17

Technology and the reflective practitioner

Tom Boyle

We should think about practice as a setting not only for the application of knowledge but also for its generation.

(Schön 1995: 19)

Introduction

The basic practice of teaching in higher education (HE) has been passed on from generation to generation. Those deeply involved in practice may find great difficulty in standing back to critically examine what is happening. It is immanent; it is the framework within which they work; it is not visible from some idealized outside vantage point. However, this situation can also be taken as a productive starting point. The ideas of Donald Schön have been very influential in this respect. For Schön, the detailed, rich knowing-in-action embedded in everyday practice can be the source of powerful, transformative knowledge. We can change inherited education structures, or at least those parts that affect our daily practice, from within.

So, do we remain trapped within the inherited structures of HE teaching, or do we allow them to provide a starting point for the generation of new knowledge, leading to more effective action? This chapter explores the challenges and options in developing the second alternative.

in Alastair Irons & Sylvia Alexander (2004)
Effective learning and teaching in computing (RoutledgeFalmer: London)
ch.17. pp.182-188

Schön and the reflective practitioner: teaching, action research and rigorous knowledge

Schön argues that the 'scholarship of application' should be valued, but his proposals go beyond this point. He argues that we should think about practice as a setting not only for the application of knowledge but also for its generation. The generation of new knowledge starts with a focus on the kinds of knowing, the 'intuitive artistry', that is already embedded in good practice. This 'knowledge in action' is often implicit and unarticulated. A major role for the reflective practitioner is to reflect on this embedded knowledge and articulate it in a way that is generalizable to other situations.

Schön argues that 'reflection in action' is often triggered by surprise. Something does not go as expected. The performer reflects during the ongoing process, restructures the way of doing something and observes the effects of the restructured action. The practitioner can build further reflection, outside of the immediate need to act. This second reflective spiral, 'reflection after the event', helps make explicit in language the nature of the problem, its relation to a class of problems, and the action strategy adopted. This theory of action may then explicitly guide how the practitioner deals with this type of situation in the future. The teacher can then observe and evaluate the impact of the changes made to their action repertoire.

The argument so far points to how a reflective practitioner may improve his or her own practice. However, Schön's primary argument is that this reflection in and on action should form the basis of a new sharable body of knowledge. This body of knowledge should be rigorous, generalizable and open to critical evaluation according to suitable criteria.

If teaching is to be seen as a form of scholarship then it must give rise to new forms of knowledge. This in turn raises the question of how these new forms should be expressed and evaluated. Schön argues that this needs to take the form of 'action research'. The basis for generalizability should not be covering 'laws' as in the mature sciences: it consists rather in framing the problem and the action strategy in such a way that it can be carried over to new situations. The relevance and effectiveness of the knowledge must still be tested in the new situation. The idea of a 'summative' test that allows one to state the final, static value of a given pedagogic technique is inappropriate and irrelevant. We have rather valuable, tested, knowledge and techniques that guide critical application in new situations. The criteria applied to generalizability should be rigorous. We should not expect physical science-type laws, but we should expect evaluated knowledge and techniques that provide informed guidance on how to solve everyday problems of professional practice.

How this 'professional' knowledge is produced, generalized and evaluated may be different from the core discipline view of 'legitimate' academic knowledge.

This difference may produce problems in acceptance of the status and rigour of the techniques and results of action research. This point is returned to later in the chapter. It is useful first to examine a practical problem in the teaching of computing, and critically examine the relevance of Schön's idea of the reflective practitioner.

The crisis in the teaching and learning of programming

Schön states that reflection in and on action is triggered by surprise: professional practice, which should provide the solution to a problem, does not work or does not work as intended. The scale of the reaction, however, is sometimes closer to shock than simply surprise. The 'surprising' problem is that we don't know how to teach programming. Programming is fairly fundamental in computing. We have been teaching programming for a long time (as long as the discipline has been around); yet there are fundamental and deep problems. These problems have been extensively reported in the conferences and workshops run by the LTSN Subject Centre for the Information and Computer Sciences. Take this quote: 'Anyone who has presented an introductory programming module will be all too familiar with students who appear to be totally unable to grasp the basic concepts. Others who come to supervise final year dissertations will have been faced with students who insist that they want to avoid programming at all costs' (Jenkins and Davy 2001: 1).

There are strong, well-established models for how to teach programming. These models are strongly influenced by the discipline base of computing – ideas of structured programming, object-oriented programming etc. It should generate some surprise that they do not apparently work very well. How do lecturers react when faced with this problem? These are both quotes from computing lecturers at e-learning events: 'Our students can't abstract' and 'Our students are unteachable'. The first exclamation I have heard many times over the years; the second is a bit more extreme. These lecturers were really interested in teaching and learning in computing. The fact that they should come up with such remarks is rather disturbing. Rather than trigger reflection on 'failing' practice, the failed practice is explained away because it is the fault of the students

Let's rephrase the quotes: 'We don't know how to help our students learn about the abstractions that are so important in computing', and 'We don't know how to teach our students'.

The first stage is to recognize the nature of the problem – it is a problem with our practice. To blame the students, before we have fully investigated our own practice, is a cop-out. If there are problems in the teaching of computing, the key questions are: what are the sources of these problems, and what can we do about it? Can we reflect on our practice and produce an improved understanding? Can this in turn lead to improved practice and more effective outcomes for the students? When faced with significant problems like this, can the technology we as a discipline

develop help us to solve this type of problem? Has technology any particular role in helping us to deal with significant problems like this?

Technology and reflective practice in computing

Schön paid particular attention to the role of information technology (IT) in facilitating reflective practice. His ideas had been shaped by his participation in Project Athena at MIT. For many disciplines, IT acted to create a challenge - this strange new technology had implications for teaching that had to be responded to. In computing, however, the technology was familiar. It was used in everyday teaching. The 'surprise' value it had for many other disciplines did not apply. This often led to complacency in reacting to the impact of IT on the teaching of computing as a subject.

It is well beyond the scope of this chapter to provide a comprehensive review of technology-enhanced reflective practice in computing. Despite the barriers mentioned earlier there has still been considerable action research in computing that uses IT as a catalyst for change. The teaching of programming has been a particularly rich area of study. The brief review will concentrate on exemplar studies in this area. The rich base from which these exemplars are generated (across the whole curriculum, as well as in programming) is reported especially in the annual conferences and workshops of the LTSN Subject Centre for the Information and Computer Sciences (LTSC-ICS 2003), and in the publications of SIGCSE of the ACM (SIGCSE 2003).

Boyle *et al.* (1994) used a combination of an emerging technology (hypertext) and constructivist pedagogical ideas to produce a novel learning environment for programming. The CLEM system implemented a 'guided discovery' pedagogy for learning Modula-2 that led to marked improvements in module pass rates (Boyle *et al.* 1994). At a broader level, Ben Ari (2001) has argued for the widespread adoption of constructivist ideas in the teaching and learning of computing. Hohmann *et al.* (1992) focused on the higher order patterns, or schemas, typical of expert behaviour rather than program syntax. They described their *SODA* system as a tool for the doing and learning of software design. Marshall (1995) exploited the technological affordances of the then emerging worldwide web to build a series of tools for learning aspects of computing, including the language C++. Again the work revolved around a linking of pedagogical ideas with the opportunities offered by the technology.

Not all proposed developments are closely tied to the technology. Jenkins (2002) suggests a number of techniques for motivating and supporting students that are often focused more on the social-educational environment. Davis (2001) also points to organizational changes to deal with issues such as the diversity in student knowledge and abilities. A 'blended' approach to improving the learning of introductory programming was adopted in a major project at London Metropolitan University and Bolton Institute. The blended approach involves significant changes in both the offline and online aspects of the learning environment. In this case changes were made to the curriculum for learning Java (including an emphasis on a sequence of programs producing visible and engaging graphical output), to the social organization of the course, and to the development of extensive e-learning support. The most novel feature of the e-learning was the provision of a series of 'learning objects' in both text and multimedia form. These were based on a combination of software engineering and constructivist pedagogical principles (Boyle 2003). Access to the learning objects was provided through a conventional VLE (WebCT). These learning objects were specifically developed to be 'reusable' across different institutions. A comprehensive evaluation of this project was carried out. Marked improvements in pass rates were found across all four modules in the two institutions. These ranged from 12 to 23 percentage point improvements (Chalk *et al.* 2003).

This brief review of exemplars has focused on programming, but the impact of technology-assisted learning in exploring and extending pedagogical practice has been felt from studies of software system behaviour (e.g. Milne and Rowe 2002) all the way through to information systems development (e.g. Ward 2000).

Developing rigorous action research: barriers and opportunities

Universities have to learn to accommodate forms of scholarship based on reflective practice. One major impediment is the power of disciplinary in-groups. Criteria for high-quality research tend to exclude pedagogical research based on reflective practice. The Research Assessment Exercise (RAE), despite the lip-service paid to pedagogy, has a very traditional focus on what counts as good research. Schön (1995), however, points to a second major impediment – the struggle of those interested in teaching and learning to turn their practice into appropriately rigorous research.

At this point it seems relevant to critique Schön's position. Schön speaks of competent practice and the generation of transferable knowledge from reflection on that practice. But what happens when the practice is not 'competent'? What happens when there is a crisis in the use of traditional techniques of teaching and learning? One source of knowledge is reflective practice. However, if this remains within the bounds of the discipline then the core aspect of the problems may not be tackled. Certain problems may require more dramatic solutions than can be generated by subject-based reflection. They require input from the wider field of pedagogical scholarship. The 'new' pedagogical ideas need to be synthesized with deep subject-based knowledge. When this combination is achieved, significant progress can be made.

Two recurring problems in learning computing, for example, are dealing with complexity and abstraction. The particular combination of complexity and abstraction

encountered often leads to significant problems. There is, however, an extensive literature on research dealing with these problems that may be highly relevant to solving problems within computing (Boyle 1997). The discipline of reflective practice requires not only reflection on our own practice, but wider scholarship that will ground the attempted solutions in deeper pedagogical knowledge.

A further barrier for the impact of valid new ideas is 'opinion-based' practice. It is still the norm in HE that each lecturer can make his or her individual decisions about pedagogy. We need to move towards 'evidence-based practice' where we actively examine our assumptions, seek evidence as to their effectiveness and are prepared to change when the evidence indicates this need.

Extended reflective practice, where we critically examine our assumptions and practices from a pedagogical as well as a subject base, is very demanding. Tutors who engage in this activity require support and recognition. They also require a forum for the exploration, exchange and evaluation of these ideas. The LTSN Subject Centres have worked hard to create these support structures. The LTSN Subject Centre for the Information and Computer Sciences has run an extensive series of workshops and an annual conference to provide for the critical exchange of ideas and innovations. The new Higher Education Academy should provide a continuation of this subject-based support within an overarching organization dealing with generic as well as subject-based issues. From a very poor base a decade ago, explicit support and recognition for good teaching is increasing. This is very welcome. Major challenges remain in understanding, acting on, and transforming the teaching and learning of computing. The challenge for the reflective practitioner is great. As a community we need to fight for the widespread recognition of this rigorous form of scholarship.

References

Ben-Ari, M. (2001) 'Constructivism in Computer Science Education', *Journal of Computers in Mathematics and Science Teaching*, 20(1): 45–73.

Boyle, T. (1997) *Design for Multimedia Learning*. London: Prentice Hall.

Boyle, T. (2003) 'Design Principles for Authoring Dynamic, Reusable Learning Objects', *Australian Journal of Educational Technology*, 19(1): 46–58. http://www.ascilite.org.au/ajet/ajet19/res/boyle.html.

Boyle, T., Gray, J., Wendl, B. and Davies, M. (1994) 'Taking the Plunge with CLEM: The Design and Evaluation of a Large Scale CAL System', *Computers and Education*, 22(1/2): 19–26.

Chalk, P., Boyle, T., Pickard, P., Bradley, C., Jones, R. and Fisher, K. (2003) 'Improving Pass Rates in Introductory Programming', paper presented at the 4th Annual Conference of the LTSN-ICS, August.

Davis, H. (2001) 'Managing Diversity: Experiences of Teaching Programming Principles', in *Proceedings of the 2nd Annual Conference of the LTSN Centre for the Information and Computer Sciences*, pp. 53–9. Belfast: University of Ulster.

Hohmann, L., Guzdial, M. and Soloway, E. (1992) 'SODA: A Computer-aided Design Environment for the Doing and Learning of Software Design', in *Proceedings of the Computer Assisted Learning 4th International Conference*, ICCAL '92. Berlin: Springer-Verlag, pp. 307–19.
Jenkins, A. (2002) 'On the Difficulty of Learning to Program', in *Proceedings of the 3rd Annual Conference of the LTSN Centre for the Information and Computer Sciences*, pp. 53–8. Belfast, University of Ulster.
Jenkins, T. and Davy, J. (2001) 'Diversity and Motivation in Introductory Programming', *Italics*, 1(1): http://www.ics.ltsn.ac.uk/pub/italics/issue1/tjenkins/003.html
LTSN-ICS (2003) *The LTSC-ICS*. http://www.ics.ltsn.ac.uk/
Marshall, A. D. (1995) 'Developing Hypertext Courseware for the World Wide Web', in H. Maurer (ed.) *Educational Multimedia and Hypermedia: Proceedings of Ed-Media 95*. Norfolk, VA: AACE, pp. 418–23.
Milne, I. and Rowe, G. (2002) 'OGRE - 3D Program Visualization for C++', in *Proceedings of the 3rd Annual Conference of the LTSN Centre for the Information and Computer Sciences*, p. 102. Belfast: University of Ulster.
Schön, D. A. (1995) 'Knowing-in-action: The New Scholarship Requires a New Epistemology', *Change*, November/December: 27–34.
SIGCSE (2003) *The SIGCSE*. http://www.acm.org/sigcse
Ward, R. (2000) 'Pedagogy and Technology in Computer-Based Learning: Achieving the Right Balance', in *Proceedings of the Pedagogy versus Technology LTSN-ICS Workshop*. http://www.ics.ltsn.ac.uk/pub/pedagogy/WOLVER/index.htm

Conclusion