Designing labs and exercises to promote justified self-efficacy in computer science students.

Aston University

Dr. Lucy Bastin, Engineering and Applied Science

The problem...

Many computer science modules build on skills and competencies which students are assumed to have acquired in introductory modules. However, students can enter modules by a variety of routes; many lack confidence with what are seen as basic programming skills, and find it difficult to reflect on their exact levels of skill in the many component areas that make up deep knowledge of software engineering. This can reduce their persistence on important challenges, and seriously impact on achievement.

In the module described here, common problems had historically been:

- An unwillingness to debug code, leading to poor performance on coursework.
- Problems with generalising specific lab challenges to unfamiliar problems
- Anxieties about independent work (and hence high levels of plagiarism)
- Unwillingness to use unfamiliar technical tools such as Javascript debuggers.

The students...aka the solution.

This module (*Web Development*) was a core second-year requirement, but was also taken as an option by third-year students from many other courses. Past experience of practical programming was highly variable, and this led many students to assume that others in the class were more expert than them, and to be anxious and concerned about this.

I found most difficult. Mainly because the module was focused upon CS students, it was "assumed" as the "norm" that because most CS students are up to scratch on most of this coding, that the rest of should be too. Labs particularly were way to fast for CB students, falling back so quick that it was hard to catch up.
Please, Please, Please!! Do not offer this module to final year CB students if you are not willing to go at their pace, it may seem tideous to the CS students, but it is grossly unfar to move at their pace and not go at ours if you are ofering this module to CB students! Separate lab sessions should be made for CB students and the weighting of coursework / exam should be changed to suit CB students if you would like to run this module for both CS CB students.
The teaching/feedback was at a very high standard. However, at times the lectures felt overloaded with information. Although, the extra material was included for completeness, it affected the key points been learnt in a more efficient way. Condensing the slides/material into a more relevant format would have helped. Apart from that a well taught module.

Comments from past years showed that one group in particular felt overwhelmed. Though this cohort actually performed well, their 'self-efficacy' [1], or *belief that their efforts would make a difference* was clearly low.

'Self-efficacy' in brief

Self-efficacy [1] is not the same as simple self esteem or confidence. It refers to the level to which an individual believes in their own capabilities to achieve a specific goal. Self-efficacy tends to correlate positively with performance and when cause/effect is further investigated, it appears that these positive results are often mediated more by persistence and engagement than by innate ability. Self-efficacy tends to be increased when we experience meaningful accomplishments and feel that we have influence and agency, rather than by unspecific praise for effort or participation.

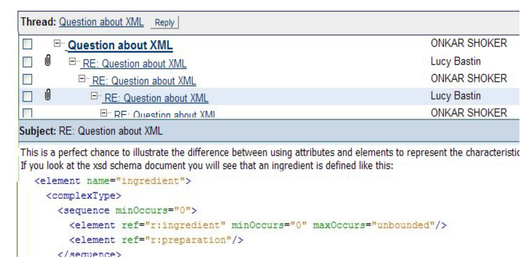
Strategies for improvement *(designed around factors identified [1] as contributing to self-efficacy)*

- An initial **diagnostic assessment of confidence** on specific topics, through an anonymous questionnaire.
- Class **discussion of the results**, clearly signposting how topics contribute to the body of knowledge, and how/when they would be covered.
- Generate **many small opportunities for students to succeed in finding and fixing problems**, so that the process was habitual and less threatening *(Mastery Experience [1])*
- Replace model code solutions with **narrated video walkthroughs of labs** which demonstrate a large variety of commonly-encountered errors, and how to fix them.
- **Refer to mistakes made in industry and consultancy**, and their solutions, to show that errors can be part of experimentation and innovation, not an embarrassment. *(Modelling [1.3])*
- Require **student questions to be posted on a discussion board**, and use this to build up an FAQ page. *(aiming towards Social Persuasion [1])*
- Supply **specific exercises based around fixing broken code**, to (a) demonstrate that problems *can* be tracked and fixed (b) make the use of debugging tools routine and familiar.

EVALUATING THE OUTCOMES: 1. Increased confidence across a broad range of topics

Key: ***4 = Very confident*** ***3 = Confident*** ***2 = Quite confident*** ***1 = Not at all confident***	**Start of module** Min – max **(mean) ****	**How was this addressed?**	**End of module** Min – max **(mean)**
How the Web works – theory *(e.g., protocols and languages)*	2-4 (**2.76**)	**Covered in lectures and labs** as a matter of course.	2-4 (**2.90**)
Working with the DOM *(e.g., updating paragraph elements in a document)*	1-4 (**1.84**)	All students had in theory already passed this topic, but knowledge was highly variable: The first 3 labs were adapted to **revise the topic**, and re-use **previous year's materials.**	2-4 (**2.41**)
Debugging Java *(e.g., identifying a problem within a 'while' loop)*	1-3 (**2.23**)	This should be a basic skill for students at this stage. Practice exercises were introduced, e.g. **code with deliberate bugs** which students fixed with progressively less instruction.	2-4 (**2.97**)
Debugging javascript *(e.g., identifying a problem stemming from a null var)*	1-3 (**2.08**)	An unfamiliar activity which requires the use of a novel tool (Firebug). **Deliberate errors** were introduced into lab example code for students to track down. All but 2 **labs explicitly required Firebug to be used**, and several **video tutorials** demonstrated its use.	1-4 (**2.50**)
Breaking code to see the effects *(e.g., overflowing a Java Array or referencing null variables)*	1-4 (**2.16**)	I wanted to encourage students to 'play' and experiment without fear that their work would be destroyed. **Lab demonstrators** showed how to temporarily, and safely, generate error messages. **Lab videos showed many common code errors** and fixes.	2-4 (**2.77**)
Client-server interactions *(e.g., POST form responses)*	1-4 (**2.05**)	**Covered in lectures and labs** as a matter of course.	2-4 (**2.50**)
XML syntax *(e.g., rules to which XHTML documents conform)*	1-3 (**2.05**)	An intimidating topic for many students, generating many threads on the **discussion board**. **Extra labs and tutorials** were produced, with **model answers and examples**.	2-4 (**3.00**)

2. Collaboration – not collusion

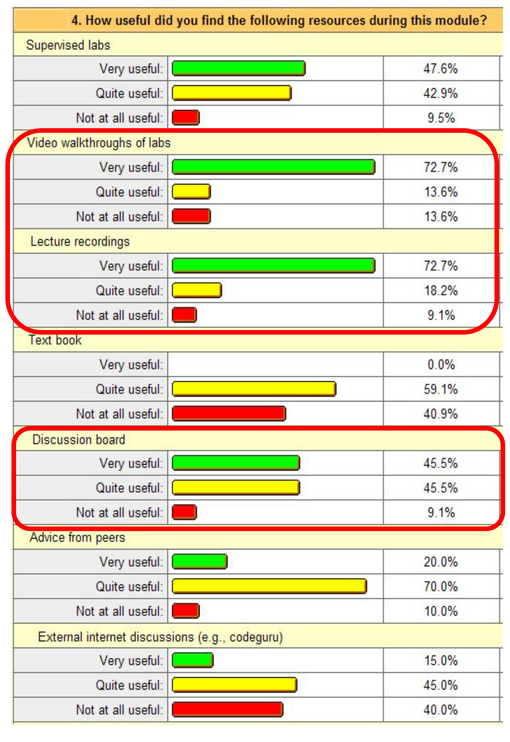


Students taking the module: **76**
Hits on the discussion board: **2989**
Messages posted: **78**
Students who posted : **19 (25%)**
Questions answered by another student: **20%**

Only 4 plagiarism / colllusion cases *
(as opposed to 15 in the previous year)

* Code assessed using the Stanford University Measure Of Software Similarity tool

3. Independent working



Discussion board and recordings were heavily used outside formal teaching hours(VLE statistics)

4. Reflection on skills and needs

43% of students felt that they were capable of generalising what they had learnt outside its specific context.

I feel that what I have learnt in this module enables me:	
to handle some specific problems in Web Development if they are similar to those tackled in labs:	57.1%
to handle a wide variety of unfamiliar problems in Web Development:	9.5%
to handle unfamiliar problems in Web Development, and also tackle other programming challenges:	33.3%

As well as specific technical training, some students mentioned that they had gained more general critical and analytical skills.

5. Which of your skills do you feel were strengthened by the module?
Developing JSPs/Servlets
Experimenting just to try out if things worked, General web development, How to find things out myself.
improved my understanding to the web apllications and the different approaches to form web pages
Java programming, XHTML and JSP etc
I always think about what kind of things goes on the background, how a peace of code not doing what I expected it to do? how is it interpreted and executed?
Java, making websites with MVC.
On the spcific side, XML and javascript, but more generally, things like how to track down errors and recognise common problems.
programming skills

Students varied in the degree of help they wanted, but all believed they were capable of 'fixing the problem' themselves.

In labs, when I ask for help:	
I would prefer just clues and hints as to how to solve my problem.:	45.0%
I would prefer to be directed to the information in lecture notes, to read it myself.:	20.0%
I would like the demonstrator to tell me how to fix the problem.:	35.0%
I would like the demonstrator to fix the problem for me.:	0.0%

5. Better achievement on practical assignments

Between 2009 and 2010, the average module mark increased from 48.5 to 58.6. 60% of this rise was due to increased marks on practical coursework. Exam performance also improved, particularly on questions which assessed experience or critical analysis. While the range of coursework marks in both years ran from 0 to 100%, the 25th percentile shifted upwards from 15 to 51%.

What next?

Encourage **students to post their own videos**, to show problems being fixed by 'someone like me'.

Measure self-efficacy using a standardised scale [2], to permit meta-analysis and longitudinal study.

Continue to design labs which provide **many and early opportunities for success.**

Encourage students to **reflect on how they physically respond to uncertainty and novel challenges,** and whether this affects their impression of their own ability.

References / inspirations

1. Bandura, A. (1986). Social Foundations of Thought and Action. Prentice-Hall.
2. Compeau, D. R., & Higgins, C. A. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly*, 19 (2), 189-211.
3. Gist, M., Schwoerer, C., & Rosen, B. (1989). Effects of alternative training methods on self-efficacy and performance in computer software training. *Journal of Applied Psychology, 74*, 884-891.
4. Klinger, C. M. (2006). Challenging negative attitudes, low self-efficacy beliefs, and math-anxiety in pre-tertiary adult learners. In *Connecting voices in adult mathematics and numeracy: practitioners, researchers and learners.*

** Survey statistics for 2010 are based on 66 students (87% of cohort).