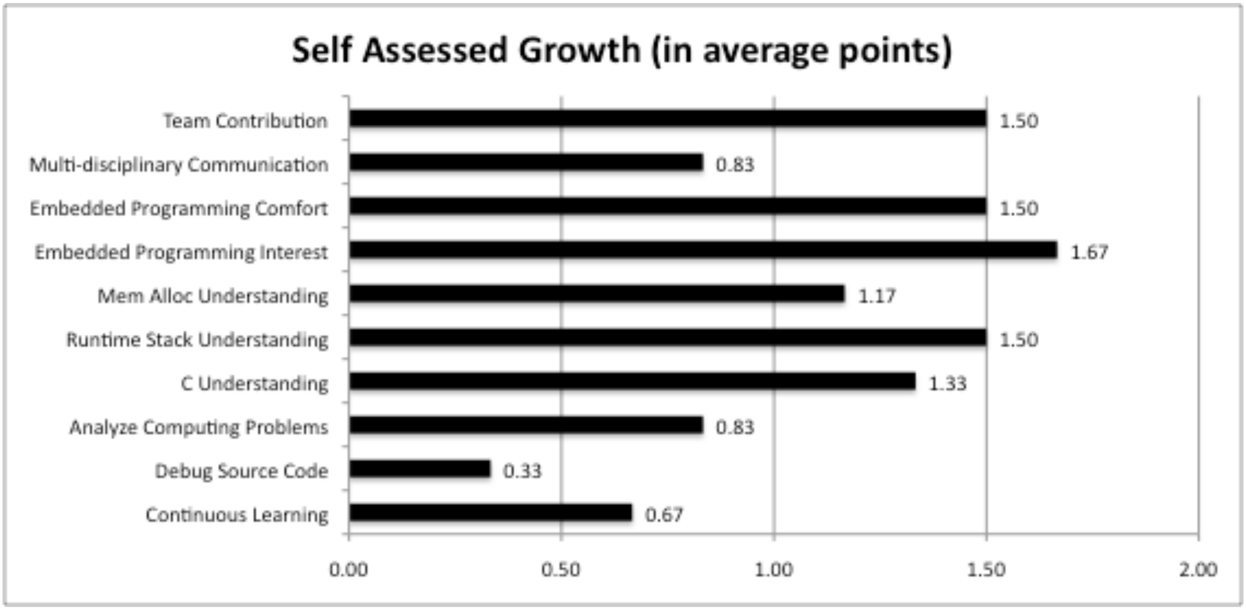


Figure 3: Self Assessment Measured as Growth

B. Team Assessment

Figures 4 and 5 show the team assessment question and averaged responses, respectively. In their responses, each student assessed the effectiveness of the entire CubeSat Software Team at the beginning of the semester and presently (at the end of the semester).

Question #2: For each sub-question, please provide a rating (from poor to excellent) of the overall team.

2.1) How would you rate the overall effectiveness of the CubeSat Software Team (at the beginning of the semester)?
2.2) How would you rate the overall effectiveness of the CubeSat Software Team (now)?

Figure 4: Team Assessment Survey Questions

From the student responses in Figure 5, it seems that the team's overall effectiveness grew over the first semester, from an aggregate rating that was below "average" (2.67) to a rating above "average" (3.83). This increase of 1.16 points marks a significant growth in the students' perception of their team's overall ability, and is concomitant with the growth each student reported in their own ability to contribute to a team (in Figures 2 and 3).

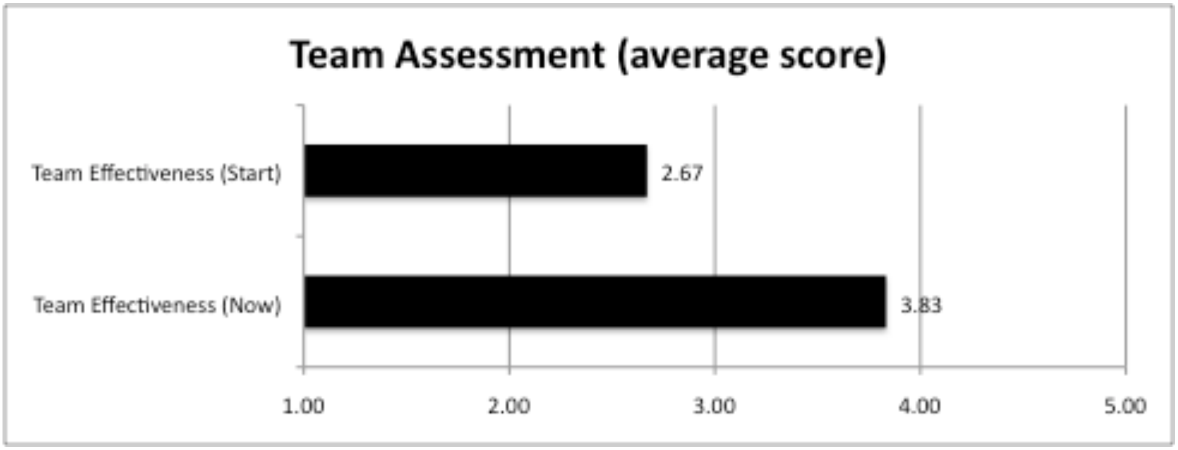


Figure 5: Team Assessment Responses

C. Assessment of Agile Effectiveness

In Figure 6, we provide our final question on the student survey. In this question students are asked to what extent they agree with 10 positive statements about the impact of the Agile methodology on the CubeSat project. Student responses are aggregated in Figure 7.

Students demonstrated some agreement with all 10 statements, with the lowest aggregate score being the neutral agreement of 3.33. Overall, students identified the Scrum standup meeting as the strongest contributor to project success, with aggregate scores above 4 for each of statements 3.1 – 3.3. The Scrum standup contributed to better communication between teammates (exchange of technical ideas, 3.1), better ability to contribute to the team, and better overall productivity of the team. These responses are encouraging, and suggest that frequent meetings of short duration can be very useful in order to keep students on-task as well as foster team spirit.

By comparison, the process of planning each sprint was reported to be slightly less effective than the Scrum standup. Aggregate scores of 3.67 were given as answers to question 3.4 - 3.6, suggesting that Sprint planning meetings, and specifically the included process of Planning

Poker, moderately contribute to useful estimates of effort, as well as team productivity. However, the results suggest that improvements could be made to the allocation of sprint work to each team member, especially with the nearly-neutral answer to 3.7.

Question #3: Please indicate your response to each statement with {strongly disagree, disagree, neutral, agree, strongly agree}.

3.1) Standup meetings facilitated an exchange of technical ideas between teammates working on different tasks.
3.2) Standup meetings contributed to my ability to contribute to the team.
3.3) Standup meetings contributed to the overall productivity of the team.
3.4) Sprint planning meetings contributed to an efficient allocation of work to each team member.
3.5) Planning Poker generated useful estimates of task effort.
3.6) Planning Poker contributed to the overall productivity of the team.
3.7) The tasks I was assigned during each sprint were appropriate for the amount of time given.
3.8) Agile Project Management Methodology generated more utility/usefulness than overhead for this project.
3.9) Agile Project Management Methodology has helped the team remain flexible to shifting requirements.
3.10) If possible, I would employ Agile Project Management Methodology in future engineering projects.

Figure 6: Agile Effectiveness Survey Questions

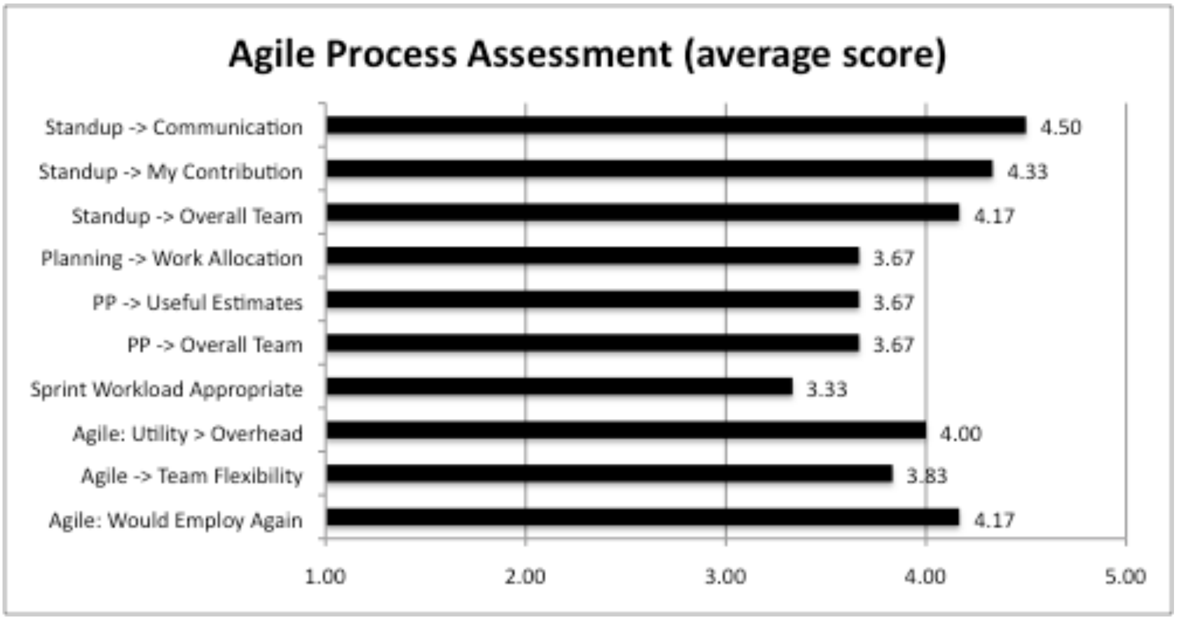
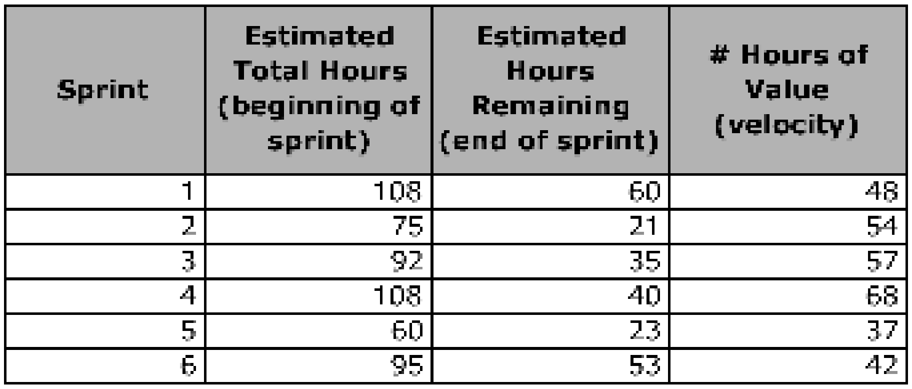


Figure 7: Agile Effectiveness Responses

The responses to statements 3.8 through 3.10 indicate agreement that the Agile methodology has contributed to team success (with the answer of 3.83 for statement 3.9 indicating near-agreement). In general, students feel that Agile provides more utility than overhead, and has helped the team remain flexible despite evolving and incomplete requirements. Students appreciate the strengths of the Agile Project Management Methodology sufficient to use it in their future projects, should the opportunity arise.

D. Sprint Productivity

To estimate the productivity of the Software Team, we utilize artifacts of the Agile/Scrum methodology, namely the total estimated hours of work remaining, which is tracked daily by each team member for each of their tasks. Initial estimates are provided by the Sprint planning meeting, which includes the aforementioned Planning Poker[9] session. In Table 1, we track these hours for each of the first six sprints of the Fall 2013 semester. In the second column, we report the total estimated work hours across the entire Sprint Backlog, as determined by our Sprint planning meeting. In the third column, we report the estimated work hours remaining at the end of the sprint, summed across all tasks that are not complete. The difference between these numbers is the estimated velocity of the team, or the amount of work product (in units of completed work hours) that the team can produce each sprint. Please note: this velocity does not reflect the total number of real hours that the team collectively worked over the sprint. Rather, the velocity is derived from the estimated number of work hours remaining at the beginning and end of the sprint.



Sprint	Estimated Total Hours (beginning of sprint)	Estimated Hours Remaining (end of sprint)	# Hours of Value (velocity)
1	108	60	48
2	75	21	54
3	92	35	57
4	108	40	68
5	60	23	37
6	95	53	42

Table 1: Estimation of Team Velocity By Sprint

However, velocity cannot be used as a direct measure of team productivity[11], as the estimates of work effort tend to be somewhat arbitrary, and take a few sprints to converge. To compare a team's productivity to that of other teams requires calculating each team's acceleration, i.e. the rate at which their velocity is changing. Acceleration can be calculated one of two ways: a) by calculating from sprint to sprint, or b) by calculating from the first sprint to the current sprint. The formulae for these calculations are as follows…

$$acceleration_{n\ (\text{sprint-to-sprint})} = (velocity_n - velocity_{n-1}) / velocity_{n-1}$$

$$acceleration_{n\ (\text{from sprint 1})} = (velocity_n - velocity_1) / velocity_1$$

…where $velocity_i$ is the velocity of the i^{th} sprint, and $acceleration_i$ is the calculated acceleration of the i^{th} sprint.

In Figure 8, we graph the acceleration in sprints 2-6 using both calculation methods. In this figure, acceleration is reported as a percent. Both methods result in the same overall graphed trend, which shows the team's acceleration increasing from sprint 2 to sprint 4, followed by a major dip in productivity during sprint 5, and a renewed increase in acceleration during sprint 6.

The apparent dip in productivity during sprint 5 seems to result from two potentially-related factors. The first is that this sprint was interrupted by the Thanksgiving holiday, during which time the campus was closed, and little useful work was being performed. The second is that the estimated hours of the sprint 5 tasks summed to 60, which is by far the smallest estimated allocation of work given to the team during any sprint this semester. In fact, the sprint 5 velocity is very close to the sprint 6 velocity, as much productive work was completed during sprint 5, despite far less work being assigned for that sprint.

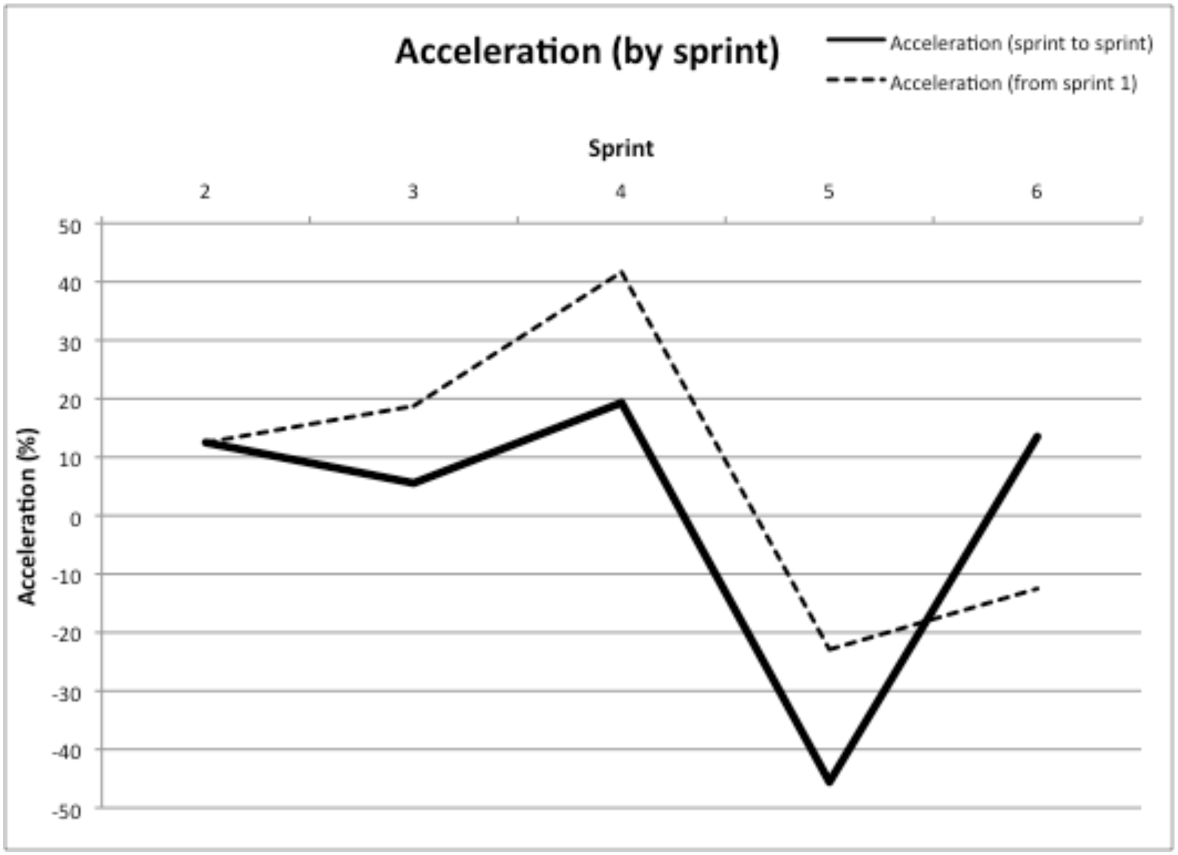


Figure 8: Team Acceleration (by sprint)

Several factors contributed to the Software Team's productivity in the CubeSat project, in which new languages and software paradigms had to be learned during the tight implementation schedule. One was the frequent face-to-face communication provided by the Scrum standup meetings, which was given the most positive overall response by the team in Figure 7. This communication really brought the team closer as a unit, and helped to break the ice during the early stages of ramp-up, as the team familiarized themselves with embedded programming and with each other. In addition to communication, the Scrum standup allowed the team to track each other's progress and work obstacles very closely multiple times per week, allowing implementation and scheduling problems to be addressed as quickly as they occurred.

In addition to the frequent communication between members of the CubeSat Software Team, there were weekly multidisciplinary meetings that allowed the Software Team to interface with other engineering teams assembling the CubeSat. This perspective on the final CubeSat assembly, including how the disparate pieces interface with each other, enriched the paradigms of embedded programming that the Software Team was learning. Furthermore, exposure to schematic diagrams and electrical engineering concepts expanded the vernacular of the Software Team, and helped them collaborate more closely with engineers from other disciplines.

Although acceleration is a straightforward estimate of a team's productivity, its use is limited, as a team's method for estimating remaining work effort is innately imperfect, and the accuracy of the estimate stabilizes over a handful of sprints as the team's process matures[11]. Thus, although studying acceleration trends helps to track how a team's productivity changes over time, the numeric values of acceleration have limited application.

VI. Summary

During the Fall 2013 semester students at California State University, Northridge participated in a multidisciplinary senior project to design, build, and test a CubeSat. The scope of this project has provided an excellent opportunity for computer science students to collaborate with engineering students. In addition to its value as a motivational multidisciplinary project, the project has given students an opportunity to collaborate with local industry on an actual mission that will be launched into space.

This paper has focused on the Agile Project Management Methodology used and the aspects of the project that coincide with traditional computer science education. After the first semester the students' experience on the project was assessed with a survey. The results of this survey show that students:

- Assess that their own abilities in the related student outcomes were strengthened by their participation in the project
- Assess that their ability to contribute to a team has improved
- Have gained interest and confidence in the embedded systems field
- Agree that the Agile methodology has contributed to team success

Additionally, statistical artifacts of our Agile process have allowed us to track the software team's productivity from sprint to sprint. A model of team acceleration demonstrates a positive trend in student effectiveness over the course of the Fall semester. The CubeSat team continues to work on this project and anticipates that the satellite will be complete and ready for environmental testing by Summer 2014.

The faculty's own experience with the CubeSat teams has yielded a number of insights regarding large multi-disciplinary projects. First, we have witnessed a successful pairing of an Agile-managed software team with classical/Waterfall-managed engineering teams on the same project. This is an important affirmation that Agile can be applied outside the box of software-centric projects, which have largely been its domain over the past decade. We have also learned that a significant amount of faculty supervision is required for student projects at this level of complexity. The faculty provided the glue that held the project together, by defining and scheduling high-level tasks, offering advice to project teams, and by facilitating much of the communication between the teams of different disciplines. We found that hand-picking our students via the interview process was an effective way of gauging the interest and motivation of prospective teammates, and advise such an approach when doable. Finally, the faculty feel it would have been ideal for all the students in the class to be enrolled in one single Senior Design Project class or similar elective, as opposed to separate Senior Design classes by major. Although it was infeasible in this situation, such an organization would have leveled the playing field by allowing student work to be evaluated and tracked in a unified manner, and may have enhanced the sense of all team members working toward a single purpose.

Bibliography

1. Criteria for Accrediting Engineering Programs, 2014 – 2015. http://www.abet.org.
2. J. Allenstein, et al, *Examining the Impacts of a Multidisciplinary Engineering Capstone Design*, ASEE Annual Conference, 2013.
3. L. E. Carlson and J.F. Sullivan, *Hands-on engineering: learning by doing in the integrated teaching and learning program*. International Journal of Engineering Education 15 (1999): 20-31.
4. A. Zhang, et al, *Utilizing Project-based Multidisciplinary Design Activities to Enhance STEM Education*, ASEE Annual Conference, 2012.
5. J. Highsmith, *Agile Project Management*, Boston: Addison-Wesley, 2004.
6. K. Schwaber and J. Sutherland, "The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game," Scrum.org, 2011.
7. I. Sommerville, *Software Engineering, 8th Edition*, Essex: Pearson Education, 2007.
8. *The State of Scrum: Benchmarks and Guidelines*, June 2013, http://www.scrumalliance.org/why-scrum/state-of-scrum-report.
9. *Planning Poker: An Agile Estimation and Planning Technique*, http://www.mountaingoatsoftware.com/agile/planning-poker.
10. Accreditation Board for Engineering and Technology (ABET), "Criteria for Accrediting Computing Programs," Computing Accreditation Commission, Baltimore, MD, 2012.
11. S. Ambler, *Acceleration: An Agile Productivity Measure*, October 2008, https://www.ibm.com/developerworks/community/blogs/ambler/entry/metric_acceleration.