Educating professional software engineers: pathways and progress in the Australian experience

Rick Duley
Edith Cowan University

David Veal
Edith Cowan University

Stanislaw P. Maj
Edith Cowan University

Educating Professional Software Engineers: Pathways and progress in the Australian experience

Rick Duley D Veal S P Maj
{r.duley, d.veal, p.maj} @cowan.edu.au
Edith Cowan University, Perth, Western Australia

Abstract

Australia has seized the international initiative in the recognition of Software Engineers as professionals. Of the 37 universities in Australia offering undergraduate courses in computing, eleven offer courses in Software Engineering which are accredited by the Institute of Engineers, Australia and which may lead the graduate to membership of the Institute. In this way, the Institute has plausible claim to being the first national professional engineering body in the world to have accredited four-year undergraduate software engineering degrees as professional qualifications.

This paper traces the development of the relationship between the Institute of Engineers and the computing industry and looks at the changes this relationship has wrought in the content and emphasis of tertiary software engineering education.

1 Pathways

1.1 Beginnings

For more than fifteen years, the Institute of Engineers, Australia, (IEAust) has shown interest in the evolution of the software industry. IEAust's Working Party on Software Engineering reported in May 1985 with the conclusion that:

> "*...at that time, 'software engineering' was more correctly characterised as a specialist activity within the computer field than as a new engineering discipline.*" [4, p.3]

Only eleven years later, in 1996, the University of Melbourne received IEAust accreditation for its baccalaureate of Engineering in Software Engineering (the first in Australia to do so). By 1999 eleven of the 37 universities in Australia offering undergraduate computing degrees were offering software engineering degrees under the auspices of IEAust.

> *"A further fifteen or so accredited professional engineering degree programs have sufficient software content and coverage of computing topics to prepare graduates for careers in software engineering, provided that they select the appropriate alternatives. Many of these courses are accredited by the Australian Computer Society as well as IEAust."* [4, p.5]

In this way, IEAust has plausible claim to being the first national professional engineering body in the world to have accredited four-year undergraduate software engineering degrees as fully professional qualifications. (In the UK, graduates of accredited courses may, through the British Computer Society, become Chartered Engineers [8, p.264] but the Department of Trade and Industry has expressed some opposition to their being registered as engineers [9, p.25])

1.1.1 Enter the ACS

Pre-existing Australian state-based computer societies merged in 1966 to form the Australian Computer Society (ACS) with a mission to advance professional excellence in information technology [1]. After some years of enquiry, the ACS launched, in 1996, a concerted effort to gain the recognition of the Australian Council of Professions (the governing body of the governing bodies of professional societies in Australia) as a fully-fledged professional organisation. IEAust was approached by the ACS to assist in this push, and, after a period of restructuring and reformation during 1996-98, the ACS received recognition in 1999. Currently, IEAust and the ACS are working towards a formal agreement which will result in a Joint Board on Software Engineering which will have oversight of accreditation standards and procedures, examination and registration of the Professional Software Engineer (PSE).

1.2 Educating the PSE

Traditionally, undergraduate computer courses in Australia have fallen under one of three headings: Computer Science, Information Systems (or Information Technology) and Computer Systems Engineering. Software engineering, it is well known, fits none of these categories.

> *"...it became increasingly apparent to us and to others that the goals of software engineering and computer science, while similar, are distinct."* [7, p.288]

This difference is echoed by IEAust:

> *"It is generally accepted that computer science is the predominant underlying discipline on which software engineering is based; however, software engineering has different goals to computer science (viz. learning in order to build, rather than building in order to learn).* [4, p.2]

Furthermore, it is long recognised that the education of practitioners in the emerging field of software engineering would require a different approach to that traditionally applied to computer science.

> *"Since software engineers work in a product-oriented field, they require a different kind of education than that typically provided by research-oriented computer science departments."* [2, p.595]

In the first place, undergraduate science courses in Australia are of three years duration. Whether or not the bases of software engineering could be transmitted within that timespan was one of the first questions which begged an answer especially when the normal extent of an undergraduate engineering course is four years.

> *"The cruel face of reality commands us to implement such an integrated programme over* [three years], *and so we have to make many compromises to the ideal. We believe it is possible to accomplish these compromises so that the result is clearly a degree with an engineering rather than a science emphasis. ...Thus our curriculum is focussed on educating software engineers through a mixture of Computer Science fundamentals, controlled Software Engineering practice in project units, and uncontrolled commercial experience through our cooperative programme (which incidentally adds an extra year to the degree, which consists of three academic years and one year of cooperative 'industry based learning'."* [3, pp.106-7]

This extension of the course, common among the eleven IEAust accredited courses, still leaves the academic duration of the course short in comparison to the normal engineering undergraduate course. (For example, the Mechanical Engineering undergraduate course at the University of Western Australia is of four years duration including only twelve weeks of practical work experience.)

2 Progress

Juggling the concurrent requirements of duration and content has required a reshaping of the SE curricula. It is this

Table 1 : Courses Used for Comparison

Courses Used for Comparison		
University	**CS**	**SE**
0	BIT	BSE
1	BSc(CS)	BE(SE)
2	BIT	BE(SE)
3	BCS	BCSE(SE)
4	BIS	BE(SE)
5	BCS	BSE
6	BSc(CS)	BE(SE)
7	BCS	BE(S)
8	BApp.Sci.(CS)	BApp.Sci.(SE)
9	BIT	BSE
10	BSc(CS)	BE(SE)

Table 2 : Knowledge Area Comparison

Knowledge Area Comparison			
Topic	**CC'91**	**CC'01**	**IEAust**
Algorithms	✓	✓	✓
Architecture		✓	
Artificial Intelligence	✓	✓	✓
Computational Science		✓	
Consumer Computing			✓
Data Structures	✓	✓	✓
Database & Information Retrieval	✓		
Graphics, Visualization and Multimedia		✓	✓
Human-Computer Communication	✓	✓	✓
Information Management		✓	
Introduction to a Programming Language	✓	✓	
Net-centric Computing		✓	
Numerical & Symbolic Computation	✓		
Operating Systems	✓	✓	✓
Programming Languages	✓	✓	✓
Requirements Analysis			✓
Robotics	✓		
SD Process Modeling			✓
Security and Encryption			✓
Social, Ethical and Professional Issues	✓	✓	
Software Engineering Tools			✓
Software Methodology and Engineering	✓	✓	
Software Metrics			✓
Software Reliability			✓
Software Testing			✓
Spatial Systems			✓

curricular restructuring which attracted the attention of the authors who instituted a survey of the universities involved in the education of potential PSEs.

2.1 The Survey

Curricula were gathered from all eleven universities offering IEAust accredited Software Engineering (SE) undergraduate degrees.

Note: for the purposes of this paper the universities studied are only referred to by number (0 .. 10) and no particular positional inference should be taken.

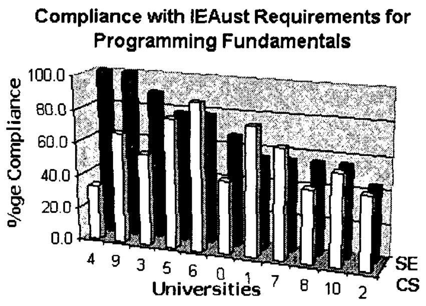

Figure 1 : IEAust Compliance - Programming Fundamentals

As well as the curriculum for the SE degree, the curriculum for another, related, undergraduate computing degree was obtained from each university in the study. Selected courses for each university are shown in Table 1. To avoid exacerbating any divergence between the degrees offered by any one university, the authors selected for each non-SE (loosely termed Computer Science (CS)) degree an SE major where available.

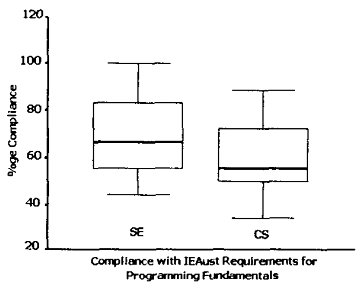

Figure 2 : IEAust Compliance - Programming Fundamentals - Boxplot

Three major curricular outlines were chosen for purposes of comparison:

1. Curriculum 2001 (CC'01): details from the March 6, 2000 draft version [6]
2. Curriculum 1991 (CC'91): details from the summary published in Communications of the ACM, June 1991 [5]
3. IEAust : Sample course outline details taken from a discussion paper *"Software Engineering as a Professional Engineering Discipline"* published in March 1999 [4]

Table 3 : Introductory Languages

Introductory Languages		
University	**CS**	**SE**
0	Eiffel	Eiffel
1	Java	Java
2	Java	Java
3	C++	C++
4	—	Haskell
5	—	C++
6	Java	Java
7	Java	Java
8	Java	C
9	Java	Java
10	Eiffel	Eiffel

Table 2 lists for comparison the Subject (or Knowledge) Areas considered in each document.. (They show the increased emphasis placed on SE by IEAust in that the Knowledge Areas shown in shaded rows could well be placed under the one heading *'Software Methodology and Engineering'*. Conversely, Information Management and Net-centric Computing are omitted from the IEAust listing.)

From each pair of university curricula, details were entered in a spreadsheet which permitted direct comparison between the individual university curricula and each of the three general curricular outlines — CC'91, CC'01 and IEAust.

2.1.1 Limitations

Understanding the results of the survey requires cognisance of some limitations of precision imposed by the nature of this initial survey.

2.1.1.1 Core Units

In each curriculum, only core units were considered. Universities vary in the degree of latitude allowed students in the matter of electives. It is reasonable to assume that all of the universities provide educational coverage of each of the subject areas through a combination of core and elective units. However, the authors elected to keep to the units a graduate must have taken rather than to speculate on the units a graduate might have taken.

2.1.1.2 Course Description

Unit content, in each case, was judged solely on the Course Description as given at the web-site (or University Handbook). It is accepted that this might not necessarily reflect the totality of the subject matter dealt with in the unit but the authors could, in this study, only operate on the information made available to a prospective student.

2.1.1.3 Terminology

Terminological differences presented arguably the most difficult aspect of the survey from the authors' point of view.

For example, all three curricular outlines were written with procedural high-level languages in mind. Table 3 shows the specified introductory programming languages used in each of the courses studied.

Descriptive terminology for Java does not parallel that for procedural languages, so a difficulty arises in defining the point at which *'Abstract Data Types'* might been covered in the Course Description.

As a further example, a decision had to be made as to whether the sentence *"The subject is dedicated to the introduction of object-oriented programming principles, using the Java programming language"* covers the topic *"Fundamental Programming Constructs"*.

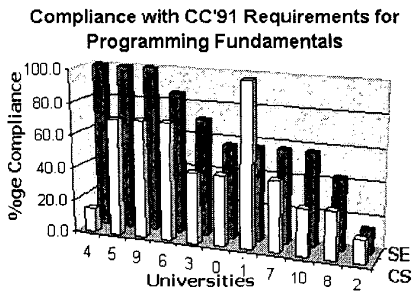

Figure 3 : CC'91 Compliance - Programming Fundamentals

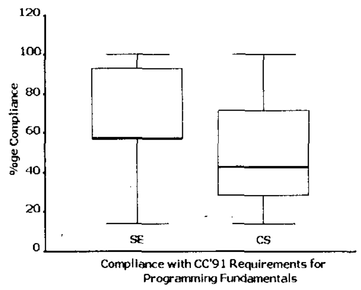

Figure 4 : CC'91 Compliance - Programming Fundamentals - Boxplot

2.1.1.4 Judgment

As described in the previous section, much of the interpretation of the raw data had to be based on purely subjective judgment. In accepting this problem, every attempt was made to be consistent throughout even if not pedantically and precisely correct. Given this and the other shortcomings, the results of the survey do not lend themselves to rigorous statistical analysis. However, the authors contend that the results of graphical analysis are valid and give an informative picture of the current situation.

2.2 The Results

2.2.1 Programming Fundamentals

While it is true that SE is not Programming (and that Programming is not SE) it is true that the implementation of a design into code is a fundamental and crucial part of the software lifecycle. For this reason, we might expect that in a degree course which focuses on SE rather than the broader spectrum of CS the curriculum would show an increased emphasis on the programming fundamentals.

Consider Figure 1. Values for the SE degrees are (typically) shown in the rear row (darker columns). In all but three cases — 6, 1 and 7 — the values for SE are higher, some — especially 4 — substantially so. Boxplotting the figures (Figure 2) clarifies the shift in emphasis.

Interestingly, when the compliances are graphed against the requirements for Curriculum 1991, the results are even more clear-cut indicating a distinct change in emphasis as the courses

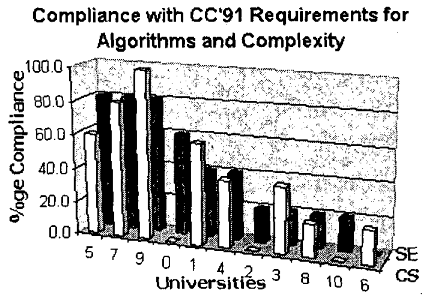

Figure 5 : CC'91 Compliance - Algorithms and Complexity

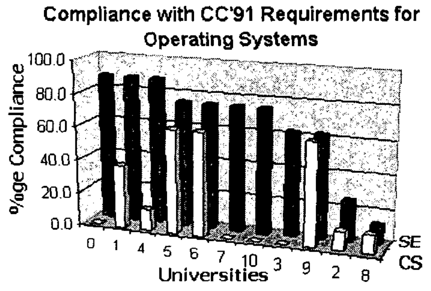

Figure 6 : CC'91 Compliance - Operating Systems

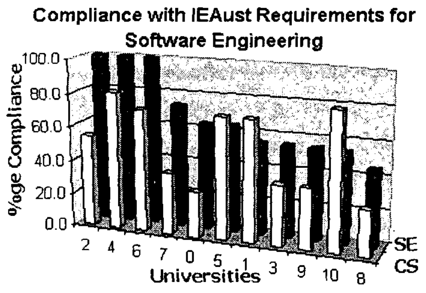

Figure 7 : IEAust Compliance - Software Engineering

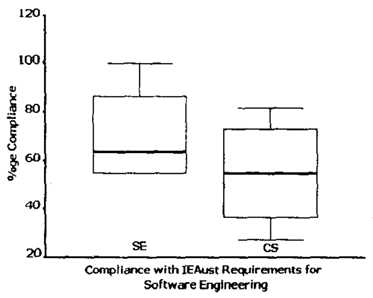

Figure 8 : IEAust Compliance - Software Engineering - Boxplot

focus on Software Engineering (Figure 3 and Figure 4). University 1's result in Figure 3 is an aberration which may be explained by the dropping of one CS unit from the core in favour of an Engineering unit.

2.2.2 Algorithms and Complexity

In alignment with the (CC'01) Knowledge Unit on Algorithms and Complexity, the IEAust specifications deal only with complexity and computability and the resultant graphical information is simplistic and unenlightening. However, when the raw data is graphed against the requirements for Curriculum 1991 (Figure 5) it is evident again that some change has occurred. While the coverage of the Knowledge Area given by courses 9, 1 and 3 has dropped, courses 0, 2 and 10 now address the topic — in fact, only 6 does not. While the result might not be as profound as that for Programming Fundamentals, it does show that the change in emphasis has an effect.

2.2.3 Operating Systems

A similar situation applies when considering the subject of Operating Systems but, again, when the raw data is compared to the requirements for Curriculum 1991 (Figure 6) the change in emphasis quite clear. This came as something of a surprise to the authors who expected that Operating Systems, being of a technical nature were more likely to be emphasised in a course of Computer Science than one on Software Engineering. As can be seen, substantial changes in the emphasis on the subject have occurred and now all the universities concerned now deal with the subject and most of them quite thoroughly.

2.2.4 Software Engineering

It was only to be expected that courses seeking accreditation from IEAust would reflect the increased emphasis on Software Engineering revealed in Table 2. This expectation is realised in Figure 7 and Figure 8.

As mentioned above, where applicable the authors selected CS courses with SE majors, and this may be taken to explain the higher than might be expected compliance of the CS courses as shown in the graph. However, despite this, an overall change in emphasis is clearly visible.

2.3 The Conclusions

IEAust, in initiating the recognition of Software Engineering as a distinct and fully professional engineering discipline, has had a profound effect on relevant tertiary curricula. In every area of knowledge investigated by the authors, SE course content had changed significantly from that previously offered for traditional computer science.

Despite the limitations of the survey conducted by the authors, graphs of data obtained from IEAust accredited university curricula show the core of each SE syllabus placing greater emphasis on the requirements of SE than was previously the case with a CS syllabus. In the view of the authors, this confirms the distinct and individual nature of SE as a discipline in its own right and demonstrates the willingness of tertiary education institutions to respond to the needs of that discipline. Further work is proposed to address the limitations of this preliminary survey.

References

[1] Australian Computer Society. About the Australian Computer Society [Web Page]. [2000].

[2] N.E. Gibbs, The SEI Education Program: The Challenge of Teaching Future Software Engineers *Communications of the ACM*, vol. 32, pp. 594-605, May, 1989.

[3] D.D. Grant and R. Smith, Undergraduate Software Engineering - An Innovative Degree at Swinburne *The Australian Computer Journal*, vol. 24, pp. 106-113, Aug, 1992.

[4] IEAust Working Group on Software Engineering, Software Engineering as a Professional Engineering Discipline: Discussion paper Mar, 1999. (unpublished).

[5] IEEE-CS/ACM Joint Curriculum Task Force, Computing Curricula 1991: A Summary *Communications of the ACM*, vol. 34, pp. 69-84, Jun, 1991.

[6] IEEE-CS/ACM Joint Task Force on Computing Curricula, Computing Curricula 2001 (draft) Mar 6, 2000. IEEE-CS/ACM. [on line] : `http://www.computer.org/education/cc2001/`.

[7] M.J. Lutz and J.F. Naveda, The Road Less Traveled: A Baccalaureate Degree in Software Engineering *Proceedings of ACM/SIGCSE 97*, pp. 287-291, 1997.

[8] L.R. Neal and A.D. Irons, Integrating Professionalism into Undergraduate Degree Courses in Computing *ITiCSE '98. Proceedings of the 6th annual conference on the teaching of computing/3rd annual conference on integrating technology into computer science education*, pp. 264-267, 1998.

[9] Dr. A. Underwood, Position Paper: Certification of Software Engineers (unpublished). Perth, WA.