# Social Context, Singular Focus

David R. Chesney
Electrical Engineering and Computer Science Department
The University of Michigan
Ann Arbor, MI, USA
chesneyd@umich.edu

***Abstract*— In the Computer Science Department at the University of Michigan (UM), we have spent the past five years building social context into several courses in the traditional computer science (CS) curriculum. Specifically, freshman- and senior-level, project-based classes have been designed to both teach significant and appropriate academic content, while also building software games and apps for children with cognitive and/or physical disabilities within the university's associated hospital. Children with disabilities provide the *context* for these courses, while the *content* remains representative of a traditional curriculum.**

**The cadence of the course(s) is typically: propose, pitch, form groups, design, build, test, and improve. Each student submits a written proposal for the project. A group of experts read all of the proposals, and choose the 'best' for pitching to the other students in the course. Groups are formed, using student preferences, based upon the pitches. The newly formed groups follow an engineering process that includes gathering requirements and designing the software system. Student groups partition the work, and implement three iterations (alpha, beta, final) of their designs. Groups are re-formed, and the newly formed groups test code from each other's groups. Finally, original development groups have the ability to make final improvements to their code. Student groups are offered the opportunity to continue work on promising projects through subsequent independent studies or internships. If the software project is 'minimally viable', the groups are encouraged to continue development and work with the university's Center for Entrepreneurship or Office of Technology Transfer toward eventual commercialization.**

**During the past academic year (2013-2014), one of the senior-level courses had the singular focus of developing software apps and games for a 13 year old girl with athetoid cerebral palsy. Her family was actively involved in the course, filtering project ideas and offering suggestions for improvement. Some of the most promising apps are currently installed in the family's house for methodical review and eventual improvement. This paper will discuss the general process that is used in this style of course, the specific approach that was used during the 2013-2014 academic year, and offer general suggestions for implementation in disciplines other than computer science.**

***Keywords—software engineering, social context, collaborative learning.***

The author wishes to thank the Mott Golf Classic, Mott Family Network, and UM Center for Research on Learning and Teaching (CRLT) for generously funding this work.

## I. INTRODUCTION

In the Computer Science Department at the University of Michigan (UM), we have spent the past several years building social context into several courses in the traditional computer science (CS) curriculum. Specifically, freshman- and senior-level, project-based classes have been designed to both teach significant and appropriate academic content, while also building software games and apps for children with cognitive and/or physical disabilities within the university's associated hospital. Children with disabilities provide the *context* for these courses, while the *content* remains representative of a traditional curriculum.

The 'recipe for success' in the courses is similar, although the expectations are tempered by the students' academic level. In brief, health professionals introduce a disability to the classroom of students. The students are given access to a wealth of technology to address the disabilities. Some traditional lectures are incorporated into the class, but much of the students' time is spent developing approaches, games, and apps for children with the disability using all available technology.

As examples, during the 2011-2012 and 2012-2013 academic years, student groups in EECS481 Software Engineering used the Microsoft Kinect gaming sensor to create games and apps for children with autism spectrum disorder (ASD) [1]. During the most previous academic year (2013-2014), students used multiple different technologies (such as Intel Interactive 3D cameras, Google Glass, and Makey-Makey), to build communication and playful tools for a young girl with cerebral palsy [2, 3, 4].

This paper covers the general approach used in these courses, with specific discussion of the differences between the freshman- and senior-level course. The paper is arranged as follows. First, the courses are described with focus on what is similar and what is different between them. The approach that is described is easily applied to other disciplines of engineering. Then, both expected and unexpected results of our efforts are discussed. Finally, the paper is summarized and references are offered.

## II. THE COURSES

The specific courses that are described in this paper are ENG100, Introduction to Engineering; and EECS481, Software Engineering. All freshman engineering students at UM must take a section of ENG100, Introduction to Engineering. ENG100 is, as aptly named, an introduction to the engineering process. The course is commonly referred to as a DBT course, which stands for Design / Build / Test. In this course, students work on a semester long project applying engineering principles to a topic in the context of the professor's expertise. As an example, a naval engineering student might have his students create a small submarine, while an aerospace engineering professor might have her students build a blimp. The author teaches a course where students design, build, and test a computer game. The students are introduced to a basic software engineering model of understanding the domain, implementing a software game, then testing the game.

As seniors, students must complete a Major Design Experience (MDE) course. An MDE course is often referred to as a 'capstone' experience. Basically, capstone courses are the final courses that students take prior to graduation. They simulate an industry / production experience, and give students a chance to apply all (or much) of their acquired academic expertise to large project, typically while working in a group. There are several Computer Science MDE courses, one of which is EECS481 Software Engineering. EECS481 covers the following material: pragmatic aspects of the production of software systems, dealing with structuring principles, design methodologies and informal analysis. Emphasis is given to development of large, complex software systems. Finally, a term project is usually required.

### *A. Similarities between ENG100 and EECS481*

This section of the paper describes the similarities between the two courses, as there is much in common. The subsequent section describes the differences.

There is a strong Technical Communication (TechComm) component in both courses**.** In fact, in the curriculum for all engineering disciplines, TechComm plays a pivotal role. TechComm assignments include both written and oral communication, such as written proposals, oral pitches, written progress reports, oral final reports, and written user manuals. All of the TechComm assignments help create an engineering student who is not only able to do good work, but is also able to communicate that they have done so.

Typically in any semester, we have an early introduction to the disability by a medical professional from our local, affiliated hospital. The medical professional initially informs the students about the disability, vets the project proposals, and is often available for questions from the students throughout the semester. They are also typically present for end-of-semester progress reviews of the projects.

The general approach to both courses is: Propose, Pitch, Form Groups, Design, Build, Test, and Improve. Each of the steps in the approach is briefly described below.

**Propose**: After an introduction to the course material, disability, and technology, each student in the class submits a 1-2 page proposal of their software game or app idea. The proposal describes gameplay, intended audience, and how the proposed idea might be an aid to a person with the disability. All proposals are reviewed and vetted by course staff and medical professionals and the 'best' proposals are invited to pitch their ideas to other students in the class in the hopes of recruiting a group.

**Pitch**: As mentioned, students who have the 'strongest' proposals are invited to pitch their ideas to the class. Pitches are typically 3-5 minutes in length and are somewhat informal. Content of the pitch is usually self-introduction, description of game, and skills that the student needs (and is currently lacking) to make the project particularly strong. Students often receive some extra-credit points for pitching to the class.

**Form Groups**: Based upon pitches, students self-select their top three choices for groups. Course staff takes these selections into consideration, insofar as possible, when determining group assignments. Typical group size is 3-5 students. Attempts are made in group assignment to not isolate a single, under-represented student in a group. That is, group dynamics are generally much healthier if multiple, under-represented students (e.g., two females) are assigned to the same group [5].

**Requirements and Design**: Newly formed student groups must come up with some form of documentation of Requirements and Design. In ENG100, the freshman typically will write a user manual that describes their anticipated software product. In EECS481, the seniors follow a more traditional software development cycle. That is, a formal requirements document and design document are written prior to code development.

**Build**: This process step is 'where the rubber hits the road'. Student groups work to develop three iterations of their software product – alpha, beta, and final releases. Alpha releases are proof-of-concept releases that show initial effort and potential risks when compared to the original requirements and design. Beta releases are much stronger releases that show near-final quality software products and are used for the testing phase (discussed next). Final releases are, as expected, the final products. There is often a written and / or oral report at the end of each build phase.

**Test**: Group membership is rearranged for the testing phase. That is, students are placed into different groups for testing then they were in for development, and no students test their own product. Students are encouraged to find bugs in each other's projects, and in fact, are rewarded for doing so. This step takes approximately one week.

**Improve / Maintenance**: Finally, original development groups are reassembled to consider all of the software bugs that were found in the testing phase. Final improvement to the software product generally makes for an outstanding final product.

Fundamentals of individual accountability are necessary because much of the project related work is group-based [6]. These ensure that all students are contributing to the projects

and that the group is high-functioning. As part of measuring accountability, several evaluations are used for each student: self-, peer-, supervisor-, subordinate-, and course staff. That is, each student may have up to five evaluations used to determine a final project 'multiplier' at the end of the semester.

Finally, in addition to the process mentioned above there is additional significant academic content. As part of holding students accountable to the content, pop quizzes are sometimes given. Students are often at a disadvantage due to the quizzes, so an additional homework assignment is given that can be substituted for any other homework or quiz. The additional homework assignment is called Random Act of Kindness. Students are encouraged to commit a Random Act of Kindness for someone else on campus, and report on it in TechComm memo format. The RAOK must be for a complete stranger and cost no more than $5 (if it costs anything at all).

### *B. Differences between ENG100 and EECS481*

Clearly there are differences between the expectations of a freshman- and senior-level course. The expectations are based upon academic content and appropriate technology. The freshman-level course is taught to first or second semester freshman. In many ways, they are 'glorified high school seniors'. The content is relatively light by college academic standards, but still challenging enough to help the freshman make decisions about potential engineering major (e.g., mechanical, electrical, naval, or computer science). Much of the content for the freshman is directed toward how to survive and thrive in the new environment called college. However, freshman will also learn multiple programming languages and how to use those languages for software product development. Also, in the freshman course, not much new technology is used. Every so often, a student group will want to develop on some new gaming platform, but most often, student project groups develop projects using only a keyboard and a mouse.

The sophistication and expectations for the project developed by senior-level students is significantly higher. Students have worked for four years developing their academic expertise, and this course is an opportunity to shine. Many have had summer internships with leading tech companies, such as Microsoft, Google, and Apple. Therefore, they often bring a rich blend of expertise to the student projects.

Also, the course infrastructure has accumulated some very useful and interesting technology for the students to use. Examples include Google Glass, Microsoft Kinect sensors, Intel 3D Interactive cameras, Apple iPads, and Makey-Makey's [7]. Perhaps the Makey-Makey deserves a brief, further description. It is basically a circuit board that can be used to turn anything that conducts electricity into an input device for a computer. Did you know that play-doh, fruit, and graphite conduct electricity? That means that students can build mock-ups of input devices to drive their software out of bananas and oranges and see how the children will play with them. Students are encouraged to use combinations of the technologies in unique ways in order to best address the intended disabled population.

### *C. The 'Secret Sauce'*

Much of the success of this approach can be attributed to the close relationship between the College of Engineering (CoE) and the affiliated children's hospital (CS Mott). Working with caring and dedicated medical professionals at Mott has resulted in a very positive long-term relationship.

As mentioned earlier, EECS481 students have used the Microsoft Kinect to build games and apps for children on the autism spectrum during both the 2011-2012 and 2012-2013 academic years. One of the medical professionals was able to bring a group of children with autism to play the beta releases of the games with the EECS481 students. An expected result was excellent feedback on game improvements. However, a completely unexpected result was a heightened level of motivation for the EECS481 students during the software development cycle. That is, they were much more motivated after meeting the children for whom they were building the apps.

With that success in mind, the focus was shifted from a general disability (like ASD), to a very specific disability and one special child named Grace [2, 3, 4]. Grace is perhaps unique in that her personality is happy, genuine, and infectious. Grace's parents were also able and willing to be involved in the class during both the fall 2013 semester and the winter 2014 semester. Grace and her family drove the 90 minutes from her home to attend class approximately 6 times per semester. They were also generally available to answer student questions.

Grace's involvement gave the students a unique opportunity to work with a very special child, and to be completely motivated by her. At the end of the semester, the games and apps were ported and installed at Grace's home for future use.

## III. THE AFTERMATH

The original intent of these courses was simply to make students more aware of social concerns. We have had (happy) unexpected results while pursuing this 'awareness'. This section of the paper addresses some of those unanticipated successes.

### *A. Demographics*

One of the motivations for building social context into ENG100 and EECS481 is the hope that the context would make the courses more attractive to under-represented populations, particularly female students [8]. With this motivation in mind, the following tables describe the male/female demographics in these two courses over the past five years.

Table I shows the male/female demographics from all offerings of ENG100 over the past five years. Table II shows the same for EECS481. There were not significant trends exhibited by the data, with the exception of ENG100 during the W14 semester. In this case, the percentage of females was approximately twice its usual value (27% vs. values typically between 8-17%). Time will tell whether or not the W14 data is a 'blip', or a promising long-term trend.

Table III shows male/female enrollment in the College of Engineering, Computer Science (CS) majors, and Computer Engineering (CE) majors for the most recent academic year. It should be noted that female enrollment in ENG100 during W14 (27%) is about twice the CS female enrollment (14%). Again, time will dictate whether this is a trend.

Enrollment in ENG100 is intentionally capped at 60 students, and it is only offered once/year. Enrollment in EECS481 is targeted at ~60-70 students. However, due to the popularity of the course and increased enrollment in CS, EECS481 is now offered every semester with a slightly increased enrollment cap. In short, the courses are very popular and typically are full within 72 hours of enrollment being opened for any given semester.

TABLE I. ENG100 INTRODUCTION TO ENGINEERING DEMOGRAPHICS

| Term | Population | | |
|---|---|---|---|
| | ***Total Students*** | ***%Female*** | ***%Male*** |
| F09 | 59 | 8% | 92% |
| W11 | 61 | 16% | 84% |
| F11 | 57 | 9% | 91% |
| F12 | 58 | 17% | 83% |
| W14 | 59 | 27% | 73% |

TABLE II. EECS481 SOFTWARE ENGINEERING DEMOGRAPHICS

| Term | Population | | |
|---|---|---|---|
| | ***Total Students*** | ***%Female*** | ***%Male*** |
| F09 | 27 | 19% | 81% |
| F10 | 42 | 16% | 84% |
| W12 | 48 | 10% | 90% |
| W13 | 71 | 14% | 86% |
| F13 | 74 | 11% | 89% |
| W14 | 64 | 16% | 84% |
| F14 (proj.) | 70 | 19% | 81% |

TABLE III. COLLEGE OF ENGINEERING DEMOGRAPHICS

| Term | Population | | |
|---|---|---|---|
| | ***Total Students*** | ***%Female*** | ***%Male*** |
| F13 (All) | 5950 | 24% | 76% |
| F13 (CS) | 652 | 12% | 88% |
| F13 (CE) | 224 | 12% | 88% |
| W14 (All) | 5609 | 25% | 75% |
| W14 (CS) | 750 | 14% | 86% |
| W14 (CE) | 228 | 14% | 86% |

### *B. Design Viability and Commercial Success(es)*

In many ways, the quality of the products designed in these classes is dictated by 'the market', in addition to the grades assigned to students at the end of the semester. That is, commercial and clinical success is directly related to the quality of the software products. In these regards, the efforts described in this paper have been bountifully rewarded. Several of the projects have received national and international recognition, resulted in start-up companies, and/or proven effective as therapeutic tools for children with disabilities. Examples of each type of success are described below.

As mentioned, some of the projects have had remarkable success, both inside academia and out. One of the applications is called ASK, which is an acronym for Assistive Scanning Keyboard [9]. ASK is a mobile app that scans the rows of a traditional keyboard, using the entire screen as an input button. Once the user chooses a row, the app scans the individual items in the row. Again, the user can touch anywhere on the screen to make a selection, thus eliminating the need for a user to touch individual keys – a task that is often difficult for someone with limited fine motor skills. ASK won many national and international awards including:

- First place Bay Area Entrepreneurship competition;
- Second place Mobile World Congress Mobile Application Development;
- Winner of Student da Vinci awards for Multiple Sclerosis;
- Intel Innovators Award.

These successes led to the student-driven development of an operating system (OS) to support assistive technology. Both the OS and one of the student developers of ASK has been involved in the course since fall semester 2010.

Another game, called PATH, was created to exercise large motor skills for kids with autism spectrum disorder (ASD) and has also had significant success. The project was originally developed during the winter 2012 semester. Although the game was originally developed for children with ASD, a neurosurgeon from Mott Children's Hospital saw the tool and thought that it would benefit her patients with brachial plexus injuries. Brachial plexus injuries occur typically during a difficult birth, and result in nerve and muscle damage to either the left or right arm. Although the severity of the injuries varies, children with these type of injuries must have constant exercise of the damaged limb in order for the function of the limb to not regress. The game (PATH) uses a Microsoft Kinect and is now finishing a clinical trial with approximately 20 patients. Analysis of the data from the clinical trial is forthcoming, and looks promising. It should be noted that the software might also work with stroke patients, although the user interface would need to be modified.

The success of these software tools has encouraged us to develop a formal, workable approach to move the software from an academic exercise to clinical trial with a patient population. We currently anticipate 1-2 clinical trials of newly developed software to occur each year.

### C. Corporate Structure and Intellectual Property

Frankly, the success of these courses, both in terms of enrollment and viable software products, necessitated a method of handling the software products after the completion of the semester. That is, ASK and PATH (mentioned in the previous section) were only the beginning of a series of successful products. We have approximately fifteen (15) new software products from each class during each semester. Typically, somewhere between 3-5 products are 'minimally viable' at the end of each semester. Therefore, we needed to develop a structure that could maintain the software products even after the semester was over and the students have moved on to other courses or life after the university.

One of the author's obligations as faculty for the students is to inform them of their intellectual property (IP) rights related to the software games and apps that they develop. Basics of software *licensing* and *assignment* are covered. For reference, software licensing is the allowance of others to use the developed software at no cost. At the beginning of the semester, students are encouraged to license their software both to the families involved in the course, and to CS Mott Children's Hospital.

At the end of the semester, students are asked to consider assigning their IP to the university, the hospital, and / or to a non-profit company that exists as a 'holder' for the IP. Of course, the assignment is decoupled from course grade assignment, so as to eliminate the possibility of conflict of interest. If the IP is assigned, then the university Office of Technology Transfer becomes involved to try to further develop the software product. If the IP is not assigned (that is, the students maintain possession of the IP), then the Center for Entrepreneurship becomes involved to create opportunities for the student groups to further develop the business concepts around their software ideas.

Finally, because of the increasing number of software products developed by students in both courses, a corporate entity is currently being developed to 'hold' the IP. The corporate structure will be either a non-profit or limited profit corporation. This provides a home for the software games and apps should the students decide that they are going to work in corporate America and have no need / desire to further develop the software product.

### D. Unanticipated Secondary Effects

Clearly, the primary goal of ENG100 and EECS481 is to help children with cognitive and physical disabilities. A lower threshold of success is simply that the students in the courses are more aware of differently-abled children. The higher threshold of success is creation of viable products, and the corporate structure to support ongoing development and maintenance efforts.

There have been many unexpected secondary effects of our efforts. First, many large corporations have become aware of our efforts and have donated equipment (or otherwise become involved) in the cause. As an example, these companies have donated the following types of equipment: Kinect sensors from Microsoft; 3D cameras from Intel; iPads and MacBooks from Apple; and access to Watson from IBM.

Simply stated, we are equipment-rich for our efforts and are constantly reviewing the literature and internet for new equipment that might be useful, such as the Makey-Makey that was mentioned earlier, and Android wearable devices.

Finally, these efforts have had the unanticipated benefit of garnering much community goodwill and public relations coverage. Some of the public relations coverage has included interviews on public radio [10], nationally televised info-mercials during UM sporting events [11], and an invitation to deliver a TEDx talk [12], and coverage in alumni and fund-raising efforts. Success begets success, and although this coverage was not a planned consequence of the efforts, it has resulted in more corporate and alumni awareness (and thus more donations of equipment, …).

### E. Open Questions

There may be some additional open questions about attainment of student learning outcomes and collaborative learning. Many of these important research questions have been addressed anecdotally, rather than quantitatively. Students offer reflection that "… this course taught me much about the real application of the engineering process to help people".

Students are learning! Students are working in groups, both as freshmen and seniors, to develop real products for children with disabilities that have proven to have significant commercial and/or clinical value. After all, is that not what an engineering education is all about?

## IV. Summary

The metaphor of sailing might be appropriate. A sailor spends a lifetime gaining expertise about the craft, so that she/he can survive and thrive in any weather situation. However, on any given day, the sailor never knows which way the wind will blow.

The path to our current state at UM has been similar. The original intent was never towards success – it was simply to offer a different context to the course material that is taught as part of freshman- and senior-level software development courses. That is, rather than using contrived examples, we wrapped the course context around social good. Those efforts evolved into specific application of using interesting hardware and developing software to aid children with cognitive or physical disabilities at our associated children's hospital.

Based upon our initial unexpected successes, we are building the infrastructure to support potential future successes. We have a consistent approach for each of the classes (Design, Build, and Test). We have a structure to support future development of the software products should they prove commercially viable. The collaborative effort driven by students in developing assistive technology for children with cognitive and physical disabilities has resulted in immeasurable goodwill on the local and broader stage.

## Acknowledgment

The author wishes to thank the *Mott Golf Classic*, *Mott Family Network*, and *UM Center for Research on Learning and Teaching* (*CRLT*) for generously funding this work.

## REFERENCES


[1] http://www.youtube.com/watch?v=CUT-Chcffqc. Video of EECS481 Software Engineering students hacking autism.

[2] http://www.engin.umich.edu/college/about/news/stories/2013/september/software-engineering-class-aims-to-help-one-teen-communicate. Initial article about Grace.

[3] http://www.engin.umich.edu/college/about/news/stories/2013/december/software-class-demos-projects-to-help-one-teen-communicate. Mid-project article about Grace.

[4] dme.engin.umich.edu/grace/#-home. Final article and website about Grace, including video montage.

[5] Turning Student Groups into Effective Teams, by R Brent, R Felder, B Oakley, and I Elhajj. Journal of Student Centered Learning, Vol. 2, No. 1, 2004. http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/Oakley-paper(JSCL).pdf. Paper describing best method(s) for group formation.

[6] Cooperative Learning in Technical Courses: Procedures, Pitfalls, and Payoffs, by R Felder and R Brent. Report to the National Science Foundation, 1994. http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/Coopreport.html. Paper describing elements of cooperative learning.

[7] http://www.makeymakey.com/. Makey-makey website.

[8] Considering Context: A Study of First-Year Engineering Students, by D Kilgore, C Atman, K Yasuhara, T Barker, and A Morozov. In Journal of Engineering Education, Vol. 96, Issue 4, October, 2007. http://www.engr.washington.edu/caee/CAEE_Briefs_PDFs/Considering Context_Kilgore_ICREE07.pdf. Paper describing social context as motivation for URM's (particularly women).

[9] http://forum.engin.umich.edu/2010/11/mobile-communications-technology-for.html. Article about ASK (Assistive Scanning Keyboard) that allows keyboard access to people with fine motor skill disabilities.

[10] http://michiganradio.org/post/these-arent-your-normal-video-games. Audio recording of public radio interview (April 2013).

[11] http://www.youtube.com/watch?v=egt19owIAQI. Video about Grace on Big Ten Network.

[12] http://www.youtube.com/watch?v=GO9HSiUMylE. YouTube video of TEDx talk.