Computer Science Accreditation Council

Accreditation Criteria for Computer Science, Software Engineering and Interdisciplinary Programs

Approved August 2011

Table of Contents

OBJECTIVES

- MAINTAIN HIGH EDUCATIONAL STANDARDS
- PROMOTE AND ADVANCE COMPUTER SCIENCE AND SOFTWARE ENGINEERING EDUCATION
- FOSTER COOPERATIVE APPROACHES

Abstract

These guidelines are written to provide assistance to faculty and administrators involved in the accreditation of computer science, software engineering and inter-disciplinary programs within public and private not-for-profit universities. They specify the objectives of accreditation, the various steps in the process, and the essential and highly desirable qualities of accreditable programs. Questions and suggestions for improvements may be sent directly to the CIPS Accreditation Secretariat (accreditation@cips.ca), which will ensure that they are considered

1.0 Introduction

The Computer Science Accreditation Council is an autonomous body established by the Canadian Information Processing Society (CIPS).

The Council has as its objectives:

1. To formulate and maintain high educational standards for Canadian universities offering computer science and software engineering programs, and to assist those institutions in planning and carrying out educational programs.

2. To promote and advance all phases of computer science and software engineering education with the aim of promoting public welfare through the development of better educated computer professionals.

3. To foster a cooperative approach to computer science and software engineering education among industry, government, and educators to meet the changing needs of society.

The purpose of accreditation is to recognize programs whose graduates will have received an outstanding undergraduate education in computer science or software engineering – an education informed by state of the art research, the mathematical underpinnings of computer science, and the needs and applications of industry. This version of the accreditation criteria incorporates principles of outcomes based accreditation. Historically, much weight in accreditation decision-making was placed on inputs, such as number of courses taught, and lists of topics in the curriculum. The trend in all accreditation agencies is to shift the emphasis instead towards outcomes, i.e., measuring the extent to which program objectives have been met. This document reflects this new approach. However, for the period ending in 2015, programs that meet the previous accreditation criteria will still be accreditable.

2.0Method of Evaluation

Programs submitted for accreditation will be evaluated on the basis of data submitted by the institution in the form of a self-study report and other supporting documentation, together with the report of an on-site visit by a carefully selected team representing the Council.

The self-study report should follow a structured outline to be described in Sections 4 though 8, and involves answering a series of questions and completion of tables. During the process of creating the report, the institution should demonstrate to itself and to the Council that it can meet the accreditation criteria or, if not, it should demonstrate that it is aware of the shortcomings and has a concrete plan to rectify them. In particular, the report should demonstrate how all aspects of the program, including students, faculty, resources and curriculum together enable the achievement of a set of defined program objectives, discussed in Section 4. The self-study report will be used as primary input for the analysis of the program by the on-site visiting team.

The purpose of the site visit is three-fold:

First, the site visit should assess factors beyond those described in the questionnaire. The intellectual atmosphere, the morale of the faculty and the students, the calibre of the staff and the student body, and the character of the work performed are examples of intangible qualitative factors that are difficult to document in a written statement.

Second, the visiting team should help the institution assess its weak as well as its strong points.

Third, the team should examine in further detail the material compiled by the institution and relating to:

1. Control and organization of the institution.

2. Education programs offered and degrees conferred.

3. The basis of and requirements for admission of students.

4. Number of students enrolled:

    a. in the college, faculty or division as a whole,

    b. in the individual educational programs.

5. Teaching staff and teaching loads.

6. Commitment to and support for research.

7. Resources:

   a. financial: total budget, non-salary portion of budget and salary scales,

   b. physical: classrooms, laboratories, equipment and offices,

   c. support staff: administrative, clerical, laboratory, research and technical,

   d. library.

8. Curricular content of the program.

9. Actual course selections, as reflected by a sample of anonymous transcripts.

10. Innovative and special features of the program.

3.0 Glossary

Graduate Attributes: A set of program-level objectives describing knowledge and skills that students of academic programs should achieve by the time of graduation. In this document, a suggested set of graduate attributes is given in Section 4.1.

Interdisciplinary program: A program in which computer science is taught alongside one or more Other Disciplines to form an interdisciplinary degree program. Such programs may be ‘joint programs’, tightly integrating sets of courses from different disciplines. They may also involve a set of ‘building blocks’ (often called ‘majors’) that can be used to create a variety of interdisciplinary combinations that include computer science. Regardless of how programs are structured, it is essential that computer science is the ‘equal’ of the other discipline or disciplines; in other words, programs in which computer science is only a ‘minor’ are not eligible. In this document, unless otherwise specified, all general criteria, as well as computer-science-specific criteria, also apply to interdisciplinary programs.

Outcome: Something that is measurable that allows you to determine that an objective has been met.

Quality Indicator: Qualitative or quantitative data used to help assess whether an objective has been met.

Rubric: A document describing how an exam, assignment or other student activity should be evaluated, specifically mapping to the learning objectives that should be assessed.

4.0 Objectives and Outcomes

Each program must have a set of graduate attributes, describing what students should know and be capable of doing following graduation.

Institutions and individual programs may define their own graduate attributes, but it is strongly suggested that they start with the attributes presented in Section 4.1. Institutions may: adopt these verbatim; add to them; or reword some of them. If institutions use graduate attributes that are substantially different from those described in Section 4.1, then they must provide a document mapping them to those given in Section 4.1. For example, a typical situation in which it would be reasonable for an institution to add or refine a graduate attribute would be when the program focuses on preparing graduates to work in a particular domain (e.g., bioinformatics) or a particular sub-specialty of computer science (e.g., artificial intelligence).

4.1 Graduate Attributes

The following is the default set of program-level objectives for all CSAC-accredited programs. Since these refer to what students should know and be able to do following graduation, they are referred to as Graduate Attributes.

A graduate of a computer science or software engineering program must be able to:

1. Demonstrate Knowledge: Competently apply knowledge in a) software engineering, b) algorithms and data structures, c) systems software, d) computer elements and architectures, e) theoretical foundations of computing, f) discrete mathematics and g) probability and statistics.

2. Analyse and Solve Problems: Use appropriate knowledge and skills, including background research and experimentation, to identify, investigate, abstract, conceptualize, analyse, and solve complex computing problems, in order to reach substantiated conclusions.

3. Design Software and Systems: Design and evaluate solutions for complex open-ended computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, as well as economic, cultural, societal, and environmental considerations

4. Use Appropriate Resources: Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of their strengths and limitations.

5. Work Individually and in a Team: Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings

6. Communicate Effectively. Communicate with the computing community and with society at large about complex computing activities by being able to comprehend and write effective reports, design documentation, make effective presentations, and give and understand clear instructions

7. Act Professionally. Act appropriately with respect to ethical, societal, environmental, health, safety, legal, and cultural issues within local and global contexts, and with regard to the consequential responsibilities relevant to professional computing practice.

8. Be Prepared for Life-Long Learning: Learn new tools, computer languages, technologies, techniques, standards and practices, as well as be able to identify and address their own educational needs in a changing world in ways sufficient to maintain their competence and to allow them to contribute to the advancement of knowledge.

9. Demonstrate Breadth of Knowledge. Possess knowledge in areas other than computer science and mathematics so as to be able to communicate effectively with professionals in those fields.

The above attributes are derived from the following two sources and are designed to be as consistent as possible with these sources: The Graduate Attributes of the Seoul Accord, which is the international accord affording portability of accreditation among various countries, and the Graduate Attributes of the Canadian Engineering Accreditation Board. The latter alignment allows departments that wish to seek accreditation of their software engineering programs by both CSAC and the Engineers Canada Canadian Engineering Accreditation Board (CEAB) to reuse a substantial portion of CEAB self-study reports for CSAC purposes.

4.2 Quality Indicators and Outcomes

The following is the default set of program-level objectives for all CSAC-accredited programs. Since these refer to what Evidence should be provided that the graduate attributes defined for each program have been fulfilled. In other words, there must be evidence that what students actually know and are capable of doing following graduation correspond with the defined program-level objectives. This is achieved by *quality indicators*. These are qualitative and quantitative data gathered by the institution.

The institution should gather quality indicators in each of the following areas: Faculty (Section 5), Students (Section 6), Curriculum (Section 7) and Resources (Section 8). Suggestions for quality indicators are provided in each corresponding section. The self-study and accreditation process largely involves studying and verifying the quality indicators to ensure that the outcomes, in terms of education of students, correspond to the defined objectives.

4.3 Quality Enhancements

Evidence should be provided that the department seeks to better meet the defined objectives by regularly reviewing graduate attributes and the quality indicators, and then taking actions that should lead to better results.

5.0 Faculty

The heart of any educational program is the faculty. A competent, qualified, and forward-looking faculty gives an overall scholarly and professionally responsible atmosphere to the operation. An excellent faculty will usually identify and overcome problems in other areas and continue to provide a program worthy of accreditation, but no degree of excellence in other areas can continually offset the handicap presented by poor faculty quality or inadequate numbers of faculty. Thus, the first consideration for a program to be acceptable for accreditation is the presence and future assurance of a continuing critical mass of quality faculty. Educational institutions seeking CSAC accreditation of programs must have allocated the resources necessary to achieve a critical mass of quality faculty who are committed to professionalism, and must be committed to maintaining the allocations required for its continuation.

The proper size of the faculty depends on the enrolment and objectives of the program(s) being accredited, as well as factors that go beyond undergraduate education such as the amount of sponsored research, direction of graduate research, extension and continuing education activities, and involvement in professional and technical societies.

The number of faculty members must be large enough to provide a broad range of experience and capability and to provide meaningful technical interaction among the faculty members so as to support these interests. The faculty should for the most part occupy permanent positions to ensure continuity and stability. Institutions with limited enrolment and resources are encouraged to select and emphasize a smaller number of quality programs rather than to compromise standards by initiating or trying to maintain programs with inadequate faculty support.

To function effectively as teachers, faculty members must devote a significant amount of their time to seeking new understanding through research and scholarship, industrial interaction, instructional innovation, consulting, or other professional development activities. A significant common aspect of these activities is communication of ideas to other practicing professionals, scientists, and engineers outside the home institution.

Teaching loads must leave enough time for professional development of the faculty. Sabbatical leaves are important to faculty development, for they offer the individual an opportunity to develop professionally and allow for visiting faculty. Other evidence of institutional interest in faculty development, such as adequate resources for professional development, should be present.

Suitable quality indicators regarding faculty for the self-assessment and accreditation report include:

- The proportion of full-time faculty
- The teaching load (number of courses taught per year)
- Sustained levels of research and research grants
- Some indication that recent hiring is leading to faculty renewal
- Gender distribution of faculty
- Job satisfaction of faculty as expressed in interviews or surveys
- Student satisfaction with faculty as expressed in evaluations
- Distribution of faculty research areas and expertise over the topics mentioned in Graduate Attribute 1
- Knowledge and skills of faculty in areas corresponding to the other Graduate attributes, particularly professionalism (Graduate Attribute 8)

5.0 Faculty

To evaluate the quality of the faculty, the visiting team will examine the data presenting the quality indicators in the self-study report, as well as the CVs of the faculty members and any collective agreement. It will also meet with groups of faculty members. The team will gather further insights from discussions with staff, students and administrators.

The self-study report and the visiting team should focus primarily on the faculty teaching computer science and software engineering. For mathematics, and the Other Discipline(s) of an interdisciplinary program, the visiting team will not necessarily need to meet faculty members, nor examine a complete set of CVs, but will examine institutional policies and procedures to satisfy themselves that faculty members in these disciplines are comparable to faculty members in computer science. For example, the visiting team will need to ensure that, in general, the faculty members in mathematics and the Other Disciplines have appropriate backgrounds and teaching loads.

6.0 Students

An accredited program must have good students. Student selection and retention standards must be appropriate to the program. When students transfer from other institutions or from a branch campus, standards for evaluation and selection of these students should be clearly enunciated, and should show that these students are of similar quality and have substantially the same knowledge as those students who have taken all their work on the main campus. When computer science courses are regularly taken on other campuses, the main campus faculty should be involved in the development and assessment of curricular content.

A student advisory system is an important component in any educational program. The advisory system should embrace course selection and similar matters, and it should also include career guidance. Various aspects of professionalism and ethics (Graduate Attribute 7) may be dealt with through the guidance system. In an interdisciplinary program it is particularly important that students receive quality advice regarding all the disciplines involved in the program.

Curriculum and career guidance is best handled by well-informed faculty members who are given the time and administrative support for personal interaction with individual students. Both faculty members and advisors should be familiar with accreditation policies and guidelines, professionalism issues, professional certification, ethical codes of conduct (e.g., CIPS Code of Ethics and the CIPS I.S.P.).

The level of guidance needed will be a function of the flexibility of the curriculum in a particular school. Care should be taken by advisors not to assume responsibility of choice, which should be exercised by the student.

Suitable quality indicators regarding students for the self-assessment and accreditation report include:

- Feedback from employers, assessed through questionnaires that ask employers the extent to which students they hire possess the graduate attributes defined for the program
- Jobs offered for co-op and internship programs, and the proportion of students who find satisfactory employment following graduation
- Prizes and scholarships awarded, especially external ones
- Student’s satisfaction with their program and progress as assessed through questionnaires and interviews
- Attrition rates
- Admission averages
- Graduation averages

To assess the quality of students, the visiting team will interview groups of students and alumni, study transcripts and samples of student work, as well as analyse the data presented in the self-study report. Where possible, the team will also speak to employers, such as members of the department’s industrial advisory board.

7.0 Curriculum

In specifying criteria for achieving accreditation in any discipline, there is a relationship between a) establishing minimum standards or objectives, b) focusing on *achievement* of graduate attributes instead of being overly prescriptive, and c) encompassing flexibility within which students can set individual goals and tailor their programs. The Computer Science Accreditation Council has adopted the following criteria to strike a balance among the above.

This section refers to units of instruction called *courses*. Different institutions define courses and programs in different ways. For purposes of clarity in this document, a course is a single semester course, of roughly 12 weeks in length consisting of approximately 36 lecture hours. Thus a four-year undergraduate program would consist of roughly forty (40) courses.

It is important to note that although this document refers to courses, a specific program may meet a suggestion of a 'course in a specific subject area' by material covered in two or more courses.

In addition, although a course is defined as 36 hours of lectures, it is understood that pedagogical methods such as Problem Based Learning, studios, and online learning may to a greater or lesser extent be used instead of lectures.

An *introductory course* this document has no university-level course as a required prerequisite. An *intermediate or advanced course* has at least one university-level course as a required prerequisite. In many universities, introductory courses are referred to as 'first year courses', while an intermediate or advanced course might be referred to as an 'upper year course'.

It is recognized that institutions assign titles to courses in differing ways, and it is also recognized that courses are offered by different academic units, potentially within different institutions. In determining the extent to which courses meets the requirement specified herein, it is the *contents* of the courses, as evinced by the course outlines, learning objectives and teaching materials, which are of significance, and not the course titles or the particular academic unit offering the courses.

Note for Quebec Institutions: In Canadian provinces other than Quebec, a student typically obtains his or her degree after 16 years of study, including grades 1 through 12, plus four years of university. In Quebec, a student typically obtains his or her degree after 16 years: 11 years of primary and secondary school, 2 (general curriculum, pre-university) or 3 (technical curriculum leading to university) years of CEGEP, and 3 or 2 years of university. The criteria below (explained in terms of numbers of courses) are specified assuming a university program of four (4) years (40 courses). In order to satisfy the criteria, a program from Quebec may therefore include up to: 10 CEGEP courses for students who have obtained a (general curriculum) pre-university CEGEP diploma; 15 CEGEP courses for students who have obtained a technical CEGEP diploma (leading to a university program).

7.1 Evidence That Graduate Attributes Have Been Met

Central to outcome-based self-evaluation and accreditation is demonstrating that the Graduate Attributes, as discussed in Section 4, have been met. For the remaining subsections (Sections 7.2 to 7.9) demonstrable evidence should be provided that the program ensures that graduates have fulfilled each of the nine graduate attributes. Such evidence can include mappings from course-level objectives to graduate attributes, rubrics for assignments and tests indicating which graduate attributes are being assessed, and other quality indicators.

Learning objectives should be presented for core courses covering computer science or software engineering topics, as well as topics in other related disciplines where appropriate. These objectives should be derived from the graduate attributes (Section 4.1) and should describe what students will have achieved by the end of each course. Each objective should be expressed as at least a sentence, with an active verb. Verbs implying deeper learning, such as ‘calculate’, ‘design’, ‘evaluate’, ‘apply’, ‘solve’, ‘create’, ‘build’, ‘determine’, ‘develop’, ‘assess’, ‘use’, ‘lead’ and ‘present’ are preferable to verbs implying more passive learning, such as ‘know’ and ‘understand’, although the latter would be appropriate for some types of knowledge.

In the same manner that the graduate attributes should be approved and reviewed (Section 4.3), the learning objectives of courses should be approved by the department and reviewed periodically. Part of this review should be to ensure that the course learning objectives and graduate attributes remain in alignment. Evidence should be presented that the objectives are actually *applied* in courses. Such evidence might include that they are present in course syllabi distributed to students, and that textbooks, lecture notes, assignments and exams match the objectives. The use of *rubrics* for course activities is highly recommended. Rubrics describe what is to be expected of students in each course activity, and should indicate which learning objectives students would be demonstrating by successfully completing the activity.

Taken together, the objectives of the courses taken by each student, regardless of the path the student chooses, should satisfy the graduate attributes discussed in Section 4.1. To demonstrate this, the self-study report should contain a table for each required course (or course group) describing how the course contributes to each graduate attribute.

For a first accreditation (or a first accreditation under these revised criteria incorporating graduate attributes), it might be the case that learning objectives are developed for courses *retrospectively*. However, in subsequent accreditations it should be clear to the visiting team that the course level learning objectives are maintained and applied on an ongoing basis.

7.2. Overall Distribution of Courses

To meet the graduate attributes, it is expected that an accredited program will include the equivalent of forty courses of study (not necessarily all at the same institution) and lead to a Baccalaureate degree or equivalent. Furthermore, the program must ensure that *all* students must meet the graduate attributes to be eligible for the degree. The next two subsections describe the distribution of courses that would generally be expected for different types of programs. Prior to the era of outcomes-based accreditation, the numbers of courses were considered minimums. As of these current criteria, priority is given to demonstration that the graduate attributes described in Section 4.1 have been met. The numbers of courses listed can now be seen as guidelines only, and may be dropped in a future version of these criteria.

7.2.1 Expected Minimum Number of Courses for Computer Science and Software Engineering Programs

A. Fifteen (15) courses in computer science, software engineering or computer engineering (Graduate Attribute 1a through 1e; ; see Sections 7.3.1 to 7.3.3 for details)

B. Five (5) courses in mathematics or statistics (Graduate Attribute 1f and 1g; see Section 7.4.1), and

C. Ten (10) courses in subjects *other than* computer science, software engineering, computer engineering, mathematics and statistics (Graduate Attribute 9; see Section 7.5.1).

The set of courses in each area should exhibit some breath and some of the courses in each area should be at an intermediate or advanced level. This thirty-course requirement leaves each student with the equivalent of one year of study to satisfy additional institutional requirements and to accomplish personal objectives.

7.2.2 Expected Minimum Number of Courses for Interdisciplinary Programs

Ten (10) courses in computer science, computer engineering or software engineering (Graduate Attribute 1a through 1e; see Sections 7.3.1 and 7.3.4 for details).

B. Ten (10) courses from the Other Discipline or disciplines of the interdisciplinary program, with no discipline having fewer than five (5) courses (See section 7.5.2).

C. Three (3) courses in mathematics (Graduate Attribute 1f and 1g; see Section 7.4.2). In the case where mathematics is one of the other disciplines, the total number of math courses should be 10, not 13; in other words, courses in B and C can be double-counted.

D. Three (3) courses in non-technical disciplines, defined as disciplines other than computer science, natural sciences, mathematics and engineering (Graduate Attribute 9; see Section 7.5.3 for details). If one of the Other Disciplines is non-technical, then this requirement is already met; in other words courses in B and D can be double-counted.

The above requirements leave each institution and/or student with between 14 and 20 courses to satisfy additional program requirements (which may include more CS, more mathematics, and/or more of the Other Disciplines) and to accomplish personal objectives. However, to ensure that there is a balance between the Other Disciplines and computer science, the total number of courses required for the Other Disciplines combined should be similar to the number of computer science courses required.

7.3 Breadth of Coverage in Computer Science Topics

Similarly to the program as a whole, the computer science component of the program must offer a breadth of exposure to different topics, along with a depth of understanding.

7.3.1 Key Areas of Computer Science

All students in all accredited programs, including computer science, software engineering and interdisciplinary programs should be required to take courses which satisfy graduate attributes 1 (a), (b), (c) (d) and (e) described in Section 4.1. The following paragraphs expand on the range of knowledge and skills that students might acquire in the respective areas. Introductory courses in computer programming and computer usage would not be expected to meet these requirements. See sections 7.3.2 to 7.3.4 regarding specifics of how to apply this section to computer science, software engineering and interdisciplinary programs respectively.

This classification of five subtopics is not intended to limit the scope of computing nor the areas to be included or excluded at particular institutions, but rather it is illustrative of a range of topics that an accredited program would be expected to provide.

Graduate Attribute 1a: Software engineering, including requirements specification, software design and software architecture, software development, software testing, software maintenance, and other topics related to software process (For Software Engineering programs this list is expanded in Section 7.3.3).

Graduate Attribute 1b: Algorithms and data structures, including data structures such as stacks, trees, lists, queues, etc,; abstract data types, established solutions to classical problems (e.g., sorting and searching), and analysis of algorithms.

Graduate Attribute 1c: Systems software, including operating systems concepts, virtual memory management, management of distributed, parallel, and concurrent processes, transaction processing, logging, security, and computer networking.

Graduate Attribute 1d. Computer elements and architectures, including computer organization, digital device and communications technology, logical and physical hardware design.

Graduate Attribute 1e. Theoretical foundations of computing, including models of computation, analysis of algorithms, fundamentals of program specification and verification, computational complexity, grammars and automata.

7.3.2 Key Areas of Computer Science to be Learned in Computer Science Programs

The curriculum of a computer science program must require every student to have acquired knowledge and skills in each of the five areas in Section 7.3.1. An advanced skill level is not required in all five areas, but all students should acquire a reasonably broad knowledge of each area at the level typically provided by a first course in the area.

Advanced knowledge: In addition, each computer science student should acquire knowledge and skills at an *advanced* level, demonstrating both breadth and depth, in at least two key areas of computer science, and in one third of the suggested minimum number of computing courses described in Section 7.2.1a (i.e., 5 advanced computing courses). Advanced level knowledge and skills refer to those that would normally be attained from a second or subsequent course in the area.

7.3.3 Key Areas of Computer Science to be Learned in Software Engineering Programs

Software engineering programs must require every student to have attained broad knowledge and skills in each of the last four areas of Section 7.3.1 (Graduate Attributes 1b, 1c, 1d and 1e), and deeper knowledge in software engineering (Graduate Attribute 1a).

The following describes knowledge and skills in software engineering (Graduate Attribute 1a) that students in software engineering programs would be expected to have learned:

a) Software requirements (elicitation, analysis, specification, validation)

b) Software design and architecture (architectural structures, styles and patterns, structural vs. behavioural descriptions, strategies and methods)

c) Software construction and maintenance (abstraction mechanisms, assertions, debugging, coding style, programming environment and tools, including configuration management)

d) Software testing and quality assurance (unit, integration and system testing, black/white box testing techniques, validation and verification, reliability)

e) Software engineering management and process (metrics, estimating, planning, scheduling)

f) Applications of software engineering to various areas (embedded, real-time or distributed systems, database-centric systems, etc.)

g) Human computer interaction

h) Standards of Practice (Practice Guides, Recommended Practices, Practice Standards), including those endorsed by CIPS (e.g., ISO/IEC 12207)

It would normally be the case that the proper acquisition of software engineering knowledge and skills in these areas would imply the completion of at least three (3) courses in software engineering.

Breadth in key areas of computer science: In addition to knowledge of software engineering as defined above, students should acquire a reasonably broad knowledge in each of the other key areas of computer science defined in Graduate Attributes 1b, 1c, 1d and 1e. It is suggested that this be at a level typically provided by a first course in each area.

Advanced knowledge: Each software engineering student should acquire knowledge and skills at an advanced level, demonstrating both breadth and depth, in at least two key areas of computer science, one of which would be software engineering. Also each student should acquire advanced knowledge and skills in one third of the suggested minimum number of computing courses described in Section 7.2.1a (i.e. five (5) advanced computing courses). Advanced level knowledge and skills refer to those that would normally be attained from a second or subsequent course in the area.

7.3.4 Coverage of the Key Areas for Interdisciplinary Programs

All students in an accredited interdisciplinary program must have attained knowledge and skills in at least three of the five areas listed in Section 7.3.1. This knowledge and these skills must include Graduate Attribute 1b (Algorithms and data structures) and specifically include coverage of data structures, abstract data types and established solutions to classical problems (e.g., searching and sorting). A high level of skill is not required in all three of these topics, but all students should acquire a reasonably broad knowledge of each topic at the level typically provided in a first course.

Advanced knowledge: In addition, students in interdisciplinary programs should acquire knowledge and skills at an advanced level, demonstrating both breadth and depth, in at least two key areas of computer science (Graduate attributes 1a, 1b, 1c, 1d and 1e). Advanced level knowledge and skills refer to those that would normally be attained from a second or subsequent course in the area.

7.3.5 Exposure to Multiple Programming Languages and Paradigms

Students graduating from an accredited program should have proficiency in at least one programming language and exposure to a variety of programming languages. Exposure to a variety of programming paradigms – procedural, object-oriented, logical or functional, sequential and concurrent – is also important.

7.3.6 Significant Design Experience [Graduate Attribute 3]

Students graduating from an accredited program should have had the chance to develop a complete significant system, or make a major modification to an existing system, at some point in their studies, whether it be in course projects, a final 4th-year project, or an internship or in some other manner. This design experience should be open-ended in the sense that there is ‘no right answer’, and should enable the student to integrate their knowledge from most, if not all, of the areas of computer science listed in Section 7.3.1, as well as knowledge of mathematics, domain knowledge and, where appropriate, with consideration for economics, societal issues, safety, etc.

7.3.7 State of the Art Tools and Practices [Graduate Attribute 4]

Accredited programs must prepare students to meet the computing challenges they will face after graduation, whether they embark on careers immediately or continue their education. Thus, as part of their undergraduate education, students must be well-grounded in state-of-the-art computing practices. An accredited program should expose students to several computing configurations, including varied hardware, operating systems, and programming environments.

7.3.8 New Areas of Computing

Computer Science is a rapidly developing and growing subject. At the current time, these newer developments include such areas as mobile devices and networks, health and medical informatics, cyber security, bioinformatics, data mining, quantum computation, and so forth. While it is not to be expected that an accredited program will include material in all of these areas, accredited programs should nonetheless demonstrate that they recognize the rapidly evolving nature of the subject, and should include some of the newer areas of the subject, particularly within the intermediate and advanced courses which they offer to their students.

7.4 Mathematics [Graduate Attribute 1f to 1g]

7.4.1 Mathematics for Computer Science and Software Engineering Programs

Students in accredited computer science and software engineering programs must have attained knowledge and skills in the following three areas: (i) discrete mathematics, (ii) differential and integral calculus, and (iii) probability and statistics. Students are expected to receive a solid grounding in logic, Boolean algebra and graph theory. Other topics of particular relevance to computing should also be offered, including a selection from linear algebra, set theory and modern algebra, numerical analysis, differential equations, optimization, and queuing theory.

7.4.2 Mathematics for Interdisciplinary Programs

Students in accredited interdisciplinary programs must have attained knowledge and skills in the following two areas (i) discrete mathematics, and (ii) probability and statistics. In addition, students are expected to learn the basics of logic, Boolean algebra and the basics of graph theory.

7.5 Breadth and Depth in Topics Outside Computing and Mathematics [Graduate Attribute 9]

Computing applications can be found in all human endeavours, so education in any discipline outside computer science has the potential to prepare students in a field of direct relevance to their future livelihood.

The diversity of backgrounds needed by various computing professionals necessitates flexibility within these accreditation requirements. Innovation in establishing institutional requirements or in promoting each individual's ability to reach personal goals is encouraged. Nevertheless, this section identifies topics that enhance a program's ability to meet the needs of computing students. As before, the goal of breadth in exposure must be balanced by the goal of depth in understanding, which can best be achieved through the selection of complementary courses having a common focus.

7.5.1 Details of Breadth Requirements for Computer Science and Software Engineering Programs

Students in computer science and software engineering should be encouraged to choose courses such that their programs include sufficient breadth in other domains, so as to be able to communicate with people working in those domains, to develop software for those domains, and to in general be well-rounded graduates. It is suggested that to meet this requirement, students be required to take:

* Five (5) courses in science, engineering (but not computer or software engineering), or business

* Five (5) courses in humanities or social science

Topics in physics and electrical engineering are basic to many aspects of computing, and courses in these areas are particularly encouraged. Challenges such as the endeavour to map the human genome underline the value of education in other fields of natural science as well, including chemistry, the earth sciences, and the life sciences, especially when integrated with computer studies.

A thorough grounding in business fundamentals is important to prepare students of computer science to contribute to Canadian industry. Relevant knowledge and skills in this area include particularly accounting, business organization, economics, and auditing, but also include marketing, personnel management, and production management.

Students graduating in computing must also exhibit skills in non-technical disciplines. Through knowledge and skills in humanities or social science, students will gain understanding of political theories and processes, knowledge of individual and group social interactions, appreciation of cultures and history, sensitivity to the literary and fine arts, and fluency in languages. Not only is this important background for future self-study, but several aspects have direct bearing on particular areas of computing: linguistics is central to natural language processing, cognitive science provides mechanisms to evaluate human-computer interaction, law can be applied to assessing liability of computer professionals, and ethics helps to evaluate social implications of computing.

7.5.2 Details of the Other Discipline(s) for Interdisciplinary Programs

The Other Disciplines of interdisciplinary programs (B in Section 7.2.2) combined must be the 'equal' of Computer Science and must be designed to provide a coherent education. In order to achieve this, it is suggested that:

A. The courses in the Other Disciplines may include courses supporting those disciplines, and each Other Discipline may include courses from several related departments.

B. There be at least ten (10) courses in the Other Disciplines, with no single discipline having less than five (5) courses.

C. At least two (2) of the courses in each Other Discipline should be advanced courses, defined as courses that would normally be taught in the latter two years of study and build upon the introductory and intermediate courses.

D. There must be a structure to the set of courses required in the Other Disciplines; in other words, allowing students to choose any random set of 10 courses is not appropriate, although allowing students to select from several groups of electives would be fine. Allowing students to select a custom program would also be fine provided this process is carefully guided by an advisor, such that the resulting program is coherent and meets the program objectives.

E. It is highly desirable, although not essential, for courses in the Other Disciplines to build upon computer science and mathematics courses, and vice versa. For example, a course in another discipline could involve computing, and a course in computer science could present application programs related to the other discipline when illustrating computing concepts.

F. There must be an approval process in the institution for the sets of courses comprising the Other Disciplines, and there must be a process in place within the institution to regularly evaluate the curriculum in the Other Disciplines as well as the delivery of the courses.

7.5.3 Details of the Breadth Requirement for Interdisciplinary Programs

The breadth requirement of interdisciplinary programs (D in Section 7.2.2) ensures that students attain knowledge and skills beyond the purely technical. Through knowledge and skills in humanities or social science, students will gain understanding of political theories and processes, knowledge of individual and group social interactions, appreciation of cultures and history, sensitivity to the literary and fine arts, and fluency in languages. Not only is this important background for future self-study, but several aspects have direct bearing on particular areas of computing: linguistics is central to natural language processing, cognitive science provides mechanisms to evaluate human-computer interaction, law can be applied to assessing liability of computer professionals, and ethics helps to evaluate social implications of computing.

A grounding in business fundamentals is useful to prepare students of computer science to contribute to Canadian industry. Relevant knowledge and skills in this area include accounting, business organization, economics, and auditing, but also include marketing, personnel management, and production management.

7.6 Non-Trivial Problem Solving in Teams [Graduate Attributes 2 and 5]

A significant component of an accredited program must be practical in nature; students must have direct experience with non-trivial problem-solving. This component helps to develop students' creativity in solving open-ended problems through practice in formulating problem statements and specifications, considering alternative solutions, determining feasibility and cost, and communicating the results including detailed systems descriptions. Projects and courses that require teamwork are strongly encouraged.

In the case of software engineering programs, students must have direct experience with the major phases of the software development life cycle.

Whereas practical aspects should be included at all levels of the program, the major portion of the practical component is to be satisfied in advanced courses. Students, individually and as members of teams, must be required to design and implement a system, program, process, or device to achieve stated objectives.

The problem solution should include the establishment of objectives and criteria, analysis, synthesis, documentation, implementation, testing, and evaluation. It is desirable that problems include a variety of realistic constraints such as economic factors, performance thresholds, ergonomics, compliance with standards, interoperability with other systems, and conformance to ethical, professional and legal restrictions.

7.7 Written and Oral Communication Skills [Graduate Attribute 6]

As an adjunct to specific areas of instruction, the program must also include training to develop students' written and oral communications skills. Not only should students be taught to present information both verbally and in writing, but they should practice collecting information through reading and listening and assembling the information for various audiences. The realization of this requirement in the program can be tailored to suit other institutional goals, but the involvement of the computer science faculty must be evident.

7.8 Professionalism [Graduate Attribute 7]

Aspects of professionalism are to be emphasized throughout the curriculum. A specific course or courses in social implications of computing may be offered, but ethical and legal issues surrounding computing, including the social responsibility of programmers and computer users, must be emphasized in courses throughout the program so that students learn that these aspects are part of computing, not merely tangential disciplines.

7.9 Coordination of Program Schedules in Interdisciplinary Programs

A particular concern in interdisciplinary programs is that students are able to take the courses they need and want from the collaborating disciplines, without conflicting schedules unduly restricting their choice, or requiring the program to take excessive time to complete.

8.0 Resources

All the disciplines in an accredited program must have buildings, offices, laboratories, equipment, support staff, and fiscal resources that are appropriate for the characteristics of the program that is being undertaken. Evidence to this effect should be presented.

The availability of sufficient computing resources and support staff is of vital importance to the Computer Science aspect of the program. An appropriate variety of computational and network facilities must be readily accessible to all students and faculty, and access should be provided not only during scheduled laboratory class hours but also at other times.

The program must have competent administrative and technical support and services. Salary budgets must be consistent with the faculty size and student enrolment. Current expense budgets must allow reasonable amounts of travel and supplies. Computer budgets must allow students and faculty enough computer time that they use it as an effective learning aid.

There must be adequate access to electronic and other reference resources, such as the ACM and IEEE digital libraries. The collections must be maintained and refreshed so as to remain current, and there must be a breadth of materials included. Electronic networking sufficient to provide students and faculty access to external resources is also important.

Suitable quality indicators for the self-assessment and accreditation report include the following, all assessed relative to the student population

• Budget for resources

• Computers and software in labs

• Numbers and levels of expertise of technical and support staff

• Satisfaction of students and faculty with the resources available

• Sufficiency of the resources to teach the courses discussed in the Curriculum section, and to meet the Program Objectives

To evaluate the quality of resources, the visiting team will inspect them while touring the facilities, and will interview students, staff and faculty. The team will also study budgets and policies in place for ensuring the resources are maintained and replaced as they become obsolete.

9.0 Contact Information

CIPS Accreditation Secretariat

5090 Explorer Drive
Suite 801
Mississauga, Ontario
L5G 2E4
L4W 4T9
T. 905 602 1370
F. 905 602 7884
accreditation@cips.ca

www.cips.ca