**Subject:** sslcom
**From:** Mike Holcombe <m.holcombe@dcs.shef.ac.uk>
**Date:** Tue, 22 Apr 2008 16:33:30 +0100
**To:** Sara North <sara.north@sheffield.ac.uk>

**Subject:** comments on teaching resources and course structure
**From:** Nicholas Rutherford <aca04ntr@sheffield.ac.uk>
**Date:** Mon, 21 Apr 2008 03:50:37 +0100
**To:** Richard Clayton <R.H.Clayton@sheffield.ac.uk>

Richard
Further to one of the SSL meetings (and in advance of the pending) last year where you mentioned graduate feedback on teaching resources, I thought I would send you some comments as a current student.
These are gripes I have had and heard over the duration of my course, regarding a number of different modules.

Teaching resources
---
Not enough lecturers hand out paper copies of their notes. Having to print our own is inconvenient and expensive, yet a small amount relative to the fees we pay.

Not enough core text books are present in the library. For example, Dexter Kozen's book for Machine Languages is oversubscribed currently, and I have also not been able to find Judith Gersting's book Mathematical Structures for Computer Scientists in the IC. There are lots of engineering maths books, but not many on the other types of maths we need.

The library and course's resources on programming languages are very poor. Particularly poor coverage can be found of PHP and Python, while Perl has many books yet its use is discouraged for coursework such as software hut?

Would it be possible to expand on existing material on subjects such as Unix, Mainframes, Networking, different languages and so on, so that they are readily accessible to students on the course, enabling them to discover new areas of the subject where lectures may not be offered. Text books are nice, and may already be in the library, but other resources such as videos, ebooks, websites, may also be nice.

With regard to the Lewin Lab I would suggest rather than spending money on new computers, that money is spent on the facilities themselves. The

room is not well suited to group work - other facilities should perhaps be provided for this. Could the desks be laid out better to cater towards groups working together?

I would also like to support the call for better remote-access support & performance for students with their own computers (which is surely the vast majority in some form or another).
Understanding SSH and other methods for this sort of thing adds yet another notch to the bow of graduates who have been exposed to it.

Software Engineering
---
Many useful things in software engineering, particularly IDEs and CVS systems have not been taught or mentioned at all. We are thrown in the deep end with eclipse (there was perhaps one lesson, which is not sufficient).

Conversely we have been provided with the Sheffield Management Tool to organise and log our efforts in group projects. I and many other students would like to suggest that members of staff use this tool for their own group projects so that they can gain an appreciation of how poorly suited to its task it is, and that it is neither stable nor properly designed for its intended task.

We are taught a number of conflicting software engineering methodologies through the course. Although this is interesting, it is unfortunately producing an undesired effect of students producing systems with methodologies they do not at all understand, and effectively run around with their trousers around their ankles trying to use them and learn them at the same time, with no clear idea of why they are doing it.
This often results in them simply building code without any heed to what they are supposed to be doing, which is a waste of our time and yours.

To speak more directly of the modules, in the first year we were exposed to some rather bureaucratic and unfriendly methods and notations. Personally I don't think that scaring first years is a good way to endear them to the subject.
If these methods are false, then why are we taught them and expected to provide successful systems using them, particularly as novices.

To add insult to injury in the second year we study (to great, and very insightful depth) the Discovery Method, in addition to the flaws of these previously mentioned systems. I believe this module is of great worth; however the workload is far too high. The group work takes a lot of time, and many failed to keep up with the theory required to complete it, diminishing the intention of the practical work, which is surely reinforcement by application.

Now we come to Software Hut, another well conceived module, yet flawed once-more. Having been taught at great pains a software development technique in the previous semester, here we throw it away and use XP instead, which although more easy to get to grips with, has led to many groups simply writing code without any use of the methods they have learned since starting the course. I am afraid I don't see how this teaches or proves anything.

I would suggest a rethinking of how this is taught, perhaps one module which compares and contrasts as an introduction, then picking one or two to teach properly and giving practical experience with them. It's a tricky one, and I don't know quite what to suggest, but I do not believe that the current method is ideal. Perhaps continuous assessment on understanding a methodology rather than using it would be more successful.

It would be nice to be taught something, then to use it later, rather than to have to learn it and use it at the same time.
Following the 07/08 autumn exams I felt I had a good grasp of the Discovery Method, as did many other students I have spoken to. I believe that it would have been a good idea to harness this into a practical module. Maybe this could have been done in one semester? Mid-term exam followed by practical work to finish?

I do not believe this is the failing of any one module. Rather I believe the course structure needs to be rethought, or at least critically examined.

In the short-term it would be nice to have some form of seminar or presentation done by the proponents of the various techniques in the department, to summarise for the students if you like, the reasons why they might use one technique or another at some point, and what it is important for them to remember.

I'll finish by saying that I am sorry this email is so long. It seems there was a lot to say.
I hope this can be taken on board to some extent. I feel something needs to be said, as not only am I concerned about my own learning, but I am concerned about those who come later getting the best opportunity they can, particularly given the spiralling cost of coming to university.

Thanks
Nick Rutherford (the other 2Y SSL Rep)