Software Engineering Education: a place in the sun?

ICSE-16 Workshop Summary Report

Anthony Finkelstein

Department of Computing, Imperial College, London SW7 2BZ.
acwf@doc.ic.ac.uk

Virtually every research specialisation in software engineering would be prepared to claim that its particular concern is the most important in the field. I have heard such claims made (I may indeed, in rasher moments, have made such claims) for software maintenance, reuse, formal specification, requirements engineering, and others too numerous to mention.

It may thus seem strange that I should, without blushing and in these widely circulated Proceedings, wish to claim such a status for software engineering education, a specialisation barely regarded as respectable among the majority of researchers.

What are the grounds for this claim? Virtually all the technologies which we believe hold promise of improving software development are dependent on professionally skilled and educated staff. Software engineering education remains our most powerful means of technology transfer and hence of narrowing the gap between what is known in the research community and what is applied in industry and commerce. Despite economic hiccups the skills shortage is still a critical component of the omnipresent software crisis. Industry is spending a large proportion of its software development budget on training and on recruitment to offset the costs of fundamental education problems.

Given the obvious and pressing case for the importance of software engineering education; given, in addition, that most academics are also educators and most industrial researchers devote a substantial proportion of their time to training and education in one form or another; how can we account for the scant attention and limited respect accorded to software engineering education? I can only assume that the dominant factors are cultural or sociological. It certainly is not that software engineering education does not pose major technical challenges, over and above the general problems of teaching a complex professional subject in a stimulating and intellectually coherent manner. Let us rehearse these problems.

Software engineering is, in large part, about scale. Illuminating the essence of a software engineering technique and motivating the students with convincing arguments for its value, without giving examples which are so large as to submerge the student in extraneous detail is extremely difficult.

Software engineering education, with its emphasis on specification and design, makes great demands on students ability at abstraction. The ability to filter away extraneous detail and to make meaningful simplifications in pursuit of a better understanding - to look at "what" rather than "how" - is not a common one. More confusingly the ability of students to abstract differs sharply from student to student in ways which cannot be easily predicted from previous educational attainment or background. Finding an appropriate educational strategy to remedy this is not easy.

However much, as researchers, we bemoan the fact, much of software engineering is still concerned with heuristics, tips, rules-of-thumb acquired through long and often painful experience. Students encountering software engineering for the first time do so with a predominantly scientific (or at any rate mathematical) educational background. Our inability to back up our assertions with strong empirical evidence (or in many cases by rigorous reasoning and proof) runs counter to much of what students have been taught to expect. It is all to easy for them to dismiss the subject as "waffle".

Software engineering builds on application domain expertise. In almost all industrial software engineering practice specifications are based on a large body of common assumptions about the domain and shared knowledge of it held by client and

0270-5257/94 $3.00

software engineer. In education, by contrast, students have little common experience (except of the educational system itself) and only rudimentary understanding of the operation of commerce and industry. Even the most elementary examples may well be inappropriate. Try talking about payroll systems to students who have never received a pay slip or about domestic washing machines to students who have never used one. Even able students may have no intuitions about the operation of a simple device such as a drinks vending machine (the standard text book example). This is to set aside such applications as C3I, process control, accounts receivable, manufacturing systems and so on. Access to and communication of rich domain information poses a fundamental challenge to software engineering educators.

Many of the software engineering methods and techniques which we wish to impart are "heavyweight" in the sense of carrying a large notational overhead or deploying tools which require significant associated training and make use of substantial computational resources. Within the constraints of the educational setting these methods and techniques are difficult to use. Stripping them down, or developing "lightweight" equivalents may be very difficult.

Above all problems the most serious is boring our students. It is also the problem that, as engineering educators, we are most embarrassed to talk about. With a large amount of material to impart, in a necessarily restricted time, and some basic skills that it is absolutely essential all students acquire, the temptation to over-teach or to skip the intellectual challenges, question marks and interesting sidelines, is often overwhelming. It is also difficult to justify novel and risky educational techniques on core curriculum material. We are therefore required to make informed judgements about the importance of different topics, which questions to raise and which to slur, and so on.

The problems of software engineering education would be sufficiently serious to merit concern even if we were able to devote substantial resources to them. Unfortunately higher education is, in most countries, under economic pressure and resources are scarce. This is compounded by a shortage of people, particularly in academic computing, with the skills and motivation to teach software engineering combined with the pressure from industry and accrediting bodies to increase the amount of software engineering and related material in the curriculum.

It is highly appropriate, in these circumstances, that a workshop on software engineering education is to take place at ICSE. It highlights the growing importance which is being attached to software engineering education as a area of concern; it gives respectability to a field frequently and unjustifiably denied it; it provides an opportunity for key research and industrial players to have access to, and participate in, discussion on this topic. Further, the discussion of software engineering education in an genuinely international setting may improve intellectual cross-fertilisation and exchange of ideas.

Clearly I write this prior to the workshop and with access only to the submissions, which will appear in a separate Proceedings. However, it is already clear that the focus of the workshop will be less on some of the "traditional" topics in software engineering education, such as curricula and project work, and more on issues such as distance education, use of new teaching technologies, evaluation and assessment, course evolution and review, transferrable skills, and intercalated education and training. This reflects, I believe, a maturing of debate on software engineering education as more institutions gain substantive experience. It also reflects a certain "hard nosed" demand for techniques which are immediately applicable rather than generalised observations on curricula. These changes may in turn be due to the greater proportion of institutions with programmes in place as against those introducing new programmes and courses in software engineering who have formerly dominated discussion on software engineering education. There remain some issues which are not well covered in the submissions. Prominent among these is the issue of research training in software engineering which is, I feel, worthy of some attention.

In conclusion, software engineering education presents challenging research problems worthy of serious attention by the wider software engineering community. I anticipate a stimulating discussion.