Accreditation of Software Engineering Programs by the CSAC

Guy Tremblay
UQAM & Member of CSAC

CUSEC
January 17th, 2003

Overview

- What is the purpose of accreditation?
- What is CSAC?
- How does a CS/SE program become accreditated?
- Is there a core body of knowledge for software engineering?
- How is this emerging BOK reflected in CSAC's accreditation criteria?

1 Accreditation of CS and SE programs by CSAC

“Accreditation recognizes that programs meet published, generally accepted criteria for sound education in the discipline and provide evidence of the quality of a CS or SE degree.”

Accreditation objectives:

- Promote public welfare through development of better-educated professionals.
- Ensure that a program has a purpose appropriate for higher education and has resources and services sufficient to accomplish its purpose.
- Foster a cooperative approach to IT education involving industry, government, and educators to meet needs of society.
- Provide opportunity of improvement to educational institutions (strengths and weaknesses).

CSAC

= Computer Science Accreditation Council

- Established in the 1970s
- Autonomous body of Canadian Information Processing Society (CIPS)

Role and objectives:

- Formulate and maintain high educational standards for CS and SE programs, including definition of accreditation criteria.
- Review and accredit undergraduate CS and SE programs at Canadian universities *on a voluntary basis*.
- Avoid rigid standards to prevent conservatism and encourage planned experimentation.

2 Procedures and criteria for CSAC accreditation

Overview of accreditation process

1. University fills a detailed questionnaire.
2. Accreditation team (three persons) visits university (two days).
3. Draft report is prepared by visiting team.
4. Draft report is reviewed by university.
5. Council approves report and gives accreditation decision.

Fully *confidential* process . . . until accreditation is *obtained*.

Criteria (report and visit)

- Control and organization of institution;
- Students (admission, standing, graduation);
- Faculty (morale and calibre, teaching load, research funding);
- Resources (financial, physical, support staff, library);
- Curriculum.

Curriculum

- Breadth and depth:
  - 15 CS/SE courses
    (at least one course in each of six key sub-areas).
  - 5 mathematics courses.
  - 10 courses *which are neither CS/SE nor mathematics.*
- Development of oral and written communication.
- Professionalism (social, ethical and legal issues).
- Presence of a significant practical component.

3 Body of knowledge for SE

The emergence of a *profession* requires the existence of a well-documented core body of knowledge (BOK).

SE's BOK still immature and *evolving*, yet there has been three major efforts at defining a BOK for SE:

1. SWE-BOK
2. Guide to the SWEBOK
3. SEEK

3.1 SWE-BOK

- Triggered by an FAA initiative to “improve the SE competencies of its technical and management staff” . . . at which point they were unable to find an *existing* SE BOK.
- SEI Technical Report (1999) : “A Software Engineering Body of Knowledge (Version 1.0)”.

I. Computing fundamentals	• Algo. and data str.	• Comp. arch.	• Math. foundations
	• Op. syst.	• Prog. lang.	
II. Software product eng.	• Requirements	• Design	• Coding
	• Testing	• Operation and maint.	
III. Software management	• Project	• Risk	• Quality
	• Configuration	• Acquisition	
IV. Software domains	• AI	• DB	• HCI
	• Num./symb. comput.	• Simulation	• Real-time

3.2 Guide to the SWEBOK

- Project sponsored by IEEE Computer Society and corporate sponsors (Raytheon, SAP, Rational, Mitre, NRC, NIST, etc.).
- Goal was to identify *generally accepted* core SE knowledge *for people with five (5) years experience.*
- Two year (*major*) effort lead (2001) to a “Guide to the Software Engineering Body of Knowledge (Trial Version 1.00)”.
- Used a *bottom-up* approach to the identification of the major Knowledge Areas (KAs).

Ten major KAs for SE:

1. Software requirements
2. Software design
3. Software construction
4. Software testing
5. Software maintenance
6. Software configuration management
7. Software engineering management
8. Software engineering process
9. Software engineering tools and methods
10. Software quality

List of *Related disciplines*:

- Computer science
- Mathematics
- Project Management
- Computer and systems eng.
- Management and Management sciences
- Cognitive sciences and human factors

3.3 SEEK

- Sponsored by IEEE Computer Society and ACM
- Spin-off of CC-2001 : Computing Curricula Soft. Eng. (CCSE)
- So far :
  - Set of guiding principles;
  - Overall structure for SE Education Knowledge Areas (SEEK Areas);
  - Draft chapters of curriculum and KAs.

Some key CCSE principles (a few out of 11) :

- SE draws its foundations from a wide variety of disciplines.
- Rapid evolution of field ⇒ need for ongoing review process of curriculum.
- Guidance of SE curricula must be based on an appropriate definition of SE knowledge.

Overall structure of SEEK Areas:

1. Fundamentals : math., comp., eng., modeling
2. Professional practice : group dynamics, comm. skills, prof.
3. Software requirements
4. Software design
5. Software construction
6. Software verification and validation
7. Software evolution
8. Software process
9. Software quality
10. Software management
11. Systems and application specialties

4 Improving the CSAC's SE criteria

The current SE's criteria are "pre-SWEBOK" and can be improved.

Guiding "principles":

- The SE BOK is still evolving ⇒ we should not be too specific.
- The SE criteria should be a *superset* of those for CS.
- A SE program should cover each of the key KAs, not necessarily with a specific course for each KA but clearly not with a single course.

=> For an undergraduate program (with CS core), a number of SE KAs can be merged.

- A "practical" component must be present for some application domains (SE is not done in the *abstract*).

The proposed areas for the SE part of curriculum:

1. Software requirements
2. Software design and architecture
3. Software construction and maintenance
4. Software testing and quality
5. Software management and process
6. Application domains (embedded, real-time, distributed, HCI, DB)

Other CS curriculum criteria must also be satisfied:

- CS courses in each of the following areas: algorithms and data structures, programming languages, systems software, computer elements and architecture, theoretical foundations.
- Mathematics (mostly discrete) courses.
- Courses *which are neither CS/SE nor mathematics.*

- Development of oral and written communication.
- Aspects of professionalism, social implications of computing, ethical and legal issues.
- Presence of a significant practical component.

5 Conclusion

- The core of a CSAC's accreditated SE program *remains in CS*.
- Major recent efforts at defining the SE BOK should be taken into account ... but not too rigidly as BOK is still evolving.
- Revision of CSAC's criteria still remains to be approved by CIPS (after appropriate reviews).
- **Final wish**: let's hope Canadian engineers and computer scientists, as in other countries, can better *cooperate* to improve the emergence and professionalisation of this important field.