Syllabus Stability

Over two 2 hour meetings we discussed the core syllabus (1917-1927-2911) to find areas where there were differing understandings of what was to be taught or in what depth, and also to makes some minor adjustments to the syllabus where there was general agreement but without revisiting the whole syllabus in any major way. There should be a full syllabus review after 5 years of the old syllabus being established (ie in 2011 or 2012). There should be a quick review of 1911 and 1921 immediately in accordance with the review of 1917. We had some discussions about 1911 already at the meetings and subsequently richard and gabrielle have made some specific plans for change in 1911 (also outlines briefly below)

Attendees: Richard Buckland, Gabrielle Dittu, Peter Ho, Alan Blair, Wayne Wobke, Oliver Diesel, John Potter, Maurice Pagino, ..? (who have i forgotten?)

The Modified Syllabus

The revised syllabus is at https://wiki.cse.unsw.edu.au/~info/cgi-bin/moin.cgi/CoreCourses/contents

Summary of changes: adts moved from 1917 to 1927, abstraction and programming to apis still need to be taught in 1917. a range of material was identified as optional. this means it is not required to be taught by the syllabus and future courses cannot depend on it having already been taught. lics are welcome to include this material **provided that** (ofcourse) the core material is covered soundly. It is important that courses do not overload students with extra material but not give students a sufficiently sound understanding of the core material. for example - microcontrollers/assembly in 1917. gdb was moved into 1917 core. tables, hashing, hash algorithms moved from 2911 to 1927. tdd and crc cards moved to 2911 core, generics confirmed optional.

commentary - the general thrust of the changes was moving material from the surrounding courses into 1927. this can only work if 1927 does inherit students who can already program in c and are confident with pointers and dynamic structures. there is no time in 1927 now to teach students how to program. tress as a programming exercise should be introduced in 1917, perhaps as an example when doing recursion, tree pruning algorithms and BFS of trees is still 1927 content.

The objectives of each course

1917 - at the end of the course students can confidently program small program in c, in particular they have mastered memory, pointers, malloc. Their

code follows professional coding practise including understanding and writing code free of security vulnerabilities, they have an understanding of testing and debugging. our less tangible objectives are that they should have developed a sense of the pleasure of computing and are highly motivated to work hard and learn independently, and have started to develop a sense of community within their yeargroup.

All students passing this course need to be confident programmers.

1927 - at the end of this course students understand space and time complexity, and know the standard basic computer science data structures and algorithms - eg trees and graphs and algorithms on them

2911 - at the end of this course students know how to design software solutions to problems using standard data structures and algorithms (covered in 1927) or by designing their own. they learn OO design, the main algorithm design methodologies, and also cover team programming theory and practise of.

Actions

- standard info pages
- planned changes and testing
- teaching soft skills
- post mortems (pass rates, passing student competencies, student feedback)
- pre-testing in 1927 in the coming semester
- 1911 fail rate, soft skills, involvement of other schools
- prac exams
- evidence
- future syllabus review
- exams for this coming session in advance
- assignment synchronisation