Computing Curricula Software Engineering: Position Paper

David Budgen
School of Computing & Mathematics
Keele University
Staffordshire, UK
d.budgen@cs.keele.ac.uk

1. Introduction

My position in terms to the questions asked of this panel is that I see this curriculum as being a further step in the progression of the discipline and the development of the associated educational framework and materials.

The rest of this short document begins by providing some brief background about my own interests and involvement—largely because this strongly influences the position that I have adopted and because it also explains some of my responses to the additional issues raised in the introductory document. After that comes a short section discussing the (very limited) empirical work addressing needs, and then the section containing a more detailed answer to the question and the associated issues. The final section then expands a little on my overall response, in order to explain my views about where we should be headed for the next step.

2. Background

Any panel member naturally brings along their own particular interests and prejudices arising from these. In my own case, my long-term interest has been in the general area of software design as demonstrated through my book *Software Design* (Budgen, 2003). Arising from that has been a gradually-developing interest in component-based and service-based systems (how do we teach others to design with larger abstractions?) and also an interest in empirical studies (how do we know what works and doesn't work, and what evidence can we present to support these views?).

My interest in software engineering education was fostered and enhanced by a six-month spell with the newly-formed Software Engineering Institute's *Education Group* in 1986, and through subsequent contact over the following seven years. My experiences with the SEI's Curriculum Modules are described in (Budgen & Gibbs, 1989; Budgen & Tomayko, 2003; Budgen & Tomayko, 2004).

One thing we included in Budgen & Tomayko (2004) was an informal assessment of the coverage of the set of SEI curriculum modules (as planned in 1986) compared with that of the CCSE, and particularly with the element of this known as the SEEK (Software Engineering Education Knowledge). At the structural level at least, we concluded that the differences were not particularly great. (The most obvious developments were the addition of *Professional Practice* and *Software Process* as topics, together with a decreased emphasis upon issues such as concurrency, with the latter perhaps due mainly to the 'temporal' influence of the SEI's interest in Ada).

Indeed, it is this as much as anything which has influenced my response. In their time, the SEI curriculum modules were an important development, as in its turn is the CCSE. Both have involved considerable effort and have employed expert judgement backed up by a review process. So, in many ways I see the CCSE as an incremental step, and as I explain later, the nature of these initiatives also points to the way forward.

3. How well do we do at present?

One of the difficulties for software engineering is the lack of solid *evidence* about the effectiveness and limitations of our ideas (Budgen *et al*., 2004; Glass *et al*., 2004; Kitchenham *et al*., 2004a). The problem is no less significant for software engineering education—the papers submitted for the Conference on Software Engineering and Training each year too often reveal not only a lack of relevant empirical data, but also of unfamiliarity with the techniques needed to collect this in any 'reliable' manner!

A few years ago, Tim Lethbridge sensibly took the approach of surveying the views and experiences of graduates of computing programmes to see what had been most useful and least useful topics from those that they had learned as part of their degree (Lethbridge, 2000). While the outcomes are important, the paper did also have some methodological limitations—although in all honesty, our own attempt to repeat this with a more tightly-defined population only encountered different problems (Kitchenham *et al*., 2004b)!

Despite the differences in the respondent profiles (those used by Lethbridge were usually in North America whereas ours were all in the UK, and his had longer experience in industry than ours), there was much in common in the findings, especially in terms of the topics rated most highly. Both surveys also identified a tendency for courses to over-emphasise mathematical

Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)
0730-3157/04 $20.00 © 2004 **IEEE**

topics, while giving too low a prominence to general business topics.

4. A watershed or an increment?

To return to the key question posed for the panel—namely whether it "will produce the type of graduate that industry so obviously needs", we begin by addressing the three issues identified as related to this.

4.1 The CCSE volume and its development

The processes adopted for developing this volume have a number of strengths and weaknesses. Strengths include:

- the thoroughness of the development process, and the emphasis upon consultation and review;
- its clear positioning in the computing spectrum;

While on the debit side:

- there is a strong North American flavour, this is particularly evident in the later chapters which seem less useful in (say) a European context;
- while the process was thorough, its structure was not always wholly explicit;
- only limited *evidence* was available to the compilers, and none that could be readily subjected to systematic review (Kitchenham *et al*., 2004).

4.2 Its worth and likelihood of adoption

In the absence of national frameworks and regulatory structures for software engineering, the CCSE does offer real value as a compendium of expert judgement. (This is particularly true for the SEEK, which is likely to be a useful benchmark). As with any such document, there is a need to maintain and update both the structure and the topics over time, but that does not detract from its worth.

We should also note that this role is particularly important in a university context, where it may be necessary to make the case for new curriculum developments to central faculty boards. The ability to cite the CCSE as an authoritative source may be very helpful to many university teams!

4.3 Effect upon the balance between CS and SE

One problem here is that many of us may not have the luxury of making the distinction very strongly. Within the UK context at least, many universities find themselves running CS, SE and IS programmes as a matter of economic necessity. It would seem unlikely that we will see separate and distinct departments running such programmes in the future.

4.4 Will this produce the type of graduate we need?

Certainly not on its own, although it is likely to be a useful resource. What we need are graduates who have both a good grounding in the subject as well as the ability to think clearly and deploy that knowledge. The CCSE volume helps with addressing the first part of this, but we need to do some quite serious thinking about the second.

As an example of the problems that we face, the issue of how well we teach our students to use *abstraction* is a particularly important one, as indicated both by the surveys described in §3, and also in (Denning, 2004). We increasingly create systems from existing building blocks, and the challenge of doing this is considerable enough, let alone teaching others how it might be done. The CCSE could be viewed as taking a rather 'traditional' programming-centric view in this respect.

5. Conclusions

No curriculum, however good, will solve our problems on its own. We need to take and use this effectively (and hence need measures of what is effective too). My position is therefore that the CCSE is a useful and quite important stepping-stone, but it is not a watershed in the education process—unfortunately, I think there are many more tasks ahead of us.

The SEI's programme to develop curriculum modules back in the late 1980's represented a determined attempt to bootstrap software engineering education, using expert judgement with some degree of consultation. The process used to develop the CCSE has moved on a step from this, but was still heavily dependent upon expert judgement—even if this has drawn upon a larger community and longer experience. For the next step, we therefore need to begin to replace the use of expert judgement with *evidence* that has been carefully assembled and systematically reviewed, and then…

References

Budgen D. (2003). *Software Design*, second edition. Pearson Addison-Wesley.

Budgen D. & Gibbs N.E. (1989). The education programme of the Software Engineering Institute, Carnegie Mellon University, *Soft. Eng. J.*, **4**(4), 176-185.

Budgen D. & Tomayko J.E. (2003). Norm Gibbs and his Contribution to Software Engineering through the SEI Curriculum Modules, in *16th Conf. on Softw. Eng. Educ. & Training*, IEEE Computer Society Press, 3-13.

Budgen D. & Tomayko J.E. (2004). The SEI Curriculum Modules and their influence: Norm Gibbs' legacy to software engineering education, *J. of Sys. & Softw.*, accepted for publication.

Denning P.J. (2004). The field of programmers myth, *Comm. ACM*, **47**(7), 15-20.

Glass R.L., Ramesh V. & Vessey I. (2004). An analysis of research in computing disciplines, *Comm. ACM*, **47**(6), 89-94.

Kitchenham B.A., Dybå T & Jørgensen M. (2004a). Evidence-Based Software Engineering, in *Proceedings of ICSE 2004*, IEEE Computer Society Press, 273-281.

Kitchenham B.A., Budgen D., Brereton P. & Woodall P. (2004b). An investigation of software engineering curricula, *J. of Sys. & Softw.*, accepted for publication.

Lethbridge T. (2000). What knowledge is important to a software professional?, *IEEE Computer*, **33**(5) 44-50.