Early adopter proposal regarding the Software Engineering 751 elective course

Nasser Giacaman and Oliver Sinnen
The Department of Electrical and Computer Engineering, The University of Auckland, New Zealand
n.giacaman@auckland.ac.nz *and* o.sinnen@auckland.ac.nz

Software Engineering at the University of Auckland

Software Engineering (SE) at the University of Auckland is an Engineering specialisation administered by the Department of Electrical and Computer Engineering and co-taught with the Computer Science department. The course discussed in this proposal (SoftEng 751: High Performance Computing) is an elective available to both final-year undergraduate students and taught Masters students:

- Undergraduate students: For undergraduate students, SE is a 4-year Bachelor of Engineering (BE) degree. To be eligible for the award of honours, i.e. BE(Hons), students are required to select 700-level courses (such as SoftEng 751) for their final-year electives as well as achieve a high grade average. Students select 4 of approximately 11 possible courses for this component of their degree.
- Taught Masters students: The taught Masters degree is 1 year, also with a Software Engineering specialisation option, and is known as the Master of Engineering Studies programme. Students select 8 of approximately 15 possible courses, with SoftEng 751 being one of those options.

Inception of SoftEng 751

SoftEng 751 is officially a new course in the University Calendar. It was officially recognised and introduced in Semester 1 of 2012 (February to June) for its inaugural undertaking. It evolved out of another course (SoftEng 461: Special Topic in Software Engineering) with the realisation of the ever-growing importance of parallel computing. Consequently, this was acknowledged with the establishment of SoftEng 751 as a dedicated HPC course into the SE curriculum available to both undergraduate honours and taught Masters students. The course description refers to SoftEng 751 as “project-based hands-on high performance computing and programming”. This is purposely generic to accommodate the ever-changing developments of the PDC research area.

Parallel and Reconfigurable Computing lab (PARC) and its link with SoftEng 751

In supporting parallel computing research at the University of Auckland, the PARC lab was founded by Oliver Sinnen. Also heavily involved in this lab is Nasser Giacaman. The lab primarily focuses on the shared memory aspects of parallel computing, with facilities such as 64-core, 16-core, 8-core systems and numerous multi-core Android mobile devices (smartphones and tablets). The lab focus is on the development of desktop and mobile applications, targeting the current and future wave of multi-core systems. Since SoftEng 751 is project-based and co-taught by both Oliver and Nasser, the course largely includes research projects closely linked to the latest development from the PARC lab.

Overall structure of SoftEng 751

Just like all courses at the University of Auckland, SoftEng 751 involves 12-weeks of lecture slots. Half-way during the 12-weeks is a 2-week semester break, but students continue to study during their own time (e.g. on projects) for their respective courses. Lectures are held 3 times per week, in 50-minute slots. During its inaugural undertaking, SoftEng 751 was structured as follows. The first 3.5 weeks included traditional lectures, where students were taught the fundamentals of multi-threading and parallel computing, specifically focusing on shared memory systems (i.e. multi-cores). As shown in the next section, these weeks focus on the most essential topics identified by the Curriculum Initiative.

The major theme for the course was user-interactive applications, such as those typically expected on desktop systems and mobile devices. During the 2nd week, students are provided with a list of project topics and brief descriptions to contemplate. At the end of the 4th week, students are required to submit their top 5 preferences of a project to undertake for the remainder of the course in groups of 2-3 students. Students are also encouraged to propose their own project, provided it relates to the wider high performance computing arena. Students are informed of their assigned project before the beginning of the 5th week, consequently providing for approximately 10 weeks of “project-time”. Ultimately, each group has a different project.

The remainder of the lecture slots turn into presentations. Each lecture slot includes 3x 15min presentations (i.e. by 3 different groups). This allowed for 2 rounds of presentations over the period of the course (as there were 21 groups). The first round required the group to articulate to the class the nature and goals of their project, and assessment of this round was lenient (considering most projects were in their early

stages). In the second round of the presentations, students were expected to demonstrate considerable progress since the first round. Attendance to the presentations was highly advised, ("even if your group isn't presenting"), as the end-of-semester test (contributing 25% of the final grade) would involve specific questions regarding the student projects. In this way, projects (and their subsequent presentations) expose students to a wider range of PDC topics that are identified by the Curriculum Initiative as recommending students have a knowing and comprehensive understanding of those topics. In addition to the class presentations, groups meet on a weekly basis with lecturers for a deeper discussion on their project progress.

SoftEng 751 and the NSF/IEEE-TCPP Curriculum Initiative on Parallel & Distributed Computing

In line with the Curriculum Initiative's introduction, the major goal and theme of SoftEng 751 is to motivate students by embracing PDC as an integral component of their professional career. The first 3.5 weeks of the course primarily targets the vital parts of the programming topics of the Curriculum Initiative that have been identified as requiring a learning level (using Bloom's classification) of students being able to "*apply" (A)* and *"comprehend" (C)* the concepts respectively:

- Architecture topics (table 1 of the Curriculum Initiative)
  - Classes
    - Multicore (C)
  - Memory hierarchy
    - Cache organisation (C)
- Programming topics (table 2 of the Curriculum Initiative)
  - Parallel Programming paradigms
    - Shared memory (A), Task/thread spawning (A), SPMD (C), Data parallel (A), Parallel loop (C)
  - Parallel Programming notations
    - Compiler directives/pragmas (C), Libraries (C)
  - Semantics and Correctness
    - Synchronisation (A), Concurrency defects (C)
  - Performance issues
    - Computation (C), Performance metrics (C)

Students are essentially expected to *apply* the above topics after the first 3.5 weeks of taught lecture slots. Java is used during this phase of the course. A major component of these lectures includes live-demonstrations and coding examples during class. A component of the end-of-semester test includes questions assessing students by expecting them to *apply* those topics (i.e. understanding at the code-level).

From the group presentations, students are exposed to a wide variety of topics identified by the Curriculum Initiative: GPUs (K), SMT (K), Distributed memory (C), Libraries (C), Cloud/grid (K), amongst others. As suggested by the Curriculum Initiative, it is not expected for students to *apply* these topics, but merely *comprehend* (C) or *know* (K) them. This is reflected in the end-of-semester test, as students must discuss open-ended questions regarding these topics (i.e. understanding at the conceptual level rather than code-level). This approach of having students present their own projects to the class (recall that each group has a different topic) allows for a wider, albeit shallower, exposure to numerous PDC topics that would be otherwise difficult in a traditional lecture setting. Group presentation slides are made available for the other students to study from.

In summary: a combination of traditional lectures at the beginning of the course provides students with the essential *application* skills (as identified by the Curriculum Initiative), while projects (and their regular presentations) during the remainder of the lecture slots exposes students to *know* or *comprehend* a wider number of PDC topics. Also, the way that projects are undertaken easily allows for new topics to be integrated into the course when necessary in the future.

Budget: It is anticipated to use the stipend for purchasing multi-core Android devices for students to evaluate their projects, as most tend to already possess multi-core laptops (but not multi-core tablets or smartphones). By having physical devices, we hope to motivate students as most were only using the Emulator.

Student feedback (from an end-of-semester anonymous evaluation)

- *The class discussions were effective in helping me to learn*: 92% agree + strongly agree
- *The lecturer stimulated my engagement in the learning process*: 96% agree + strongly agree
- *The lecturer stimulated my interest in the subject*: 96% agree + strongly agree