Group Members: Ayman Yousif & Ryan Lortie
SE3I03 Group Project (Final Draft Long Paper)
Jan 29, 2005

"A change for the better"

Background:

http://www.cas.mcmaster.ca/cas/undergraduate/SEbrochure.pdf

"A guiding principle of McMaster's Software Engineering Programme is the integration of theory and practise. Rather than have some courses identified as theoretical and others identified as practical, our approach is to integrate theory and practise in every course. No theory should be taught without showing the student how to use it. No practical problem solving technique should be taught unless theory shows that the technique is sound. Students are given practical (computer) assignments that teach them how theoretical concepts can be applied."

"Our number one issue is that our coders don't know how to talk to people and our software engineers don't know how to code. There's this huge disconnect..."

Matthew Fisch, Kaz, Inc.

"Some people don't have the 'coder' skills. They have the 'design' skills. Getting the 'design' and 'coder' together allows the person to step up to the next level."

Ron DeFulio, Professional Flight Management

Objective:

Our goal is to ensure that all the Software Engineering upper year students are comfortable with programming and applying programming principles. Most of the Software Engineering students are in a state that is "Programming is being the problem" instead of "Programming to solve problems". We would like to propose some ways of changing from our current state to the state of which programming becomes a tool that is to the students' advantage to solve problems, rather than being the burden.
After all, everyone expects a Software Engineer to be a proficient programmer.

Approach:

First of all, SE2S03 needs to be split evenly between C and Java. Common programming concepts like recursion, data structures (Trees…) … could be taught in either language or both. Also try to introduce some tools that help in understanding how programs work in a computer and how some programming concepts work (visually). For example programs that illustrate how pointers work or how recursion works inside of the computer visually.

We need to introduce mandatory labs for SE2S03 where students will have to solve problems in the lab under the supervision of TAs. These labs should cover most of the topics that the professor feels are important foundations and students need to be comfortable with them.

SE2AA4 assignments should give the students less freedom in using the programming language of their choice to solve assignments. They need to be split evenly between C and Java. There should be assignments to illustrate how Object-Oriented design works. Explaining it in class and reading it is never enough. Students should be able to program in an Object-Oriented fashion, and Java would be a good language to use for such purposes.
Lastly, try to incorporate more programming into other non-programming courses and teach the students how programming helps in solving problems. Such courses could be SE2C03, SE3X03, and SE2MX3 …

Impact:

Software Engineering concepts and principles will sink in better for the students. Learning the concepts and applying them will ensure that the students are at least familiar with the concepts. For example, Object-Oriented design. Once it's taught in class and applied in assignments or labs, the students will have a feel for what Object-Oriented design really is.

Students will struggle less in higher year courses. For example, SE3A04 in which students were asked to design and implement a project in Java. A lot of the students were shocked because of their limited knowledge of Java and their weak programming bones. This problem would diminish if we intensify the programming as advised in the Approach.

Graduates will be more confident and very proficient in programming and can easily learn other programming languages that they need for certain projects or jobs. Intense programming also helps students to develop logic skills which are very important in many fields.

After all, the department will get a good reputation when its graduates are good designers and programmers.

Cost:

The main cost of these changes is some time from a few professors to reorganize the suggested courses. The changes are not very major, so there will not be any significant financial costs associated.

Another cost that is not definite might be hiring more TAs. I believe that the TAs taking on the tutorials of SE2S03 are enough to take on the Labs, but one important factor is that those TAs should be qualified and be able to address students' needs.

Lastly, these changes may or may not increase the work load on the students. But I believe that these changes are critical even if they introduce a little bit more work load. They strengthen the students' foundations, which is critical and invaluable.