



Higher Education Research and Development Society of Australasia, Inc

# **Transforming Knowledge into Wisdom**

## **Holistic Approaches to Teaching and Learning**

*Proceedings of the*

## **27<sup>th</sup> HERDSA Annual Conference**

4-7 July 2004

Miri, Sarawak

Armarego, J. (2004) Towards achieving Software Engineering wisdom, in *Transforming Knowledge into Wisdom, Proceedings of the 27th HERDSA Annual Conference, Miri, Sarawak, 4-7 July 2004: pp 5.*

Published 2004 by the  
Higher Education Research and Development Society of Australasia, Inc  
PO Box 27, Milperra, NSW 2214, Australia  
[www.herdsa.org.au](http://www.herdsa.org.au)

ISSN: 0155-6223  
ISBN: 0 90 8557 58 2

This research paper was reviewed using a double blind peer review process that meets DEEWR requirements. Two reviewers were appointed on the basis of their independence, expertise and experience and received the full paper devoid of the authors' names and institutions in order to ensure objectivity and anonymity. Where substantial differences existed between the two reviewers, a third reviewer was appointed. Papers were evaluated on the basis of originality, quality of academic merit, relevance to the conference theme and the standard of writing/presentation. Following review, this full paper was presented at the international conference.

Copyright© 2004 HERDSA and the authors. Apart from any fair dealing for the purposes of research or private study, criticism or review, as permitted under the Copyright, Design and Patent Act, 2005, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers at the address above.

# Towards achieving Software Engineering wisdom

**Jocelyn Armarego**

Murdoch University Perth Australia  
jocelyn@eng.murdoch.edu.au

***Abstract:** This paper provides a background for changes made to the Software Engineering (SE) curriculum at Murdoch University. It charts the progress made by SE staff in gaining wisdom in SE education issues as they introduce innovation, evaluate and then modify the curriculum based on reflection. The learning environment that has evolved fosters self-directed learning and reflective practice through a co operative (cognitive) apprenticeship based on Design Studios. SE students benefit through the increased opportunity to learn to make appropriate use of knowledge gained through their studies (and hence acquire SE wisdom in their own right). SE staff benefit from the double-loop approach as the espoused theory of teaching is aligned with the theory in practice.*

***Keywords:** Software engineering education; design studio; problem-based learning.*

## Introduction

Murdoch University's School of Engineering Science has, since 1995, provided a suite of programs in Software Engineering (SE). Our teaching objectives are focused on producing engineers with a special skill in software. We expect graduates from our Bachelor of Engineering (BE(SE)) to find career opportunities in both professional engineering industries that have a strong interest in software as well as the full range of IT disciplines where the design and implementation of quality software is considered a priority. Pursuing these objectives has meant a gradual shift from more traditional engineering learning as we address characteristics specific to SE.

One of the issues that have plagued SE education has historically been that of integration – that the methods, techniques, tools, etc acquired within a few isolated units do not permeate the students' approach to other software-related tasks within their programme of study.

A second issue is the multi-disciplinary nature of the skills and knowledge required to be active as competent professionals. In SE, underlying disciplines of central importance are psychology, computer science and discrete mathematics, while disciplines such as physics and continuous mathematics only support some applications (Zucconi, 1995). She suggests that, as well as technical competence, an SE needs to be well organised, able to work as a member of a multi-disciplinary team, and able to work within the scope of the employer's policies and procedures and society's tenets.

Added to these is the ubiquitous need to engage in life-long learning. The speed with which technology evolves, the multiplicity of its impact on society and the ramifications of that impact mean that metacognitive and knowledge construction skills as well as adaptability become vital.

In order to address these issues, Murdoch Engineering has moved progressively from a traditional learning model to exploiting models that enhance creativity, adaptability and productive thinking. Implementation of problem-based learning (PBL) in individual units has led to a Design Studio model that incorporates PBL within an integrated environment.

### **Background: Educating for an engineering discipline of software**

Over 35 years ago, those involved in the development of software agreed that one mechanism for dealing with the intrinsic difficulties (e.g., complexity, visibility, and changeability (Brooks, 1986)) of developing software was to embed its production within an applied science environment. Royce (1970) was the first to note explicitly that an engineering approach was required, in the expectation that adhering to a defined, repeatable process would enhance software quality.

This ethos is exemplified in the various definitions provided for the engineering discipline of software [my emphasis]:

*Engineering is the systematic application of scientific knowledge.... Software engineering is that form of engineering that applies the principles of computer science and mathematics to achieving cost-effective solutions to software problems.* SEI software engineering definition (Ford, 1990)

*The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.* IEEE Standard. 610-1990 (IEEE, 1999)

Increasingly, approaches to educating software developers model scientific and engineering methodologies, with their focus on process and repeatability. In general this education is based on a normative professional education curriculum, in which students first study basic science, then the relevant applied science (Waks, 2001), so that learning may be viewed as a progression to expertise through task analysis, strategy selection, try-out and repetition (Winn & Snyder, 1996).

Software technology is seen as a rapidly shifting landscape: new methods, tools, platforms, user expectations, and software markets underscores the need for SE education that provides professionals with the ability to adapt quickly (Garlan, Tomayko, & Gluch, 1997). In addition software development incorporates insight-driven knowledge discovery (Guindon, 1989) facilitated by opportunistic behaviour (Guindon, 1990; Visser, 1992). Participants in the process must remain sensitive to progressive modifications (Gigch, 2000) which lead not to a problem-solution, but to an “evolved fit” acceptable to all stakeholders within the problem space.

Industry therefore requires professionals who integrate into the organisational structure, and, rather than cope specifically with today’s perceived problems, have models, skills and analytical techniques that allow them to evaluate and apply appropriate emerging technologies. Professional practitioners with such skills become agents of change (Garlan et al., 1997).

Macauley and Mylopoulos (1995) acknowledge that a standard university lecture cannot achieve what industry requires. Others also note the inadequacy of formal education in training competent software professionals (Lethbridge, 2000; Robillard, 1999). For Macauley

and Mylopoulos efficient software development activities “*require a certain level of knowledge and maturity which can only be gained through experience in dealing with practical problems*”.

Attempts to address these issues have been made in the area of software development education, where the traditional “lecture + laboratory work + assessment tasks” are augmented by either a capstone project which simulates a start-to-finish development environment or an industry-based placement, typically towards the completion of the qualification. These are seen to provide opportunities for both authentic and experiential learning, with emphasis not so much on acquiring knowledge as on increasing students’ ability to perform tasks. While accepted as valuable, this approach is flawed in several respects:

- the opportunity (project or placement) is presented as an aid to content learning rather than a substitute;
- it focuses on *know-how* which will allow students to gain competence to practice within given frameworks (but not necessarily outside of them, therefore limiting adaptability);
- students are expected to transfer skills acquired to the world of work, but without them necessarily being rooted in cognitive content and professional judgement;

(based on Savin-Baden (2000)).

Although these provide experiential learning opportunities, learning from experience is not automatic: it requires transfer (the ability to apply something learned in one situation to another setting (Kearsley, 2000)) to be enabled. This transfer is enhanced where there is a focus on metacognitive strategies and reflection. It is this facet that is often missing from capstone projects and placements.

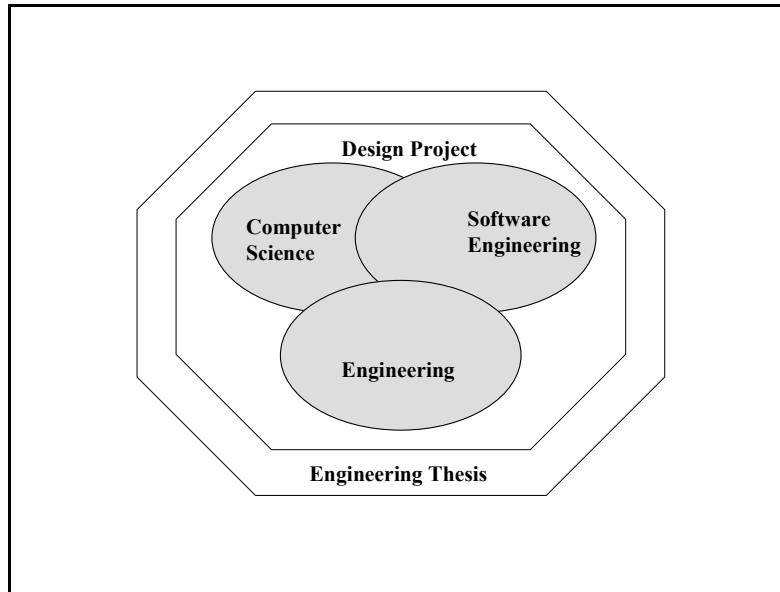
As Waks (2001) explains, in this normative model of professional education science provides “*a rational foundation for practice*” [original emphasis], with practical work at the last stage of the curriculum, where students are expected to apply science learned earlier in the curriculum to real-life problems. He continues that the crisis of the professions arises because real-life problems do not present themselves neatly as cases to which scientific generalisations apply. This becomes, then, an issue of espoused theory versus theory-in-use (Argyris & Schön, 1974).

The poor fit between the characteristics of action in the domain and those of the learning model produce an “incorrect” learning environment, where the learner is not directed to the important features of the domain (Gobet & Wood, 1999). Patel et al. (2000) argue that learners in a traditional setting focus on skills that will yield higher grades as an immediate objective. With the relevance of domain knowledge not fully understood, cognitive skills related to exam techniques acquire importance though they do not model real life situations. The learning, in many cases, is reduced to assignment hopping with “just-in-time” and “just-enough” learning to fulfil the assessment tasks.

In an attempt to address these issues, Murdoch Engineering has focused over the past several years on balancing a foundation in the content with elements of creativity and experience based on practice.

## Integrating an SE curriculum

The original curriculum for the BE(SE) may be viewed as three intersecting components (Figure 1), all within an envelope that integrates the knowledge gained.



**Figure 1: BE(SE) Curriculum components**

The primary components:

- *Computer Science* – these units cover fundamental aspects (e.g., programming, algorithm analysis, database and operating system concepts) and form the basis of technical knowledge and skills in software and hardware
- *Software Engineering* – these units focus on SE theory and practice and form the basis of core knowledge and skill in software development and evolution
- *Engineering* – these units offer knowledge and skills in engineering practice and principles and are common to all our engineering students. They include natural sciences, mathematics, management, ethics provide the basis for:
- *Design Project/Engineering Thesis* – these are also common to all engineering students, though the domain of application targets the appropriate discipline of study. While the *Project* may be industry-based, it is run under controlled conditions, and carefully monitored by academic staff. The *Thesis* focuses on industry and may be linked to work-place experiences.

Underlying these is a common set of support material and resources, including web resources, process tools and documentation templates. Students are encouraged to apply this material as much as possible, and in some instances are formally required to do so. This integrated environment is described in Armarego et al. (2001).

In 2002 the model was changed so that the Design Project and Thesis (i.e. group and individual capstones) were merged and effectively outsourced – students now undertake a graduate-level internship with appropriate local industry, under the supervision of both academic and industry mentor.

## Towards SE wisdom: Phase 1

The reduced opportunity for group-based projects was one trigger for the restructure of some of the core SE units (specifically Advanced Software Design (ASD II)). Other triggers included:

- students in their final year of studies raised issues regarding the need to apply knowledge acquired to more “real world” scenarios – and to engage with the material on offer
- to provide students with a taste of the types of *wicked* problems they would encounter during their internship. Exposure to the uncertainties and inconsistencies associated with real problems in a controlled environment enhances graduates’ potential to deal in their own turn with ill-structured problems within an organisational context
- a belief that learning beyond the foundation stages may best be achieved through situational problems with rich contextual information (Dreyfus & Dreyfus, 1986).

As detailed in Armarego (2002) a PBL model was applied, based on Koschman et al. (1994) and in the context of the phased development of a software product. Student evaluation undertaken in weeks 4, 7, 11 and 13 highlighted both concerns and benefits:

- the need to learn new content as well as adapt previous knowledge;
- dependence on other members of the large team (13 members) both for achieving the tasks and for critical assessment components through peer and self assessment;
- the lack of stability in teams and task (students were rotated into and out of teams, roles and problem component) requiring a need to “come up to speed” very quickly at each change.

Representative positive comments include:

- “We learn so much ‘practical’ stuff from this project, it would be good to get another chance to actually do it right”;
- “Learnt a lot about design skills and approaches for problems”
- “Interesting group experience”;
- “You need more practical application of the theory you teach (ASD II style)”.

The restructured ASD II course was seen to provide students with a number of opportunities:

- to identify, analyse and solve a number of issues, repetitively. This acts as preparation for professional employment;
- to practise the art as well as science of SE in a controlled setting;
- to test the understanding of theory, its connection with application, and develop theoretical insight;
- to deal with incompleteness and ambiguity;
- to think independently and work co operatively, fostering insight into individual strengths and weaknesses;

(Armarego, 2002).

An important issue was the alignment that existed between the learning and the assessment (Elton, 2000). It included extensive formative components, and focussed on self and peer assessing of both the artefacts during production and the group process.

## Towards SE wisdom: Phase 2

An unexpected problem was encountered during the unit: while students were comfortable with the idea of directing their own learning during the Design Project and Thesis, they felt (very strongly, at times) that within a formal unit, they should be taught. As noted in Armarego & Clarke (2003) a reasonably high level of teacher direction in prerequisite SE units (specifically Requirement Engineering (RE)) did not challenge student expectations of traditional learning. Initial resistance to changes made to ASDII showed that these expectations were still evident two years later in their studies and highlighted a need to emphasise student-centred learning earlier in the curriculum – the final year was too late.

The nature of the RE component of software development (opportunistic, exploratory, creative, emergent (Bubenko, 1995; Guindon, 1989; Maiden & Gizikis, 2001; Nguyen & Swatman, 2000)) implies a need to incorporate creativity-enhancing activities within the curriculum. Cropley and Cropley (1998) suggest that the process of creativity and innovation in engineering is poorly understood and not adequately fostered in undergraduate teaching. This deficiency results in an engineering culture that is frequently resistant to the factors that promote creativity and innovation. Yet providing a learning environment that enhances the opportunity for creative thinking has the potential for long-term benefits to SE students. There is evidence that students who have been taught to explore different ways to define problems (the prime objective of RE) engage in more creative problem solving over the longer term (Baer, 1988).

A focus on flexibility and productive thinking is also necessary, so that students learn to use past experience on a general level, while still being able to deal with each new problem situation in its own terms. Gott et al. (1993) posit that this adaptive/generative capability suggests the performer not only knows the procedural steps for problem solving but *understands when to deploy them and why they work*, in effect is wise in the use of them.

Funding was gained to address this issue. A creative PBL model (Figure 2) was developed to

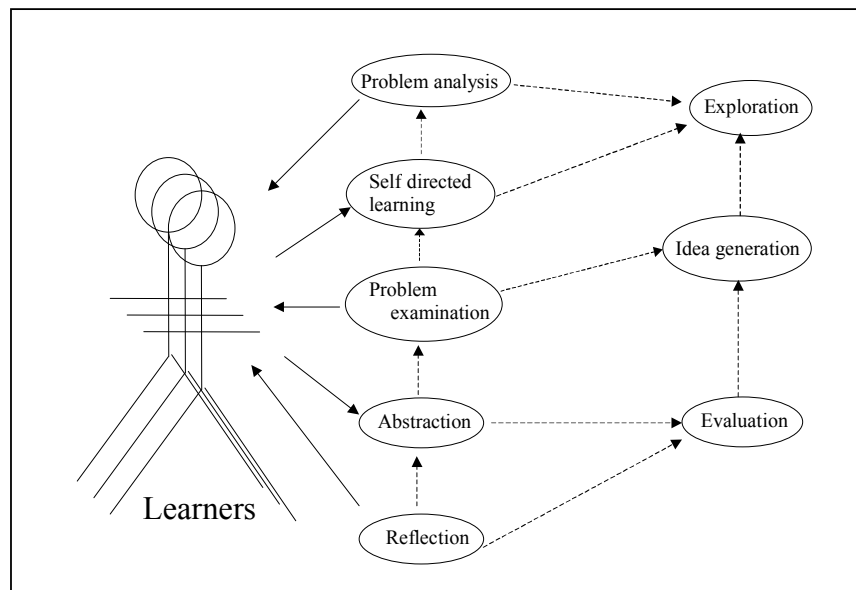


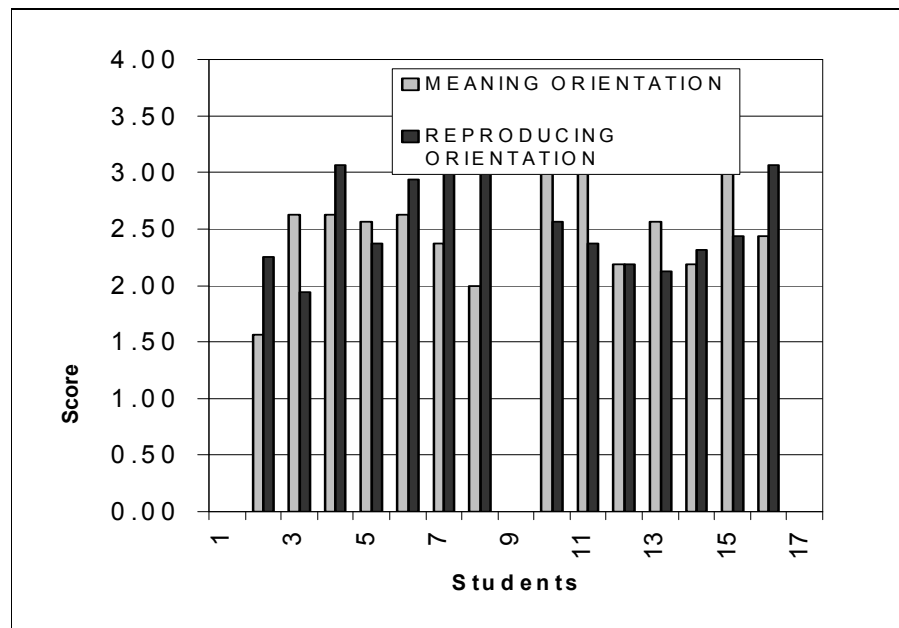
Figure 2: The creative PBL model

focus on creativity and divergent thinking and address difficulties in flexibility noted by Thomas et al. (2002). Instead of students aimed at finding the single, best, “correct” answer to a standard problem in the shortest time (convergent thinking) the aim is to redefine or discover problems and solve them by means of branching out, making unexpected associations, applying the known in unusual ways, or seeing unexpected implications. The PBL process was used to anchor elements of creativity adapted from Edmonds & Candy (2002). This model was applied in 2003 (Armarego, 2004; Armarego & Clarke, 2003).

Two issues were highlighted during the evaluation to this approach:

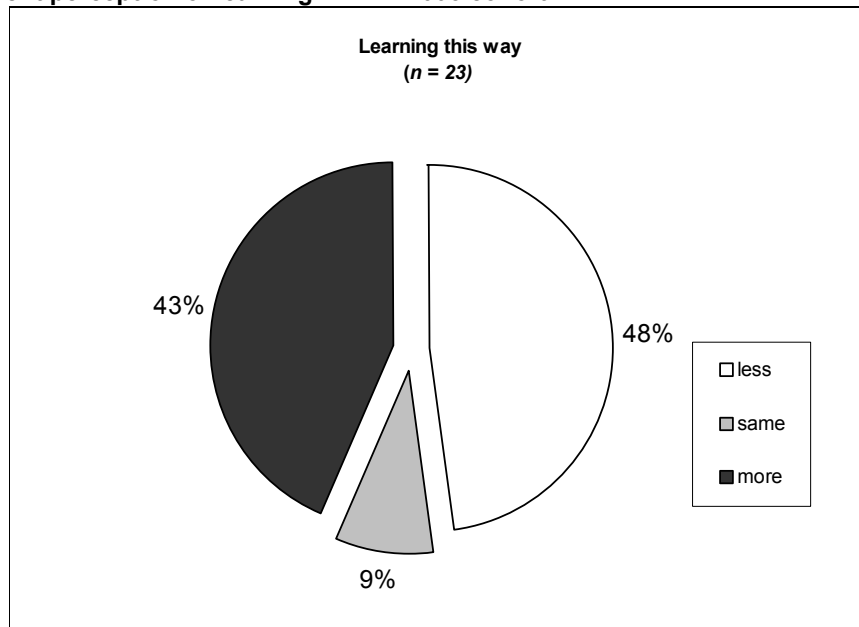
- Assessment – redesign of the unit was time-intensive, therefore the backup of being able to run in previous (workshop-based) mode was necessary. There then existed a mismatch between the espoused theory (student-centred, problem-based learning) and the final assessment component, an exam. As noted in Elton (2000) “*we want students to learn with understanding and be assessed for it*”. As displayed in Table 1 a post-hoc Approaches to Study Inventory (using a 32-item instrument confirmed by Richardson (1990)’s work to possess adequate internal consistency and test-retest reliability) showed that students were very much sitting on the fence between learning for meaning (mean 2.53, standard deviation 0.43) and learning for reproduction (mean 2.56, standard deviation 0.41). However, learning for understanding is less reliably assessed than memory learning and learning that achieves some form of creativity will be quite radically different for different students (Elton, 2000). The assessment approach applied in ASD II appears appropriate for addressing this issue.

**Table 1: Approaches to study survey – 2003 student cohort taking RE unit**

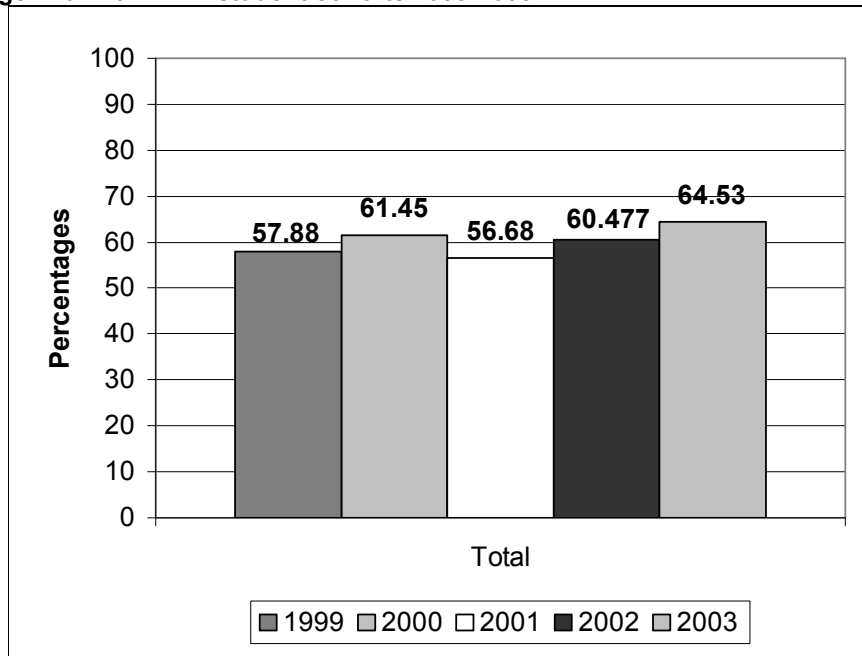


- Does this learning model allow for the diversity of learning styles students exhibit? Students again were found to be sitting on the fence with regards to their perception that they had learnt more or less from this approach (see Table 2), although as a cohort the class did well (see Table 3).

**Table 2: Student perception of learning in RE - 2003 cohort**



**Table 3: Average final mark - RE student cohorts 1999-2003**



### **Towards SE wisdom: Phase 3**

While these interventions have some measure of success, at least in terms of learning outcomes, one issue to be addressed is that innovation introduced into one unit may be undermined if traditional approaches are maintained elsewhere in the students' programme – so that benefits may only be apparent or are enhanced if it is introduced across the entire curriculum. During 2004 a complete restructure of the final two years of the BE programs is being undertaken to introduce an integrated Design Studio approach.

In attacking the normative professional education curriculum, Schön looks to an alternative epistemology of practice “*in which the knowledge inherent in practice is understood as artful doing*” (Schön, 1983). He notes that in the ordinary form of practical knowledge practitioners do not think about what they are doing, except when puzzled or surprised. Schön named this reflecting-in-action, and argued that it is central to our ability to act effectively in unique, ambiguous, or divergent situations.

For Schön, practitioners (including engineers), have their own “esoteric” knowledge codes woven right into their practices. They apply tacit knowledge-in-action, and when their messy problems do not yield to it, they “reflect-in-action,” and in the languages specific to their practices. Even when they do stop to reflect *on* action, they think in the language of practice, not the language of science. For Schön the ideal site of education for reflective practice is the Design Studio. Under the close supervision of a master practitioner serving as coach the novice learns the vocabularies of the professional practice in the course of learning its “operational moves”. In making the moves, talking about them and even talking about their talk about them (meta-reflection), the novice and master “*negotiate the ladder of reflection*” (Schön, 1987).

Schön’s view of professional practice as design has three implications:

- it is learnable but not didactically or discursively teachable: it can be learned only in and through practical operations;
- it is holistic: its parts cannot be learned in isolation. It must be learned as a whole because all components of a situation have meaning;
- designing depends upon the ability to recognise desirable and undesirable qualities of the discovered world. But novice students do not possess this ability, and it cannot be conveyed to them by verbal descriptions, *only* in the operational context of the task.

### **Achieving SE wisdom**

This view of professional education has implications for the design of teaching:

- academic *learning* must be situated in the domain of the objective: the activities must match that domain;
- academic *teaching* must address both the direct experience of the world, and the reflection on that experience that will produce the intended way of representing it;

(Laurillard, 1993).

The progression to Design Studios in SE is a journey being undertaken by Murdoch Engineering academics in empowering graduates to be industry-ready. SE staff benefit from the double-loop approach as the espoused theory of teaching becomes aligned with the theory in practice. It provides learning situations to examine and experiment with our theories of action (Argyris & Schön, 1974). For the student, the collaborative nature of the learning environment that has evolved transcends the classroom, fostering self-directed learning and reflective practice through a co operative (cognitive) apprenticeship (Berkenkotter & Huckin, 1995) that integrates class and work experience. The future will test the wisdom of undertaking this journey.

## References

- Argyris, C., & Schön, D. A. (1974). *Theory in practice: Increasing professional effectiveness*. San Francisco: Jossey Bass.
- Armarego, J. (2002). *Advanced Software Design: a case in problem-based learning*. Paper presented at the CSEET2002 15th Conference on Software Engineering Education and Training, Covington (Ke).
- Armarego, J. (2004). *Student perceptions of quality learning: evaluating PBL in Software Engineering*. Paper presented at the Seeking Educational Excellence: 13th Teaching Learning Forum, Perth.
- Armarego, J., & Clarke, S. (2003). *Preparing students for the future: learning creative software development - setting the stage*. Paper presented at the Learning for an Unknown Future: Proceedings of the Annual International HERDSA Conference, Christchurch (NZ).
- Armarego, J., Fowler, L., & Roy, G. G. (2001). *Constructing Software Engineering Knowledge: development of a learning environment*. Paper presented at the In search Of a Software Engineering Profession: CSEE&T2001 14th Conference on Software Engineering Education and Training, Charlotte (NC).
- Baer, J. M. (1988). Long term effects of creativity training with middle-school students. *Journal of Adolescence*, 8, 183-193.
- Berkenkotter, C., & Huckin, T. N. (1995). *Genre Knowledge in Disciplinary Communication: Cognition/Culture/Power*. Hillsdale (NJ): Lawrence Erlbaum Associates.
- Brooks, F. P. (1986). *No silver bullet - essence and accidents of software engineering*. Paper presented at the Proceedings of Information Processing 86: the IFIP 10th World Conference, Amsterdam.
- Bubenko, J. (1995). *Challenges in Requirements Engineering: keynote address*. Paper presented at the RE'95: Second IEEE International Symposium on Requirements Engineering, York (UK). Retrieved from <http://http://www.dsv.su.se/~janis/RE95.html>
- Cropley, D. H., & Cropley, A. J. (1998). *Teaching Engineering Students to be Creative - Program and Outcomes*. Paper presented at the Australasian Association of Engineering Education: 10th Annual Conference.
- Dreyfus, H. L., & Dreyfus, S. E. (1986). *Mind over Machine*. New York: Free Press.
- Edmonds, E., & Candy, L. (2002). Creativity, art practice and knowledge. *Communications of the ACM*, 45(10), 91-95.
- Elton, L. (2000). *Matching teaching methods to learning processes: dangers of doing the wrong thing righter*. Paper presented at the 2nd Annual Conference of the Learning in Law Initiative. Retrieved from <http://http://www.ukcle.ac.uk/lili/2000/elton.html>
- Ford, G. (1990). *SEI Report on Undergraduate Engineering Education, Undergraduate Software Engineering Education* (No. CMU/SEI-90-TR-003). Pittsburgh (PA): Software Engineering Institute/Carnegie Mellon University.
- Garlan, D., Tomayko, J. E., & Gluch, D. (1997). *Agents of Change: Educating Future Leaders in Software Engineering*.
- Gigch, J. P. v. (2000). Metamodeling and problem solving. *Journal of Applied Systems Studies*, 1(2), 327-336
- Gobet, F., & Wood, D. (1999). Expertise, models of learning and computer-based tutoring. *Computers & Education*, 33, 189-207.
- Gott, S. P., Hall, E. P., Pokorny, R. A., Dibble, E., & Glaser, R. (1993). A naturalistic study of transfer: adaptive expertise in technical domains. In D. K. Detterman & R. J. Sternberg (Eds.), *Transfer on Trial: intelligence, cognition and instruction* (pp. 258-288). Norwood (NJ): Ablex.
- Guindon, R. (1989). The process of knowledge discovery in system design. In G. Salvendy & M. J. Smith (Eds.), *Designing and Using Human-Computer Interfaces and Knowledge Based Systems* (pp. 727-734). Amsterdam: Elsevier.
- Guindon, R. (1990). Knowledge exploited by experts during software systems design. *International Journal of Man-Machine Studies*, 33, 279-304.
- IEEE. (1999). *IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 610-1990, IEEE Standards Software Engineering, Volume 1: The Institute of Electrical and Electronics Engineers*.
- Kearsley, G. (2000). *Explorations in Learning & Instruction: the theory in practice database*. Washington (DC): George Washington University. Retrieved from <http://http://www.gwu.edu/~tip/engineer.html>
- Koschmann, T. D., Myers, A. C., Barrows, H. S., & Feltovich, P. J. (1994). Using technology to assist in realising effective learning and instruction: a principled approach to the use of computers in collaborative learning. *The Journal of the Learning Sciences*, 3(3), 227-264.
- Laurillard, D. (1993). *Rethinking University Teaching: a framework for the effective use of educational technology*. London: Routledge.
- Lethbridge, T. C. (2000). What knowledge is important to a software professional? *IEEE Computer*, 33(5), 44-50.

- Macauley, L., & Mylopoulos, J. (1995). Requirements Engineering: an educational dilemma. *Automated Software Engineering*, 4(2), 343-351.
- Maiden, N., & Gizikis, A. (2001). Where do requirements come from? *IEEE Software*, 18(5), 10-12
- Nguyen, L., & Swatman, P. A. (2000). *Complementary Use of ad hoc and post hoc Design Rationale for Creating and Organising Process Knowledge*. Paper presented at the Proceedings of the Hawaii International Conference on System Sciences HICSS-33, Maui (Hawaii).
- Patel, A., Kinshuk, & Russell, D. (2000). Intelligent tutoring tools for cognitive skill acquisition in life long learning. *Educational Technology & Society*, 3(1), 32-40.
- Richardson, J. T. E. (1990). Reliability and replicability of the Approaches to Studying questionnaire. *Studies in Higher Education*, 15, 155-168.
- Robillard, P. N. (1999). The role of knowledge in software development. *Communications of the ACM*, 42(1), 87-92.
- Royce, W. W. (1970). *Managing the development of large software systems: concepts and techniques*. Paper presented at the IEEE WESCON.
- Savin-Baden, M. (2000). *Problem-based Learning in Higher Education: untold stories*. Buckingham (UK): Society for Research into Higher Education and Open University Press.
- Schön, D. A. (1983). *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books.
- Schön, D. A. (1987). *Educating the Reflective Practitioner: Towards a New Design for Teaching in the Professions*. San Francisco: Jossey-Bass Inc.
- Thomas, J. C., Lee, A., & Danis, C. (2002). Enhancing creative design via software tools. *Communications of the ACM*, 45(10), 112-115.
- Visser, W. (1992). Designers' activities examined at three levels: organisation strategies and problem-solving processes. *Knowledge-Based Systems*, 5(1), 92-104.
- Waks, L. J. (2001). Donald Schon's Philosophy of Design and Design Education. *International Journal of Technology and Design Education*, 11, 37-51.
- Winn, W., & Snyder, D. (1996). Cognitive perspectives in psychology. In D. H. Jonassen (Ed.), *Handbook of Research for Educational Communications and Technology* (pp. 112-142). New York: Simon & Schuster Macmillan.
- Zucconi, L. (1995). *Essential knowledge for the practicing Software Engineer and the responsibilities of university and industry in her education*. Paper presented at the Software engineering education: 8th SEI Conference on software engineering education, New Orleans.

Copyright © 2004 Jocelyn Armarego: The author assigns to HERDSA and educational non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive licence to HERDSA to publish this document in full on the World Wide Web (prime sites and mirrors) on CD-ROM and in printed form within the HERDSA 2002 conference proceedings. Any other usage is prohibited without the express permission of the authors.