Learning requirements engineering within an engineering ethos

Author(s): Armarego, Jocelyn

Year: 2004

Source: AWRE'04 (9th Australian Workshop on Requirements Engineering), Adelaide, 6-7 Dec. 2004, pp. 11.1-11.11.

Official URL: http://awre2004.cis.unisa.edu.au/

Learning Requirements Engineering within an Engineering Ethos

Jocelyn Armarego
School of Engineering Science, Murdoch University Western Australia
jocelyn@eng.murdoch.edu.au

Abstract

An interest in educating software developers within an engineering ethos may not align well with the characteristics of the discipline, nor address the underlying concerns of software practitioners. Education for software development needs to focus on creativity, adaptability and the ability to transfer knowledge. A change in the way learning is undertaken in a core Software Engineering unit within a university's engineering program demonstrates one attempt to provide students with a solid foundation in subject matter while at the same time exposing them to these real-world characteristics. It provides students with a process to deal with problems within a metacognitive-rich framework that makes complexity apparent and lets students deal with it adaptively. The results indicate that, while the approach is appropriate, student-learning characteristics need to be investigated further, so that the two aspects of learning may be aligned more closely.

1. Introduction

There is an increase in interest in educating software developers by means of an engineering approach. – the growth in undergraduate engineering programs for software attests to this.

However, studies of practitioner perspective, and of the software development process itself suggest this approach is flawed – it does not focus on the underlying characteristics of software development, nor adequately address the needs of practitioners.

Education for software developers needs to encompass more than the ability to apply knowledge gained, flexibility and creativity in the application of knowledge is also required. A competent practitioner not only knows the procedural steps for problem solving but also understands when to deploy them and why they work.

This paper notes the concerns raised in practitioner studies and proposes a change in the way software development may be learnt in an engineering environment in order to address these concerns.

2. Educating Software Engineers

Over 35 years ago, those involved in the development of software agreed that one mechanism for dealing with the intrinsic difficulties (eg complexity, visibility, and changeability [1]) of developing software was to embed its production within an applied science environment. Royce [2] 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. The underlying assumption of this approach is that the world works rationally and that therefore "good" software development is achieved by applying scientific investigative techniques [3].

This focus on engineering is mirrored in the education of software developers. Where Engineers Australia (Institution of Engineers, Australia) accredited two undergraduate programs for the engineering of software in the mid-1990s (Melbourne, Murdoch), by 2002 this figure approached 20. A similar trend is being seen in the US, with an exponential growth in offerings of undergraduate software engineering degrees.

Increasingly, therefore, 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 [4], so that learning may be viewed as a progression to expertise through task analysis, strategy selection, try-out and repetition [5].

3. The nature of software

Recent work argues that such an approach to learning software development should be regarded as flawed. Rather than being situated in a rational world, software is a collaborative invention: its development an exploratory and self-correcting dialogue [6].

In this alternative view the process of defining and designing a system is seen as one of insight-driven knowledge discovery [7] facilitated by opportunistic behaviour [8, 9]. Participants in the process must remain sensitive to progressive modifications [10] which lead not to a problem-solution, but to an 'evolved fit' acceptable to all stakeholders within the problem space.

The quintessential *creativity* of this process [11-14] is hampered by strict adherence to engineering and science methodologies. These:

- restrict essential characteristics such as opportunism [7]
- assist in adding accidental complexity through their attempts to control the RE's professional practice. (Sutcliffe and Maiden [15] suggest strict adherence to methods and procedures may restrict natural problem-solving) and
- impose a plan at odds to inherent cognitive planning mechanisms and hence interfering with the management of knowledge (Visser and Hoc [16] suggest that, in practice, a plan is followed only as long as it is cognitively cost-effective).

More broadly, 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 [17].

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 [18]).

As Waks [4] 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.

Therefore, although projects and work placements 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 [19]) 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.

4. Practitioner perspectives of SE education

In his *Point/Counterpoint* discussion, Bach stated that one reason software engineering is not more seriously studied is the common industry belief that most of the books and classes that teach it are impractical [20]. An overview of the studies undertaken to gain a practitioner perspective indicates that such an indictment is not too far from the mark.

Industry 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* [17].

Practitioner-based studies (eg [21-23] and in the Australian context [24-27]) assist us in building a profile of a practicing Software Engineer. They show us that, to paraphrase Fielden [28], in addition to traditional technical skills, software development professionals of the 21st century need to

- understand the learning process as a meta-skill and to develop flexibility in thinking
- have a deep understanding of self and others in complex human activity systems

- be adept in questioning underlying cultural, political, and intellectual assumptions
- be tolerant, compassionate, and at ease with multiple realities in complex systems
- value people as agents of change and technology as the tool
- value subjective involvement in technological areas
- allow time to explore new ideas and to reflect on possible processes and outcomes
- develop balanced approaches both structurally and creatively to managing change.

Model IT-focussed curricula address profession-specific knowledge and skills required to undertake professional graduate employment within the discipline. Initial competence (ie cognitive attributes à la Bloom taxonomy [29]), is also developed though perhaps not to an appropriate level: the curricula indicate that a graduate within the broad IT disciplines should emerge from formal education with a competency of *application* (or on occasion at a the lesser level of *comprehension*) [30-32].

5. Addressing practitioner needs

Macauley and Mylopoulos [33] acknowledge that a standard university lecture cannot achieve what industry requires. For them efficient software development activities "*require a certain level of knowledge and maturity which can only be gained through experience in dealing with practical problems"*. Others also note the inadequacy of formal education in training competent software professionals [23, 34].

The nature of software development, and in particular the RE component of it (opportunistic, exploratory, creative, emergent [7, 13, 35, 36]) implies a need to

- incorporate *creativity*-enhancing activities within the curriculum
- foster *adaptability* in students by providing for divergent as well as convergent thinking
- focus on metacognitive strategies and reflection as an aid to *transfer* of the skills and knowledge learnt.

Glass [37] suggests that discipline and creativity are the *odd couple* of software development – the discipline imposed by methodology, for example, forms a frame for the opportunistic creativity of design. Cropley and Cropley [38], however, suggest that the process of creativity and innovation in engineering is poorly understood and not adequately fostered in under- graduate teaching. This deficiency results in an engineering culture that is frequently resistant to the factors that promote creativity and innovation.

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 [39] 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.

The implication of this is the explicit development of metacognitive strategies, and the ability to reflect *in* as well as *on* action [40]. The recurring findings from Scott's work on applying a Professional Capability Framework (eg [41]) is the high ranking of *Intellectual Capability* (defined by two components, Way of Thinking (incorporating cognitive intelligence and creativity) and Diagnostic Maps (developed through reflection on experience)).

Turner [24] suggests tradition and inertia act as some of the formidable barriers to substantive revisions to curricula in line with the findings of practitioner-based studies. Yet providing a learning environment that enhances, for example, 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 (perhaps best exemplified in Requirements Engineering) engage in more creative problem solving over the longer term [42].

One avenue for incorporating the needs for flexibility, creativity and reflection in Software Engineering education is to address the pedagogical aspects rather than the content. The educational dilemma becomes one of providing an educational base that enables software developers to both create and engineer the systems they build: to be adaptable to the changing environment that is inevitable in their chosen discipline. One approach to addressing this dilemma is described in the following sections.

6. Educating Requirements Engineers

Education for Requirements Engineers based on traditional learning models tends to emphasise technical knowledge, and is based largely on notations and prescribed processes. Although [43] suggest this is a requirement of the software domain, it is at odds with the inherent characteristics associated with real problems, especially in requirements where [35]:

- complexity is added to rather than reduced with increased understanding of the initial problem
- metacognitive strategies are fundamental to the process
- problem-solving needs a rich background of knowledge and intuition to operate effectively
- a breadth of experience is necessary so that similarities and differences with past strategies are used to deal with new situations.

The School provides a number of degree programs focussing of the development of software. *Requirements Engineering* (ENG260) is the first of the core SE units, currently offered in semester 1 of the second year of study. During their first year students have been immersed in a scientific/engineering paradigm where problem-solving through laboratory procedure, repeatability of experimentation and rigour in mathematics are key learning objectives. ENG260 provides a contrast to this learning environment that some students find difficult to assimilate.

Although due process and procedure has its place, the focus of the unit is on divergent thinking and the development and evaluation of alternatives. In this unit they are asked to ignore the problem-**solving** (coding) of a situation presented (students come to the course with some competence in programming), and to explore and then formulate the problem itself. However, experience in teaching RE has shown that this is a challenge to students' expectations of learning:

- they expect there to exist a definitive solution to the problems with which they are presented (à la science/mathematics)
- they expect to define the problems only in terms of the programming language with which they are familiar (currently Java)
- they expect a fundamentally competitive class environment to exist
- they expect their 'wild ideas' to be laughed at and ultimately rejected, and therefore are inhibited in expressing them.

The approach taken, based on Problem-based Learning (PBL), is an attempt to provide students with a solid foundation in subject matter while at the same time exposing them to the real-world characteristics noted above. It provides students with a process to deal with problems within a metacognitive-rich framework that makes complexity apparent and lets students deal with it adaptively. The course material has been reworked for a PBL environment, and taught in this mode from February 2003.

6.1 Characteristics of PBL

As an ideology, Problem-based Learning is rooted in the experiential learning tradition, but with a number of different forms according to the nature of the field and goals of the learning situation [44]. Through its emphasis on problem and student-centredness, PBL is seen to:

- acknowledge the base of student experience
- emphasise student responsibility for learning
- cross boundaries between disciplines
- intertwine theory with practice
- focus on the *process* of knowledge acquisition rather than the products of that process
- change staff roles from instructor to facilitator
- focus on student self and peer assessment
- focus on communication and interpersonal skills so that students understand that to relate their knowledge, skills beyond their area of technical expertise are required.

It has been argued [45] that problem-based learning is an educational strategy that required three components to be differentiated:

- an integrated curriculum organised around real-world problems rather than disciplines and with an emphasis on cognitive skills
- small groups, tutorial instruction and active learning conditions to facilitate problem-based learning
- outcomes such as the development of skills and motivation together with the development of an ability to be lifelong learners.

Focussing on the solution of authentic problems as a context for learning also accords well with theories of expertise - learning beyond the initial stages may best be achieved through situational case studies with rich contextual information [46]. Its supporters claim PBL results in increased motivation for learning, better integration of knowledge across disciplines and greater commitment to continued professional learning [46]. As well as offering the flexibility to cater for a variety of learning styles, the focus moves from dealing with content and information in abstract ways to using information in ways that reflect how learners might use it in real life [47].

6.2 Enhancing creativity

Three components of Amabile's general theory of creativity:

- domain relevant skills - the more skills the better, and the ability to imagine/play out situations
- creativity-relevant processes - including breaking perceptual (the way you perceive a situation) and cognitive (the way you analyse) set and breaking out of performance 'scripts', suspending judgement, knowledge of heuristics, adopting a creativity inducing work style (eg tolerance for ambiguity, high degree of autonomy, independence of judgement). and
- intrinsic task motivation

are seem to influence positively creative potential. These were applied to the learning environment developed. A PBL process, as adapted by Koschmann et al [48] (see Table 1), used to anchor the student's learning.

Activities identified by Edmonds and Candy [49] as elements of creativity were embedded into the PBL environment. This Creative PBL model (Figure 1) was developed to focus on creativity and divergent thinking, so that, instead of students aimed at finding the single, best, "correct" answer to a standard problem in the shortest time (convergent thinking) they aimed at redefining or discovering problems and solving them by means of branching out, making unexpected associations, applying the known in unusual ways, or seeing unexpected implications.

Table 1 PBL Stages [48]

PBL Stage 1: *problem analysis* the rich context is mined for important facts, sub-problem(s) and alternate solution paths generated
PBL Stage 2: *self-directed learning* the learning agenda is determined by the information needed to evaluate the alternatives proposed
PBL Stage 3: *problem re-examination* based on findings, solution paths are added, deleted or revised
PBL Stage 4: *abstraction* an articulation process to increase the utility of the knowledge gained in specific contexts
PBL Stage 5: *reflection* a debriefing of the experience to identify improvement in the learning process.

This approach also had the value of addressing issues identified by Thomas et al [14]. They suggest

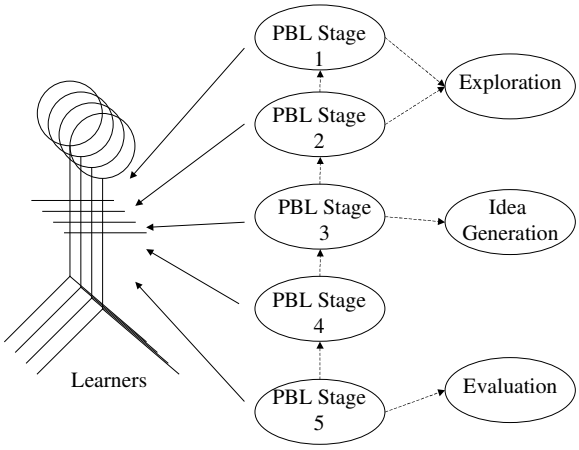

Figure 1. The Creative-PBL process

there is a widening gap between the degree of flexibility
and creativity needed to adapt to a changing world and the capacity to do so. These difficulties are attributed to:

- individuals or groups not engaging in effective and efficient processes of innovative design. As examples of *structuring failure*, people typically fail to spend sufficient time in the early stages of design: *problem finding and problem formulation*, then often bring critical judgment into play too early in the idea generation phase of problem solving. As another example, empirical evidence shows that people's behaviour is path-dependent and they are often unwilling to take what appears to be a step that undoes a previous action even if that step is actually necessary for a solution [50]
- evidence suggests individuals have a large amount of relevant *implicit knowledge* they often will not bring to bear on a problem. Providing appropriate strategies, knowledge sources or representations can significantly improve an individual's effectiveness in problem solving and innovation [50]
- the appropriate level, type, and directionality of motivation are not brought to bear [51].

7. How did we do?

7.1 Establishing a problem context

The PBL environment focuses on the secondment of the class to a (virtual) organisation – collaboration between a software house and the university. *MurSoft* requires a team to work, on short-term placement, on a project to develop gaming software to be used as an educational resource within a tertiary institute. This provides an authentic context for learning: students will have an opportunity, within their final year of study, to undertake an internship with a software-based organisation.

In order to ensure the team will integrate well, the students are initially provided with a very small problem to define. This problem introduces students to the *MurSoft* environment, and also serves the purpose of introducing the PBL process. Students are given some little time to familiarise themselves with other members of the class (since the rest of semester was to be spent on collaborative tasks) and with the lecturer, who takes on the role of academic consultant (not the client, but a resource students have access to). All interaction with the client is undertaken through web-based material: memos, minutes of meetings, telephone messages, 'talking heads', press releases etc provide the problem triggers required. Triggers act as prompts to students to undertake some task identified in the PBL process.

Unit content is centred on online teaching material and a recommended text, which act as a constraint: students initially explore this material in order to achieve the learning outcomes they have identified in a problem component, rather than having unlimited access to resources on the Internet and elsewhere. This is a significant issue: RE is a relatively new discipline, with varying approaches taken in its description. It is important at this early concept-learning stage that students are not confused or frustrated by the presentation of too many alternate viewpoints, tools, definitions for the same concept etc. This is likely to occur if students are to explore freely during the *self-directed learning* stage of the PBL process. On the other hand, it is important that students become aware that other views exist. Again, providing environment constraints adds to the authentic approach: as graduates, students will be expected to follow the operating procedures standardised within the employing organisation.

8. Evaluating the results

Both formal and informal assessment was undertaken over the semester: data may be categorised as:
quantitative assessment:

- the major assessment of the unit was based on group work (three components)
- the exam modelled previous exams, and was based on questions that had been used before, so in theory it was possible to compare how well students performed in comparison to previous cohorts
- two individual components (a Performance Review and a Portfolio) and

qualitative assessment:

- in-semester year surveys - the year co-ordinator asks for comments/problems regarding all the units undertaken over the semester. These surveys are conducted within the Engineering discipline in week 4 and 11
- students completed an end of semester unit assessment –this is University-based
- as noted above, one of the final components of their formal assessment was to prepare for a Performance Review. As well as some more technically based issues (eg how easy would it be to go to design from the specification developed by your team) students were asked for their impressions on their team performance and asked to comment on whether they thought they learnt less or more this way.

8.1. Quantitative assessment

The results achieved by these students will not be described here, except to note that, as shown in Figure 2 the PBL environment did not appear to unduly disadvantage the students.

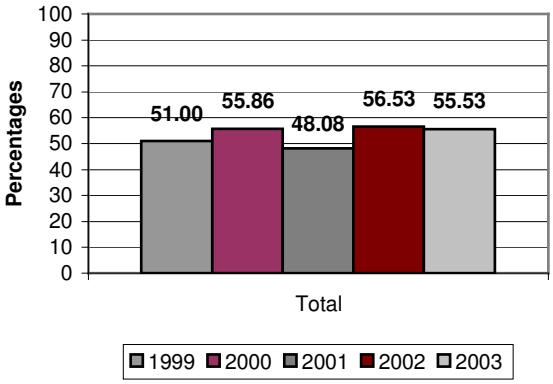

Figure 2a). Average raw exam mark ENG260

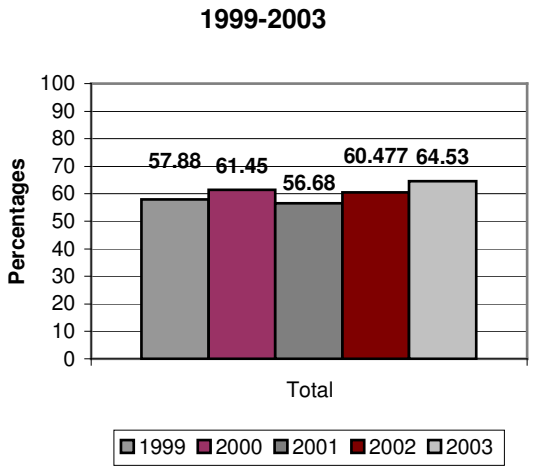

Figure 2b). Average final mark ENG260 – 1999-2003

8.2. Qualitative assessment

The Engineering discipline within the School informally surveys all students within each year group to identify general problems that are both unit-specific, and that relate to the mix of units undertaken. Students are asked to identify good and bad points during Weeks 4 and 11 of semester (ie usually near the first point of feedback and towards the end formal classes).

As the list of representative comments shows, some elements considered 'bad' by the students (eg *learning by doing*) are a highlight of the PBL process. This may be a reflection of student approach to study or preferred learning style, and deserves further investigation.

As can be seen from Figure 4, the class was fairly evenly divided on the point of learning more or less from this approach: comments on a lack of *mastery* of subjects: (less every time new content arrives); of

Week 4

Good:
–"helps with thinking about all areas of a problem(good for other units)"
–"interesting, practical, well presented"
–"it's really good"
Bad:
–"very vague on assessment and what specifically needs to be completed"
–"inability to work alone"
–"no lecture or tutorial"
–"don't really like how it's structured"
–"don't know what is going on"

Week 11

Good:
–"learn what you like at your own pace"
–"more practical training & real time example"
–"probably useful"
–"easy to get help for unit"
Bad:
–"objectives sometimes unclear"
–"learning by doing"
–"only get the general idea and concept of unit later in semester"
–"not very structured"
–"hard to determine what we are supposed to be working towards"

focusing on components addressed by the project, on *delegating* and *relying* on others for concepts indicate less content learning. Towards the end of semester, some of these students still felt lost and confused:

self teaching is not one of my fortés

stated one student, perhaps with a hint of despair.

Students felt they learnt more in the areas of *research*, *communications* (confidence to speak up; need to be heard & get ideas across) and *team skills*. They added *concepts easier to grasp; forced to learn more for project relevant components*; and, finally they had to grapple with various perspectives from others. In summary *there were: ample resources & up to us to take it*.

Other feedback also shows that, although a great deal of effort went into preparing the PBL environment, more scaffolding is required. Students need greater preparation in order to tackle a different learning model (eg a better understanding of the PBL process), and support structures (examples, guidelines)

so that they have a clear indication of the appropriateness of their learning.

8.3. Addressing the issues

The attributes of a Problem-based Learning classroom [44] provide a framework for future learning [52]. While acquiring specific domain knowledge is one of the unit objectives, adaptability and flexibility as a basis for insight and true novelty of thinking is equally important. The implication of this is effort spent on abstraction and reflection, well supported through the PBL process. Its supporters also argue that PBL best provides an effective environment for professionals who need to access diverse knowledge. In addition, the positive influences of an appropriate environment on the development of creative potential add support the adoption of PBL for RE education. The issues highlighted by Thomas et al are also addressed:

- the importance of *problem analysis*: in ENG260 this stage is a critical outcome. Problem-solving habit is challenged by the need to generate alternate solution paths, starting from the unknown and progressing to a description of the problem itself, and the knowledge needed to deal with it
- the value of alternative perspectives and prior knowledge is fostered through participation in a *collaborative* environment. Critical appraisal and self appraisal skills are developed through the use of reflection tools such as the 4SAT [53]
- although external motivation is difficult to eliminate within an undergraduate degree, PBL is seen to foster *intrinsic motivation* through the authenticity of the tasks undertaken [52]. Emphasis on elements that foster external motivation (such as exams) is gradually being reduced as less appropriate to this style of learning. This is an important point.

9 Approaches to study

As Elton [54] states: "*we want students to learn with understanding and be assessed for it*". A post-hoc *Approaches to Study Inventory* (using a 32-item instrument confirmed by Richardson [55]'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). Figure 3 is a graphic representation of these results. Figure 4 confirms this: it represents the student response to the question of whether they felt they learnt more or less

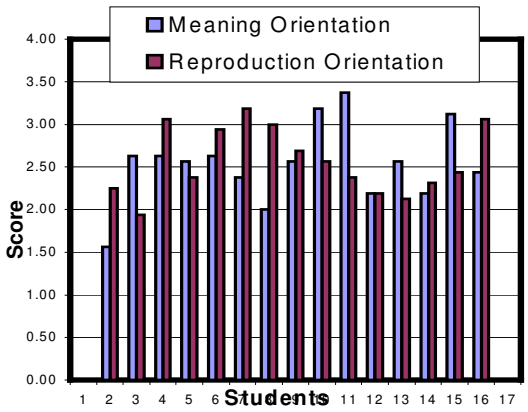

Figure 3. Approaches to Study survey – 2003 student cohort taking ENG260

using this approach. This was one of several reflective comments embedded in the final assessment described above.

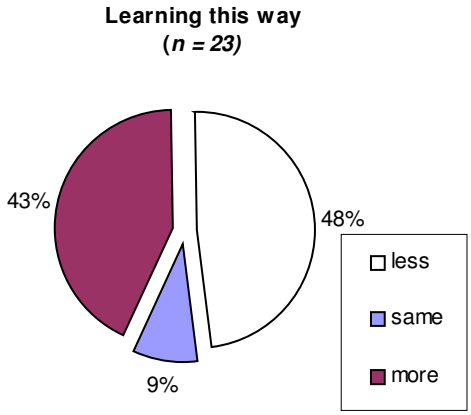

Figure 4. Student perception of learning in ENG260 - 2003 cohort

While this result is of some concern, it should be noted, however, that 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 [54]. However, this is an area that needs to be explored further.

10. The future

The PBL environment these students have experienced may be considered a creative one: one of the aims of its development has been to enhance divergent thinking and the creative potential of students. It would seem, however, that such an environment may not match the learning characteristics

of the student cohort. Tracking the cohort through subsequent units will go some way to confirming (or not) the value of PBL in Software Engineering education. This is critical in the context of a strategic move away from traditional lecture/tutorial/lab-style learning within the discipline area at this University. Research into student approaches to study provides some insight that will assist in further offerings of this unit and of others within the engineering degree programs.

However, to end on a positive note, some members of this student cohort have progressed to subsequent units. These (*Advanced Software Design I and II*) are taught following a Design Studio model. Although it is too early in semester to undertake any evaluation of their learning, a comment overheard during a workshop session is promising. One group of three students was reporting (to each other) on their progress in constructing an Object-Z specification. One student remarked that he found he could just follow the template. But, he said,

> *that seemed like cheating so I had to go back to the notes and work out how to do it properly*

Of even greater interest, other members of his group concurred.

10. References

[1] F. P. Brooks, "No silver bullet - essence and accidents of software engineering," presented at Proceedings of Information Processing 86: the IFIP 10th World Conference, Amsterdam, 1986.

[2] W. W. Royce, "Managing the development of large software systems: concepts and techniques," presented at IEEE WESCON, 1970.

[3] S. L. Pfleeger, "Albert Einstein and empirical software engineering," *IEEE Computer*, vol. 32, pp. 32-37, 1999.

[4] L. J. Waks, "Donald Schon's Philosophy of Design and Design Education," *International Journal of Technology and Design Education*, vol. 11, pp. 37-51, 2001.

[5] W. Winn and D. Snyder, "Cognitive perspectives in psychology," in *Handbook of Research for Educational Communications and Technology*, D. H. Jonassen, Ed. New York: Simon & Schuster Macmillan, 1996, pp. 112-142.

[6] J. Bach, "Reframing requirements analysis," *IEEE Computer*, vol. 32, pp. 120-122, 1999.

[7] R. Guindon, "The process of knowledge discovery in system design," in *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, G. Salvendy and M. J. Smith, Eds. Amsterdam: Elsevier, 1989, pp. 727-734.

[8] R. Guindon, " Knowledge exploited by experts during software systems design," *International Journal of Man-Machine Studies*, vol. 33, pp. 279-304, 1990.

[9] W. Visser, "Designers' activities examined at three levels: organisation strategies and problem-solving processes," *Knowledge-Based Systems*, vol. 5, pp. 92-104, 1992.

[10] J. P. v. Gigch, "Metamodelling and problem solving," *Journal of Applied Systems Studies*, vol. 1, pp. 327-336, 2000.

[11] M. Lubars, C. Potts, and C. Richer, "A review of the state of the practice in requirements modeling," presented at International Symposium on Requirements Engineering, San Diego, 1993.

[12] N. A. M. Maiden and A. G. Sutcliffe, "Exploiting reusable specifications through analogy," *Communications of the ACM*, vol. 34, pp. 55-64, 1992.

[13] N. Maiden and A. Gizikis, "Where do requirements come from?," *IEEE Software*, vol. 18, pp. 10-12, 2001.

[14] J. C. Thomas, A. Lee, and C. Danis, "Enhancing creative design via software tools," *Communications of the ACM*, vol. 45, pp. 112-115, 2002.

[15] A. G. Sutcliffe and N. A. M. Maiden, "Analysing the novice analyst: cognitive models in software engineering," *International Journal of Man-Machine Studies*, vol. 36, pp. 719-740, 1992.

[16] W. Visser and J. Hoc, "Expert software design strategies," in *Psychology of Programming*, J. M. Hoc, T. R. G. Green, S. R, and G. D. J, Eds. San Diego (CA): Academic Press, 1990, pp. 235-247.

[17] D. Garlan, D. P. Gluch, and J. E. Tomayko, "Agents of Change: Educating Future Leaders

in Software Engineering," *IEEE Computer*, vol. 30, pp. 59-65, 1997.

[18] M. Savin-Baden, *Problem-based Learning in Higher Education: untold stories*. Buckingham (UK): Society for Research into Higher Education and Open University Press, 2000.

[19] G. Kearsley, "Explorations in Learning & Instruction: the theory in practice database," George Washington University, Washington (DC) 2000.

[20] J. Bach, "SE education: we're on our own," *IEEE Software*, vol. 14, pp. 26,28, 1997.

[21] E. M. Trauth, D. Farwell, and D. M. S. Lee, "The IS expectation gap: industry expectation versus academic preparation," *MIS Quarterly*, vol. 17, pp. 293-307, 1993.

[22] D. M. S. Lee, "Organizational entry and transition from academic study: examining a critical step in the professional development of young IS workers," in *Strategies for Managing IS/IT Personnel*, M. Igbaria and C. Shayo, Eds. Hershey (PA): Idea Group, 2004, pp. 113-141.

[23] T. C. Lethbridge, "What knowledge is important to a software professional?," *IEEE Computer*, vol. 33, pp. 44-50, 2000.

[24] R. Turner and G. Lowry, "Education for a technology-based profession: softening the Information Systems curriculum," in *Current Issues in IT Education*, T. McGill, Ed. Hershey (PA): IRM Press, 2003, pp. 153-172.

[25] R. Snoke and A. Underwood, "Generic attributes of IS graduates - a Queensland study," presented at Proceedings of the 10th Australasian Conference on Information Systems, Wellington (NZ), 1999.

[26] G. Scott and W. Yates, "Using successful graduates to improve the quality of undergraduate engineering programs," *European Journal of Engineering Education*, vol. 27, pp. 60-67, 2002.

[27] O. Minor, "Theory and Practice in Requirements Engineering: an investigation of curricula and industry needs." Koblenz (Germany): University of Koblenz-Landau, 2004.

[28] K. Fielden, "Training Information Systems professionals to balance at the edge of chaos in a technical world," presented at Proceedings of IFIP Working groups 8.2 and 8.6 joint Working Conference on Information Systems: Current Issues and Future Changes, Helsinki, 1998.

[29] B. S. Bloom, *Taxonomy of Educational Objectives: the classification of educational goals Handbook 1: cognitive domain*. New York: David Mackay, 1956.

[30] A. Sobel, "CC-SE: Computing Curricula -- Software Engineering public draft II," Joint Task Force on Computing Curricula, ACM and IEEE Computer Society, 2003.

[31] G. Engel, "Computing Curricula 2001: Computer Science - final report," Joint Task Force on Computing Curricula, ACM and IEEE Computer Society, 2001.

[32] J. T. Gorgone, G. B. Davis, J. S. Valacich, H. Topi, D. L. Feinstein, and H. E. Longenecker, "IS 2002: model curriculum for undergraduate degree programs in Information Systems." Park Ridge (IL): ACM, 2002.

[33] L. Macauley and J. Mylopoulos, "Requirements Engineering: an educational dilemma," *Automated Software Engineering*, vol. 4, pp. 343-351, 1995.

[34] P. N. Robillard, "The role of knowledge in software development," *Communications of the ACM*, vol. 42, pp. 87-92, 1999.

[35] J. Bubenko, "Challenges in Requirements Engineering: keynote address," presented at RE'95: Second IEEE International Symposium on Requirements Engineering, York (UK), 1995.

[36] L. Nguyen and P. A. Swatman, "Complementary Use of ad hoc and post hoc Design Rationale for Creating and Organising Process Knowledge," presented at Proceedings of the Hawaii International Conference on System Sciences HICSS-33, Maui (Hawaii), 2000.

[37] R. L. Glass, *Software Creativity*. Englewood Cliffs (NJ): Prentice-Hall, 1995.

[38] D. H. Cropley and A. J. Cropley, "Teaching Engineering Students to be Creative - Program and Outcomes," presented at Australasian Association o f Engineering Education: 10th Annual Conference, 1998.

[39] S. P. Gott, E. P. Hall, R. A. Pokorny, E. Dibble, and R. Glaser, "A naturalistic study of transfer: adaptive expertise in technical domains," in *Transfer on Trial: intelligence, cognition and instruction*, D. K. Detterman and R. J. Sternberg, Eds. Norwood (NJ): Ablex, 1993, pp. 258-288.

[40] D. A. Schön, *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books, 1983.
[41] G. Scott and D. Wilson, "Tracking and profiling successful IT graduates: an exploratory study," presented at Proceedings of the 13th Australasian Conference on Information Systems, 2002.
[42] J. M. Baer, "Long term effects of creativity training with middle-school students," *Journal of Adolescence*, vol. 8, pp. 183-193, 1988.
[43] D. Budgen, *Software Design*. Harlow (Essex): Pearson Education Ltd, 2003.
[44] D. Boud, "Problem-based learning in perspective," in *Problem-based Learning in Education for the Professions*, D. Boud, Ed. Sydney: Higher Education Research Society of Australasia, 1985, pp. 13-18.
[45] H. J. Walton and M. B. Mathews, "Essentials of problem-based learning," *Medical Education*, vol. 23, pp. 542-548, 1989.
[46] H. L. Dreyfus and S. E. Dreyfus, *Mind over Machine*. New York: Free Press, 1986.
[47] R. Oliver and C. McLoughlin, "Using web and problem-based learning environments to support the development of key skills," presented at Responding to Diversity: Proceedings of ASCILITE '99, Brisbane, 1999.
[48] T. D. Koschmann, A. C. Myers, H. S. Barrows, and P. J. Feltovich, "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*, vol. 3, pp. 227-264, 1994.
[49] E. Edmonds and L. Candy, "Creativity, art practice and knowledge," *Communications of the ACM*, vol. 45, pp. 91-95, 2002.
[50] J. C. Thomas, D. Lyon, and L. Miller, "Aids for Problem Solving," IBM T. J. Watson Research Report, New York RC-6468, 1977.
[51] T. M. Amabile, *The Social Psychology of Creativity*. New York: Springer Verlag, 1983.
[52] B. G. Wilson and P. Cole, "Cognitive teaching models," in *Handbook of Research for Educational Communications and Technology*, D. H. Jonassen, Ed. New York: Simon & Schuster Macmillan, 1996, pp. 601-621.
[53] C. Zimitat and H. Alexander, "The 4 Step Assessment Task (4SAT)," *Integrity, Innovation and Integration*, vol. 4, pp. 692-697, 1999.
[54] L. Elton, "Matching teaching methods to learning processes: dangers of doing the wrong thing righter," presented at 2nd Annual Conference of the Learning in Law Initiative, 2000.
[55] J. T. E. Richardson, "Reliability and replicability of the Approaches to Studying questionnaire," *Studies in Higher Education*, vol. 15, pp. 155-168, 1990.