Hi Nick (cc Mike H as chair of SSLCOM),

Thanks for your mail, and I'm grateful for your feedback.

I'm unwell at the moment with a chest infection, so I won't be present at SSLCOM today, however if there are any issues that you want to discuss please raise them at the meeting, and I will talk to Mike Holcombe when I am back at work. I have responded to your specific comments in detail below. However there are a couple of issues which form a backdrop to my responses.

The first of these has to do with student centred learning, and page 10 of the undergraduate handbook sums this up very well: "Study at a university is quite different from other forms of education. You are given all the facilities necessary to learn, but you are largely responsible for your own learning and progress. We expect students to be reasonably mature and self-motivated; unlike schoolteachers, we do not expect to have to chase up individuals on a weekly basis if they are frequently absent or fail to submit work. Your time at university is precious; it is up to you to make the best of it." This philosophy means, for example, that you might be introduced to an IDE, but that we expect that you will be motivated to learn how to use it in your own time. This approach might suggest laziness on the part of teaching staff; but remember that unlike schoolteachers we have a threefold responsibility of teaching (which includes MSc and PhD students as well as undergraduates), research, and administration. There is simply not enough time to teach skills in Python, Perl, Unix, Eclipse etc., and actually there is overwhelming evidence that passive teaching of this kind of material is far less effective in terms of student learning than a problem-based approach where students have to find out things for themselves. So there is an educational benefit to throwing you in at the deep end; you acquire the ability to learn new skills quickly, which will be important for your subsequent career, and of course this flexibility is one of the ways that a university education differs from a series of training courses that would simply equip you with skills in Java programing, using eclipse etc.

The second issue has to do with how the university, and DCS, is organised. Each year we are committed to teaching a particular set of modules, and like the proverbial supertanker it is very difficult to make changes during the year. One of the problems with the modular system is that learning becomes packaged and it is sometimes hard to see how modules link to each other. This is something that we could do better, and I would appreciate practical ideas and suggestions.

Specific responses below:

Nicholas Rutherford wrote:

> 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.

In my experience (and I'm playing devil's advocate here) printing out paper copies of lecture notes for a class of 100 students is also inconvenient and expensive, and giving students a set of lecture notes kills off their interest in a lecture and actively prevents them from making their own notes. Providing electronic versions online after the lecture, which the students can use as a reference (no need to print them out) to amplify their own notes, results in a more effective lecture. I would argue that note taking - identifying important points and concepts - is a valuable skill, which is not encouraged by distributing paper copies of notes. You are, of course, free to disagree :-) , but I hope you can see that there is a good reason for not always providing a full set of printed notes.

> 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.

There are at least 10 copies of these books in the IC and St Georges' library, some of which are available on short loan, and others of which can be reserved. There are other books by different authors that cover similar material, and you can also pick up the Judith Gersting book on Amazon very cheaply secondhand, so I am not convinced that this is a major problem.

> 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.

Thes points are addressed in part in my preamble about student centred learning - the resources are all there, and we expect you to be motivated to discover and make use of them. This will probably only make sense once you graduate and start work, but as a student you have more time available to you than you will ever have again. Make sure you don't waste it!

> 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?

It may be possible to arrange part of the Lewin Lab for group work, but given the group work space in the IC and the use of the Lewin lab for teaching large classes it is hard to justfy changing the layout of the whole lab. I suggest you raise this at SSLCOM as this is outside my area of responsibility.

> 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.

It would be helpful to raise specific needs at SSLCOM as again this is outside my area of responsibility. In my experience however ssh and sftp are not hard to learn (surely all you need to know is to type ssh ivy.dcs.shef.ac.uk in a terminal window?), and again comes under the umbrella of my remarks about student centred learning.

> 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.

You need to raise this issue with the appropriate member of staff - I know nothing about this tool, nor why we ask students to use it.

> 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.

These are helpful comments, and I am sure that we could improve the way we link these different methodologies to specific project based modules. Your idea of a seminar is a good one. However, I would also observe that you have been exposed to a number of widely different approaches to software engineering, and you are now in an excellent position to see that advantages and disadvantages of each, so that you can select an appropriate

methodology for a particular problem. Much better than 'training' in a single approach, which will not always be the best one.

> 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)

I hope I have been able to address your comments in a positive way.

Best wishes,

Richard

--
Dr Richard H Clayton
Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP
Telephone +44 (0)114 222 1845
http://www.dcs.shef.ac.uk/~richard