2018-2019 SET Assessment Report

1 Program Mission and Educational Objectives

The mission of the Software Engineering Technology (SES <<yng/ocmyng/ocm 1 Tf l TJ 0.002 Tc -0.002 Tw 0 -1.228 T

The Program Educational Objectives of Oregon Tech's Software Engineering Technology Program are to produce graduates that:

1. Use their knowledge of engineering to creatively and innovatively solve difficult computer systems problems;
2. Regularly engage in exploring, learning and applying state-of-the-art hardware and software technologies to the solution of computer systems problems;
3. Will be an effective team member that contributes to innovative software design solutions to the resolution of real world problems;
4. Will communicate effectively both as an individual and within multidisciplinary teams.

					Fall 2018
Klamath Falls	173	177	147	157	159
Wilsonville	116	128	136	116	111

B. an ability to select and apply a knowledge of mathematics, science, engineering, and technology to engineering technology problems that require the application of principles and applied procedures or methodologies;

C. an ability to conduct standard tests and measurements, and to conduct, analyze, and interpret experiments; and to apply experimental results to improve processes;

D. an ability to design systems, components, or processes for broadly-defined engineering technology problems appropriate to program educational objectives;

E. an ability to function effectively as a member or leader on a technical team;

F. an ability to identify, analyze, and solve broadly-defined engineering technology problems;

G. an ability to apply written, oral, and graphical communication in both technical and non-technical environments; and an ability to identify and use appropriate technical literature;

H. an understanding of the need for and an ability to engage in self-directed continuing professional development;

I. an understanding of and a commitment to address professional and ethical responsibilities including a respect for diversity;

J. a knowledge of the impact of engineering technology solutions in a societal and global context; and

K. a commitment to quality, timeliness, and continuous improvement.

4 Curriculum Map

The Bachelor of Science in Software Engineering Technology degree requires 187 credit hours as prescribed by the curriculum outline.

Curriculum

Required courses and recommended terms during which they should be taken:

Freshman
Year Fall

- CST 116 - C++ Programming I Credit Hours: 4
- CST 162 - Digital Logic I Credit Hours: 4
- MATH 111 - College Algebra Credit Hours: 4
- WRI 121 - English Composition Credit Hours: 3

Total: 15 Credit Hours

Winter

- CST 126 - C++ Programming II Credit Hours: 4
- CST 130 - Computer Organization Credit Hours: 3
- MATH 112 - Trigonometry Credit Hours: 4
- SPE 111 - Public Speaking Credit Hours: 3
- WRI 122 - Argumentative Writing Credit Hours: 3

Total: 17 Credit Hours

Spring

- CST 120 - Spr|(~(€ $ Credit Hours: WRI 122 -

Junior Year
Fall

- CST 229 - Introduction to Grammars Credit Hours: 3
- CST 316 - Junior Team-Based Project Development Credit Hours: 4
- CST 324 - Database Systems and Design Credit Hours: 4 4

MATH 327	Discrete Mathematics		X									
MATH 465	Mathematical Statistics		X									
MGT 345	Engineering Economy											
PHY 221	General Physics with Calculus		X									
PHY 222	General Physics with Calculus											

5 Assessment Cycle

PSLO	2018-2019	2019-2020	2020-2021
A. an ability to select and apply the knowledge, techniques, skills, and modern tools of the discipline to broadly-defined engineering technology activities;	X		
B. an ability to select and apply a knowledge of mathematics, science, engineering, and technology to engineering technology problems that require the application of principles and applied procedures or methodologies;		X	

6 Assessment Activities

6.1 PSLO A: an ability to select and apply the knowledge, techniques, skills, and modern tools of the discipline to broadly-defined engineering technology activities

6.1.1 Assessment activities:

1. Junior Project (CST 316-336): Evaluate documentation developed winter quarter
2. Senior Project (CST 412-432): Evaluate the code submitted as part of the final deliverable spring quarter
3. Indirect: An exit survey was given to graduating seniors. As part of the survey, students were asked to rate their proficiency on each of our PSLO's.

6.1.2 Rubric

The following rubric was used for both direct measurements.

Category: A	4 Highly Proficient	3 Proficient	2 Some Proficiency

6.1.5 Discussion

Both campuses had seniors scoring below juniors. The instructors from both campuses reported that the low scores had more to do with motivation than with technical ability. It is unclear whether this is simply a cohort problem. Given that the student populations are distinct, it seems unlikely that both locations would experience a cohort problem in the same year.

Another possible explanation for the drop in senior scores is that students get burnt out by senior year. This in turn could just be senioritis, or it could be an indication that there is something in our program that causes this.

This year's data isn't sufficient to determine the cause of the drop. We will monitor next years seniors to see if the problem repeats itself. If so, we will need to look for systemic problems in our program.

6.2 PSLO D: an ability to design systems, components, or processes for broadly defined engineering technology problems appropriate to program educational objectives

6.2.1 Assessment activities:

1. Junior Project (CST 315-336); Evaluate design documentation
2. Senior Project (CST 412-432); Evaluate the Use Case, Object Model, and Dynamic Model documents
3. Indirect: An exit survey was given to graduating seniors. As part of the survey, students were asked to rate their proficiency on each of our PSLO's.

6.2.2 Rubric

The following rubric was used for both the direct measurements

7.2 PSLO D: an ability to design systems, components, or processes for broadly defined engineering technology problems appropriate to program educational objectives

The data showed a significant problem writing for our junior class. The previous iteration of assessing this PSLO did not show as significant a problem with writing. We evaluate this year's juniors to see if the problem is systemic or if it is a cohort problem. We will also look for other courses where we can give students an opportunity to write design specifications so they have practice before getting to junior project.

7.3 PSLO I: an understanding of and a commitment to address professional and ethical responsibilities including a respect for diversity

Based on the data we collected, students did a good job evaluating ethical situations. We will look at

Anecdotal evidence suggests the Linux command line environment did help students to understand the difference between a source file and an executable and the process by which you turn one into the other. However, more work is needed to determine if the effect is big enough to switch all our intro classes to the Linux environment.

8.3 Teach some sections of CST 116 using C style IO instead of C++ style IO.

We had observed some confusion in students when they encountered file IO. They thought "cout" was a statement, so when they encountered file IO where there was an operation that was similar to what they did with `cout`, but didn't include that "statement", they were confused. We were hoping that a functional approach to IO instead of an operator overloading approach would relieve the confusion.

The classes that were done with the C style IO approach did seem to go better. However, another section done in parallel with the C++ approach, where additional time was spent trying to avoid the confusion also went well. As a result, we don't know if the different approach is better than just a more careful treatment of the material.