Conforming Curricula for Software Engineers: Observations from the Australian experience

Rick Duley, S P Maj, D Veal
Edith Cowan University, Perth, Western Australia

Abstract

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 Australia has seized the initiative in the recognition of Software Engineers as professionals and 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. 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. 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. Juggling the concurrent requirements of duration and content has required a reshaping of curricula. It is this curricular restructuring which attracted the attention of the authors who instituted a survey of the eleven universities involved in the education of potential professional Software Engineers which has produced graphical evidence confirming the distinct and individual nature of SE as a discipline and demonstrated the willingness of tertiary education institutions to respond to the needs of that discipline. This paper reports on moves in Australia towards the recognition of software engineering as a bona fide profession in its own right and presents the results of the survey showing the changes in curricular definition which have taken place as universities move to support the new discipline.

1. Background

"Software Engineering" (SE), as a term describing a distinct engineering discipline, was more of an aspiration than a fact when it was first used at NATO conferences in 1968 and 1969. Conference organisers were challenging the belief that software development was essentially art and inspired creativity rather than more traditionally based on precision, discipline and attention to detail[1]. Today its product controls automobiles and aircraft, watches and washing machines, rockets and robots. As that came to be the case it became evident that such a practice could no longer remain an heuristic process and Software Development (SD) has been a subject of study

for many years, especially through the British Computer Society (BCS, founded 1957), the US Carnegie-Mellon University Software Engineering Institute (CMU-SEI, founded 1984), and the European Software Institute (ESI, founded 1993). Software process improvement has been shown to produce considerable return on investment — for example, Raytheon's Equipment Division showed a two-fold increase in productivity and $7.70 return on every dollar invested over a five-year period[2]— yet the application of software process improvement programs and the adoption of software best practices (SBP) are yet to become universal. In 1997 the ESI SBP survey found that even under the conditions of extreme mission-criticality (Aircraft and Spacecraft) Best Practice adoption only reached 60% alongside a mean adoption level of 51%[3]. (In conformance with the ESI reports: "We define *"best practice"* as a management practice that is widely recognised as good and that is recommended by most practitioners and experts in the field.")

Likewise, traditional professional engineering control bodies have, for many years shown interest in the evolution of the software industry culminating (in the USA) with the action in June of 1998 of the Texas Board of Professional Engineers (TPBA) in adopting SE as a distinct discipline under which engineering licences can be issued[4]. Candidates with a TPBA-accredited engineering degree and 12 years experience, a non-accredited degree and 16 years experience or a postgraduate degree and six years experience might apply for exemption from the Fundamentals of Engineering Examination and/or the Principles of Practice Examination to be established by a joint working group of the Association of Computing Machinery (ACM) and the Institute of Electronic and Electrical Engineers – Computer Society (IEEE-CS) and thus register immediately. (It is worth noting, however, that State Boards had been registering Engineers who predominately worked with software for more than 40 years[5] but without calling them Software Engineers.)

In Australia, The Institute of Engineers, Australia, (IEAust) had also been following the development of the discipline closely, noting in May of 1985 that it was more correctly characterised as a specialist activity within the computer field than as a new engineering discipline. 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 (ACS) as well as IEAust[6]. 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[7] but the Department of Trade and Industry has expressed some opposition to their being registered as engineers[8]) 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).

Computer Science (CS) is distinct from SE — the former building in order to learn and the latter learning in order to build[9]. 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 — working in a product-oriented field, SEs require a different kind of education than that typically provided by research-oriented computer science departments[10].

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)

Undergraduate science courses in Australia are of three years duration and 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. Creation of a three-year academic SE program required many compromises to the ideal. Curricula have been developed which focussed on educating software engineers through a mixture of Computer Science fundamentals, controlled Software Engineering practice in project units, and uncontrolled commercial experience through a cooperative program (which incidentally adds an extra year to the degree, which consists of three academic years and one year of industry based learning[11] and this extension of the course to four years by default 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.) This juggling of the concurrent requirements of duration and content attracted the attention of the authors who instituted a survey of the eleven accredited universities which are, for the purposes of this paper referred to only by number (0 .. 10) and no particular positional inference should be taken.

2. The Survey

If curricula had changed to meet the needs of the new discipline, then how had they changed? What had been added, what was still there, what had been left out? These were the questions the authors set out to answer. Our methodology was simple — two curricula were gathered from each of the eleven universities, one being of the SE degree itself and the other of a corresponding CS degree (ref : Table 1) and details from these courses were entered into a spreadsheet for comparison with three major curricular outlines. These outlines were:

1. Curriculum 2001 (CC'01): details from the March 6, 2000, draft version[12]
2. Curriculum 1991 (CC'91): details from the summary published in the Communications of the ACM[13]
3. IEAust: Sample course outline details[6].

These revealed a variation in the coverage of the knowledge areas specified in each document as demonstrated in Table 2 including the increased emphasis placed on SE by IEAust in that the Knowledge Areas itemised 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.

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			✓

3. Survey Limitations

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 without procedural high-level languages specified. In the survey, two universities used Eiffel for both courses, one C++ for both courses, four Java for both courses and the rest varied. Obviously, 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"*.

Furthermore, in each curriculum studied, only core units were considered. Universities vary in the degree of latitude allowed students in the matter of electives and 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 <u>must</u> have taken rather than to speculate on the units a graduate <u>might</u> have taken. Also, 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 only operate on the information made available to a prospective student.

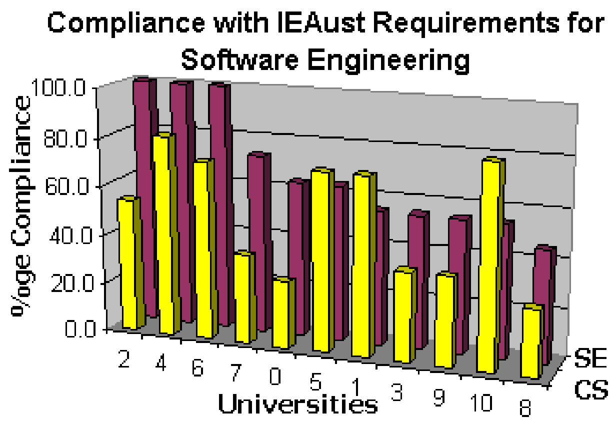

Figure 1 : IEAust Compliance for Software Engineering

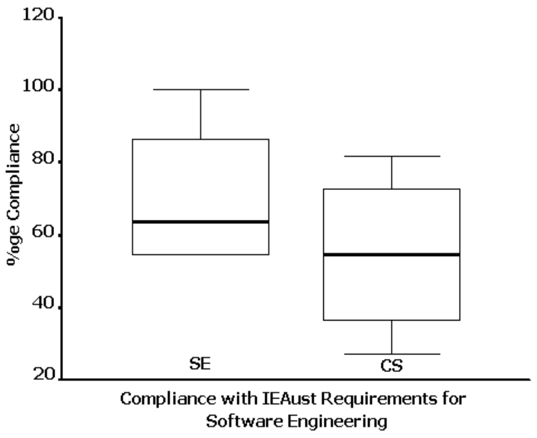

Figure 2 : IEAust Compliance for Software Engineering

Because of the above factors, interpretation of the data contained in the curriculum guides had to be purely subjective. Every attempt was made to remain consistent in the examination of each knowledge area but because of these shortcomings the results do not lend themselves to rigorous statistical analysts. Graphical analysis, however, produced results which the authors believe to be valid and to give an informative picture of the current situation.

4. The Results

It was only to be expected the curricula seeking IEAust accreditation would reflect the increased emphasis on SE shown in Table 2 and, clearly, Figure 1 and Figure 2 confirm this change. (In the bar charts the SE courses are represented in the darker, rearmost row) Variations discovered, however, were not always what the authors expected. For example, it might be expected that the CS curricula would place much more emphasis on Operating Systems and yet Figure 3 which shows a far greater compliance on the part of the SE curriculum!

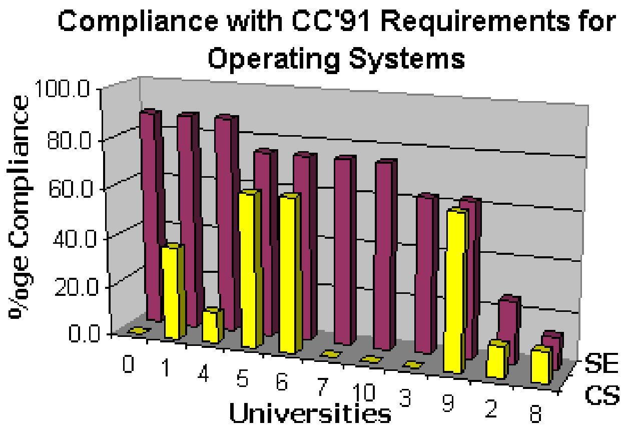

Figure 3 : CC'91 Compliance for Operating Systems

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. yet consider Figure 4 and Figure 5. While there is a substantial shift in emphasis for the majority of the SE courses the best and worst for both types of courses stay the same.

For another example, Algorithms and Complexity (A&C) are a cornerstone of

Computer Science and it might be expected that CS courses would pay these topics particular attention yet once again we see in Figure 6 that SE courses generally have a higher A&C content.

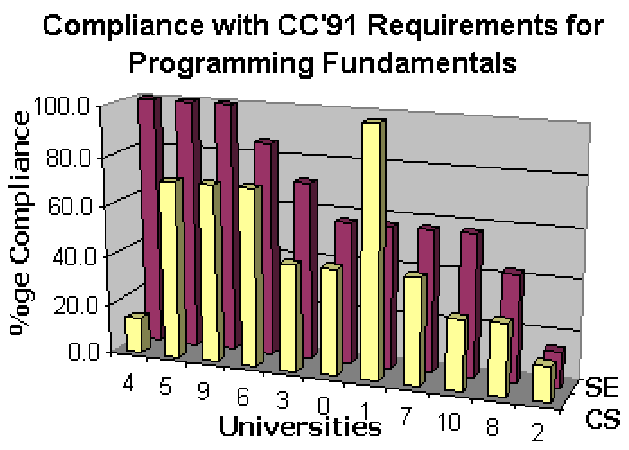

Figure 4 : CC'91 Compliance for Programming Fundamentals

5. More Questions than Answers

Curriculum 2001 was the source of the Knowledge Area (KA) classifications investigated during the survey and in each one this general trend of greater SE compliance would appear. This raised the question as to how it could be that SE could be consistently so much closer to the general curriculum specification? Greater SE compliance was to be expected in comparison to the IEAust curriculum, but it remained consistent regardless of the curriculum used for comparison. How could that be explained?

In search of a reply, the authors chose one KA which only appears in CC'01 — Intelligent Systems — and applied the same methodology on the premise that *"legacy"* CS courses would not appear at all. Our theory was that Intelligent Systems is a relatively new field at undergraduate level and CS courses tend to be well established and undergo relatively little change. Then we looked at the most regular topic of all — mathematics — expecting more even results. This is a subject which underpins CS (and Computer Engineering) and which remains important — albeit at a slightly lower level — in the development of software. The level of treatment of the subject in the various curricula should, therefore, be similar. Both studies produced surprising results (Figure 7).

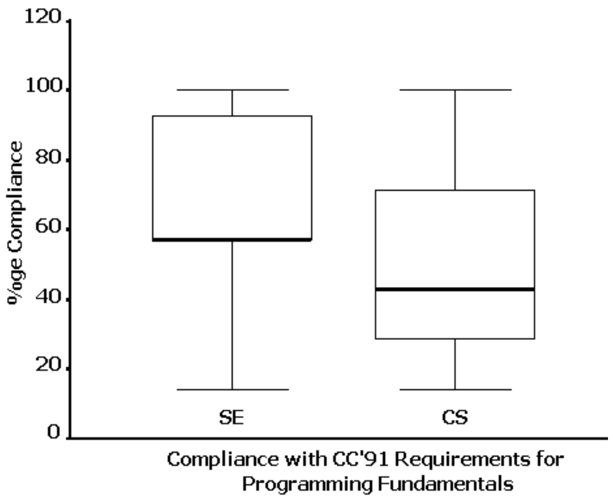

Figure 5 : CC'91 Compliance for Programming Fundamentals

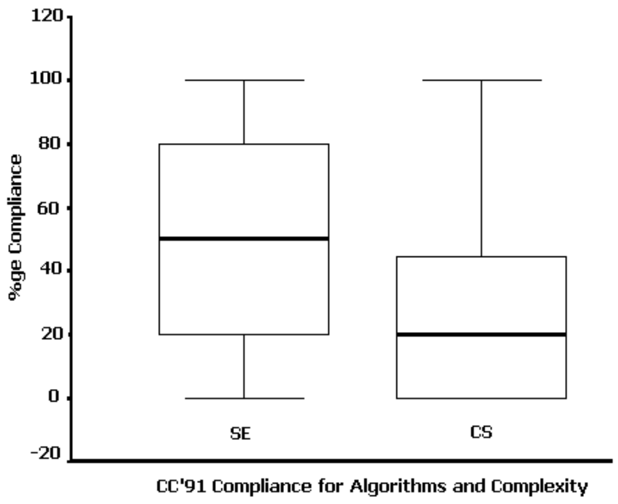

Figure 6 : CC'91 Compliance for Algorithms and Complexity

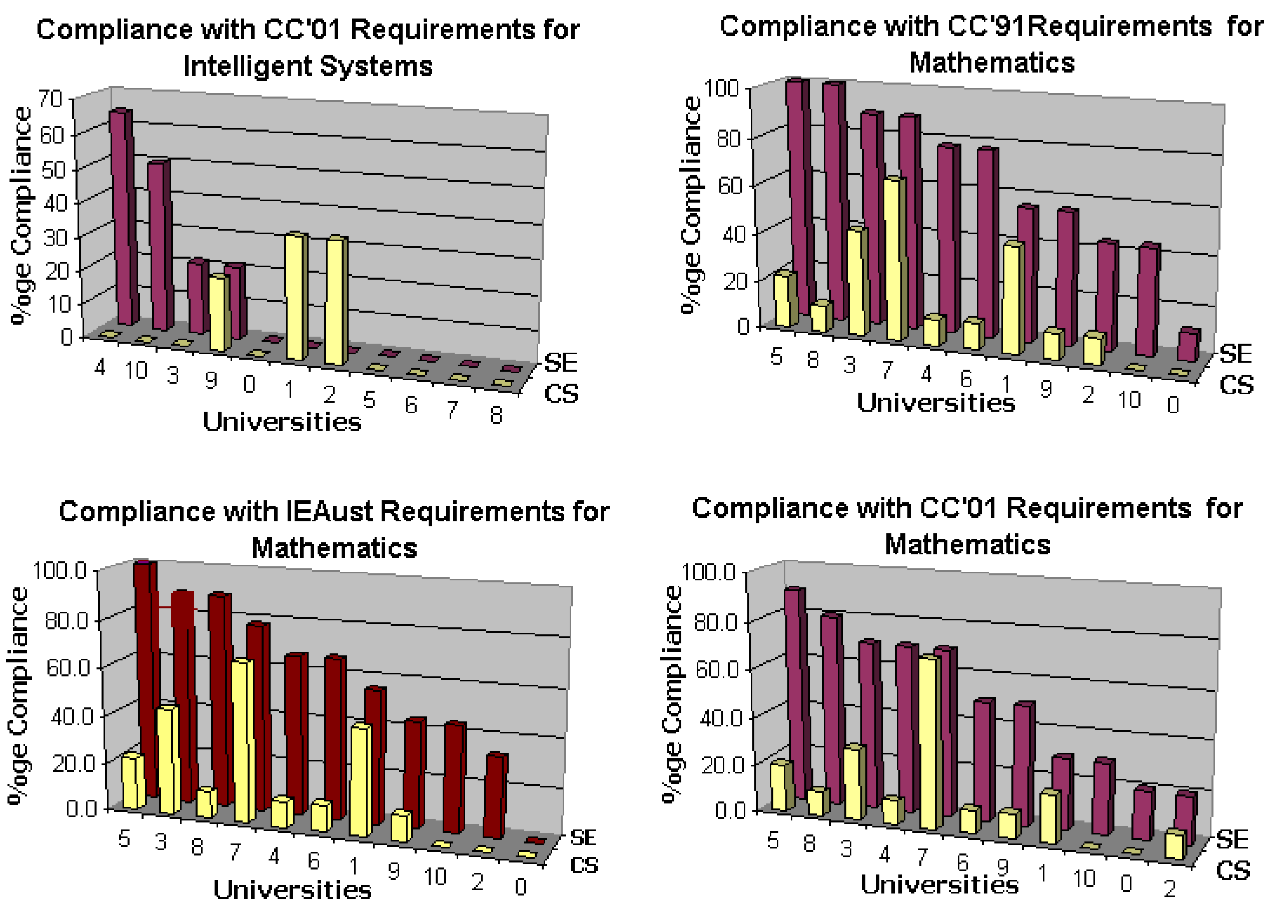

Figure 7 : Further Results to Puzzle

6. Some Possible Answers and Another Question

Edith Cowan University is not IEAust accredited so following considerable internal discussion of this curious result, the authors approached colleagues at institutions which are accredited and assembled a pool of opinions including the following:

1. an accredited SE degree is a year longer allowing more time to cover the subjects — although this explanation is countered by the fact that most of the curricula require the extra year to be spent in industrial experience
2. while accredited courses are usually 50% traditional engineering, the remaining time is generally devoted more specifically to SE and can leave out matters commonly included in a CS degree — although this explanation is countered by the finding that the SE curricula perform well in comparison with the standard CS curricula
3. SE students are doing an engineering degree and the mathematical requirements may be greater and at a higher level and catered for on the engineering side leaving hands free on the SE side for computer matters — and which might also explain the exceptional

performance of the SE curricula in the mathematics comparisons in Figure 7
4. in seeking accreditation, designers of IEAust SE courses are more specific in their description of the course — after all, we are, here, talking about the course not the students — it must be remembered, however, that non-IEAust universities in Australia usually seek accreditation with the ACS giving the same incentive for specificity.
5. Intake requirements in mathematics tend to be higher for an engineering degree than for computing degrees. Therefore the students in an IEAust accredited course would start with an advantage in mathematics.
6. IEAust accredited courses, as start-up courses may tend to have higher staff **:** student ratios.

The authors wish here to acknowledge with gratitude the generosity of Jocelyn Armarego and Geoff Roy (Murdoch University, Perth), Doug Grant (Swinburne Institute of Technology, Melbourne) and Dale Stanborough (Royal Melbourne Institute of Technology, Melbourne) for taking time to consider and answer our query.

None of the answers above appears satisfactory except as an aspect of an overall solution, and the authors look forward to further work in finding out what really has happened to Australian computer curricula.

7. Conclusions

Considerable efforts are being made in Australia to define and refine the concept of the Industrial Software Engineer. With the active and enthusiastic support of the Institute of Engineers, Australia, and the Australian Computer Society undergraduate curricula are being created and taught within a classical engineering environment which holds the promise of a disciplined approach to the development of software for an industrial environment. However, a recent survey of the SD industry in Western Australia showed that less than 5% of the respondents were involved in the 'hard' industries (manufacturing, utilities etc.) and figures for more industrially developed Europe the figure is less that 15% [3]. Therefore, while the authors sincerely applaud the progress being made in the SE field we feel that it might also be important to make similar steps of definition and refinement in the areas of Information Technology and more general software development. There remain many more questions to be answered.

Bibliography

1. M.J. Lutz and J.F. Naveda, The Road Less Travelled: A Baccalaureate Degree in Software Engineering *Proceedings of ACM/SIGCSE 97*, pp. 287-291, 1997.
2. R. Dion, Process Improvement and the Corporate Balance Sheet *IEEE Software*, pp. 28-35, Jul, 1993.
3. European Software Institute, 1997. European Software Institute. Bilbao (ESP).
4. Texas Board of Professional Engineers. (98) Board Establishes Software Engineering Discipline [Web Page]. http://www.tpbe.state.tx.us/sofupdt.htm [June 1, 2000].

5. J. Charles, A License to Code *IEEE Software*, pp. 119-121, Sep, 1998-Oct 31, 1998.
6. IEAust Working Group on Software Engineering, Software Engineering as a Professional Engineering Discipline: Discussion paper Mar, 1999. (unpublished).
7. 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.
8. Dr.A. Underwood, Position Paper: Certification of Software Engineers (unpublished). Perth, WA.
9. F.P. Brooks, The Computer Scientist as a Toolsmith II *Communications of the ACM*, 39, (3)pp. 61-68, Mar, 1996.
10. N.E. Gibbs, The SEI Education Program: The Challenge of Teaching Future Software Engineers *Communications of the ACM*, 32, (5)pp. 594-605, May, 1989.
11. D.D. Grant and R. Smith, Undergraduate Software Engineering - An Innovative Degree at Swinburne *The Australian Computer Journal*, 24, (3)pp. 106-113, Aug, 1992.
12. 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/.
13. IEEE-CS/ACM Joint Curriculum Task Force, Computing Curricula 1991: A Summary *Communications of the ACM*, 34, (6)pp. 69-84, Jun, 1991.

Biographical Information

RICK DULEY graduated with First Class Honors in Computer Science in 1996. He is currently a Doctoral Research Student at Edith Cowan University working in the field of Software Engineering Education. Coming from a Heavy Industry and Mining background he has a special interest in things industrial and in the application of engineering principles to software construction. r.duley@ecu.edu.au

Dr S P MAJ is a recognized authority in the field of industrial and scientific information systems integration and management. He is the author of a text book, *'The Use of Computers in Laboratory Automation'*, which was commissioned by the Royal Society of Chemistry (UK). His first book, *'Language Independent Design Methodology - an introduction'* , was commissioned by the National Computing Centre (NCC). Dr S P Maj has organized, chaired and been invited to speak at many international conferences at the highest level. He has also served on many national and international committees and was on the editorial board of two international journals concerned with the advancement of science and technology. As Deputy Chairman and Treasurer of the *Institute of Instrumentation and Control Australia (IICA*) educational sub-committee he was responsible for successfully designing, in less than two years a new, practical degree in *Instrumentation and Control* to meet the needs of the process industries. This is the first degree of its kind in Australia with the first intake in 1996. It should be recognized that this was a major industry driven initiative. p.maj@ecu.edu.au

DAVID VEAL received his honors degree in Theoretical Physics from the University of York in England. After completing a Grad.Dip.Ed. from the University of Keele he lectured in Physics at South Devon College UK for 10 years. He now lives in Western Australia where he has taught Computing and Physics at high school level. He is a Doctoral student in Computing Science at ECU in Perth, Western Australia. He is investigating competency-based techniques in Computing Science as well as the modeling of computers to aid student understanding. d.veal@ecu.edu.au