Undergraduate Computer Science

Assessment Report - 2015/2016

Identifying Information

Name of Program: Computer Science

Type of Program: Major

College of Arts and Sciences Division: Sciences

Name/Title/Email of Submitter: Sami Rollins, Professor and (outgoing) Chair, srollins@cs.usfca.edu

Name/Email of Additional Individuals who Should Receive Feedback: David Wolber, Professor and (incoming) Chair, wolberd@usfca.edu

Mission Statement

Students who graduate with a Bachelor of Science (B.S.) degree in Computer Science will be prepared for both graduate school and for software development careers. The curriculum provides a solid base in computer science fundamentals that includes software design and development, problem solving and debugging, theoretical and mathematical foundations, computer systems, and system software.

Program Learning Outcomes

We have updated our PROJECT program learning outcome to incorporate feedback received on our December 2015 report.

- THEORY: Explain and analyze standard computer science algorithms and describe and analyze theoretical aspects of various programming languages.
- APPLICATION: Apply problem-solving skills to implement medium- and large-scale programs in a variety of programming languages.
- SYSTEMS: Describe the interactions between low-level hardware, operating systems, and applications.
- PROJECT: Demonstrate effective communication and organization as part of a team of software developers or researchers collaborating on a large computer program.

Brief Summary of Most Recent Assessment Plan

We are rebooting assessment in Computer Science and have not had a structured plan in place for a few years.

Academic Program Review

Date of most recent Academic Program Review's External Reviewer Visit: November 18-20, 2015

Date of most recent Action Plan Meeting: May 16, 2016

Brief Summary of the most recent Action Plan: The action plan for our 2015/2016 program review is still in progress.

Curricular Maps

The curricular maps for our undergraduate program are as follows:

Our courses curricular map is also available at: https://docs.google.com/spreadsheets/d/1_r8xAepHOVYfDNBwdqf1jo0y1UgGFgLKEgz44QZMqjo/edit?usp=sharing

Our ILO curricular map is also available at: https://docs.google.com/spreadsheets/d/1bMG77J2OhSLLezGDCDazR2dPvdeH1PJCE3QBbIsWR9A/edit?usp=sharing

ILO Curricular Map

	PLO1	PLO2	PLO3	PLO4
Institutional Learning Outcomes X Program Learning Outcomes	THEORY: Explain and analyze standard computer science algorithms and describe and analyze theoretical aspects of various programming languages.	APPLICATION: Apply problem-solving skills to implement medium- and large- scale programs in a variety of programming languages.	SYSTEMS: Describe the interactions between low-level hardware, operating systems, and applications.	PROJECT: Demonstrate effective communication and organization as part of a team of software developers or researchers collaborating on a large computer program.
Institutional Learning Outcomes				
1. Students reflect on and analyze their attitudes, beliefs, values, and assumptions about diverse communities and cultures and contribute to the common good.				
2. Students explain and apply disciplinary concepts, practices, and ethics of their chosen academic discipline in diverse communities.		X		X
3. Students construct, interpret, analyze, and evaluate information and ideas derived from a multitude of sources.	X	X	X	X
4. Students communicate effectively in written and oral forms to interact within their personal and professional communities.				X
5. Students use technology to access and communicate information in their personal and professional lives.	X	X	X	X
6. Students use multiple methods of inquiry and research processes to answer questions and solve problems.	X	X	X	X
7. Students describe, analyze, and evaluate global interconnectedness in social, economic, environmental and political systems that shape diverse groups within the San Francisco Bay Area and the world.				

Courses Curricular Map

	PLO1	PLO2	PLO3	PLO4
Program Learning Outcomes X Courses	THEORY: Explain and analyze standard computer science algorithms and describe and analyze theoretical aspects of various programming languages.	APPLICATION: Apply problem-solving skills to implement medium- and large- scale programs in a variety of programming languages.	SYSTEMS: Describe the interactions between low-level hardware, operating systems, and applications.	PROJECT: Demonstrate effective communication and organization as part of a team of software developers or researchers collaborating on a large computer program.
Courses or Program Requirement				
110: Introduction to Computer Science I	I	I	I	I
112: Introduction to Computer Science I	I	D	I	I
212: Software Development	D	D		D
SYSTEMS:			D	
220: Introduction to Parallel Programming				
221: C and Systems Programming				
245: Data Structures and Algorithms	D	D		D
315: Computer Architecture			M	
326: Operating Systems			M	
THEORY:	M			
345: Programming Language Paradigms				
411: Automata Theory				
414: Compilers				
APPLICATIONS:		M		
333: Introduction to Database Systems				
336: Computer Networks				
360: Data Visualization				
419: Computer Graphics				
420: Game Engineering				
451: Data Mining				
480: Computers and Society				
398/498: Directed Reading and Research		D		D
490: Senior Team Project		M		M
	Key:			
	I = Introductory			
	D = Developing			
	M = Mastery			

Methods

Questions

This year, we chose to collect data regarding the following question:

> Are students prepared to implement a large-scale project in the CS 212 - Software Development course?

Though CS 212 is designed as a second-year course, for a variety of reasons students end up taking the course at different points in their program. Because of this, in this year's assessment report we explore how the course learning outcomes are met, and then explore several indirect assessment measures to evaluate whether the course may be modified to better meet the needs of our students.

This question is not directly related to our most recent program review as we are in a program review year and did not have enough information from the program review before developing our plan for this year.

This question is related to PLO 2: APPLICATION: Apply problem-solving skills to implement medium- and large- scale programs in a variety of programming languages.

The direct method of assessment we employ is to examine solutions to homework and project assignments.

CS 212 Course Overview

CS 212 is designed as a sophomore-level course that requires students to implement a large piece of software that is well designed and efficient. In 2015/2016, three instructors taught three individual sections of the course. The structure and requirements for each section were almost identical, and the three professors collaborated to ensure that students were required to complete roughly the same assignments across all sections.

For most course learning outcomes outlined below, students implement an extremely structured and small programming homework assignment as well as a large programming project. For some course learning outcomes, the student implements only a project. Each project builds on the previous and, at the end of the semester, successful students will have iteratively built a single large piece of software typically comprised of about 2,000 lines of source code.

Mastery Learning

CS 212 uses a mastery learning approach. Though homework assignments have strict deadlines, students work on the project assignments until they are complete, pass all test cases, and meet the high standards of code design set by the instructor. Each project is typically submitted two to three times. Each student meets with the instructor individually for a code review. During code review the student receives feedback and then must implement changes based on the instructor's suggestions.

If a student completes a project assignment then he/she has demonstrated mastery of the topic and receives full credit minus any deductions for small things like not following directions. Some students may not complete all projects. A student who fails to complete all projects is graded on the number of projects completed. In essence, the goal is for a student who earns a C to have an A-level understanding of 75% of the topics rather than a C-level understanding of all topics as may be typical.

Course Learning Outcomes

The course learning outcomes (CLOs) for CS 212 are as follows:

- CLO1: Implement a program that uses several complex data structures.
- CLO2: Implement a program that uses threads and concurrency.
- CLO3: Implement a program that uses introductory elements of web applications, including HTML.
- CLO4: Implement a program that uses advanced features of web applications, including a web server and relational database.

Rubrics

As described above, students complete one small programming homework and a large programming project for most CLOs, with the exception of CLO4. Each programming project is revised and resubmitted until it demonstrates mastery.

The rubric used to evaluate CLOSs 1, 2, and 3 is as follows:

Unsatisfactory	Amateur	Acceptable	Exceptional
The student's homework solution is incomplete. It does not pass all test cases and/or demonstrates poor design practices. The student did not submit a project or submitted a project that did not pass most test cases.	The student's homework solution passes all test cases and demonstrates good design practices. The student did not submit a project or submitted a project that did not pass most test cases.	The student's homework solution is incomplete. It does not pass all test cases and/or demonstrates poor design practices. The student's project solution only passes most test cases or has some minor design flaws.	The student's homework solution passes all test cases and demonstrates good design practices. The student's project solution passes all test cases and demonstrates appropriate design practices.

The rubric used to evaluate CLO 4 is as follows:

Unsatisfactory	Amateur	Acceptable	Exceptional
The student did not attempt the project.	The student's solution correctly implemented some features but was missing many features or had significant design flaws.	The student's solution implemented most features but contained minor design flaws.	The student's solution implemented all features and was well designed.

Results

The following three tables illustrate the results of our direct assessment for the three sections of the course offered in 2015/2016.

Table 1: Direct assessment results for Fall Section 01.

		Unsatisfactory		Amateur		Acceptable		Exceptional	
	N	Freq	Pct	Freq	Pct	Freq	Pct	Freq	Pct
CLO1	25	1	4%	0	0%	2	8%	22	88%
CLO2	25	2	8%	2	8%	3	12%	18	72%
CLO3	25	6	24%	6	24%	3	12%	10	40%
CLO4	25	15	60%	3	12%	6	24%	1	4%

Table 2: Direct assessment results for Fall Section 02.

		Unsatisfactory		Amateur		Acceptable		Exceptional	
	N	Freq	Pct	Freq	Pct	Freq	Pct	Freq	Pct
CLO1	15	1	6.6%	0	0%	1	6.6%	13	86.6%
CLO2	15	3	20%	1	6.6%	5	33.3%	6	40%
CLO3	15	6	40%	3	20%	1	6.6%	5	33.3%
CLO4	15	10	66.6%	4	26.6%	1	6.6%	0	0%

Table 3: Direct assessment results for Spring Section 01.

		Unsatisfactory		Amateur		Acceptable		Exceptional	
	N	Freq	Pct	Freq	Pct	Freq	Pct	Freq	Pct
CLO1	27	1	3.7%	1	3.7%	2	7.4%	23	85.2%
CLO2	27	2	7.4%	0	0%	3	11.1%	22	81.5%
CLO3	27	2	7.4%	3	11.1%	2	7.4%	20	74.1%
CLO4	27	7	25.9%	6	22.2%	10	37.0%	4	14.8%

We have also considered several indirect measures of assessment as follows.

Figure 1: Student grade in CS 212 and student combined GPA for CS 110 and 112.

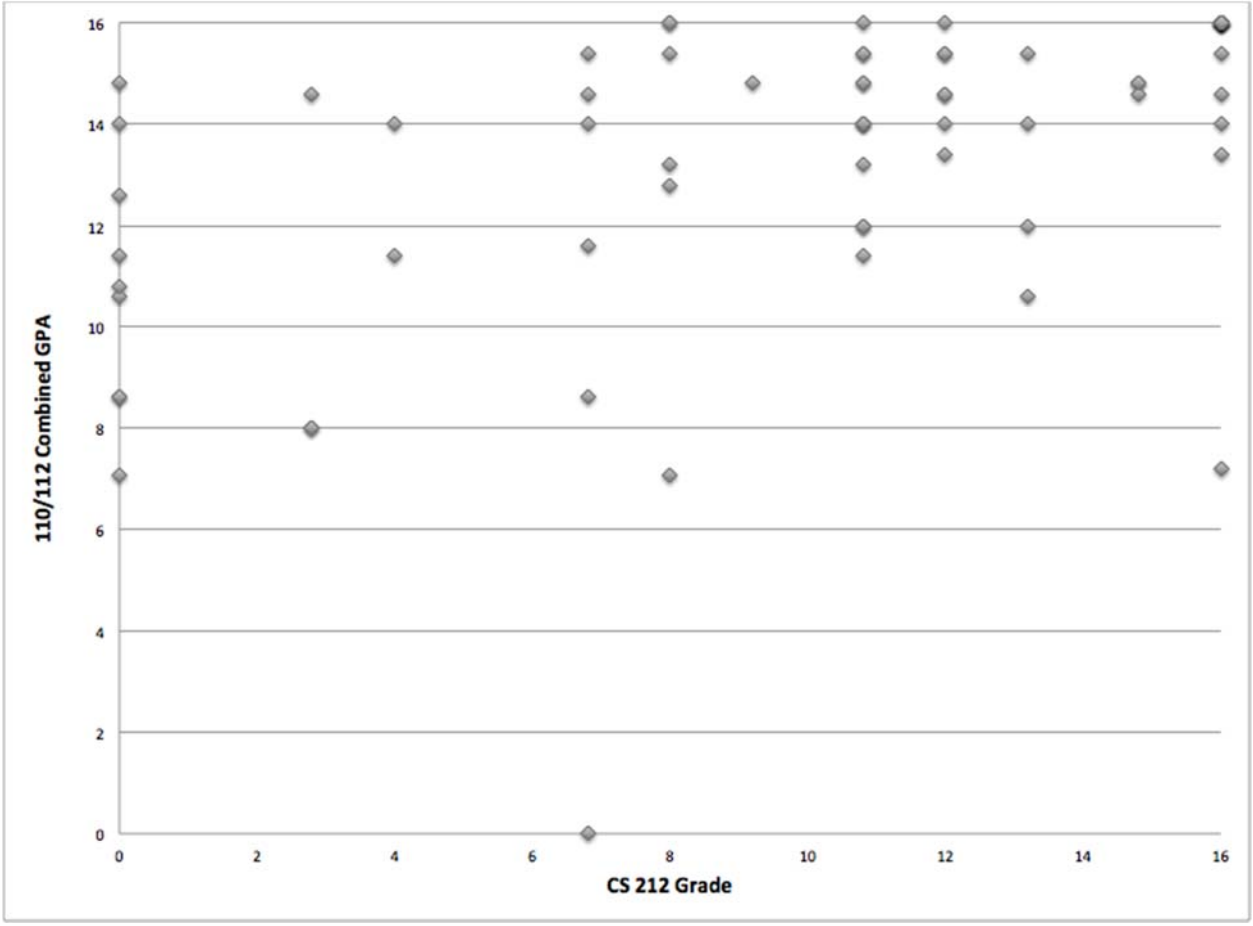

Figure 1 illustrates, for each student, his/her final grade in CS 212 and his/her combine GPA in our 110/112 introductory sequence. We expected to find a correlation between the intro sequence GPA and the CS 212 grade, however no such correlation is obvious.

Figure 2: Student grade in CS 212 and number of CS courses completed prior to CS 212.

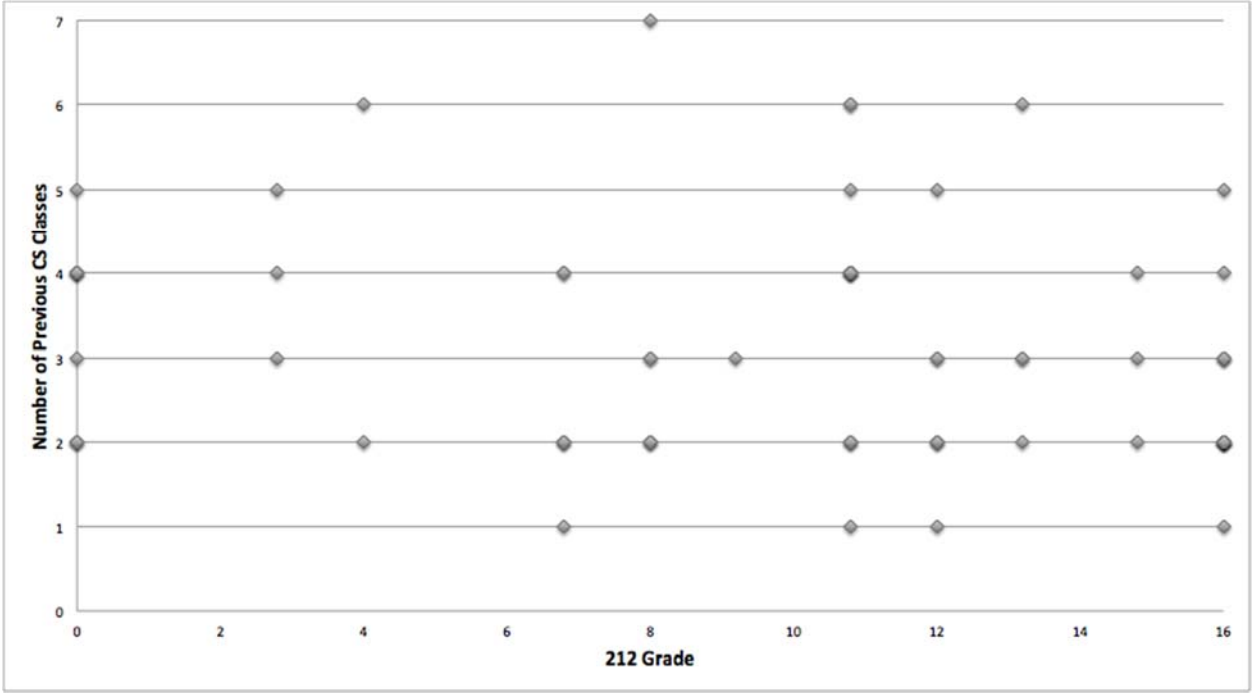

Figure 2 illustrates, for each student, his/her final grade in CS 212 and the number of CS courses taken prior to CS 212. This number does not include concurrent CS courses and does include courses that the student did not pass. Again, we expected to see higher 212 grades for students who had completed more CS classes prior to 212, however this was not the case.

We have long advised students that though CS 245 - Data Structures is not a strict prerequisite of CS 212 it is recommended to complete the course before attempting CS 212. Looking solely at final grades, however, we found that only 25.8% of students taking CS 212 in 2015/2016 took CS 245 before CS 212. Of the students who had not taken 245 or took it concurrently, 77.6% passed CS 212 with a C or better, and only 22.4% failed. More surprisingly, of the 25.8% of students who did complete 245 before 212, a sizable 53% failed and 47% passed. This is helpful information for CS advisors and suggests that students are well served by taking CS 212 early in their program.

The final indirect measure we consider is the failure rate of CS majors versus non-majors. Of the students who took the course in 2015/2016, 63.6% of them were CS majors and, of those students, only 14.3% did not earn a passing grade. The remaining 36.7% of students were non-majors (many CS minors) and a surprisingly large 58.3% of the non-majors did not earn a passing grade. One hypothesis is that non-majors are not prepared for the amount of work the class requires.

Discussion and Closing the Loop

It is noteworthy that the results include data for some students who essentially quit coming to class and submitting assignments part way through the semester however, for a variety of reasons, were unable to withdraw from the course.

It is also noteworthy that the structure of the assignments did vary somewhat from fall to spring. In the fall, the project associated with CLO3 included some additional complex material that was not covered in the spring section. Though this material is unquestionably valuable for the students, we experimented with leaving it out in the spring and note that the number of students who were able to achieve the exceptional rating increased because of the more narrow focus. We plan to reconsider whether the CLOs should be updated to include the more complex material, or whether we should leave it out of the curriculum for the course going forward.

The mastery learning approach results in a large percentage of students achieving the exceptional rating for topics covered early in the semester with a significant falloff for topics later in the semester. While this seems to indicate that students learn the early topics better than they would if we used a traditional approach, it is difficult to say for certain that is the case. Based on anecdotal observation, it would be informative to add the following additional element to the rubric: the number of submissions required for the student to demonstrate mastery. In some cases, students submit their work four or five times. Each submission results in extensive feedback from, and often a one-on-one meeting with, the instructor or teaching assistant. This extra help makes it difficult to assess whether the student is mastering the concepts on his/her own or just applying the recommendations made by the instructor.

Based on our indirect measures, we were both pleased and surprised to find that students are not more likely to succeed in 212 if they take it later in their careers. This reinforces the current structure of our program. It is clear, however, that non-majors need to be carefully advised about the workload and expectations of the course. This could suggest that we should look more carefully at the structure of our CS minor.