ACM CCECC

Serving computing education communities since 1991.

Software Engineering Metrics

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. 5. Analyze the execution of searching and sorting algorithms.

Emerging Standard
Describes the execution trace of one searching algorithm and one sorting algorithm.
Developed Standard
Analyzes the execution of various searching and sorting algorithms.
Highly Developed Standard
Evaluates the execution of various searching and sorting algorithms including a recursive solution.

CS. 6. Apply fundamental theorems and rules of Calculus to differentiate and integrate algebraic, trigonometric, inverse trigonometric and transcendental functions.

Emerging Standard
Exhibits limited understanding when differentiating and integrating functions using differential calculus; additional review and support is required.
Developed Standard
Independently differentiates and integrates various functions using differential calculus.
Highly Developed Standard
Independently and with a thorough understanding differentiates and integrates various functions using differential calculus.

CS. 7. Apply fundamental theorems and rules of Calculus to evaluate limits and analyze the continuity of various functions.

Emerging Standard
Exhibits limited understanding when computing limits and determining continuity; additional review and support is required.
Developed Standard
Independently computes limits and analyzes the continuity of various functions.
Highly Developed Standard
Independently and with a thorough understanding computes limits and analyzes the continuity of various functions.

**CS. 8. Apply fundamental theorems and rules of differentiation to solve problems that model real-world situations.**

Emerging Standard
Requires a great deal of assistance to solve problems drawn from real-world situations.
Developed Standard
Independently solves problems drawn from real-world situations.
Highly Developed Standard
Very capably and independently solves problems drawn from real-world situations.

**SE. 1. Apply mathematical induction and other techniques to prove mathematical results.**

Emerging Standard
Recognizes valid proofs that use mathematical induction and other techniques.
Developed Standard
Given a simple problem, such as an identity, constructs a mathematical proof by induction.
Highly Developed Standard
Constructs mathematical proofs by induction and other techniques.

**CS. 9. Apply secure coding techniques to object-oriented programming solutions.**

Emerging Standard
Describes secure coding techniques of an object-oriented program, such as public versus private members, data integrity, and data typing.
Developed Standard
Applies secure coding techniques to an object-oriented program.
Highly Developed Standard
Devises a fully secure object-oriented program.

**CS. 10. Apply the program development process to problems that are solved using fundamental programming constructs and predefined data structures.**

Emerging Standard
Summarizes the phases of the program development cycle.
Developed Standard
With guidance during the design phase, produces working code and performs some testing.
Highly Developed Standard
Develops a working program solution by implementing design, coding, and testing that includes error checking.

**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. 13. Compare and contrast the primitive data types of a programming language; describe how each is stored in memory; and identify the criteria for selection.**

*Emerging Standard*
Names the built-in data types of the programming language.

*Developed Standard*
Differentiates among the built-in data types and explain when it is appropriate to choose one over another.

*Highly Developed Standard*
Consistently produces programming solutions with the correct data types implemented.

**SE. 2. Construct a preliminary investigative report for a proposed system that includes scheduling and plans for mitigating potential risks.**

*Emerging Standard*
Constructs a project plan that identifies major milestones and a development schedule.

*Developed Standard*
Constructs a project plan that identifies major milestones and a development schedule that includes contingencies for recovering from schedule set-backs.

*Highly Developed Standard*
Constructs a project plan with revisions that includes options analysis and estimates risk associated with different options.
CS. 14. Construct multiple-file or multiple-module programming solutions that use class hierarchies, inheritance, and polymorphism to reuse existing design and code.

Emerging Standard
Describes when inheritance and the use of class hierarchies is an appropriate design strategy.

Developed Standard
With guidance, produces a programming solution using inheritance and polymorphism.

Highly Developed Standard
Designs and constructs a programming solution using the features of inheritance and polymorphism appropriately.

CS. 15. Construct object oriented programming solutions for reuse, using ADTs that incorporate encapsulation, data abstraction, and information hiding.

Emerging Standard
Summarizes the concepts of encapsulation, data abstraction, and information hiding and explains how they apply to object-oriented programming.

Developed Standard
Organizes programming solutions that include encapsulation, information hiding, and data abstraction.

Highly Developed Standard
Constructs reusable software components that incorporate encapsulation, data abstraction, and information hiding.

CS. 16. Construct symbolic models of applied problems described in words.

Emerging Standard
Requires a great deal of assistance to interpret word problems drawn from a variety of sources into symbolic representations.

Developed Standard
Independently interprets word problems drawn from a variety of sources into symbolic representations.

Highly Developed Standard
Very capably and independently interprets word problems drawn from a variety of sources into symbolic representations.

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. 18. Create programming solutions that use data structures and existing libraries.

Emerging Standard
Produces programming solutions that use existing library code.
Developed Standard
Organizes programming solutions that incorporate appropriate data structures and pre-existing code.
Highly Developed Standard
Designs and develops programming solutions that use data structures, pre-existing libraries, and individual library code.

CS. 19. Decompose a program into subtasks and use parameter passing to exchange information between the subparts.

Emerging Standard
With guidance translates a problem into a programming solution with subtasks.
Developed Standard
With guidance for program analysis and design, decomposes a problem into program components that share data.
Highly Developed Standard
Independently analyzes a problem, formulates a design strategy, and decomposes a problem into program components that share data.

SE. 3. Decompose complex systems using best practice object-oriented analysis and design tools and techniques.

Emerging Standard
Completes a static analysis and design that Identifies the classes/objects of a complex system.
Developed Standard
Using static design patterns, successfully identifies the classes/objects and specifying the relationships among them.
Highly Developed Standard
Successfully models the dynamic interactions among the objects and external interfaces in addition to a well-crafted static analysis and design.

CS. 20. Describe the language translation phases of compiling, interpreting, linking and executing, and differentiate the error conditions associated with each phase.

Emerging Standard
Defines the programming language terms of compiling, interpreting, linking, executing, and error conditions.
Developed Standard
Describes the programming language translation phases of compiling, interpreting, linking, and executing, and differentiates the error conditions associated with each phase.

Highly Developed Standard
Compares the programming language translation phases of compiling, interpreting, linking, and executing, and distinguishes the error conditions associated with each.

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. 23. Design and develop secure and fault-tolerant programs that mitigate potential security vulnerabilities.
Emerging Standard
Summarizes important characteristics of software assurance, such as the elimination of buffer overflows, memory leaks and back-door access.
Developed Standard
Produces a fault-tolerant program using the foundations of software assurance to mitigate potential security vulnerabilities.
Highly Developed Standard
Designs and develops a secure and fault-tolerant programming solution utilizing principles of software assurance.

SE. 7. Design and implement consistent and coherent user-centered interfaces that comply with UI standard practices.
Emerging Standard
Creates a structured, simple UI.
Developed Standard
Using static design patterns, successfully identifies the classes/objects and specifying the relationships among them. Creates well-structured, consistent UI that provides all needed options and user feedback.
Highly Developed Standard
Creates a clear, consistent UI that is flexible enough to prevent user input errors.

CS. 24. Differentiate between the object-oriented, structured, and functional programming methodologies.

Emerging Standard
Recognizes the differences and similarities of the object-oriented, structured, and functional programming methodologies.
Developed Standard
Differentiates between the object-oriented, structured, and functional programming methodologies.
Highly Developed Standard
Compares and contrasts the three prominent methodologies of object-oriented, structured, and functional programming.

SE. 4. Evaluate and test software system requirements that considers both validation and verification.

Emerging Standard
Describes the major phases of the testing process for a medium-sized team project.
Developed Standard
Develops a testing plan and schedule to individually test all requirements of a medium-sized team project.
Highly Developed Standard
Successfully implements a testing plan for a medium-sized team programming project.

CS. 25. Examine the logical validity of arguments and proofs as they apply to Boolean expressions.

Emerging Standard
Identifies the properties and structures of Boolean algebra.
Developed Standard
Analyzes the steps to simplify a Boolean expression.
Highly Developed Standard
Constructs a proof using the laws of Boolean algebra.

CS. 26. Illustrate the basic terminology and properties of graphs and trees.

Emerging Standard
Defines terms and properties for graphs and trees.
Developed Standard
Given a problem description illustrates appropriate trees, binary search trees, weighted, directed and undirected graphs solutions.
Highly Developed Standard
Applies mathematical proofs to verify the properties of graphs.

CS. 27. Perform binary and hexadecimal conversions of numbers.
Emerging Standard
Converts binary numbers to their decimal equivalent.
Developed Standard
Converts positive numbers between bases 2, 10, and 16.
Highly Developed Standard
Performs two's complement to represent negative integers in binary.

CS. 28. Perform computations using recursively defined functions and structures.
Emerging Standard
Explains how a simple recursive function is evaluated.
Developed Standard
Computes the correct result produced by a recursive algorithm.
Highly Developed Standard
Constructs recursive algorithms for the solution of problems.

CS. 29. Produce algorithms for solving simple problems and trace the execution of computer programs.
Emerging Standard
Defines the steps necessary to solve a programming problem.
Developed Standard
Produces a working programming solution for a given algorithm.
Highly Developed Standard
Develops a generic solution for an algorithm that can be used to solve a range of related problems.

CS. 30 Produce graphical user interfaces that incorporate simple color models and handle events.
Emerging Standard
Differentiates between good and bad design concepts for human-computer interfaces.
Developed Standard
Produces programming code of a graphical user interface that utilizes a simple color model effectively and efficiently handles events triggered by user interaction.
Highly Developed Standard
Develops programming code for a graphical user interface that incorporates the concepts of good HCI design.

SE. 5. Solve problems involving sets, relations, functions, and congruences.
Emerging Standard
Defines the concepts of sets, relations, functions, and congruences.
Developed Standard
Solves problems about sets, relations, functions, and congruences.
Highly Developed Standard
Evaluates a problem and constructs an appropriate solution choosing among sets, relations, functions, and/or congruences.

**CS. 31. Use fundamental concepts of Calculus to construct graphs of polynomial, rational and exponential functions.**

Emerging Standard
Constructs graphs of polynomial, rational and exponential functions with frequent errors and omissions; displays a weak understanding.
Developed Standard
Independently constructs graphs of polynomial, rational and exponential functions.
Highly Developed Standard
Independently and with a thorough understanding constructs graphs of polynomial, rational and exponential functions.

**CS. 32. Use graphs and trees to solve problems algorithmically.**

Emerging Standard
Explains standard algorithms for graphs and trees, such as Eulerian circuits, spanning trees, and Kruskal’s algorithm.
Developed Standard
Applies traversal methods for graphs and trees.
Highly Developed Standard
Verifies the correctness of graph algorithms using mathematical proofs.

**SE. 6. Use methods of combinatorics to solve counting problems.**

Emerging Standard
Recognizes the need for combinatorial techniques such as combinations or permutations to solve a problem.
Developed Standard
Solves counting problems using combinatorial techniques such as combinations and permutations.
Highly Developed Standard
Decomposes a complex problem into combinatorial procedures.

**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.

**CS. 34. Verify program correctness through the development of sound test plans and the implementation of comprehensive test cases.**

Emerging Standard
Produces test plans for object-oriented programming solutions that considers code coverage.

Developed Standard
Analyzes a program and devises a test plan that examines code coverage and develops test cases for data coverage.

Highly Developed Standard
Constructs a test driver for code coverage and creates a formal test plan choosing comprehensive test cases for data coverage.

---

**Association for Computing Machinery**

- [Login](#)
- [Acceptable Use](#)
- [Privacy Policy](#)

Copyright © 2018 ACM CCECC