Constructing Engineering Knowledge: development of an online learning environment

Jocelyn Armarego, Lynne Fowler, Geoffrey G Roy
School of Engineering, Murdoch University
{jocelyn, lynne, geoff}@eng.murdoch.edu.au

Abstract

With the development of new undergraduate degree programmes within Murdoch University's School of Engineering, the decision was made to offer courses, as much as practical, online. This provides numerous challenges to be addressed, including considerations of curriculum design and learning issues. Within the Software Engineering program, an infrastructure has been developed to address these issues and to enable students to exercise a measure of control over their learning experiences.

1. Introduction

One of the challenges being addressed within education, and higher education in particular, is that of providing students with life-long learning skills. The speed with which technology evolves, the multiplicity of its impact on society and the ramifications of that impact mean that more than technical competence with specific tools and techniques is necessary. Currently accepted models of learning, based on the constructivist approach, suggest that where learners constructs personal meaning, by engaging in

dialogue – internally or with others, in order to obtain consensus, and
reflection – multiple perspectives and challenges provide opportunity for reflection and introspection in order to make sense of experience gained [16],

they develop the skills to build their own knowledge, and hence take control of their own learning.

This paper describes one approach, within the School of Engineering at Murdoch University, which attempts to foster learner-centred knowledge construction. The Web is seen as a medium that supports student control of the learning process [17] and is said to be well suited to domains of conceptual complexity and case-to-case irregularity [2]. Many areas of Engineering (and in particular Software Engineering) fit this category of material. However, the decision to offer courses online as the default means that issues related specifically to education incorporating the Web have to be addressed. These are discussed in relation to the online environments developed within the School.

2. Context – the Murdoch environment

Murdoch University School of Engineering provides a suite of programs in Software Engineering, from a 4-year undergraduate degree in Engineering, through post-graduate and masters programs to PhD in Software Engineering. The discussions in this document refer in general to the undergraduate degree.

Murdoch's Bachelor of Engineering (Software Engineering) (BE(SE)) was the second

0-7695-1059-0/01 $10.00 © 2001 IEEE

program in Australia to receive full accreditation from the Institution of Engineers, Australia (IEAust). This means that our graduands are fully accredited professional engineers, with all that implies. From a curriculum point of view, this means that the program must conform to IEAust's requirements as well as those of the Australian Computer Society (which has also accredited the program). If viewed from a United States perspective, this equates to accreditation by both ABET and CSAB.

Our teaching objectives are focused on producing graduate professional engineers with a special skill in Software Engineering. We will expect our graduates to find career opportunities in both traditional 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.

3. Curriculum design considerations

The development of our BE(SE) programme preceded the publication of the SWEBOK [6], but in review we are reasonably well satisfied that our course conforms with these proposals. It is also closely aligned with the recently published sample curricula as proposed by the Working Group Software Engineering Education and Training (WGSEET) [4] (which can also be found at http://faculty.db.erau.edu/hilburn/se-educ).

One of the issues that have plagued Software Engineering education has historically been that of integration – that the methods, techniques, tools etc acquired within a few isolated SE courses do not permeate the students' approach to other software related tasks within their program of study. This is addressed to some extent through the disciplined framework to "design-build-deliver" artefacts that is at the core of Engineering education. Initiation of the SE programs in Engineering rather than attempting to migrate from a Computer Science framework provided an accelerated rate of change towards integration.

3.1. Curriculum components

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

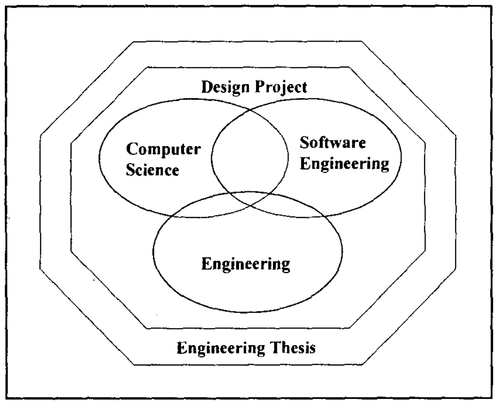

Figure 1 BE(SE) Curriculum components

The three primary components:

- ➢ *Computer Science* – these courses cover fundamental aspects of the discipline (eg programming, algorithm analysis, database and operating system concepts) and for the basis of technical knowledge and skills in software and hardware
- ➢ *Software Engineering* – these courses focus on SE theory and practice (eg, requirements, user interface, management, metrics and maintenance, organisational issues) and form the basis of core knowledge and skill in software development and evolution. Assessment in some of these courses focuses on project-based teamwork
- ➢ *Engineering* – these courses offer knowledge and skills in engineering practice and principles and include those elements of IEAust's curriculum requirements not covered in the previous components. These are common to all Engineering students within our School (eg 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. As of 2001, the proposal is to form multi-disciplinary teams of students within the *Design Project* at least. While the project may be industry-based, it is run under controlled conditions, and carefully monitored by academic staff. The *Thesis*, on the other hand focuses on industry and is usually linked to work-place experiences: the student spends 25% of the penultimate semester, rising to 50% in the last. Supervision is joint academic/industry, with the student required to complete and present a thesis based on the project.

Underlying these is a common set of support material and resources, including CASE 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.

Thus, in terms of integration, while the CS component provides the basic elements that act as foundation on which software is developed, the core of the program is engineering (both software and general) theory and practice.

3.2. Design considerations

The decision was made to offer courses, as much as practical, online. The rationale behind this has been published informally [20]. In summary the driving objectives have been:

- ➢ to develop efficient means of delivery for both on and off-campus students. From the outset the intention has been to provide the courses via distance education – this being a significant element of teaching for this University
- ➢ to provide a means of documenting a complete curriculum that is not fully dependent on individual staff interests and capabilities. From experience, many university courses are very dependent on individual staff: the "collective memory" is often limited, and replacing a lecturer may often imply re-writing courses previously taught.

Developing Web-based curricula offers a reasonable approach to achieving both of these objectives. This development is proceeding as fast the available resources will allow, and at the moment we have most of the core SE and CS courses up and running.

4. Learning considerations

The decision to focus on online learning required the consideration of several issues related specifically to education incorporating the Internet. These are more to do with teaching and learning paradigms rather than the actual content.

While the Web is seen as a medium that supports student control of the learning process, some educationalists emphasise the problems of the Web as a learning environment. Many of these are inherent in any information system:

- disorientation
- navigation inefficiency
- cognitive overload, where the amount of information provided exceeds what is needed

so that it is difficult to separate system or navigational information from the "real" answer [2]. Students are seen to need conceptual knowledge in several overlapping domains to use the Web successfully:

- information retrieval skills
- knowledge of how the system works
- subject domain
- problem solving skills.

There is therefore an element of *Catch 22* in using a medium to teach skills that the student needs to have in order to utilise the medium effectively.

These are skills, however, that will stand the Software Engineer in good stead. A second issue that has plagued Software Engineering education is the multi-disciplinary nature of the skills and knowledge required to be active as a competent Software Engineer. As an example, Zucconi [23] suggests that problem-solving, metacognition and knowledge construction skills are vital. The Web as a learning environment provides opportunity for obtaining this knowledge.

Yet another challenge involves addressing the learning styles of individual students. Firstly, a Web-based learning environment requires an infrastructure to support the students and foster their construction of knowledge, without controlling the learning process:

- to present information within an organised framework
- to evaluate whether the material is being covered appropriately
- to know what component in relation to the whole course has and still needs to be undertaken within the timeframe
- to ensure meaningful interpretations are made and learning objectives achieved.

In addition to providing means for the student to self-regulate their learning, the infrastructure must be able to provide the teacher with mechanisms to evaluate that the learning is meaningful within the requirements of the course.

A second factor to consider is the influence of course design. Miller and Miller [17] suggest that the strategies used to *present the content* and strategies used to *sequence delivery of course content* will determine to a large extent the manner in which a student interacts with the material. This course design conveys information that shapes student experience, including

- expectations about the purpose of learning
- depth of reflection and understanding
- degree of learner control

and is also expected to support, not control, the learning process.

Results of our investigation of the learning styles of our students are useful in ensuring the infrastructure can cater for this diverse set of requirements and expectations.

4.1. Learning styles

Whilst there are numerous instruments for assessing learning styles, those advocated by Kolb [14] and Soloman and Felder [21] are well known, and accepted within education theory [18]. Both instruments provide an efficient way to analyse our students' learning styles.

Kolb's *Learning Style Inventory* is a simple test based on experiential learning theory. It looks at four stages of the learning process: **concrete experience** (CE), **reflective observation** (RO), **abstract conceptualisation** (AC), and **active experimentation** (AE). A series of twelve questions are presented and the user has to rank four possible answers for each question. The users learning style can then be identified as either:

- **Accomodator**: *What if?* people. Often start with what they see and feel then plunge in and seek hidden possibilities. They learn by trial an error and self discovery
- **Diverger** : *Why or why not?* These people study life as it is and reflect on it to seek meaning. They learn by being involved and need to listen and share with others
- **Converger**: *How?* These people start with an idea and try it out, they like to
- find out how things work and learn by testing theories
- **Assimilator**: *What?* people. These people come up with ideas and then reflect on them. They like to know what the experts think

(summarised in [3]).

The results in Table 1 (building upon our previous studies [9], which are ongoing) show that our students' preferred learning styles are diverse and span different types, **Diverger**, **Assimilator** and **Converger**. The variety of student types attracted to our programs is excellent given the multi-disciplinary nature of our curriculum content. However, the challenge is to cater for this diversity. Our staff is showing a greater trend towards **Coverger** and **Assimilator** types. This is in line with Kolb [14] stating that engineering is a good career area for **Convergers** and teaching for **Assimilators**. It is interesting to note the lack of **Accomodator** types in the staff or student populations. These types are impractical people who are not directed towards goals but good risk takers [14]. Engineering appears not to appeal to this category.

Table 1. Learning Style Inventory Results (Kolb)

	No. of Clients	Accomodator	Diverger	Assimilator	Converger
Engineering Students 1st year	33	6%	27%	33.5%	33.5%
Engineering Staff	10	0%	10%	50%	40%

In contrast to Kolb's *Learning Style Inventory*, Soloman and Felder's [21] *Index of Learning Styles* assesses learning preferences on four dimensions, **active/reflective**, **sensing/ intuitive**, **visual/verbal**, and **sequential/global**. This instrument consists of forty-four simple questions with a choice between two possible answers.

Table 2. Index of Learning Style Survey Results (Soloman & Felder)

	No of Clients	Processing	Perception	Input	Understanding
Engineering Students	33	Active 55%	Sensory 70%	Visual 79%	Sequential 64%
		Reflective 45%	Intuitive 30%	Verbal 21%	Global 36%
Engineering Staff	9	Active 11%	Sensory 33%	Visual 67%	Sequential 56%
		Reflective 89%	Intuitive 67%	Verbal 33%	Global 44%

The results from Table 2 show that:

- 55% of the students learn best actively, yet our teachers are mainly reflective
- 70% of the students are sensors, yet our teachers tend to be intuitive
- 79% of the students are visual, yet traditionally material is presented to them verbally or in written form
- 36 % of the students are global learners, yet teaching is often narrowly focused.

These also build upon our previous studies [9].

A potential mismatch between the teaching styles of the staff and the learning style of students is highlighted in both Table 1 and Table 2. Students whose learning styles are compatible with the teaching style adopted within a course tend to retain information better, obtain better grades and maintain a greater interest in the course [8]. Yet the diversity of learning styles in our students suggests that *flexibility* in teaching style is of considerable importance.

5. The Online Engineering Learning Environment

In order to address these issues within the School of Engineering, two environments have been built to underlie the courses offered online. The components of this infrastructure comprise:

- elements common to the two environments
- support provided to early year students to plan and monitor their own study programme
- a navigational scheme provided for senior students with more developed study habits. This allows students to complete elements of a course at their own pace, and with some degree of choice as to the order in which topics are studied.

Examples (and screen images) of some of the teaching tools described below can be found at http://eng.murdoch.edu.au/WebTeachingDemo/ MUEpage0.html. under the *Demonstrations* heading.

5.1. Components in common

The cognitive issues of designing online material have been well documented (see, for example [7]), and are generally accepted as goals for Web-based design. Both environments are set up to present a coherent system and learning context, in keeping with these goals. Rules are established in each so that the cognitive overhead required by the medium in minimised through:

- consistency (limiting the appearance of fragmentation)
- effortless/automatic navigation
- increased orientation so that the content (not just the user interface) allows the student to identify current position, history, options, etc.

Both environments provide learning support in terms of access to discussion fora, email, bug-reporting etc. Where they differ is in the amount of direction and formal structure provided by the infrastructure. This is best described through the comparison of the environments discussed below.

5.2. Support for planning and monitoring own study programme

Moshman [19] suggests

- *exogenous* - characterised by recognition of the value of direct instruction, but with increased learner control. Formal instruction can help learners to form knowledge representations which they can later accommodate to their subsequent experience
- *endogenous* - allowing/enforcing active exploration as a mechanism for knowledge discovery. Emphasises the individual nature of each learner's knowledge construction – teacher is there to facilitate disequilibrium by providing appropriate experience
- *dialectic* - a focus on social interaction and group work. Learning occurs through realistic experience, but that learners require scaffolding provided by experts or teachers as well as collaboration with peers [5].

as categories of constructivist learning. The relationship between these categories, and some examples of learning models based on them, are illustrated in Figure 3.

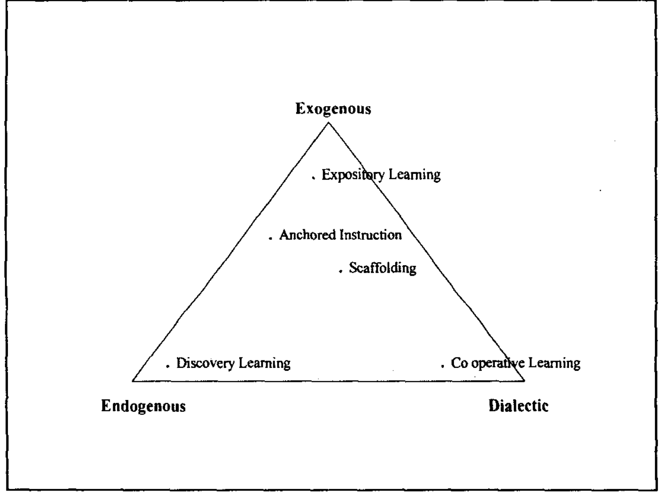

Figure 2 Moshman's constructivist categories

The environment provided in the early years of study exposes students to mechanisms that enable them to take charge of their own learning progress. This environment may be categorised as exogenous and is characterised by recognition of the value of direct instruction, but with increased learner control. This model requires opportunity for putting knowledge into practice through the use of quizzes, multiple choice and the like to provide feedback [5].

Two infrastructure tools are used extensively in this environment:

- *Monitoring progress:* Students have access to a tool that supports the planning and management of their work patterns. The *Progress Monitor* acts as planning tool in that students are provided with numerous milestones against which they may pace themselves. However, it should be noted that while students are encouraged to monitor their own progress, this is not enforced
- *Learning feedback:* While feedback on activities is standard educational practice, immediate feedback has greater effectiveness in a constructivist environment as it enables the student to alter the way information will be encoded. Learning is further enhanced where explanations are linked to multiple attempts. The *MCQ* (Multiple

Choice Questions) environment allows the teacher to set several parameters:

- whether the student can browse
- whether a set of questions can be attempted more than once
- time/date of test availability.

Questions/answers/explanations are input and optionally assigned a degree of difficulty, with a 'set' composed by including/excluding specific questions. After an attempt, the student chooses to have the test marked. Once marked short explanations can often be found under the "?" buttons. The student record database records visits, attempts and score achieved. This information is available to the course co ordinator. While the degree of difficulty feature is not greatly used at present, the ability to vary this will allow students quickly to gauge what is at their zone of proximal development (and therefore just beyond current ability), where the learning is more positively effected [22].

5.3. Support for senior students

Within the second year, and increasingly in 3rd and 4th year, SE students work within a less structured (but supportive) learning environment. In contrast to the earlier environment, the *Software Factory* environment assumes a learner-directed discovery of knowledge. Lectures and tutorials are replaced by workshops that focus on human contact and provide support through worked examples, discussion and a forum for review of understanding. The teacher interacts to convey attitudes, experience and motivation to attack the material [1]. This complements well the *dialectic* environment (with a focus on social interaction and group work) outside the Web-based component of the Software Engineering curriculum.

Web-based environments are said to draw on this *endogenous* constructivist model by allowing/enforcing active exploration as a mechanism for knowledge discovery. A study in Singapore in 1998 concluded that a strategy of minimal rote tuition and a focus on raising student motivation to explore topics at their own pace resulted in demonstrably improved success in grades [10]. Implicit, however, is the availability of support tools and scaffolding to assist the learner.

5.3.1. Context: Constructivist theory makes much of establishing a context for learning so that opportunity to construct personal meaning is enhanced [17]. Within the two clusters (each comprising four courses with an emphasis on theory or application) which make up the Software Factory, topics are categorised mnemonically. This allows for "chunking big" which focuses on connections between topics in the same category for content- and context-dependent knowledge construction [13].

5.3.2. Production line: Within each course topics are sequenced and displayed on a production line/underground map that provides alternative routes from commencement to completion. To a certain extent these provide choice in the order of topics studied and allow students to vary the sequencing of content. This degree of freedom to control access to information is not unlimited. While, in theory, the exam date is the only relevant marker for completion of the course, in practice milestones (in the form of assignments/projects) and support in the form of workshop schedules dictate the dates by which topics must be completed. External and summer students have some greater degree of freedom by not being involved in workshops.

Instead of the *Progress Monitor* provided in the early years environment, the *Software Factory* allows students to graphically indicate *inprogress/completed* information for specific topics. The expectation here is that teacher monitoring is not as vital since the students have (hopefully) better developed study skills to allow them to undertake "purposeful navigation" [17 and hence meaningful interpretation of the material.

In addition, the *Production Line* enables students to easily "explore the world" of each course – each node is directly linked to the relevant topic for browsability, although backwards/forwards links exist between topics as well. This is one mechanism for addressing the preference for global learning exhibited by some of our students.

5.3.3. Scaffolding: While a constructivist learning environment implies a focus on activities/real-world problem solving, online/interactive activities cease to be meaningful if the student hits a snag and is unable to progress from there. The purpose of scaffolding is to provide activity-sensitive help mechanisms.

The *Software Factory* provides examples both of purpose-built activity help and underlying manuals. The former takes the form of an icon on an activity screen, while the latter is best demonstrated through the underlying help in the FM (Formal Methods) topics, where help is activated through 'hot' spots in the notation itself.

Both of these mechanisms are not imposed on the student, but are readily available. Links to the help mechanisms are seamless, which enables the student to maintain focus on the learning activity, rather than on the task of retrieving aid.

Anecdotal evidence shows dependence on the scaffolding (especially the Z manual) decreases over the semester. However, the scaffold is never withdrawn, but afterwards acts as a reference tool in the same way that a dictionary or user manual does.

Other tools (such as the CASE tools) act both as scaffolding and impart necessary skills – using the CASE tool, for example, won't allow students to perform "illegal" moves. This is a learning outcome in its own right.

6. Challenges

Obviously this approach to teaching/learning provides us with some challenges:

- ensuring off campus students are included in a collaborative learning environment without face-to-face contact
- ensuring students are not swamped with information – that objectives and outcomes for each course can be discerned without face-to-face cues from academic staff
- ensuring the student's preferred learning styles are taken into consideration within the environment
- shortcomings in the evaluation of student learning from online courses – student feedback provides us with some information.

We are working towards addressing these issues.

One approach, following a suggestion by Felder [8], has been to discuss with students their learning styles and the strengths and weaknesses associated with each. We have now incorporated a topic into our first year Foundation Unit (which the majority of students complete) to survey and discuss student learning styles. This then gives the student an awareness of issues surrounding their learning and how to get the best from the learning environments that will be offered to them.

In conclusion, what we hope we are providing is a rich learning environment that encourages multiple learning styles and multiple representations of knowledge and supports the communications and negotiation processes between members of the class community, as they become life long learners and competent Software Engineers.

References

[1] R M Allen, P L Lee, N Nelissen and S S Shastri, "An integrated learning environment for Control Engineering education", *First year in higher Education: proceedings of the 3rd Pacific Rim Confeence*, Auckland, 1998, pp 1-11

[2] D S Brandt, "Constructivism: teaching for understanding of the Internet", *Communications of the ACM*, 40(10), 1997, pp 112-117.

[3] S Burns, "There's more than one way to learn", *Australia Wellbeing*, 33, 1989, pp 4244.

[4] D Bagert et al, "Guidelines for Software Engineering Education: version 1.0", Working Group on SE Education and Training, CMU/SEI-99-TR-032, 1999, http://www.sei.cmu.edu/publications/documents/99.reports/99tr032/99tr032 abstract.html

[5] B Dalgarno, "Constructivist computer assisted learning: theory and techniques", *Making New Connections: proceedings of ASCILITE '96*, University of Adelaide, 1996, pp 127-148.

[6] R Dupuis et al, A Guide to the Software Engineering Body of Knowledge: a Straw Man version, IEEE Computer Society, 1998, http://www.swebok.org/

[7] S Ebersole, "Cognitive issues in the design and deployment of interactive hypermedia: implications for authoring WWW sites", *Interpersonal Computing and Technology: an electronic journal for the 21st century*, 5(1-2), 1997, pp 12-36, http://jan.ucc.nau.edu/~ipct-j/1997/n2/ebersole.html

[8] R Felder, "Reaching the second tier: learning and teaching styles in college science education", *Journal of College Science Teachin,g* 23(5), 1993, pp 286 -290.

[9] L Fowler, M Allen, J Armarego and J Mackenzie, "Learning styles and CASE tools in Software Engineering", in [11], 2000, pp 169-180, http://cleo.murdoch.edu.au/confs/tlf/tlf2000/fowler.html

[10] R S Gilliver, B Randall, and Y M Pok, "Learning in cyberspace: shaping the future", *Journal of Computer Assisted Learning*, 14, 1998, pp 212-222.

[11] A Herrmann and M M Kulski (eds), *Flexible Futures in Tertiary Teaching: Proceedings of the 9th Annual Teaching Learning Forum*, Curtin University of Technology Perth (WA), 2000, http://cleo.murdoch.edu.au/confs/tlf/tlf2000/contents.html

[12] R Ibrahim (ed), *Software engineering education: 8th SEI CSEE Conference, Proceedings*, New Orleans, USA, Springer-Verlag, Berlin, 1995.

[13] D H Jonassen, "Hypertext as cognitive tools" in [15] 1992, pp 147-148.

[14] D A Kolb, Experiential Learning Experience as the Source of Learning and Development, Prentice-Hall New York, 1984.

[15] P A Kommes, D H Jonassen, and J T Mayes (eds), Cognitive Tools for Learning Springer-Verlag, Berlin, 1992.

[16] D M Laurillard, Rethinking University Teaching: a framework for effective use of educational technology, Routledge, London, 1993.

[17] S M Miller and K L Miller, "Using instructional theory to facilitate communications in Web-based courses", *Educational Technology & Society*, 2(3), 1999, http://ifets.ieee.org/periodical

[18] S M Montgomery, "Addressing diverse learning styles through the use of multimedia", *Engineering Education for the 21st Century: Proceedings of the 25th Annual Frontiers in Education Conference*, 1995, pp 3a2.13-3a2.21 http://fre.www.ecn.purdue.edu/FrE/asee/fie95

[19] D Moshman, "Exogenous, endogenous and dialectical constructivism", *Developmental Review*, 2, 1982, pp 371-384.

[20] G G Roy, J Armarego, T Woodings and C P Lam, "Internet-based Software Engineering education at Murdoch University", *Forum for Advancing Software Engineering Education (FASE)*, 9(10) (117th Issue), 1999.

[21] B R Soloman and R Felder, *Index of Learning Styles (ILS)*, 1999, http://www2.ncsu.edu/unity/lockers/users/f/fel der/public/ILSpage.html

[22] L S Vygotsky, Mind and Society: the development of higher psychological processes, Harvard University Press, Cambridge (MA), 1978.

[23] L Zucconi, "Essential knowledge for the practicing Software Engineer and the responsibilities of university and industry in her education", in [12], 1995, pp 5-13.