Computer Science Assessment Plan

NOTE: The assessment plan and results are depicted in the Criterion 3 and Criterion 4 sections of this program's self-study for accreditation under ABET, Inc. These sections are on the following pages.

CRITERION 3. STUDENT OUTCOMES

A. Student Outcomes

List the student outcomes for the program and indicate where the student outcomes are documented.

The outcomes describe the core competencies expected of all Computer Science graduates from SDSM&T. The focus of the program is on preparing graduates for software development careers that emphasize mathematical, scientific, and engineering applications. The program also prepares students for their careers by emphasizing communication, teamwork, ethics, and by examining the local, global, and societal impacts of innovation and technological advancement. At the time of graduation, all students will:

1. possess a strong foundation in the software development process;
2. be able to solve problems using a variety of programming languages and have extensive experience with at least one high-level language;
3. have a background in computer hardware and experience with a variety of operating systems;
4. possess an extensive background in mathematics and an appreciation of the scientific method;
5. have an understanding of the theoretical foundations of computing;
6. have developed effective communication skills and have experience working with teams;
7. possess an understanding of professional, ethical, legal, security and social issues and responsibilities.

The department website http://www.mcs.sdsmt.edu/view.php?p=3601 is the primary repository for all assessment materials, including the program outcomes.

B. Relationship of Student Outcomes to Program Educational Objectives

Describe how the student outcomes prepare graduates to attain the program educational objectives.

While all outcomes support the educational objectives to some degree, the table below (Table 3.1) indicates the programmatic Student Outcomes which most strongly support each objective. Outcomes 1 (software development), 2 (language skills), and 3 (hardware and systems) are foundational skills required for any position in industry. These are essential for the achievement of Objectives 1 (mastery of the field) and 5 (life-long learning). The ability to pursue an advanced degree (Objective 2) and to do research or contribute to the field of computer science in other ways (Objective 4) is enhanced by Outcomes 4 (mathematics and science), 5 (theoretical foundation), and 7 (social responsibility). Leadership potential (Objective 3) is a difficult skill to teach but Outcomes 6 (communication skills) and 7 (social responsibility) are vital tools for an aspiring leader.

	Program Objectives				
SDM&T Computer Science Program	Graduates who have entered industry will have demonstrated a mastery of their field.	Graduates who continued their education beyond the bachelor's level will have the necessary background to successfully complete advanced degrees.	Graduates will have demonstrated their ability to assume leadership roles through career advancement or by assuming responsibilities beyond those expected of entry-level positions.	Graduates will be involved in their profession and make contributions to the field of computer science	Graduates will have the requisite foundation for life-long learning and will possess the skills to adapt and thrive in the rapidly-changing field of computer science.
Student Outcomes					
1. possess a strong foundation in the software development process;	X		X		X
2. be able to solve problems using a variety of programming languages and have extensive experience with at least one high-level language;	X				X
3. have a background in computer hardware and experience with a variety of operating systems;	X				X
4. possess an extensive background in mathematics and an appreciation of		X		X	X

the scientific method;					
5. have an understanding of the theoretical foundations of computing		X		X	
6. have developed effective communication skills and have experience working with teams;		X	X		X
7. possess an understanding of professional, ethical, legal, security and social issues and responsibilities.			X	X	

Table 3.1: Mapping Program Student Outcomes to Program Objectives

C. Process for the Establishment and Revision of the Student Outcomes

Describe the process used for establishing and revising student outcomes.

The primary stakeholders in the development and revision of the student outcomes are (1) industry (2) students and alumni (3) the faculty and (4) the university academic leadership.

Constituents	Mechanism for Participation
Industry	The industrial advisory board, recruiters
Graduates	Alumni surveys, departmental involvement
Current students	Senior exit surveys
Faculty	Curriculum committee
Academic leadership	Associate Provost for Academic Affairs and Assessment, Annual report

Table 3.2: Primary Stakeholders for Computer Science Program

1. *Industry.* The CS program has had an Industrial Advisory Board (IAB) since 1999. The board typically meets on campus every two years but provides input during the intervening time as requested. The IAB folder contains the PowerPoint presentations for each IAB meeting since its creation. The initial IAB presentations are made to the students, the faculty, and the administration (separately) allowing the IAB to incorporate feedback from each group into their final presentation. The final version

is submitted to the department and is an important tool in the assessment process. Each meeting begins with an examination of the program's mission statement, objectives, and student outcomes.

In Fall 2012, minor changes were made to the Objectives, and a substantial rewrite of the Outcomes occurred. The content of the Outcomes was not significantly changed but items were rewritten to reflect current trends and priorities. The shorter list was also perceived as easier for students to understand and less of a "laundry list." Table 3.3 lists the Outcomes prior to the IAB meeting and the revised outcomes. One recommendation from the IAB was to drop the references to "scientific computing" in the mission, objectives, and outcomes because it appeared too restrictive. The Curriculum Committee did not immediately implement this suggestion but considered the impact this would have on industry's perception of the special niche the program fills.

In February 2013, the Curriculum Committee voted to adopt the recommended change. Recruiters visit campus twice each year and meet with the Department Head, Dr. Riley, to discuss our program. Their input is another valuable source of information from both industry and alumni since many recruiters are graduates of SDSM&T.

Student Outcomes before 2012 IAB Meeting	Student Outcomes after 2012 IAB Meeting	Notes
1. have a strong foundation in the software development process;	1. possess a strong foundation in the software development process;	substance unchanged
2. be able to read and write program code in a variety of programming languages and have extensive experience with at least one high-level language;	2. be able to solve problems using a variety of programming languages and have extensive experience with at least one high-level language;	substance unchanged
3. have experience in programming for and using a variety of computer operating systems;	3. have a background in computer hardware and experience with a variety of operating systems;	Combined 3, 6
4. possess problem solving and algorithm development skills;	4. possess an extensive background in mathematics and an appreciation of the scientific method;	Old 4 deleted – redundant with 2. New 4 combines 8, 9.
5. have a strong understanding of the theoretical foundations of computing;	5. have an understanding of the theoretical foundations of computing;	unchanged
6. have a strong background in	6. have developed effective	combined 10,

computer hardware;	communication skills and have experience working with teams;	11
7. has the knowledge to produce effective conceptual and physical database systems;	7. possess an understanding of professional, ethical, legal, security and social issues and responsibilities	Old 7 deleted. IAB considered this too course-specific. New 7 combined 12, 13, mirrors ABET/CAC wording
8. possess an extensive background in computer-related mathematics;		
9. have an appreciation of the scientific method;		
10. have developed and practiced effective communication skills;		
11. have experience working in teams;		
12. understand and respect the professional standards of ethics expected of a computer scientist		
13. have an appreciation for the societal/ global impact of computing		

Table 3.3: Mapping Previous Student Outcomes to Current Student Outcomes

2. *Graduates.* The department has attempted to gather statistically valid data on alumni opinions about the program but, as is often the case, survey response rates are too low to be meaningful. The CS program is fortunate to have strong alumni involvement in a variety of ways which provides an alternate means of gathering input from graduates of the program. Innovative Systems has opened an office on campus with the express purpose of hiring current CS students as interns. Alumni who work for Innovative are an integral part of the department as senior design sponsors, guest lecturers, financial supporters, and advocates with the administration. The same is true for EROS Data Center, GoldenWest, Raven Industries, L-3 Communications, and private consultants in town. EROS and Raven have a campus presence, L-3 management visits the laboratories they have built on a regular basis, and alumni at Golden West have partnered with Innovative to assist our mobile development efforts. Given the geographic isolation of the program, the degree to which alumni of the program participate in setting the direction and ensuring the currency of the offerings continues to be a source of pride for the program. One benefit of being a

small department is that faculty members routinely maintain contact with graduates of the program. Although this is an informal mechanism, it has been a source of information about the perceived level of preparation for graduate studies. For example, when the curriculum committee considered dropping Theory of Computation from the list of offered electives, emails from graduates who completed advanced degrees were helpful in deciding to keep the course. While informal mechanisms are not scientific, they can provide supplemental guidance.

3. *Current Students.* The main mechanism for soliciting input from current students is through the Exit Interviews. All graduating seniors are invited to participate in a group discussion about the strengths and weaknesses of the program. Almost all of the graduating seniors will have had at least one industrial experience and can provide insightful comments about the program's ability to prepare graduates for industry careers. Student comments on the IDEA forms – the student evaluation surveys used at the end of a course – also provide valuable assessment information. The Department Head looks for common threads within a class, such as the request by students in Software Engineering to switch the class to agile development in preparation for senior design, and also for common themes across classes, such as a request for additional coverage of Linux earlier in the curriculum. Individual class issues are discussed with the faculty member as part of the annual review. Issues which span multiple courses in the curriculum are referred to the curriculum committee by the Department Head.

4. *Faculty.* All computer science faculty members serve on the CS Curriculum Committee. This is a cohesive group with considerable power to change the curriculum and the assessment program. The curriculum committee meets once per year to review and refine the Program Objectives and Student Outcomes. This meeting occurs after the IAB meeting, if one is held that year. As noted above, substantial changes to the Program Outcomes and minor changes to the Program Objectives were made in Fall 2012. Notes from the Curriculum Committee meetings are kept by Ed Corwin.

5. *Academic Leadership*. Dr. Kate Alley serves as the Associate Provost for Academic Affairs and Assessment. She is responsible for reviewing the Student Outcomes and Program Objectives to ensure that they align with the current strategic plan and stated institutional objectives. She is also responsible for monitoring program objectives for compliance with both ABET and HLC accreditation standards. Results of the Annual Assessment Report are communicated to her. Dr. Alley is a great asset to the departmental assessment efforts.

D. Enabled Student Characteristics

Characteristics a) through i) are not required student outcomes. They are, however, required to be enabled by your program. In other words, the program must provide every student with the opportunity to attain each characteristic. Indicate how the curriculum enables each characteristic.

The program has designed its Student Outcomes, and the Course Outcomes that support them, to enable the following characteristics in students about to graduate from the program:

(a) An ability to apply knowledge of computing and mathematics appropriate to the discipline

(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

(c) An ability to design, implement, and evaluate a computer-based system, process, component,
or program to meet desired needs

(d) An ability to function effectively on teams to accomplish a common goal

(e) An understanding of professional, ethical, legal, security and social issues and responsibilities

(f) An ability to communicate effectively with a range of audiences

(g) An ability to analyze the local and global impact of computing on individuals, organizations,
and society

(h) Recognition of the need for and an ability to engage in continuing professional development

(i) An ability to use current techniques, skills, and tools necessary for computing practice.

*(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.**

*(k) An ability to apply design and development principles in the construction of software systems of varying complexity.**

(*Additional outcomes for Computer Science Programs)

A mapping of Student Outcomes to the Enabled Characteristics is provided in Table 3.4. One of the primary measurements for assuring attainment of Student Outcomes is by monitoring student mastery of material in designated courses. Thus, the Course Outcomes from required courses which support the Student Outcomes are also provided. As noted in the table, all of the Enabled Characteristics are supported by multiple Course Outcomes. The philosophy of the department is that these characteristics need to be reinforced throughout the curriculum. Data that demonstrates the level of attainment across all four years of the curriculum is collected and analyzed by the Curriculum Committee.

ABET/CAC Characteristics	Enabled by	
	Program Student Outcome	Supported by Course Outcomes (required courses)
(a) An ability to apply knowledge of computing and mathematics appropriate to the discipline	2, 5, 4	Every required course contributes to some extent to this outcome.
(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution	2, 3	CSC 150, CSC 250, CSC 300, CSC 314, CSC 317, CSC 372, CSC 421, CSC

		456,CSC 465/467, CSC 461
(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs	1, 2, 3	Every required course (except CSC 110 and CSC 251) contributes to some extent to this outcome.
(d) An ability to function effectively on teams to accomplish a common goal	6	CSC 250, CSC 300, CSC 372, CSC 465/467, CSC 470
(e) An understanding of professional, ethical, legal, security and social issues and responsibilities	7	CSC 110, CSC 150, CSC 465/467, CSC 470, CSC 484
(f) An ability to communicate effectively with a range of audiences	6	CSC 250, CSC 300, CSC 372, CSC 465/467, CSC 470
(g) An ability to analyze the local and global impact of computing on individuals, organizations, and society	7	CSC 110, CSC 150, CSC 465/467, CSC 470, CSC 484
(h) Recognition of the need for and an ability to engage in continuing professional development	Objective 5 supported by Outcomes 1, 2, 3, 4, 6	Every required course (except CSC 110 and CSC 251) contributes to some extent to this outcome.
(i) An ability to use current techniques, skills, and tools necessary for computing practice.	1, 2, 3	Every required course (except CSC 110 and CSC 251) contributes to some extent to this outcome.
(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.	1, 2, 5	All advanced required courses contribute to this attribute to some extent. In particular, CSC 300, CSC 372, CSC 484, and senior design . Several electives also strongly support this attribute including CSC 445 Theory of Computation, CSC 423 Computer Security, and CSC 412 Cryptography.
(k) An ability to apply design and development principles in the construction of software systems of varying complexity	1, 2	All programming courses contribute to this attribute to some extent. Software engineering and the senior design sequence allow students to complete large projects in a team environment.

Table 3.4: Mapping of Student Outcomes and Course Outcomes to Enabled Student Characteristics

A complete list of required and elective courses, and how each contributes to the attainment of Student Outcomes, is provided in Table 3.5 (required courses) and Table 3.6 (elective courses). The columns numbers equate to the Student Outcomes listed in table 3.1. Full descriptions of the Course Outcomes for each course and the mapping of those outcomes to the Student. Outcomes are provided in the syllabi and in a condensed format in the supplemental assessment materials provided with the course displays.

Required Courses	S.O.#1	S.O.#2	S.O.#3	S.O.#4	S.O.#5	S.O.#6	S.O.#7
CSC 110 Intro. to CS				X			X
CSC 150 Computer Science I	X	X					X
CSC 250 Computer Science II	X	X				X	
CSC 251 Discrete Mathematics				X	X		
CSC 300 Data Structures	X	X	X	X	X	X	
CSC 314 Assembly Language	X	X	X	X	X	X	
CSC 317 Computer Organization	X		X		X		
CSC 372 Analysis of Algorithms	X	X		X	X	X	
CSC 421 GUI/OOP	X	X			X		
CSC 456 Operating Systems		X	X	X	X	X	
CSC 461 Programming Languages	X	X			X		
CSC 465/467 Senior Design	X	X				X	X
CSC 470 Software Engineering	X		X		X	X	X
CSC 484 Database Systems	X				X		X

Table 3.5: Mapping Course Outcomes to Student Outcomes (Required Courses)

Elective Courses	S.O.#1	S.O.#2	S.O.#3	S.O.#4	S.O.#5	S.O.#6	S.O.#7
CSC 410 Parallel Processing	X	X	X	X	X		
CSC 412 Cryptography				X			
CSC 415 Robotics			X			X	
CSC 416 Autonomous Systems				X			X
CSC 426 Computer Security			X		X		X
CSC 433 Computer Graphics	X	X		X		X	
CSC 445 Theory of Computation				X	X		
CSC 447 Artificial Intelligence	X	X			X		
CSC 449 Pattern Recognition				X			
CSC 464 Image Processing	X	X		X		X	

Table 3.6: Mapping Course Outcomes to Student Outcomes (Elective Courses)

CRITERION 4. CONTINUOUS IMPROVEMENT

This section of your Self-Study Report should document your processes for regularly assessing and evaluating the extent to which the student outcomes are being attained. This section should also document the extent to which the student outcomes are being attained. It should also describe how the results of these processes are being utilized to effect continuous improvement of the program.

Assessment is defined as one or more processes that identify, collect, and prepare the data necessary for evaluation. Evaluation is defined as one or more processes for interpreting the data acquired though the assessment processes in order to determine how well the student outcomes are being attained.

Although the program can report its processes as it chooses, the following is presented as a guide to help you organize your Self-Study Report.

A. Student Outcomes Measurement

It is recommended that this section include (a table may be used to present this information):

1. A listing and description of the assessment processes used to gather the data upon which the evaluation of each student outcome is based. Examples of data collection processes may include, but are not limited to, specific exam questions, student portfolios, internally developed assessment exams, senior project presentations, nationally-normed exams, oral exams, focus groups, industrial advisory committee meetings, or other processes that are relevant and appropriate to the program.
2. The frequency with which these assessment processes are carried out.
3. The expected level of attainment for each of the student outcomes.
4. Summaries of the results of the evaluation process and an analysis illustrating the extent to which each of the student outcomes is being attained.
5. How the results are documented and maintained.

Assessment is an institutional priority as noted in the Strategic Priorities provided in Section 2: "Ensure a legacy of excellence through dedication to continuous quality improvement." All faculty members are required to participate in assessment activities, and the quality of that participation is a component of the annual performance review. Faculty are required to provide data for the assessment instruments described in this section, participate in the IAB and the focused curriculum reviews, and complete course-embedded assessments on the prescribed schedule.

Ten data-collection instruments have been selected by the department for use as the centerpieces of the assessment process. Additional data-collection instruments are used, but the following ten instruments are systematically applied and the data analyzed, discussed, and acted upon according to a program-wide schedule:

1. Major Field Test (MFT) results
2. Industrial Advisory Board (IAB) feedback

3. Structured Course-Embedded Assessments
4. Capstone Project Evaluation
5. Senior Exit Interviews
6. Outstanding Recent Graduate Awards
7. Student Competition Involvement
8. Co-op reports
9. Undergraduate Research Involvement
10. Placement rates/ Starting Salaries

The department has included alumni surveys in this list in previous years, but it has become increasingly difficult to collect a sufficient number of responses to produce meaningful statistics. The original survey was six pages long and had a very low return rate. The CS faculty pared the survey down to two pages, but the response rate was still low. An online survey tool was tried but did not increase response rates. Dr. Riley instituted a newsletter as a mechanism for increasing alumni engagement which may produce results in the next year or two. At the moment, the alumni prefer responding to individual emails and a significant amount of feedback is gained in this way. The department will continue to search for ways to increase participation by alumni.

The Annual Assessment Report (AAR) is the mechanism by which all data (qualitative and quantitative) yielded by the ten instruments listed above (and described below) are brought together and used by the faculty as a whole to achieve the following:

- A review of the overall 'health' of the program and creation of a "state of the department" annual summary that includes the findings of the annual Focused Curriculum Review (FCR)
- A review of student attainment as measured by the assessment data gathered that year
- Documentation of the maturation of our assessment instruments and the effectiveness of the course-embedded assessments

Supplementing the data yielded by the ten instruments listed above are data and input gathered or received on an opportunistic basis with relevance to specific outcomes or program performance. Examples of such additional data include information on faculty stability, enrollment, statewide initiatives, and shifts in campus leadership or priorities, etc.

Implementation of the AAR was motivated by departmental preparations for the 2007 ABET/CAC visit and our experience of the value of performing a "state of the program" annual review. Copies of the AAR are provided in the accompanying documentation. The 2013 ABET/CAC review process will serve as the 2012-2013 review.

1. Major Field Test (MFT)

Type of measurement – Objective.
Frequency of collection – Every year.
Frequency of analysis – Every three years. Although scores are reviewed every year, action is not taken as a result of any one year's results.

How collected and from whom – The exam is given as part of the Senior Design course. All students enrolled in Senior Design are required to take it, and a portion of the grade is based on the student's participation.

Benchmarks – The exam provides feedback in three areas.

Programming Fundamentals – We believe our students receive a solid foundation in programming with a degree of rigor that exceeds that of many other schools. Thus, our benchmark is to be above average in this category and to achieve the 80% mark.

Computer organization, architecture, operating systems – We believe our students should be somewhat above the national average in this category. Our benchmark is 70%.

Discrete Structures, Algorithms, theory – We believe our students should be above average in algorithms and mathematics but average in theory. Our benchmark is an aggregate score of 75%.

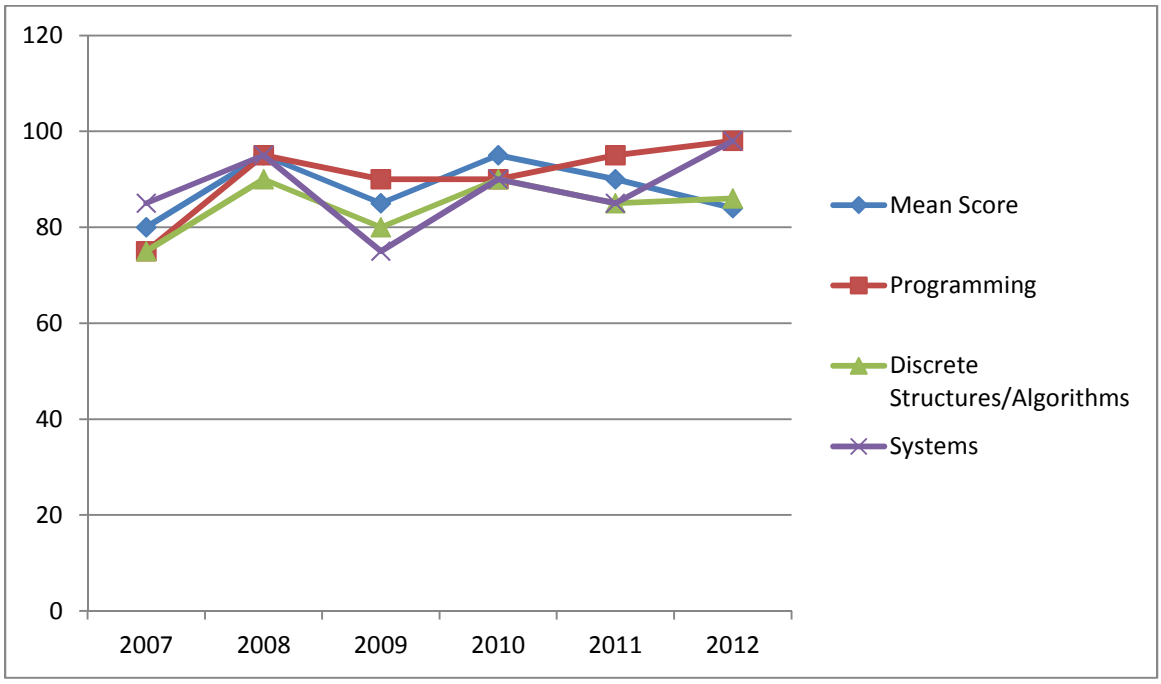

Table 4.1: Summary of MFT results from 2007 – 2012
All students in Senior Design are required to take the MFT. The scores reported are for approximately 15-20 students per year.

As noted above, all students enrolled in Senior Design take the MFT (i.e., between 15 and 20 students per year); nonetheless, no one year provides sufficient information to motivate change. The averages for the past six years are: Programming – 91%, Discrete Structures and Algorithms – 85%, Systems – 88%, and an overall average of 88%. The faculty reviewed these results annually and considered three-year averages of student performance as part of the focused curriculum reviews. The MFT scores have not indicated weaknesses in our program that require remediation.

2. Industrial Advisory Board (IAB)

Type of measurement – Objective/Subjective. Objective feedback is provided by the participants through statistics on the numbers and types of jobs our alumni have in different sectors of the industry. Subjective feedback is provided on how well prepared graduates are to enter the workforce.
Frequency of collection – Every two years.
Frequency of review – Every two years.
How collected and from whom – Alumni and other industrial contacts come to campus for intensive sessions focusing on curriculum, student preparation, and other issues identified for a given meeting. Representatives are chosen from diverse companies and market segments within the industry (defense, networking, algorithm development, database-driven, hardware-driven, etc.).
Benchmarks – At each meeting, the council is asked to identify the next big change taking shape in the industry and to define how well SDSM&T is positioned to take advantage of the coming changes. They also review the program's objectives, outcomes, and mission statement and provide feedback to the State of the Department presentation. The benchmark is to receive a "passing" score on this question. That is, the goal is for the IAB to determine that the curriculum is evolving in the right direction and only minor course corrections are required to keep the program on track to address developing needs in industry.

Year	Recommendations
2012	Delete database and networking from outcomes Database earlier in the curriculum Add networking Increase exposure to programming paradigms Add mobile computing elective More software engineering tools, unit testing Add multi-year projects Increase oral presentation opportunities
2010	Add database and networking to program outcomes Define focus areas within CS Improve collaboration with CENG Add networking
2008	Multi-semester projects More focus on software engineering

Table 4.2: Summary of IAB recommendations from 2008-2012

A full set of IAB notes and recommendations is available in a separate binder. Table 4.2 summarizes the recommendations since the last ABET visit. All recommendations are seriously considered, and, after a feasibility analysis, some are selected for implementation and review. The recommendations that have been implemented are discussed in more detail in Subsection B, Continuous Improvement, below.

3. Course-Embedded Assessments

Type of measurement – Objective/Subjective.
Frequency of collection – Rotation provided below (Table 4.3).
Frequency of review – Course dependent. Electives and some upper-level required courses are reviewed every two years. Lower-level courses or courses with significant content changes are evaluated more frequently. Measurement of attainment can be done in a variety of ways including selected test questions, programs, presentation rubrics, and other graded events. Course-embedded assessments are used as part of the annual faculty performance review process by the Department Head and by the CS Curriculum Committee as part of the annual curriculum review.
How collected and from whom – At the start of every semester, the faculty member reviews the course outcomes for an assigned course and documents the assessments designed to specifically address each outcome. The faculty member collects the data; completes the "Course-Embedded Assessment Worksheet" (to include the results, an evaluation of the results, and comments on improvements, changes, or experiments); and posts the final course assessment report on a shared drive. The uniformity and simplicity of the worksheet allows all faculty members to easily review and understand the results of each semester's cycle and sets the stage for discussion and group analysis of the sufficiency of coverage or student attainment of a particular skill.

Use of course-embedded assessments has evolved as the faculty explores the right balance between uniformity of standards and individual faculty member autonomy and creativity. Initial work on carefully defining outcomes resulted in a large number of overly specific, topic-focused outcomes (e.g., be able to write a bubble sort). As the faculty have used this technique, shared and discussed the effectiveness of various assessments, and revisited the task of designing course-embedded assessments semester after semester, the process has matured. Outcomes have been rewritten to reflect broader categories of knowledge not tied to a particular textbook or teaching approach, and the faculty has experienced the value of the expanding archive of Course-Embedded Assessment summaries (and accompanying materials). The archive and numerous examples of the forms discussed here will be available to evaluators.

Benchmarks – The faculty reports statistics on student performance on key 'graded events' throughout a course. To ensure that assessments are meaningful (i.e., can be analyzed over time even as techniques are continuously improved), the faculty focuses on the percentage of students demonstrating acceptable competence. Grades per se are not used as assessments; however, a 'graded event' (e.g., specific test questions, programming assignments, etc.) may serve as the assessment of a specific outcome and also contribute to a grade.

Because grades are the lingua franca of faculty members, acceptable competence is defined variously (e.g., as 70% or better, as a "C" or better, etc.). Regardless of how it is defined on a scaled score or via a grade, the program consensus is that "acceptable competence" means the outcome has been sufficiently mastered to provide a solid basis for ongoing learning and further mastery at more advanced levels. A shared focus on "acceptable competence" facilitates faculty attention to the 'big picture' and helps it

avoid an overly rigid, numbers-driven evaluation process that can preclude the benefits that come from having a small, integrated, cooperative faculty apply professional judgment to the evaluation of results across multiple sections or across clusters of closely related outcomes.

A pattern of poor performance on a topic across multiple semesters, sections, or professors motivates shifts in material coverage throughout the curriculum. For example, consistent difficulty with file handling across all sections of CSC 150 resulted in increased coverage in CSC 250. Course-embedded assessments in CSC 150 revealed this weakness.

	Fall 2011	Spring 2012	Fall 2012	Spring 2013	Fall 2013	Spring 2014
CSC 110			X			
CSC 150	X	X		X		X
CSC 250	X	X			X	
CSC 251	X	X	X		X	
CSC 300	X	X		X		X
CSC 314				X		
CSC 317				X		X
CSC 372	X				X	
CSC 410				X		
CSC 412	X				X	
CSC 415					X	
CSC 421	X				X	
CSC 426				X		
CSC 433			X			
CSC 445						X
CSC 447		X				X
CSC 449			X			
CSC 456		X		X		
CSC 461		X				X
CSC 464				X		
CSC 465	X				X	
CSC 467		X				X
CSC 470		X				X
CSC 484	X			X		
Mobile					X	
Networking						X

Table 4.3: Schedule for course-embedded assessments

Two fundamental questions are addressed when reviewing the course-embedded assessments:

1. How did the program perform overall in ensuring each outcome is attained—as evidenced by the results of all embedded assessments for the specific outcome?

2. How did each course compare to (multiple) previous offerings of the course?

No one data point is compelling in this analysis. Rather, the faculty looks for trends or significant changes in attainment (positive or negative) that may have resulted from changes or experiments in the course. Tables 3.5 and 3.6 in Section 3: Student Outcomes provide mappings between course outcomes and Student Outcomes.

Each outcome is evaluated to determine the percentage of students acquiring acceptable competence. As described above, this standard can be referenced by the scaled score of 70 or better or by the grade of "C" but the standard must mean sufficient mastery to enable continued learning at increasingly advanced levels. To maintain and reinforce this standard, each faculty member's course-embedded assessment reports were incorporated into the annual review process in beginning in 2011. The department head reviews and provides formal written feedback on the sufficiency and effectiveness of the embedded assessments and the usefulness of data yielded by the assessments to the overall program-wide task of evaluating attainment of each outcome.

Initially, course-embedded assessment reports and the department head's feedback on the same were used primarily in the annual evaluation process. However, by 2012-2013, sufficient data had been collected to allow for the aggregation of multiple reports and meta-analyses of results. Table 4.4. shows such an aggregated analysis. The scaled score of 70 or greater was used as our numerical reference for "acceptable competence" in the analysis shown in Table 4.4. The results were reviewed during a summer Curriculum Committee meeting in 2013, but will become part of the start-of-term Curriculum Committee meeting agenda in subsequent years.

The low achievement in CSC 150 is somewhat of a concern, but the nature of the course (i.e., it is a service course for engineering majors who are not interested in programming) makes those numbers understandable.

The results for SO #3 and SO #4 are the lowest, but still above the program benchmarks with over 70% of students achieving the required mastery for those outcomes. However, the committee did examine the content and achievement in the courses contributing to those outcomes. As noted above, the program has consolidated its coverage of hardware topics into three courses, Assembly Language, Computer Organization and Architecture, and Operating Systems. The change was motivated by the assessment process, but performance in this area must be monitored to determine if too large a change was made to this aspect of the curriculum.

CSC 300 and CSC 470 contribute to the "variety of operating systems" portion of SO #3. The results are consistent with student surveys which indicate a greater need for Linux earlier in the curriculum. Students master Linux by the end of their junior year but struggle in the sophomore year. Therefore, additional coverage of Linux in CSC 300 is being considered. As always, it is dangerous to draw conclusions from a single data point, thus the faculty will closely monitor these items in the 2013-2014 academic year. While special attention will be paid to SO #3 and SO #4 in the next iteration, the initial review of the performance averages indicates that the curriculum as a whole is enabling students to achieve the programmatic Student Outcomes.

	SO #1	SO #2	SO #3	SO #4	SO #5	SO #6	SO #7
CSC 110							85
CSC 150	63	64					85
CSC 250	88	76				80	
CSC 251				64	76		
CSC 300	70	82	70	80	80	85	
CSC 314		74	74	73	75		
CSC 317			80	73	87		
CSC 372	85	80		75	70	85	
CSC 421	93	93			93		
CSC 456			80		66		
CSC 461	100	100			100		
CSC 465	95	100				83	100
CSC 470	90		80		80	95	85
CSC 484	86				74		81
Averages	85	84	77	73	80	86	87
Std. Dev.	12	13	5	6	10	6	7

Table 4.4: Summary of student attainment supporting student programmatic outcomes

1. possess a strong foundation in the software development process;
2. be able to solve problems using a variety of programming languages and have extensive experience with at least one high-level language;
3. have a background in computer hardware and experience with a variety of operating systems;
4. possess an extensive background in mathematics and an appreciation of the scientific method;
5. have an understanding of the theoretical foundations of computing;
6. have developed effective communication skills and have experience working with teams;
7. possess an understanding of professional, ethical, legal, security and social issues and responsibilities;

4. Capstone Project Evaluation

Type of measurement – Objective/Subjective. The capstone experience requires a year-long team project, typically with an industry sponsor. This is a critical component of the curriculum. Students are evaluated by multiple stakeholders on teaming, written and

verbal communication skills, and professionalism. Ethics, societal impacts, security, and legal issues are an integral part of the capstone experience. Attainment of desired skills is measured throughout the year.
Frequency of collection – Every year.
Frequency of analysis – Every year.
How collected and from whom – Multiple instruments provide assessment data to the program.
(1) Formal input is provided by the sponsor via a rubric. This includes an evaluation of the students' ability to function as a team, their communication skills, their understanding of the business model of the project, and their technical skills.
(2) Formal input is provided by the faculty on communication skills and technical accomplishments via a rubric.
(3) Students are required to participate in the Senior Design Fair which is open to the public and held in conjunction with Alumni Weekend. Alumni primarily provide informal feedback on the technical accomplishments of the project. While not quantified in a rubric, their input is valuable to the program.
Benchmarks – Successful completion of the project is tantamount to certifying a student as industry-ready. Students can, and have, been "fired" from their capstone project. The benchmark is for 90% of the students to successfully complete all the requirements of the project and the accompanying activities.

2012-2013	2011 – 2012	2010 – 2011
95%	80%	100%

Table 4.5: Percentage of students who successfully complete Senior Design

In addition, the capstone project is the final opportunity to assess proficiency in a number of outcomes that are introduced throughout the curriculum. The data in Table 4.x represents the percentage of students who scored 80% or higher on the graded work for each topic.

	2012 – 2013	2011 – 2012	2010 – 2011
Globalization	100%	90%	100%
Ethical/social responsibility/legal	100%	90%	100%
Communication	95%	80%	100%
Teaming	95%	80%	100%

Table 4.6: Student attainment of outcomes in capstone experience

The data and student work are supplied in the accompanying course displays and assessment materials, both on paper and online.

5. Senior Exit Interviews

Type of measurement – Subjective. Seniors are asked to identify strengths and weaknesses of the existing program and to suggest improvements.
Frequency of collection – Every year.
Frequency of analysis – Every year.
How collected and from whom – Small Group Instruction Diagnostic (SGID) for graduating seniors. Seniors meet with Dr. Kyle Riley, the Department Head, and do a group assessment using the SGID process.
Benchmarks – At least 75% positive responses to #4, 5, 6, 7, 9, and 17 (see Table 4.7).

The raw data from the SGID process is included in a separate binder, but a summary of three questions is presented in table 4.8 below: greatest strength, greatest weakness, would you recommend this program to others. The survey instrument is given below.

1. (**interest in the field**) What attracted you to Computer Science as a field of study?
2. (**interest in this program**) What attracted you to the SDSM&T Computer Science program?
3. (**accreditation**) Does program accreditation by Accreditation Board for Engineering and Technology (ABET) matter to you?
4. (**advising**) Did you have access to good advising?
5. (**degree program**) Were the requirements and expectations of the degree clear?
6. (**course quality**) Were the courses challenging and productive for you?
7. (**course scheduling**) Were courses offered sufficiently often at the right times?
8. (**electives**) What elective courses not now offered would you suggest offering?
9. (**preparation**) Do you feel prepared for work outside the school?
10. (**co-curricular opportunities**) Were there opportunities to be involved in organizations?
11. (**program positives**) What is going well in the program?
12. (**program *less-than* positives**) What is not going well in the program?
13. (**personal responsibility**) What could you have done to improve your educational experience?
14. (**faculty responsibility**) What can the faculty do to improve things in the program?
15. (**collective responsibility**) What can the students do to improve things in the program?
16. (**staff responsibility**) What can the administration do to improve things in the program?
17. (**advice to others**) Would you recommend this school/dept to friends and family members?
18. (**graduate school**) Would you consider graduate work?

Table 4.7: Questions for SGID Senior Exit Interview

	Advising	Degree program	Course quality	Scheduling	Preparation	Recommend?
2007-08*	The SGID was not held due to a blizzard that cancelled school					
2008-09	Yes	Yes	Yes, but overlap between advanced digital and COA	Yes	Yes But more GUI needed	Yes
2009-10	Yes, but Math faculty aren't as know-ledge-able as CS faculty	Yes		Yes, courses are hard here!	75% Yes. Add more OOP in the curriculum	Yes, but come prepared to work
2010-11	75% Yes	75% Yes Gen. Ed. is hard to figure out	Yes	75% Yes Overlap in scheduled times between electives and required courses	75% Yes Want a course in networking	Yes
2011-12	Yes	Yes Flowchart and check sheet made a big difference	Yes Older faculty more challenging (perceived as positive)	No Schedule not available far enough in advance	Yes	Yes
2012-13	Yes	Yes	Yes Especially courses strong on theory	Electives at incon-venient times, overlap required courses	Yes	Yes

Table 4.8: Summary of key answers. "Yes" indicates unanimous agreement.

	Strength	Weakness
2007-08	The SGID was not held due to a blizzard that cancelled school	
2008-09	Small department, hard classes	No dominant answer
2009-10	Strong fundamentals, teaching C++ as primary language, learning assembly	Worrying whether an elective with small enrollment will make or not
2010-11	Professors	Overlap in content between OS and COA
2011-12	Hands-on focus	Scheduling/offerings of electives
2012-13	Faculty, small department	Not enough electives

Table 4.9: Summary of responses to strengths/weaknesses

The faculty review the student input every year and have made curriculum changes in response to their suggestions. These are detailed in Subsection B below.

6. Outstanding Recent Graduate Awards

Type of measurement – Objective.
Frequency of collection – Every year.
Frequency of review – Every year.
How collected and from whom – The computer science faculty review database information provided by the Alumni Association on the career paths of students who have graduated in the past 10 years. The Outstanding Recent Graduate Award recognizes a graduate who has a record of high achievement in his or her profession. The faculty review the data and select a nominee. Through 2011, a university-wide committee selected approximately five outstanding alumni each year. This process has changed recently to allow one awardee per program. This change will necessitate a change in how this award is used for program assessment in the future.

Year	Awardee	Company
2008	Todd Youngman	IBM
2010	Jason Dorsey	Valicore Technologies
2011	Toren Kopren	Hewlett-Packard
2013	Jason Lamont	Raven Industries

Table 4.10: Outstanding Recent Graduate Awardees in CS since 2008

As the table indicates, computer science graduates continue to be competitive for this prestigious award. The award is a side benefit of the process, however. The true value in

selecting a nominee for this campus-wide award is the review of graduate achievements. This process provides data to assess objectives 1 – 4.

Benchmarks – The goal is to have a computer science student selected as a top achiever each year. A more realistic benchmark is to have a graduate selected once every three years. As mentioned above, this benchmark will change due to the change in the selection process.

7. Student Competition Involvement

Type of measurement – Objective. Faculty track the number of students involved in competition teams. This includes interdisciplinary teams such as the Unmanned Aerial Vehicle team (UAV), Robotics, other Center of Excellence for Advanced Manufacturing and Production (CAMP) teams, and the ACM Programming Team.
Frequency of collection – Every year.
Frequency of review – Every year.
How collected and from whom – Team-project advisors and student participants provide the data.
Benchmarks – The institution has a strong team-project orientation. The goal is to give all students the opportunity to work on a competitive team. However, many students do not have the time or inclination to work on such teams. The benchmark is to have approximately 15% of the students involved in a team competition of some sort. That is typically 15 – 20 students per year.

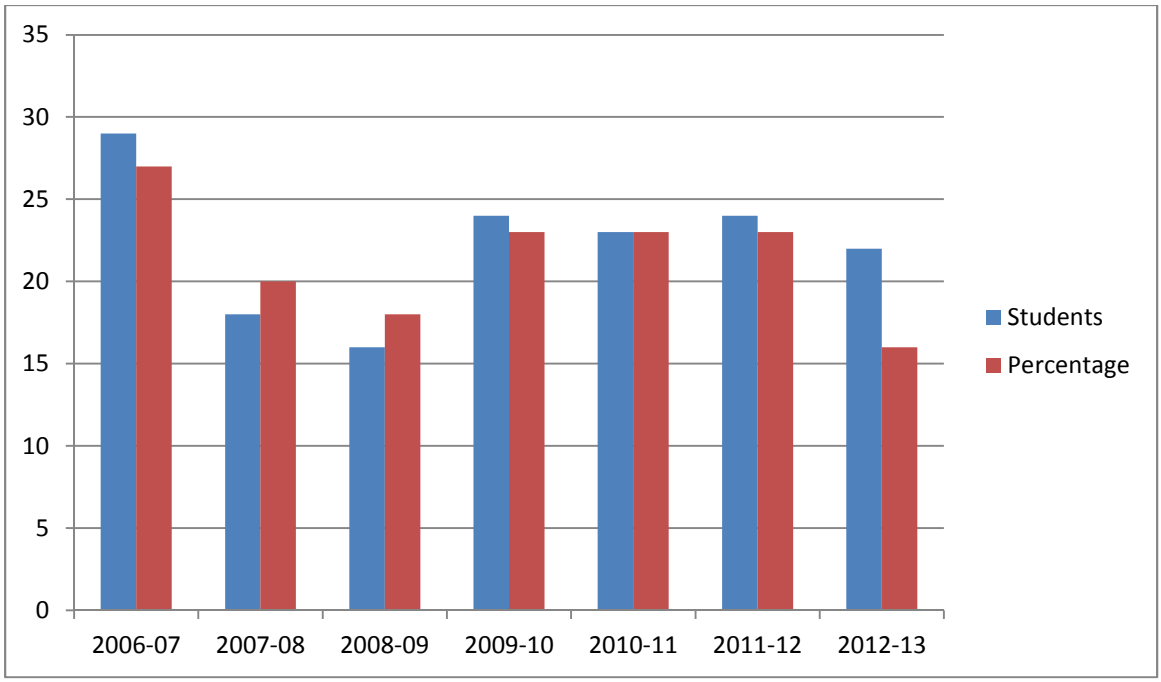

Table 4.11: Number of students and percentage of majors participating in student competition teams

As the chart indicates, the department is meeting its benchmarks relative to competitive team involvement. The department is expecting more students to join these teams in the near future due to the increase in enrollment in the past two years. Since many of the new students are currently at the freshman and sophomore levels, team participation is currently lagging behind enrollment as reflected in the percentage of student participants in 2012-2013 shown in Table 4.11.

The ACM Programming Team has enjoyed particular success in the past five years, earning two trips to the World Finals (Orlando in 2011 and St. Petersburg, Russia in 2013). The team has qualified for the World Finals five times since 1998 and been one spot from qualifying three additional times. SDSM&T routinely wins the state, and it is not uncommon, as was the case in Fall 2012, that the top five teams in the state were all from SDSM&T.

8. Co-op Reports

Type of measurement – Objective/Subjective. Co-op has been identified as an important educational experience and students are encouraged to participate. The number of students who are offered (and accept) co-ops, plus the number who are offered permanent positions after completing a co-op, is an objective measure of the employability of our students. Employers are asked to complete a survey which also provides a subjective measure of the technical abilities, communication skills, and other attributes of students who have not yet completed the program.
Frequency of collection – Every year.
Frequency of review – Every year.
How collected and from whom – Career Planning provides data on the number of co-ops offered and the number of permanent positions offered after the co-op. Dr. Corwin, the co-op coordinator, performs an assessment of the employer surveys.
Benchmarks – The goal is for every student to have the opportunity to participate in a co-op. The benchmark is for at least one-third of the graduates to have done a co-op.

The department strongly encourages students to gain practical experience through summer employment and co-op experiences. The chart below would suggest that few students are able to do so when in fact, essentially *all* of the students who graduate from the program have completed a meaningful work experience in computer science, a co-op, or an undergraduate research experience prior to graduation. The number of co-op students reflects (1) the student must pay for three credits to participate in a co-op and many students choose not to do so and (2) the student must be willing to do advanced planning and have all co-op paperwork completed prior to starting work. Again, many students fail to do so. A survey of students in Senior Design in Fall and Spring 2012-2013 showed that 85% of the students in the class already had significant off-campus work experience in computer science through co-op, summer employment, or part-time employment. If industry-sponsored on-campus projects are included in the count, 100% of the students had industry experience prior to graduation.

The co-op report is an important piece of objective data from a key constituent. The informal feedback from employers who hire our students is also extremely positive.

AY	Total	A's	B's	C's	D's	F's	W
2006-07	3	2	1				
2007-08	5	2	2				1
2008-09	4	2	1				1
2009-10	2		1				1
2010-11	8	8					
2011-12	5	2	3				
2012-13	3	3					

Table 4.12: Summary of co-op grades (based on employer evaluations)

9. Undergraduate Research Involvement

Type of measurement – Objective. The faculty track the number of students involved in undergraduate research each year.
Frequency of collection – Every year.
Frequency of review – Every year.
How collected and from whom – Collected from faculty members who direct undergraduate research.
Benchmarks – The goal is for all students to have the opportunity to participate in undergraduate research to encourage them to pursue graduate school. However, given that only a small percentage of students go on to graduate school, the benchmark is to have at least as many students participate in undergraduate research as continue to graduate school each year. This is approximately three students per year.

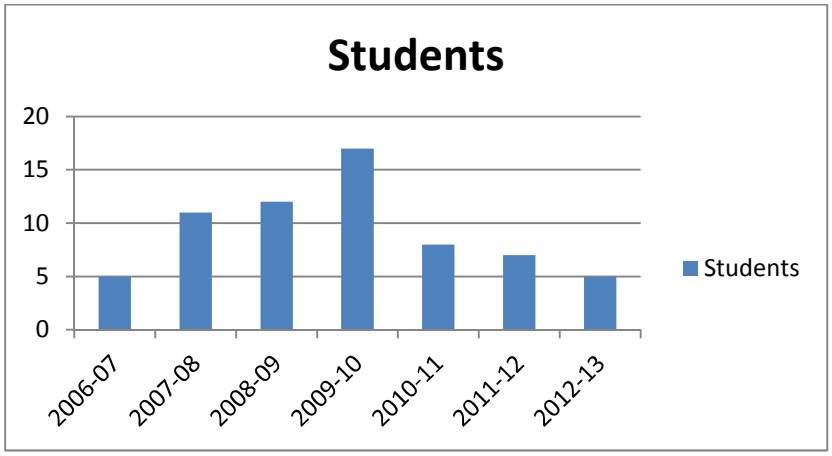

Table 4.13: Summary of undergraduate research participation

The data suggests that fewer students are participating in undergraduate research, prompting the faculty to consider whether this was a concern or not. A comparison with the numbers of students on competition teams shows an increased interest in the team

opportunities. Projects like the UAV team and the Robotics team have significant research components and are attracting students who would otherwise have participated in undergraduate research. One of the goals for the undergraduate research offering is to encourage students to consider graduate school. The UAV team and Robotics team are accomplishing this as effectively as undergraduate research. A number of students continue to explore topics of individual interest each year, and the faculty are satisfied that all students who wish to participate in a research project have the opportunity to do so. The spike in participation in 2009-2010 was due to "Team Bob" an undergraduate team funded by the NSF through the AMP Center to build web tools for running the center.

10. Placement rates/ Starting Salaries

Type of measurement – Objective.
Frequency of collection – Every year.
Frequency of review – Every year.
How collected and from whom – Collected by the Career Planning Office.
Benchmarks – The goal is 100% full employment with an average salary comparable to the national average. Given that some students are place bound, that is, they need to work in a particular place, and that location is often a small town in South Dakota, a realistic benchmark is 85% placement. Given the low salaries in South Dakota and the fact that about half of our graduates remain in the state, the benchmark is that the average salary for all graduates will be 80% of the national average.

AY	# Grads/ #reporting	Military/ Other	Graduate school	Industry	Total	% Placement	Average starting salary
2006-07	17/17	2	3	11	16	94%	$55,881
2007-08	14/13	1	2	10	13	93%	$56,423
2008-09	16/15	0	4	11	15	94%	$55,425
2009-10	14/13	1	6	7	13	100%	$57,000
2010-11	12/11	0	4	8	12	100%	$55,100
2011-12	20/20	0	4	16	20	100%	$68,646

AY	In SD	Outside SD	Average Salary	National average salary	Percentage of National Average
2006-07	5	6	$55,881	$53,396	105%
2007-08	1	9	$56,423	$60,416	93%
2008-09	8	3	$55,425	$61,467	90%
2009-10	3	4	$57,000	$61,112	93%
2010-11	5	3	$57,100	$66,084	86%
2011-12	5	11	$68,646	$64,400	106%

Table 4.14: Placement percentages and starting salaries

The charts demonstrate that the program is meeting its benchmarks relative to placement and starting salaries. The placement figures are often affected by a lack of reporting. Informal evidence suggests that computer science graduates have achieved 100% employment since 2007. The salary data is provided by the Career Planning Office and is not separated into in-state versus out-of-state offers, but knowledge of local offers suggests that students who leave the state are receiving substantial offers. A point of pride advertised by the university is that the average starting salary for an SDSM&T graduate is greater than the average starting salary for a Harvard graduate ($62,696 for SDSM&T versus $54,100 for Harvard). Tuition at SDSM&T is roughly one-quarter that of Harvard.

Assessment Instruments											
Program Outcomes	Annual Assessment Report	MFT	IAB	Course-Embedded Assessments	Capstone Project	Senior Exit Interviews	Outstanding Recent Grad Awards	Student Competition Involvement	Co-op Reports	Undergrad Research Involvement	Placement Rates/Starting Salaries
1. possess a strong foundation in the software development process;	**yearly**	**yearly**	**every 2 years**	**every 2 years**	**yearly**	**yearly**	**yearly**		**yearly**		**yearly**
2. be able to solve problems using a variety of programming languages and have extensive experience with at least one high-level language;	**yearly**	**yearly**	**every 2 years**	**every 2 years**	**yearly**			**yearly**	**yearly**	**yearly**	
3. have a background in computer hardware and experience with a variety of operating systems;	**yearly**	**yearly**	**every 2 years**	**every 2 years**							
4. possess an extensive background in mathematics and an appreciation of the scientific method;	**yearly**			**every 2 years**				**yearly**	**yearly**	**yearly**	
5. have an understanding of the theoretical foundations of computing;	**Yearly**	**yearly**	**every 2 years**	**every 2 years**						**yearly**	
6. have developed effective communication skills and have experience working with teams;	**Yearly**			**every 2 years**	**yearly**	**yearly**	**yearly**	**yearly**	**yearly**	**yearly**	**yearly**
7. possess an understanding of professional, ethical, legal, security and social issues and responsibilities.	**Yearly**			**every 2 years**	**yearly**	**yearly**			**yearly**		

Table 4.15: Summary of assessment instruments, frequency of collection, and mapping to program outcomes

B. Continuous Improvement

Describe how the results of evaluation processes for the student outcomes and any other available information have been systematically used as input in the continuous improvement of the program. Describe the results of any changes (whether or not effective) in those cases where re-assessment of the results has been completed. Indicate any significant future program improvement plans based upon recent evaluations. Provide a brief rationale for each of these planned changes.

The continuous improvement process is driven by the Annual Assessment Report and the annual faculty evaluation process. Since 2007, the department has written an annual report detailing changes to the program, citing motivations for the changes, and evaluating changes made in previous years. Copies of these reports are available in the binder labeled "Computer Science Annual Assessment Reports" and online at http://www.mcs.sdsmt.edu/abet. Annual faculty evaluations are not available, but the course-embedded assessments that are part of that review are available in the binder labeled "Course-Embedded Assessment" and online at the URL listed above.

Numerous changes, large and small, have been made to the program in the past six years. Following are detailed descriptions of the major changes made in the areas of assessment, faculty, and curriculum that were motivated by departmental assessment activities. The instrument that identified the issue, the response, and the evaluation of the response are provided as examples of the continuous process of "closing-the-loop" that is an integral part of the departmental assessment process.

For each of the items listed, an assessment instrument triggered a review, data was collected, options were considered with input from our constituents, a decision was implemented, and the results were reviewed.

Two significant changes to the program, the move from 128 to 120 credit hours for the degree and the new administrative structure, are discussed in the Overview section of this document. The departmental response to these externally-initiated changes is also described there. However, evaluating the new curriculum will be a critical component of departmental assessment activities for the next few years.

1. Changes to the assessment process.

How identified: IAB, Focused Curriculum Review.

Actions taken:

As noted in *Section 2: Program Objectives*, the department has a process for reviewing and refining program objectives. A similar process is in place for reviewing all assessment instruments, the programmatic outcomes, the course-specific outcomes, and the mapping between course outcomes and program outcomes.

(a) Review of program outcomes – The IAB meets every two years, and the first order of business is to review the program's Mission Statement, Program Objectives, and Program Outcomes. The IAB is routinely asked to address all issues through the lens of two questions: (1) where is industry going and (2) how well positioned is SDSM&T to take advantage of these new directions. In 2010, the IAB felt that database coverage was insufficient in the program and encouraged the department to add database as a programmatic outcome. The growth in web programming, e-commerce, and mobile applications were compelling arguments to add a program outcome directed at strengthening database coverage. The list as published in 2010 is below.

1. have a strong foundation in the software development process;
2. be able to read and write program code in a variety of programming languages and have extensive experience with at least one high-level language;
3. have experience in programming for and using a variety of computer operating systems;
4. possess problem solving and algorithm development skills;
5. have a strong understanding of the theoretical foundations of computing;
6. have a strong background in computer hardware;
7. has the knowledge to produce effective conceptual and physical database systems**;**
8. possess an extensive background in computer-related mathematics;
9. have an appreciation of the scientific method;
10. have developed and practiced effective communication skills;
11. have experience working in teams;
12. understand and respect the professional standards of ethics expected of a computer scientist;
13. have an appreciation for the societal/ global impact of computing.

Table 4.16: Program outcomes 2010

When the IAB met in 2012, they were again asked to consider the program outcomes. Many of the same people were present, and, after a long discussion, the group proposed removing the database outcome and rewording the other outcomes to produce a leaner, simpler list. They reiterated that database was a critical component of the curriculum but was not on the same level as theory or mathematics. The proposed list, adopted by the department in October 2012, is presented below.

1. possess a strong foundation in the software development process;
Comment: unchanged
2. be able to solve problems using a variety of programming languages and have extensive experience with at least one high-level language;
Comment: problem solving should be tied to programming
3. have a background in computer hardware and experience with a variety of operating systems;
Comment: this reflects the shift in hardware emphasis to systems
4. possess an extensive background in mathematics and an appreciation of the scientific method;
Comment: all mathematics is computer-related.
5. have an understanding of the theoretical foundations of computing;
Comment: unchanged except to remove "strong." Modifier was unnecessary.
6. have developed effective communication skills and have experience working with teams;
Comment: teaming requires communication
7. possess an understanding of professional, ethical, legal, security and social issues and responsibilities.
Comment: use ABET wording. Is an improvement over current wording.

Table 4.17: Program outcomes 2012 and rationale for each change

Program outcomes are reviewed every two years and perhaps it is the nature of a rapidly-changing discipline that the program outcomes need adjustment in every review cycle. While it is doubtful that the current set of outcomes will remain static for more than a few years, the process is in place to manage the necessary changes and adapt as the discipline evolves.

(b) Addition of the annual assessment report. This instrument was motivated by preparation for the ABET visit six years ago. It provides a snapshot of the state of the department at the end of each year and serves as a vehicle for recording collected assessment data. The faculty finds this to be a useful mechanism for storing the "departmental memory" and collecting information necessary for both HLC and ABET accreditation. The content has changed slightly throughout the past six years, but the basic format has served the department well. This is now a mature instrument that is central to the departmental assessment activities.

(c) Replace course evaluations with course-embedded assessments. In 2010, the department re-evaluated its assessment of individual courses and how they contribute to the programmatic outcomes as part of an assessment "tune-up." The instrument in use at the time was student evaluations. The comments could be useful, but often the results were too tied to individual performance in the class to provide meaningful insight into potential changes for a course. The Curriculum Committee decided to substitute course-embedded assessments as the mechanism of choice for individual course reviews. The format for a course-embedded assessment was defined, standards articulated, and a schedule set for reviews in each course. Details on how these are generated, the content, and the benchmarks established are provided in the description of course-embedded assessments above. With the exception of Robotics and Theory of Computation, a course-embedded assessment has been done on every class in the curriculum. Summer of 2013 was the first opportunity to consider the aggregated results.

The initial reaction to the information generated from the course-embedded assessment data is positive, but this is not a mature process and likely to evolve over the next three years. The Curriculum Committee review of the process found the ability to aggregate multiple inputs to a single outcome to be useful, but multiple years of data must be collected before clear evidence exists to direct curriculum changes. As noted above, the Curriculum Committee will re-evaluate the move to three hardware courses, consider increasing Linux coverage in CSC 300, and collect additional data on strengthening mathematics coverage with the next round of course-embedded assessments.

(d) Comprehensive review and reformulation of course outcomes. In 2011, the department was forced to consider how the curriculum could transition from 128 credits to 120 credits. The focused curriculum review that year concentrated on that task, but, as a consequence, all course content and all course outcomes were reconsidered. The task-specific outcomes were replaced with higher-level achievements. The Curriculum Committee reviewed the changes in course outcomes and, in some cases, shared the changes with students in their courses. Students responded positively to a shift from the "laundry-list" approach to broader outcomes.

2. Changes to the curriculum

How identified: IAB, Senior Exit Interviews, student discussion groups, industry feedback (employers, recruiters, co-op), Advising Survey, FCR. The FCR includes comparison to the ACM/IEEE Computer Science Curriculum (2008 revision). The next review will consider the proposed Computer Science Curriculum 2013 currently in development. All the major constituents (students, faculty, industry, and alumni) are important contributors to curriculum revisions.

Actions taken:

A comprehensive list of differences between the curriculum of 2007 and that of 2013 is present in *Section 4: Curriculum*. This discussion focuses on the assessment tools that motivated the changes, the steps taken, and the evaluation of the results.

(a) CS elective offerings

Three suggestions for improving electives offerings were identified by the senior exit surveys and the IAB: add an elective in networking, add an elective in mobile computing, and publish a two-year rotation of elective offerings well in advance to facilitate scheduling.

1. Networking – this course has been added to the elective offerings and will be offered for the first time in Spring 2014. An assessment of the course will be performed in summer 2014.
2. Mobile Computing – this course has been added to the elective rotation and will be offered for the first time in Fall 2013. An assessment of the course will be performed in Spring 2014.
3. Rotation – Staffing shortages have made it difficult to establish and follow an elective rotation. Given the current stability, the curriculum committee was able to publish the following elective rotation schedule to the students in Fall 2012. At the semi-annual departmental "advising day", held April 2, 2013, students were enthusiastic about the number, content, and scheduling of the elective offerings. The negative

comments were that we can only offer courses every other year which prevents some students from taking particular electives. Program enrollment would need to increase substantially to offer current electives more frequently. The alternative, offering fewer electives more frequently, was not viewed as a reasonable alternative by the students.

Even Spring (e.g. S14)		**Even Fall (e.g. F14)**
Theory of Computation		Parallel
Artificial Intelligence		Robotics
Networks		Graphics
Probabilistic Robotics*		
Odd Spring (e.g. S15)		**Odd Fall (e.g. F15)**
Security		Cryptography
Image Processing		Robotics
Mobile Computing		Computer Vision (CS/CENG)
Robot Planning*		

* indicates 7xx course, only open to select undergraduates

Table 4.18: Elective course offering schedule

(b) Science and Mathematics electives

Feedback from alumni through recruiters and the IAB motivated the department to relax the rigid requirements on math and science courses. The science requirement now allows a student to take two semesters of a lab science chosen from a list of courses approved by the program. Only University Physics is required. The new flexibility enables broader opportunities for graduates in bio-tech and the geosciences, two growth areas in South Dakota. All science courses approved for the CS program are required courses for majors in the specified discipline. Similarly, replacing the Differential Equations requirement with a math elective was motivated by students expressing a desire to be better prepared for Cryptography by taking Abstract Algebra. The statistics course designed for CS majors was replaced by the statistics course required for Math majors to reflect the program's philosophy that CS majors should take out-of-major courses that count in the major for those disciplines.

(c) Drop HUM 375

This action was taken to address student concerns expressed in the senior exit interviews and in student opinion surveys in Senior Design. Humanities 375, Computers in Society, was originally designed to meet the needs of the CS program and primarily focused on ethics. The course has become a popular elective for many majors, and the content has expanded to address issues of interest to a wider audience. Many of these new topics did not meet the needs of our program. It was also noted that a discussion of many ethical questions requires technical knowledge which is better suited for coverage in computer science courses. For these reasons, the curriculum committee voted to respond to student

input and drop HUM 375 as a required course. Coverage of ethics was increased in the CS curriculum, particularly in Senior Design, and is now done in the context of computer science; a feature that students felt was not emphasized in HUM 375. Table 4.17 lists the required courses where the "soft skills" enumerated in Enabled Student Characteristics (a) – (k) are systematically evaluated. Elective offerings reinforce these skills as well.

	Freshman	Sophomore	Junior	Senior
Globalization	CSC 250		Various electives	CSC 465/467 Various electives
Ethical/social responsibility	CSC 110 CSC 150 CSC 250	CSC 314		CSC 465/467
Communication	ENGL 101	ENGL 279	ENGL 289 CSC 470	CSC 465/467
Legal		CSC 314	CSC 470	CSC 465/467
Teaming	CSC 250	CSC 300	Various electives	CSC 456 CSC 465/467 Various electives

Table 4.19: Coverage of soft skills in computer science courses

(d) Add CSC 110/111

This change was motivated by the FCR and by feedback from advising. CSC 110 (1 credit) is a course that focuses on college survival and an overview of the field of computer science. Many freshmen do not have a sense of the broad array of career options open to them in their chosen field and can be discouraged by the difficulty of learning their first programming language. The course has only been offered one time, but student feedback was positive with the suggestion of even greater emphasis on career opportunities.

CSC 111 (2 credits) is an optional course designed to give students with no experience a more gentle introduction to programming. Unfortunately, the state of South Dakota will no longer require a computing class for high school graduation effective 2013-2014. The CS program is being proactive in addressing the almost-certain reduction in computing experience that entering freshmen will have within the next two years. Historically, the completion rate in CSC 150 in Spring semesters, when many underprepared students take the course, is typically around 70% as noted in Table 4.18. Many of these students indicated in the department advising sessions that CSC 150 (CS 1) was too large a first step into programming. In addition, the prerequisite or co-requisite of Calculus I for CS I blocks many students from taking that course until their second, or possibly third, semester. CSC 111 was designed to allow students to take a computer science class in

their first semester and take smaller steps learning programming. Of the 11 students who successfully completed CSC 111, only five were eligible to continue on to CSC 150 due to the mathematics co-requisite. Of those five, three earned a grade of "B" and two earned a grade of "F". The course-embedded assessment from the first offering of that class showed students struggled with Python but felt the course was worthwhile. The students recommended a lab experience to accompany the lectures, and such a lab will be added starting in Fall 2013.

CSC-150

Term	Count	A	B	C	D	F	(% D or Better)
2009SP	85	11	17	25	10	22	**74.1%**
2009FA	103	25	33	22	12	11	**89.3%**
2010SP	135	21	30	48	17	19	**85.9%**
2010FA	99	19	14	27	6	33	**66.7%**
2011SP	142	33	35	25	12	37	**73.9%**
2011FA	149	31	37	36	17	28	**81.2%**
2012SP	142	20	32	38	7	45	**68.3%**
Total	855	160	198	221	81	195	**77.2%**
Percent		19%	23%	26%	9%	23%	

Table 4.20: Grades in CSC 150

3. Changes to individual courses

How identified: Course-embedded assessments, Focused Curriculum Reviews, and the IAB. Other inputs motivated the curriculum-wide review, notably the change to 120 credit hours, but a collection of small changes were identified and implemented to remove redundant coverage, ensure coverage of emerging topics, and respond to student and industry input.

Actions taken:

(a) Require Graphical User Interfaces with Object-Oriented Programming (GUI/OOP). This course was initially offered as an elective, but an FCR in conjunction with the IAB in 2008 recommended that OOP coverage be strengthened in the curriculum. Initial coverage is provided in CS2 (CSC 250) and additional coverage is provided in Programming Languages (CSC 461). However, an entire course devoted to GUI & OOP solidifies the principles and provides the depth industry expects from our graduates. This change also allowed Programming Languages to shift from 4 credits to 3 credits. This has had a positive influence as assessed in Senior Design. Teams are able to work on a greater variety of industry projects as a result of increased proficiency with GUI & OOP.

(b) Introduce databases earlier in the curriculum. The IAB recommended in 2010 that students see an introduction to databases before the end of the sophomore year. An assignment using MySQL was added to Data Structures in Fall 2012. The students found the assignment challenging and interesting, but the time required to incorporate the necessary background meant other topics had to be dropped. Database Design (CSC 484)

is now a prerequisite to Senior Design, ensuring that students get exposure to databases before their senior year. The next FCR will discuss alternatives for early introduction of databases.

(c) Increase experience with software engineering tools. The IAB recommended incorporating software engineering tools earlier in the curriculum, particularly prior to Senior Design. The use of source code control such as SVN, project management software such as Trello, additional emphasis on testing capabilities in programming environments such as Visual Studio, and agile development methodologies, were discussed in Data Structures and incorporated into Software Engineering. The impact of the change to Software Engineering will be determined in Fall 2013 when those students participate in Senior Design. The expectation is that teams will progress on their projects faster than in previous semesters. This will be assessed in May 2014.

(d) Increase presentation opportunities for students. Surprisingly, this was recommended by the student group that met with the IAB. Students working on research or projects outside of normal coursework have the option of presenting at the Department Colloquium Series and the Undergraduate Research Symposium which was created in 2010 to recognize excellence in undergraduate research across campus. Two or three students take advantage of these opportunities each year. Since 2010, Dr. Jeff McGough has been moving the Senior Design experience to a more professional level by requiring teams to work with an external client and by requiring participation in the campus-wide Senior Design Fair. These two changes have significantly increased the number, level, and quality of presentations required of students prior to graduation. In spring 2013, the Software Engineering students were required to present their projects to an external audience for the first time to prepare them for Senior Design.

(e) Multi-semester projects. The IAB felt that the senior-design experience should extend a full year. The senior exit survey confirmed that students also wanted the option of working on more complex projects and working on projects that continue from one year to the next. The curriculum committee adopted a two-semester senior design sequence and moved Software Engineering to the second semester of the junior year. Previously, Software Engineering was the first semester of the senior year and Senior Design was the second semester. Students started their projects in Software Engineering, but little progress was made until Senior Design due to the need to learn the Software Engineering material.

Currently, Software Engineering is a prerequisite for Senior Design making a three-semester sequence of project work. The tools and techniques needed for Senior Design are covered in Software Engineering allowing students to immediately focus on their projects. This does create scheduling challenges for some students but advisors remind students every semester to look at the prerequisite chain and plan accordingly. Typically, one student per year is in a scheduling bind and these students are exclusively transfer students or students who have changed majors. The greater challenge is keeping the juniors out of Senior Design because the students are very eager to enroll in the course.

Extensive involvement from industry partners such as Innovative Systems has made the senior design experience a realistic work experience. An employee of Innovative Systems, Brian Butterfield, is assigned to the department to help manage projects in Senior Design and Software Engineering, and bring current industry practices into the classroom.

Students have also been encouraged to do projects that support the competition teams such as Lunabotics and UAV which provide the experience of working on multi-year, multi-disciplinary projects. Project evaluations from Innovative Systems plus feedback from the Senior Design Fair reviewers show a marked increase in senior design project quality over the past three years. This was a positive change that the department will continue. Senior Design projects are available for review at http://www.mcs.sdsmt.edu/abet.

4. Changes to faculty and staff

How identified: IAB, Focused Curriculum Review

Actions taken:

Changes to the faculty are noted in *Section 6: Faculty*. This section describes the assessment process in place for determining the composition of the faculty. The number of faculty is driven by external budget factors, but the department has always enjoyed the support of the administration. The number of vacancies in the department for the past five years was a result of many market forces, low salaries, a geographic location that is not desirable to many candidates, abrupt departures for personal reasons, and a temporary re-assignment to address institutional needs. The budget always allowed for full staffing, and we are now at full strength.

The number of faculty and the areas of expertise are evaluated every year.

	2007-2008	2008-2009	2009-2010	2010-2011	2011-2012	2012-2013
Full-time (tenure-track)	5.5 [a]	5 [b]	7	5.5 [c]	4.5 [c]	6.5 [d]
Full-time (instructor)	2	2	1	1	1	1
Part-time	2	1	1	3	2	1

[a] Wei left mid-year to take a job in industry
[b] Corwin and Logar were on Sabbatical Spring 2009
[c] Logar served as Dean of Graduate Education 2010-2012
[d] Karlsson joined in January 2013

Table 4.21: Faculty in the computer science program

The number of faculty is certainly critical to the operation of the program. An evaluation of department needs through a focused curriculum review and input from the IAB established both the number of faculty needed to deliver a quality curriculum and the areas of expertise needed in the new hires. The eight full-time faculty members committed to the 2013-2014 academic year is the strongest cohort the program has enjoyed in the past decade and is capable of delivering the strongest curriculum the program has offered in a decade. While the quality of the program did not suffer during the lean years of 2010-2012, the strength going forward has fostered a renewed sense of excitement among the faculty.

C. Additional Information

Copies of any of the assessment instruments or materials referenced in 4.A and 4.B must be available for review at the time of the visit. Other information such as minutes from meetings where the assessment results were evaluated and where recommendations for action were made could also be included.

Binders containing assessment materials can be found with the course display materials and online at http://www.mcs.sdsmt.edu/abet.