Accreditation of Monash University Software Engineering (MUSE) Program

Sita Ramakrishnan
Clayton School of IT, Faculty of IT,
Monash University, Australia

sita.ramakrishnan@infotech.monash.edu.au

Abstract

Engineering programs in Australian Universities are accredited by Engineers Australia (EA) based on certain strict guidelines. This paper discusses the undergraduate SE curriculum and accreditation effort undertaken over the last ten years at Monash University in order to achieve a successful outcome. The paper describes how the SE curriculum has evolved over this period at Monash and maintained its product quality by benchmarking against various international efforts such as the CMU-SEI effort in early 1990s, ACM/IEEE efforts on Software Engineering Body of Knowledge (SWEBOK, versions 2001-2004) and the curriculum guidelines for each major area of computing in Computing Curricula (CC2001) such as a Software Engineering volume (SE2004). Currently at Monash, student-centric evaluations are used to determine the teaching/learning outcome and in-form the world through the web to support the University's quality assurance and improvement strategies. We discuss our effort in providing an aligned, evidence-based approach to quality assurance for continued accreditation of MUSE.

Keywords: accreditation, curriculum, software engineering, teaching/learning outcomes, quality system process

Introduction

In Australia, you must graduate from an accredited engineering program to be assured graduate membership of Engineers Australia (EA). Assessment of an engineering program for accreditation by Engineers Australia is based on the curriculum: structure and content, the teaching and learning environment and the quality assurance framework (http://www.ieaust.org.au/membership/accreditation.html).

At Monash University, we have had three versions (iterations) of our Bachelor of Software Engineering (BSE) curriculum over the past ten years. In the next section, we discuss the evolution of our BSE curriculum in detail. The curriculum has to satisfy the product quality requirements from a number of perspectives such as local and international students, academics teaching into the program, accreditation bodies such as Australian Computer Society (ACS), IE Aust. (also known as Engineers Australia (EA)), international student recruitment agencies and Australian University Quality Agency (AUQA), and the fed-

eral government in terms of funding. In this age of global competition for students between Universities, it is important to show evidence of course (product) quality based on accreditation by professional bodies such as ACS & EA, university quality rating for teaching and learning offered by AUQA, and student evaluation of units & courses to target good students to enroll in the SE program. The external product quality must be supported by sound internal processes as part of the quality systems in the Faculty and University. The alignment of the product and process is necessary for ensuring a quality teaching/learning program (Barrie, Ginns & Prosser, 2005; Barrie & Prosser, 2003; Ramakrishnan, 2003) and for informing the students, academics, accreditation bodies, the auditing body such as AUQA and the international student recruitment agencies. The product here refers to the course curriculum, structure and content, and is detailed in the section on "Product Quality Requirements". The accreditation body is not only interested in the course curriculum quality but also in the quality system processes including the teaching & learning environment in place in the School /Faculty in charge of delivering the SE course. The approval and evaluation process and quality assurance consideration associated with internal organizational committees in the University, external bodies such as EA & AUQA, and performance-based funding model for teaching and learning are explored in the section on "Quality System Process". This is followed with the section on "Related work on SE curriculum". We conclude with a summary.

Evolution of the Undergraduate Software Engineering Program

In the first iteration in the mid 1990s, we considered the educational requirements of a SE curriculum based on CMU/SEI and ACM model curricula (CMU/SEI-90-TR-3) for undergraduate SE education (Ramakrishnan & Schmidt, 1998). It should be noted that with each of these iterations of the curriculum, we have checked for conformance against what may be treated as a reference framework at the time for a SE curriculum. In iteration 1, we grouped the core units of the four year undergraduate software engineering program into four clusters: software development process, software analysis, software system construction and programming foundations and these clusters were supported by units in Mathematic Foundations, Project Management, Communication Skills and Industry-based Project. As software engineering is not just about software, we included the science of engineering such as Physics, and Engineering Units such as Electrical Engineering, Telecommunication, Mechano-informatics, Reliability Engineering and Engineering Management. The inclusion of engineering units also met the needs of the Faculty of Engineering at Monash and the accrediting body, Engineers Australia (known as IE Aust. in the 1990s). We produced this SE curriculum in 1998 and commenced delivery of the program in 1999 under the joint management of the Faculty of Engineering and Faculty of IT at Clayton campus, Monash University. We received a provisional IE Aust. accreditation for the BSE course until 2003 along with other Engineering courses in the Faculty of Engineering.

Negotiations as to the structure and management of the program continued between the Faculty of Engineering and Faculty of IT had continued through out 1998, and resulted in an agreement to transfer management of the Bachelor of Software Engineering (BSE) to the Faculty of IT, and a complementary transfer of Bachelor of Computer Science & Engineering (BCSE) to the Faculty of Engineering and its renaming to Bachelor of Computer Systems Engineering. The re-branding of the 2 programs occurred with BSE more focused on "software" engineering and BCSE more focused on "hardware" and telecommunication technology. As a result of this re-branding and transfer of BSE management to the Faculty of IT, we removed the science unit such as Physics and Engineering units from the set of BSE core units. Students were still able to take some of these units as electives in their study.

In this second iteration of the BSE program at Monash (2000-2004), we decided to check our revised SE curriculum (see Table 1: Bachelor of Software Engineering at Monash University: Revised Course Map – second iteration) for conformance against the reference framework at the time: Software Engineering Body of Knowledge (SWEBOK, version 2001, IEEE Stoneman version 0.7, Apr 2000). SWEBOK is an all-inclusive term that describes the sum of knowledge within the profession of software engineering (http://www.swebok.org/). We needed to and did make a clear distinction between the software engineering body of knowledge (SWEBOK) and the contents of software engineering curricula. The accreditation visit by Engineers Australia in Aug 2003 to accredit Engineering courses at Monash included this version of the BSE. An innovative web-based environment called DoIT (Ramakrishnan & Cambrell, 2002) was demonstrated to the accreditation panel. DoIT system provides a visual interpretation for the students and the academics of what students have learnt in our BSE in terms of SWEBOK for the four years of study. We had very positive responses regarding our BSE curriculum and the DoIT project during our accreditation period from leading academics in SE such as Prof Bertrand Meyer and Prof Alain Abran, École de technologie supérieure - Université du Québec. Prof. Meyer attended a session of the accreditation panel visit and made very strong positive remarks about the Monash SE curriculum. Prof. Meyer and Prof. Alain Abran were very impressed with the DoIT system demonstration.

We received a conditional accreditation from Engineers Australia, and one of the major recommendations was to put more "engineering" into our BSE program and recognize that SE is both a computing and an engineering discipline. The accrediting body wanted our BSE to include core and/or elective sequences sourced from the Faculty of Engineering in order to broaden the exposure of students to professionals in other disciplines, and to strengthen an awareness of engineering risks associated with practical system limitations. Engineers Australia also reported that our BSE program fulfilled one of their important requirements that all professional engineering programs include a major, capstone thesis/project activity. Students in the capstone project are required to address an open-ended complex problem with broad-based multi-disciplinary considerations and embracing the full design cycle. The capstone project is well orchestrated in the BSE program – Software Engineering Studio Project (Ramakrishnan, 2003). Students work in teams on an industry specified project, negotiating a formal analysis and the development of a legal IP agreement before embarking on the design and development cycle. The projects embrace every aspect of Software Engineering product development in a real industry setting and conclude with the formal processes of product acceptance testing, presentation, and documentation. Student team progress is tracked and assessed via interviews, peer reviews, journal & diary entries of project meetings minutes. Students are exposed to professional practice thorough a regular interface with the industry client. However, the accreditation panel recommended that we introduce a mandatory internship (paid relevant SE/Engineering vacation employment for 12 weeks in summer) as part of the four year study as a formal industry based work placement. This has been instituted in the third iteration of our BSE program.

Table 1: Bachelor of Software Engineering at Monash University: Revised Course map (second iteration)

First year				
S1	**CSE1301** Computer Programming	**CSE1401** Introduction to Software Engineering	MAT1841 Mathematics for Computer Science I	**Free Elective**
S2	**CSE1303** Computer Science	**CSE1402** Technical Documentation for Software Engineers	MAT1830 Mathematics for Computer Science II	**Free Elective**
Second year				
S1	**CSE2201** Software Engineering (SE) Practice	**CSE2303** Formal Methods I	**CSE2304** Algorithms and Data Structures	**CSE2/3324** Computer Architecture
S2	**CSE2302** Operating Systems	**CSE2305** Object Oriented Software Engineering	**BUS2176** Project Management	**CSE2/3325** Multimedia Programming and the World Wide Web
Third year				
S1	**CSE4213** Formal Methods in SE	**CSE3308** Software Engineering: Analysis and Design	**CSE2/3391** Unix Tools / **CSE2/3395** Perl Programming	**Free Elective**
S2	**CSE3302** Software Engineering Project	**CSE3322** Programming Languages and Implementation	**CSE3323** The Computer Industry: Historical, Social and Professional Issues	**Free Elective**
Fourth year (Pass degree)				
S1	**CSE4002** Software Engineering studio project (Full year project)	CSE4431 System verification and validation, quality and standards	Approved elective	**Free Elective**
S2		**CSE4333** Parallel Systems	Approved elective	**Free Elective**
Fourth year (Hons stream)				
S1	**CSE4002** Software Engineering studio project (Full year project)	CSE4402 Software engineering research project (Full year project) (Hons students only by invitation based on results)	**CSE4431** System verification and validation, quality and standards	Approved elective
S2			**CSE4333** Parallel Systems	Approved elective

Table 2: Bachelor of Software Engineering at Monash University: Revised Course map (third iteration)

for student intake 2006 refer to *http://www.infotech.monash.edu.au/units/* for handbook entries

First year				
S1	FIT1001 Computer systems	FIT1002 Computer programming	MAT1841 mathematics for Computer Science I	ENG1061 Engineering profession
S2	FIT1010 Introduction to Software Engineering	FIT1008 Computer Science	MAT1830 Mathematics for Computer Science II	Approved elective
Second year				
S1	FIT2022 Computer Systems	FIT2010 Database	FIT2004 Algorithms and data structures	FIT2024 Software Engineering Practice
S2	FIT2001 Systems analysis and design	FIT2008 Networks and data communications	FIT2014 Theory of computation	FIT2043 Technical documentation for software engineers
Third year				
S1	FIT3086 IT project management	FIT3077 Software engineering: architecture and design	FIT3042 Systems tools and programming languages	Approved elective
S2	FIT4001 Parallel and distributed systems	FIT3013 Formal methods for software engineering	FIT3084 Multimedia programming and the www	Approved elective
Summer	12 week Industry Placement			
Fourth year (Option 1)				
S1	FIT4002 Software Engineering studio project (Full year project)	FIT4004 System verification and validation, quality and standards	Approved elective	Approved elective
S2		Approved elective	Approved elective	Approved elective
Fourth year (Option 2)				
S1	FIT4002 Software Engineering studio project (Full year project)	FIT4003 Software engineering research project (Full year project) (Hons students only by invitation based on results)	FIT4004 System verification and validation, quality and standards	Approved elective
S2			Approved elective	Approved elective
Fourth year (Option 3)				
S1	FIT4013 Software engineering research project (Full year project) (Hons students only by invitation based on results)		FIT4004 System verification and validation, quality and standards	Approved elective
S2			Approved elective	Approved elective

In the third iteration of the BSE program at Monash (2005-current), we decided to adopt all of the Engineers Australia's recommendation in Dec 2004 to obtain the full accreditation for BSE. We also checked our revised BSE curriculum (see Table 2: Bachelor of Software Engineering at Monash University: Revised Course Map – third iteration) against current reference frameworks as discussed next. This revised BSE curriculum was benchmarked against current reference frameworks of: Software Engineering Body of Knowledge (SWEBOK, version 2004, http://www.swebok.org, updated in Feb. 2005), Software Engineering (SE) 2004 and Computing Curriculum (CC) 2001/2005. Next, we discuss some basic details of these frameworks. In the 2004 guide to the software engineering body of knowledge (SWEBOK), the IEEE Computer Society established a baseline for the body of knowledge in the field of SE. However, it should be noted that between 1993 and 2000, the IEEE Computer Society and the Association of Computer Machinery (ACM) had promoted the professionalization of SE through their joint Software Engineering Coordinating committee (SWECC). The body of knowledge has been developing and evolving over the past four decades and will need to continue to evolve as SE matures. Since 2001, IEEE has contracted the Software Engineering Management Research Laboratory at the University of Quebec at Montreal (UQAM) and École de technologie supérieure - Université du Québec to manage the SWEBOK project (Bourque & Dupuis, 1998). Under Prof. Abran's executive editorship of SWEBOK2004, SWEBOK has become an ISO Technical report: ISO TR 19759. An evolution process has been designed to update the SWEBOK document and this will be synchronized with the ISO regular review process. In 2001, the ACM and IEEE-CS published Computing Curricula 2001, which contains curriculum recommendations for undergraduate programs in computer science (CS). CC2001 report also called for additional discipline-specific volumes for each of computer engineering (CE), information systems (IS), and software engineering (SE). The CC2001 task force decided to produce a set of curriculum guidance documents and have produced CS2001, CE2004, IS2002, and SE2004. SE2004 is a document that provides recommendations for undergraduate program in software engineering (http://sites.computer.org/ccse/). SE2004 was sponsored by the Association for Computing Machinery (ACM) and the IEEE Computer Society. The primary purpose of SE2004 is to provide guidance to academic institutions and accreditation bodies about what should be included in an undergraduate SE education. SE2004 includes the Software Engineering Education Knowledge (SEEK), a list of topics that all SE graduates should know, as well as a set of guidelines for implementing curricula and a set of proposed courses. More information on the overall computing curriculum effort is available on the IEEE Computer Society Education Board web site at http://www.computer.org/education/cc2001, http://www.sigcse.org/cc2001/ and http://www.acm.org/education/curricula.html.

During this iteration of the curriculum in 2005, we also had to accommodate the restructuring of all undergraduate programs in the Faculty of IT at Monash. The restructuring involved the introduction of a set of common first year level core for all courses in the Faculty of IT in Feb. 2006. The common units are: Programming unit using Java, Computer Systems unit, Systems Analysis unit, Database unit, Network & Data Communication unit, IT Project Management and IT in Organization. The common core units are meant to provide a solid initial foundation of basic principles and practice, and flexibility to move between various computing courses in the Faculty. We have retained the 2 Mathematics units in the first year of study and moved some of the common core first year level units where appropriate in the revised BSE course structure. We also replaced the IT in organization core unit with the Engineering Profession unit from the Faculty of Engineering to comply with the accreditation requirements of Engineers Australia that students develop a sense of engineering ethos and understand the responsibilities of being an engineer from year 1 of their study. With the evolution and maturing of the SE field, languages such as Java, model based architecture driven approaches have become more mainstream. This is reflected in our latest iteration of the BSE curriculum in our Programming, and Architecture & De-

sign units. In general, students have been positive towards the change to Java as the 1st language and with the common 1st year core in the Faculty of IT. Apart from changing the programming language in year 1 & 2 from C++ to Java, and updating the SE Analysis & Design unit, most of the units are a direct mapping from the 2nd iteration of BSE. Database and Data communication which were electives in the 2nd iteration have been made core in the 3rd iteration. A couple of theoretical units such as Operating systems and Formal Methods I have been merged to be offered as Theory of Computation in the 3rd iteration. A mandatory 12 weeks industry placement has been introduced in 2006 for BSE students and the approved electives include Engineering Faculty units as per the IE Aust. recommendation. It must be noted that existing cohort of BSE students are able to complete their program with the old structure (iteration 2) of the course. The first year of the new structure (iteration 3) was made available to the new cohort of students in Feb 2006. Year 2, 3 & 4 of iteration 3 will be made available in 2007, 2008 & 2009 respectively. The BSE program at Monash was accredited by IE Aust. in Feb. 2006 and the full accreditation is valid till the intake in 2008.

Product Quality Requirements - Course Curriculum: Structure and Content

Program structure: An overview

The Bachelor of Software Engineering (BSE) Program is available at Clayton Campus of Monash University, Australia in on-campus mode only. BSE is a four year full-time program. The course commences with the establishment of a sound foundation in introductory information technology and mathematics. All information technology units have approximately one-third laboratory-based programs. In the later years, the introduction of major software engineering projects builds the students' self-reliance and planning capabilities in both individual and team-based environments. Project management units strengthen the formal basis of management skills. Elective units are provided to allow specialisation in some aspect of the field of study, with free electives to permit broadening of intellectual and personal horizons.

The course structure balances four major strands:

1. Synthesis: software systems construction and design, including methodologies and notations.
2. Analysis: software artifact analysis including mathematical foundations, evaluation and measurement.
3. Processes: software and team management including software lifecycle and software projects.
4. Systems: understanding, abstracting, reusing and maintaining systems and components, including exposure to the architecture and principles of large systems such as operating systems and distributed systems.

Some units fall clearly into only one of these strands. Others, particularly early units, may address several strands. The four-year course is based upon the four-year engineering degree structure, from which it is derived. In particular, the honours program is integral with the four years of study and is undertaken in the fourth year, with enrolment in the honours research stream predicated upon students reaching a credit level of performance in the first three-year levels. This standard of performance is determined from a weighted average of results over the first three levels, with first level having a weight of one, second level a weight of two, and third levels a weight of three. These results, together with results in the fourth and final level, are used to determine final

grades, with final- level results having a weight of six and the overall result is graded according to the honours system (I, IIA, IIB, III).

BSE Curriculum, DoIT, Generic Attributes of a Software Engineer and Product Quality

We reiterate that the BSE program was first introduced in 1998 at Monash and the structure given provisional approval by IE Aust. in 1998 until 2003.The program was first offered in 1999 and was revised (2nd iteration) in 2000. The third iteration of the course structure occurred for the new intake in Feb. 2006. Next, we list the generic attributes or capabilities that a graduate must develop in a degree program as required by the accrediting body, IE Aust.

Our BSE program must ensure that the graduates develop to a substantial degree the generic attributes or capabilities, (a) – (j) listed below, to satisfy IE Aust. accreditation of professional engineers. The generic attributes are as follows:

a) ability to apply knowledge of basic science and engineering fundamentals

b) ability to communicate effectively, not only with engineers but also with the community at large

c) in-depth technical competence in at least one engineering discipline

d) ability to undertake problem identification, formulation and solution

e) ability to utilise a systems approach to design and operational performance

f) ability to function effectively as an individual and in multi-disciplinary and multi-cultural teams

g) understanding of the social, cultural, global and environmental responsibilities and the need for sustainable development

h) understanding of the principles of sustainable design and development

i) understanding of professional and ethical responsibilities and commitment to them

j) expectation of the need to undertake lifelong learning, and capacity to do so.

We show how it was covered in our units to show the elements of total learning experiences in the four year SE program during the accreditation visit in 2003 (see Table 3). As outlined in the IE Aust. Accreditation Manual, our Software Engineering Curriculum provides an integrated set of learning activities and experiences to the students and endeavours to capture the following elements in the kinds of percentages suggested by IE Aust:

- mathematics, science, engineering principles, skills and tools appropriate to the discipline of study (not less than 40%)
- an engineering discipline specialisation (about 20%)
- integrated exposure to professional engineering practice, including management and
- professional ethics (about 10%)
- more of any of the above elements, or other elective studies, hardware (about 10%).

Table 3: Elements of total learning experiences in the four year SE program shown through generic attributes

BACHELOR OF SOFTWARE ENGINEERING DEGREE
(part of Accreditation document, Aug 2003)

Column Keys:
CP Credit points;
A Maths, Engineering principles, skills, tools related to S.E. -relate to SWEBOK (not less than 40%)
B Design, Analysis & Projects (about 20%)
C Discipline Specialisation relate to SWEBOK & PMBOK (about 20%)
D Exposure to Professional Practice & Professional ethics (about 10%)
E more of any of the above elements, or other elective studies (about 10%)
F ~%

Levels 1-4	CP	<-------A 40%-------->		B 20%	C 20%	D 10%	E 10%	F ~%	Totals
Core units		show Math % separately							
Level 1		Rest of A%	Math%						
MAT1841	6		100 (6.0)						
MAT1830	6		100 (6.0)						
CSE1301	6	100 (6.0)							
CSE1303	6	50 (3.0)		40 (2.4)				10 (.6)	
CSE1401	6	40 (2.4)		20 (1.2)	20(1.2)	20 (1.2)			
CSE1402	6	90(5.4)				10(.6)			
(Level 1 core - % denoting proportions of total learning experience)									
	36	16.8	12	3.6	1.2	1.8		0.6	36
		47%	33.33%	10%	3.33%	5.00%		1.67%	
Level 2									
CSE2303	6	20 (0.8)	40(1.6)	40(1.6)					
CSE2304	6	50(3.0)	25(1.5)	25(1.5)					
CSE2201	6	20 (1.2)		60 (3.6)	10(.6)	10(.6)			
CSE2/3324	6	40(2.4)						60 (3.6)	
CSE2302	6	30(1.8)		70(4.2)					
BUS2176	6	50(3.0)		30(1.8)		20(1.2)			
CSE2305	6	40(2.4)		40(2.4)	20(1.2)				
CSE2/3325	6	40(2.4)		40(2.4)	20(1.2)				
(Level 2 core - % denoting proportions of total learning experience)									
	48	17	3.1	17.5	3	1.8		3.6	48
		35%	6.46%	36.46%	6.25%	3.75%		7.50%	
Level 3									
CSE3213	6	80(4.8)			20(1.2)				
CSE3308	6	25(1.5)		50(3.0)	25(1.5)				
CSE3391/	3	80(2.4)		20(.6)					
CSE3395	3	80(2.4)		20(.6)					
CSE3302	6	30(1.8)		50(3)	10(.6)	10(.6)			
CSE3322	6	30(1.8)		70(4.2)					
CSE3323	6					100(6.0)			

Levels 1-4	CP	<-------A 40%-------->		B 20%	C 20%	D 10%	E 10%	F ~%	Totals
(Level 3 core - % denoting proportions of total learning experience)									
	36	14.7		11.4	3.3	6.6			36
		41%		31.67%	9.17%	18.33%			
Level 4 (Core for Hons. Stream and Pass degree)									
CSE4002	12	20(2.4)		20(2.4)	30(3.6)	30(3.6)			
CSE4431	6	30(1.8)		30(1.8)	30(1.8)	10(.6)			
CSE4333	6	20(1.2)		40(2.4)	40(2.4)				
(Level 4 core - % denoting proportions of total learning experience)									
	24	5.4		6.6	7.8	4.2			24
		23%		27.50%	32.50%	17.50%			
Level 4 (Core for Hons. Stream)									
CSE4402	12				95(11.4)	5(.6)			24
					95.00%	5.00%			
Overall core CP totals for Hons. Stream									
	156	53.9	15.1	39.1	26.7	15		4.2	156
Overall core percentages for Hons. Stream									
		34.55%	9.68%	25.06%	17.12%	9.62%		2.69%	
Total Core Credit Points (36+48+36+24) = 144 CP for Pass Degree with 48 CP Electives									
Overall core CP totals for Pass degree									
	144	53.9	15.1	41.5	15.3	14.4		4.2	144
Overall core percentages for Pass degree									
		37.43%	10.49%	28.82%	11.88%	10.00%		2.92%	

Next, we briefly mention the aims of innovative project, DoIT, that was undertaken in 2001 for establishing a quality curriculum product and how it was realized.

The BSE curriculum content was checked for coverage of Software Engineering Body of Knowledge by manually checking the details of each unit (week by week lecture notes and assessments). Then, we produced a customisable in-forming product quality environment called DoIT in 2001. The digital portfolio available from DoIT enables our BSE students to view their progression in learning, manages their knowledge capabilities and also contributes to innovation in institutional quality audit process. More details about DoIT are available in (Ramakrishnan & Cambrell, 2002). DoIT works on a number of levels:

- Innovative learning system *for students* to learn about what skills they have learnt (as per SWEBOK) as they move through the course.
- Active curriculum where *the students and academics* teaching into our BSE can view whereabouts in the course across all subjects, a theme (knowledge areas as per SWEBOK) is taught.
- Assist in the *accreditation process of our BSE* by the accrediting bodies such as ACS and Engineers Australia as our course is mapped to the core knowledge areas as articulated by SWEBOK.
- Customizable to produce a curriculum tracking system along the lines of what is required by *Monash Graduate Attributes project* (*Monash 2020 vision of Monash graduate capabilities*) as part of Australian University Quality Agency's (AUQA) audit requirement.

One can observe from the DoIT output (Ramakrishnan & Cambrell, 2002) that some of our BSE units have a very focussed content and cover generic attributes (as required by IE Aust.) such as:

a) ability to apply knowledge of basic science and engineering fundamentals (CSE1301,CSE1303, CSE3391/3395) and

b) ability to communicate effectively, not only with engineers but also with the community at large (CSE1402, CSE1303);

whereas other units have a broader focus and cover and assess a wider set of graduate attributes, drawing on knowledge and capability from different units. For example,

c) in-depth technical competence in at least one engineering discipline is covered at various levels of Bloom's taxonomy. In Software Engineering (SE) units such as CSE1401 (Introduction to SE), CSE2201 (SE Practice), CSE2305 (OOSE), CSE3308 (OO Analysis & Design), CSE4002 (SE Studio project), CSE4431 (Systems V&V, Quality & Standards) and CSE3213 (Formal Methods in SE), various SWEBOK areas are covered at different levels which can be seen in the DoIT output.

d) ability to undertake problem identification, formulation and solution – is part of most of our BSE units and forms the basis of assessing students' capability.

e) ability to utilise a systems approach to design and operational performance – a number of units from level 1 – 4 cover and assess this important aspect from the week by week descriptions of units such as: CSE2201 (SE Practice), CSE2303 (Algorithms & Data Structures), CSE2/3324 (Computer Architecture), CSE2302 (Operating System), CSE3302 (SE Project), CSE4002 (SE Studio Project) and CSE4431 (Systems V & V, Quality & Standards).

f) ability to function effectively as an individual and in multi-disciplinary and multi-cultural teams with the capacity to be a leader or manager as well as an effective team member – Students work on group projects and learn about various aspects of working effecting in teams in various roles, employing processes such as PSP, TSP and ISO9000 standards and are assessed for their role(s) in such team work in CSE2201, CSE3308, CSE3302, CSE4431 and CSE4002.

g) understanding of the social, cultural, global and environmental responsibilities of the professional engineer, and the need for sustainable development – in CSE3323 (Computer Industry: History, Social & Professional issues), such issues are explored and students are formally assessed on their understanding of such issues.

h) understanding of the principles of sustainable design and development – A number of our final year SE Studio projects (CSE4002) have been in the manufacturing sector where the students are exposed to the requirements of ISO 14001. In CSE4002, we cover "the main principles of professional management and are based on the cyclical process of 'plan, implement, check and review'. The structure of ISO 14001 also provides a common basis for integration with elements of occupational safety and health management, and quality management systems such as ISO 9000" (Refer http://www.iso14000.org/). Some of these issues forms part of their Quality manual in CSE4002. In March 2000, Monash University adopted an environment policy (Refer http://www.adm.monash.edu.au/ohse/environment/index.html) and has initiated a number of projects and initiatives since that time. Students are enthusiastic in doing their bit by double sided printing of journal, conference papers, class notes, drafts of their thesis and so on.

i) understanding of professional and ethical responsibilities and commitment to them – Such topics are covered in CSE3323 and students are expected to observe these ideas in SE Studio project (CSE4002) and in their thesis. Monash is always improving on their standards in issues such as plagiarism, cheating, copyright and IP issues.

j) expectation of the need to undertake lifelong learning, and capacity to do so – Motto of Monash is 'Ancora imparo' - 'I am still learning'. In our BSE program, we cover both the fundamental areas of SE knowledge as well as the applied and new technology areas. They learn about problem solving skills. By continuing to learn, they keep learning about how to adapt to new technology and thrive on new situations and problems as they encounter them. BSE Students undertaking the Honours stream in their final year of study are required to enroll and successfully complete a solo research thesis component.

We also implemented another innovative project called Monash University Software Engineering (MUSE) Studio Lab Facility in 2002. MUSE Studio Lab is the hardware/software infrastructure facility made exclusively available to our final year BSE students for their final year Software Engineering project. In 2002, we had a MUSE Studio Lab with 14 machines and 2 servers to house some server software, testing tools and for students' project assets. In 2003, MUSE Studio grew to 2 labs with 30 Win XP/Linux dual boot machines with servers to house tools and project assets as in 2002. The projects are sourced from the industry and students undertake these projects in groups of 4 –5 over two semesters of study in their final year program (Ramakrishnan, 2003). We also designed a MUSE portal to enable the students in 2003-2004 to manage the software assets in student teams' final year software engineering project (Capstone project). The accrediting body checked our electronic resources such as DoIT, teaching resources as well as physical facilities such as the MUSE Lab, the Monash library and lecture theatres and was satisfied with the infrastructure available.

Our students rate the MUSE Studio project very highly as they are given the opportunity to practice the SE skills they have learnt in the previous years of study on a team-based project for an industry client. They also often get to learn new technology in implementing the solution in a real-world setting. This kind of collaborative industry relevant project in the final year of SE education addresses some of the concerns of Computer Science education expressed by some academics (Arora & Chazelle, 2005; Narasimhan, 2006).

Quality System Process

The Bachelor of Software Engineering program's curriculum has been designed in such a way as to support a quality input into the syllabus content of each of the units offered into the program. The School of Computer Science & Software Engineering (called Clayton School of IT currently) and the Faculty of IT have quality processes in place regarding program planning, curriculum development, and regular course curriculum and content review, which is discussed next. The accreditation panel checked the documentation provided on quality systems in place at Monash during the panel visit in Aug 2003 and were satisfied with the procedures in place.

Curriculum Development and Review

The Associate Dean, Teaching of the Faculty in consultation with the Head of School and the Course Director oversees any new course initiatives and curriculum development activities in the School. Course directors are responsible for leading product group meetings to discuss any new proposals or revisions to existing units before they are tabled at the Faculty undergraduate education subcommittee. Each of the units is proposed by an academic in the school and submitted for approval using the MONATAR (online unit description template used in the Faculty of IT) sys-

tem. The Faculty undergraduate program subcommittee (UPSC) members review it and any revisions are recorded in the MONATAR system. It is then forwarded to the Faculty education committee (FEC) by the chair of UPSC and, after approval by FEC, it is recorded in the university handbook as a new offering.

Teaching and Learning Environment

Units offered in the Faculty of IT at Monash used to be housed in the courseware web page of the School offering the Course and was openly accessible by students, Monash community and the outside world. The content of the unit often consisted of: lecturer's details, venue, semester & year of offering, an overview, week by week details, assessment details such as hurdle requirements, assignments, exams etc. Some units also offered a feedback facility for discussion amongst students and/or lecturer, tutors and students. However, since 2004, Monash University has been piloting the use of Monash University Studies Online (MUSO) which is an internet-based teaching platform based on WebCT Vista 3. MUSO is a web-based course management system and the Faculties including the IT Faculty are promoting MUSO as the teaching and learning environment and repository for storing and managing unit information. It is seen as being more than a webpage of courseware information about a unit offering, and as an efficient course management tool for preparing the unit content, for communicating with students, managing assessments and grade, and for improving learning outcome by engaging with students using various learning styles that is appropriate for that unit. This standardized unit view has received positive feedback from students. However, only students enrolled in the unit and the lecturer offering the unit and his/her tutors can view the content, which means that it is not open for other colleagues in the School/Faculty/University other than through explicit permission changes by the MUSO administrator. So, the teaching and learning environment in MUSO is closed to the outside world, which is the current policy of Monash University.

Quality Feedback Processes

The Monash approach to quality matches the Australian University Quality Agency's (AUQA) principle that an University needs to identify and define what has be achieved in the University as a whole and its various areas, and adhere to a quality cycle for continuous improvement to achieve the objectives. The quality cycle includes: planning, acting, evaluating and improving. Evaluations are conducted regularly for monitoring short term measures and provide formative feedback for internal purposes for example using unit evaluations. Evaluations conducted for reviewing purposes consider both formative and summative feedback and are overseen by external parties such as AUQA.

Students enrolled in a unit are given the responsibility of filling Unit Evaluation forms at the end of each semester of study. Students also evaluate the lecturer's teaching of the unit by filling in teaching feedback questionnaires. The University's Centre for Higher Education Quality (CHEQ) was established in 2000 to lead and support the development of quality assurance and improvement in all areas of Monash University's operations. CHEQ processes the Monash Questionnaire Series on Teaching questionnaires centrally and results are sent to the school and the staff. Such unit evaluations are meant to be treated seriously by the academics and any comments taken on board to improve future deliveries. Graduates of Australian Universities are invited, to respond to the Course Experience Questionnaire (CEQ) and a Graduate Destination Survey (GDS) around four months after completing a course of study. These are national surveys administered by each university under the national co-ordination of Graduate Careers Australia. At Monash, CHEQ checked CEQ for: the quality of teaching, the clarity of goals and standard; the nature of assessment; the level of workload; and the enhancement of their generic skills. This was the regime

under which the SE program operated when the accreditation panel (IE Aust.) visited Monash in 2003.

The need for Monash to have systematic feedback from students was identified by CHEQ and administered through another questionnaire, Monash Experience Questionnaire (MEQ) in 2003 and in 2005. One of the aims of MEQ was to collect these feedbacks to assist Monash in the preparation of the audit in Sep. 2006 by AUQA. MEQ is a lead indicator on the course experience questionnaire which is used as a measure in the learning and teaching performance fund for the Universities from the federal government.

AUQA panel's questions with respect to BSE were focused on these quality assurance measures and our planned improvement strategies. We were not only able to mention the unit evaluation methods in place but also about performance-based incentive by funding staff for travel to conferences for doing well in unit evaluations. Monash institutional policy and strategic direction for teaching and learning have had positive impact on AUQA audit in 2006 at Monash, Faculty of IT restructuring & some of the changes and/or fine tuning of quality processes.

A centre for the advancement of learning and teaching (CALT) was established in 2006 at Monash to coordinate the University's strategic direction of instituting systematic quality assurance teaching and learning processes and support for staff and students. The two centers of CALT and CHEQ work collaboratively with the faculties across Monash as part of systematic continuous improvement strategy in response to unit evaluation data from the students to achieve improvements in student learning experiences and student satisfaction levels. This means that from 2006, we have a more stringent and open quality process for in-forming the various stakeholders including current and prospective students in place for re-accreditation of Monash University Software Engineering by Engineers Australia (EA) in 2008.

Feedback from Alumni

Some of the comments from BSE alumni are:

"The Testing subject has helped me in my role as a Performance Tester. " (graduate of 2002)

"The most value I gained from University came from the more pure software engineering subjects such as our studio project which provided far more experience then I had realized at the time" (graduate of 2003).

"I particularly enjoyed subjects that involved group work such as Software Engineering Project and Software Engineering Studio Project" (graduate of 2004).

"I have found that because of my academic background the most demanding tasks that I currently have - i.e. requirements elicitation, analysis, design and modelling of procedures and processes, resource acquisition and management of stakeholder expectations etc. are incredibly easy to do and do well ……. Hoorah for SWEBOK" (graduate of 2005).

Related Work on SE Curriculum

This section looks at a number of papers published in this area of SE curriculum and accreditation process since 1998. We commence with our paper (Ramakrishnan & Schmidt, 1998) where we discussed the educational requirements of a software engineering curriculum and showed how we arrived at a course structure that was suited to our local requirements at Monash University and with an eye on accreditation requirements from the Institution of Engineers, Australia (IE Aust.) as well as from the Australian Computer Society (ACS). The course structure and curriculum conformed to the CMU/SEI or ACM model curricula (Ford, 1990). In 1999, IEEE Software pub-

lished an article titled "Software Engineering Programs are not Computer Science Programs" by the eminent academic, Prof David Parnas (1999). Perhaps, his concern that engineering disciplines have a well documented body of knowledge for each of its established engineering disciplines to meet the guidelines of Engineering accreditation bodies and that a corresponding body of knowledge is missing for Computer Science has been addressed in Computing (Computer Science) Curricula (CC/CS curricula2001). At Monash, we can state that CS and SE units complement each other and coexist and cooperate in much the same way as science and engineering departments do. However as Prof. Parnas points out, science programs are subject to review mainly by the Universities for quality assurance purposes, and for science programs, there is no rigid accreditation requirement in North America by the Canadian Engineering Accreditation Board (CEAB) or by the Accreditation Board for Engineering and Technology (ABET) or by the Accreditation Board Engineers Australia (EA) in Australia. Prof Parnas states that the accreditation process for engineering programs is an effective way of raising the quality of the educational programs, and for raising the professional status of SE, SE should be treated as another specialty in engineering and accorded similar accreditation process. In 1999, Engel reported on Software Engineering Coordinating Committee's initial progress on accreditation guidelines for undergraduate software engineering programs. Prof. Bertrand Meyer presented his view on software engineering curriculum in an IEEE Computer article (Meyer, 2001) and discussed five complementary elements of a software curriculum as: principles, practice, applications, tools and mathematics. Meyer argues for a balance between the conceptual and operational aspects in a software curriculum. An IEEE Software article in 2002 (Saiedian, Bagert, & Mead, 2002) try to dispel myths and conceptions regarding SE programs. They report that instead of pigeonholing SE education into one model, we should foster stronger communication between various faculty groups, between universities and industry. They also state the importance of seeking industry feedback through forums such as industry liaison boards and try to incorporate changes into the programs in an appropriate fashion. In 2004, Kruchten argued to put engineering into software engineering in his paper at the Australian Software Engineering Conference (Krutchen, 2004). He argued that software engineering as a mature engineering discipline has a long way to go. He argued that software engineering is different from other engineering disciplines: software is not governed by physical laws such as in Physics, is easy to change, has low manufacturing costs, accommodates iterative development, and has no international borders. He stated that engineering in the software world (in bits) cannot compare exactly to engineering in the hardware world (of atoms). Kruchten wants the SE educators to promote a solution-focused engineering mindset in the SE programs in order to build software products that satisfy user requirements and not focus just on technology. He stated that computing curriculum for software engineering (ACM/IEEE Computing Curriculum, 2003) is a step in the right direction. van Vliet (van Vliet, 2005) reported on a few more myths in SE education and his message was that there is more to software engineering than engineering. In his view, social dimension is equally important and user interface design should not be relegated to a "related discipline" in both SWEBOK and SEEK. He would also like to see model-driven development (MDA) and service-oriented architectures (SOA) in SWEBOK and SEEK.

Summary

In this article, we have discussed how we have developed a SE curriculum and managed an evolution process for our undergraduate Bachelor of Software Engineering program. We have described the quality characteristics of the curriculum structure and content, teaching & learning environment, and quality assurance framework by focusing on five aspects. These five aspects are: i) using reference frameworks to check against our course structure and content, ii) building a quality teaching and learning environment, iii) engaging with external bodies such as industry

liaison boards, internal committees such as the school, faculty and university undergraduate education committees and with industry clients for capstone projects as part of the quality assurance process in assisting to deliver professional engineering programs, iv) the accreditation process with Engineers Australia (EA) and with Australian Computer Society (ACS), and v) CALT & CHEQ involvement with Faculty of IT during AUQA audit and for improving student learning experience. We have thus shown evidence of a continually improving, aligned quality product and process in place for continued accreditation of the Monash Software Engineering Program by Engineers Australia.

References

ACM/IEEE Computing Curriculum. (2003) Software Engineering. Joint IEEE Computer Society/ACM Task Force on Computing Curriculum, July 2003.

Arora, S., & Chazelle, B. (2005). Is the thrill gone? *Communications of the ACM, 48*(8), 31-33.

Barrie, S., Ginns, P., & Prosser, M. (2005). Early impact and outcomes of an institutionally aligned, student focused learning perspective on teaching quality assurance. *Assessment and Evaluation in Higher Education, 30*(6), 641-656.

Barrie, S.C., & Prosser, M. (2003). An aligned, evidence-based approach to quality assurance for teaching and learning. Presented at the *Australian Universities Quality Forum,* Adelaide, June 13-15

Bourque, P. & Dupuis, R. (Eds.). (2001). *A guide to the software engineering body of knowledge* (trial edition). Los Alamitos, CA: IEEE CS Press.

Engel, G.L. (1999). Program criteria for software engineering accreditation program. *IEEE Software, 16*(6), 31-34.

Ford, G. (1990). *SEI report on undergraduate software engineering education.* Technical-Report CMU/SEI-90-TR-3. Software Engineering Institute, Carnegie Mellon University.

Krutchen, P. (2004). Putting the "engineering" into software engineering. Presented at the *Australian Software Engineering Conference* (ASWEC'04), IEEE Computer Society, pp. 2-8.

Meyer, B. (2001). Software engineering in the academy. *IEEE Computer, 34*(5), 28-35.

Narasimhan, L. V. (2006). A second opinion on the current state of affairs in computer science education – An Australian perspective. *Issues in Informing Science and Technology, 3,* 445-458. Available at http://informingscience.org/proceedings/InSITE2006/IISITNara114.pdf

Parnas, D.L. (1999). Software engineering programs are not computer science programs. *IEEE Software, 16*(6), 19-30.

Ramakrishnan, S. (2003). MUSE studio lab and innovative software engineering capstone project experience. Proceedings of the *8th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE2003)* (pp. 21-25), Thessoloniki, Greece, June 2003, ACM Publication.

Ramakrishnan, S. & Cambrell, A. (2002). An in-forming web-based environment for a bachelor of software engineering degree – DoIT. In E Cohen and E Boyd (Eds.), Proceedings of the *Informing Science and IT Education Conference (IS 2002)* (pp. 1291-1299), Cork, Ireland, June 19--21, 2002 Available at http://proceedings.informingscience.org/IS2002Proceedings/papers/ramak053infor.pdf

Ramakrishnan, S., & Schmidt, H. (1998). A study of software engineering education requirements within a semiotic framework. Proceedings of the *Software Engineering: Education & Practice (SE:E&P 98)* (pp.213-220), Dunedin, New Zealand, IEEE Computer Society Press.

Saiedian, H., Bagert, D., & Mead, N.R. (2002). Software engineering programs: Dispelling the myths and misconceptions. *IEEE Software, 19*(5), 35-41.

van Vliet, H. (2005). Some myths of software engineering education. Presented at *ICSE'05*, St. Louis, Missouri, USA, ACM Publication.

Biography

Sita Ramakrishnan is a senior academic in the Clayton School of IT, Faculty of IT, Monash University, Australia. She holds a PhD in Validating Interoperable Distributed Software and Systems. She has active research interests in modeling and validation of distributed software components, component-based and service-oriented architectures and testing, web technologies in education, teaching and learning. She has published refereed papers in International Journals & Conferences on software engineering on quality, reuse, software metrics, evaluation, testing and SE Education. She has been an organizing and Program committee member of a number of International conferences and reviewed a number of conference and journal articles. She has played a leading role in the curriculum development of Bachelor of Software Engineering course at Monash University. She is Director of Software Engineering degree program in the Faculty. She managed the process of formal accreditation of the software engineering course program by the Institution of Engineers of Australia and Australian Computer Society. Dr Sita Ramakrishnan is a member of IEEE.