Improvements for Inf2-SEPP

Cristina Adriana Alexandru

Cristina.Alexandru@ed.ac.uk

Introduction to Inf2-SEPP

- Compulsory UG2 course for most of our degrees
- Very large: 224 students this year
- Employing 39 teaching support members of staff: 2 TAs, 16 tutors, 9 demonstrators, 10 coursework markers, 2 exam markers
- Split into: software engineering (SE) part and professional practice (ProP) part

Current format

- **Assessment** is 100% coursework:
  - For SE (75%): groupwork during a SE project in 3 parts: requirements, design, implementation/testing
  - For ProP (25%): an individual essay on professional issues during SE project part 3
  - Marking criteria-based per task, additive between tasks; some criteria conditioned by other tasks.
- 3 X 50-minute lectures/week: 2 for SE, 1 for ProP/guest lecture.
- 2 X drop-in 50-minute **labs**/week starting in week 2, to work on coursework; No exercise sheets.
- 5 X 1.5-hour **tutorials** every 1-2 weeks starting in week 2; Exercise sheets for practice of concepts for coursework.
- 2 X 50 -minute **office hours/week**: one for SE, one for ProP

Problems and questions

1) Budget being spent on student support in labs/tutorials:

- Attendance in labs and tutorials has been low, with surges before coursework deadlines
- The budget on tutor/demonstrator hours would better be spent on marking and TA work

Questions: How can I make labs/tutorials more resource effective

- **Some ideas:**
  - Motivating students to attend tutorials more- e.g. credit for bringing pre-prepared solutions
  - Experimenting with larger tutorial groups, but how large is 1) still interactive enough for students, 2) manageable?
  - Merging tutorial and lab components by having longer labs with tasks, but are labs a good idea for non-programming parts?

Problems and questions

2) Workload/hours spent designing coursework each year:

- Similar tasks, but different case study each year due to: students making repositories public, student request for solutions=> writing case study very labour intensive
- Additional small experimentation/redesign each year
- These have resulted in exceeding time budgeted for the TAs

Question: How can I make coursework design less labour intensive?

Some ideas:

- System spec that can be extended in many ways; Adding to it each year.
- Re-using older case study, with some strategic changes that makes it hard to adapt past solution.

Problems and questions

3) Motivating practical work

- The course focuses on requirements, design, UML diagramming, teamwork, professional practice, writing good code and tests… but code and tests come late in the coursework (week 8)
- The student rep mentioned that many students want to code more in this course… but maybe some of the unheard students are glad it's ***not*** coding-intensive?

Question: How can I offer the opportunity for some- but not all- to do more coding in a SE course? Or else, how to motivate students on the value of the existing work?

Some ideas:

- Applying agile approach to work throughout so that students do iterations involving code straight from the onset; But this would be a large change and we wouldn't be able to maintain CW parts-> BoS approval?
- Getting students to catch-up on programming through small preparatory tasks in CW parts 1 and 2. E.g. working with a codebase (library, API) that they would use later. But would this be assessed? The CW is already large.

Problems and questions

4) Motivating guest lecture attendance

- Very low attendance in online guest lectures this year

Question: How can I encourage students to attend guest lectures, which are very useful for them?

Some ideas:

- Making them face-to-face could help
- More advertising
- The guest lecturers could also give out tasks/ even coursework ones
- Guest lecturers demonstrating things in practice

Do you have any suggestions?

1) How can I make labs/tutorials more resource effective
2) How can I make coursework design less labour intensive, while still launching solutions each year?
3) How can I offer the opportunity for some students- but not all- to do more coding in a SE course? Or else, how to motivate students on the value of the existing work?
4) How can I encourage students to attend guest lectures, which are very useful for them?

Thank you!