OFFICE OF EDUCATIONAL ASSESSMENT
Improving Learning Through Assessment

OEA Report 16-02

UW Academic Challenge and Engagement Study (UW ACES):

Computer Science and Engineering

Catharine Beyer, Office of Educational Assessment
Crystal Eney, Director of Student Services & Raven Alexander[1] and Elise Dorough, Academic Advisers
September 2015

INTRODUCTION

Research on learning in college shows that learning is profoundly shaped by the goals, practices, cultures, and values of the academic disciplines[2], particularly the disciplinary practices in students' majors. Therefore, if we are to understand the kinds of experiences that students find intellectually rigorous (and, thus, engaging), we need to examine challenge in the major. Understanding challenge in the major is important because at every stage of their college experience, students report that they want to be challenged, that they perform better in courses that are challenging, and that they value classes that stretch their thinking and ask them to demonstrate learning more than they value classes that ask little of them.[3] Although learning about where students experience challenge is important, asking students to describe challenging learning experiences in their majors requires some prior understanding of how those majors operate. The Office of Educational Assessment (OEA) designed the UW Academic Challenge and Engagement Study (UW ACES) to accommodate these needs.

METHOD

Qualitative methods are recommended when researchers are seeking to understand the complex learning experiences of students, as well as the meaning they ascribe to those experiences[4]; therefore, we designed the UW ACES to be primarily an interview study.[5] Using a "citizen science" model, OEA asked departmental advisers if they would be willing to volunteer to interview seniors in their departments who came in to advising to apply for graduation. Advisers are knowledgeable about their academic programs, understand disciplinary practice in their departments, and are trusted by students in the major, so they have the best chance of gathering good information from seniors about their experiences in the major.

Sixty-six advisers from 32 undergraduate programs volunteered to participate. During the 2012-13 academic year, the volunteer advisers asked students if they would participate in brief (5-10 minute) interviews about challenge in the major. If the students agreed, advisers asked them to respond to four open-ended questions, entering students' responses directly into a Catalyst survey form that OEA researchers had designed for that purpose. The questions were as follows:

---

[1] Raven Alexander provided the first coding of all interview responses. We are very grateful for her help and insights.
[2] Beecher & Trowler, 2001; Bransford et al., 2000; Beyer et al., 2007; Donald, 2002; Pace and Middendorf, 2004; Wineburg, 2001, 1991; Neumann et al., 2002; Shulman, 1988; Biglan, 1973.
[3] Beyer, et al., 2007.
[4] Merriam, 2001.
[5] One participating department asked students to respond to the open-ended questions in writing.

1. What do you consider to be the most challenging work that you had to complete in this major? And by "challenging" I mean doing the work that stretched your thinking the most. This can be anything—a project, a paper, an exam question, homework, something else you did related to the major.
2. What made the project/class/activity challenging?
3. What did you do or learn that enabled you to meet those challenges?
4. What do you think you learned by completing this project/class/activity?

In addition, advisers asked students in what course the challenging work took place and how many quarters they had until they graduated.

Researchers in OEA conducted training workshops in interviewing skills with all participating advisers, provided individual departments with survey customization if required, and monitored all resulting interviews, reporting back to advisers about the interviews they had conducted. By the end of the academic year, departmental advisers had interviewed 1,237 students, about 17% of the total 2012-13 graduating class. Students' responses were analyzed using a constant comparison method[6], an inductive process designed to let themes emerge, rather than imposing assumed categories on students' comments.

STUDY LIMITATIONS

If we interviewed students post-graduation, they would be likely to identify their capstone courses or their advanced senior-level courses as the ones asking for their most challenging work. However, because we wanted to attach the interview to a time when students would normally see their academic advisers, we interviewed students when they came into the advising office to apply for graduation, which often meant that they were two or three quarters away from graduation. Senior-level courses, particularly capstone or capstone-like classes, are those which students often say are their most challenging and satisfying. Although interviewing students as they applied for graduation meant that we might not gather information about late-senior year courses, we felt that it would be interesting to departments to learn the kinds of challenges that lead to and prepare students for those more advanced experiences.

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT RESULTS

The Computer Science and Engineering (CSE) Department was one of the UW ACES' 32 participating departments. Crystal Eney, Raven Alexander, and Elise Dorough asked 40 students if they were willing to be interviewed for the study and all of them agreed. The 40 students who were interviewed represented about 21% of the189 seniors in CSE who graduated with bachelor's degrees during the 2012-2013 school year.[7]

Students were asked which academic pathway in the major—Computer Science or Computer Engineering—they were completing for their undergraduate degrees. Thirty (75%) of the interviewees said they were enrolled in the Computer Science track and 10 (25%) said they were in the Computer

[6] Merriam, 2001.

[7] The number of undergraduate degree completions is based on the 2012-13 UW Profiles reports published by the UW Office of Planning and Budgeting (https://uwprofiles.uw.edu/Viz/View/13-SummaryandTrendsDegreeAttributes)

Engineering track. Even though the number of Computer Engineering majors is relatively small, when there were differences in the responses of students in these two areas of concentration, we note them in this report.

Quarters to Graduation and Where Students Experienced Challenge

The majority of the interviewees had two (45%) or three (43%) quarters to complete before graduating. About 12% were graduating in the same quarter that they interviewed. As Table 1 shows, a greater percentage of the CSE interviewees in the Computer Engineering track had a full year to complete than did students in the Computer Science track.

Table 1. Time to degree at point of interview

Quarters to graduation	All N=40	Computer Science N=30	Computer Engineering N=10
1	12%	13%	10%
2	45%	50%	30%
3	43%	37%	60%

Ten (25%) students listed more than one course as the site of their greatest challenges. Altogether students identified 17 courses in the major as the sites of their greatest challenges. About half of them were 300-level courses; however, the course mentioned most frequently was CSE 451, identified by 25% of the interviewees as the site of their most significant challenge, followed by CSE 331, mentioned by six (15%) of the interviewees . The list of courses and the number of students mentioning each course (if more than one student noted it) is as follows:

CSE 311 (2)
CSE 312 (5)
CSE 322
CSE 331 (6)
CSE 332
CSE 333 (2)
CSE 351 (3)
CSE 352 (3 one with Oskin)
CSE 370
CSE 401
CSE 403 (2)
CSE 421 (4)
CSE 440
CSE 446.
CSE 451 (10)
CSE 461
CSE 499

In addition, individual students mentioned the following as the sites of their most significant challenges in the major:

- An internship
- ETH courses during study abroad, especially algorithmic game theory
- Spring 2011
- STAT 390
- The whole program, no specific course

Differences by track between the courses mentioned are shown in Table 2. A greater percentage of Computer Science majors identified 400-level courses (including CSE 451) than did Computer Engineering majors. This is not surprising as a greater percentage of the interviewees in the Computer Engineering concentration had a full year of coursework to complete than the percentage of Computer Science majors who had a year to graduation.

Table 2. Courses in which most challenging work took place by academic track

Computer Science N=30	Computer Engineering N=10
CSE 311 (2)	CSE 312 (2)
CSE 312 (3)	CSE 331 (4)
CSE 322	CSE 332 (3)
CSE 331 (2)	CSE 421 (2)
CSE 333 (2)	CSE 451
CSE 351 (3)	
CSE 352	
CSE 370	
CSE 401	
CSE 403 (2)	
CSE 421 (2)	
CSE 440	
CSE 446	
CSE 451 (9)	
CSE 461	
CSE 499	

1. Students' Greatest Challenges

Students were asked: "*What do you consider to be the most challenging work that you had to complete in this major? And by "challenging" I mean doing the work that stretched your thinking the most. This can be anything—a project, a paper, an exam question, homework, something else you did related to the major*." About 30% of the interviewees gave more than one response to this question.

Two strong themes and a few minor themes emerged from students' responses.

A course or courses. The majority of interviewees (70%) identified a whole course or courses as presenting them with their greatest challenges in the major. Two of the 28 students who gave this response also said that any math-related courses they had taken were challenging for them, and both of these students were Computer Engineering majors. The following quotations illustrate this category of response:

- *This OS class [CSE 451] is blowing my mind. So much information and things going on that you have to keep track of. Really eye-opening to see what all is going on to make a computer work.*
- *The one that jumped out at me was, of course, CSE 331. I guess OS (CSE 451) would also be a challenging course.*
- *In terms of workload—331, with Professor Ernst. A good experience. [It was my] first quarter in the department, and the standard of quality expected was very high. It was very theoretical for a software engineering class. Also, Algorithms, which was deep,*

pretty advanced math that I wasn't used to yet, taught by Professor Ruzzo. It taught me how much I didn't know, more than anything.

- *I'd say it would be a class—CSE 312. It stretched my thinking because it revolves around statistics, that's what they are trying to teach. I've never taken a stat course, and it was presented in a way that didn't quite match my learning style—pure power point, vs. whiteboard. Whiteboard is more dynamic.*
- *Computational theory class. It was nuts. Turing machines. It was rad, really cool. I was just explaining to someone about non-computable numbers the other night.*
- *Everything after Intro, but, probably CSE 403, Software Engineering, or CSE 331, Software Design & Implementation.*

A project/projects. Thirty percent of the interviewees identified a project as their most challenging work in the major. For example:

- *I think I would say the most challenging project was the main course project in CSE 352. It's the first thing that comes to mind.*
- *Difficult question. It could be everything! Full projects are the toughest. The early designs affect how the project turns out. Early on you have to figure out how the system will work.*
- *The compilers project was pretty challenging.*
- *In CSE 403, the project we did went into unfamiliar territory with the android project. No one in our group of four had done that, and we had to connect to the database.*
- *It would be a tie between the HCI project where we redefined team work for me, and the hardware course, CSE 351, which was a very different way of looking at computers.*
- *The Husky map project for Professor Ernst's CSE 331 class.*
- *The OS [CSE 451] projects. All of those were the most difficult. I spent the most time on them.*

Other. Two or three students identified the following as their most challenging work in the majors:

- Course concepts, such as probability and parallelism (3 students, for example: *"Probability from CSE 312."*)
- Course requirements, such as homework and exams (3 students, for example: *"Homework for Machine Learning, CSE 446.")*
- Time management and prioritizing work (2 students, for example: *"I think it largely had to do with time management.")*
- Using C programming language (2 students, for example: *"Transitioning from Java to C programming language…")*

Finally, individual students identified the following as challenges:

- Research
- Coursework taken during study abroad experience (ETH)
- An internship

Differences in responses of Computer Science and Computer Engineering students. Table 3 shows the differences in the responses of students in the two academic paths in the CSE major. As the table shows, students in the Computer Engineering track identified courses as challenges more frequently than did those in Computer Science. In addition, Computer Science interviewees identified more kinds of challenges than did Computer Engineering students.

Table 3. Greatest challenges of Computer Science and Computer Engineering interviewees[8]

Greatest challenge	All	Computer Science	Computer Engineering
Course(s)	70%	67%	90%
A project/projects	30%	30%	30%

2. What Made Those Activities/Classes Challenging?

When asked what had made the activities they had described especially challenging, about 78% of the 40 interviewees identified more than one aspect of the activity. Four strong themes and several themes with a moderate or small amount of agreement across interviewees emerged from students' responses to this question.

The time the activity required. One third of the interviewees noted that they had spent a great deal of time on the course or project they had described. Often they noted that they had to complete that work on a tight time schedule. The following quotations illustrate this category of responses:

- *CSE 331 was grandiose in too little time.*
- *The amount of work and the amount of time, and that it was my first quarter. And, strict, thorough grading. How code was structured mattered as well as output.*
- *It moved way too fast. I felt like I was from a different planet. Vocabulary, terms—we were not speaking the same language. I passed, but it was not something to be proud of.*
- *It makes you put in a lot of thinking hours, and a lot more hours on the projects, so it's an intense course, but they are not grunt hours.*
- *It was intense and jam-packed. Hours and hours of studying.*
- *A couple assignments [in that class] took more time on one assignment than I took on anything ever.*

The subject/concepts were unfamiliar. Thirty percent of the CSE majors we interviewed said that the challenge in the activity they had described was that the subjects or the ideas under study were unfamiliar to them. Several of these students also noted that they had to teach themselves the background information in that subject in order to complete the work required. The following are examples of this category of response:

- *There was a lot of background knowledge assumed that you would acquire on your own, background research and math that you were assumed to be able to self-teach.*

[8] Percentages do not add to 100% because students sometimes identified more than one challenge.

- *I had to spend a lot more time teaching myself outside of class. It stretched my thinking because it revolves around statistics. I've never taken a stats course.*
- *I guess it was not something I had seen in coursework or personal work. It was such an abstract concept, it took awhile to get my head around it. It was fun though.*
- *The asynchronous programming. It was only the second programming project I'd done with concurrency and asynchrony. It added a lot of complexity.*
- *The amount of work assigned, and the jump in material. [It was my] first time coding a really large project and using new tools (source control and editing things to the console). I was using a lot of new things that I ended up using in 400-level classes.*
- *The grad course was very mathematically intense and designed for people with math and biology backgrounds, and they assumed you knew a lot about the brain, which I didn't.*

Building a project from scratch /figuring things out on your own. About 28% of the interviewees said that the challenge involved in the activity they had described was building something from the ground up on their own. These students spoke of the challenges in the decision-making processes required to complete such work, the challenge in the open-ended nature of many of these projects, and the fact that decisions made early in the process affected the success of the final project when it was too late to go back and revise their work. In the words of some of these interviewees:

- *It's moving beyond coding and rules and you get into specification, style, and more general approach. [That's challenging] because in most of your other courses you have the skeleton and you just fill in the topics for whatever elements you're discussing*
- *The whole idea was to build a processor from scratch—applying Boolean algorithms, applying simple concepts to make the project work, creating adders, pointers.*
- *I think the transition from the intro courses where you are given a rigid specification and then now a high level design spec. Having the liberty to make real nitty-gritty design decisions is nice, but you are working with a team and need to spend a long time deciding what to do and revising. There are a lot of design meetings.*
- *It was that you're building something that you normally just have, and you have to figure out the best way to do it, but all you have are the bare minimum building blocks.*
- *The method is relatively straight-forward once you know what the problem is, but it was figuring out the problem that was the crux of the question.*

A new way of thinking. Ten (25%) of the interviewees said that the activity they had described was challenging because it presented them with a new way of thinking, and four of this group said that the new thinking they were required to do was more abstract or theoretical than they were used to. For example:

- *It was essentially an entirely new way of thinking about things, nothing I had ever encountered before. Also, the size of it. It took the entire quarter.*
- *It was different than anything I had done before, a different type of thinking. You have to be more aware of resource management, so that threads don't step on other threads'*

toes. You have to think about the algorithms from multiple perspectives vs. a more linear approach. With multi-threading, you can't depend on OS scheduling certain things at certain times. I found it really cool because you had to program with a different mentality with multiple threads.

- *Probably operating system because it was a different type of thinking.*
- *You have a set of tools, but the answer's not obvious. It required a different way of thinking than I was used to.*

Problems with the class—its structure or the amount of instruction given. About 15% of the interviewees identified problems with the way the course was structured that made their work in it challenging. As three students noted:

- *CSE 331 wasn't structured. The professor utilized the MIT curriculum and crammed that into eight weeks. The material content and the project content didn't meld. You learn good design practices, but you learned late in the game.*
- *The style of the class [was challenging]. A lot of showing examples and not as much written instructions about what to do.*
- *Instructions were vague at best, and there were often unexplained formulas, and homework did not really line up with lectures.*

The math. Six (15%) of the students said that what made the course, project, or other activity they had identified as challenging was the mathematics involved. In the words of three of them:

- *Programming is really easy, so it was just as difficult as math normally is. Everything else was easier.*
- *Also, Algorithms—deep, pretty advanced math that I wasn't used to yet. It was taught by Professor Ruzzo. It taught me how much I didn't know, more than anything.*
- *There was little programming, so it's all about writing proofs, proving theorems, and applying them to application problems. I think it's very different than other courses, because it's less programming and more mathematical. At the same time I find it one of the most useful because as a software engineer, data structures and algorithms are top priorities, and I don't think a lot of students realize that.*

Working in a team. About 13% of the interviewees spoke of the challenges of working in a team. For example:

- *The OS study itself and the project was very hard. In the group project, there was some conflict with other team members so that was also challenging. We had to split the work and combine it later on. One of my teammates didn't communicate, so we didn't know what his progress was.*
- *Coordination [in the team] as well as figuring out the pros and cons with the tools we needed to decide on. A lot of times we would pick some software and then find out that it could have gone a lot more smoothly if we had gone down another path, but at that point we had already frozen it. Communication is key as well as coordinator.*

- *Having to coordinate with that many people and that we were all on the same page and all getting along. One of our teammates was deaf, so that too was tricky. We had to be careful of fragmenting and going in different directions. Overall I think we did a good job.*

Use of an unfamiliar or disliked programming language. Four (10%) of the interviewees mentioned a programming language that they either disliked or were unfamiliar with that made the work they had described challenging. As two of them said:

- *Feeling lost with C (programming language).*
- *The fact that I had to use Java. Aside from [having only a] basic knowledge of Java, Java's exception handling was more difficult than in 143.*

Other. Two or three of the interviewees identified the following as particularly challenging aspects of the work they had described:

- The broad scope of the project (3)
- The lack of documentation or online information available for working with an operating system (3)
- High standards (3)
- Lack of interest in the subject (2)
- Not as good at hardware as software/does not like hardware as much as software (2)

Finally, one student said that working with metadata was particularly challenging.

Differences in responses of Computer Science and Computer Engineering students. Table 4 shows the differences in responses of Computer Science and Computer Engineering students to the question of what made the activity they described challenging. As the table shows, interviewees in the Computer Science track found unfamiliar subjects/concepts more challenging than did Computer Engineering interviewees. The latter group found that building things from scratch and new ways of thinking to be more challenging than did Computer Science students.

Table 4. What made activities challenging, Computer Science and Computer Engineering majors[9]

Challenge	All N=40	Computer Science N=30	Computer Engineering N=10
Time required	33%	33%	30%
Unfamiliar subjects/concepts	30%	33%	0%
Building from scratch/figuring things out on own	28%	27%	40%
New way of thinking	25%	17%	50%
Problems with the class structure	15%	17%	10%
The math	15%	13%	20%

[9] Percentages do not add to 100% because students sometimes identified more than one challenging aspect of activities.

3. What enabled students to meet those challenges?

About 48% of the interviewees identified more than one source of help for meeting the challenges they described. For example, this student speaks of getting help from linking previous knowledge with current demands, from team mates, and from the professor:

> *A lot of things [helped]. The biggest thing was being able to read a high level spec and connect the dots to figure out which of my skills that I've learned would fit here. Thinking about the pros and cons and, of course, interacting with members of the team and coming to agreement. Putting differences aside when you don't have the same vision. There were no specific resources [that helped], but group mates and the professor helped.*

Overall, three strong themes and several minor themes emerged from interviewees' responses to this question.

My own efforts. More than half of the interviewees (58%) noted that their own efforts helped them meet the challenges they described. The efforts students identified included a wide range of activities, including reviewing coding technique, reading and rereading the textbook, using homework to improve understanding, looking online for additional information, asking questions, focusing on coursework for extended periods of time, and improving time management skills. Two of these students mentioned persistence and refusing to give up on a problem. The following quotations illustrate this category of response:

- *In order to do proofs, you just need to sit down and spend a significant amount of time to get it done. Switching back and forth between things does not help at all. Multitasking while trying to prove things does not work as well as multitasking while programming.*
- *Learning how to use the software we had to use.*
- *Hours and hours of studying. Working over the slides over and over again.*
- *Homework. I homeworked the living hell out of it. Homework saved me in everything I've done in school. My brain is not on the same fire as a 19-year old. I have to look at problems multiple times, burn more time than others. I spend lots of time on assignments.*
- *Living in the library. I guess I made friends with really good study habits—just putting in the time.*
- *I learned to ask dumb questions. That is the core. I realized that everyone was there at some point. Eventually it gets beaten into your head.*

Peers/working effectively in groups. More than a third (35%) of the Computer Science and Engineering majors interviewed said that they had received help from their peers and from effective group work. In the words of six of them:

- *Made friends and learned together. It was a real group effort.*
- *Don't give up, just keep working on it. There's like a lot of really small problems that were hard to stop. I actually ended up discussing it with a lot of people who were stuck and we worked it out in team work even though we were supposed to submit*

individually. In a three hour lab, the students stayed after the TA's all left, so we helped each other.

- *I guess pulling from what everyone else knows and what they research, asking them questions. Strong communication [helped].*
- *We had a lot of meetings. Apparently we met more than most teams in the class, talked online, coordinated everything. Everyone was pretty competent. Due to weekly meetings we could figure out what worked and what didn't.*
- *Talked to lots of classmates, so that together we're less confused.*
- *Learned to read through a ton of code. Looked it up on the internet. I had a partner and we both tackled different parts and explained it to each other.*

Faculty/TAs/Office Hours. Twenty percent of the interviewees said that they had received help in meeting challenges from faculty and TAs. For example:

- *There are multiple functions that do the same thing and you have to know which ones the windows developers would use. I kept asking Gary and Mark about the functions, what the differences were.*
- *A lot of the students help out and professors are always helpful, and the TA, of course. Pretty much everyone.*
- *I talked to professor when I had trouble.*
- *Talked to friends and worked together, and went to professor's office hours.*
- *Spent a lot of time working with the professor (Rajesh Rao) who taught the course. He helped guide the project.*

Trial and error. Thirteen percent of the majors whom we interviewed said that the trial and error process was important to their learning to meet the challenges they had described. As three of them noted:

- *Trying and failing a lot of times.*
- *We had lots of samples in class, and that was helpful. We went over and proofed stuff in class to learn techniques that way. Trial and error, getting feedback.*
- *In 352, working through a problem and trying different things. Lots of trial and error.*

Other. In addition to these themes, two or three students said the following had helped them meet the challenges they had described:

- Good instruction in class (3)
- Linking the present case to previous experience/skills (3)
- Learning to identify the problem first before trying to solve it (2)

Finally, individual students gave the following responses regarding what had helped them meet challenges:

- Learned how to use software needed to complete project
- Learned to spend more time planning options before starting project

- Broke the project into smaller problems
- *Nothing! I didn't learn to meet this challenge.*

Differences in Computer Science and Computer Engineering students' responses. As Table 5 shows, a greater percentage of the Computer Engineering students credited faculty and TAs with helping them meet those challenges than did Computer Science majors.

Table 5. What helped Computer Science and Computer Engineering interviewees[10]

Source of help	All N=40	Computer Science N=30	Computer Engineering N=10
Learned/used study habits	58%	80%	80%
Peers	35%	43%	50%
Faculty/TA	20%	17%	40%
Trial and error	13%	10%	20%

3. What did students learn by completing this project/class/activity?

We asked students what they felt they had learned by meeting the challenges they had described, and about 45% of the 40 interviewees mentioned more than one lesson learned. One strong theme emerged from their responses, along with a number of responses given by fewer than five interviewees.

Content/concepts. Twenty-one (53%) of the Computer Science and Engineering students we interviewed mentioned aspects of course content when asked what they had learned by meeting the challenges they had described. Six of these students spoke of learning more about algorithms; three spoke of learning more about programming; three mentioned learning hardware concepts; and three mentioned learning OS concepts. In addition, students spoke of learning about probability, neuroscience, design skills, and other topics. The following serve as examples of this category of response:

- *Algorithm analysis and a lot about probability.*
- *Other major programming languages other than java, a better understanding of lower-level, how to use GDB (the debugging tool for C), better understanding of how systems work.*
- *Some of how the mobile development framework works under android*
  Learning a lot about hardware and low-level assembly.
- *The inner working of the OS in general, specifically for Windows.*
- *Learned to look through code, a little bit of C. I really felt like I understood how operating systems worked. They aren't a mystery anymore, which was pretty cool.*
- *I learned about neuroscience.*

Personal gains. Eight (20%) of the interviewees noted that they had made personal gains in meeting the challenges they had described, including a stronger work ethic, better time management, confidence in

[10] Percentages do not add to 100% because students sometimes identified more than one help in meeting challenges.

their own abilities, and learning to deal with difficulties. When asked what they had learned, three of them said:

- *Don't give up.*
- *I learned how to deal with difficult situations, that not every situation will fit my needs, and I have to do something about it to survive.*
- *Since we went through a lot of challenges, it gave me the courage to take other courses.*

What goes on underneath the surface of computing. About 13% of the interviewees said that they had gained insights into how computing works. In the words of three of these students.

- *I learned a lot from doing that assignment. One of the top things was that it that gave me a new perspective on computing; specifically, it taught me a lot about what goes on underneath the surface.*
- *I learned how a computer works from the bottom up—build from the simple signals all the way up to how processors work.*
- *I learned how to really translate the theory behind an OS into practice, and to learn more about how the operating system works. They are so fundamental that knowing how they work underneath is really interesting.*

How to complete a project from the ground up. Ten percent of the interviewees said that they had learned how to complete a project from start to finish. As two of them put it:

- *I think I'm learning, so if I have task to do outside of college from the ground up, I'll have a process to work effectively to complete the task.*
- *Just the ability to take on large and vague products and create something concrete out of them.*

How to work with others. About 10% of the interviewees spoke of having learned to work with others effectively. For example:

- *I learned to work with people.*
- *Definitely how to work better in a team. How to coordinate and make sure you are taking on your fair share without under- or over-shooting. And also coordinating to make sure everyone else is getting their work done too. I learned to trust other people too.*

Other. In addition to these themes, two or three interviewees gave the following responses:

- Found a course/career I was interested/not interested in pursuing (3)
- Realized my own weaknesses, such as needing better study skills (3)
- Learned new problem solving strategies (2)
- Gained a better understanding of CSE field (2)
- Learned to translate theory to practice (2)

Also, individual students identified the following as what they had learned by meeting the challenges they had described:

- To ask questions
- How to connect skills to the task at hand
- How to manage information quickly
- The value of finding the right answer
- What industry projects are like
- To think of code for its intentions rather than its syntax
- How to reverse-engineer something

Differences in Computer Science and Computer Engineering students' responses. Regarding what they had learned from meeting the challenges they described, interviewees on the Computer Science track were more likely to mention content than those in the Computer Engineering track. The latter group were more likely to note personal gains, such as building a stronger work ethic or persistence, and learning how to complete projects from start to finish than did the students focusing on Computer Science, as Table 6 shows.

Table 6. What students learned from challenges, Computer Science and Computer Engineering interviewees[11]

What learned?	All N=40	Computer Science N=30	Computer Engineering N=10
Content/concepts	53%	53%	40%
Personal gains	20%	17%	30%
What goes on underneath the surface	13%	10%	20%
Complete project from the ground up	10%	3%	30%

SUMMARY

The image of challenges in the Computer Science and Engineering major that emerges from what students described in these interviews included both classes and projects that introduced students to new concepts and practices, that required many hours of students' time and effort (including self-teaching), and that yielded deeper understanding about how computing works. These challenges were spread across 300- and 400-level courses, which suggests that students were meeting challenges throughout the major.

When asked what was their most significant challenge in the major, nearly four out of five (70%) of the CSE interviewees named a course or courses that had presented them with significant challenges. About 30% of the interviewees identified a project.

When asked why those activities were challenging, the most frequently-given responses were that they required a great deal of time, that the subjects and concepts were unfamiliar to students (given only by students in the Computer Science track), that they were building something from scratch primarily on their own, and that the activity required a new way of thinking (given primarily by students in the Computer Engineering track).

[11] Percentages do not add to 100% because students sometimes identified more than one thing learned.

More than half the students said that their own efforts had helped them meet the challenges they had described, often noting putting in long hours, doing and re-doing homework, not giving up in the face of difficulty, asking questions when they needed help, teaching themselves program languages and other technology, and other kinds of work. Students also noted that they had met the challenges they had described because of help from peers—either via study groups, class conversations, or working in teams on projects. Finally, students noted the help of faculty and TAs in their understanding of concepts and development of skills

In terms of what they learned by meeting the challenges they had described, the most-frequently given response was that students had learned new concepts and content. Students also said that they had made personal gains—such as increases in self-confidence, that they had learned about how computers work "underneath the surface," that they learned how to complete a project "from the ground up," and that they had developed better team skills.

Differences between the responses of students in the Computer Science and Computer Engineering tracks suggested that Computer Engineering students' challenges may be a bit more project-focused than content-focused and may rely more on faculty/TA input than the challenges described by students in the Computer Science track. However, because of the small number of Computer Engineering students interviewed, it was hard to know if the differences we noted between the two tracks in CSE were meaningful.

Finally, students' responses to the UW ACES interview questions suggest that Computer Science and Engineering majors felt that the challenges they described dealing with were rewarding. The responses of the following CSE students—as well as those of the two students in the box that follows this summary—underscore research on student learning that shows that when an assignment is challenging for students and when faculty and TAs help students meet those challenges, students become more engaged in the course material than they are when tasks are easy:

- *In 332 last spring, we were introduced to parallelism and concurrency and one of our projects had to do with that. I found it really cool because you had to program with a different mentality with multiple threads.*
- *It was proof-based, but the proofs were not standard math proofs. They were proofs about models of computation. You really had to learn to do proofs in another way that is applicable to CS. You had to think about computation in an abstract way. I thought it was cool.*
- *I guess it was not something I had seen in coursework or personal work. It was such an abstract concept, it took a while to get my head around it. It was fun though.*

Two Students' Responses to All Four Questions

Computer Science interviewee

Course(s) where greatest challenges occurred: CSE 451

Q1. What was the most challenging work you did? I think OS was the most challenging course. I spent most time on that last fall. I learned a lot of C++ skills while taking that class.

Q2. Why was it challenging? The OS study itself and the project were very hard. In the group project, there was some conflict with other team members, so that was also challenging. We had to split the work and combine it later on. One of my teammates didn't communicate, so we didn't know what his progress was. Mostly the CSE courses use Java. There was very little in C++, so that was challenging.

Q3. What helped you meet that challenge? We asked the TA a lot of time to help us with the technical problems. Also we had team meetings and tried to solve the teammate issue. During meetings we talked about it and he got better.

Q4. What did you learn by meeting that challenge? I learned a better understanding of operating systems, how they work, and since we went through a lot of challenges, it gave me the courage to take other courses.

Computer Engineering interviewee

Course(s) where greatest challenges occurred: CSE 332

Q1. What was the most challenging work you did? In CSE 332 last spring, we were introduced to parallelism and concurrency and one of our projects had to do with that. I found it really cool because you had to program with a different mentality with multiple threads. After finishing, we ran on single core vs. quad core and measured the performance.

Q2. Why was it challenging? It was different than anything I had done before, a different type of thinking. You have to be more aware of resource management, so that threads don't step on other threads' toes. You have to think about the algorithms from multiple perspectives vs. a more linear approach. With multi-threading, you can't depend on OS scheduling certain things at certain times. To make sure the code will do what you want with multiple threads working together.

Q3. What helped you meet that challenge? Professor Grossman was teaching the course and he was a great teacher. He took a good approach because he knew most folks didn't know this stuff, so he took a ground-up approach. He explained it well. It was kind of like a self-exploring project where you had to try things to see what worked and what didn't and use results to improve the approach and tweak the algorithms. Trial and error.

Q4. What did you learn by meeting that challenge? The number one thing was that it was something I would like to be doing with my career. I really found it fascinating. Most of the time I'm not looking forward to assignments, but I looked forward to this. It was really fun for me. I learned about a different way of thinking of algorithms more in depth than I had before.

SOURCES

Beecher, T. & Trowler, P.R. (2001). *Academic tribes and territories: Intellectual enquiry and the culture of disciplines.* Suffolk, UK: St. Edmundsbury Press.

Beyer, C. H., Gillmore, G. M., and Fisher, A. T. (2007). *Inside the undergraduate experience: The University of Washington's Study of Undergraduate Learning*. San Francisco: Jossey-Bass.

Biglan, A. (1973). The characteristics of subject matter in different academic areas. *Journal of Applied Psychology, 57*(3), 195-203.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.) For the National Research Council. (2000). *How people learn: Brain, mind, experience, and school.* Washington, D. C.: National Academy Press.

Donald, J. G. (2002). *Learning to think: Disciplinary perspectives*. San Francisco: Jossey-Bass.

Merriam, S. B. (2001). Qualitative research and case study applications in education. San Francisco: Jossey-Bass.

Neumann, R., Parry S., & Becher, T. (2002). Teaching and learning in their disciplinary contexts: A conceptual analysis. *Studies in Higher Education, 27*, 405-417.

Pace, D. & Middendorf, J. (Eds.) (2004). *Decoding the disciplines: Helping students learn disciplinary ways of thinking*. San Francisco: Jossey-Bass.

Shulman, Lee S. (1988). A union of insufficiencies: strategies for teacher assessment in a period of educational reform. *Educational Leadership, 46*(3), 36-42.

Wineburg, S. (2001). Interview with Randy Bass. *Visible Knowledge Project*, Georgetown University, from http://crossroads.georgetown.edu/vkp/conversations/participants/html. Accessed 10/12/06.

Wineburg, S. (1991). On the reading of historical texts: Notes on the breach between school and academy. *American Educational Research Journal, 28*(3), 495-519.