The Use of an Open-Ended Project to Improve the Student Experience in First Year Programming

Carol Hulls, Chris Rennick, Sanjeev Bedi, Mary Robinson, William Melek
University of Waterloo

Abstract *- Prior to 2010, Mechanical and Mechatronics Engineering students at the University of Waterloo were taught an introductory programming course using C++ in first year. Historically, the emphasis was on learning syntax; practising problem-solving was a distant second priority. In addition, many students were noticeably disengaged in lectures, and the assessments used were not authentic.*

Starting in 2010, a course project was implemented to address these concerns. The project was immediately well received by students, as evidenced by a noticeable number of students going well beyond the minimum project requirements and the variety of projects implemented. Since the project was introduced, the students have been able to successfully answer less structured final exam questions. The increase in problem-solving and thinking skills more than offsets the reduction in language-specific facts. The logistics, challenges and resources required to implement a project of this scope will be described.

Keywords: First Year, Open-ended, Project, Programming

1. INTRODUCTION

The curriculum for first year Mechanical and Mechatronics engineering (MME) students at the University of Waterloo (UW) is the typical mix of introductory level courses. One goal for their first programming course is to teach the specifics of how to write code, however the most important goal is to introduce students to problem-solving, dealing with ambiguity and making decisions about design. As an outcome for the course, students should be able to take a problem description and break it down into sub-problems (to be implemented as functions or methods), and to make simple design decisions. In addition, because all engineering students at UW are enrolled in a co-op program, there is the need to give the students the knowledge and skills they need for the workplace.

Prior to 2010, the Digital Computation (GENE 121) course focussed on the language specifics of C++. Most of the design decisions were made by the teaching team, as programming assignment and exam questions carefully laid out the details of the functions and then provided appropriate test cases to be used to test the code. The course was content heavy; students would refer to it as “the C++ course” because syntax was seen as most important. Mechatronics engineering (MTE) students found the course dull but could see the importance of learning the material; mechanical engineering (ME) students frequently asked, “Why are we taking this course?” Exit surveys of fourth-year students often identified it as a problem course. Failure rates, particularly for the ME students, were generally trending upwards. There was a large mismatch between the goals of the course, and how it was delivered and assessed.

In response to these issues, in 2009, the content for the ME offering of the course was changed. An informal survey of the syllabi for other introductory programming courses showed that Queen’s University was using RobotC and the LEGO Mindstorms NXT (NXT) as a platform for its course [1] and that students from all disciplines were finding this programming course interesting and engaging. The GENE 121 teaching team also felt that there was value in ME students learning about programming software to work with sensors and motors. The draft plan was to teach RobotC as a supplement to C++ with the goal of having the students use it as an interesting platform to practice coding decisions, loops, and functions. Over the course of the planning semester, the idea of a course project was introduced. The project was later extended to MTE students’ introductory programming course, which allowed integration between the programming and mechatronics concepts courses.

The development of the course project, including changes made to the course content and delivery, are described in the next two sections. Students were surveyed about the project at the end of the term and upper-year students (third and fourth year) have expressed their opinions with regards to the project in focus groups. The results, discussed in Section 4, show that the project has been very successful. Finally, a discussion of the challenges and resources required in offering a project of this nature are given.

2. INITIAL ACTIVITY

The redesigned GENE 121 course launched in January of 2010 to a class of approximately 135 ME students. The redesigned course covered typical C++ topics such as selection statements, loops and arrays for the first 6 weeks of the 12-week term; RobotC was started in approximately week 7. Four weeks of RobotC assignments were completed with the "standard robot configuration" in the lab. This robot had a two motor tank-drive with an ultrasonic sensor pointed forwards and a touch sensor acting as a front bumper, as shown in Figure 1. After four weeks of using the NXT robots in the lab to solve well-defined problems, the students signed out a robot to start working on their group project.

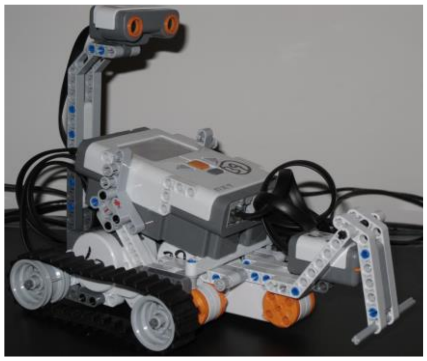
Figure 1 Standard robot configuration

Two term assignments were removed from the course to allow students time in the lab to work on their final project. In the first four iterations of the activity (winter and spring 2010 and 2011), students, working in teams of 3-4 students, were provided with two options: to construct and program one of the three pre-defined project ideas (a Roomba analog, an alarm clock which could hide from the user or a maze-solving robot), or to define their own project. The students had to submit their proposed idea before they were allowed to sign out one of the LEGO parts kits. In the first year, with no previous precedence for what was possible, most students opted to build a maze solving robot. Students were told repeatedly that they should not spend a significant amount of their own time on the project (i.e. outside of lab time), and were aware that the project only carried a 3% weighting (equivalent to two assignments) in their final course grade. They were also told that any mechanical changes should be kept to a minimum, as they would not be provided with any marks for mechanical design/construction. The minimum software requirements for the project were set at an easily achievable level as the teaching team were unsure as to whether the activity would be a success or not. Table 1 gives the initial project requirements.

The deliverables for the project were a list of tasks that the robot would perform, which was used to assess the success of their project demonstration, a demonstration, and a final report. The demonstrations were done in front of their classmates and the teaching team in their final lab session of the term. A few days after their demo, a short (2-3 page) report was due summing up what was accomplished by the team, including a copy of their project software.

Table 1 Initial project software requirements

The use of all three motors, one of which should have controlled movement
The use of the Ultrasonic Sensor
The use of timers, the motor encoders, or both
The use of at least one button or touch sensor
Repetition (i.e. loops)
Decisions (i.e. if statements)
2 non-trivial functions with appropriate parameter passing

Their software was assessed by the teaching team for matching project requirements, meeting course programming style, and for showing appropriate software design. The enthusiasm of the entire class of students was highly evident on final project demo day, buoying the teaching team's confidence that the project was the right choice for the course. A small number of groups also decided to ignore the project suggestions and develop their own idea. One of the memorable projects was a robot "bartender" which used a scissor lift to raise drinks to the user.

After the first semester of the course project, the teaching team noticed that the long break from C++ at the end of the term (two weeks with no assignments, plus typically a week between end of classes and the final exam) was detrimental to student success on the final exam, so for spring 2010 the project was moved forward by one week to allow a C++ assignment in the final week of term. The only other change for the spring implementation of the course was that a different professor taught the course. In both terms, the same three full-time staff members were assigned to the course (for approximately 8 hours a week, each) and handled the majority of the course deliverable grading. Spring 2010 showed an increase in original project ideas, though maze solving robots were still over-represented among the course project ideas. This is likely because the task is easily understood by students and very few mechanical changes were needed as the sensors were already set up quite well to solve this problem. The weighting of the course project in spring was increased to 5% after the initial success of the activity, and to match the reality that students were spending more time on the project than the teaching team intended.

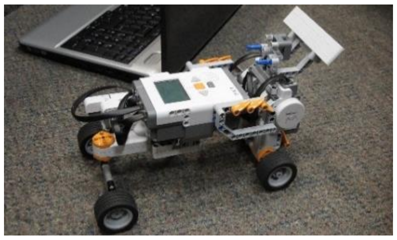
Figure 2 Autonomous parallel parking vehicle (2010)

The project carried on with only minor tweaking in the winter and spring 2011 semesters. The list of project ideas was expanded to include a number of original student ideas from 2010 which proved successful. Figure 2 above shows the first implementation of a project which could autonomously parallel park a vehicle. These new ideas would typically require a significant alteration of the physical robot. Student enthusiasm remained very high throughout the initial four offerings of the course project, and the number of projects where students went above and beyond what was required of them rapidly increased. By this point, the project implementation, from a teaching perspective, was quite mature and required only minor alterations from term to term. The length of the report was increased as the students were required to explain in more detail the design decisions that were made throughout their project.

3. EXPANSION TO MECHATRONICS

In fall 2011, the course instructors saw promise in the project to integrate concepts between GENE 121 and the Introduction to Mechatronics course MTE 100, so GENE 121 for MTE students was rearranged to make room for RobotC and the final NXT project. Like with the ME course, two assignments were removed from the course, along with approximately three lectures to be able to teach RobotC to the students. This necessitated the removal of some advanced C++ topics like vectors and operator overloading, and a slight re-design of their follow-on programming course (taken in their second semester) to reflect the change in student knowledge level. MTE 100 teaches the principles of the design cycle, practicing communication (e.g. verbal, written and CAD/drafting) along with a number of other topics. The hope was that the course project would give the students a meaningful subject for their reports and practice the stages of the design cycle. To meet this goal, the project timeline, deliverables and requirements were all expanded. The minimum project requirements are shown in Table 2.

The course project was tied to the concepts taught in MTE 100 in multiple places throughout the term. The first major deliverable of the project was a new Design Report. This three-page report had to include a description of their project, a discussion of their constraints and criteria, hand sketches of mechanical design alternatives, and finally an application of their criteria to choose the preferred solution. The content of this report was marked for MTE 100 and was a milestone for GENE 121. This design report was due in week nine of the semester, expanding the timeline of the project to four weeks from the original two. As with the ME version of the course, the students demonstrated their working projects in a lab session in week 11, and the now longer 8-12 page final report was due a few days later (the report was made longer to reflect the mechanical design work required).

Table 2 Fall 2011/2012 project requirements

A mechanical re-design of some, or all of the robot and/or an additional mechanical component
The use of three motors, one with controlled movement
The use of at least three different sensors (which can include the motor encoders)
The use of timers
Repetition and Decisions (i.e. loops and ifs)
4 non-trivial functions with appropriate parameter passing

The MTE students were extremely enthusiastic about the project, almost universally coming up with their own project ideas instead of using one of our alternatives. By this point, the list of possible project ideas had been reduced to simply titles with no description of what the project would or could look like.

The success of this expanded project gave us the confidence to run a similar, expanded version with the ME students in winter and spring 2012. Some of the deliverables with strong ties to MTE 100 (like the hand sketching) were removed from the project in favor of software related things like a program flowchart. This term marks the first time that mechanical design was worth any marks in the project for the ME students (it was worth 2/20 marks). Mechanical design was marked for mechanical efficiency and appropriateness to the overall project. This expanded version of the project has continued to be run (with minor changes) up to and including the fall 2014 term.

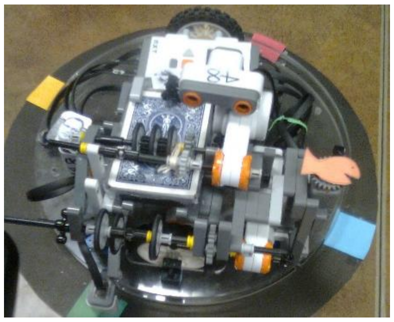

Figure 3 "Go Fish" card dealing robot

Even as the project requirements continued to rise, students still routinely went above and beyond to develop extremely sophisticated projects including robots which could solve Rubik's cubes, play "Go Fish" as shown in Figure 3, CNC milling machines, and scanners and plotters. By fall 2014 the project requirements were significantly higher than in the initial 2010 iteration of the project. These have been summarized in Table 3 below.

Table 3 Fall 2014 project requirements

A mechanical re-design of some, or all of the robot and/or an additional mechanical component
The use of 3 motors, one with controlled movement
The use of at least 4 different sensors which can include communication to a laptop, or an additional NXT robot
The use of timers
Repetition and Decisions (i.e. loops and ifs)
6 non-trivial functions (4 for groups of 3 students) with appropriate parameter passing. Each group member must write at least one function. At least one function must return something and at least one must have parameters.

4. IMPACT ON STUDENT LEARNING

4.1 Survey Results

An online, voluntary student survey was conducted for several offerings of the course at the end of the term, asking students to reflect on the project. MTE students' comments were then cross-referenced with their course grades. For the MTE students, approximately 57% in 2014 provided feedback, while in 2013 the participation rate was approximately 41%. The class was well represented, although slightly more high achieving students and slightly fewer average students completed the survey. Although the survey for the ME students was anonymous, the participation rate was lower with about one third of students participating in each term. The lower participation rate reflects the entirely voluntary nature of the survey. Table 4, below, shows student responses on level of interest for MTE students in both 2013 and 2014.

Table 4 Level of interest (MTE)

	2013	2014
Disliked the task	0	1
Rather have done something else	2	1
Indifferent	2	0
It was interesting	11	24
It was very interesting	47	86

When asked about how the project impacted their confidence level, a large percentage of students in MTE stated that they were more confident programmers after completing the project (Table 5). While the percentage is lower for students who did not perform as well in the course, this seems to be largely the result of letting others in the group handle the programming tasks. The results are not as clear for the ME students, as approximately half of the students felt they were stronger programmers as a result of the project, while many of the remaining students were uncertain (Table 6). Some of the uncertainty is likely due to students being uncomfortable with the subject (many students choose ME over MTE due to a dislike or fear of programming).

Table 5 More Confident after project (MTE 2014)

	Quartile		Overall
	low	high	
No I didn't program the robot	4	0	5
No, it wasn't useful	1	0	5
Maybe	2	3	14
Yes, a little	13	9	54
Yes, definitely	3	13	34

Table 6 Stronger programmer after project (ME, 2014)

	Winter	Spring
Yes, definitely	9	7
Yes, I think so	11	9
Maybe	10	8
No	10	3

Almost all of the students, from weaker students to the stronger ones, found the project interesting. In conversations with students, the teaching team has noted that those students who do not find the project interesting are often the ones who are uncertain as to whether engineering, or more specifically MTE or ME, was a good choice for them. In fact, a discussion of the project can often be the starting point to explore a student's choice of program in the face of difficulties.

Most of the students found the project to be useful, particularly the MTE students, as it encompasses, in a limited way, many of the areas that they will study as an undergraduate student. For these students, the project has always involved both a software and a mechanical design, so its focus has been broader compared to the ME students, who ironically, are given a project that focuses almost exclusively on the software design. ME students do think that there should be an increase in the mechanical design part of the project, and this is being addressed in a new version of the programming course that debuts in 2015.

4.2 Class of 2015 Focus Group Results

The MTE graduating class of 2015 was surveyed in their last term of study. In 2010, in their first term on campus, they were given an Arduino based platform that had the microcontroller board attached to a fixed set of crawler tracks. A reflectance sensor allowed students to be able to perform line following. The project was memorable to the students for two reasons: the unreliability of the platform, and that they were given the small freedom to design their own bumper mechanism. Difficulties with the mechanical construction meant that the wheels literally fell off some of the robots. Incomplete documentation meant that the motor operation was poorly understood and the software to run the motors was poorly written. While the teaching team had forgotten the bumpers, the students still remembered having to construct them to form a touch sensor, almost four years later. They commented that they wished the rest of the project had been more open-ended but that the project did build some confidence in their abilities. Still, while they clearly felt an open-ended project in their first term had been a good experience, they did worry that it would be too much of a challenge to make sure that students were successful. Having not experienced the NXT project themselves, it was interesting to note their uncertainty with regards to the capabilities of first year students.

Finally, when asked, most of the students did not recall some of the specific details of C++, for example vectors and operator overloading, that were covered during their term. They felt that the language specifics could easily be picked up after the course, or that students could be offered this information online as a supplement for any who wished to have more detail. This finding reinforced the decision to cut some of the language-specific details in favour of introducing RobotC and the open-ended project.

4.3 Class of 2016 Focus Group Results

The MTE students who started in 2011, and who are now at the end of their third year, were offered the first version of the NXT project that involved both the design and implementation of software to control the robot, and the design and implementation of a mechanical design as described previously. This group of students loved the freedom of the open-ended project. An earlier exercise in the term, working with a fuel cell car was not as well liked because it was too constrained [2]. For the NXT project, they liked that they were able to get something working to show to others in the class at the end of term [3]. In addition, some students mentioned that working with the sensors and motors was a good introduction to mechatronics systems. One complaint that was echoed by many of the students was that while the project in their first semester had been a good experience, there was a lack of open-ended projects until their sixth semester (i.e. second half of third year). They also complained about exercises in the interim semesters that were "spoon fed" to them.

In their sixth semester at UW, they were again being offered an open-ended project. Many students found this challenging because they stated that they wish there were some sort of transition between the NXT project which was constrained in terms of the equipment used (which was reliable and easily connected) to the challenges of their much less constrained upper-year project. They felt that sometime in their second year would be an ideal time to offer an open-ended project that would allow students to transition to the more challenging requirements. They did not suggest, however, that the third-year project should be made less challenging.

When asked, the third-year students did not feel that they had lacked any C++ specifics when they were taught the course. After being prompted further with regards to specific topics, which corresponded to the ones that had been removed to make room for the project, many stated that they had not needed the knowledge, and amongst those that had used this knowledge they had been comfortable with learning it on their own.

The-third year students unanimously agreed that the project gave them something tangible to put on their resume, and talk about with prospective employers, while searching for their first co-op positions.

Overall, the students found the project to be a positive experience that contributed to their ability to handle uncertainty, and provided them with an enjoyable means of practising programming.

4.4 Instructor Perspective

The course redesign to include the project has been very successful from the instructor perspective as well. During the term, students are challenged to think of how to break down the larger task that they want to accomplish into sub-tasks. This freedom means that they have to make judgement calls as to what a function should accomplish, and what information should be passed to a function and returned from it. How to handle debugging when integrating the system, both for the software and for the NXT components, becomes a real challenge to the students rather than something that is mentioned in lectures and forgotten soon after. On the final exam, the largest question has changed from being very pre-scripted to being open-ended. Students are given a small task, and told to state their assumptions, state any potential issues with having the robot solve the given task, and then to code non-trivial functions and the main program. For example, in fall 2014, the students were given a description as to how the NXT robot might be used, with some modification, to vacuum leaves from the drains on streets, and then they were to write the software to accomplish this task. The course is now meeting the objective of including problem-solving and decision making, and more importantly, it is part of the assessment.

A large, loosely defined project that spans multiple weeks serves as an introduction to the types of projects that students will encounter on their four-month work terms. In addition, the students have to work in teams to complete a task by a specific deadline, where the steps are not laid out for them ahead of time. Issues such as how to handle testing of components and integrating the work of other team members are highlighted for the students, primarily through the realization that they did not take into account these issues and that it would have been better to do so, and to do so early.

While supporting the project is challenging, and leads to a very intense two to three weeks, the time period is also marked by high student-instructor interaction and interesting discussions. Project demo day is the highlight of the term for the teaching team.

5. CHALLENGES AND RESOURCES

The project is not without its difficulties. The first major challenge is one of student and teaching team workload. A number of factors are present here: the scope of the student project(s); the level of performance expectation of the final product; the support required from the teaching team members to give advice on hardware and software design issues, troubleshooting and debugging; and finally the issue of writing (for the students) and grading (for the teaching team) the project deliverables. The second major challenge is true of all group based projects at an undergraduate level: assessing the individual contribution of each team member versus what the team as a whole has completed.

Students in their first year of engineering are generally not capable of setting an appropriate scope for their project without some level of teaching team intervention. At the beginning of our project deployment, before expertise had been built in the teaching team and in the student body (i.e. through information passed on by upper-year students), the projects were much simpler. Through a combination of the teaching team building their knowledge base of what is possible, and through the competitive nature of the students wanting to "one-up" the previous class(es), the complexity has increased. This growth has been encouraged, and requirements have been increased to match. The scope of the students' project is investigated at multiple points throughout the term by the teaching team, and any issues are addressed as soon as possible. The first layer of this check is performed when the students submit their initial one paragraph description of their project. At this level it is possible to catch problematic and/or obviously unsafe projects like "We would like to build a blimp which will require 2m^3 of helium to fly". Once the initial idea has been vetted by the teaching team, further "sanity checks" of the project are performed with each project deliverable. The students have always been allowed to revise their project idea right up to their final demonstration day. One other method of helping students scope their project is to allow them to use "off-the-shelf" software or hardware, developed by others, within their project. Project requirements are applied only to the work done by the students' themselves in these cases.

To further aid the students in setting an appropriate project, the teaching team has always told a consistent message about performance expectations of their final product. Because this is a first-year project, and in many cases this is the first complete product that the students have designed, built and tested themselves, the teaching team allows students to "nudge" their projects where needed. The students are not expected to have in-depth signal conditioning on their sensor inputs, for example. For their final demonstration, the project does not need to work the first time, nor does it need to show all aspects of its performance in one run.

The nature of a student-driven, open-ended project means that the teaching team needs to be comfortable with ambiguity, with dealing with a wide spectrum of issues and questions ranging from hardware design decisions to control systems questions to programming problems, and be able to debug issues in all these areas. The teaching team for this project is made up mostly of full-time staff members, with an occasional undergraduate TA mixed in when the class size is particularly large. This consistent set of people has meant that the teaching team has been able to build on past experiences to better judge when they should be intervening with a student group, and to make suggestions as to what is possible. Without consistency in the teaching team make-up, some method of passing this information from term to term would be required.

The teaching team have used two methods to control the amount of time students will invest in the written project deliverables. The first is a hard page limit on both the introductory design report and the final report. The second method is using the concept of a telescoping report [2]. Since the introduction of the initial design report in fall 2011, approximately the first third of the final report was the content of their initial report. The students are then able to incorporate feedback and lessons learned throughout the project, to improve these sections from their initial submission without having to write them from scratch for the final report.

Like all team based projects, it is difficult to assess the contributions to the project made by specific individuals. One step we have taken towards this end is to require each student to write one function, and the author of each function needs to be labelled in their final code submission. While it would be an academic offense for the group to be dishonest about this, it is relatively difficult to prove any dishonesty. The course compensates by requiring that students obtain a 40% on their final exam before their midterm, project and assignment grades count towards their final mark.

5.1 Resources

The resources used to run this project for class sizes of between 100 and 200 students are summarized in Table 7 below. Attrition for the major, costly components has been quite low over the last five years. There have been few hardware failures: we are still using the original rechargeable battery packs, for example. We are quite careful when students are returning the kits to us to count the major components and make sure they are all present. Otherwise, students must reimburse us for the missing parts. The small pieces, on the other hand, are nearly impossible to count and track so losses are expected.

Table 7 Required Resources

Ongoing Resources	One-Time Purchases
RobotC annual license	1 LEGO Mindstorms NXT set for every 3-4 students
1 faculty member	**Optional** (but recommended): 1 additional set of LEGO construction materials for every 3-4 students
2-4 technical staff, 8-12 hrs/wk.	
0-2 undergraduate TAs	**Optional:** Additional 3rd party sensors (e.g. HiTechnic gyro sensor) [4]
In fall terms: 3-5 undergraduate TAs with MTE 100	

In addition to the physical and personnel resources required, the public demonstration day has also required a lab large enough to hold the entire class (usually just half the class in the case of ME) and all of their projects. Implementing the final demo day in this format has added to the success of the project over the years as the students routinely have not seen what their classmates have been working on in advance of this day. While it would be possible to implement the demo day on a smaller scale, especially if a large space were not available, it would weaken the impact of the project for the students [3] [5].

6. CONCLUSION

The increase in student enthusiasm, and the corresponding increase in the student perception of programming was worth all the effort to incorporate the final project into GENE 121. From our focus group sessions with upper-year students, it was clear that the content that was removed to make space for the project was not missed. Also from these sessions, students who did not complete the project noticed an absence of open-ended problems in their early years in the program.

Having students face and deal with ambiguity, test their own systems, and communicate their ideas in a classroom setting can be quite challenging. The course project, as it has evolved over the years, has addressed these quite well, if only at an introductory level.

The project will continue to evolve and be delivered to students in Mechanical and Mechatronics Engineering in their first programming courses. Data will continue to be collected to measure the efficacy of this project.

ACKNOWLEDGEMENTS

Thank you to the Department of Mechanical and Mechatronics Engineering, and specifically the Chairs Jan Huisson and Pearl Sullivan for supporting the project continuously over the last five years.

The in-class support provided by June Lowe, Fiona Yiu, David Lau and Michael Stachowsky was invaluable. The projects would not be what they are now without their help.

Thank you to the many WEEF TAs who have aided students in completing their projects.

Thank you to WEEF for their support of the project.

REFERENCES

[1] M. Greenspan, K. Rudie and S. Simmons, "Introducing Computer Programming with Lego Robots," in *Proceedings of the 2010 Canadian Engineering Education Association (CEEA) Conference*, Kingston, 2010.

[2] C. Hulls, C. Rennick, M. Robinson, W. Melek and S. Bedi, "Integrative Activities for First-Year Engineering Students – Fuel Cell Cars as a Linking Project Between Chemistry, Mechatronics Concepts and Programming," in *Canadian Engineering Education Association Conference*, Canmore, 2014.

[3] G. Kuh, High-Impact Educational Practices: What They Are, Who Has Access to Them, and Why They Matter, Washington: AAC&U, 2008.

[4] "HiTechnic," [Online]. Available: www.hitechnic.com. [Accessed 21 April 2015].

[5] G. Kuh and K. O'Donnell, Ensuring Quality & Taking High-Impact Practices to Scale, AAC&U, 2013.