

What Should Graduating Software Engineers Be Able To Do?

A. J. Cowling

*Department of Computer Science,
University of Sheffield,
Regent Court, Portobello Street,
Sheffield, S1 4DP, United Kingdom
Email: A.Cowling @ dcs.shef.ac.uk*

Abstract

This paper is concerned with trying to characterise the skills that students should develop during the course of a degree programme in software engineering. It is based on a generic framework that has been developed within the UK to describe the abilities that engineering students should possess on graduation, and the paper discusses how this framework could be applied to software engineering graduates at the levels of both bachelor's and master's degrees. The discussion covers the kinds of systems that graduates should be capable of developing, the process model within which this development can be described as taking place, and the levels of ability that could be expected for each of the activities within this process model.

Keywords

Software engineering education, software engineering curriculum, engineering practice, engineering abilities, software development skills.

1. Introduction

The fundamental purpose of any engineering degree programme is to equip students to function as engineers, and this involves them both in acquiring knowledge and learning how to use it. Most of the recent discussion of the curriculum for software engineering (SE from now on) has focused on the aspect of knowledge, starting with the specific knowledge that is required by a practising engineer, as documented in the SE Body of Knowledge project [1] (SWEBOK from now on). In a degree programme this must then be underpinned by more basic knowledge, and the part of the Computing Curricula 2001 project [2] (CC2001 from now on) concerned with developing an SE volume is attempting to document this whole package of SE Education Knowledge [3] (SEEK from now on). To complement these efforts, it therefore seems timely to give some attention to the role of the skills that software engineering students should be expected to develop while studying on these degree programmes, and this is the purpose of this paper.

The starting point for the paper is a problem that had been encountered in the UK with the current accreditation criteria for British engineering degrees, and the background to this problem is described in section 2. Section 3 describes the project that was undertaken to try to produce a generic solution to this problem, and section 4 outlines the principles used in trying to instantiate this for programmes in SE. Sections 5, 6 and 7 then discuss various aspects of this instantiation, and particularly the differences between bachelor's and master's level programmes. Finally, section 8 summarises the conclusions of the paper.

2. Background

In the UK, the engineering profession is regulated by a set of bodies that form a hierarchical structure with two levels. At the top level is the Engineering Council, which sets the generic standards for what it terms registration (ie certification), in the form of a document known as SARTOR [4] (an acronym for "Standards and Routes to Registration"). The lower level of the hierarchy consists of the various professional institutions for different branches of engineering, such as the British Computer Society and the Institution of Electrical Engineers. These administer the processes of registration and accreditation, on the basis of guidelines (such as [5]) that interpret the requirements of SARTOR for their particular discipline.

The processes are under constant development, and the most recent major development (the 1997 version of SARTOR) was mainly to bring UK standards closer to those in other European countries. This involved emphasising the distinction between "Chartered Engineers" (those responsible for innovating technology) and "Incorporated Engineers" (those responsible for managing existing technology), and in particular it raised the educational requirements for chartered engineers from a bachelor's degree to master's level qualifications.

Along with this, it expected the master's programmes to be aimed at the most able students, in the sense of progressing through material at a rate that would stretch them, and so possibly be beyond the less able. It could not specify this directly in terms of the "speed of the course", though, for there is no defined unit in which to measure it: "knowledge units per hour" would not be practical, since the size of a knowledge unit is usually described in terms of curriculum hours. The alternative approach, of trying to define directly the amount of material to be covered, would effectively have led to a national curriculum for engineering, and this was neither feasible at the generic level nor politically acceptable. Thus, the approach that was taken was to define this requirement in terms of the ability of the students, as measured by their qualifications on entry, although this was also recognised to be politically unsatisfactory.

To try to improve on it, an academic body, the Engineering Professors Council, set up a project to try to develop a standard that could be used to define the levels of achievement of graduates from engineering degree programmes. The initial results of this were published in December 2000, in the form of what they called their Graduate Output Standard [6], which is a generic document, in the sense of applying to all branches of engineering. An attempt has been made to create an instantiation of it for bachelor's degree programmes in SE [7], and this paper seeks to extend this to master's level programmes as well, but before discussing how this could be done it is necessary to describe the framework set by the generic standard.

3. The Generic Standard

The generic standard focuses entirely on the abilities and skills that the graduates will have developed to actually do engineering, and it is framed in terms of a generic model of the process by which engineers develop systems, which has six main stages. For each stage it then identifies the key activities, and hence the abilities of graduates to carry out these activities. This therefore gives a structure of 25 "Ability to" statements, divided into six groups, plus another 6 statements covering the application of general transferable skills. The complete set is listed in table 1.

The actual levels of ability implied by these statements are effectively determined by the complexity of the systems being developed, and so the generic standard requires instantiations to be created for different branches of engineering, so as to define one or more benchmark systems. Then, each "Ability to" statement in such an instantiation will be interpreted in the context of these benchmark systems.

Table 1. The generic "Ability To" statements

1. Ability to exercise Key Skills in the completion of engineering-related tasks at a level implied by the benchmarks associated with the following statements.
a) Communication
b) Information Technology
c) Application of Number
d) Working with Others
e) Problem Solving
f) Improving Own Learning and Performance
2. Ability to transform existing systems into conceptual models
a) Elicit and clarify client's true needs
b) Identify, classify and describe engineering systems
c) Define real target systems in terms of objective functions, performance specifications and other constraints (ie, define the problem)
d) Take account of risk assessment, and social and environmental impacts, in the setting of constraints (including legal, and health and safety issues)
e) Select, review and experiment with existing engineering systems in order to obtain a database of knowledge and understanding that will contribute to the creation of specific real target systems
f) Resolve difficulties created by imperfect and incomplete information
g) Derive conceptual models of real target systems, identifying the key parameters
3. Ability to transform conceptual models into determinable models
a) Construct determinable models over a range of complexity to suit a range of conceptual models
b) Use mathematics and computing skills to create determinable models by deriving appropriate constitutive equations and specifying appropriate boundary conditions
c) Use industry standard software tools and platforms to set up determinable models
d) Recognise the value of Determinable Models of different complexity and the limitations of their application
4. Ability to use determinable models to obtain system specifications in terms of parametric values
a) Use mathematics and computing skills to manipulate and solve determinable models; and use data sheets in an appropriate way to supplement solutions
b) Use industry standard software platforms and tools to solve determinable models
c) Carry out a parametric sensitivity analysis
d) Critically assess results and, if inadequate or invalid, improve knowledge database by further reference to existing systems, and/or improve performance of determinable models
5. Ability to select optimum specifications and create physical models
a) Use objective functions and constraints to identify optimum specifications
b) Plan physical modelling studies, based on determinable modelling, in order to produce critical information
c) Test and collate results, feeding these back into determinable models
6. Ability to apply the results from physical models to create real target systems
a) Write sufficiently detailed specifications of real target systems, including risk assessments and impact statements
b) Select production methods and write method statements

c)	Implement production and deliver products fit for purpose, in a timely and efficient manner
d)	Operate within relevant legislative frameworks
7. Ability to critically review real target systems and personal performance	
a)	Test and evaluate real systems in service against specification and client needs
b)	Recognise and make critical judgements about related environmental, social, ethical and professional issues
c)	Identify professional, technical and personal development needs and undertake appropriate training and independent research

The other key feature of the standard is that it defines a threshold, meaning that every graduate should be expected to have achieved every one of these abilities to at least the specified level. Hence, the individual "ability to" statements in an instantiation must define minimum levels of capability, rather than average levels, and so the benchmark systems must similarly specify minimum rather than average levels of complexity.

4. The Standard and Software Engineering

The previous work on instantiating this standard for SE was aimed just at the level of the UK's honours bachelor's degree, which is also recognised (eg by the Washington Accord [8]) as equivalent to the bachelor's degree in North America. As yet, though, such degrees are very rare in the rest of Europe, although they are beginning to be introduced in a process of harmonisation of standards resulting from the Bologna Declaration [9]. Currently, though, in these countries most degrees in professional disciplines are at the master's level, and this is particularly true for engineering [10]. Hence, the main aim of this paper is to extend the previous work to cover master's degrees as well. As with the bachelor's level instantiation, this has involved three main issues. The first is the definition of a benchmark system or systems. The second is to map the process model used for the generic standard into typical SE process models, since there are significant differences between the two. The third is then to define the actual levels of ability that should be specified for each of the generic "Ability to" statements.

Underlying all of these issues is the impact that such a standard might have on the way in which students' work is assessed. The project to create the generic standard was intended to lead eventually to a specification of output standards that could replace the current specification of input qualifications in the UK's accreditation criteria. This would imply that programmes seeking accreditation should be able to show that their graduates were developing these abilities to the specified levels, and so the assessment mechanisms in the programmes would need to be able to support this. Many of these abilities are quite general, and so their assessment is not a simple process [11], but a major component of it will be the assessment of practical projects that the students undertake, and particularly capstone projects. This results in a variety of links between an instantiation of the standard and the requirements for such projects.

5. Benchmark Systems

In particular, any requirements for benchmark systems will effectively become requirements for the kinds of systems that should be developed during such projects, and practical constraints on such projects, such as curriculum time, will also constrain what can reasonably be specified as benchmark systems. In the trial instantiations that were produced as part of the original work on the generic standard, this had been reflected in the choice of specific example systems, such

as an audio amplifier with a particular range of performance characteristics as a benchmark system for electronic engineering. In proposing requirements for an instantiation for SE at the bachelor's level, however, key concepts from project estimation methods (such as function points [12] and object points [13]) were used to describe the complexity of benchmark systems more directly. Thus, target ranges were developed for four aspects of system complexity: the number of entities or business classes in the data model; the number of relationships or associations per entity or business class; the number of menu functions in the external view of the processing model; and the number of sub-systems in the internal view of this model.

In extending these requirements to master's level, a key issue was how the constraints of curriculum time for capstone projects might vary. This requires further study, but it appears that typically a master's project has more time allocated than a bachelor's project, by a factor that lies roughly in the range 1.2 to 1.5. Using typical parameters from effort estimation models (eg [14]), this would mean that a master's project might be expected to be roughly between 1.1 and 1.3 times more complex than a bachelor's project, which should be reflected in the requirements for benchmark systems. The other important constraint is Miller's limit of seven plus or minus two [15], which implies that each aspect may need to involve a minimum of ten components in order to ensure that systematic development methods have to be employed, which is obviously an important feature in demonstrating practical ability to do SE.

For the data model, experience of running various kinds of projects in the undergraduate curriculum at Sheffield [16] had shown that for this purpose the most important components for the effective complexity were not the business classes (or entities), but the associations (or relationships). Since five business classes gives ten possible pairs that could be involved in associations, the requirement that had been derived from this was that a benchmark system at bachelor's level should have between five and seven business classes. Then, the need to avoid some particularly simple topologies for the model had given rise to a second requirement, that at least two of the business classes should each be involved in more than two associations.

At master's level this could be scaled up to give a range such as six to nine business classes, but because of the threshold property of the requirements it is only the lower limit that is vital. Here, the possible variations in the number of associations is actually more significant than the difference between five and six business classes, and so there is little point in having different ranges for the two levels, and it is proposed that this requirement should be the same at both.

For the processing model, the requirement that had been proposed at the bachelor's level was that the external view should contain between twenty and thirty basic functions, but the arguments used to derive this were based closely on the provision of separate functions to create, edit and delete data for each of the main business classes and associations. In considering how to adapt this range for the master's level, a number of the projects that had been undertaken by students in the author's department were reviewed. This demonstrated clearly that this notion of expecting separate functions was not valid, since in many cases the data sets had fairly complex structures, so that functions could not sensibly be broken down into separate external operations on different business classes. Hence, it was concluded that it was inappropriate to require benchmark systems to have any particular level of complexity for the external view of the processing model, and the only requirement should be based on the internal view of it.

Here the underlying structure is invariably hierarchical, but in practice the complexity can be measured for this purpose by the number of components (ie sub-systems) in the top layer of the hierarchy, which will be determined by the basic architecture of the system. In many cases this will use three layers, for the user interface, the business logic and the persistent storage, and so the basic requirement proposed at bachelor's level was that a benchmark system should involve three sub-systems, but without any requirement that these be organised as layers. To reflect the different ways in which software engineers actually develop sub-systems, though, the other

requirement that was imposed was that at least one of them should be developed by configuring existing components, using the interfaces that they provided, and at least one should be developed from first principles.

Qualitatively, these two requirements for the processing model are obviously still valid at master's level, and quantitatively the parameters described above would not justify any changes to the numbers. Consequently, the specification of a benchmark system is identical at both levels, and is as summarised in table 2. This does not mean that typical master's projects will not be more complex than the bachelor's ones, but the additional complexity will arise in other ways, such as a more specialised context for the development of the system, and hence more difficulty in analysing the requirements or designing a system to meet them.

Table 2. The proposed specification for a benchmark software system.

At both bachelor's and master's level, a benchmark software system is any system whose complexity lies within the following bounds.	
i)	Its data model consists of between five and seven entities or business classes, of which at least two are each involved in more than two relationships or associations.
ii)	Its processing model consists of three sub-systems, of which at least one is to be developed by configuring existing components using the interfaces that they provide, and at least one is to be developed from first principles.

One other issue is that of whether the specification given in table 2 should be augmented with actual examples of typical systems, as have been given in the other trial instantiations. This, though, is really for others to judge rather than the author, and so must be left as an aspect of this work that still requires further development.

6. Development Process Models

The most difficult part of instantiating the generic standard for SE was trying to reconcile the process model underlying the generic standard with the ones commonly used in SE, since they also assume different sets of product models. As table 1 implies, the generic standard assumes just three kinds of product model, which it calls conceptual, determinable and physical models respectively, and the definitions of these are given in table 3. The way in which they are meant to be used at different stages in the generic process can be inferred from table 1, and in the following description references to stages of this process identify the sections of this table.

Requirements analysis is stage 2, and is described as delivering a conceptual model, or possibly a set of alternatives, so that it also includes some aspects of concept design. These designs are formalised in stage 3, to deliver a range of determinable models, which are then evaluated and optimised in stages 4 and 5, with stage 4 relying mainly on mathematical analysis and stage 5 on the use of physical models for validation. Stage 6 translates the selected optimal design into a suitable form for the production process, and stage 7 covers the activities of project evaluation and further development, of both the project and the person.

Hence, the distinction between conceptual and determinable models is fundamental to this process, but it has little relevance to the current state of SE, where the main kinds of models used in both requirements analysis and design are diagrammatic ones, typically expressed in UML [17], and so they would all be classed here as conceptual models. As described in a separate paper [18], within SE the distinction that the generic standard makes between these two kinds of models would usually be regarded as the one between qualitative and quantitative models. Also, reflecting the need to model problem domains as well as solutions, SE puts much more emphasis

on the different perspectives from which modelling is carried out [19], and in particular it typically uses the term conceptual model for one (usually qualitative) that is developed during requirements analysis from the conceptual perspective [20]. Furthermore, as work on methods integration (such as the precise UML project [21]) progresses, so qualitative models developed from the specification perspective may well become sufficiently precise and formalisable (if not actually formal) that they should be classed instead as determinable rather than conceptual models. This therefore leads to the mapping for product models that is given in table 4, and this applies to both bachelor's and master's levels.

Table 3. Definitions of the product models used in the generic standard.

Conceptual model	A graphical, diagrammatic, symbolic or otherwise mentally apprehensible representation of an engineering system illustrating the relationship between key parameters in a form that may be transformed into a determinable model. (For example, the model used in a process diagram, a circuit diagram, a pipe network, a structural frame, a magnetic field pattern.)
Determinable model	A mathematical, computer/numerical, or logical representation of a conceptual model which enables the key system parameters to be firmly decided or definitely ascertained. (For example, a finite element computer model, a set of algebraic equations.)
Physical model	A physical representation of all or part of a real target system capable of being tested practically to determine or verify key system parameters. A prototype of the real target system. (For example, a wind tunnel test, a materials test, a field trial.)

With this mapping of the product models, there are then three main differences that need to be accommodated in mapping SE process models into the generic one. These are: the different breakdown of the design activity, the different treatment of construction, and the extent to which feedback loops are incorporated.

Table 4. The proposed mapping for the kinds of product models.

Generic Product Models	SE Product Models
Conceptual model, as the term is used in the activities of the requirements analysis stage.	Qualitative model, constructed from the conceptual perspective.
Conceptual model, as defined.	Qualitative model.
Determinable model, as the term is used in the activities corresponding to the architectural design stage.	Qualitative model plus derivable formal models or specifications.
Determinable model, as the term is used in the activities corresponding to analysis of performance or quality.	Quantitative model.
Physical model.	Prototype.

In SE the design activity is usually just broken down into two stages, architectural design and detailed design, whereas the generic model assumes three stages. Given the mapping that is proposed for the product models, two of these stages (3 and 4 in table 1) can be regarded as corresponding roughly to the two stages that are recognised in SE, which just leaves stage 5 in

table 1 unaccounted for. This, though, is primarily a reflection on the relative immaturity of SE, where it is still commonplace (particularly in student projects) to regard any reasonably feasible solution as sufficient, rather than expecting any form of optimal solution. As the discipline matures, so one can expect that the activity of optimising designs (as in stage 5 of the generic process) will become increasingly significant, and this should start to be reflected first in degree programmes at master's level, and then work its way down to bachelor's level.

For the treatment of construction, the generic model implicitly assumes that the engineer only has to design and manage this, rather than actually undertake it, since this is the normal practice in other branches of engineering. By contrast, SE still treats it as an activity that is conducted by engineers rather than technicians, and one approach (particularly associated with agile methodologies such as XP [22]) is that this reflects the inherent flexibility of software, and can be expected to continue. The opposing approach is that, particularly as code generation tools develop, so the main focus of SE should move from construction to design, as in other branches of engineering. Until this has happened, though, graduates from SE programmes at either level will still need to achieve a reasonable level of competence at actually undertaking construction (ie activity 6(c) of table 1), rather than simply managing it (ie activity 6(b)). Again, though, one can expect to see changes in this occurring first at master's level, and then working down to bachelor's level.

The third feature of the generic process model is the very limited occurrence of explicit feedback loops by comparison with typical SE processes: indeed, it is effectively a straight waterfall model with big-bang delivery. In part this reflects what other branches of engineering would regard as inherent limitations on their kinds of artefacts, which mean that they could not be developed on any kind of evolutionary or incremental basis, although arguably their view is only valid because they fail to look at the lifecycles over a long enough time scale. Within SE, though, these sort of development processes are commonplace, and so the activities of managing them are important. In particular, there is an expectation that a student of SE at either bachelor's or master's level should, in the course of their capstone project, take some role in planning its process, rather than simply following one prescribed by a project supervisor. The proposed instantiation of the standard for SE at bachelor's level could not reflect this, since the relevant activities just do not appear in the generic model. The only way in which this could be solved for either level would be to create an additional "ability to" statement, along the lines of "plan and manage a development process that will match the client's true needs", and add this somewhere to the set in table 1 – probably in section 2, or maybe 7.

7. Levels of Ability

In the trial instantiations that were produced as part of the project to develop the generic standard, the phrase "ability to" was left unqualified, except for one that suggested that lower levels of attainment (such as "experience of" or "knowledge of") might be substituted in some of the statements. In part this reflected the threshold characteristic of the standard, which means that every graduate should be expected to achieve the specified level in every one of the abilities listed in table 1, and this effectively implies a binary decision. In working towards an instantiation for SE, it was felt that this approach was unlikely to be adequate at both bachelor's and master's levels, where the difference between the two was likely to be reflected in a need to specify just what level of ability was being required in performing the various activities. Thus, in the proposals at bachelor's level an attempt was made to define the level of ability more precisely, using the general wording "a software engineering graduate must be capable of carrying out the basic activity correctly, but not necessarily getting every detail right, or understanding every fine nuance of either the activity or the system to which it is being applied".

By contrast, for master's graduates one would expect a higher level of ability, but it is not obvious how much higher this should be, and whether it should apply to all of the "ability to" statements or just some of them. One would expect the generic guidelines for accreditation in the Engineering Council's SARTOR document to shed some light on this, and the UK's Quality Assurance Agency (QAA) has also produced a National Qualifications Framework [23]. This tries to describe the different levels of post-school qualifications in the UK, from sub-degree ones through to doctorates, although it does so in terms that are generic across all disciplines.

Most of what SARTOR has to say about the distinction between the two levels is expressed in terms of knowledge that is to be gained. For instance, master's graduates should have "a broader and more general educational base", an "increased depth and range of specialist knowledge", and this should demonstrate a "greater extent of industrial relevance". It does also make some statements about abilities, but the majority of these (such as "use IT effectively", or "manage projects, people, resources and time") are common to both levels. The most notable difference is that master's level includes an additional statement, "be creative and innovative", and also one or two other statements specify more complex forms of ability at master's level.

In addition, SARTOR requires master's graduates to have undertaken both an interdisciplinary group and an individual research project (either of which could play a capstone role, but does not have to), whereas a bachelor's graduate simply has to undertake an individual capstone project. In terms of the key skills that are listed in section 1 of table 1, this means that master's graduates can be expected to have achieved a much higher level than bachelor's graduates in ability 1(d) ("working with others"), but otherwise the differences in the abilities that should be demonstrated at the two levels lie mainly in the contexts in which the projects are carried out. In particular, at master's level SARTOR expects that the abilities relating to requirements analysis (in section 2 of table 1) should be demonstrated in application domains that require both a wider range of knowledge than at bachelor's level, and possibly also more specialised knowledge, and this need for additional knowledge also feeds through into the design activities.

Table 5. The general abilities required by the UK National Qualifications Framework.

Honours Bachelor's Level	Master's Level
apply the methods and techniques that they have learned to review, consolidate, extend and apply their knowledge and understanding, and to initiate and carry out projects;	deal with complex issues both systematically and creatively, make sound judgements in the absence of complete data, and communicate their conclusions clearly to specialist and non-specialist audiences;
critically evaluate arguments, assumptions, abstract concepts and data (that may be incomplete), to make judgements, and to frame appropriate questions to achieve a solution - or identify a range of solutions - to a problem;	demonstrate self-direction and originality in tackling and solving problems, and act autonomously in planning and implementing tasks at a professional or equivalent level;
communicate information, ideas, problems, and solutions to both specialist and non-specialist audiences;	continue to advance their knowledge and understanding, and to develop new skills to a high level;

Much the same is true of the National Qualifications Framework, which describes (in very general terms) both the kinds of knowledge that graduates should possess at each level, and the abilities that they should be able to demonstrate to apply this knowledge. The latter are set out in table 5, but their organisation renders point-by-point comparisons difficult. Where they can be

made, though, they reinforce the approach of the accreditation criteria, in that the main difference is in the complexity of the problems to be solved, and the levels of knowledge, originality and creativity required to solve them.

This requirement to apply a greater depth and range of specialist knowledge at master's level is reflected to some extent in the first draft of the SEEK, which proposes a list of what it calls "Systems and Applications Specialities", and it suggests that students may either specialise in one or more of these areas, or by deeper study of some of the core knowledge areas. Putting the ideas from these documents together thus suggests that, at master's level, the "either ... or" should disappear, so that these students should be required to study both one or more of these application areas, and also specialisms within the core. This could then be incorporated into the "ability to" statements, in two ways. Firstly, it could be specified that at master's level the benchmark system should be one that is drawn from the specialist application area, and so should incorporate requirements that reflect the specialised features of this area. Secondly, in place of the general wording suggested at the beginning of this section, one would expect master's graduates to be able to get more details right in carrying out the activities, and in particular to understand more of the nuances of the activities or the systems to which they are being applied.

The application of this higher level of knowledge could then be demonstrated by specifically demanding a higher level of performance from master's graduates in the ability to evaluate the significance of details, particularly in those activities that they have studied to greater depth. Beyond this, though, the specification of the particular activities where this knowledge would be demonstrated could not be done generically, and so instead it would have to be specified for each individual degree programme by its designers, so as to reflect the particular focus of that programme.

8. Conclusions

The overall conclusion of this paper is that the generic graduate output standard can be instantiated for SE, at both bachelor's and master's levels, and this provides a good definition of what graduates in SE should be able to do, as an important complement to descriptions of what they should know. In particular, it provides a sound basis for characterising the kinds of systems that they should be expected to develop competently, although whether this needs to be supplemented by descriptions of typical example systems is still an open question.

The work described here has, though, highlighted two weaknesses of the generic standard. One is that its set of product models needs to be made richer, to reflect the extent to which SE, as a socio-technical branch of engineering, needs to model problem domains as well as solutions. The other is that it needs to recognise the more extensive set of feedback loops that exist within typical SE development processes, and in particular it needs to incorporate explicitly an ability to manage these processes.

The work has also highlighted three weaknesses of the current state of SE, reflecting its relative immaturity as a branch of engineering. One is the extent to which it still relies on relatively informal product models, rather than rigorous ones. The second is the very limited attention given to trying to achieve optimal designs, rather than ones that are merely feasible. The third is the extent to which it still treats construction as an engineering activity, rather than simply a technical one.

As SE matures, one can expect that developments in all of these areas will appear in master's level programmes first, and then work their way down to bachelor's level ones. This illustrates the final conclusion, which is that while master's level graduates can be expected to demonstrate greater levels of the abilities described here than bachelor's graduates, the differences between these two levels need to be described mainly in terms of the additional knowledge that is being

applied. This can not be done at the level of SE as a discipline, but rather it will need to be documented separately for each individual degree programme.

References

- 1 IEEE Computer Society, *Guide to the Software Engineering Body of Knowledge: Trial Version (0.95)*, IEEE Computer Society (May 2001), and also at <<http://www.swebok.org/stoneman/version095.htm>>.
- 2 IEEE-CS and ACM Joint Task Force on Computing Curricula, Approved Final Draft Version of Computing Curricula 2001 (15th December 2001), <<http://computer.org/education/cc2001/final/index.htm>>.
- 3 A E K Sobel (ed), *Computing Curricula - Software Engineering Volume: First Draft of the Software Engineering Education Knowledge (SEEK)*, 28th August 2002, <<http://sites.computer.org/ccse/artifacts/FirstDraft.pdf>>.
- 4 Engineering Council, *Standards and Routes to Registration (SARTOR)*, 3rd edition, London (1997).
- 5 British Computer Society, *Guidelines on Course Exemption & Accreditation*, Swindon, UK (September 2001), and at <<http://www1.bcs.org.uk/>>.
- 6 Engineering Professors Council, *The EPC Graduate Output Standard: Interim Report of the EPC Output Standard Project*, Occasional Paper No. 10, Coventry, England (December 2000), and at <<http://www.engprofc.ac.uk/op/op10.html>>.
- 7 A J Cowling, Towards a Graduate Output Standard for Software Engineering, to appear in J B Thompson & H Edwards, *Post-Summit Proceedings of the International Summit on Software Engineering Professionalism*, (co-located with the International Conference on Software Engineering (ICSE) 2002), Orlando, Florida, to be published by Sunderland University Press (2002).
- 8 Washington Accord, *Recognition of Equivalency of Accredited Engineering Education Programs Leading to the Engineering Degree*, (1989), <<http://www.washingtonaccord.org/>>.
- 9 Confederation of EU Rectors Conferences and the Association of European Universities (CRE), *The Bologna Declaration on the European space for higher education: an explanation*, (2000), <<http://europa.eu.int/comm/education/socrates/erasmus/bologna.pdf>>.
- 10 R Kirby (ed), *European Engineering Yearbook 1996*, Cambridge Market Intelligence Ltd with FEANI, London, (1996).
- 11 Engineering Professors Council Assessment Working Group, The EPC Engineering Graduate Output Standard: Assessment of complex outcomes, Institution of Mechanical Engineers, London, England (January 2002), and at <<http://www.engprofc.ac.uk/pap/EPC%20AWG%20Report%20TF%2012Feb02.pdf>>.
- 12 C R Symons, *Software Sizing and Estimating: Mark II Function Point Analysis*, Wiley, New York (1991).
- 13 B W Boehm, B Clarke, E Horowitz *et al*, Cost Models for Future Life Cycle Processes: COCOMO 2.0, *Annals of Software Engineering* **1** (November 1995), pp 57-94.
- 14 B W Boehm, *Software Engineering Economics*, Prentice Hall, Englewood Cliffs NJ (1981).
- 15 G A Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information, *Psychological Review* **63**, 81 – 97 (1956).
- 16 A J Cowling, The first decade of an undergraduate degree programme in software engineering, *Annals of Software Engineering* **6**, 61 – 90 (1998).
- 17 G Booch, J Rumbaugh & I Jacobson, *The Unified Modelling Language User Guide*, Addison Wesley, Reading MA (1999)
- 18 A. J. Cowling, Modelling: A Neglected Feature in the Software Engineering Curriculum, these proceedings.
- 19 S Cook & J Daniels, *Designing Object Systems: Object-Oriented Modeling with Syntropy*, Prentice Hall, Englewood Cliffs NJ (1994)
- 20 M Fowler with K Scott, *UML Distilled: Applying the Standard Object Modeling Language*, Addison Wesley, Reading MA (1997).
- 21 A Evans, *The precise UML group: main details*, <<http://www.cs.york.ac.uk/puml/maindetails.html>> (2002).
- 22 K Beck, *Extreme Programming Explained: Embrace Change*, Addison Wesley, Reading MA (2000).
- 23 QAA, *The framework for higher education qualifications in England, Wales and Northern Ireland*, Quality Assurance Agency, Gloucester, England (2001).