

RESPONSE

to the

ABET Computing Accreditation Commission's DRAFT STATEMENT FOR REVIEW AND COMMENT

from the

COMPUTER SCIENCE DEPARTMENT WORCESTER POLYTECHNIC INSTITUTE Worcester, MA

Department Head: Dr. Michael A. Gennert
Email: michaelg@cs.wpi.edu

Primary contact: Dr. David C. Brown
Email: dcb@cs.wpi.edu
Telephone: (508) 831-5618
FAX: (508) 831-5776

Provost: Dr. John Orr
Email: orr@wpi.edu

Dates of Visit: November 2-4, 2008

Team Chairperson: Dr. Stuart Zweben

Program Evaluators: Dr. Jenq-Foung Yao
Dr. Mudasser Wyne

Response To: Dr. Stuart Zweben <zweben@cse.ohio-state.edu>
Dr. Harold Grossman <grossman@cs.clemson.edu>
Dr. Gale Yaverbaum <gjy1@psu.edu>
CAC, ABET <CAC@abet.org>

Response date: Tuesday, February 03, 2009

1. Introduction

This document constitutes the response of Worcester Polytechnic Institute (WPI) to the ABET CAC preliminary report that finds weaknesses in our Computer Science program.

1.1 Background: WPI

The WPI Computer Science (CS) Department supplies a broad general education using an innovative, non-traditional approach to undergraduate education which places much of the responsibility for the detailed design of a suitable undergraduate program on the student. As a consequence, WPI places a significant emphasis on high quality advising so that students can benefit from the flexibility our program provides. Programs are structured to emphasize the importance of mastering all the key courses. Courses and suggested sequences are reviewed on a yearly basis. In addition, required projects serve both to integrate previously taught material and to motivate independent learning.

A major aspect of WPI's innovative educational philosophy is that **there are no required courses**. Instead, WPI's high academic standards are met by the completion of Qualifying Projects and Distribution Requirements. Distribution Requirements specify sets of courses meeting curricular objectives. Students elect to take particular courses within each set in order to satisfy the requirements. These Distribution Requirements, the sets of recommended background courses for advanced courses, and the strong system of academic advising, ensure an effective, well-balanced program for all students.

1.2 WPI: CS Program Strengths

We greatly appreciate the visiting team's observation of some of our major strengths. We concur that the IQP and the MQP together provide important experiences that integrate course material and extend it. These significant projects, as well as a large number of other projects within courses, allow ample opportunities for the application of principles, as well providing important real-world experiences. Our graduates are sought after, and our alumni report that they have been extremely well prepared for life after WPI. We agree that our department's faculty members are strongly committed to undergraduate education, and that we operate in a highly collegial atmosphere.

1.3 Summary of Shortcomings

The ABET CAC team report that they found four areas of Program Weakness, and one Program Concern. The weaknesses are in Criterion 2, Criterion 3, Criterion 4 and Criterion 9, while the concern is also in Criterion 9.

1.4 Post-visit Communication

Since the CAC team's visit the WPI CS department has communicated with the team chair on several occasions via email.

The department provided a statement within seven days of the visit that was intended to provide corrections to statements of fact expressed during the exit interview. This document is included in Appendix A: WPI CS response to ABET Accreditation Weakness 4 (i.e., Concepts of Programming Languages). The department's assertion was

that our coverage of the concepts of programming languages was strong enough not to constitute a weakness.

Next the department informed the team chair that it had modified its Program Educational Objectives and Program Outcomes.

Finally we provided a summary of the coverage of our computer organization and architecture courses relative to the SIGCSE Computer Architecture syllabus. This was accompanied by a query about what exactly the key problems were that concerned the team about our department's coverage. This document can be found in Appendix B: WPI CS courses and the SIGCSE Computer Architecture syllabus.

2. Weakness: Criterion 2: Educational Objectives

On Tuesday December 16, 2008 the WPI CS Department discussed and passed a motion to change the Program Educational Objectives to read:

In support of its goals and mission, the WPI Computer Science undergraduate program's educational objectives are to graduate students who will:

- achieve professional success due to their mastery of Computer Science theory and practice;
- become leaders in business, academia, and society due to a broad preparation in mathematics, science & engineering, communication, teamwork, and social issues;
- pursue lifelong learning and continuing professional development;
- use their understanding of the impact of technology on society for the benefit of humankind.

These objectives now appear on the department's web pages at <http://www.cs.wpi.edu/About/objectives.html>.

They will be published in the addendum to the AY09-10 WPI Undergraduate Catalog.

These new objectives will be used in our web-based data collection process (see section 1.E., p.22, of the Self-Study report).

The CS department feels that this action removes the Criterion 2 weakness detected by the CAC visiting team.

3. Weakness: Criterion 3: Outcomes

On Tuesday December 16, 2008 the WPI CS Department discussed and passed a motion to change the Program Outcomes to read:

Based on the educational objectives, the specific educational outcomes for the WPI Computer Science undergraduate program are that by the time of

graduation CS majors will have achieved:

1. an understanding of programming language concepts;
2. knowledge of computer organization;
3. an ability to analyze computational systems;
4. knowledge of computer operating systems;
5. an understanding of the foundations of computer science;
6. an understanding of software engineering principles and the ability to apply them to software design;
7. an understanding of human-computer interaction;
8. completion of a large software project;
9. knowledge of advanced computer science topics;
10. an understanding of mathematics appropriate for computer science;
11. knowledge of probability and statistics;
12. an understanding of scientific principles;
13. an ability to design experiments and interpret experimental data;
14. an ability to undertake independent learning;
15. an ability to locate and use technical information from multiple sources;
16. an understanding of professional ethics;
17. an understanding of the links between technology and society;
18. an ability to participate effectively in a class or project team;
19. an ability to communicate effectively in speech;
20. an ability to communicate effectively in writing.

These outcomes now appear on the department's web pages at <http://www.cs.wpi.edu/About/outcomes.html>.

They will be published in the addendum to the AY09-10 WPI Undergraduate Catalog.

These new outcomes will be used in our web-based data collection process (see section 1.E., p.22, of the Self-Study report).

The CS department feels that this action removes the Criterion 3 weakness detected by the CAC visiting team.

4. Weakness: Criterion 4: Continuous Improvement

This weakness has been addressed by adding threshold performance criteria and by closing the feedback loop, as suggested by the CAC visiting team.

4.1 Action Taken: Threshold Performance Criteria

On Tuesday January 20, 2009 the WPI CS Department discussed and passed the following motion:

The WPI Computer Science Department moves to establish an 80% approval level against which to measure alumni evaluation of the department's

performance, using a yearly survey based on the department's adopted objectives and educational outcomes.

This motion establishes an initial threshold that can be used for evaluating alumni data. It also allows for the possibility of refining the threshold in the future. The performance thresholds that are reasonable and appropriate for our program are very hard to establish. We start this use of thresholds with something simple with which we can continue to evaluate our progress towards improving our program, as well as allowing us to evaluate the thresholds themselves and the procedures for their use.

4.2 Action Taken: Using Feedback to Guide Improvement

On Tuesday January 20, 2009 the WPI CS Department discussed and passed the following motion:

The WPI Computer Science Department moves that the Education Committee, with the cooperation of the Accreditation Coordination Committee, will on a yearly basis use the department's Threshold Performance Criteria to evaluate the collected alumni survey data, will propose appropriate action, will deliver their conclusions to the department, will monitor any resulting performance changes, and will document these activities.

We need to ensure that we make data evaluation a formal part of our department's standard practice, and clearly demonstrate that some program changes are triggered by this reflection. Note that we do expect some program changes to be unrelated to this simple feedback loop, just as they have been in the past.

The Education Committee is the department's formal body entrusted with considering changes to our undergraduate program, and consequently they need to play a role in acting during the continuous improvement process, and documenting that action.

The alumni survey is usually done in February with data analyzed in about April. As already reported, feedback to the department of these results, and subsequent discussion, is usually done via email as soon as they are available, during the department's retreat in the summer, and at the start of A term in a department faculty meeting. The Education Committee will establish a similar schedule for their evaluation, determination of appropriate remedial action, and feedback to the department. They will ensure that the process is documented, and forms part of the committee's yearly schedule of activities. This action will complement the department's existing procedure of regular MQP reviews and reports.

The CS department feels that this action, in conjunction with the action reported in section 4.1, removes the Criterion 4 weakness detected by the CAC visiting team.

5. Weakness: Criterion 9: Program Criteria

This weakness is due to the CAC visiting team's detection of inadequate coverage in the Computer Organization and Architecture area, and the Programming Language Concepts area.

5.1 Action Taken: Computer Organization and Architecture

The CAC draft statement points out that “a student may graduate without taking courses that cover several core computer organization and architecture topics”, naming the following examples: storage systems, main memory organization and operation, cache memories, instruction pipelining, buses, RISC/CISC and SIMD/MIMD.

Our coverage in CS core and advanced/elective courses of the topics mentioned above is as follows.

Storage systems: 3013 (and 4515)

Main memory organization and operation: 2011

Cache memories: 3013 (and 4515)

Instruction pipelining: (4515)

Buses: 3013

RISC/CISC: (4515)

SIMD/MIMD: (4513, 4515)

Note that while there are no *required* courses, both 2011 and 3013 are taken by more than 90% of our students, while 4513 and 4515, advanced/elective courses, are both taken by more than 50% of our students.

This means that *Instruction Pipelining*, *RISC/CISC*, and *SIMD/MIMD* are in fact not covered by the courses that almost all of our students take.

The space available in both 2011 and 3013 in which to add extra topics is *extremely* limited. We will add Instruction pipelining to 2011: this has already been agreed to by the current instructor. We will add SIMD/MIMD to 3013: this has already been discussed with the potential instructors. The remaining topic will be left for an advanced course to cover.

5.2 Action Taken: Programming Language Concepts

The CAC draft statement points out that “the coverage of programming language concepts in courses required of all students appears to omit topics such as parameter types, static and dynamic binding, and language translation phases.”

First, it should be pointed out again that WPI CS has no *required* courses. However, as was pointed out in the document entitled “WPI CS response to ABET Accreditation Weakness 4” (see Appendix A), provided a week after the visit, at least 95% of our majors take the 1101/1102, 2102, 2303 sequence of courses. That sequence covers parameter types as well as static and dynamic binding, as indicated by Table 1 of that document, and as presented in our course display materials during the CAC visit.

While “language translation phases” does not appear explicitly as a topic until the advanced course *CS 4533 Techniques of Programming Language Translation*, we expect students to at least be aware of ‘parsing’ after the introductory sequence, especially as 1102 includes implementing interpreters for basic domain-specific languages.

As a consequence of this we feel that no action is needed to address this portion of the weakness, as adequate coverage of programming language concepts is already provided.

The CS department feels that this information, in conjunction with the action reported in section 5.1, removes the Criterion 9 weakness detected by the CAC visiting team.

6. Concern: Criterion 9: Program Criteria: Lab work

We are pleased to see that the visiting team found no evidence that there was any student who graduated recently without having a laboratory experience in their science courses. However, as we have no *required* science/engineering courses, the team is correct that it “may” be possible for a student to avoid courses that provide a laboratory experience.

The CS program’s distribution requirements require five courses in Basic Science and/or Engineering Science. Courses satisfying the science requirement must come from the BB, BME, CE, CH, CHE, ECE, ES, GE, ME, PH and RBE disciplines. At least three courses must come from the laboratory sciences, BB, CH, GE, PH, and at least two courses are from one of these disciplines, ensuring some depth in at least one of these sciences.

The department feels that it is important to allow students to select the particular combination of science and engineering courses that best fits their carefully tailored program: to restrict their choice further would not be appropriate.

It should be noted, however, that WPI’s DESIGNS document,

<<http://www.wpi.edu/Admin/OAA/Designs2/design103.html#cs>>,

produced by the Office of Academic Advising with CS department guidance, is intended to provide advice to CS freshmen about their course selection. This document suggests that those freshmen should take Physics in both C and D terms (Spring 2009), thus providing a laboratory experience. The fact that no evidence was found that there was any student who graduated recently without laboratory experience suggests that our students do, in fact, follow our advice.

7. Conclusion

Once again the WPI CS department would like to thank the ABET CAC visiting team for their thorough and careful examination of our program. We appreciate that WPI's unusual and extremely flexible approach presents challenges to ABET visitors. We have taken the team's feedback very seriously. We feel that the information provided above, the actions already taken, and the actions still to be taken combine to address all of the issues raised in the draft statement.

**Appendix A: WPI CS response to ABET Accreditation
Weakness 4 (Concepts of Programming Languages).**

WPI CS response to ABET Accreditation Weakness 4

The visiting ABET/CAC team identified a weakness regarding Criterion 9:

Criterion 9. Students have the following amounts of course work or equivalent educational experience: Computer science: One and one-third years that includes:

- coverage of the fundamentals of algorithms, data structures, software design, concepts of programming languages and computer organization and architecture. [CS]
- an exposure to a variety of programming languages and systems. [CS]
- proficiency in at least one higher-level language. [CS]
- advanced course work that builds on the fundamental course work to provide depth.

The following paraphrases the comments we received from the ABET/CAC team:

The visiting team judged coverage of Concepts of Programming Languages to be marginally adequate as a consequence of dropping CS 2135 Programming Language Concepts. Although the team seems aware that material from CS 2135 has been incorporated into our new introductory course sequence, they were concerned that the main coverage of this material is in CS 4536 Programming Languages, which is only an elective.

Executive Summary

We believe that the claim that “coverage of concepts of programming languages is marginally adequate” is a factual error. It is inadequate to judge the coverage of programming language concepts solely by considering the number of students who take CS 4536 Programming Languages.

For the past five years the department has taught a revised curriculum that exposes students to increasingly more complex programming languages: in the revised three-course sequence, students learn to program in Scheme, Java, C and C++. In doing so, the students learn core programming language concepts and acquire the ability to understand the important tradeoffs between these languages. Our revised curriculum enables students to compare different languages and concepts over several courses, enabling them to reason about their strengths and weaknesses.

Response

For the 2004-05 Academic Year, the Computer Science department revised its introductory course sequence. The original four-course sequence was:

- CS 1005 (Introduction to programming) or CS 1006 (Object-oriented introduction to programming)
 - **Description:** *introduces structured programming with emphasis on modular design and functional decomposition (CS 1005) or introduces*

computer programming, with emphasis on object-oriented programs (CS 1006).

- CS 2005 Data Structures and programming techniques
 - **Description:** *continues the development of discipline in programming design, style and expression, and debugging and testing.*
- CS 2135 Programming Language Concepts
 - **Description:** *introduces the student to the fundamental concepts of programming languages, models of programming languages, and the basic concepts of language translation.*
- CS 2136 Paradigms of Computation
 - **Description:** *introduces students to advanced concepts in computational systems and programming languages and builds upon the functional approach to programming acquired in CS 2135. Topics include Object-oriented, logic programming, stream programming, and parallel systems and programming.*

The CS faculty, after assessment and evaluation, observed several problems with this sequence of 1000- and 2000-level programming courses:

1. The courses lacked sufficient emphasis on program design (as opposed to just programming). Program design skills are increasingly important both on the job and for Major Qualifying Projects (MQPs), especially those done for project centers and external sponsors.
2. CS 1005 had not been as accessible or interesting to novice programmers and non-majors as we would like.
3. The ideal path to courses such as CS 3733 (Software Engineering) was too long, leading many students to jump to 3000-level courses after 2005. While this was fine for some students, many lacked the programming skills for the 3000-level courses after just 2005.
4. Many students taking upper-level CS courses struggled with using pointers and other C/C++ features effectively. Often, the students in this situation had been novice programmers in 1005, so they spent more time in 1005/2005 worrying about data structures and programming, and didn't adequately master pointers.

To address these concerns, we adopted two core principles:

1. Focus on program design from the very first course.
2. Order the languages used in the programming sequence to cover languages with fewer core concepts before those with more complicated core concepts.

The revised curriculum now includes:

- CS 1101 (Introduction to Programming Design) or CS 1102 (Accelerated Introduction to Programming Design). We advise our first year students with “prior programming experience” to take CS 1102.

- **Description:** *introduces principles of computation and programming with an emphasis on program design. Students are expected to design, implement, and debug programs in a functional programming language. In addition, CS 1102 covers selected advanced topics in functional programming (such as macros, lazy programming with streams, and programming with higher-order functions).*
- CS 2102 Object-oriented Design Concepts
 - **Description:** *introduces students to an object-oriented model of programming. Building on the design methodology and the programming language concepts covered in CS 1101/CS 1102, this course shows how programs can be decomposed into classes and objects. Students are expected to design, implement, and debug object-oriented programs composed of multiple classes and over a variety of data structures.*
- CS 2303 Systems Programming Concepts
 - **Description:** *introduces students to a model of programming where the programming language exposes details of how the hardware stores and executes software. Building on the design and programming language concepts covered in CS 2102, this course covers manual memory management, pointers, the machine stack, and input/output mechanisms. Students are expected to design, implement, and debug programs in C and C++.*

CS 1102 is identical to the way CS 2135 was taught the three years prior to the introduction of the new curriculum. CS 1101 introduces basic data types (records, lists, and trees) and computation elements (functions, conditionals, recursion) using a functional language. CS 1102 adds more complex concepts, such as higher-order functions, macros, implementing interpreters for basic domain-specific languages and program transformations.

We believe the revised curriculum strengthens the knowledge of programming language concepts by our undergraduate students. An inherent danger in a curriculum with a single “Programming Language Concepts” course is that the material is covered as a “survey course” where students program for a week or two in each of several languages, leaving students to infer the differences between the languages. Our revised three-course sequence achieves the same coverage with greater depth.

Given the Programming Languages (PL) concepts associated with the Computing Curricula 2001 (<http://www.sigcse.org/cc2001/PL.html>), Table 1 describes where students encounter these concepts in our revised curriculum.

Table 1: Core PL topics from Computing Curricula Body of Knowledge

PL Topics	Coverage in courses
PL2: Virtual Machine	The concept of a virtual machine (CS 2102) Security issues arising from running code on an alien machine (CS 2102)
PL4. Declarations and types	The conception of types as a set of values with together with a set of operations (CS 1101, CS 2102) Declaration models: static binding (CS 2102, CS 2303), dynamic

	binding (CS 1101/02), visibility, functional scope (CS 1101/02), inner/anonymous classes (CS 2102), imperative scope (CS 2303), and lifetime (CS 2303) Overview of type-checking (CS 2102, CS 2303) Garbage collection (CS 2102)
PL5. Abstraction mechanisms	Procedures (CS 2102, CS 2303), functions (CS 1101/02, CS 2102, CS 2303), and iterators (CS 2102) as abstraction mechanisms Parameterization mechanisms by-value (CS 1101/02, CS 2102), by-reference (CS 2303) Type parameters and parameterized types (CS 1101, CS 2102) Modules in programming languages (CS 2102)
PL6. Object-oriented programming	Java (CS 2102) and C++ (CS 2303)

Table 2 contains enrollment information for CS majors who matriculated at WPI in September 2004 (with an expected year of graduation of 2008) and September 2005 (with an expected year of graduation of 2009). These are the first two groups of CS undergraduates to matriculate under the revised curriculum.

Table 2: Enrollment percentages

	1101	1102	2102	2303	4233	4536	536
Matriculated 2004	41%	35%	78%	74%	74%	46%	11%
Matriculated 2005	49%	49%	98%	95%	73%	25%	2%

Notes:

1. 74% of the 2004 cohort enrolled in CS 1101/02, CS 2102 *and* CS 2303.
2. 95% of the 2005 cohort enrolled in CS 1101/02, CS 2102 *and* CS 2303.
3. Nearly 75% of CS undergraduates complete CS 4233: Object-oriented Analysis and Design, which covers the OO paradigm in further depth.
4. CS 4536 Programming Languages is offered every other year

Given the evidence from Table 2, we are confident that a high percentage of our undergraduate students follow our advice and enroll in CS 1101/02, CS 2102, *and* CS 2303. They are exposed to programming languages with increasing complexity, furthering their understanding of core programming languages concepts. Along the way (and continuing into CS 2011 Introduction to Machine Organization and Assembly Language) the run-time environment shifts from using interpreters as virtual machines to using compilers for a specific operating system and hardware platform.

Another element of the revised curriculum is that the CS 1101/1102 courses are allowed to count as Computer Science credit for graduation. In the original curriculum prior to 2004, CS 1105 and 1106 were explicitly denied to count as Computer Science credit. By revising and enhancing the content of the first introductory Computer Science course, we believe we have strengthened the foundation of programming languages concepts so important to Computer Science majors.

We believe that the ABET/CAC team may have missed the synthesizing sequence of courses which provides suitable coverage of programming language concepts, as required by Criterion 9.

Contact: David C. Brown & Michael A. Gennert.
WPI CS, 11th November 2008.

Appendix B: WPI CS courses and the SIGCSE Computer Architecture syllabus.

This document outlines how WPI Computer Science courses address the topics in the ACM-SIGCSE syllabus for Computer Architecture.

WPI course number

CS- CS- CS- CS- CS- CS-
2011 2303 3013 4513 4514 4515

AR1. Digital logic and digital systems [core]						
• Overview and history of computer architecture						
• Fundamental building blocks (logic gates, flip-flops, counters, registers, PLA)	•					
• Logic expressions, minimization, sum of product forms	•					
• Register transfer notation	•					
• Physical considerations (gate delays, fan-in, fan-out)						
AR2. Machine level representation of data [core]						
• Bits, bytes, and words	•					
• Numeric data representation and number bases	•					
• Fixed- and floating-point systems	•					
• Signed and twos-complement representations	•					
• Representation of nonnumeric data (character codes, graphical data)	•	•				
• Representation of records and arrays	•					
AR3. Assembly level machine organization [core]						
• Basic organization of the von Neumann machine	•					
• Control unit; instruction fetch, decode, and execution	•					
• Instruction sets and types (data manipulation, control, I/O)	•					
• Assembly/machine language programming	•					
• Instruction formats	•					
• Addressing modes	•					
• Subroutine call and return mechanisms	•					
AR4. Memory system organization and architecture [core]						
• Storage systems and their technology						
• Coding, data compression, and data integrity						
• Memory hierarchy			•			
• Main memory organization and operations	•					
• Latency, cycle time, bandwidth, and interleaving						•
• Cache memories (address mapping, block size, replacement and store policy)						•
• Virtual memory (page table, TLB)			•			

• Fault handling and reliability			•			
AR5. Interfacing and communication [core]						
• I/O fundamentals: handshaking, buffering, programmed I/O, interrupt-driven I/O	•		•			
• Interrupt structures: vectored and prioritized, interrupt acknowledgment			•			
• External storage, physical organization, and drives			•	•		
• Buses: bus protocols, arbitration, direct-memory access (DMA)			•			
• Introduction to networks				•	•	
• Multimedia support			•	•		
• RAID architectures			•	•		
AR6. Functional organization [core]						
• Implementation of simple datapaths	•					
• Control unit: hardwired realization vs. microprogrammed realization	•					•
• Instruction pipelining						•
• Introduction to instruction-level parallelism (ILP)						•
AR7. Multiprocessing and alternative architectures [core]						
• Introduction to SIMD, MIMD, VLIW, EPIC						•
• Systolic architecture						
• Interconnection networks (hypercube, shuffle-exchange, mesh, crossbar)						
• Shared memory systems						•
• Cache coherence						•
AR8. Performance enhancements [elective]						
• Superscalar architecture						•
• Branch prediction						•
• Prefetching						•
• Speculative execution						•
• Multithreading						•
• Scalability						•
AR9. Architecture for networks and distributed systems [elective]						
• Introduction to LANs and WANs				•	•	
• Layered protocol design, ISO/OSI, IEEE 802				•	•	
• Impact of architectural issues on distributed algorithms				•	•	
• Network computing					•	
• Distributed multimedia				•	•	