Educating 'Agents of Change'

Jocelyn Armarego
Murdoch University Australia
jocelyn@eng.murdoch.edu.au

Abstract

This paper describes the journey undertaken by staff and students in an undergraduate SE program. It charts the progress to a learning environment that fosters reflective practice through a Problem-based (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. SE staff benefit from the double-loop approach: the espoused theory of teaching is aligned with the theory in practice as innovation is introduced, evaluated then reflected on to initiate further development in the learning environment. The wisdom of our approach will be tested by the success of our graduates in engaging with the profession in practice.

1. Introduction

Murdoch University 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 our graduates to find career opportunities in both professional engineering industries that have a strong interest in software as well as in 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. Our School has focused over the past several years on balancing a foundation in the content with elements of creativity and experience based on practice, as we address characteristics specific to SE. This paper describes the journey undertaken by both staff and students in the pursuit of this balance within an engineering educational context.

2. Issues in SE education

Software development has been described as a 'craft'. The negative connotations of this label include an inability to consistently guarantee a quality product, fit for the purpose for which it was developed, produced on time and within budget. A mid-1990s study of over 8,000 projects [1] reported only 16.2% of software was successful. These rates do not significantly differ from those reported in the 1970s and 1980s [2].

2.1. Engineering software

Those involved in the development of software agreed that one mechanism for dealing with the intrinsic difficulties (eg complexity, visibility, and changeability [3]) of developing successful software was to embed its production within an applied science environment. Royce [4] was the first to note explicitly that an engineering

Proceedings of the 18th Conference on Software Engineering Education & Training (CSEET'05)
1093-0175/05 $ 20.00 **IEEE**

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 "good" software development is achieved by applying scientific investigative techniques [5].

This interest in engineering is mirrored in the education of software developers, with an exponential growth in offerings of undergraduate software engineering degrees. Increasingly, this education focuses on process and repeatability, modelling scientific and engineering methodologies. In general it is based on a normative professional education curriculum, in which students first study basic science, then the relevant applied science [6], so that learning may be viewed as a progression to expertise through task analysis, strategy selection, try-out and repetition [7]. As Waks [6] explains, in this normative model 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.

2.2. Creating software

There are positive implications as well for the label 'craft'. Each system is considered a unique synergy between the hardware, software and organisational context in which it will be utilised. This approach suggests that the development process cannot be repeatable, as the forces at play will differ for each context, continually changing as understanding of the characteristics of the developing system grows in all stakeholders. From this perspective software is a collaborative invention: its development an exploratory and self-correcting dialogue [8]. The process of defining and designing a system is seen as one of insight-driven knowledge discovery [9] facilitated by opportunistic behaviour [10, 11].

The quintessential *creativity* of this process [12-15] may be hampered by strict adherence to engineering and science methodologies. These:

- ? restrict essential characteristics such as opportunism [9]
- ? assist in adding accidental complexity through their attempts to control the RE's professional practice. (Sutcliffe and Maiden [16] 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 (Hoc [17] suggest that, in practice, a plan is followed only as long as it is cognitively cost-effective).

2.3. Practicing software

Another issue is the multi-disciplinary nature of the skills and knowledge required to be active as competent professionals. In SE, Zucconi [18] suggests, the 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 also suggests 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.

This equates well with the stated needs of practitioners. Practitioner-based studies (eg [19-21] and in the Australian context [22-25]) assist us in building a profile of a

practicing Software Engineer. They show us that, to paraphrase Fielden [26], 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.

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. 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 [27].

2.4. Teaching software

This relates to a further issue that needs to be addressed: the 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. Professional practitioners with such skills become *agents of change* [27].

However, Patel, Kinshuk, and Russell [28] argue that learners in a traditional setting predominantly constitute students preparing for a career – classroom-based students. With the relevance of domain knowledge not fully understood by students, the focus shifts to skills that will yield higher grades as an immediate objective. 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 fulfill the assessment tasks. This defeats the objective of providing a well-balanced learning experience.

3. Learning software engineering

Macauley and Mylopoulos [29] 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 [21, 30], while 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 [31].

A need to address these issues:

- an increasing focus on a single discipline (engineering) as the education of choice for software (hence with an emphasis on scientific generalisation)
- the potential for misalignment with industry needs
- an acknowledged need for life-long learning

within the constraints of an accredited curriculum has led our School on a journey of exploration of the pedagogy of learning.

3.1. Phase 1: engaging with authentic practice

The current curriculum for the BE(SE) integrates three primary components:

- *Computer Science* – these courses cover fundamental aspects (eg programming, algorithm analysis, database and operating system concepts) and form the basis of technical knowledge and skills in software and hardware
- *Software Engineering* – these courses focus on SE theory and practice and form the basis of core knowledge and skill in software development and evolution
- *Engineering* – these courses offer knowledge and skills in engineering practice and principles and are common to all our engineering students. They include natural sciences, mathematics, management, ethics

to provide the basis for:

- *Engineering Internship/ Thesis* – this is also common to all engineering students, though the domain of application targets the appropriate discipline of study. While the *Internship* is wholly industry-based in that the student is an 'employee' of the organisation, it is carefully monitored by academic staff. The *Thesis* may also be linked to work place experiences, but the student is not employed during the duration of the project.

The reduced opportunity for group-based projects due to the introduction of the semester-long internship/ thesis was one trigger for the restructure of some of the core SE units. Other triggers included:

- students in their final year of studies had raised issues regarding the need to apply knowledge acquired to more 'real world' scenarios – and to engage with the material on offer
- a need to provide students with a taste of the types of *messy* problems they would encounter during their internship. Exposure to the uncertainties, inconsistencies and idiosyncrasies associated with real problems 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 [32].

As described previously [33] a Design Studio model was applied, anchored by a PBL process based on Koschman et al [34] and in the context of the phased development of a software product. Student evaluation undertaken in weeks 4, 7 11 and 13 highlighted student concerns:

- the need to learn new content as well as adapt previous knowledge
- dependence on other members of the team (10+ 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

but also identified benefits:

- ? *"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 ([this course's] style)"*.

The restructured course was seen to provide students with a number of opportunities [33]:

- ? 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.

However, an unexpected problem was encountered early in the semester. While students were comfortable with the idea of directing their own learning in a capstone-project and thesis environment, they felt (very strongly, at times) that within a formal course they should be *taught*. This perception could be traced back to a reasonably high level of teacher direction in prerequisite SE units, which did not challenge students' expectations of traditional learning. The initial student resistance to changes made showed that these expectations were still evident two years later in their studies.

3.2. Phase 2: creative engineering

Reflection on the learning experience highlighted a need to emphasise student-centred learning earlier – the final year was too late. This led to a review of pre-requisite courses, with a view to making pedagogical changes early in the SE curriculum. Opportunities to focus in greater depth on issues raised in the discussion of SE education were also identified. One additional issue could also be addressed: SE education has historically been plagued by a lack of integration - the methods, techniques, tools, etc acquired within a few isolated courses do not permeate the students' approach to other software-related tasks within their program of study. An attempt could now be made to present a more holistic approach to SE education within the School.

This second wave focused on the first SE course students encounter, currently in semester 1 of Year 2 of the 4-year degree program. The core component – Requirements Engineering (RE) - provided an appropriate environment for attacking student expectations of a learning environment.

Education for REs based on traditional learning models tends to emphasise technical knowledge, and is based largely on notations and prescribed processes [35]. Although Budgen [36] suggests this is a requirement of the software domain, it is at odds with the inherent characteristics associated with real problems, especially in RE where [37]:

- ? 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 nature of software development, and in particular the RE component of it (opportunistic, exploratory, creative, emergent [9, 14, 37, 38]) 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 [39] 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 [40], 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 [41] 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 [42]. The recurring findings from Scott's work on applying a Professional Capability Framework (eg [43]) 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)).

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.

3.2.1. Attacking expectations: 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. The RE course (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 course is on divergent thinking and the development and evaluation of alternatives. Students are asked to ignore the problem-solving (coding) of a situation presented 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 (so that co-operation and collusion have blurred definitions)
- they expect their 'wild ideas' to be laughed at and ultimately rejected, and therefore are inhibited in expressing them.

The Creative Problem-based Learning (*CreatePBL*) approach taken is a refinement of that implemented in Phase 1 above. In an attempt to provide students with a solid foundation in introductory concepts in SE/RE while at the same time exposing them to the real-world characteristics, there is a greater need to ground student learning in the pedagogy.

As an ideology, PBL 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]. 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.

Its supporters claim PBL results in increased motivation for learning, better integration of knowledge across disciplines and greater commitment to continued professional learning [32]. 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 [46].

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.

Activities identified by Edmonds and Candy [47] as elements of creativity were embedded into the PBL environment. This *CreatePBL* 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.

This approach also had the value of addressing issues identified by Thomas et al [15]. They suggest 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 the

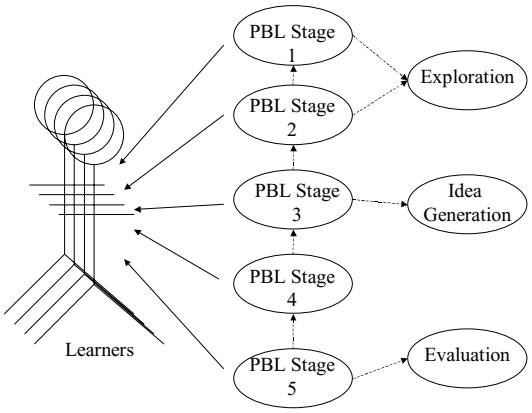

Figure 1. The *CreatePBL* process

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 [48]

? 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 [48]

? the appropriate level, type, and directionality of motivation are not brought to bear [49].

3.3. 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 - these are conducted within the Engineering discipline in week 4 and 11
  - students completed an end of semester course 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.

3.3.1. Quantitative assessment: The results achieved by these students will not be described here: there are many factors to be addressed, including characteristics of the cohort. However, it can be noted that the PBL environment did not appear to unduly disadvantage students: average raw exam and final marks were at worse within the range established by previous cohorts of students (see Figure 2).

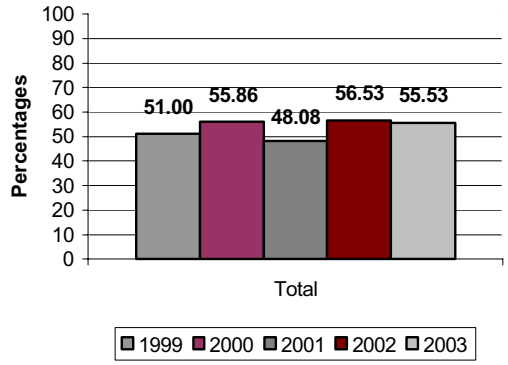

Figure 2a). Average raw exam mark 1999 – 2003

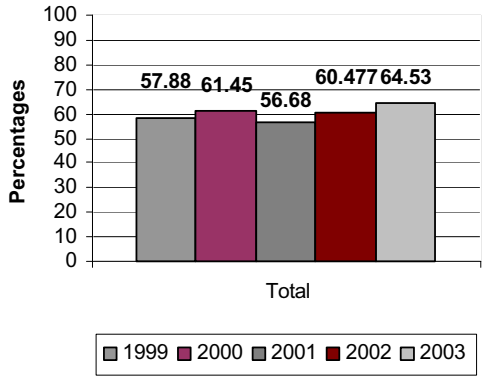

Figure 2b). Average final mark 1999-2003

3.3.2. Qualitative assessment: Within Engineering we survey all students in each year group to identify general problems that are both course-specific and that relate to the mix of courses undertaken. During Weeks 4 and 11 of semester (ie usually near the first point of feedback and towards the end formal classes) students are asked to identify good and bad points of each course. As the lists of representative comments below show, 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 has been the subject of further investigation.

As a part of the "Performance Review" component of the coursework students were asked to comment (anonymously, if preferred – only submission/no submission was recorded as an assessment object) on the course by addressing the following questions:

? *what is good/bad about a unit structured this way*. Figure 3 shows the results. While some students could appreciate the authentic nature of the environment, this learning style did not sit well with all

? *things to add/change/delete in how the virtual secondment and project were organised*. Figure 4 shows that students felt they lacked guidance in the form of examples and exercises to use for benchmarking their performance. In addition, they expressed concern that such a large component of their assessment was based on group work.

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"

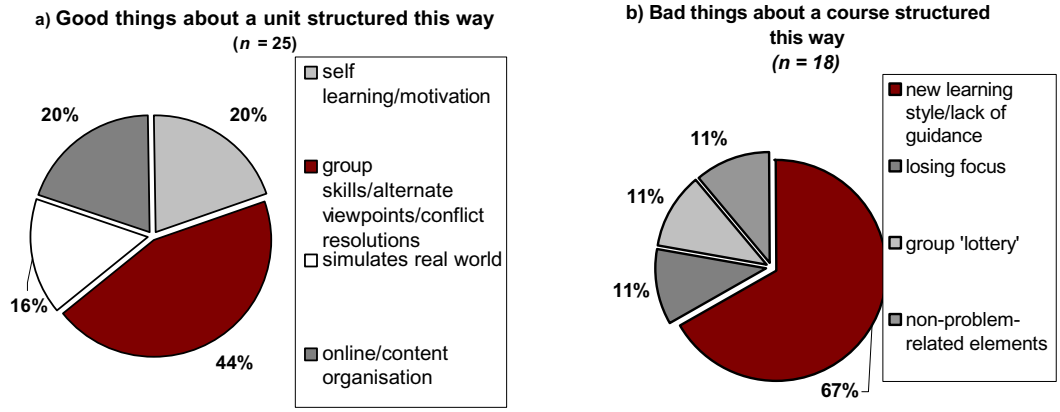

Figure 3 Recoded comments from Performance Review: good/bad things about a course structured this way

Figure 5 shows that 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 only *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 fortes*" 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"*.

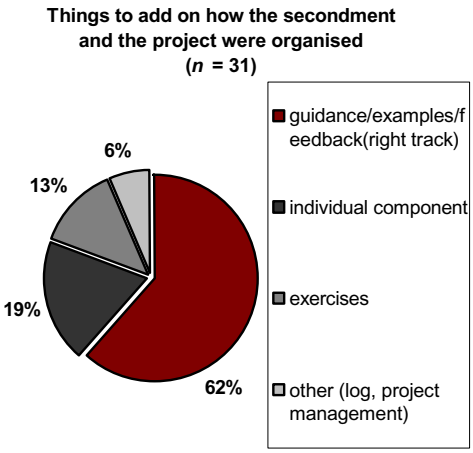

Figure 4. Recoded comments from Performance Review: things to add

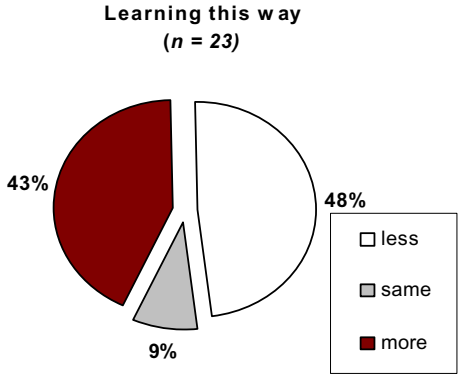

Figure 5. Student perception of learning in ENG260 - 2003 cohort

Other feedback re-enforced the conclusion 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.

3.3. Phase 3: reflecting on practice

While these interventions have had some measure of success, at least in terms of learning outcomes, one issue to be addressed is that innovation introduced a few courses may be undermined if traditional approaches are maintained elsewhere in the students' program – 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*"[42]. 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.

According to 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'* [50].

4. Conclusion

As we learn more about *how* students learn, and *what* they need to learn in order to practice as competent professionals in their chosen discipline, we move further from traditional teaching and closer to the concept of learning as a reflection on professional practice undertaken by both teachers and learners.

This view of professional education has implications for the design of teaching [51]:

? 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.

The progression to Problem-based Design Studios in SE is a journey being undertaken by academics of this University. In empowering graduates to be industry-ready SE staff benefit from a 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 [52]. 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 [53] that integrates class and work experience. Preliminary results from a longitudinal study of students involved in this approach suggest they integrate into their internship more easily.

However, the future will test the long-term wisdom of our approach.

5. References

[1] Standish, "Most Programming Projects Are Late," Standish Group, West Yarmouth (MA) 1995.

[2] J. Mann, "The Role of Project Escalation in Explaining Runaway Information Systems Development Projects: A Field Study," Georgia State University, 1996.

[3] 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.

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

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

[6] 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.

[7] 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.

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

[9] 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.

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

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

[12] 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.

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

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

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

[16] 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.
[17] 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.
[18] L. Zucconi, "Essential knowledge for the practicing Software Engineer and the responsibilities of university and industry in her education," presented at Software engineering education: 8th SEI Conference on software engineering education, New Orleans, 1995.
[19] 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.
[20] 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.
[21] T. C. Lethbridge, "What knowledge is important to a software professional?," *IEEE Computer*, vol. 33, pp. 44-50, 2000.
[22] 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.
[23] 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.
[24] 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.
[25] O. Minor, "Theory and Practice in Requirements Engineering: an investigation of curricula and industry needs." Koblenz (Germany): University of Koblenz-Landau, 2004.
[26] 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.
[27] 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.
[28] A. Patel, Kinshuk, and D. Russell, "Intelligent tutoring tools for cognitive skill acquisition in life long learning," *Educational Technology & Society*, vol. 3, pp. 32-40, 2000.
[29] L. Macauley and J. Mylopoulos, "Requirements Engineering: an educational dilemma," *Automated Software Engineering*, vol. 4, pp. 343-351, 1995.
[30] P. N. Robillard, "The role of knowledge in software development," *Communications of the ACM*, vol. 42, pp. 87-92, 1999.
[31] J. Bach, "SE education: we're on our own," *IEEE Software*, vol. 14, pp. 26,28, 1997.
[32] H. L. Dreyfus and S. E. Dreyfus, *Mind over Machine*. New York: Free Press, 1986.
[33] J. Armarego, "Advanced Software Design: a case in problem-based learning," presented at CSEET2002 15th Conference on Software Engineering Education and Training, Covington (Ke), 2002.
[34] 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.
[35] L. Nguyen and P. A. Swatman, "Essential and incidental complexity in requirements models," presented at Fourth International Conference on Requirements Engineering Education, Schaumburg (Il), 2000.
[36] D. Budgen, *Software Design*. Harlow (Essex): Pearson Education Ltd, 2003.
[37] J. Bubenko, "Challenges in Requirements Engineering: keynote address," presented at RE'95: Second IEEE International Symposium on Requirements Engineering, York (UK), 1995.
[38] 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.
[39] R. L. Glass, *Software Creativity*. Englewood Cliffs (NJ): Prentice-Hall, 1995.
[40] D. H. Cropley and A. J. Cropley, "Teaching Engineering Students to be Creative - Program and Outcomes," presented at Australasian Association of Engineering Education: 10th Annual Conference, 1998.
[41] 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.
[42] D. A. Schön, *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books, 1983.
[43] 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.

[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] 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.
[47] E. Edmonds and L. Candy, "Creativity, art practice and knowledge," *Communications of the ACM*, vol. 45, pp. 91-95, 2002.
[48] J. C. Thomas, D. Lyon, and L. Miller, "Aids for Problem Solving," IBM T. J. Watson Research Report, New York RC-6468, 1977.
[49] T. M. Amabile, *The Social Psychology of Creativity*. New York: Springer Verlag, 1983.
[50] D. A. Schön, *Educating the Reflective Practitioner: Towards a New Design for Teaching in the Professions*. San Fransisco: Jossey-Bass Inc, 1987.
[51] D. Laurillard, *Rethinking University Teaching: a framework for the effective use of educational technology*. London: Routledge, 1993.
[52] C. Argyris and D. A. Schön, *Theory in practice: Increasing professional effectiveness*. San Francisco: Jossey Bass, 1974.
[53] C. Berkenkotter and T. N. Huckin, *Genre Knowledge in Disciplinary Communication: Cognition/Culture/Power*. Hillsdale (NJ): Lawrence Erlbaum Associates, 1995.