Computer Science III Metrics

Description

This course is the first in a three-course sequence that provides students with a foundation in computer science. Students develop fundamental programming skills using a language that supports an object-oriented approach, incorporating security awareness, human-computer interactions and social responsibility.

Among the associate-degree transfer programs, the Computer Science I-II-III coursework provides a common "core" body of knowledge in computing. The complete course sequence is designed in such a manner that students progress in knowledge, proficiency and professional maturity in several specific areas, including software engineering principles and professional and ethical conduct.

The progression of software engineering topics across CS I-II-III originates in CS I, where there is an emphasis on using a cyclic approach for program development by iterating through designing, coding, and testing program modules. Complemented by algorithm analysis, students are encouraged to think abstractly about problems and to begin developing processes for decomposing problems into organized parts. Encouraging clear documentation, good naming conventions and consistent secure coding style contribute to a disciplined approach to writing programs.

The progression of software engineering topics across CS I-II-III continues in CS II, where greater emphasis is placed on abstraction and sound software design principles, engaging students in the development of secure software components that solve a wide range of related problems and can be reused. The students determine the necessary elements of simple ADTs (such as a counter or a date) and then construct them; by their very nature, these components must be well-documented to encourage reuse. Additionally the students write assertions such as pre-conditions and post-conditions describing each class method, thereby encouraging students to think deeply about a simple problem before coding. After coding, the components must be well-tested, and therefore the use of test plans and test drivers are practiced. These activities reinforce the notion of constructing software from well-defined, independent pieces and complement the study of using existing library classes and APIs in software solutions.

The progression of software engineering topics across CS I-II-III concludes in CS III, where students are asked to step beyond the programmer role and take a broader view of software development; to consider its lifecycle from problem description to maintenance. Students first practice with analysis and design of medium-sized systems. Standard modeling tools are introduced and the students complete the phases of analysis, design, implementation and testing of a medium-sized team project that includes documents such as UML diagrams or CRC cards in addition to test plans. Students consider design patterns and write applications using data structures and templates. The software engineering topics are integrated with professionalism and ethics, as well as software and information assurance topics, such as security concerns and liabilities of computer-based systems.
The progression of the emphasis on professional and ethical conduct across CS I-II-III originates in CS I, where the curriculum is designed to consider the historical context of computing and programming as well as examining issues involving ethical conduct; plagiarism, intellectual property and software piracy issues are presented. Typically, student requirements for submitting original work as well as college policies regarding the use of computing resources and acceptable computing behavior on campus and the Brookings Institute "10 Commandments of Ethical Computing" can be used as relevant discussion starters.

The progression of the emphasis on professional and ethical conduct across CS I-II-III continues in CS II, which builds upon this foundation by examining societal issues, the Internet, and professionalism. Now that the students have gained some experience with developing programs, they can begin to see "what can go wrong" and the possible consequences to the user of their program, at a personal level, such as infinite loops and program crashes. Additionally, students are confronted with broader implications by design considerations regarding databases and data accessibility; ethical concerns regarding personal data, privacy and property rights should be explored. Integrating these topics with the software development process, security issues, and relevant cases of software errors will help students recognize that their work can have individual as well as societal consequences and encourage them to think carefully about the design and implementation of their programs.

The progression of the emphasis on professional and ethical conduct across CS I-II-III concludes in CS III, where a broader view is presented - encompassing computing sciences as a profession. Standards of professional behavior, organizations and publications are examined as well as a variety of occupational roles in the computing field. Course content materials presenting case studies of significant software failures amplify the topics of risks and liabilities. The students should start to recognize that invariably software production involves ethical choices. Incorporating these topics with the software lifecycle, engineering, human factors, and software assurance considerations will help students internalize the significance of professional and ethical behavior and subsequently demonstrate it through their individual and group projects.

Minimum Contact Hours

42

Assessment Rubric

CS. 3. Practice the tenets of ethics and professional behavior promoted by computing societies; accept the professional responsibilities and liabilities associated with software development.

Emerging Standard
Studies the tenets of ethics and professional behavior promoted by international computing societies, such as ACM and IEEE-CS.

Developed Standard
Practices the tenets of ethics and professional behavior promoted by international computing societies, and recognizes the liabilities associated with software development.

Highly Developed Standard
Displays ethical and professional behavior associated with the responsibilities of software development.

CS. 4. Analyze the efficiency of recursive algorithms.

Emerging Standard
With guidance interprets a recursive method.

Developed Standard...
Analyzes a recursive method and correctly predicts its output.
Highly Developed Standard
Evaluates recursive algorithms in terms of efficiency and time and space tradeoffs.

**CS. 11. Assess the appropriateness of using recursion to solve a given problem.**

Emerging Standard
Explains the utility of recursion to solve certain problems.
Developed Standard
Compares and contrasts the tradeoffs in terms of recursive and non-recursive solutions.
Highly Developed Standard
Justifies when to choose a recursive solution over a non-recursive solution (and vice versa) in terms of efficiency, Big-O, and comprehensibility.

**CS. 12. Compare and contrast a range of searching and sorting algorithms and analyze time and space efficiencies.**

Emerging Standard
Uses various searching and sorting algorithms, and investigates time and space tradeoffs.
Developed Standard
Compares and contrasts a range of searching and sorting algorithms for time and space efficiencies.
Highly Developed Standard
Critiques searching and sorting algorithms, including recursive solutions, for various algorithmic efficiencies and evaluates them in terms of Big-O.

**CS. 17. Create effective, efficient and secure software, reflecting standard principles of software engineering and software assurance.**

Emerging Standard
Calculates the risks and liabilities of a computer-based solution using standard software assurance and engineering principles.
Developed Standard
Creates an effective, efficient and secure solution, utilizing principles of software assurance and software engineering.
Highly Developed Standard
Judges the safety and security of a software solution.

**CS. 21. Design and construct programming solutions using a variety of recursive techniques.**

Emerging Standard
Converts a simple recursive algorithm into a working recursive method.
Developed Standard
With guidance develops recursive programming solutions for applications that use data structures such as trees and lists.
Highly Developed Standard
Independently designs and develops recursive programming solutions for applications that use backtracking and data structures such as trees and lists.

**CS. 22. Design and develop reusable software using appropriate data structures and templates.**

Emerging Standard
Differentiates among the classic data structures and selects a suitable data structure for use in an application.

Developed Standard
With some guidance designs and develops applications using appropriate data structures for a given problem.

Highly Developed Standard
Independently designs and develops applications using appropriate data structures and incorporates reusable software components in the solution.

**CS. 33. Use standard analysis and design techniques to produce a team-developed, medium-sized, secure software application that is fully implemented and formally tested.**

Emerging Standard
As part of a team, produces an executable, medium-sized software application that meets some program requirements and includes design documentation and some evidence of testing.

Developed Standard
As part of a team, produces a working, medium-sized software application on time that meets many program requirements including design and some test plan documentation.

Highly Developed Standard
As part of a team, successfully develops a medium-sized, secure software application on time that meets all program requirements including design and formal test plan documentation.

[Association for Computing Machinery](https://www.acm.org/)

- [Login](https://www.acm.org/login)
- [Acceptable Use](https://www.acm.org/acceptable-use)
- [Privacy Policy](https://www.acm.org/privacy-policy)

Copyright © 2021 ACM CCECC