Update of CS BS Program Self-Assessment

Prof. D. Petkovic, Chair, CS Department

Prof. W. Hsu

11/14/12

1. Introduction and background

In this report we present the updates and ongoing assessments we have implemented in our BS program in Computer Science after our last Accreditation Board for Engineering and Technology (ABET) accreditation visit in 2009. We have submitted an extensive self-evaluation of our MS program separately, in July 2012.

We had an ABET accreditation visit in 2007 which resulted in conditional accreditation, with two concerns that were to be clarified in a subsequent ABET visit. This subsequent visit occurred in 2009, and all remaining concerns were cleared. The Department obtained ABET accreditation until the 2013/14 school year.

The recent changes to our BS program were made as a result of:

- ABET accreditation visit and feedback
- Our own ongoing self assessment of the program
- Feedback from our students and transfer students
- Feedback from our most important “feeder” of transfer students, City College of San Francisco
- Ongoing feedback from our industrial partners
- CSU-wide and SFSU initiatives on “writing in the disciplines”
- CSU’s original LDTP (Lower Division Transfer Protocol) directive to streamline entry sequence courses; this involves facilitating the transfer of 12 basic CS units from community colleges. (This initiative has been abandoned subsequently, but we decided to implement it since it was beneficial to our transfer students.)

This report is organized as follows. In Section 2 we present a brief program overview (from the SFSU bulletin). In Section 3 we present some basic statistics about our program since 2009, such as enrolment, graduation rate, transfer rate etc. It is important to note that our enrolment grew considerably in the last 2 years (over 28% for the whole program) and that, in spite of the budgetary challenges, we have always managed to offer all core required courses every semester, and maintained excellent graduation rates. Section 4 covers motivations for and summary of

changes in our entry level course sequence (with new courses CSC 220, 230, 256, 340), outlines the procedures we use to assess achievement of learning objectives in these courses, and the results of these assessments. Section 5 presents our philosophy and experiences in the design and implementation of the "writing in the discipline" course CSC 300, satisfying GWAR requirements, as well as its initial assessment. Section 6 presents the motivations behind making mandatory our popular Software Engineering course CSC 640, and its initial assessment and review.

2. CS BS Program Overview (from SFSU Bulletin)

The primary mission of the Department of Computer Science is to prepare students for employment as computer specialists in the software development and computer manufacturing industries and for continued study toward advanced degrees. The department offers a broad curriculum covering the major areas of the computing discipline integrated with teamwork and group projects. Students are exposed to the fundamentals of computing architecture and computing theory, and focus their studies on the areas of software and system development. Faculty work directly with students to develop in them the skills and knowledge of computing professionals.

Department faculty are committed to teaching excellence. They remain current in the rapidly changing field of computing technology through continued research and publications, direct consultation with local industries, and seminar programs that bring top researchers to campus to speak on current developments in the field. Programs such as the Supervised Industrial Research Program and Practicum option provide opportunities for graduate students to complement their academic learning with real world experience.

The department has an additional commitment to sharing its knowledge and skills with the rest of the University community, with local schools, and with community based organizations of the Bay Area.

The Bachelor of Science program stresses a basic foundation in mathematics and physics, and a thorough study of the fundamentals of the discipline --- software development, computer architecture, operating systems, programming languages, algorithms, and the theoretical foundations of computer science. A wide variety of elective courses such as database systems, artificial intelligence, computer architecture and graphics, game development, and software engineering allow students to emphasize study in selected areas. Ethical and social issues of computing are discussed throughout the curriculum. Senior courses include teamwork and group

project assignments in order to better prepare students for the future job markets. A number of skill-based courses have been added to provide necessary training in the latest software technologies. The Bachelor of Science program is ABET accredited.

The Master of Science in Computer Science prepares students for a wide variety of careers in computing or related industries as well as for advanced study toward a Ph.D. Our program blends strong and practical education with high quality research and project activities. In addition to a general program covering the breadth of the computing discipline, the department offers three sub-specializations or concentrations: Software Engineering; Computing for Life Sciences; and Computing and Business. The concentration in software engineering covers the design and development of large yet useable software systems in a world where software teams are distributed globally. The concentration in computing for life sciences covers the development of computer applications and technologies aimed at supporting the burgeoning biotechnology industry. The concentration in computing and business provides a blended approach providing students with a solid technical background in computer science as well as core topics in modern business practices. The research component of our graduate studies has been significantly increased and involves students in pursuing research projects, writing papers and attending conferences. The program is supported by the SF State Center for Computing for Life Sciences (http://cs.sfsu.edu/ccls/index.html) and several specialized laboratories which provide research and project focus as well as opportunities to collaborate with faculty and students from other SF State departments and industry.

Bachelor of Science in Computer Science

Students intending to enter this program at the freshman level should have completed two years of algebra and one semester of trigonometry in high school. One year each of high school geometry and physics, as well as basic knowledge of computer organization and programming, are very desirable.

All lower division courses (course numbers below 300) included among the degree requirements are available at many community colleges in California; students intending to enter the program upon transferring to San Francisco State University from a community college should take as many of those courses there as possible.

Students should plan their program of study in the major with the help of a departmental adviser as soon as possible in order that the correct sequence of courses is taken and a proper set of electives is chosen. It is also suggested that students consult with an adviser before selecting courses to meet the General Education requirements. (See program below for acceptable science electives.)

Students are encouraged to participate in the Computer Science Cooperative Education Program at SF State. Under this program, they may obtain industrial employment related to their academic studies. This combination of on-the-job training and academic experience can greatly enhance the value of an undergraduate degree in Computer Science.

The Department has 11 full time faculty, four lecturers, and one office manager. The department website is at www.cs.sfsu.edu. A webpage with information on the undergraduate program, including program description and syllabi of all courses, is at http://www.cs.sfsu.edu/undergrad/undergraduate.html

3. Some relevant Statistics (BS and MS programs)

Table 3.1 shows some relevant statistics for the period 2009-2012.

Table 3.1: Computer Science Enrollment Statistics for Fall 2009 – Fall 2012

Year	Semester	# BS CS Majors	FTF Admits Enrolled	New Transfers Admits Enrolled	BS Degrees Awarded/ Applied*	#MS CS Majors	MS Degrees Awarded/ Applied*	Total FTES
2009	Fall	273	31	33	13	94	14	181
2010	Spring	249	2	3	25	102	13	153.1
2010	Fall	276	37	50	18	103	17	169.9
2011	Spring	267	1	20	18	100	9	168.3
2011	Fall	286	41	27	11	95	13	192.7
2012	Spring	306	1	25	18	87	11	193.5
2012	Fall	389	64	51	18*	93	5*	219.5

The table shows significant growth in overall enrolment over the last two years (appr. 14% each year), with the number of degrees awarded holding steady, in spite of budgetary challenges.

4. Changes and Assessments of new entry-level course sequence: CSC 220, CSC 230, CSC 256, CSC 340

Introduction

In this section we present changes to our entry-level course sequence that were implemented starting Fall 2010.

These changes were motivated by a number of important factors. They were developed by a CS Department Committee chaired by Prof. J. Wong, with Profs. B. Levine, W. Hsu, J. Dujmovic and D. Petkovic from 2008 to 2010. A majority of the CS Department voted in November 2009 to approve these changes. We have had extensive consultation with SFSU Computer Engineering and Mathematics departments, as well as with our main "feeder" for transfer students – the CS Department at City College of San Francisco. They all supported these changes.

The goals and motivation for these changes were:

- To modernize the CS program according to new Association for Computing Machinery (ACM) and Institute for Electrical and Electronic Engineers (IEEE) curricula guidelines, and feedback from ABET Accreditation
- To standardize our 200-level course sequence with the rest of CSU and major universities, thus making it easier to attract transfer students (this also satisfies the previously requested LDTP changes)
- To improve student graduation rate by restructuring MATH prerequisites in early course sequence

The newly proposed program has the same number of units as before (71). This includes 12 units that can be also counted as GE, thus conforming to maximum number of units allowed for the program.

We developed extensive transfer and transition plans, both for CS majors as well as for Computer Engineering majors and transfer students. The proposed changes took effect in Fall 2010. Details of the transition plans are available at http://cs.sfsu.edu/undergrad/transition.html . Our majors have for the most part been able to progress through the transition at a satisfactory rate. As can be seen in our enrollment statistics, the number of transfer students has increased significantly since the changes.

Motivations and Implementations

As mentioned earlier, our main motivations for revising the entry-level course sequence were two-fold: to modernize our program in line with recent ACM curricular recommendations, and to

standardize our program with the rest of the CSU and major CS programs in the country. We will address each of these motivations in turn.

Modernization of the program: The Association for Computing Machinery's recent curricular guidelines for Computer Science de-emphasized in-depth study of hardware, and increased the emphasis on programming and algorithms. We addressed these changes by replacing two hardware-oriented courses, ENGR 356 Basic Computer Architecture, and CSC 310 Assembly Language Programming and Intro to Computer Organization, with two new courses, the focused hardware course CSC 256 Machine Structures, and the software-oriented course CSC 340 Programming Methodology. These changes are also consistent with CS Department student and faculty feedback.

Standardizing our entry-level sequence: Prior to Fall 2010, our program had an ongoing problem: our entry-level sequence of 200-level courses did not conform to standard ACM guidelines, and was different from most other competitive schools. This required, for example, transfer students from both community college and other 4-year institutions to take an extra 3-unit class at SFSU as compared with, for example, San Jose State University. The differences were due in part to mixing lower and upper division course content in a way that prevented effective articulation. This hampered our recruiting of transfer students.

After the changes we implemented for Fall 2010, our entry-level sequence now conforms to ACM guidelines; articulation with all community colleges (along the lines originally proposed by the LDTP rules) is also very straightforward, without adding extra units. This has made us much more competitive in attracting transfer students, who today comprise 30-40% of our students. We have distributed the course content amongst the lower division courses in a way that better aligns with common practices in similar CS programs. The (approximate) mapping of the changes involving the entry-level course sequence can be summarized as follows:

CSC 330 replaced by new CSC 230

CSC 310 replaced by new CSC 256

CSC 213 replaced by new CSC 220

CSC 313 replaced by new CSC 340

In this proposed revision, the articulation with community colleges will be done by articulating CSC 210 (same as in current program), CSC 220, CSC 230 and CSC 256, for a total of 12 units. (These changes also conform to the LDTP requirements).

We addressed the shift of upper division courses to lower division (CSC 330 and CSC 310) by incorporating a new upper division course (CSC 340) to strengthen the computer science skills of students transitioning to upper division courses.

Before implementing our changes, we consulted with the Chair of the CS Department at City College of San Francisco, Prof. Constance Conner; she was very supportive of our changes in general and believed that they will help the transfer of CCSF students to SFSU.

Descriptions and Assessments of New Courses

In this section, we present objectives, descriptions, learning outcomes and assessments of each of the four revised courses: CSC 220, CSC 230, CSC 256, and CSC 340.

CSC 220 Data Structures

Objective: This course uses Java programming language to illustrate linear and non-linear data structures, including lists, stacks, queues, trees, tables and graphs. Also covered are recursion, iteration over collections and sorting and searching topics, including Big O notation and hash table.

MAJOR LEARNING OBJECTIVE	ASSESSMENT METHOD
Object-oriented programming: Java	1,2,3
ADT: bags and sets	1,2,3
Recursion	2,3
Introduction to computational complexity	2,4
Fundamental computing algorithms: sorting algorithms	2,4
ADT: lists, stacks, queues	1,2,4
ADT: priority queues	2,4
ADT: dictionaries	2,4
ADT: hash tables	2,4
ADT: trees	2,4

1-Programming Project; 2-Home work, 3-Midterm exam, 4-Final Exam

Midterm exam: Average points = 70, max points = 100
Programming Project Average: 80
Final examination: Average points = 70

Passed with grade C or better: 75%
Passed with grade less than C and must repeat the class: 10%
W or WU grade: 15%

CSC 230 Discrete Mathematical Structures for Computer Science

Objectives: The goal of this course is to introduce students to ideas and techniques from discrete mathematics that are widely used in Computer Science. We study topics in such areas as sets, logic, proof techniques, induction procedures, relations, functions, graphs, trees, combinatorics, and recursive procedures. All CSC majors should take this course.

Learning Outcomes: At the end of this course, students will be able to

- Solve mathematical problems with discrete objects by identifying and using appropriate discrete mathematical methods
- Understand and create proofs of simple concrete problems
- Use correct logical inference to conduct deductive proofs
- Define and analyze functions
- Choose an appropriate method for counting discrete objects
- Solve the shortest-path problem by using Dijkstra's algorithm
- Analyze time complexity of an algorithm given in a pseudo code by deriving its big Oh

Assessments:

Outcome	Assessment methods
Set theory and their applications	HW 1, midterm 1, final
Proof techniques	HW 2, 3, midterm 1, final
Basic logic for propositions and predicates	HW 3, midterm1, final
Relations and their applications	HW 4, midterm 2, final
Functions and their analysis	HW 5, midterm 2, final
Introduction to analysis of algorithms	HW 6, midterm 3, final
Graph theory and tree data structures	HW 7, midterm 3, final
Counting methods and theories	HW 2, 8, final
Recursive functions and their solutions	Final

Midterm exam #1: average =83/100; standard deviation=22.
Midterm exam #2: average =61/100; standard deviation=28.
Midterm exam #3: average =80/100; standard deviation=25.

Final exam: average point=94.6/145; standard deviation=37.5.

44 students registered in Fall 2011
Passed with grade C or better: 84.1%
Passed with grade less than C and must repeat the class: 9.1%
Failed the class: 6.8%
Approved withdrawals: 0%

CSC 256 Machine Structures

Objectives: Introduce students to various topics in computer architecture and machine language. The student who completes this course will encounter and understand differing computer architectures, become familiar with multiple computer instruction sets, understand the basics data representation and gain assembly language proficiency.

Learning Outcomes: At the end of this course, students will be able to

- Translate C/C++ code into assembly language
- Perform simple optimizations by hand
- Trace and debug at the assembly level
- Understand and extend simple CPU implementations
- Understand basic interrupt/exception handling
- Make simple performance estimates for assembly code

Assessments for Spring 2012:

Outcome	Evaluation
Translate C/C++ code into assembly language	HW 3, 4, quiz 2, midterm 1, final
Perform simple optimizations by hand	HW 3, 4, quiz 2, midterm 1, final
Trace and debug at the assembly level	HW 2, 3, 4
Understand and extend simple CPU designs	HW 5, 6, 7, 9, midterm 2, final
Understand basic interrupt/exceptions	Final
Make simple performance estimates	Final

Quiz 2 average: 81.5% (38 passed, 5 failed)
Midterm 1 average: 73.4% (30 passed, 13 failed)
Midterm 2 average: 85.8% (38 passed, 5 failed)
Final average: 71.8% (31 passed, 12 failed)

Overall: passed with C or above: 37, failed: 6

CSC 340 Programming Methodology

Objectives:

- Introduction and advanced concepts in C++
- Students will develop several medium sized programs in C++
- Cover widely-used searching and sorting algorithms
- Cover commonly used graph algorithms
- Enhance the students' programming skills

Learning Outcomes: At the end of this course, students will

- Be able to write medium-sized C++ programs utilizing STL and an integrated development environment
- Determine which of the common sorting and searching algorithms to utilize for given problems
- Be able to apply and implement graph algorithms in practice

Assessments for Spring 2012:

Note: the instructor has fleshed out the high-level learning outcomes in the syllabus, resulting in the more detailed outcomes listed below.

Outcome	Evaluation
Data types and binary encoding	HW 1, midterm
Control structures and basic algorithms	HW 2, midterm
Recursive functions and algorithms	HW 3, midterm
String processing and text file manipulation	HW 4, midterm
Procedural programming techniques	HW 5, midterm
Classes and object-oriented programming	HW 6, final
Pointers, dynamic objects, command line arguments	HW 7, final
Relative and binary files	HW 8, final
Important non-numeric algorithms	Final
Important numeric algorithms	Final

Midterm exam: Average = 60, standard deviation = 22, max = 100
Final examination: Average = 78.4

Passed with grade C or better: 60.8%
Passed with grade less than C and must repeat the class: 19.6%
Failed the class: 7.1%

Approved withdrawals: 12.5%

Changing of prerequisites for CS entry sequence

We continuously self-evaluate student progress in entry sequence courses and their ability to complete this sequence on time. Most 210 instructors had observed that students can succeed in 210 with a pre-calculus background; Calculus I was too restrictive as a prerequisite. Similarly, 220 and 230 instructors did not feel that Calculus II was necessary as preparation for 220 and 230. Hence, we revised the prerequisites for 210/220/230 effective Fall 2012.

The old prerequisites for each course were:

210 prerequisites: Math 226 (Calculus I)
220 prerequisites: Math 227 (Calculus II), CSc 210
230 prerequisites: Math 227 (Calculus II), CSc 210
340 prerequisites: CSc 220, CSc 230

The new prerequisites are:

210 prerequisites: Math 109 (Pre-calculus)
220 prerequisites: Math 226 (Calculus I)
230 prerequisites: Math 227 (Calculus II)
340 prerequisites: CSc 220, CSc 230, Math 227

This will enable CSc majors to start the core programming sequence one semester early, with more semesters over which to distribute the demanding technical classes in the major. In addition, it is now easier for non-majors to take 210, which is a recommended elective in many departments.

5. "Writing in the Discipline" GWAR Course CSC 300

This course was designed based on recommendations and requirements of a CSU-wide effort to institute "writing in the disciplines". Prior to the introduction of the GWAR requirement, CS students had to write a mandatory technical essay. This has now been replaced with our approved GWAR course CSC 300. We started offering it as a mandatory course every semester starting Fall 2010.

Objectives: Privacy and security, legal and ethical issues in software development. Communication relevant to software development such as reports, contracts, requirements, documentation, collaboration, e-mail, presentations. Study and use of basic tools for SW development and collaboration. This course satisfies the GWAR requirement

Expanded Description:

Introduction to ethical, social and professional issues in the field of computer science and software development. Professional and ethical responsibilities in computer science and software development. Privacy and personal data in on-line world.

Intellectual property rights and protection. Legal perspective, copyright laws, patents, trade secrets, trademarks, cheating and plagiarism, piracy, federal regulations, software protection, public domain, fair use, use of open software

Effective written communication and tools used in computer science and in collaborative software development such as:

- e-mail, forums, collaborative and group communication
- Executive summaries, abstracts and short technical reports
- Software contracts, requirements, specifications, use cases
- Software documentation
- Use of text in WWW
- Coding style
- Creation of effective technical presentations (e.g. using PowerPoint)

Intro to basic tools for SW development at SFSU

- SFSU IT environment and basic tools
- Brief overview of Windows, Mac and Unix
- Remote text editors
- Tools for basic SW development at SFSU CS Department
- Use of tools that might raise ethical and privacy issues
- Appropriate use of intellectual property

GWAR assessments

Learning objective	Assessment method
Evaluate ethical issues that are associated with accessing on-line information or developing software	Quizzes, written assignment (4-5 pages); in class discussions
Understand, evaluate and apply the various legal, privacy and intellectual property concerns related to on-line information and software	Quizzes, written assignment (4-5 pages); in class discussions
Use written communication effectively in the context of computer science and collaborative software development	Final paper (9 pages), 50% of the grade

For Spring 2012:

- 80% passed the quizzes
- Average grade 3.5

In Spring 2013 we plan to add several new writing topics/assignments such as executive summaries, abstracts, use cases and requirements, all in the context of SW engineering. Ideas and material for this has been adapted from similar topics taught as part of CSC 640 SW Engineering.

6. CSC 640 Software Engineering

There is now a consensus across industry and academia that to be successful in today's workplace, computer science students must learn practical Software Engineering (SE) teamwork skills. These are defined as the twofold ability: (i) to learn and effectively apply SE processes in teamwork settings, and (ii) to work as a team to develop satisfactory software (SW) products which have requested features, and are delivered on budget and on schedule. The need for improved teaching and training in this area is evidenced by statistics on the unacceptably high incidence of failure of industrial SW projects: about ten percent are abandoned, about one third fail, and over half experience cost and schedule overruns. The evidence also indicates that these failures stem primarily from failures in communication, organization and teamwork aspects of SE and are not due to the SW technology.

This observation resulted in development of one of the most important high level learning objectives (Objective 3) in our program, namely:

Students will be able to solve problems working in group settings. This translates to the following outcomes; students will demonstrate:

- *3.1 Knowledge of basic SW engineering methods and practices, and their appropriate application*
- *3.2 Knowledge and application of collaborative tools for SW development*
- *3.3 Successful implementation of teamwork behavior and policies in a large class project*

It has been a challenge for the academic community to teach classes that cover this topic in a practical manner, and also to provide reliable and effective assessment techniques. We started a SW Engineering class CSC 640 in 2003, with the main goal to address this learning objective, including teaching students how to work in globally distributed teams. In time, we expanded this class into a global development environment, by involving our partners from Fulda University in Germany and Florida Atlantic University. We also use this class to research and explore better methods of teaching practical SW engineering, including global SW engineering, and to develop effective assessment methods for teamwork learning. We have published 7 refereed papers on this topic and pioneered some innovative assessment methods. This work was the basis of our NSF TUES Award 1140172 in 2012 of $ 200 K for novel machine learning methods for assessment and prediction of teamwork learning.

In 2009, based on feedback from our ABET accreditation visit, and as part of the revision of our BS program, we made this class (CSC 640) mandatory.

The CSC 640 catalog description (from the SFSU bulletin): "Practical methods and tools for SW Engineering, including organizational teamwork".

We will outline our teaching methods, then list specific ways we assess the outcomes related to this learning objective. Specifically, we defined teamwork in SW engineering as the ability: (i) to

learn and effectively apply *SE processes* in a teamwork setting, and (ii) to work as a team to develop satisfactory software (SW) *products*.

Course objectives: Course will offer comprehensive and advanced coverage of practical methods and tools of SW Engineering, as well as its organizational, teamwork and communicational aspects. Special emphasis will be on Iterative, Incremental, Agile and User Centered Design methodologies and on global SW engineering, where teams are located in geographically and culturally dispersed areas. Students will engage in a group project in order to experience and practice key aspects of SW engineering in setting that simulates real SW company. Intended audience includes SW developers, tech leads and managers.

Expanded Description

· Introduction and motivation for Software Engineering

· Overview of several basic SE methodologies with emphasis on Iterative and Incremental

Development and User Centered Design

· Usability and UI design principles and practice

· Basic components of SW Engineering process: Planning; Requirements and Specifications;

Iterative Design, Rapid Prototyping, Mockups; Software Design; Coding and documentation

techniques (high level only);

· SW Engineering related to Web application development

· Open source SW development and management (NEW)

· Software Configuration Management, Delivery, Installation, and Documentation

· Software Metrics, Performance and Usability Measurements

· Software QA and Testing

· Software Maintenance

· Project Management issues

· Teamwork and Communication as integral part of SW Engineering

· Issues related to global SW engineering

· Basics of IP, licensing, digital rights management and copyright

· SW Engineering ethics

· Real life examples and cases from instructor and students

· Guest and student presentations

· Final Group Project including several milestones, interaction with instructor, and final demo

CSC 640, designed to simulate a real-life SW development environment, was started in 2003 at SFSU by Prof. Petkovic. Prof. Petkovic had observed the need for better SE teamwork skills from the students he hired during his tenure in the SW industry. Prof. Todtenhoefer (Fulda University, Germany) joined the teaching of CSC 640 as a globally distributed project in 2006, and Prof. Huang (Florida Atlantic University) joined in 2008. The course goals are to teach students practical and team-intensive SE, to expose them to working in global software teams, and to experiment with teaching methods and assessments to improve the instruction of SE teamwork. This course combines traditional classroom teaching of SE concepts, processes, methods and principles coupled with an intensive class project completed by students working in teams of 5 or 6. The instructors (PIs) play multiple roles as teachers, customer representatives, and CEOs/CTOs for the student teams, which are treated as small SW companies. The same SW development project (such as developing a new application for an online real-estate web site, a movie rental web site, or a restaurant reservation application) is assigned to all of the student teams. Each team works to complete the same project using the same SE tools. Students in the same team are awarded the same team grade at the end of the semester. The project requires completion of five well-defined, *team* milestones: *high-level design, low-level design, prototype, beta release,* and *final delivery with demo.* These are preceded by an *individual* milestone 0 where each student demonstrates proficiency in using basic SW development tools for the class project. Students meet under the supervision of the instructors and on their own. They are furnished with and required to use modern collaborative and development tools which are provided free of charge and/or made available via open source distribution. These classes have typically involved 50 to 75 students each Fall semester, working in about ten *local* and four *global* teams. The global teams are composed of students from multiple schools across three time zones and two continents.

As part of this joint effort, new methods for teaching and (qualitative) assessment of practical SE, as well as some preliminary analysis of factors distinguishing local and global student team dynamics, have been developed. Weekly on-line surveys are administered during the class to collect measures of the effort expended by each student in meetings, coding, and other project-related activities, along with student self-evaluations of their team's effectiveness and the factors that supported or inhibited teamwork and productivity. Simple manual analysis of surveys identified, for example, physical distance and time zone differences as inhibiting factors for global teams, consistent with observations of other researchers. During class and in meetings with student teams, PIs log weekly observations of team behavior which are then used for grading and assessment. The student learning outcomes, the assessments, and student team grading — all of which were independently developed by the PIs — incorporate both *process* and *product* components of teamwork as recommended by SW professionals, and are consistent with industry best practices. A rubric for grading and assessment of the students' actual application of SE teamwork best practices used in CSC 640 is presented in Table 6.1 below, and is assessed by observations and extensive data logging during the class by minimum of two instructors.

Table 6.1. Rubric for SE teamwork process and product for grading student teams in PIs joint classes

Measured Team Outcomes for SE *process*	Measured Team Outcomes for SE *product*
1. Fraction of the team participating at the meetings with the instructor 2. Quality and timing of follow-up on outstanding issues 3. Ability to deal with feedback constructively 4. Producing the non-SW and SW deliverables on time 5. Quality and completeness of non-SW deliverables (requirements, use cases, UI mockups, design documents, test plans etc.) 6. Number of teamwork issues that instructor had to deal with (whether resolved or not) 7. Ability to apply best SE process and teamwork practices 8. Ability to effectively use the SE tools (e.g. collaborative tools, version control, issue trackers)	1. Correctness and reliability of operation 2. Functionality of the product vs. desired requirements and use cases 3. Ease of use, user interface 4. Architecture of the developed system 5. Database design 6. Performance 7. Code quality (coding principles, style, documentation) 8. Presentation style and effectiveness of final product demonstration

The specific assessment methods and instruments used in CSC 640 for the learning objectives 3.1 – 3.3 listed above are shown in Table 6.2.

Table 6.2: Assessments for CSC 640 (data from Fall 2011)

Learning objective	**Assessment method**	**Comment**
Knowledge of basic SW engineering methods and practices, and their appropriate application	Final, in class, closed book exam Carries 45% of the class grade	Average 36/45
Knowledge and application of	Milestone 0 requiring students	98% pass in last 3 years

collaborative tools for SW development	to install and demonstrate use of basic SW development tools for class final project Carries 5% of the grade	
Successful implementation of teamwork behavior and policies in a large class project – ability to follow SW engineering *process*	See Table 1 – obtained by instructors' observations and monitoring Carries 25% of the grade	Average 23.75/25
Successful implementation of teamwork behavior and policies in a large class project – ability to produce SW engineering *product*	See Table 1 – obtained by instructors' observations and monitoring of final project demo Carries 25% of the grade	Average 23.13/25

We also received numerous e-mails form students thanking us for the class and stating that it either helped them get a job or in their current job.

7. Conclusions

In this report we presented the updates and ongoing assessments we have implemented in our BS program in Computer Science after our last Accreditation Board for Engineering and Technology (ABET) accreditation visit in 2009. This report, together with ABET accreditation reports provides ample information about our BS program. We have submitted an extensive self-evaluation of our MS program separately, in July 2012.

The report documents recent changes to our BS program such as:

- Change in our entry sequence (course modification streamlining of prerequisites etc.)
- Introduction of “Writing in the Disciplines” GWAR course CSC 300
- Introducing a mandatory course “SW Engineering” CSC 640 (previously was an elective)