adaptive maintenance: Software maintenance to make a computer program usable in a changed environment.

conciseness: Software attributes that provide implementation of a function with a minimum amount of code.

consistency: Software attributes that provide uniform design and implementation techniques and notations.

correctability: The degree of effort required to correct software defects and to cope with user complaints.

corrective maintenance: Maintenance to correct hardware or software faults.

cross-reference tool: A software maintenance tool that lets the user determine where a variable is used or where a particular procedure is called on.

emergency maintenance: Unscheduled corrective maintenance to keep a system operational. [IEEE Std P1219-1992]

expandability: The degree of effort required to improve or modify software functions' efficiency.

forward engineering: Using software products derived from an existing system, together with new requirements, to produce a new system. [IEEE Std P1498-1995]

impact analysis: Identifies all system and software products that a change request affects and develops an estimate of the resources needed to accomplish the change. This includes determining the scope of the changes to plan and implement work, accurately estimating the resources needed to perform the work, and analyzing the requested changes’ cost and benefits.

inverse engineering: The process of obtaining a high-level representation of the software from the source code. Inverse engineering provides a more abstract view of the system with the intent of recapturing design and requirements information. [British view] See also reverse engineering.

maintainability: The average effort required to locate and fix a software failure. 1. Software maintainability measurements consist of correctability, expandability, and testability. 2. Software maintainability measurements consist of consistency, simplicity, conciseness, modularity, and self-descriptiveness.

maintenance manual (MM): A software engineering project deliverable document that enables a system’s maintenance personnel (rather than users) to maintain the system. Contrast with operator’s manual and users’ manual.

maintenance personnel: Software engineers who maintain software systems.

maintenance plan: A document that identifies the management and technical approach to use in maintaining a software product. Typical topics include tools, resources, facilities, and schedules. [ANSI/IEEE Std 830-1998]

maintenance project: A software development project described as maintenance to correct errors in an original requirements specification, to adapt a system to a new environment, or to enhance a system.

mean time to repair (or restore) (MTTR): The average time the maintenance team requires to implement a change and restore the system to working order.

modularity: Software attributes that provide a structure of highly independent components.

operation and maintenance phase: The period in the software life cycle during which a software product is employed in its operational environment, monitored for satisfactory performance, and modified as necessary to correct problems or respond to changing requirements or a changing environment.

perfective maintenance: Improvements in software’s performance or functionality—for example, in response to user suggestions and requests.

preventive maintenance: 1. Designing a software system that is easy to maintain. 2. Continuously upgrading a system to enable it to cope with current and future changes.

reengineering: The complete cycle of performing reverse engineering followed by forward engineering.

regression testing: Functional testing that follows modification and maintenance. Regression testing determines whether the modification has altered software functions that were remain unchanged.

reuse: Building a software system at least partly from existing pieces to perform a new application.

reverse engineering: Determining what existing software will do and how it’s constructed (to make intelligent changes).

self-descriptiveness: Software attributes that explain a function’s implementation.

simplicity: Software attributes that provide implementation of functions in the most understandable manner.

software maintenance: Modifying a software system or component after delivery to correct faults, improve performance, add new capabilities, or adapt to a changed environment. [IEEE Std 610.12-1991]

testability: The effort required to test software.