Compilation of Software Engineering Terms

allocation -- (1) The process of distributing requirements, resources, or other entities among the components of a system or program. (2) The result of the distribution in (1). [IEEE Std 610.12-1990]

analysis -- In system/software engineering, the process of studying a system by partitioning the system into parts (functions or objects) and determining how the parts relate to each other to understand the whole.

analytic technique (Software Quality -- Methods for SQA and V&V) -- *See analysis. See also technique.*

application frameworks -- A sub-system design made up of a collection of *abstract and concrete classes* and interfaces between them. Frameworks are often instantiation of a number of patterns.

architectural design -- (1) The process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system. (2) The result of the process in (1). [IEEE Std 610.12-1990]

architectural structure -- Synonym for architecture. See *architecture.*

architectural structures and viewpoints (Software Design -- Software Architecture) -- *See architectural structure, architectural view, architectural viewpoint*

architectural style -- (1) Defines a family of systems in terms of a pattern of structural organization. Commonly used styles include pipes and filters, layers, rule-based systems, and blackboards. [Shaw and Garlan 1996] (2) Characterizes a family of systems that are related by sharing structural and semantic properties.

architectural styles and patterns (Software Design -- Software Architecture) -- *See architectural styles, design patterns.*

architectural view -- A representation of a whole system from the perspective of a related set of concerns. [IEEE Std 1471-2000]

architectural viewpoint -- A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis. [IEEE Std 1471-2000]

architecture -- The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. [IEEE Std 1471-2000]

asset protection (Software Engineering Management -- Organizational Management and Coordination) -- See *asset, protection.*

audit -- An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria. [IEEE Std 610.12-1990]

audit, reviews, and inspections (Software Quality -- Methods for SQA and V&V) -- *see audit, reviews, and inspections*.

code (1) In software engineering, computer instructions and data definitions expressed in a programming language or in a form output by an assembler, compiler, or other translator. (2) To express a computer program in a programming language. (3) A character or bit pattern that is assigned a particular meaning; for example, a status code. [IEEE Std 610.12-1990]

code design (Software Construction and Implementation) -- Design of code. *See also code, design.*

code documentation (Software Construction and Implementation) -- *See documentation*.

code of ethics standard -- A standard that describes the characteristics of a set of moral principles dealing with accepted standards of conduct by, within, and among professionals. [IEEE Std 610.12-1990]

code tuning (Software Construction and Implementation) -- Changes made to program source code for the purpose of optimizing performance, usually to increase speed or reduce memory usage.

coding -- (1) In software engineering, the process of expressing a computer program in a programming language. (2) The transforming of logic and data from design specifications (design descriptions) into a programming language. [IEEE Std 610.12-1990]

configuration auditing -- In configuration management, an independent examination of the configuration status to compare with the physical configuration. *See also audit.*

configuration control -- In configuration management, an element of configuration management consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. [IEEE Std 610.12-1990]

configuration identification -- In configuration management, (1) An element of configuration management consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation. (2) The current approved technical documentation for a configuration item as set forth in specifications, drawings, associated lists, and documents referenced therein. [IEEE Std 610.12-1990]

configuration management (CM) -- In system/software engineering, a discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. [IEEE Std 610.12-1990]

configuration status accounting.-- In configuration management, an element of configuration management consisting of the recording and reporting of information needed to manage a configuration effectively. This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes. [IEEE Std 610.12-1990]

construction -- The general name for detailed design, coding, unit testing, and related activities?the collection of activities focused on creating source code.

construction planning (Software Construction and Implementation) -- *See construction, planning.*

construction QA (Software Construction and Implementation) -- *See construction, software quality assurance.*

construction tools (Software Construction and Implementation) -- Tools used during construction. *See also tool*

control -- In management, all the activities that ensure that actual work goes according to plan. It measures performance against goals and plans, reveals when and where deviation exists, and, by putting in motion actions to correct deviations, helps ensure the accomplishment of plans. [Thayer & Thayer *Project Management* 1997] 2. In engineering, the monitoring of system output to compare with expected output and taking corrective action when the actual output does not match the expected output. *Also known as feedback control.*

control process (e.g., problem tracking and resolution techniques) Software Engineering Management - Enactment) -- *See control.*

cost estimate -- 1. The estimated cost to perform a stipulated task or acquire an item. 2.The product of a cost-estimation method or model. [Thayer & Thayer *Project Management* 1997]

customer -- 1. In system/software system engineering, an individual or organization who specifies the requirements for and formally accepts delivery of a new or modified hardware/software product and its documentation. The customer may be internal or external to the parent organization of the project and does not necessarily imply a financial transaction between customer and developer. 2. The person or persons who pay for the project and usually (but not necessarily) decide the requirements; the customer may or may not be the user. [ANSI/IEEE Standard 830-1984]

customer relations (Software Engineering Management -- Organizational Management and Coordination) -- In system/software engineering, a set of agreements, both formal and informal, on how the customer will be treated by the developer. [Thayer & Thayer *European Perspective* 1993] *See also customer.*

data -- In system/software engineering, a representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means. [IEEE Std 610.12-1990]

data collection (Software Engineering Management - Measurement) -- The process of bringing together data from different sources to a central point for the purpose of being processed by a computer. [Thayer & Thayer *European Perspective* 1993] *See also data.*

data design -- Design of a program?s data, especially table design in database applications.

data design and management (Software Construction and Implementation) -- *See data design, data management.*

data management -- In software development, a term referring to systems offering users an interface that blocks the majority of file handling details. This leaves the user free to concentrate on the logical properties of the data. [*Oxford Dictionary of Computing 1990*]

data-structured design - - A software system engineering design methodology used for business applications and other systems with well-understood data structures. It produces a program structure by designing input/output data structures and relies on decomposition based on the higher-level nature of these data structures. [Thayer & Thayer *Software Requirements* 1997] *Contrast with object-oriented design, functional design.*

data-structure-oriented design (Software Design -- software design strategies and methods) -- *See data-structured design.*

deliverable determination (Software Engineering Management - Planning) -- *See deliverable document.*

deliverable document -- In software system engineering, a document that is deliverable to a customer. Examples are users' manual, operator's manual, and programmer's maintenance manual. [Thayer & Thayer *Software Requirements* 1997] *See also documentation.*

design -- (1) The process of defining the architecture, components, interfaces, and other characteristics of a system or component. (2) The result of the process in (1). [ANSI/IEEE Std 610.12 1990]

design framework-- *Synonym for architecture or sometimes simply for design*.

design of families of programs and framworks (Software Design -- Software Architecture) *See families of programs, design frameworks*

(design patterns (Software Design -- Software Architecture) A description of the problem and the essence of its solution so that the solution may be reused in different settings. The pattern is not a detailed specification, but a description of accumulated wisdom and experience. [Buschmann et al, 1996]

development tools and methods (Software Engineering Tools and Methods) -- *See tool, method.*

disaster recovery (Software Design -- Software Design Concepts) -- In computer system operations, the return to normal operation after a hardware or software failure. *See also disaster, recovery.*

distributed computing (Software Design -- Software Design Concepts) -- The spreading of computation and data across a number of computers connected by a network.

documentation -- (1) A collection of documents on a given subject. (2) Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results. (3) The process of generating or revising a document. (4) The management of documents, including identification, acquisition, processing, storage, and dissemination. [IEEE Std 610.12-1990]

dynamic -- Pertaining to an event or process that occurs during computer program execution; for example, dynamic analysis, dynamic binding. *Contrast with* static. [IEEE Std 610.12-1990]

dynamic techniques (Software Quality -- methods for SQA and V&V) -- *See dynamic*.

enactment (Software Engineering Management) -- The establishment of something by law, ruling, or other authoratative acts.

engineering economics (Professionalism and Engineering Economics) ?The methods and models for analyzing choices that software projects must make related to project costs and cost impacts. *See also economics, engineering.*

error (1) The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. For example, a difference of 30 meters between a computed result and the correct result. (2) An incorrect step, process, or data definition. For example, an incorrect instruction in a computer program. (3) An incorrect result. For example, a computed result of 12 when the correct result is 10. (4) A human action that produces an incorrect result. For example, an incorrect action on the part of a programmer or operator. [IEEE Std 610.12-1990]

error processing (Software Construction and Implementation) -- The detection of and responses to errors in a program. *See also error*.

ethics (Professionalism and Engineering Economics) -- A system of moral principles governing the appropriate conduct for an individual or group. [MSN Encarta 2000] *See also code of ethics standard.*

evaluation - (1) Determination of fitness for use. (2) The process of determining whether an item or activity meets specified criteria. [IEEE Std P1498, 1995]

evaluation tools -- An automatic or manual tool used for the evaluation of a software product or process. *See also evaluation, tool.*

execute -- To carry out an instruction, process, or computer program. [IEEE Std 610.12-1990]

families of programs -- Sets of programs that are related by sharing significant portions of requirements, design, and code, e.g., a program family might include one version of a program developed for an English-speaking audience, a second version of a program developed for a German-speaking audience, and a third version for a Japanese-speaking audience .

fault tolerance (Software Design -- Software Design Concepts) -- (1) The ability of a system or component to continue normal operation despite the presence of hardware or software faults. (2) The number of faults a system or component can withstand before normal operation is impaired. (3) Pertaining to the study of errors, faults, and failures, and of methods for enabling systems to continue normal operation in the presence of faults. [IEEE Std 610.12-1990]

functional decomposition -- In software engineering, the partitioning of higher-level system functions into smaller and smaller pieces to render them more manageable and understandable.

functional design -- (1) The process of defining the working relationships among the components of a system. (2) The result of the process in (1). [IEEE Std 610.12-1990] *Contrast with object-oriented design, data-structure design.*

function-oriented design - Software Design -- Software Design Strategies and Methods) -- *See functional design. See also architectural design.*

general design concepts (Software Design -- Software Design Concepts) -- A fundamental idea that can be applied to designing a system. For example, information hiding is a software design concept. [Gomaa 1997] *See also design.*

general strategies (Software Design Strategies and Methods (Software Design) -- The overall plan and direction for performing design. For example, *functional decomposition* is a software design strategy. [Gomaa 1997]

human factors in software design (e.g., human-computer interaction) (Software Design) -- *See human-engineering, user interface.*

human-engineering -- In system/software system engineering, is a multidisciplinary activity that applies the knowledge, derived from psychology and technology, to specify and design high quality human-machine interfaces. The human engineering process encompasses the following steps: activity analysis, semantic analysis and design, syntactic and lexical design, user environment design, and prototyping. . [Thayer & Thayer *Software Requirements* 1997]

implementation -- (1) The process of translating a design into hardware components, software components, or both. (2) The result of the process in (1). [IEEE Std 610.12-1990]

implementation plan (Software Engineering Management -- Enactment) -- *See implementation, coding.*

initiation and scope (Software Engineering Management) -- A non-specific term for work performed early in a software project that includes high-level statements of requirements and rough estimates of project cost and schedule.

inspections -- A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. Types include code inspections; design inspections. [IEEE Std 610.12-1990]

maintenance -- (1) The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment. (2) The process of retaining a hardware system or component in, or restoring it to, a state in which it can perform its required functions. [IEEE Std 610.12-1990]

maintenance tools and methods (Software Engineering Tools and Methods) -- *See tool, method.*

management of SCM process (software Configuration Management) -- *See configuration management (CM).*

management of tools and methods (Software Engineering Tools and Methods) -- *See management, tools, methods.*

measurement (Software Engineering Management) -- The process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules. [Fenton & Pfleeger 1996]

measurement applied to SQA and V & V (Software Quality) -- *See SQA measurement, V&V measurement.*

meaurement of program goals (Software Engineering Management -- Measurement) -- *See measurement, program goals*.

measurement of software and its development (Software Engineering Management - Measurement) -- *See software measurement.*

methods for SQA and V & V (Software Quality) -- *See SQA methods, V&V methods.*

metric -- A quantitative measure of the degree to which a system, component, or process possesses a given attribute. [IEEE Std 610.12-1990]

metric process implementation (Software Engineering Management - Planning) -- *See process implementation metric.*

monitor process (Software Engineering Management - Enactment) -- *See control.*

object-oriented design (Software Design -- Software Design Strategies and Methods) -- A software development technique in which a system or component is expressed in terms of objects and

connections between those objects. [IEEE Std 610.12-1990] *Contrast with functional design, data-structure design.*

organizational management -- All management activities that result in the design of a formal structure of tasks and authority. Organizing involves the determination and enumeration of the activities required to achieve the objects of the organization, the grouping of these activities, the assignment of such groups of activity to an organizational entity or group identifiers, the delegation of responsibilities and authority to carry them out, and the provisions for coordination of authoritative relationships. [Thayer & Thayer *Project Management* 1997]

organizational coordination (Software Engineering Management) Coordination of organizational management activities. -- *See also organizational management*.

pattern -- An abstraction from a concrete form that recurs in specific non-arbitrary contexts; within software, “pattern” most often refers to the key aspects of a common design structure. [Buschmann et al, 1996]

performance -- The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage. [IEEE Std 610.12-1990]

performance analysis techniques and tools (Software Design -- Software Design Quality Analysis and Evaluation) -- Techniques and tools that are used to measure and evaluate the performance of a software system*. See also performance, tools, and techniques.*

personnel -- Persons employed in any work, enterprise, service, or establishment. [Thayer & Thayer *Software Requirements* 1997]

personnnel management (Software Engineering Management -- Organizational Management and Coordination) -- The management of activities involving hiring, retaining, promoting, training, and terminating personnel. *See also personnel.*

plan implementation (Software Engineering Management -- Enactment) ?initiating and putting a plan into action. *See also execute.*

planning (Software Engineering Management) -- *See planning.*

planning (Software Maintenance) -- *See planning*.

planning for SQA and V & V (Software Quality) -- *See SQA planning*, V&V planning.

policy management (Software Engineering Management -- Organizational Management and Coordination) -- *See policy.*

post-closure activities (Software Engineering Management) -- Activities that occur after a software system has been formally accepted by its customer; these activities include but are not limited to post mortem reviews and archiving project materials. *See also project close out.*

process -- (1) A sequence of steps performed for a given purpose; for example, the software development process. (2) An executable unit managed by an operating system scheduler. (3) To perform operations on data. [IEEE Std 610.12-1990]

process definition (Software Engineering Process) Identification of a sequence of steps involving activities, constraints, and resources that are performed for a given purpose. -- *See process.*

process implementation and change (Software Engineering Process) -- *See process management.*

process infrastructure (process infrastructureSoftware Engineering Process) -- The internal structure of the software development process**,** to include lifecycle phases, documentation, baselines, reviews, and products.

process management (Software Engineering Process) -- The direction, control, and coordination or work performed to develop a product or perform a service. An example is quality assurance. [IEEE Std 610.12-1990]

process measurement (Software Engineering Process) -- Those measurements established for each step in the software engineering process that are used to determine its effectiveness. The metrics define the results of each process stage and relate them to the resources expended, errors introduced, errors removed, and various coverage, efficiency, and productivity indicators. [Gray 1992 Survey Tools]

process model (Software Engineering Process) -- A development strategy, incorporated by a software engineer or team of engineers, that encompasses process, methods, and tools. [Pressman 1997]

process planning (Software Engineering Management -- Planning) -- Planning the development and use of a software process. *See also process, planning.*

product attributes -- Characteristics of a software product. Can refer either to general characteristics such as reliability, maintainability, and usability or to specific features of a software product.

product attributes and measures (Software Design -- Software Design Quality Analysis and Evaluation) -- *See product attributes, product measures, quality attributes.*

product measure -- A metric (e.g. measures per person-month) that can be used to measure the characteristics of delivered documents and software. [Keller et al. 1990]

professional practice (Professionalism and Engineering Economics) -- Non-technical elements of a software engineer's work practices including responsibility to the public, responsibility to clients and employers, legal and ethical conduct, independence of judgment, and professional development.

project close out (Software Engineering Management) -- A plan to ensure orderly closeout of the software project. Items in the closeout plan includes a staff reassignment plan, a plan for archiving project materials, a plan for post-mortem debriefings of project personnel, and a final report to include lessons learned and analysis of project objectives achieved. [IEEE Std 1058-1998]

project management -- A system of procedures, practices, technologies, and know-how that provides the planning, organizing, staffing, directing, and controlling necessary to successfully manage an engineering project. Know-how in this case means the skill, background, and wisdom to apply knowledge effectively in practice. *See also software engineering project management.* [Thayer & Thayer *Project Management* 1997]

project plan -- A document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way that the project will be organized, e.g., a software development plan. [IEEE Std 610.12-1990]

qualitative process analysis (Software Engineering Process) -- Evaluation of software processes that are based on subjective, qualitative information rather than quantitative information, e.g. information gathered from questionnaires and interviews.

quality analysis -- (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements. (2) A set of activities designed to evaluate the process by which products are developed or manufactured. [IEEE Std 610.12-1990]

quality analysis and evaluation tools (Software Design -- Software Design Quality Analysis and Evaluation) -- *See quality analysis, evaluation tools.*

quality attribute -- A feature or characteristic that affects an item's quality. Note: In a hierarchy of quality attributes, higher-level attributes may be called quality factors, and lower level attributes may be called quality attributes. [IEEE Std 610.12-1990]

quality management (Software Engineering Management - Planning) -- That aspect of the overall management function that determines and implements the quality policy. [IEEE-Std-1074]

real-time concepts (Software Design -- Software Design Concepts) -- Pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process. [IEEE Std 610.12-1990]

requirement -- (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2). [IEEE Std 610.12-1990]

requirement analysis (software requirements) -- *See software requirements analysis***.**

requirements elicitation (software requirements) -- The process through which the customers (buyers and/or users) and developer (contractor) of a software system discover, review, articulate, and understand the requirements of the system. [Thayer 2001]

requirements engineering -- 1. a method of obtaining a precise formal specification from the informal and often vague requirements with a customer. 2. In system engineering, the science and discipline concerned with analyzing and documenting requirements. It comprises needs analysis, requirements analysis, and requirements specifications. [Sannella 1988], [Thayer & Thayer *European Perspective* 1993]

requirements engineering process (software requirements) -- *See requirements engineering.*

requirements management (software requirements) -- *See software requirements management.*

requirements validation (software requirements) -- *See validation. See also software requirements verification.*

requirements verification -- *See software requirements verification.*

resource allocation (Software Engineering Management -- Planning) -- The assignment or allocating of computer resources to different activities or tasks.

responsibility allocation -- The allocation (partitioning) of responsibility to different organizational units. *See also responsibility, allocation.*

review and evaluation (Software Engineering Management) -- *See reviews, evaluation.*

reviews -- A process or meeting during which a work product, or set of work products, is presented to project personnel, managers, users, customers, or other interested parties for comment or approval. Types include code reviews, design reviews, formal qualification reviews, and requirements reviews, test readiness reviews. [IEEE Std 610.12-1990]

risk management (Software Engineering Management) -- In system/software engineering, an “umbrella” title for the processes used to manage risk.. It is an organized means of identifying and measuring risk (risk assessment) and developing, selecting, and managing options (risk analysis) for resolving (risk handling) these risks. The primary goal of risk management is to identify and respond to potential problems with sufficient lead-time to avoid a crisis situation.

schedule and cost estimates (Software Engineering Management - Planning) -- The management activity of determining the probable cost of an activity or product and the time to complete the activity or deliver the product. *See also schedule.*

selection of measurement (Software Engineering Management - Measurement) -- The process of selection of appropriate measurements to aid in the management of a software development. *See also measurement.*

software -- Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system. *Contrast with hardware.* [IEEE Std 610.12-1990]

software acquisition (Software Engineering Management) -- The process of obtaining a system, software product or software service. [IEEE/EIA Standard 12207.0-1996] *See also software.*

software and system safety (e.g., hazard analysis) (Software Design) -- In safety-critical systems, freedom from unacceptable risk of personal harm. The likelihood that a system does not lead to a state in which human life or environment are endangered. Note: safety relates to all aspects of a system, its sub-systems, its environment, to human factors such as operator error or wrongdoing, and to incorrect data. [IEE/BCS Safety-Related Systems 1989]

software architecture (Software Design) -- *See architecture. See also software.*

software configuration auditing (Software Configuration Management) -- In configuration management, an independent examination of the configuration status to compare with the physical configuration. *See also audit.*

software configuration control (configuration control (Software Configuration Management) -- In configuration management, an element of configuration management consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. [IEEE Std 610.12-1990]

software configuration identification (Software Configuration Management) -- In configuration management, (1) An element of configuration management consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation. (2) The current approved technical documentation for a configuration item as set forth in specifications, drawings, associated lists, and documents referenced therein. [IEEE Std 610.12-1990]

software configuration management- In system/software engineering, a discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. [IEEE Std 610.12-1990]

software configuration status accounting (software configuration management) -- In configuration management, an element of configuration management consisting of the recording and reporting of information needed to manage a configuration effectively. This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes. [IEEE Std 610.12-1990]

software design- In software engineering, the process of defining the software architecture (structure), components, modules, interfaces, test approach, and data for a software system to satisfy specified requirements. [ANSI/IEEE Standard 729-1983]

software design concepts (Software Design) -- In software engineering, a fundamental idea that can be applied to designing a system. Fore example, Information hiding is a software design concept

software design context (Software Design -- Software Design Concepts) -- The framework and the environment in which the software design is developed

software design notation (Software Design) -- Software design notation or representation is a means of describing a software design. It may diagrammatic, symbolic, or textual. For example, structure charts and pseudocode are software design notations. [Gomaa 1997]

software design notation and documentation (Software Design) -- *See software design notation, software documentation*

software design process (Software Design -- Software Design Concepts) -- A systematic approach for carrying out design. It typically describes a sequence of steps for producing a design. [Gomaa 1997]

software design quality analysis and evaluation (Software Design) -- *See quality analysis, evaluation tools.*

software design strategies and methods (Software Design) -- The overall plan and direction for performing design.

software development methodology -- In software engineering: (1) an integrated set of software engineering methods, policies, procedures, rules, standards, techniques, tools, languages, and other methodologies for analyzing, designing, implementing, and testing software; and (2) a set of rules for selecting the correct methodology, process, or tools. [Thayer & Thayer *Project Management* 1997]

software documentation -- *See documentation.*

software engineering management - *See software engineering project management.*

software engineeing process -- The total set of software engineering activities needed to transform a user's requirements into software. [Humphrey 1989] *See also software development methodology.*

software engineering project management (SEPM) -- A system of procedures, practices, technologies, and know-how that provides the planning, organizing, staffing, directing, and controlling necessary to successfully manage a software engineering project. Know-how in this case means the skill, background, and wisdom to apply knowledge effectively in practice. [Thayer & Thayer *Project Management* 1997]

Software engineering tools and methods -- *See tool, method*, *software development methodology***.**

software maintenance -- *See maintenance (1)..*

software maintenance documentation (Software Maintenance) -- *See support manual.*

software maintenance management (Software Maintenance) -- *See management.*

software maintenance measurement (Software Maintenance) -- *See measurement.*

software maintenance planning (Software Maintenance) -- *See planning*.

software maintenance process (e.g., maintenance types, change impact analysis and regression testing) (Software Maintenance) -- *See process.*

software management process (software configuration management) -- *See software configuration management.*

software measurement -- *See measurement. See also software metric*.

software metric -- A quantitative measure of the degree to which a system, component, or process possesses a given attribute. [IEEE Std 610.12-1990

software metric model (Software Engineering Management -- Measurement) -- *See software metric.*

software quality-- In software engineering: 1. The totality of features and characteristics of a software product that affects its ability to satisfy given needs (for example, to conform to specifications). 2. The degree to which software possesses a desired combination of attributes. 3. The degree to which a customer or user perceives that his or her composite expectations are met. 4. The composite characteristics of software that determine the degree to which the software will meet the expectations of the customer. [ANSI/IEEE Standard 729-1983] 5. Attributes of software that affect its perceived value, for example, correctness, reliability, maintainability, and portability. 6. Software quality Includes fitness for purpose, reasonable cost, reliability, ease of use in relation to those who use it, design of maintenance and upgrade characteristics, and compares well against reliable products. [ACARD 1986] [Thayer & Thayer *European Perspective* 1993]

software quality assurance -- *See quality assurance.*

software quality concepts (Software Quality) -- *See software quality*.

software quality metric -- *See quality metric*

software release management (software configuration management) -- Management of activities surrounding release of one or more versions of software to one or more customers. Release management includes defining acceptable quality levels for release, authority to authorize the release, release procedures, and so on. *See also version.*

software requirements -- *See requirement, requirements engineering.*

software requirements analysis -- (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements. (2) The process of studying and refining system,

hardware, or software requirements. [IEEE Std 610.12-1990] (3) Reasoning and analyzing the customers and users needs to arrive at a definition of software requirements. [Thayer 2001]

software requirements management -- In system/software system engineering, the process of controlling the identification, allocation, and flowdown of requirements from the system level to the module or part level, including interfaces, verification, modifications, and status monitoring. [Thayer & Thayer *Software Requirements* 1997]

software requirements specification (software requirements) -- 1. A document that specifies the requirements for a system or component. Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards. [IEEE Std 610.12-1990] 2. A document that clearly and precisely records each of the requirements of the software system. [Thayer 2001]

software requirements verification -- Ensuring that the software requirements specification is in compliance with the system requirements, conforms to document standards of the requirements phase, and is an adequate basis for the architectural (preliminary) design phase. [Thayer 2001] *See also verification, requirements validation.*

software testing -- *See testing.*

software tool -- A computer program used in the development, testing, analysis, or maintenance of a program or its documentation. Examples include comparator, cross-reference generator, decompiler, driver, editor, flowcharter, monitor, test case generator, and timing analyzer. [IEEE Std 610.12-1990]

source code organization (Software Construction and Implementation) -- Arrangement of source code including layout of code within a single file and packaging of source code into modules, classes, physical files, and so on.

SQA (software quality assurance) -- (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements. (2) A set of activities designed to evaluate the process by which products are developed or manufactured. [IEEE Std 610.12-1990]

SQA measurement -- *See SQA, measurement.*

SQA methods -- *See SQA, methods*

SQA planning -- *See SQA, planning*

standards (Professionalism and Engineering Economics) -- Mandatory requirements employed and enforced to prescribe a disciplined uniform approach to software development, that is, mandatory conventions and practices are in fact standards. [IEEE Std 610.12-1990]

style -- A way of doing something; a particular technique or method to perform an activity.

support manual -- A document that provides the information necessary to service and maintain an operational system or component throughout its life cycle. Typically described are the hardware and software that make up the system or component and procedures for servicing, repairing, or reprogramming it. [IEEE Std 610.12-1990]

support methods -- Software methods that provide support to the development of a software system. *See also methods*, support software.

support software -- Software that aids in the development or maintenance of other software, for example, compilers, loaders, and other utilities. [IEEE Std 610.12-1990]

support tools and methods (Software Engineering Tools and Methods) -- *See support software, support methods*

system deployment -- 1. Release of software to end users or customers, i.e., ?release.-- 2. Phase of a project in which software is put into operation and cutover issues are resolved.

system integration -- In system/software system engineering, this ensures that the various segments and elements of the total system can interface with each other and with the external environment. [Thayer & Thayer *European Perspective* 1993]

system integration and deployment (Software Construction and Implementation) ?*See system integration, system deployment.*

task -- The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project members. Related tasks are usually grouped to form activities. [IEEE-Std-1074]

task and responsibility allocation (Software Engineering Management - Planning) -- *See task assignment, responsibility allocation.*

task assignment -- *See task.*

test coverage (software testing) -- The degree to which a given test or set of tests addresses all specified requirements for a given system or component. [IEEE Std 610.12-1990]

test coverage of code (software testing) -- The amount of code actually executed during the test process. It is stated as a percentage of the total instructions executed or paths traversed. [Gray 1992 Survey Tools]

test coverage of specifications (software testing) -- *See test coverage.*

test design (software testing) -- Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests. [IEEE Std 610.12-1990]

test documentation (software testing) -- Documentation describing plans for, or results of, the testing of a system or component. Types include test case specification, test incident report, test log, test plan, test procedure, test report. [IEEE Std 610.12-1990]

test execution (software testing) -- Act of performing one or more test cases.

test level (software testing) -- In software engineering, the effort associated with the execution of a set of tests governed by a common test plan, test specification, and/or test procedures document.

test management (software testing) -- *See management.*

testing strategies (software testing) -- In software engineering, one of a number of approaches used for testing software. For example, functional vs. structural testing, static vs. dynamics testing, and so forth. [Coward 1988]. [Thayer & Thayer *European Perspective* 1993]

tool -- 1. An item either used in performing an operation or necessary in the practice of a vocation or profession. 2. In engineering, a step-by-step formalized, manual, or automated process for solving engineering problems. 3. In software engineering, a process or method something used in the performance of a task or procedure used by programmers, software engineers, or managers in the performance of their tasks. *See also software tool.* [Thayer & Thayer *European Perspective* 1993]

types of tests (software testing) -- In system/software engineering, a description of different testing techniques and strategies; for example, unit, integration, system, and acceptance tests. Other descriptions are alpha (factory) and beta tests.

user interface -- An interface that enables information to be passed between a human user and hardware or software components of a computer system. [IEEE Std 610.12-1990]

V&V (verification and validation) -- The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements. [IEEE Std 610.12-1990]

V&V measurement -- See V&V, measurement.

V&V methods -- See V&V, methods.

V&V planning -- See V&V, planning.

Validation -- 1. The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. *Contrast with verification.* [IEEE Std 610.12-1990] 2. The process is a process for determining whether the requirements and the final, as-built system or software product fulfills its specific intended use [IEEE/EIA Std 12207.2, Para 6.5]

verification -- 1. The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that

phase. *Contrast with validation.* [IEEE Std 610.12-1990] 2. The verification process is a process for determining whether the software products of an activity fulfill the requirements or conditions imposed on them in the previous activities [IEEE/EIA Std12207.2, Para 6.4]

version -- (1) An initial release or re-release of a computer software configuration item, associated with a complete compilation or recompilation of the computer software configuration item. (2) An initial release or complete re-release of a document, as opposed to a revision resulting from issuing change pages to a previous release. See also: configuration control; version description document. [IEEE Std 610.12-1990]

Key terms used from standard reference.

It has been determined that for each of the following terms that the definition in reference [Webster] is adequate:

asset, communication, coordination, disaster, economics, engineering, feedback, management, methods, organization, planning, policy, project, protection, recovery, responsibility, schedule, technique

References

[ANSI/IEEE Standard 729-1983] IEEE Standard Glossary of Software Engineering Terminology, The Institute of Electrical and Electronics Engineers, Inc., NY, approved by American National Standards Institute Aug 9, 1983.

[Buschmann et al 1996] Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, Pattern-Oriented Software Architecture: A System of Patterns, Wiley, Chichester, 1996.

[Cleland and Kerzner 1985] Cleland, D.I., and H. Kerzner, *The Project Management Dictionary of Terms,* Van Nostrand Reinhold Company, New York, 1985.

[Fenton & Pfleeger 1996] Fenton, Norman E., Shari L. Pfleeger, *Software Metrics: A Rigorous & Practical Approach*, 2nd ed., PWS Publishing Company, International Thompson Computer Press, London, 1996.

[Gomaa 1997] Gomaa, Hassan, “Design Methods for Concurrent and Real-Time Systems,” in Dorfman, M, and R.H. Thayer (eds.), *Software Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1997

[IEEE Std 610.12-1990] Standard Glossary of Software Engineering Terminology, The Institute of Electrical and Electronics Engineers, Inc., NY.

[IEEE/EIA Standard 12207.0-1996] *Standard for Information Technology Software life cycle processes,* The Institute of Electrical and Electronics Engineers, Inc., NY

[IEEE/EIA Standard12207.2-1997] ***Guide for ISO/IEC 12207 lifecycle processes Implementation considerations,*** The Institute of Electrical and Electronics Engineers, Inc., NY

[IEEE-Std-1074] IEEE Standard for Developing Software Life Cycle Processes, The Institute of Electrical and Electronics Engineers, Inc., NY.

[IEEE-Std-1471] *IEEE Recommended Practice for Architectural Description for Architectural Description of Software-Intensive Systems*, The Institute of Electrical and Electronics Engineers, Inc., NY.

[Koontz and O'Donnell 1972] H. Koontz and C. O'Donnell, *Principles of Management: An Analysis of Managerial Functions*, 5th ed., McGraw-Hill Book Company, NY, 1972.

[IEEE/BCS Safety-Releated Systems 1989] IEEE/BCS, Software in Safety-Related Systems, The Institution of Electrical Engineering and The British Computer Society, October 1989.

[Ramamoorthy et al. 1984] Ramamoorthy, C.V., A. Prakash, W. Tsai, and Y. Usuda, "Software Engineering: Problems and Perspectives," IEEE Computer, Vol. 17, No. 10, October 1984, pp. 191-209. Corrected copy reprinted in R.H. Thayer (ed.), Tutorial: Software Engineering Project Management, IEEE Computer Society Press, Washington, DC, 1988.

[Sannella 1988] Sannella, D., "A Survey of Formal Software Development Methods," in R.H. Thayer and A.D. McGettrick (eds.), *Software Engineering: A European Prospective*, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[Shaw and Garlan 1996] Shaw, M. and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, Upper Saddle River, NJ, 1996.

[Thayer & Thayer *Project Management* 1997] Thayer, R.H. and M.C. Thayer, "Project Management Glossary" in Thayer, R.H. (ed.), *Software Engineering Project Management*, 2nd ed., IEEE Computer Society Press, Los Alamitos, CA, 1997, IEEE-CS Press catalog number 8000

[Thayer & Thayer *European Perspective* 1993] Thayer, R.H. and M.C. Thayer, "European Perspective Glossary" in Thayer, R.H., and A.D. McGettrick (eds.), *Software Engineering: A European Perspective,* IEEE Computer Society Press, Los Alamitos, CA, 1993, IEEE-CS Press catalog number Catalog number 2117

[Thayer & Thayer *Software Requirements* 1997] Thayer, R.H. and M.C. Thayer, "Software Requirements Engineering Glossary" in Thayer, R.H., and M. Dorfman (eds.), *Software Requirements Engineering,* 2nd ed., IEEE Computer Society Press, Los Alamitos, CA, 1997, IEEE-CS Press catalog number 7738

[Thayer 2001] Thayer, R.H. "*Software Requirements Engineering*," in supplement to Dorfman, M, and R.H. Thayer (eds.), Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, 2001, IEEE-CS Press catalog number 7609

[Webster] Merriam-Webster's Collegiate Dictionary, Tenth Edition.