

Goals for the Computer Science Major

I. Program Outcomes as defined by ABET (CAC)

The program enables students to achieve, by the time of graduation:

- 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 requirement 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.

II. Student Learning Outcomes (SLOs) for the Computer Science Program

Students will:

1. apply structured principles and good practices to the task of developing software systems.
2. understand how hardware provides the necessary structure for execution and influences the design of software.
3. understand general operating system functions and structures, comprehend system capabilities and modify systems to meet specifications.
4. effectively communicate both technical and non-technical aspects of their work in formal and informal situations.
5. understand the professional code of ethics and the need to conduct themselves in a professional manner.
6. apply formal methods to the process of constructing a system and appreciate the need to study and develop such methods.
7. analyze the processes used when designing a system and employ established frameworks to evaluate the completed work.
8. apply the principles learned in the core curriculum to various application domains, build on these principles, and stay current in their knowledge.

III. How does the process of assessment ensure that all the ABET outcomes are satisfied.

A. An ability to apply knowledge of computing and mathematics appropriate to the discipline.

In activities of all the learning outcomes students will be applying knowledge of computing and mathematics to solve a variety of problems.

B. An ability to analyze a problem, and identify and define the computing requirement appropriate to its solution.

In SLO1 students will be analyzing problems at all stages. In Stage 2 of SLO 2 they will see how specific architectural features impact the quality of the solution. In SLO6 student will apply formal methods to the analysis.

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

Design and implementations of computer-based components or programs are done in activities for SLO1, SLO2, SLO3, SLO6 and SLO8. Evaluation is done in activities for SLO1, SLO2, SLO3, SLO6, SLO7 and SLO8.

D. An ability to function effectively on teams to accomplish a common goal.

Activities in Stage 3 of SLO1, SLO3 and SLO7 require the student to be effective member of a team.

E. An understanding of professional, ethical, legal, security and social issues and responsibilities.

SLO3 and SLO5 addresses this.

F. An ability to communicate effectively with a range of audiences.

SLO4 addresses this.

G. An ability to analyze the local and global impact of computing on individuals, organizations, and society.

SLO5 addresses this.

H. Recognition of the need for and an ability to engage in continuing professional development.

SLO8 addresses this.

I. An ability to use current techniques, skills, and tools necessary for computing practice.

In all stages of SLOs 1, 2, 3, 6, 7 and 8 students will complete activities that require them to employ current techniques, tools and skills.

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.

In SLOs 1, 2, 3 and 6 students complete modeling & design activities that demonstrate an ability to apply mathematical foundations, algorithmic principles and computer science theory.

K. An ability to apply design and development principles in the construction of software systems of varying complexity.

In the activities associated with SLOs 1, 3, 6 and 7

IV. How is the assessment plan used.

Students progress towards each learning outcome in 2, 3, or 4 stages. Each stage is typically met in one course. For each stage there is a set of **activities** that are completed by the student in the associated course. The plan describes the activities in terms of content, level of difficulty and the extent of participation of the student. Each stage has a set of **expected outcomes** which broadly describe the grading rubrics associated with the activities. The attainment of these outcomes is documentable and this **documentation** is also described in the plan. In each year of the assessment cycle, a subset of the stages is assessed, as described in the calendar at the end of the year the results of the assessment are collected and used to provide feedback for future program changes.

V. Calendar of the 2010-2015 Spring assessment cycle.

VI. Details of how each SLO will be attained, measured and documented.

1. The ability to apply structured principles and good practices to the task of developing software systems.

Stage 1. (CSCI 201)

Activity completed. Student has individually completed a project that creates a simple application using 2 or more data abstractions (ADTs) and one main program. The entire system will use simple algorithms (like quadratic sorting and binary search).

Expected Outcomes. Completed the development in multiple stages; used stubs to test the system as a whole; used drivers to carry out unit testing for functional components and for the data abstractions.

Documentation. (i) Completed student projects. (ii) Answers to test questions relating to the projects.

Stage 2. (CSCI 301)

Activity completed. Student(s) may have worked individually or in small groups to complete a more complex application, using some data structures and specialized algorithms.

Expected Outcomes. In addition to the practices followed in Stage 1, the development process will separate the interface from the implementation and will introduce the student to the use of generic components (templates) as a mechanism for re-using code.

Documentation. (i) Completed student projects. (ii) Answers to test questions relating to the projects.

Stage 3. (CSCI 331)

Activity completed. Students work in groups, to develop and test pieces of a larger application that exists; use literate programming tools in their documentation.

Expected Outcomes. In addition to the practices of Stages 1 and 2, students learn to integrate software from disparate sources in their development process.

Documentation. (i) Completed student projects. (ii) Answers to test questions relating to the projects.

2. an understanding of how hardware provides the necessary structure for execution and influences the design of software.

Stage 1. (CSCI 220)

Activity completed. (i) Laboratory assignments involving building and testing circuits. (ii) Programming assignments with assembly language. (iii) Analysis and design of combinational and sequential circuits.

Expected Outcomes. Students recognize that gates can be used to build circuits for various components using by a modern computer. They learn basic CPU architecture and how it can enable execution of an assembly language program and how simple programs can be written in assembly.

Documentation. Completed project reports and programming assignments; answers to questions on tests.

Stage 2.

Activities Completed. (i) Analysis and design of different kinds of processors (single-cycle, multi-cycle, pipelined, etc.) and instruction sets (CISC, RISC etc.)

(ii) Advanced programming with assembly languages and high-level languages that is sensitive to hardware constraints and exploits hardware features.

(iii) Application of performance improvement techniques in various environments.

Expected Outcomes. Students learn different instruction sets and architectures and performance improvement techniques employed at the instruction set, the gate, the register transfer, the processor design, pipeline scheduling and the memory levels.

Documentation. Completed project reports. Answers to test questions.

3. have basis for understanding general OS functions and structures and an ability to utilize that knowledge to:

- Comprehend capabilities of current systems
- Write / modify systems software

Stage 1 (CSCI 201 and CSCI 220)

Activity Completed. The student has used a UNIX-type operating system for programming and a common UNIX text editor such as vi or emacs (CSCI 201). In programming assignments, the student has gained knowledge of how to access different types of main memory locations: stack / heap / global using different techniques: simple variable, array, pointer. (CSCI 201). The student has converted numbers between various bases: binary, decimal, hexadecimal (CSCI 220).

Expected Outcomes. The student has learned how to interact at a basic level with a UNIX-type operating system. They have become aware that the user accessible portion of a program's memory may be divided into a number of different areas. Base conversion helps the student understand how numbers may be stored within a computer system.

Documentation. (i) Completed student projects. (ii) Answers to test questions.

Stage 2 (CSCI 310)

Activity Completed: The student will have read about, heard lectures on and in many cases participated in in-class discussions on introductory operating system concepts such as: CPU scheduling, deadlock detection, primary and secondary memory, synchronization and security. Also, the student will have written (perhaps as part of a team) a program that uses threads.

Expected Outcomes: The student will be able to:

- Demonstrate use of scheduling algorithms (CPU and / or disk).
- Demonstrate deadlock detection algorithms.
- Understand primary and secondary memory management alternatives and related algorithms.
- Demonstrate knowledge of security issues; such as how to encrypt / decrypt a message using the RSA algorithm.
- Demonstrate understanding of synchronization issues through the use of synchronization primitives.

Documentation. (i) Completed student homework assignments and projects. (ii) Answers to test questions.

Stage 3 (CSCI 311)

Activity Completed.

- Installed an operating system such as MINIX by manually partitioning the disk.
- Written a program using POSIX compliant system calls.
- Written a set of programs running on one machine that communicate with each other -- such as via pipes or signals.
- Added new system calls to an operating system and/or modified an existing system call.

Expected Outcomes. The student has:

- learned how to install an operating system.
- learned how to write a program that interacts with the operating system through the use of system calls; this includes handling normal as well as abnormal (error) conditions.
- learned how to write a set of programs that communicate and/or synchronize their actions.
- gained some knowledge of common system programming tools: to control the source code / code generation process (such as *make*), to search for named files or directories (such as *find*) or strings (such as *grep*).
- developed an understanding of the challenges in modifying and building a system of programs that is very large (such as MINIX).

Documentation. (i) Completed student projects. (ii) Answers to test questions

4. the skills needed to communicate both technical and non-technical aspects of their work in formal and informal situations

Stage 1. (CSCI 201 and CSCI 301)

Activity completed. Produce design and user documentation for programs of simple to moderate complexity. Document the testing of such programs.

Expected outcomes. Students will generate documents that adequately and accurately explain the design, testing, and use of moderately complicated programs.

Documentation. Design documents, testing reports, and user instructions for programs.

Stage 2. (CSCI 331)

Activity completed. Collaboratively produce design and user documentation for collaborative programs of higher complexity. Document the testing of such programs.

Expected outcomes. Students will collectively generate documents that adequately and accurately explain the design, testing, and use of programs sufficiently complicated to necessitate collaboration in their production.

Documentation. Design documents, testing reports, and user instructions for programs.

Stage 3. (CSCI 334)

Activity completed. The student has individually completed the following activities:

- (i) written a research paper on a technical computer science topic
- (ii) given an oral presentation on the research paper
- (iii) written source code documentation
- (iv) used literate programming techniques to produce program documentation
- (v) created a user's manual for a program
- (vi) written a design document for a program
- (vii) written a test document for a program

Expected Outcomes. All activities have passed an evaluation using detailed rubrics on the D2L web site for CSCI 334. In order for an activity to pass, it must receive one of the top two ratings on each trait listed under the relevant rubric.

Documentation. Final versions of the research paper, presentation slides, design document, source code documentation, literate programming documentation, user manual, and test document.

5. an understanding of the professional code of ethics and an ability to conduct themselves in a professional manner.

Stage 1. (CSCI 201, CSCI 301, CSCI 331)

Activity completed. Exposure to applicable codes of ethics. Identification of potential stakeholders for a software project. Understand that software design can influence safety and risk. Understand the limits of testing.

Expected outcomes. Students can identify potential stakeholders for a software project. Students can identify safer/riskier design alternatives. Students understand the limits of testing.

Documentation. Answers to exam questions on these topics.

Stage 2. (CSCI 332)

Activities completed. Analysis and comparison of applicable codes of ethics. Exposure to core concepts relating to: computing history, social contexts, ethical choices vs. preferential choices, intellectual property, privacy and civil liberties, and risks.

Expected outcomes. Students understand how the applicable codes of ethics relate to them and other stakeholders. Students can cogently discuss core concepts in computing ethics.

Documentation. Essay / term paper discourse, or answers to exam questions, on these topics.

Stage 3. (Post-completion survey.)

6. ability to apply common formal methods to the process of constructing a system and an appreciation of the need to study and develop such methods.

Stage 1. (CSCI 201) Formalisms such as assertions---in particular, pre- and post-conditions---and the big-O notation.

Activity completed. Accompany functions with appropriate pre- and post-conditions. Analyze the time complexity of simple algorithms in terms of the big-O notation, and choose among competing algorithms on the basis of their big-O behaviors.

Expected outcomes. Correct pre- and post-conditions for functions. Correct analyses of time complexity and applications of time comparisons.

Documentation. (i) Internal documentation in programs. (ii) Descriptions of algorithms' time complexities in program documentation. (iii) Answers to test questions addressing assertions and time complexity.

Stage 2. (CSCI 301) Further formalisms: recursive definitions and their relation to the development of algorithms, as in binary and other trees; time complexity.

Activity completed. Algorithms based on recursive definitions for operations on trees and more general graphs. Time complexity analysis of these algorithms.

Expected outcomes. Correct algorithms for processing trees and other graphs. Algorithms chosen on the basis of their time complexities.

Documentation. (i) Programs that implement tree and graph algorithms. (ii) Program documentation explaining and justifying algorithm choices in programs. (iii) Answers to exam questions on these topics.

Stage 3. (CSCI 330) Using formal structures when describing the syntax and semantics of programming languages.

Activity completed. Concepts of lexical analysis, parsing and semantic analysis; scope, binding, control flow and abstraction in various programming languages; Lambda calculus and functional programming.

Expected outcomes. An understanding of the formal description of a high-level programming language; aspects of compiling; distinguishing features of various paradigms of programming.

Documentation. (i) Programs that implement a lexical analyzer and a parser of a prototype language, (ii) Answers to assignments, quizzes and exam questions on these topics.

7. the ability to analyze various aspects of the process used when designing a system and the ability to employ established frameworks to evaluate the completed work

Stage 1. (CSCI 201)

Maintain records of time allocation during the development and completion of programming projects. Design and report on test cases for simple programs.

Activity completed. Maintain a record of time devoted to the component tasks in the completion of programming projects. Design test cases for programs and write a document reporting on those tests.

Expected outcomes. Records of time allocation during projects. Documents describing test development and use.

Documentation. Time logs for projects and testing documents.

Stage 2. (CSCI 301)

Maintain more detailed records of time allocation and productivity throughout the development and completion of programming projects. Design, carry out, and document tests for program of moderate complexity.

Activity completed. Keep records of time allocation and effectiveness throughout the design, implementation, testing, and description of programming projects. Design and document thorough tests of the resulting programs.

Expected outcomes. Summaries of the use of time and the effectiveness of effort through the process of building a program. Testing documents.

Documentation. The documents produced in this process: logs of time and results in program development and documentation of testing.

Stage 3. (CSCI 331)

Activity completed: Peer assessments of team members performance on team-based software projects. Team based peer assessments of other teams' software projects.

Expected outcomes: Rubric based Peer assessments of team members.

Rubric & Team based peer assessments of other teams' software projects.

Documentation: The documents produced in this process.

8. An ability to apply the principles learned in the core curriculum to various application domains, build on these principles and stay current in their knowledge.

Stage 1. (CSCI 4xx) Demonstrate a grasp of the learning outcomes described in the previous goals through test questions and projects.

Stage 2. (Post-completion survey.)