Minutes of the CSS Committee
December 7, 2012
10:30am-12noon, CP 206C

Attendance: Donald Chinn (co-chair), John Mayer (co-chair), Matt Alden, Zaide Chavez, Allison Elliot Tew, Alan Fowler, Robert Friedman, Bryan Goda, Beth Jeffrey, George Mobus, David Ross, Ruth Vanderpool, Dan Zimmerman.

1. The committee approved the minutes from Oct. 12 and Nov. 9 meetings.

2. **Model syllabi.** The committee briefly discussed changes to the TCSS 481 (computer security) model syllabus. After discussions with Mobus and Bai, it was agreed that the TCSS 422 prerequisite be removed and replaced with TCSS 342 (while keeping the TCSS 325 prerequisite). The resulting model syllabus for TCSS 481 was approved by the committee.

   Sam Chung revised the TCSS 460 model syllabus, but since Zimmerman has not reviewed the edits, the committee tabled the discussion and vote. Zimmerman will review it.

3. **CSS curriculum reform discussions.**

   The committee continued discussions about changing the CSS curriculum. This meeting's discussions focused on the hardware/systems part of the curriculum and the introductory programming sequence.

   Mayer proposed that TCSS 372 (computer architecture) be an elective and that we create a new course (System Programming in C), which would be similar to CSE 333 at UW Seattle CSE 333. He proposed that this new course be a prerequisite to TCSS 422 (operating systems) and a possible program requirement (either inner or outer core, if we decide to adopt that curriculum structure). The virtues of such a course are that it would provide CSS students an in-depth introduction to C programming in the context of systems programming, which would also provide support for TCSS 422 (operating systems). Instructors of TCSS 422 could spend less time on learning C and more time on new or deeper course content and projects.

   Mobus, Goda, and Zimmerman were reluctant to eliminate TCSS 372 as a requirement. The discussion expanded to a more general discussion of the value of studying architecture principles for CS students. Mobus argued that study of architecture provides examples of widely applicable CS principles. Seeing CS principles in different forms was generally acknowledged as a desirable outcome, but Chinn and others also expressed a desire for students to have more flexibility in choosing the areas of knowledge they want to study, since computer science covers a vast range of knowledge and principles. The discussion of the systems courses will continue.

   There was a substantial discussion about the introductory programming sequences and whether there would be opportunities for students to switch from the CE (201/202/203) to CS (142/143/305) track or vice versa. Alternatives considered include: (1) one common

sequence for both majors, (2) two completely separate introductory programming sequences, (3) a CS 0 course as a common starting point, and then students would then need to choose which track to take, (4) if CSS has a separate track, then there is an option to teach TCSS 142 in Python instead of Java. These options are not mutually exclusive. After much discussion, it was recognized that because of the different approaches to delivering the introductory programming sequence in the two programs, two separate tracks was probably the best implementation. Further, there was general consensus that it would be very difficult for students to switch tracks "mid-stream," and so at best we can treat each three-course sequence as a unit that would be equivalent in terms of satisfying any prerequisites for either major, since on the whole the introductory programming sequences have similar learning objectives.

Since the CES introductory programming sequence will be taught in C and the CSS sequence primarily in Java, a lingering question is whether CES students will have trouble in TCSS 342, which currently is taught in Java. Some ideas put forth include: (1) leave TCSS 342 essentially as is and CES could offer a short course/workshop (say, 2 units) on Java programming for students who know C, (2) leave TCSS 342 as is and let CES students learn the essentials of Java on their own (since it is relatively easy for them to do so), (3) teach TCSS 342 in a language-agnostic way (for example, students can develop their projects using either C or Java). There was no clear consensus as to which of these alternatives was best.

4. Chinn presented the tentative Spring Schedule.