Software Non-functional Assessment Process (SNAP) counting unit (SCU). (1) component or activity, in which non-functional complexity and size are measured *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* The SCU is identified according to the nature of each sub-category. It may contain both functional and non-functional characteristics; therefore, sizing an SCU is performed for its functional sizing, using function point analysis (FPA), and for its non-functional sizing, using SNAP. *Syn:* SNAP counting unit

Software Non-functional Assessment Process (SNAP) sub-category. (1) component, a process or an activity executed within the project, to meet a non-functional requirement *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* A non-functional requirement may consume more than one sub-category. *Syn:* SNAP sub-category

Software Non-functional Assessment Process (SNAP) point. (1) unit of measurement to express the size of a non-functional requirement *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* The non-functional size of the software non-functional assessment process (SNAP) counting units (SCUs) is identified in a sub-category. After identifying all the SCUs that are provided to meet the non-functional requirements (NFR), the SNAP points (SPs) of all SCUs are added together to obtain the calculated SPs. *Syn:* SNAP point, software non-functional assessment process point

software. (1) computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(2)** all or a part of the programs, procedures, rules, and associated documentation of an information processing system *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1) (ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.34)* **(3)** program or set of programs used to run a computer *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 4.46)* **(4)** all or part of the programs which process or support the processing of digital information *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.49)* **(5)** part of a product that is the computer program or the set of computer programs *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.34) Note:* Software excludes information per se, such as the content of documents, audio and video recordings, graphics, and databases. Digital information which is managed by executable software (e.g. the content of documents and databases) is not considered software, even though program execution may depend on data values. *Syn:* SW *See also:* application software

acquirer. (1) stakeholder that acquires or procures a product or service from a supplier *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 4.1) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.1)* **(2)** person or organization that acquires or procures a system, software product, or software service (which can be part of a system) from a supplier *(ISO/IEC TR 12182:2015 Systems and software engineering--Framework for categorization of IT systems and software, and guide for applying it, 3.13)* **(3)** individual or organization that acquires or procures a system, software product or software service from a supplier *(ISO/IEC 25040:2011 Systems and software engineering--Systems and* *software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.1) Syn:* buyer, owner, purchaser, internal/organizational sponsor *See also:* customer

developer. (1) individual or organization that performs development activities (including requirements analysis, design, testing through acceptance) during the system or software life-cycle process *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.6) (ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.12)* **(2)** an organization that develops software products **(3)** person who applies a methodology for some specific job, usually an endeavor *(ISO/IEC 24744:2014 Software Engineering--Metamodel for development methodologies, 3.11) Note:* Developers apply methodologies via enactment. *See also:* implementer

firmware. (1) combination of a hardware device and computer instructions and data that reside as read-only software on that device *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(2)** ordered set of instructions and associated data stored in a way that is functionally independent of main storage, usually in a ROM *(ISO/IEC 2382:2015 Information technology -- Vocabulary) Note:* The software cannot be readily modified under program control.

life cycle model. (1) framework of processes and activities concerned with the life cycle that can be organized into stages, which also acts as a common reference for communication and understanding *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.27) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.22)* **(2)** framework containing the processes, activities, and tasks involved in the development, operation, and maintenance of a software product, spanning the life of the system from the definition of its requirements to the termination of its use *(ISO/IEC 15940:2013 Systems and software engineering--Software Engineering Environment Services, 2.1) Syn:* life-cycle model

version. (1) initial release or re-release of a computer software configuration item, associated with a complete compilation or recompilation of the computer software configuration item *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(2)** initial release or complete re-release of a document, as opposed to a revision resulting from issuing change pages to a previous release *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(3)** identified instance of a configuration item *(ISO/IEC TR 18018:2010 Information technology--Systems and software engineering--Guide for configuration management tool capabilities, 3.15)* **(4)** unique string of number and letter values indicating a unique revision of an item *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.54) Note:* Versions often identify revisions of software that provide unique functionality or fixes. A version typically has multiple parts, such as a major version, indicating large changes in functionality or user
interface changes, and a minor version, indicating smaller changes in functionality or user interface changes. *See also:* release

user. (1) individual or group that interacts with a system or benefits from a system during its utilization *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.70) (ISO/IEC/IEEE 15939:2017 Systems and software engineering--Measurement process, 3.40) (ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality* *models, 4.3.16) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.53) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.60)* **(2)** person who interacts with a system, product or service *(ISO/IEC 25064:2013 Systems and software engineering--Software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: User needs report, 4.17)* **(3)** the person (or persons) who operates or interacts directly with a software-intensive system **(4)** any person or thing that communicates or interacts with the software at any time *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.27) (ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.50) (ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis, B) (ISO/IEC 14143-1:2007 Information technology--Software measurement--Functional size measurement; Part 1: Definition of concepts) (ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, 3.9)* **(5)** individual or group that benefits from a system during its utilization *(ISO/IEC 25022:2016, Systems and software engineering -- Systems and software quality requirements and evaluation (SQuaRE) -- Measurement of quality in use, 4.26)* **(6)** person who performs one or more tasks with an automated system; a member of a specific audience *(ISO/IEC/IEEE 26512:2018 Systems and software engineering--Requirements for acquirers and suppliers of information for users, 3.25)* **(7)** person (or instance) who uses the functions of a CBSS via a terminal (or an equivalent machine-user-interface) by submitting tasks and receiving the computed results *(ISO/IEC 14756:1999 Information technology -- Measurement and rating of performance of computer-based software systems, 4.31)* **(8)** individual or group that benefits from a ready to use software product during its utilization *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.26)* **(9)** individual or organization that uses the system or software to perform a specific function *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.40)* **(10)** individual who or group that benefits from a system during its utilization *(INCOSE Systems Engineering Handbook, 5th ed.)* **(11)** person who interacts with an autonomous/intelligent system *(IEEE 7010-2020, IEEE Recommended Practice for Assessing the Impact of Autonomous and Intelligent Systems on Human Well-Being, 2.1)* **(12)** person who interacts with the product *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.47) Note:* User can include persons who install, operate, service, maintain, or dispose of the product. The user can perform other roles, such as acquirer or maintainer. The role of user and the role of operator can be vested, simultaneously or sequentially, in the same individual or organization. *See also:* developer, end user, functional user, indirect user, operator, secondary user

non-deliverable item. **(1)** hardware or software product that is not required to be delivered under the contract, but may be employed in the development of a product *(ISO/IEC/IEEE 24765g:2018) Syn:* nondeliverable item

qualification testing. **(1)** testing conducted to determine whether a system or component is suitable for operational use *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)* **(2)** testing conducted on a hardware element, software element, or system to evaluate conformance with specified requirements *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1) See also:* acceptance testing, development testing, operational testing

release. (1) particular version of a configuration item that is made available for a specific purpose *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.12) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.43)* **(2)** collection of new or changed configuration items that are tested and introduced into a live environment together *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(3)** collection of one or more new or changed configuration items deployed into the live environment as a result of one or more changes *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.28)* **(4)** software version that is made formally available to a wider community *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(5)** delivered version of an application which includes all or part of an application *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(6)** set of grouped change requests, established in the Application Change Management Process, which are designed, developed, tested, and deployed as a cohesive whole *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.28)* **(7)** distribution of a product increment to a customer or users *(ISO/IEC TR 24587:2021, Software and systems engineering--Agile development--Agile adoption considerations, 3.13) Note:* Release management includes defining acceptable quality levels for release, authority to authorize the release, and release procedures. *See also:* version

security. (1) protection against intentional subversion or forced failure *(ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.41)* **(2)** protection against intentional subversion or forced failure, containing a composite of four attributes: confidentiality, integrity, availability, and accountability, plus aspects of a fifth, usability, all of which have the related issue of their assurance *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.49) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.45)* **(3)** degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.6)* **(4)** protection of computer hardware or software from accidental or malicious access, use, modification, destruction, or disclosure *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(5)** combination of people, process, and technology to protect data from unauthorized access and use *(IEEE 7005 2021, IEEE Standard for Transparent Employer Data Governance, 3.1)* **(6)** degree to which an IT service protects both users' assets and access to their information so that users have the degree of information access appropriate to their levels of authorization *(ISO/IEC TS 25011:2017 Information technology--Systems and software Quality Requirements and Evaluation (SQuaRE)--Service quality models, 3.2.3)* **(7)** resistance to intentional, unauthorized acts designed to cause harm or damage to a system *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.16) Note:* Security includes authenticity, accountability, confidentiality, integrity, availability, nonrepudiation, and reliability, all of which have the related issue of their assurance. Security pertains to personnel, data, communications, and the physical protection of computer installations.

software product. (1) set of computer programs, procedures, and possibly associated documentation and data *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.31) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.54) (ISO/IEC/IEEE 23026:2015 Systems and software engineering--Engineering and management of websites for systems, software, and services information, 3.48)* **(2)** any of the individual items in (1) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** complete set of software designed for delivery to a software consumer or end-user which can include computer programs, procedures and associated documentation and data *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.46)* **(4)** set of computer programs, procedures, database- and other data structure descriptions and associated documentation *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.33) Note:* A software product can be designated for delivery, an integral part of another product, or used in development. Software products vary from large customized application software for one customer to standard software packages that are sold off the shelf to millions of customers. *See also:* software package

software unit. (1) atomic-level software component of the software architecture that can be subjected to standalone testing *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.57) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.49)*

supplier. (1) organization or individual that enters into an agreement with the acquirer for the supply of a product or service *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.60) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.45) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.52)*

(2) individual or organization that enters into a contract with the acquirer for the supply of a system, software product or software service under the terms of the contract *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.37) (ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.63)* **(3)** organization or part of an organization or individual that enters into an agreement with the application management organization for the supply of a product, service, materials, or human capacity *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.34)* **(4)** individual or organization that provides products *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.39) Note:* The acquirer and the supplier sometimes are part of the same organization. The application management organization can have internal or external suppliers. A supplier can be another application management organization, but also IT infrastructure management organizations or consultants. *Syn:* agency, contractor, distributor, producer, retailer, seller, SUP, vendor

system. (1) combination of interacting elements organized to achieve one or more stated purposes *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.38) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.61) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1:*

Guidelines for life cycle management, 3.53) **(2)** product of an acquisition process that is delivered to the user *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.1)* **(3)** something of interest as a whole or as comprised of parts *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 6.5)* **(4)** interacting combination of elements to accomplish a defined objective *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK), 1.1.6)* **(5)** set of interrelated elements considered in a defined context as a whole and separated from their environment *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.41)* **(6)** software or the combination of other components, including hardware and human managed processes *(IEEE 7002:2022, IEEE Standard for Data Privacy Process, 3.1) Note:* A system is sometimes considered as a product or as the services it provides. In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g., aircraft system. Alternatively, the word 'system' can be replaced by a context-dependent synonym, e.g., aircraft, though this obscures the system perspective. A complete system includes all of the associated equipment, facilities, material, computer programs, firmware, technical documentation, services, and personnel required for operations and support to the degree necessary for self-sufficient use in its intended environment.

validation. (1) confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.41) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.71) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.54) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.62)* **(2)** process of providing evidence that the system, software, or hardware and its associated products satisfy requirements allocated to it at the end of each life cycle activity, solve the right problem (e.g., correctly model physical laws, implement business rules, and use the proper system assumptions), and satisfy intended use and user needs *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1.35)* **(3)** the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(4)** process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(5)** confirmation in a timely manner, through automated techniques where possible, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1) Note:* Validation in a system life cycle context is the set of activities for gaining confidence that a system is able to accomplish its intended use, goals, and objectives (meet stakeholder requirements) in the intended operational environment. The right system has been built or is operating to meet business objectives. Validation demonstrates that the system can be used by the users for their specific tasks. "Validated" is used to designate the corresponding status. [ISO 9000:2005] Multiple validations can be carried out if there are different intended uses. *See also:* verification

verification. (1) confirmation, through the provision of objective evidence, that specified requirements have been fulfilled *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.43) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.72) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.55) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.63)* **(2)** evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(3)** 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 *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1.36)* **(4)** process of providing objective evidence that the system, software, or hardware and its associated products conform to requirements (e.g., for correctness, completeness, consistency, and accuracy) for all life cycle activities during each life cycle process (acquisition, supply, development, operation, and maintenance), satisfy standards, practices, and conventions during life cycle processes, and successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle activities *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(5)** confirmation in a timely manner, using automated techniques where possible, through the provision of objective evidence, that specified requirements have been fulfilled *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1) Note:* Verification is a set of activities that compares a system or system element to the required characteristics. This can include, but is not limited to, specified requirements, design description, and the system itself. The system has been built right. "Verified" is used to designate the corresponding status. Verification of interim work products is essential for proper understanding and assessment of the life cycle phase product(s). A system can be verified to meet the stated requirements, yet be unsuitable for operation by the actual users. *See also:* validation

enabling system. (1) system that supports a system-of-interest during its life cycle stages but does not necessarily contribute directly to its function during operation *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.20) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.15) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.19) Note:* For example, when a system-of-interest enters the production stage, an enabling production system is required. Each enabling system has a life cycle of its own. *See also:* software development environment

system element. (1) member of a set of elements that constitutes a system *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.62) (ISO/IEC 15026-3:2015 Systems and software engineering -- Systems and software assurance -- Part 3: System integrity levels, 3.22) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.54) Note:* A system element is a discrete part of a system that can be implemented to fulfill specified requirements. *See also:* software/system element, software element

tailoring. (1) adaptation of a software process by adding, modifying, and deleting process activities that are deemed inapplicable for the project *(Software Extension to the PMBOK(R) Guide Fifth Edition)* **(2)** process by which individual requirements in specifications, standards, and related documents are evaluated and made applicable to a specific project by selection, and in some exceptional cases, modification of existing or addition of new requirements *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.38)*

(3) determining the appropriate combination of processes, inputs, tools, techniques, outputs, and life cycle phases to manage a project *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(4)** manner in which any selected issue is addressed in a particular project *(INCOSE Systems Engineering Handbook, 5th ed.) Note:* Tailoring may be applied to various aspects of the project, including project documentation; processes and activities performed in each life cycle stage; or the time and scope of reviews, analysis, and decision making, consistent with applicable statutory requirements.

critical information. (1) information describing the safe use of the software, the security of the information created with the software, or the protection of the sensitive personal information created by or stored with the software *(ISO/IEC/IEEE 15289:2019 Systems and software engineering--Content of life-cycle information items (documentation), 5.6)*

software user documentation. (1) electronic or printed body of material that provides information to users of software *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.35)*

tutorial. (1) instructional procedure in which the user exercises system functions, often using sample data that is supplied with the system or information for users *(ISO/IEC/IEEE 26512:2018 Systems and software engineering--Requirements for acquirers and suppliers of information for users, 3.26)* **(2)** instructional information in which the user exercises software functions using sample data that is supplied with the software or information for users *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.52)*

application software. (1) software designed to help users perform particular tasks or handle particular types of problems, as distinct from software that controls the computer itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary, 4.5)* **(2)** software or a program that is specific to the solution of an application problem *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(3)** software designed to fulfill specific needs of a user *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** software of an application *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.6) Note:* Application software is the software that the application management organization produces, services, and maintains. There is also system software: the software to produce and maintain the application software and to run the application software on its platform. The application management organization is one of the users of the system software. *See also:* application

SCMPI. (1) Software Configuration Management Planned Information *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

SPMPI. (1) Software Project Management Planned Information *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

SRMPI. (1) Software Release Management Planned Information *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

software characteristic. (1) inherent, possibly accidental, trait, quality, or property of software *(ISO/IEC/IEEE*

24765:2017 Systems and software engineering--Vocabulary)

software feature. (1) software characteristic specified or implied by requirements documentation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

software test incident. (1) event occurring during the execution of a software test that requires investigation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

test objective. (1) reason for performing testing *(ISO/IEC/IEEE 29119-2:2021, Software and systems engineering--Software testing--Part 2: Test processes, 3.49)* **(2)** identified set of software features to be measured under specified conditions by comparing actual behavior with the required behavior *(ISO/IEC 25062:2006 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Common Industry Format (CIF) for usability test reports, 4.9)* **(3)** identified set of software characteristics to be measured under specified conditions by comparing actual behavior with the required behavior *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.20)*

unit requirements documentation. (1) license for software use as originally purchased or procured, and which can typically be linked directly to purchase records *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

incident. (1) anomalous or unexpected event, set of events, condition, or situation at any time during the life cycle of a project, product, service, or system *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.23) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.17) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.22)* **(2)** unplanned event or occurrence resulting in damage or other loss *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.22) See also:* software test incident

unit. (1) separately testable element specified in the design of a computer software component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** logically separable part of a computer program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** software component that is not subdivided into other components *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** software element that is not subdivided into other elements *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)* **(5)** piece or complex of apparatus serving to perform one particular function *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)*

anomaly. (1) anything observed in the documentation or operation of a system that deviates from expectations based on previously verified system, software, or hardware products or reference documents *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)*

component. (1) entity with discrete structure, such as an assembly or software module, within a system considered at a particular level of analysis *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.3.3)* **(2)** one part that makes up a system *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(3)** object that encapsulates its own template, so that the template can be interrogated by interaction with the component *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 9.26)* **(4)** specific, named collection of features that can be described by an IDL component definition or a corresponding structure in an interface repository *(ISO/IEC 19500-3:2012 Information technology--Object Management Group--Common Architecture Request Broker Architecture (CORBA)--Part 3: Components, 4.1)* **(5)** functionally or logically distinct part of a system *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)* **(6)** object with a discrete information type that is stored in a component content management system, such as a topic, prerequisite, section, image, or video *(ISO/IEC/IEEE 26531:2023 Systems and software engineering -- Content management for product lifecycle, user and service management information for users, 3.1.3)* **(7)** product used as a constituent in an assembled product, system or plant *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.4)* *Note:* A component can be hardware or software and can be subdivided into other components. Component refers to a part of a whole, such as a component of a software product or a component of a software identification tag. The terms module, component, and unit are often used interchangeably or defined to be subelements of one another in different ways depending upon the context. The relationship of these terms is not standardized. A component can be independently managed or not from the end-user or administrator's point of view. *See also:* element, unit

component testing. (1) testing of individual hardware or software components *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)*

domain analysis. (1) analysis of systems within a domain to discover commonalities and differences among them *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(2)** process by which information used in developing software systems is identified, captured, and organized so that it can be reused to create new systems, within a domain *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(3)** result of the domain analysis process *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)*

integration testing. (1) testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

integrity level. (1) value representing project-unique characteristic, such as complexity, criticality, risk, safety level, security level, desired performance, and reliability, that define the importance of the system, software, or hardware to the user *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1.15)* **(2)** degree to which software complies or must comply with a set of stakeholder-selected software or software-based system characteristics defined to reflect the importance of the software to its stakeholders *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** symbolic value representing a degree of compliance within an integrity level scheme *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** claim of a system, product, or element that includes limitations on a property's values, the claim's scope of applicability, and the allowable uncertainty regarding the claim's achievement *(ISO/IEC/IEEE 15026-1:2019 Systems and software engineering--Systems and* *software assurance--Part 1: Concepts and vocabulary, 3.3.1)* **(5)** required degree of confidence that the system-of-interest meets the associated integrity level claim *(ISO/IEC 15026-3:2015 Systems and software engineering -- Systems and software assurance -- Part 3: System integrity levels, 3.7) Note:* Generally, the intention is that maintaining limitations on a property's values related to the relevant items will result in maintaining system risks within limits. The words 'integrity level' form an indivisible label and do not depend on a concept of integrity by itself. An integrity level is different from the likelihood that the integrity level claim is met but they are closely related. The word 'confidence' implies that the definition of integrity levels can be a subjective concept. integrity levels are defined in terms of risk and hence, cover safety, security, financial and any other dimension of risk that is relevant to the system-of-interest.

life cycle processes. (1) set of interrelated or interacting activities that result in the development or assessment of system, software, or hardware products *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1) Note:* Each activity consists of tasks. The life cycle processes can overlap one another. For V&V purposes, no process is concluded until its development products are verified and validated according to the defined tasks in the validation and verification plan. *Syn:* life-cycle processes

minimum tasks. (1) those verification and validation tasks required for the integrity level assigned to the system, software, or hardware to be verified and validated *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* **(2)** those tasks required for the integrity level assigned to the software to be tested *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

software design description (SDD). (1) representation of software created to facilitate analysis, planning, implementation, and decision-making *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1.28) Note:* The software design description is used as a medium for communicating software design information and can be thought of as a blueprint or model of the system.

software requirements specification (SRS). (1) documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1.29)*

test design. (1) documentation specifying the details of the test approach for a system, software, or hardware feature or combination of features and identifying the associated tests *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1.32) Note:* commonly includes the organization of the tests into groups

design level. (1) design decomposition of the software item *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

pass/fail criteria. (1) decision rules used to determine whether a software item or a software feature passes or fails a test *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.10) Syn:* pass-fail criteria

software item. (1) aggregation of software, such as a computer program or database, that satisfies an end use function and is designated for specification, qualification testing, interfacing, configuration management, or other purposes *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(2)** source code, object code, control code, control data, or a collection of these items *(ISO/IEC/IEEE 12207:2017 Systems and* *software engineering--Software life cycle processes, 3.1.53) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.49)* **(3)** identifiable part of a software product *(ISO/IEC/IEEE 90003:2018 Software engineering -- Guidelines for the application of ISO 9001:2015 to computer software, 3.14) (IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(4)** all or part of the programs, procedures, rules, and associated documentation of an information processing system *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.20) See also:* computer software component, computer software configuration item, software configuration item

defect. (1) imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced *(ISO/IEC 23531:2020, Systems and software engineering Capabilities of issue management tools, 3.1)* **(2)** an imperfection or deficiency in a project component where that component does not meet its requirements or specifications and needs to be either repaired or replaced *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(3)** generic term that can refer to either a fault (cause) or a failure (effect) *(IEEE 982.1-2005 IEEE Standard Dictionary of Measures of the Software Aspects of Dependability, 2.1)* **(4)** fault or deviation from the intended level of performance of a system or software *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.4)* **(5)** imperfection or deficiency in a work product or characteristic that does not meet its requirements or specifications *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1) See also:* fault

fault. (1) manifestation of an error in software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** incorrect step, process, or data definition in a computer program *(ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.27)* **(3)** situation that can cause errors to occur in an object *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 13.6.3)* **(4)** defect in a hardware device or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(5)** defect in a system or a representation of a system that if executed/activated could potentially result in an error *(ISO/IEC/IEEE 15026-1:2019 Systems and software engineering--Systems and software assurance--Part 1: Concepts and vocabulary, 3.4.6) Note:* A fault, if encountered, can cause a failure. Faults can occur in specifications when they are not correct. *Syn:* bug

software reliability. (1) probability that software will not cause the failure of a system for a specified time under specified conditions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The probability is a function of the inputs to and use of the system as well as a function of the existence of faults in the software. The inputs to the system determine whether existing faults, if any, are encountered.

category. (1) specifically defined division or grouping of software based upon one or more attributes or characteristics *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** subset of categorization space, which the stakeholders are interested in, specified using a combination of one or more equivalence classes *(ISO/IEC TR 12182:2015 Systems and software engineering--Framework for categorization of IT systems and software, and guide for applying it, 3.9)*

previously developed software. (1) software that has been produced prior to or independent of the project for which the plan is prepared, including software that is obtained or purchased from outside sources *(IEEE 1228-1994 (R2002) IEEE Standard for Software Safety Plans, 3.1.2)* **(2)** software that has been produced prior to or independent of the project for which the Plan is prepared, including software that is obtained or purchased from outside sources *(IEEE Std 1228-1994 IEEE Standard for Software Safety Plans, 3.1.2)*

software hazard. (1) software condition that is a prerequisite to an accident *(IEEE 1228-1994 (R2002) IEEE Standard for Software Safety Plans, 3.1.5) See also:* system hazard

software safety. (1) freedom from software hazards *(IEEE 1228-1994 (R2002) IEEE Standard for Software Safety Plans, 3.1.6)* **(2)** ability of software to be free from unacceptable risk *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.21) Note:* ability of software to resist failure and malfunctions that can lead to death or serious injury to people, loss or severe damage to property, or severe environmental harm *See also:* system safety

software safety program. (1) systematic approach to reducing software risks *(IEEE 1228-1994 (R2002) IEEE Standard for Software Safety Plans, 3.1.7)*

system hazard. (1) system condition that is a prerequisite to an accident *(IEEE 1228-1994 (R2002) IEEE Standard for Software Safety Plans, 3.1.8) See also:* software hazard

system safety. (1) freedom from system hazards *(IEEE Std 1228-1994 IEEE Standard for Software Safety Plans, 3.1.9)* **(2)** ability of a system to be free from unacceptable risk *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.25) Note:* A systems-based approach to safety involves the application of scientific, technical, and managerial skills to hazard identification, hazard analysis, and elimination, control, or management of hazards throughout the life-cycle of a system, program, project or an activity or a product. *See also:* software safety

deleted source statement. (1) source statement that is removed or modified from an existing software product as a new product is constructed *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

developed source statement. (1) source statement that is newly created for, added to, or modified for a software product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

input primitive. (1) the effort to develop software products, expressed in units of staff-hours *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

logical source statement (LSS). (1) software instruction, independent of the physical format (lines of code) in which it appears *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* physical source statement

origin attribute. (1) classification of software as either developed or nondeveloped *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

source statements (SS). (1) encoded logic of the software product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Source statements can be classified by function as executable, data declaration, compiler directive, or comment. They can also be classified as deliverable or nondeliverable.

support staff-hour. (1) hour of effort expended by a member of the staff who does not directly define or create the software product, but acts to assist those who do *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

critical range. (1) values used to classify software into the categories of acceptable, marginal, or unacceptable *(ISO/IEC/IEEE 24765j:2021)*

critical value. (1) value of a validated metric that is used to identify software that has unacceptable quality *(ISO/IEC/IEEE 24765j:2021)*

measurement. (1) set of operations having the object of determining a value of a measure *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) --Guide to SQuaRE, 4.20) (ISO/IEC/IEEE 15939:2017 Systems and software engineering--Measurement process, 3.17)* **(2)** process to determine a value *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.34)* **(3)** use of a metric to assign a value (e.g. a number or category) from a scale to an attribute of an entity *(ISO/IEC 14102:2008 Information Technology - Guideline for the evaluation and selection of CASE tools)* **(4)** assignment of values and labels to software engineering work products, processes, and resources plus the models that are derived from them, whether these models are developed using statistical or other techniques *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK), 7)*

metric. (1) quantitative measure of the degree to which a system, component, or process possesses a given attribute *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** defined measurement method and the measurement scale *(ISO/IEC 14102:2008 Information Technology - Guideline for the evaluation and selection of CASE tools)* **(3)** measure or unit of measure that is designed to facilitate decision-making and improve performance and accountability through collection, analysis, and reporting of relevant data *(IEEE 7005 2021, IEEE Standard for Transparent Employer Data Governance, 3.1) See also:* software quality metric

quality attribute. (1) characteristic of software, or a generic term applying to quality factors *(ISO/IEC/IEEE 24765j:2021)*

constraint. (1) limitation or implied requirement that constrains the design solution or implementation of the systems engineering process and is not changeable by the enterprise *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes)* **(2)** restriction on software life cycle process (SLCP) development *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)* **(3)** a statement that expresses measurable bounds for an element or function of the system **(4)** rule that specifies a valid condition of data *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.41)* **(5)** a rule that specifies a valid condition of data **(6)** responsibility that is a statement of facts that are required to be true in order for the constraint to be met *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.41)* **(7)** restriction on the value of an attribute or the existence of any object based on the value or existence of one or more others *(ISO/IEC 15474-1:2002 Information technology -- CDIF framework -- Part 1: Overview, 4.2)* **(8)** a responsibility that is a statement of facts that are required to be true in order for the constraint to be met **(9)** externally imposed limitation on system requirements, design, or implementation or on the process used to develop or modify a system *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-*

Requirements engineering, 4.1.6) **(10)** an externally imposed limitation on system requirements, design, or implementation or on the process used to develop or modify a system **(11)** a limiting factor that affects the execution of a project, program, portfolio, or process *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(12)** externally imposed limitation on the system, its design, or implementation or on the process used to develop or modify a system *(ISO/IEC/IEEE 15026-1:2019 Systems and software engineering--Systems and software assurance--Part 1: Concepts and vocabulary, 3.1.6) Note:* A constraint is a factor that is imposed on the solution by force or compulsion and can limit or modify the design changes.

iteration. (1) performing a sequence of steps repeatedly *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** short time frame in which a set of software features is developed, leading to a working product that can be demonstrated to stakeholders *(ISO/IEC/IEEE 26515: 2018 Systems and software engineering: Developing information for users in an agile environment, 3.10) Note:* In agile, a typical iteration length is two to four weeks. *See also:* instance, invocation, mapping, sprint

mapping. (1) assigned correspondence between two things that is represented as a set of ordered pairs *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.107)* **(2)** establishing a sequence of activities according to a selected software life cycle model (SLCM) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary) See also:* instance, invocation, iteration, software life cycle model (SLCM)

process architect. (1) person or group that has primary responsibility for creating and maintaining the software life cycle process (SLCP) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

product. (1) an artifact that is produced, is quantifiable, and can be either an end item in itself or a component item *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(2)** part of the equipment (hardware, software and materials) for which usability is to be specified or evaluated *(ISO/IEC TR 25060:2010 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: General framework for usability-related information, 2.9)* **(3)** result of a process *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.36) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.34)* **(4)** intended or accomplished result of labor, or of a natural or artificial process *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.28)* **(5)** artifact that is produced, is quantifiable and is deliverable to the user as either an end item in itself or a component item *(ISO/IEC 25030:2019 Systems and software engineering--Systems and software quality requirements and evaluation (SQuaRE)--Quality requirements framework, 3.12) Note:* Generic product categories are hardware (e.g., engine mechanical part); software (e.g., computer program); services (e.g., transport); and processed materials (e.g., lubricant). Hardware and processed materials are generally tangible products, while software or services are generally intangible. Most products comprise elements belonging to different generic product categories. Whether the product is then called hardware, processed material, software, or service depends on the dominant element. Results could be components, systems, software, services, rules, documents, or many other items.
The result could in some cases be many related individual results. *See also:* activity, deliverable, result

software life cycle (SLC). (1) project-specific sequence of activities that is created by mapping the activities of a standard onto a selected software life cycle model (SLCM) *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)* **(2)** software system or software product cycle initiated by a user need or a perceived customer need and terminated by discontinued use of the product or when the software is no longer available for use *(ISO/IEC/IEEE 24765e:2015)*

CASE tool. (1) software product that can assist software and system engineers by providing automated support for software and system engineering life-cycle activities *(ISO/IEC 15940:2013 Systems and software engineering--Software Engineering Environment Services, 2.3)* **(2)** software product that can assist software engineers by providing automated support for software life-cycle activities *(ISO/IEC 14102:2008 Information Technology - Guideline for the evaluation and selection of CASE tools, 3.2) Note:* A CASE tool can provide support in only selected functional areas or in a wide variety of functional areas.

adaptive maintenance. (1) modification of a software product, performed after delivery, to keep a software product usable in a changed or changing environment *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.1) Note:* Adaptive maintenance provides enhancements necessary to accommodate changes in the environment in which a software product operates. These changes help keep pace with the changing environment.

corrective maintenance. (1) modification of a software product performed after delivery to correct discovered problems *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.4)* **(2)** maintenance performed to correct faults in hardware or software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The modification repairs the software product to satisfy defined system requirements.

perfective maintenance. (1) modification of a software product to provide enhancements for users, improvements of information for users, and recording to improve software performance, maintainability, or other software attributes *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.9) See also:* adaptive maintenance, corrective maintenance

repository. (1) organized and persistent data storage that allows data retrieval *(ISO/IEC/IEEE 26511:2018 Systems and software engineering--Requirements for managers of information for users of systems, software, and services, 3.1.24)* **(2)** location/format in which such a collection is stored *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(3)** collection of all software-related artifacts belonging to a system *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(4)** collection of all system element or software-related artifacts belonging to a system *(ISO/IEC 29110-2-1:2015 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 2-1: Framework and taxonomy, 4.49)* **(5)** place where work products and the associated information items are or can be stored for preservation and retrieval *(ISO/IEC/IEEE 42020:2019 Software, systems and enterprise -- Architecture processes, 3.19) Note:* Artifacts are, for example, the software engineering environment. Benchmarking repository is a repository which is designated for use as the source of comparative measures for the purpose of benchmarking. In a repository, work products and other items are preserved for future retrieval when needed,
whereas in a library, working data is temporarily stored and retrieved as necessary.

reverse engineering. (1) determining what existing software will do and how it is constructed (to make intelligent changes) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** a software engineering approach that derives a system's design or requirements from its code *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software maintenance. (1) totality of activities required to provide support to a software system *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.12)* **(2)** entitlement of additional rights (such as additional functionality, upgrade, or support) for a previously granted software entitlement *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.35) Note:* Pre-delivery activities include planning for post-delivery operations, supportability, and logistics determination. Post-delivery activities include software modification, training, and operating a help desk.

performance requirement. (1) measurable criterion that identifies a quality attribute of a function or how well a functional requirement shall be accomplished *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)* **(2)** system or software requirement specifying a performance characteristic that a system/software system or system/software component must possess *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** requirement that imposes conditions on a functional requirement *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* A performance requirement is an attribute of a functional requirement. *See also:* nonfunctional requirement

state. (1) at a given instant in time, the condition of an object that determines the set of all sequences of actions (or traces) in which the object can participate *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 8.8)* **(2)** condition that characterizes the behavior of a function, subfunction or element at a point in time *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-Requirements engineering, 3.1.30)* **(3)** unique value that represents the stage of progress of software in its execution *(ISO/IEC 11411:1995 Information technology -- Representation for human communication of state transition of software, 2.1) Note:* A state is a shorthand representation for the unit's interaction occurrence history. Ascribing "state" to a unit implies that it has a capability to modify future behaviors as a result of past interactions with its environment.

added source statements. (1) count of source statements that were created specifically for the software product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

commercial-off-the-shelf (COTS). (1) [software] product available for purchase and use without the need to conduct development activities *(ISO/IEC/IEEE 90003:2018 Software engineering -- Guidelines for the application of ISO 9001:2015 to computer software, 3.4) Note:* COTS software product includes the product description (including all cover information, data sheet, web site information, etc.), the user documentation (necessary to install and use the software), the software contained on a computer sensible media (disk, CD-ROM, internet downloadable, etc.). Software is mainly composed of programs and data. This definition applies also to product descriptions, user documentation and software which are produced and supported as separate manufactured goods, but for which typical commercial fees and licensing considerations do not apply. *Syn:* commercial off the shelf, Commercial-Off-The-Shelf *See also:* software product

modified-off-the-shelf (MOTS). (1) software product that is already developed and available, usable either 'as is' or with modification, and provided by the supplier, acquirer, or a third party *(IEEE 1062-2015 IEEE Recommended*

Practice for Software Acquisition, 3.1)

software acquisition process. (1) period of time that begins with the decision to acquire a software product and ends when the product is no longer available for use *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

end user. (1) person who directly uses the system for its intended purpose *(ISO/IEC/IEEE 24765e:2015)* **(2)** the person or persons who will ultimately be using the system for its intended purpose **(3)** individual person who ultimately benefits from the outcomes of the system or software *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.7)* **(4)** any person that communicates or interacts with the software at any time *(ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, 3.5)* **(5)** person or persons who will ultimately be using the system for its intended purpose *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.13)* **(6)** individual person who ultimately benefits from the ready-to-use software product functionalities *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.7) Note:* An end user will generally be defined in terms of a specific software component of a system. *Syn:* end-user *See also:* direct user, functional user, indirect user, operator, secondary user, user

function. (1) a task, action, or activity that must be accomplished to achieve a desired outcome. **(2)** defined objective or characteristic action of a system or component *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.23)* **(3)** software module that performs a specific action, is invoked by the appearance of its name in an expression, receives input values, and returns a single value *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** part of an application that provides facilities for users to carry out their tasks *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 4.21)* **(5)** elementary unit of requirements and specifications defined and used for measurement purposes *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis)* **(6)** single-valued mapping *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.65)* **(7)** transformation of inputs to outputs, by means of some mechanisms, and subject to certain controls, that is identified by a function name and modeled by a box *(IEEE 1320.1-1998 (R2004) IEEE Standard for Functional Modeling Language - Syntax and Semantics for IDEF0, 2.1.53)*

model. (1) representation of a real-world process, device, or concept *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** representation of something that suppresses certain aspects of the modeled subject *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.115)* **(3)** interpretation of a theory for which all the axioms of the theory are true *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.115)*

(4) related collection of instances of meta-objects, representing (describing or prescribing) an information system, or parts thereof, such as a software product *(ISO/IEC 15474-1:2002 Information technology -- CDIF framework -- Part 1: Overview, 4.2)* **(5)** semantically closed abstraction of a system or a complete description of a system from a particular perspective *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(6)** system of postulates, value declarations and inference rules presented as a description of a state of affairs (universe of discourse) *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 7.3)* **(7)** representation of a system of interest, from the perspective of a related set of concerns *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)* **(8)** algorithm or calculation combining one or more base or derived measures with associated decision criteria *(ISO/IEC/IEEE 15939:2017 Systems and software engineering--Measurement process, 3.27)* **(9)** abstract representation of an entity or collection of entities that provides the ability to portray, understand or predict the properties or characteristics of the entity or collection under conditions or situations of interest *(ISO/IEC/IEEE 42020:2019 Software, systems and enterprise -- Architecture processes, 3.13)* **(10)** output of a machine learning algorithm trained with a training dataset that generates predictions using patterns in the input data *(ISO/IEC TR 29119-11:2020, Software and systems engineering--Software testing--Part 11: Guidelines on the testing of AI-based systems, 3.1.46) Note:* representation of the concepts or properties of an entity and governing principles is captured in architecture models

system requirements specification (SyRS). (1) structured collection of information that embodies the requirements of the system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** structured collection of the requirements (functions, performance, design constraints, and attributes) of the system and its operational environments and external interfaces *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-Requirements engineering, 4.1.29) See also:* software requirements specification, SRS

traceability. (1) discernible association among two or more logical entities, such as requirements, system elements, verifications, or tasks *(ISO/IEC TR 29110-1:2016 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 1: Overview, 3.71) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.52)* **(2)** degree to which a relationship can be established among two or more logical entities, especially entities having a predecessor-successor relationship to one another, such as requirements, system elements, verifications, or tasks *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.69)* **(3)** degree to which each element in a software development product establishes its reason for existing *(ISO/IEC TR 18018:2010 Information technology--Systems and software engineering--Guide for configuration management tool capabilities, 3.14)* **(4)** degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor relationship to one another *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)* **(5)** degree to which the IT service outcomes can be traced to or from the user needs *(ISO/IEC TS 25011:2017 Information technology--Systems and software Quality Requirements and Evaluation (SQuaRE)--Service quality models, 3.2.3.3) Note:* Software features and test cases are typically traced to software requirements.

architectural design. (1) process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** result of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* functional design

argument. (1) independent variable *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** specific value of an independent variable *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** constant, variable, or expression used in a call to a software module to specify data or program elements to be passed to that module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

automated verification system. (1) software tool that accepts as input a computer program and a representation of its specification and produces, possibly with human help, a proof or disproof of the correctness of the program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software tool that automates part or all of the verification process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

call. (1) transfer of control from one software module to another, usually with the implication that control will be returned to the calling module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** computer instruction that transfers control from one software module to another as in (1) and often specifies the parameters to be passed to and from the module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** to transfer control from one software module to another as in (1) and, often, to pass parameters to the other module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** request for service(s) or action(s) with respect to an application or a related service *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.9) Note:* A call might concern a request for service, information or advice; disruption or error reporting (incident); request for change; assignment (for instance an instruction to start an off-schedule production run); and complaint. *See also:* go to

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 *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** to express a computer program in a programming language *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** character or bit pattern that is assigned a particular meaning *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* source code, object code, machine code, micro code

code generator. (1) software tool that accepts as input the requirements or design for a computer program and produces source code that implements the requirements or design generator *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* application

coding. (1) in software engineering, the process of expressing a computer program in a programming language *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** transforming of logic and data from design specifications (design descriptions) into a programming language *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

compatibility. (1) degree to which a product, system or component can exchange information with other products, systems or components, or perform its required functions, while sharing the same hardware or software environment *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.3)* **(2)** ability of two or more systems or components to exchange information *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** capability of a functional unit to meet the requirements of a specified interface without appreciable modification *(ISO/IEC 2382:2015 Information*

technology -- Vocabulary)
correctness. (1) degree to which a system or component is free from faults in its specification, design, and implementation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** degree to which software, documentation, or other items meet specified requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** degree to which software, documentation, or other items meet user needs and expectations, whether specified or not *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** degree to which an IT service uses the correct process and produces the correct results with accurate data *(ISO/IEC TS 25011:2017 Information technology--Systems and software Quality Requirements and Evaluation (SQuaRE)--Service quality models, 3.2.1.2)*
cycle. (1) period of time during which a set of events is completed *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** set of operations that is repeated regularly in the same sequence, possibly with variations in each repetition *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software development cycle, software life cycle
detailed design. (1) process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to be implemented *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** result of the process in (1) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* low-level design, software development process
documentation. (1) collection of documents related to a given subject *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.10)* **(2)** information that explains how to use software, devices, applications, or services *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.10)* **(3)** information that explains how to use a product *(ISO/IEC/IEEE 26512:2018 Systems and software engineering--Requirements for acquirers and suppliers of information for users, 3.11) Note:* can be provided as separate documentation or as embedded documentation or both *See also:* document, information for users
driver. (1) software module that invokes and, perhaps, controls and monitors the execution of one or more other software modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** computer program that controls a peripheral device and, sometimes, reformats data for transfer to and from the device *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* test driver
fault tolerance. (1) degree to which a system, product or component operates as intended despite the presence of hardware or
software faults *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.5.3)* **(2)** pertaining to the study of errors, faults, and failures, and of methods for enabling systems to continue normal operation in the presence of faults *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* error tolerance, fail safe, fail soft, fault secure, robustness
hardware monitor. (1) device that measures or records specified events or characteristics of a computer system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software tool that records or analyzes

hardware events during the execution of a computer program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

host machine. (1) the computer on which a program or file is installed *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** in a computer network, a computer that provides processing capabilities to users of the network *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** computer used to develop software intended for another computer *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** computer used to emulate another computer *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

implementation. (1) process of translating a design into hardware components, software components, or both *(ISO/IEC/IEEE 90003:2018 Software engineering -- Guidelines for the application of ISO 9001:2015 to computer software, 3.4) (ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** result of the process in (1) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** definition that provides the information needed to create an object and allow the object to participate in providing an appropriate set of services *(ISO/IEC 19500-2:2012 Information technology --Object Management Group--Common Object Request Broker Architecture (CORBA)--Part 2: Interoperability, 3.2.8)* **(4)** installation and customization of packaged software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(5)** construction *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(6)** system development phase at the end of which the hardware, software and procedures of the system become operational *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(7)** process of instantiation whose validity can be subject to test *(ISO/IEC 10746-3:2009 Information technology -- Open Distributed Processing -- Reference Model: Architecture, 9.1.2) See also:* coding

kernel. (1) that portion of an operating system that is kept in main memory at all times *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** a software module that encapsulates an elementary function or functions of a system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* resident control program *See also:* nucleus, supervisory program

maintenance. (1) process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.9)* **(2)** process of retaining a hardware system or component in, or restoring it to, a state in which it can perform its required functions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** actions intended to retain a product in, or restore it to, a useful and safe condition, in which it can perform the intended use *(IEC/IEEE 82079-1:2019 Preparation of information for use (instructions for use) of products: Part 1: Principles and general requirements, 3.23) Note:* In the context of dependability, maintenance is a combination of all technical and management actions intended to retain an item in, or restore it to, a state in which it can perform as required. *See also:* adaptive maintenance, corrective maintenance, perfective maintenance, software maintenance

parameter. (1) variable that is given a constant value for a specified application *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** constant, variable, or expression that is used to pass values between software modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** symbol that can take a range of values defined by a set *(ISO/IEC 15909-1:2019 Systems and software engineering--High-level Petri nets--Part 1: Concepts, definitions and graphical notation, 3.24)* **(4)** parts of the model that are learned from applying the training data to the algorithm *(ISO/IEC TR 29119-11:2020, Software and systems engineering--Software testing--Part 11: Guidelines on the testing of AI-based systems, 3.1.53) See also:* adaptation

patch. (1) modification made directly to an object program without reassembling or recompiling from the source program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software component that, when installed, directly modifies files or device settings related to a different software component without changing the version number or release details for the related software component *(ISO/IEC 19770-2:2015 (corr 2017), Information technology -- Software asset management -- Part 2: Software identification tag, 3.1.1)* **(3)** modification to a source or object program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** to perform a modification as in (1), (2), or (3) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

path. (1) in software engineering, a sequence of instructions that are performed in the execution of a computer program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** in file access, a hierarchical sequence of directory and subdirectory names specifying the storage location of a file *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** sequence of executable statements of a test item *(ISO/IEC/IEEE 29119-4:2021 Software and systems engineering -- Software testing -- Part 4: Test techniques, 3.39)*

preliminary design review (PDR). (1) review conducted to evaluate the progress, technical adequacy, and risk resolution of the selected design approach for one or more configuration items; to determine each design's compatibility with the requirements for the configuration item; to evaluate the degree of definition and assess the technical risk associated with the selected manufacturing methods and processes; to establish the existence and compatibility of the physical and functional interfaces among the configuration items and other items of equipment, facilities, software and personnel; and, as applicable, to evaluate the preliminary operational and support documents *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** review as in (1) of any hardware or software component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* critical design review, system design review

quality metric. (1) quantitative measure of the degree to which an item possesses a given quality attribute *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

recursion. (1) process in which a software module calls itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** process of defining or generating a process or data structure in terms of itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* simultaneous recursion

recursive. (1) pertaining to a software module that calls itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** pertaining to a process or data structure that is defined or generated in terms of itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* reflexive

requirements analysis. (1) process of studying user needs to arrive at a definition of system, hardware, or software requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** process of studying and refining system, hardware, or software requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** systematic investigation of user requirements to arrive at a definition of a system *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(4)** determination of product- or service-specific performance and functional characteristics based on analyses of customer needs, expectations, and constraints; operational concept; projected utilization environments for people, products, services, and processes; and measures of effectiveness *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

return. (1) to transfer control from a software module to the module that called it *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** to assign a value to a parameter that is accessible by a calling module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** computer instruction or process that performs the transfer in (1) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* return code

run. (1) in software engineering, a single, usually continuous, execution of a computer program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** to execute a computer program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* run time

SDD. (1) software design description *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation)* **(2)** *(deprecated)* software design document *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** system design document *(ISO/IEC TR 29110-5-6-2:2014 Systems and software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 5-6-2: Systems engineering--Management and engineering guide: Generic profile group: Basic profile, 4.2)*

software engineering. (1) systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software *(ISO/IEC 2382:2015 Information technology -- Vocabulary) (ISO/IEC/IEEE 24748-5:2017 Systems and software engineering--Life cycle management--Part 5: Software development planning, 3.16)* **(2)** application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK)) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.52) Syn:* SE, SWE

software requirements review (SRR). (1) review of the requirements specified for one or more software configuration items to evaluate their responsiveness to and interpretation of the system requirements and to determine whether they form a satisfactory basis for proceeding into preliminary design of the configuration items *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** review as in (1) for any software component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* This review is called software specification review by the US Department of Defense. *See also:* system requirements review

SRR. (1) software requirements review *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** system requirements review *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.2) See also:* SAR

structured design. (1) disciplined approach to software design that adheres to specified rules based on principles such as modularity, top-down design, and stepwise refinement of data, system structures, and processing steps

(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) **(2)** result of applying the approach in (1) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data structure-centered design, input-process-output, modular decomposition, object-oriented design, rapid prototyping.

stub. (1) skeletal or special-purpose implementation of a software module, used to develop or test a module that calls or is otherwise dependent on it *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** computer program statement substituting for the body of a software module that is or will be defined elsewhere *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** engineering object in a channel, which interprets the interactions conveyed by the channel, and performs any necessary transformation or monitoring based on this interpretation *(ISO/IEC 10746-3:2009 Information technology -- Open Distributed Processing -- Reference Model: Architecture, 8.1.9)* **(4)** scaffolding code written for the purpose of exercising higher-level code before the lower-level routines that will ultimately be used are available *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

task. (1) required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.66) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.51) (ISO/IEC/IEEE 24748-1:2018 Systems and software engineering--Life cycle management--Part 1: Guidelines for life cycle management, 3.58)* **(2)** in software design, a software component that can operate in parallel with other software components *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** activities required to achieve a goal *(ISO/IEC TR 25060:2010 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: General framework for usability-related information, 2.13)* **(4)** set or sequence of activities required to achieve a given goal *(ISO/IEC 25023:2016, Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Measurement of system and software product quality, 4.12)* **(5)** recommended action intended to contribute to the achievement of one or more outcomes of an architecture process *(ISO/IEC/IEEE 42020:2019 Software, systems and enterprise -- Architecture processes, 3.23) Note:* Related tasks are usually grouped to form activities.

test readiness review (TRR). (1) review conducted to evaluate preliminary test results for one or more configuration items; to verify that the test procedures for each configuration item are complete, comply with test plans and descriptions, and satisfy test requirements; and to verify that a project is prepared to proceed to formal testing of the configuration items *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** review as in (1) for any hardware or software component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* code review, formal qualification review, design review, requirements review

function point (FP). (1) a unit which expresses the size of an application or of a project *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis)* **(2)** unit of measurement to express the amount of functionality an information system (as a product) provides to a user *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1)* **(3)** a measure of the delivered software functionality *(IEEE 1045-1992, (R2002) IEEE Standard for Software Productivity Metrics, 3.2)*

afferent. (1) pertaining to a flow of data or control from a subordinate module to a superordinate module in a software system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* efferent

assemble. (1) to translate a computer program expressed in an assembly language into its machine language equivalent *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** process of constructing from parts of one or more identified pieces of software *(ISO/IEC/IEEE 24765j:2021)* **(3)** activities for combining and connecting implemented system elements or aggregates to support specific goals, i.e. integration, verification, validation, manufacturing, and production *(ISO/IEC TS 24748-6:2016 Systems and software engineering--Life cycle management--Part 6: System integration engineering, 3.1.3) See also:* compile, disassemble, interpret

big-bang testing. (1) type of integration testing in which software elements, hardware elements, or both are combined all at once into an overall system, rather than in stages *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

call list. (1) ordered list of arguments used in a call to a software module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

case. (1) single-entry, single-exit multiple-way branch that defines a control expression, specifies the processing to be performed for each value of the control expression, and returns control in all instances to the statement immediately following the overall construct *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** Computer Aided Software Engineering *(ISO/IEC 14102:2008 Information Technology - Guideline for the evaluation and selection of CASE tools, 4) Syn:* multiple exclusive selective construct *See also:* go to, jump, if-then-else. multiple inclusive selective construct

catastrophic failure. (1) failure of critical software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

checkout. (1) testing conducted in the operational or support environment to ensure that a software product performs as required after installation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

chief programmer. (1) leader of a chief programmer team; a senior-level programmer whose responsibilities include producing key portions of the software assigned to the team, coordinating the activities of the team, reviewing the work of the other team members, and having an overall technical understanding of the software being developed *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* backup programmer, chief programmer team

chief programmer team. (1) software development group that consists of a chief programmer, a backup programmer, a secretary/librarian, and additional programmers and specialists as needed, and that employs procedures designed to enhance group communication and to make optimum use of each member's skills *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* backup programmer, chief programmer, egoless programming

code review. (1) meeting at which software code is presented to project personnel, managers, users, customers, or other interested parties for comment or approval *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* design review, formal qualification review, requirements review, test readiness review

cohesion. (1) manner and degree to which the tasks performed by a single software module are related to one another *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** in software design, a measure of the strength of association of the elements within a module *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the*

Software Engineering Body of Knowledge (SWEBOK)) Note: Types include coincidental, communicational, functional, logical, procedural, sequential, and temporal. *Syn:* module strength *See also:* coupling

coincidental cohesion. (1) type of cohesion in which the tasks performed by a software module have no functional relationship to one another *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* communicational cohesion, functional cohesion, logical cohesion, procedural cohesion, sequential cohesion, temporal cohesion

common storage. (1) portion of main storage that can be accessed by two or more modules in a software system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* common area, common block *See also:* global data

common-environment coupling. (1) type of coupling in which two software modules access a common data area *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* common coupling, common environment coupling *See also:* content coupling, control coupling, data coupling, hybrid coupling, pathological coupling

communicational cohesion. (1) type of cohesion in which the tasks performed by a software module use the same input data or contribute to producing the same output data *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coincidental cohesion, functional cohesion, logical cohesion, procedural cohesion, sequential cohesion, temporal cohesion

comparator. (1) software tool that compares two computer programs, files, or sets of data to identify commonalities or differences *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Typical objects of comparison are similar versions of source code, object code, database files, or test results.

computer program. (1) combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

(2) syntactic unit that conforms to the rules of a particular programming language and that is composed of declarations and statements or instructions needed for a certain function, task, or problem solution *(ISO/IEC 2382:2015 Information technology -- Vocabulary) See also:* software

computer software component (CSC). (1) functionally or logically distinct part of a computer software configuration item, typically an aggregate of two or more software units *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* computer software configuration item, software configuration item, software item

computer software configuration item (CSCI). (1) aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* software configuration item (SWCI) *See also:* computer software component, hardware configuration item, configuration item, software configuration item, software item

computer system. (1) system containing one or more computers and associated software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** system containing one or more components and elements such as computers (hardware), associated software, and data *(ISO/IEC 25024:2015 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of data, 4.3) Syn:* computing system *See also:* data processing system

computer-aided software engineering (CASE). (1) use of computers to aid in the software engineering process *(ISO/IEC 15940:2013 Systems and software engineering--Software Engineering Environment Services, 2.2) Syn:* computer aided software engineering *See also:* integrated development environment

configuration. (1) arrangement of a computer system or component as defined by the number, nature, and interconnections of its constituent parts *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** in configuration management, the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** arrangement of a system or network as defined by the nature, number, and chief characteristics of its functional units *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** requirements, design, and implementation that define a particular version of a system or system component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(5)** manner in which the hardware and software of an information processing system are organized and interconnected *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(6)** collection of objects able to interact at interfaces *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 10.2) See also:* configuration item; form, fit, and function; version

configuration item (CI). (1) item or aggregation of hardware, software, or both that is designated for configuration management and treated as a single entity in the configuration management process *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.15) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.11)* **(2)** component of an infrastructure or an item which is or will be under control of configuration management *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.7)* **(3)** aggregation of work products that is designated for configuration management and treated as a single entity in the configuration management process *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)* **(4)** any system element or aggregation of system elements that satisfies an end use function and is designated by the acquirer for separate configuration control *(IEEE 15288.1:2014 IEEE Standard for Application of Systems Engineering on Defense Programs, 3.1)* **(5)** item or aggregation of software that is designed to be managed as a single entity and its underlying components, such as documentation, data structures, scripts *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.12) Note:* Configuration items can vary widely in complexity, size and type, ranging from an entire system including all hardware, software and documentation, to a single module or a minor hardware component. CIs have four common characteristics: defined functionality; replaceable as an entity; unique specification; formal control of form, fit, and function. *See also:* hardware configuration item, computer software configuration item, configuration identification, critical item

consistency. (1) degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software attributes that provide uniform design and implementation techniques and notations *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* traceability

content coupling. (1) type of coupling in which some or all of the contents of one software module are included in the contents of another module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* common-environment coupling, control coupling, data coupling, hybrid coupling, pathological coupling

control coupling. (1) type of coupling in which one software module communicates information to another module for the explicit purpose of influencing the latter module's execution *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* common-environment coupling, content coupling, data coupling, hybrid coupling, pathological coupling

control data. (1) data that select an operating mode, direct the sequential flow of a program, or otherwise directly influence the operation of software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

conversion. (1) modification of existing software to enable it to operate with similar functional capability in a different environment *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

coupling. (1) manner and degree of interdependence between software modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** strength of the relationships between modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** measure of how closely connected two routines or modules are *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(4)** in software design, a measure of the interdependence among modules in a computer program *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK), 2.1.4) Note:* Types include common-environment coupling, content coupling, control coupling, data coupling, hybrid coupling, and pathological coupling. *See also:* cohesion

CPC. (1) computer program component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* computer software component

CPCI. (1) computer program configuration item *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* computer software configuration

cross-reference generator. (1) software tool that accepts as input the source code of a computer program and produces as output a listing that identifies each of the program's variables, labels, and other identifiers and indicates which statements in the program define, set, or use each one *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* cross-referencer

CSC. (1) computer software component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** cloud service customer *(ISO/IEC TS 25052-1:2022, Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE): cloud services--Part 1: Quality model, 3.3.3)*

CSCI. (1) computer software configuration item *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* SWCI

data coupling. (1) type of coupling in which output from one software module serves as input to another module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* input-output coupling *See also:* common-environment coupling, content coupling, control coupling, hybrid coupling, pathological coupling

data structure-centered design. (1) software design technique in which the architecture of a system is derived from analysis of the structure of the data sets with which the system must deal *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* input-process-output, modular decomposition, object-oriented design, rapid prototyping, stepwise refinement, structure clash, structured design, transaction analysis, transform analysis

demodularization. (1) in software design, the process of combining related software modules, usually to optimize system performance *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* downward compression, lateral compression, upward compression

design language. (1) specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document a hardware or software design *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** standardized notation, modeling technique, or other representation scheme and its usage conventions, shown to be effective in representing and communicating design information *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Types include hardware design language, program design language. *See also:* requirements specification language

design phase. (1) the period in the software life cycle during which definitions for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** the period in the software life cycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* detailed design, preliminary design

design review. (1) formal, documented, comprehensive, and systematic examination of a design to determine if the design meets the applicable requirements, to identify problems, and to propose solutions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** process or meeting during which a system, hardware, or software design is presented to project personnel, managers, users, customers, or other interested parties for comment or approval *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Types include critical design review, preliminary design review, system design review. *See also:* code review, formal qualification review, requirements review, test readiness review

development testing. (1) formal or informal testing conducted during the development of a system or component, usually in the development environment by the developer *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** testing conducted to establish whether a new software product or software-based system (or components of it) satisfies its criteria *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The criteria will vary based on the level of test being performed. *See also:* acceptance testing, operational testing, qualification testing

developmental configuration. (1) in configuration management, the software and associated technical documentation that define the evolving configuration of a computer software configuration item during development *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The developmental configuration is under the developer's control, and therefore is not called a baseline. *See also:* allocated baseline, functional baseline, product baseline

disassembler. (1) software tool that disassembles computer programs *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

diversity. (1) in fault tolerance, realization of the same function by different means *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software diversity

downward compatible. (1) pertaining to hardware or software that is compatible with an earlier or less complex version of itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* upward compatible

downward compression. (1) in software design, a form of demodularization in which a superordinate module is copied into the body of a subordinate module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

See also: lateral compression, upward compression

efferent. (1) pertaining to a flow of data or control from a superordinate module to a subordinate module in a software system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* afferent

egoless programming. (1) software development technique based on the concept of team, rather than individual, responsibility for program development *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Its purpose is to prevent individual programmers from identifying so closely with their work that objective evaluation is impaired.

embedded software. (1) software that is part of a larger system and performs some of the requirements of that system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

encapsulation. (1) software development technique that consists of isolating a system function or a set of data and operations on those data within a module and providing precise specifications for the module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** concept that access to the names, meanings, and values of the responsibilities of a class is entirely separated from access to their realization *(IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.54)* **(3)** idea that a module has an outside that is distinct from its inside, that it has an external interface and an internal implementation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data abstraction, information hiding

engineering change. (1) alteration in the configuration of a hardware/software configuration item or items, delivered, to be delivered, or under development, after formal establishment of their configuration identification *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** in configuration management, an alteration in the configuration of a configuration item or other designated item after formal establishment of its configuration identification *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* configuration control, engineering change proposal, deviation, waiver

entry point. (1) point in a software module at which execution of the module can begin *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** point in a test item at which execution of the test item can begin *(ISO/IEC/IEEE 29119-4:2021 Software and systems engineering -- Software testing -- Part 4: Test techniques, 3.27) Note:* An entry point is an executable statement within a test item that can be selected by an external process as the starting point for one or more paths through the test item. It is most commonly the first executable statement within the test item. *Syn:* entrance, entry *See also:* exit, reentry point

error model. (1) in software evaluation, a model used to estimate or predict the number of remaining faults, required test time, and similar characteristics of a system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* error prediction model

exit. (1) point in a software module at which execution of the module can terminate *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** data movement that moves a data group from a functional process across the boundary to the functional user that requires it *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.9) Note:* An exit is considered to account for certain associated data manipulations (e.g. formatting and routing associated with the data to be exited). *Syn:* exit type *See also:* entry point, return

flowcharter. (1) software tool that accepts as input a design or code representation of a program and produces as

output a flowchart of the program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*
formal parameter. (1) variable used in a software module to represent data or program elements that are to be passed to the module by a calling module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* argument (3)
generic program unit. (1) software module that is defined in a general manner and that requires substitution of specific data, instructions, or both, in order to be used in a computer program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* instantiation
hardware. (1) physical equipment used to process, store, or transmit computer programs or data *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.21)* **(2)** all or part of the physical components of an information system *(ISO/IEC 2382:2015 Information technology -- Vocabulary) Syn:* HW *See also:* software
hybrid coupling. (1) type of coupling in which different subsets of the range of values that a data item can assume are used for different and unrelated purposes in different software modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* common-environment coupling, content coupling, control coupling, data coupling, pathological coupling
implementation phase. (1) period of time in the software life cycle during which a software product is created from design documentation and debugged *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*
incremental development. (1) software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* waterfall model, data structure-centered design, input-process-output, modular decomposition, object-oriented design
information hiding. (1) software development technique in which each module's interfaces reveal as little as possible about the module's inner workings and other modules are prevented from using information about the module that is not in the module's interface specification *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** containment of a design or implementation decision in a single module so that the decision is hidden from other modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* encapsulation
input-process-output. (1) software design technique that consists of identifying the steps involved in each process to be performed and identifying the inputs to and outputs from each step *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* A refinement called hierarchical input-process-output identifies the steps, inputs, and outputs at both general and detailed levels of detail *See also:* data structure-centered design, input-process-output chart, modular decomposition, object-oriented design, rapid prototyping
input-process-output (IPO) chart. (1) diagram of a software system or module, consisting of a rectangle on the left listing inputs, a rectangle in the center listing processing steps, a rectangle on the right listing outputs, and arrows connecting inputs to processing steps and processing steps to outputs *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* summarizes the process activities and their inputs and outputs from/to external actors; some inputs are categorized as controls and enablers. *Syn:* IPO diagram *See also:* block diagram, box diagram, bubble

chart, flowchart, graph, structure chart

installation and checkout phase. (1) period of time in the software life cycle during which a software product is integrated into its operational environment and tested in this environment to ensure that it performs as required *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

instrument. (1) in software and system testing, to install or insert devices or instructions into hardware or software to monitor the operation of a system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

instrumentation. (1) devices or instructions installed or inserted into hardware or software to monitor the operation of a system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

integration. (1) process of combining software components, hardware components, or both into an overall system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** activities of combining several implemented system elements and activating the interfaces to form a realized system (product or service) that enables interoperation between the system elements and with other systems to satisfy system requirements, architecture characteristics and design properties *(ISO/IEC TS 24748-6:2016 Systems and software engineering--Life cycle management--Part 6: System integration engineering, 3.1.5) Note:* Integration can apply to the implemented system elements which compose a system and the necessary life-cycle related activities, and also to connect a system-of-interest with external interoperating systems or enabling systems.

main program. (1) software component that is called by the operating system of a computer and that usually calls other software components *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* routine, subprogram

manufacture. (1) in software engineering, the process of copying software to disks, chips, or other devices for distribution to customers or users *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

manufacturing phase. (1) period of time in the software life cycle during which the basic version of a software product is adapted to a specified set of operational environments and is distributed to a customer base *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

map program. (1) software tool, often part of a compiler or assembler, that generates a load map *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

modular programming. (1) software development technique in which software is developed as a collection of modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data structure-centered design, input-process-output, modular decomposition, object-oriented design, rapid prototyping, stepwise refinement, structured design, transaction analysis, transform analysis

modularity. (1) degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.7.1)* **(2)** software attributes that provide a structure of highly independent components *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* cohesion, coupling, modifiability

monitor. (1) software tool or hardware device that operates concurrently with a system or component and supervises, records, analyzes, or verifies the operation of the system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** collect project performance data with respect to a plan, produce performance measures, and report and disseminate performance information. *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition) Syn:* execution monitor *See also:* hardware monitor, software monitor

object-oriented design. (1) software development technique in which a system or component is expressed in terms of objects and connections between those objects *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data structure-centered design, input-process-output, modular decomposition, rapid prototyping, stepwise refinement, structured design, transaction analysis, transform analysis

operating system. (1) collection of software, firmware, and hardware elements that controls the execution of computer programs and provides such services as computer resource allocation, job control, input/output control, and file management in a computer system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* OS

operation and maintenance phase. (1) period of time 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 to respond to changing requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

order clash. (1) in software design, a type of structure clash in which a program must deal with two or more data sets that have been sorted in different orders *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data structure-centered design

parser. (1) software tool that parses computer programs or other text, often as the first step of assembly, compilation, interpretation, or analysis *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

pathological coupling. (1) type of coupling in which one software module affects or depends upon the internal implementation of another *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* common-environment coupling, content coupling, control coupling, data coupling, hybrid coupling

pipeline. (1) software or hardware design technique in which the output of one process serves as input to a second, the output of the second process serves as input to a third, and so on, often with simultaneity within a single cycle time *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1)*

portability. (1) ease with which a system or component can be transferred from one hardware or software environment to another *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** capability of a program to be executed on various types of data processing systems without converting the program to a different language and with little or no modification *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(3)** degree of effectiveness and efficiency with which a system, product, or component can be transferred from one hardware, software or other operational or usage environment to another *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.8)* **(4)** property that the reference points of an object allow it to be adapted to a variety of configurations *(ISO/IEC 10746-2:2009 Information technology -- Open Distributed Processing -- Reference Model: Foundations, 15.4.1)* **(5)** degree to which a cloud service provides the ability to move data and migrate applications from one cloud service to another *(ISO/IEC TS*

25052-1:2022, Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE): cloud services--Part 1: Quality model, 3.1.7) Syn: transportability *See also:* machine-independent

preventive maintenance. (1) modification of a software product after delivery to correct latent faults in the software product before they occur in the live system *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.10)*

procedural cohesion. (1) type of cohesion in which the tasks performed by a software module all contribute to a given program procedure, such as an iteration or decision process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coincidental cohesion, communicational cohesion, functional cohesion, logical cohesion, sequential cohesion, temporal cohesion

product standard. (1) standard that defines what constitutes completeness and acceptability of items that are used or produced, formally or informally, during the software engineering process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

product support. (1) providing of information, assistance, and training to install and make software operational in its intended environment and to distribute improved capabilities to users *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

production library. (1) software library containing software approved for current operational use *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software development library, software repository, system library

program synthesis. (1) use of software tools to aid in the transformation of a program specification into a program that realizes that specification *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

programmer manual. (1) document that provides the information necessary to develop or modify software for a given computer system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Typically described are the equipment configuration, operational characteristics, programming features, input/output features, and compilation or assembly features of the computer system. *See also:* diagnostic manual, installation manual, operator manual, support manual, user manual

programming support environment. (1) integrated collection of software tools accessed via a single command language to provide programming support capabilities throughout the software life cycle *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* sometimes called integrated programming support environment. The environment typically includes tools for specifying, designing, editing, compiling, loading, testing, configuration management, and project management. *See also:* scaffolding

programming system. (1) set of programming languages and the support software (editors, compilers, linkers, etc.) necessary for using these languages with a given computer system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

prototyping. (1) hardware and software development technique in which a preliminary version of part or all of the hardware or software is developed to permit user feedback, determine feasibility, or investigate timing or other issues in support of the development process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* rapid prototyping

reentrant. (1) pertaining to a software module that can be entered as part of one process while also in execution as part of another process and still achieve the desired results *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* reenterable, re-entrant

reentry point. (1) place in a software module at which the module is reentered following a call to another module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* re-entry point

requirements phase. (1) period of time in the software life cycle during which the requirements for a software product are defined and documented *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

requirements review. (1) process or meeting during which the requirements for a system, hardware item, or software item are presented to project personnel, managers, users, customers, or other interested parties for comment or approval *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Types include system requirements review, software requirements review. *See also:* code review, design review, formal qualification review, test readiness review

requirements specification language. (1) specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document hardware or software requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* design language

retirement phase. (1) period of time in the software life cycle during which support for a software product is terminated *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software life cycle, system life cycle

reusable. (1) pertaining to a software module or other work product that can be used in more than one computer program or software system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

scaffolding. (1) computer programs and data files built to support software development and testing, but not intended to be included in the final product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* programming support environment

SDP. (1) software development plan *(ISO/IEC/IEEE 16326:2019 Systems and software engineering -- Life cycle processes -- Project management, 3)*

self-descriptiveness. (1) the degree to which a system or component contains enough information to explain its objectives and properties *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software attributes that explain a function's implementation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* maintainability, testability, usability

sequential cohesion. (1) type of cohesion in which the output of one task performed by a software module serves as input to another task performed by the module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coincidental cohesion, communicational cohesion, functional cohesion, logical cohesion, procedural cohesion, temporal cohesion

simplicity. (1) degree to which a system or component has a design and implementation that is straightforward and easy to understand *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software attributes that provide implementation of functions in the most understandable manner *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* complexity

simultaneous recursion. (1) situation in which two software modules call each other *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

sizing. (1) process of estimating the amount of computer storage or the number of source lines required for a software system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* timing

software development cycle. (1) period of time that begins with the decision to develop a software product and ends when the software is delivered *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* This cycle typically includes a requirements phase, design phase, implementation phase, test phase, and sometimes, installation and checkout phase. The phases listed above can overlap or be performed iteratively, depending upon the software development approach used. This term is sometimes used to mean a longer period of time, either the period that ends when the software is no longer being enhanced by the developer, or the entire software life cycle. *See also:* software life cycle

software development file (SDF). (1) collection of material pertinent to the development of a given software unit or set of related units *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Contents typically include the requirements, design, technical reports, code listings, test plans, test results, problem reports, schedules, and notes for the units. *Syn:* software development folder, software development notebook, unit development folder

software development library. (1) software library containing computer readable and human readable information relevant to a software development effort *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* project library, program support library *See also:* production library, software repository, system library

software development plan (SDP). (1) information item that describes the technical approach to be followed for a software development effort *(ISO/IEC/IEEE 24748-5:2017 Systems and software engineering--Life cycle management--Part 5: Software development planning, 3.15) Note:* The software development plan presents how the organization or project plans to conduct development activities. A distinction is being made between the technical and management approaches. *See also:* project management plan

software engineering environment (SEE). (1) environment that provides automated system context services and software-specific services for the engineering of software systems and related domains, such as project management and process management *(ISO/IEC 15940:2013 Systems and software engineering--Software Engineering Environment Services, 2.7)* **(2)** hardware, software, and firmware used to perform a software engineering effort *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2) Note:* It includes the platform, system software, utilities, and CASE tools installed. *Syn:* infrastructure

software library. (1) controlled collection of software and related documentation designed to aid in software development, use, or maintenance *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* program library *See also:* repository

software monitor. (1) software tool that executes concurrently with another program and provides detailed information about the execution of the other program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* hardware monitor, monitor

software repository. (1) software library providing permanent, archival storage for software and related documentation *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)*

See also: production library, software development library, system library

software tool. (1) computer program used in the development, testing, analysis, or maintenance of a program or its documentation *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

spiral model. (1) model of the software development process in which the constituent activities, typically requirements analysis, preliminary and detailed design, coding, integration, and testing, are performed iteratively until the software is complete *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* waterfall model, incremental development, rapid prototyping

SRS. (1) software requirements specification *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.2)* **(2)** system requirements specification *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary) See also:* SyRS

SSR. (1) software specification review *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.2)* **(2)** secure storage requirement *(IEEE 7005 2021, IEEE Standard for Transparent Employer Data Governance, 3.2) See also:* software requirements review

stand-alone. (1) pertaining to hardware or software that is capable of performing its function without being connected to other components *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

stepwise refinement. (1) software development technique in which data and processing steps are defined broadly at first and then further defined with increasing detail *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data structure-centered design, input-process-output, modular decomposition, object-oriented design, rapid prototyping, structured design, transaction analysis, transform analysis

structure clash. (1) in software design, a situation in which a module must deal with two or more data sets that have incompatible data structures *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* data structure-centered design, order clash

structured programming. (1) software development technique that includes structured design and results in the development of structured programs *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

support. (1) set of activities necessary to ensure that an operational system or component fulfills its original requirements and any subsequent modifications to those requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software life cycle, system life cycle

support software. (1) software that aids in the development or maintenance of other software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software or a program that aids in the development, maintenance, or use of other software or provides general application-independent capability *(ISO/IEC 2382:2015 Information technology -- Vocabulary) See also:* application software, system software

symbolic execution. (1) software analysis technique in which program execution is simulated using symbols, such as variable names, rather than actual values for input data, and program outputs are expressed as logical or mathematical expressions involving these symbols *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

system development cycle. (1) period of time that begins with the decision to develop a system and ends when the system is delivered to its end user *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* This term is sometimes used to mean a longer period of time, either the period that ends when the system is no longer being enhanced, or the entire system life cycle. *See also:* system life cycle software development cycle

system library. (1) software library containing system-resident software that can be accessed for use or incorporated into other programs by reference *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* production library, software development library, software repository

system requirements review (SRR). (1) review conducted to evaluate the completeness and adequacy of the requirements defined for a system; to evaluate the system engineering process that produced those requirements; to assess the results of system engineering studies; and to evaluate system engineering plans *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software requirements review

system software. (1) software designed to facilitate the operation and maintenance of a computer system and its associated programs *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** application-independent software that supports the running of application software *(ISO/IEC 2382:2015 Information technology -- Vocabulary) See also:* application software, support software

N 2 diagram. (1) system engineering or software engineering tool for tabulating, defining, analyzing, and describing functional interfaces and interactions among system components *(ISO/IEC/IEEE 24765e:2015) Note:* The N 2 diagram is a matrix structure that graphically displays the bidirectional interrelationships between functions and components in a given system or structure. *Syn:* N2 diagram

software-intensive system. (1) system for which software is a major technical challenge and is perhaps the major factor that affects system schedule, cost, and risk *(IEEE 1062-2015 IEEE Recommended Practice for Software Acquisition, 3.1) Note:* In the most general case, a software-intensive system is comprised of hardware, software, people, and manual procedures. *See also:* software system

software system. (1) system for which software is of primary importance to the stakeholders *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.55) See also:* software-intensive system

construction. (1) process of writing, assembling, or generating assets *(ISO/IEC/IEEE 24765j:2021)* **(2)** activity in software development consisting of detailed design, coding, unit testing, and debugging *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* the collection of activities focused on creating source code

domain architecture. (1) generic, organizational structure or design for software systems in a domain *(IEEE 1517-2010 IEEE Standard for Information Technology--System and software life cycle processes--Reuse processes, 3)* **(2)** common architecture for a product line that can embrace variability of member products *(ISO/IEC 26552:2019 Software and systems engineering--Tools and methods for product line architecture design, 3.5)* **(3)** core architecture that captures the high-level design of a software and systems product line including the architectural structure and texture (e.g. common rules and constraints) that constrains all member products within a software and systems product line *(ISO/IEC 26550:2015 Software and systems engineering--Reference model for product line engineering and management, 3.10) Note:* The domain architecture contains the designs that are intended to satisfy requirements specified in the domain model. The domain architecture documents design, whereas the domain model documents requirements. A domain architecture: 1) can be adapted to create designs for software systems within a domain, and 2) provides a framework for configuring assets within individual software systems. The term "architecture" has been deliberately redefined to more properly convey its meaning in the software reuse context.

product line. (1) set of products or services sharing explicitly defined and managed common and variable features and relying on the same domain architecture to meet the common and variable needs of specific markets *(ISO/IEC 26550:2015 Software and systems engineering--Reference model for product line engineering and management, 3.16)* **(2)** paradigm for the creation, exploitation, and management of a common platform for a family of products *(ISO/IEC 26561:2019 Software systems engineering--Methods and tools for product line technical probe, 3.9)* **(3)** family of similar products with variations in features *(ISO/IEC 26580:2021, Software and systems engineering Methods and tools for the feature-based approach to software and systems product line engineering, 3.16)* **(4)** group of products or services sharing a common, managed set of features that satisfy specific needs of a selected market or mission *(INCOSE Systems Engineering Handbook, 5th ed.) Note:* Typical goals of product lines are to lower costs, reduce time to market, and improve quality. *Syn:* product family, software and systems product line. SSPL

reuse. (1) use of an asset in the solution of different problems *(IEEE 1517-2010 IEEE Standard for Information Technology--System and software life cycle processes--Reuse processes, 3) (ISO/IEC/IEEE 24765j:2021)* **(2)** building a software system at least partly from existing pieces to perform a new application *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software project life cycle (SPLC). (1) portion of the entire software life cycle applicable to a specific project *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary) Note:* It is the sequence of activities created by mapping the activities onto a selected software project life cycle model (SPLCM).

software project life cycle model (SPLCM). (1) framework selected by each using organization on which to map the activities of IEEE Std 1074 to produce the software project life cycle (SPLC) *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

software project life cycle process (SPLCP). (1) project-specific description of the process developed by adding the organizational process assets (OPAs) to the software project life cycle (SPLC) and the OPAs *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

SPLC. (1) software project life cycle *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

SPLCM. (1) software project life cycle model *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

SPLCP. (1) software project life cycle process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

categorization scheme. (1) orderly combination of views and categories related to software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

OSF. (1) Open Software Foundation *(ISO/IEC 10746-1:1998 Information technology -- Open Distributed Processing -- Reference model: Overview)*

evaluation module. (1) package of evaluation technology for measuring software quality characteristics, subcharacteristics, or attributes *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.9) (ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.20) Note:*

The package includes evaluation methods and techniques, input to be evaluated, data to be measured and collected, and supporting procedures and tools.

boundary. (1) conceptual interface between the software under study and its users *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.9) (ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, 3.2)* **(2)** conceptual interface between the software being measured and its functional users *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.3) Note:* The boundary provides the measurement analyst(s) with a solid delimiter to distinguish, without ambiguity, what is included inside the measured software from what is part of the measured software's operating environment. The boundary defines what is external to the application; it indicates the border between the software being measured and the user; it acts as a ?membrane? through which data processed by transactions pass into and out of the application; it is dependent on the user?s external business view of the application; it is independent of non-functional and/or implementation considerations. The boundary is identical when assessing the functional size and the non-functional size. *Syn:* application boundary, software boundary

functional domain. (1) class of software based on the characteristics of functional user requirements which are pertinent to FSM *(ISO/IEC 14143-1:2007 Information technology--Software measurement--Functional size measurement; Part 1: Definition of concepts, 3.5)* **(2)** categorized functions that are generally used together *(ISO/IEC 26551:2016 Software and systems engineering --Tools and methods for product line requirements engineering, 3.16)*

functional size. (1) size of the software derived by quantifying the functional user requirements *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.33) (ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis, B) (ISO/IEC 14143-1:2007 Information technology--Software measurement--Functional size measurement; Part 1: Definition of concepts, 3.6) Syn:* FS, number of function points

functional user requirements (FUR). (1) subset of the user requirements specifying what the software shall do in terms of tasks and services *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.34)* **(2)** a subset of the user requirements describing what the software does in terms of tasks and services *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.14) (ISO/IEC 14143-1:2007 Information technology--Software measurement--Functional size measurement; Part 1: Definition of concepts, 3.8) Note:* Functional User Requirements include but are not limited to: data transfer (for example Input customer data, Send control signal); data transformation (for example Calculate bank interest, Derive average temperature); data storage (for example Store customer order, Record ambient temperature over time); data retrieval (for example List current employees, Retrieve aircraft position). User Requirements that are not Functional User Requirements include but are not limited to: quality constraints (for example usability, reliability, efficiency and portability); organizational constraints (for example locations for operation, target hardware and compliance to standards); environmental constraints (for example interoperability, security, privacy and safety); implementation constraints (for example development language, delivery schedule).

technical requirements. (1) requirements relating to the technology and environment, for the development, maintenance, support and execution of the software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

SQA. (1) Software Quality Assurance *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.3)*

SLCP. (1) Software Life Cycle Processes *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

existing software. (1) software that is already developed and available; is usable either &quot;as is&quot; or with modifications; and which is provided by the supplier, acquirer, or a third party *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

evaluator. (1) individual or organization that performs an evaluation *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.18)*

(2) competent person engaged in the verification or validation of a system or software *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.7)*

computer-based software system (CBSS). (1) software system running on a computer *(ISO/IEC 14756:1999 Information technology -- Measurement and rating of performance of computer-based software systems, 4.5) Note:* A CBSS can be a data processing system as seen by human users at their terminals or at equivalent machine-user-interfaces. It includes hardware and all software (system software and application software) which is necessary for realizing data processing functions required by its users

maintainability plan. (1) document setting out the specific maintainability practices, resources and sequence of activities relevant to software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The developer prepares the Maintainability Plan.

maintenance plan. (1) document setting out the specific maintenance practices, resources, and sequence of activities relevant to maintaining a software product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software test environment (STE). (1) facilities, hardware, software, firmware, procedures, and documentation needed to perform qualification or other testing of software *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.15) Note:* Elements include simulators, code analyzers, test case generators, path analyzers, and elements used in the software engineering environment

modeling tool. (1) tool that provides support for modeling, i.e., representing, a software product or an information system *(ISO/IEC 15474-1:2002 Information technology -- CDIF framework -- Part 1: Overview, 4.2)*

tool. (1) software product that provides support for software and system life cycle processes *(ISO/IEC 15474-1:2002 Information technology -- CDIF framework -- Part 1: Overview, 4.2)* **(2)** something tangible, such as a template or software program, used in performing an activity to produce a product or result. *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)*

cut-off date. (1) date after which changes to the software are reflected in the next, rather than the current, software release or issue of the documentation *(ISO/IEC/IEEE 24765a:2011)*

navigation. (1) means by which a user moves from one part of a software application to another *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** act of accessing documentation and viewing different topics *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.35)* **(3)** process of accessing on-screen documentation and moving between different items of information *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.27)* **(4)** process of accessing on-screen information by moving between different locations in a website or electronic document *(ISO/IEC/IEEE 23026:2015 Systems and software engineering--Engineering and management of websites for systems, software, and services information, 4.19)*

online documentation. (1) information accessed by the user through the use of software *(ISO/IEC/IEEE 24765a:2011) Note:* can be context-sensitive

production. (1) stage in the life cycle when completed software or documentation is prepared, published, and packaged for distribution *(ISO/IEC/IEEE 24765a:2011)*

software configuration management (SCM). (1) process of applying configuration management throughout the software life cycle to ensure the completeness and correctness of software configuration items (SCI) *(ISO/IEC/IEEE 24765i:2020)*

SCI. (1) software configuration item *(ISO/IEC/IEEE 24765i:2020)* **(2)** serial communication interface *(ISO/IEC/IEEE 24765d:2015)*

SCM. (1) software configuration management *(ISO/IEC/IEEE 24765i:2020)*

SEE. (1) Software Engineering Environment *(ISO/IEC/IEEE 16326:2019 Systems and software engineering -- Life cycle processes -- Project management, 5)*

candidate FSM method. (1) documented software size measurement method submitted for conformity evaluation *(ISO/IEC 14143-2:2011 Information technology -- Software measurement -- Functional size measurement -- Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1, 3.1)*

user requirements (UR). (1) requirements for use that provide the basis for design and evaluation of interactive systems to meet identified user needs *(ISO/IEC TR 25060:2010 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: General framework for usability-related information, 2.21)* **(2)** description of the set of user needs for the software *(ISO/IEC 14143-1:2007 Information technology--Software measurement--Functional size measurement; Part 1: Definition of concepts, 3.12)* **(3)** expression of perceived need from individual or group that benefits from a system during its utilization *(ISO/IEC TR 24766:2009 Information technology--Systems and software engineering--Guide for requirements engineering tool capabilities, 3.10) Note:* User requirements specify the extent to which user needs and capabilities are to be met when using the system. They are not requirements on the users. User requirements are derived from user needs and capabilities in order to make use of the system in an effective, efficient, safe and satisfying manner. User requirements comprise two subsets: functional user requirements and non-functional user requirements.
[ISO 25063:2014] In software-engineering terms, user requirements comprise both "functional" and "non-functional" requirements based on user needs and capabilities. *Syn:* usage requirements

server. (1) hardware system or software program which provides a service to clients *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** process implementing one or more operations on one or more objects *(ISO/IEC 19500-2:2012 Information technology --Object Management Group--Common Object Request Broker*

Architecture (CORBA)--Part 2: Interoperability, 3.2.14)

software testing. (1) activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)* **(2)** dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite executions domain *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK), 4)*

exploratory testing. (1) type of unscripted experience-based testing in which the tester spontaneously designs and executes tests based on the tester's existing relevant knowledge, prior exploration of the test item (including the results of previous tests), and heuristic &quot;rules of thumb&quot; regarding common software behaviors and types of failure *(ISO/IEC/IEEE 29119-2:2021, Software and systems engineering--Software testing--Part 2: Test processes, 4.9) Note:* Exploratory testing hunts for hidden properties (including hidden behaviors) that, while quite possibly benign by themselves, could interfere with other properties of the software under test, and so constitute a risk that the software will fail.

reengineering. (1) examination and alteration of software to reconstitute it in a new form, including the subsequent implementation of the new form *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK), 5.4.2)* **(2)** complete cycle of performing reverse engineering followed by forward engineering *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

management process. (1) activities that are undertaken in order to ensure that the software engineering processes are performed in a manner consistent with the organization's policies, goals, and standards *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

generally accepted. (1) knowledge to be included in the study material of a software engineering licensing exam that a graduate would pass after completing four years of work experience *(ISO/IEC TR 19759:2016 Software Engineering -- Guide to the Software Engineering Body of Knowledge (SWEBOK))*

COSMIC. (1) Common Software Measurement International Consortium *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method)*

data attribute. (1) smallest parcel of information, within an identified data group, carrying a meaning from the perspective of the software's functional user requirements *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 3.4)*

layer. (1) partition resulting from the functional division of a software system, where layers are organized in a hierarchy; there is only one layer at each level in the hierarchy; there is a superior/subordinate hierarchical dependency between the functional services provided by software in any two layers in the software system that exchange data directly; and the software in any two layers in the software system that exchange data interpret only part of that data identically *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.15)*

object of interest (-type). (1) any thing that is identified from the point of view of the functional user requirements about which the software is required to process or store data *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.19) Note:* An object of interest can be any physical thing, as well as any conceptual object or part of a conceptual object in the world of the functional user. *Syn:* object of interest type

operating environment (software). (1) set of software operating concurrently on a specified computer system *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.20) Syn:* operating environment software

triggering event. (1) event (something that happens) that causes a functional user of the piece of software to initiate (trigger) one or more functional processes *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.25) Syn:* triggering event type

application. (1) system for collecting, saving, processing, and presenting data by means of a computer *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis) (ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.1)* **(2)** coherent collection of automated procedures and data supporting a business objective *(ISO/IEC 20968:2002 Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, 10)* **(3)** cohesive collection of automated procedures and data supporting a business objective, consisting of one or more components, modules, or subsystems *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.2) Note:* The term application is generally used when referring to a component of software that can be executed. It consists of one or more components, modules, or subsystems. *Syn:* application system, automated information system *See also:* application software, information system

application boundary. (1) conceptual interface between the application and its users or other applications *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis) Note:* The boundary defines what is external to the application; it indicates the border between the software being measured and the user; it acts as a "membrane" through which data processed by transactions pass into and out of the application. Software boundary is dependent on the user's external business view of the application. *Syn:* software boundary

development project function point count (DFP). (1) a count that measures the functionality provided to the end users with the first installation of the software, developed when the project is complete *(ISO/IEC 20968:2002 Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, 10)* **(2)** activity of applying ISO/IEC 20926:2009 to measure the functional size of a development project *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.20)*

enhancement. (1) activities performed for an application that change the specifications of the application *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis)* **(2)** software change that addresses and implements a new requirement *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.7) Note:* The term "enhancement" is mainly used as a maintenance type and to classify modification requests (MR). There are three types of software enhancements: adaptive, perfective. and additive. An enhancement is not a software correction. These activities usually also change the functional size as a result (e.g. change requests). *See also:* change, correction

function point analysis (FPA). (1) a method used to acquire a measurement of the amount of functionality an application provides a user *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method --*

Definitions and counting guidelines for the application of function point analysis) **(2)** a form of functional size measurement (FSM) that measures the functional size of software development, enhancement and maintenance activities associated with business applications, from the customer's point of view *(ISO/IEC 20968:2002 Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, 10)* **(3)** method for measuring functional size as defined in ISO/IEC 29026:2009 *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.36)*

decoupling. (1) process of making software modules more independent of one another to decrease the impact of changes to, and errors in, the individual modules *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coupling

decompiler. (1) software tool that decompiles computer programs *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

delivery. (1) release of a system or component to its customer or intended user *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software life cycle, system life cycle

functional cohesion. (1) type of cohesion in which the tasks performed by a software module all contribute to the performance of a single function *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coincidental cohesion, communicational cohesion, logical cohesion, procedural cohesion, sequential cohesion, temporal cohesion

lateral compression. (1) in software design, a form of demodularization in which two or more modules that execute one after the other are combined into a single module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* downward compression, upward compression

logical cohesion. (1) type of cohesion in which the tasks performed by a software module perform logically similar functions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coincidental cohesion, communicational cohesion, functional cohesion, procedural cohesion, sequential cohesion, temporal cohesion

machine-dependent. (1) pertaining to software that relies on features unique to a particular type of computer and therefore executes only on computers of that type *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* machine-independent

machine-independent. (1) pertaining to software that does not rely on features unique to a particular type of computer, and therefore executes on computers of more than one type *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* machine-dependent, portability

macro. (1) in software engineering, a predefined sequence of computer instructions that is inserted into a program, usually during assembly or compilation, at each place that its corresponding macroinstruction appears in the program *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* macro definition *See also:* macroinstruction, macrogenerator, open subroutine

package. (1) separately compilable software component consisting of related data types, data objects, and subprograms *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** to combine related components into a single, deployable item *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1)* **(3)** namespace for the grouped elements *(ISO/IEC*

30130:2016 Software engineering --Capabilities of software testing tools) See also: data abstraction, encapsulation, information hiding

packaging. (1) in software development, the assignment of modules to segments to be handled as distinct physical units for execution by a computer *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

temporal cohesion. (1) type of cohesion in which the tasks performed by a software module are all required at a particular phase of program execution *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* coincidental cohesion, communicational cohesion, functional cohesion, logical cohesion, procedural cohesion, sequential cohesion

test bed. (1) environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

test case generator. (1) software tool that accepts as input source code, test criteria, specifications, or data structure definitions; uses these inputs to generate test input data; and, sometimes, determines expected results *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* test data generator, test generator

test driver. (1) software module used to invoke a module under test and, often, provide test inputs, control and monitor execution, and report test results *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* test harness

timing. (1) process of estimating or measuring the amount of execution time required for a software system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* sizing

timing analyzer. (1) software tool that estimates or measures the execution time of a computer program or portion of a computer program, either by summing the execution times of the instructions along specified paths or by inserting probes at specified points in the program and measuring the execution time between probes *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

transaction. (1) in software engineering, a data element, control element, signal, event, or change of state that causes, triggers, or initiates an action or sequence of actions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** activity which leads to a set of object changes consistent with a dynamic schema (and its constraining invariant schema) *(ISO/IEC 10746-3:2009 Information technology -- Open Distributed Processing -- Reference Model: Architecture, 13.7.1.1)*

transaction analysis. (1) software development technique in which the structure of a system is derived from analyzing the transactions that the system is required to process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* transaction-centered design *See also:* data structure-centered design, input-process-output, modular decomposition, object-oriented design, rapid prototyping, stepwise refinement, structured design, transform analysis

transform analysis. (1) software development technique in which the structure of a system is derived from analyzing the flow of data through the system and the transformations that must be performed on the data *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* transformation analysis, transform-centered design *See also:* data structure-centered design, input-process-output, modular decomposition, object-oriented design, rapid prototyping, stepwise refinement, structured design, transaction analysis

turnkey. (1) pertaining to a hardware or software system delivered in a complete, operational state *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*
UDF. (1) unit development folder *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software development file
upward compatible. (1) pertaining to hardware or software that is compatible with a later or more complex version of itself *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* downward compatible
upward compression. (1) in software design, a form of demodularization in which a subordinate module is copied inline into the body of a superordinate module *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* lateral compression, downward compression
user documentation. (1) information to describe, explain, or instruct how to use a system *(ISO/IEC/IEEE 24765k:2022)* **(2)** information that is supplied with the software to help the users in their use of that software *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.26) See also:* user manual
user interface. (1) components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system *(ISO/IEC TR 25060:2010 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: General framework for usability-related information, 2.23)* **(2)** ensemble of software and hardware that allows a user to interact with a computer system *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.55) (ISO/IEC/IEEE 26512:2018 Systems and software engineering--Requirements for acquirers and suppliers of information for users, 3.29)* **(3)** interface that enables information to be passed between a human user and hardware or software components of a computer system *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)* **(4)** all components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system *(ISO/IEC 25063:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) Common Industry Format (CIF) for usability: Context of use description)*
utility. (1) software tool designed to perform some frequently used support function *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** measure of value within a given value system, often measured on a scale of 0 to 100 *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*
variant. (1) fault tolerance, a version of a program resulting from the application of software diversity *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** one alternative that is used to realize particular variation points *(ISO/IEC 26557:2016 Software and systems engineering -- Methods and tools for variability mechanisms in software and systems product line, 3.17)* **(3)** instance or a value of a variation point *(ISO/IEC 26558:2017 Software and systems engineering -- Methods and tools for variability modelling in software and systems product line, 3.11)* **(4)** alternative that can be used to realize a particular variation point *(ISO/IEC 26580:2021, Software and systems engineering Methods and tools for the feature-based approach to software and systems product line engineering, 3.19)* *Note:* One or more variants correspond to each variation point to represent V_VP relationship. Selection and binding of

variants for a specific product determine the characteristics of the particular variability for the product.

waterfall model. (1) model of the software development process in which the constituent activities, typically a concept phase, requirements phase, design phase, implementation phase, test phase, and installation and checkout phase, are performed in that order, possibly with overlap but with little or no iteration *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* incremental development, rapid prototyping, spiral model

information processing requirements. (1) the set of functions required by the commissioning user of the application software product (excluding any technical and quality requirements) *(ISO/IEC 20968:2002 Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, 1.1) See also:* software

navigational aids. (1) features of software that help the user to navigate around a computer application *(ISO/IEC 20968:2002 Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, 10)*

context of use. (1) users, tasks, equipment (hardware, software and materials), and the physical and social environments
in which a system, product or service is used *(ISO/IEC 25063:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) Common Industry Format (CIF) for usability: Context of use description, 3.2)* **(2)** users, tasks, equipment (hardware, software, and materials), and the physical and social environments in which a product is used *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.2)* **(3)** conditions and constraints under which ICT products are used by specific users in a specific environment to achieve specific goals as part of the larger information system *(ISO/IEC 25030:2019 Systems and software engineering--Systems and software quality requirements and evaluation (SQuaRE)--Quality requirements framework, 3.2)* **(4)** intended operational environment for a system *(IEEE 7000:2021, IEEE Standard Model Process for Addressing Ethical Concerns during System Design, 3.1) Note:* Context of use includes direct use or use supported by assistive technologies. Environment includes physical aspects such as equipment and resources as well as social aspects such as demographics and culture.

custom software. (1) software product developed for a specific application from a user requirements specification *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.3)*

intermediate software product needs. (1) needs that can be specified as quality requirements by internal measures *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

measurement process. (1) process for establishing, planning, performing and evaluating software measurement within an overall project or organizational measurement structure *(ISO/IEC/IEEE 15939:2017 Systems and software engineering--Measurement process, 3.23)* **(2)** process of establishing, planning, performing and evaluating software measurement within an overall project or organizational measurement structure *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.18)* **(3)** process for establishing, planning, performing and evaluating systems and software measurement within an overall project or organizational measurement structure *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.21)*

software product evaluation. (1) technical operation that consists of producing an assessment of one or more characteristics of a software product according to a specified procedure *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.32)*

software quality. (1) capability of a software product to satisfy stated and implied needs when used under specified conditions *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.33)* **(2)** degree to which a software product satisfies stated and implied needs when used under specified conditions *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.3.13)* **(3)** degree to which a software product meets established requirements *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2) Note:* Quality depends upon the degree to which the established requirements accurately represent stakeholder needs, wants, and expectations. In SQuaRE standards software quality has the same meaning as software product quality.

software quality characteristic. (1) category of software quality attributes that bears on software quality *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.34) Note:* Software quality characteristics can be refined into multiple levels of sub-characteristics and finally into software quality attributes.

software quality evaluation. (1) systematic examination of the extent to which a software product is capable of satisfying stated and implied needs *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.35)*

target of process. (1) system, software product or task executed by system or software product to which measurement or evaluation process is applied *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.39)*

replication. (1) copying a software product from one medium to another *(ISO/IEC/IEEE 90003:2018 Software engineering -- Guidelines for the application of ISO 9001:2015 to computer software, 3.13)*

development project. (1) a project in which a completely new application is realized *(ISO/IEC 24570:2018 Software engineering -- NESMA functional size measurement method -- Definitions and counting guidelines for the application of function point analysis)* **(2)** project to develop and deliver the first release of a software application *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.18) Note:* It entails the specification, construction, testing, and delivery of a new application. During actualization, this project can be split up into a number of sub-projects. If these are carried out more or less in parallel, each being responsible for effectuating a certain sub-system of the total application, then each sub-project can be considered as an individual development project, if the sub-system itself is an application. Re-building an existing application, otherwise known as re-engineering, is considered as development.

requirements document. (1) document containing any combination of requirements or regulations to be met by a ready to use software product (RUSP) *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.13) See also:* requirements documentation

product description. (1) document stating properties of software, with the main purpose of helping potential acquirers in the evaluation of the suitability for themselves of the software before purchasing it *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.44) Note:* The product description is not a specification; it serves a different purpose.

consumer software package. (1) COTS software product designed and sold for end users to carry out identified functions; the software and its associated documentation are packaged for sale as a unit *(ISO/IEC/IEEE 24765a:2011)*

state name. (1) unique identifier of the state of software execution *(ISO/IEC 11411:1995 Information technology -- Representation for human communication of state transition of software, 2.1)*

assessment. (1) action of applying specific documented criteria to a specific software module, package or product for the purpose of determining acceptance or release of the software module, package or product *(ISO/IEC 14102:2008 Information Technology - Guideline for the evaluation and selection of CASE tools, 3.1)* **(2)** process that evaluates a person's fulfillment of the requirements of the certification scheme *(ISO/IEC 24773-1:2019 Software and systems engineering-Certification of software and systems engineering professionals-Part 1: General requirements, 3.2)*

architectural design phase. (1) life-cycle phase in which a system's general architecture is developed, thereby fulfilling the requirements laid down by the software requirements document and detailing the implementation plan in response to it *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

architectural design review. (1) joint acquirer-supplier review to evaluate the technical adequacies of the software architectural design as depicted in the software design descriptions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

impact analysis. (1) identification of all system and software products that a change request affects and development of an estimate of the resources needed to accomplish the change *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* 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.

requirements engineering. (1) interdisciplinary function that mediates between the domains of the acquirer and supplier to establish and maintain the requirements to be met by the system, software or service of interest *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-Requirements engineering, 4.1.19) Note:* Requirements engineering is concerned with discovering, eliciting, developing, analyzing, determining verification methods, validating, communicating, documenting, and managing requirements *See also:* software requirements engineering

baseline document. (1) system or software document that defines a work product that has been placed under configuration management *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

conciseness. (1) software attributes that provide implementation of a function with a minimum amount of code *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

correctability. (1) degree of effort required to correct software defects and to cope with user complaints *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

cross-reference tool. (1) software maintenance tool that lets the user determine where a variable is used or where a particular procedure is called on *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

database design specification. (1) document that describes the content and format of the permanent or semi-permanent data necessary for the software to carry out its functions *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

design-to-cost. (1) approach to managing a system or software project so as to hold the project to a predetermined cost *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Actual and projected costs are closely tracked, and actions such as deleting or postponing lower-priority requirements are taken if costs threaten to exceed targets. *Syn:* cost as an independent variable (CAIV), design to cost

detailed design review. (1) milestone review to determine the acceptability of the detailed software design (as depicted in the detailed design description) to satisfy the requirements of the software requirements document *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

executable requirements specification. (1) software requirement specification that is represented in an executable requirements language *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

expandability. (1) degree of effort required to improve or modify software functions' efficiency *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* extendability

external I/O device. (1) hardware input and/or output device that is outside the software system and part of the external environment *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

formal design. (1) process of using a formal method for software design *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

formal requirements language. (1) artificial language used to represent a software requirement *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The resulting formal requirements can be proven "correct" through proof-of-correctness methods. *Syn:* verifiable requirements language

hardware configuration item (HCI). (1) aggregation of hardware that is designated for configuration management and treated as a single entity in the configuration management process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* An HCI exists where functional allocations have been made that clearly distinguish equipment functions from software functions and where the hardware has been established as a configuration item. *See also:* software configuration item

integrate. (1) to combine software components, hardware components, or both into an overall system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** to pull in the changes from one child branch into its parent *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

inverse engineering. (1) process of obtaining a high-level representation of the software from the source code *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Inverse engineering provides a more abstract view of the system with the intent of recapturing design and requirements information. *See also:* reverse engineering

maintenance manual (MM). (1) software engineering project-deliverable document that enables a system's maintenance personnel (rather than users) to maintain the system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* operator manual, user manual

maintenance personnel. (1) software engineers who maintain software systems *(ISO/IEC/IEEE 24765:2017*

Systems and software engineering-Vocabulary)

maintenance project. (1) 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 *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

nonfunctional requirement. (1) software requirement that describes not what the software will do but how the software will do it *(IEEE 7000:2021, IEEE Standard Model Process for Addressing Ethical Concerns during System Design, 3.1)* **(2)** specification of how a system shall be developed and maintained, or how it shall perform in operation *(IEEE 7002:2022, IEEE Standard for Data Privacy Process, 3.1) Syn:* design constraint, non-functional requirement, performance requirement, performance attribute *See also:* functional requirement, quality requirement

performance analysis. (1) quantitative analysis of a real-time system (or software design) executing on a given hardware configuration with a given external workload applied to it *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

program librarian. (1) person responsible for establishing, controlling, and maintaining a software development library *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

programming. (1) general activity of software development *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** designing, writing, modifying, and testing of programs *(ISO/IEC 2382:2015 Information technology -- Vocabulary) Note:* especially construction activities. *See also:* construction

requirements traceability tool. (1) software development tool that establishes a traceability among itemized software requirements specifications, design elements, code elements, and test cases *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* It also supports various associated query, analysis, and report-generation capabilities.

software component (SC). (1) entity with discrete structure, such as an assembly or software module, within a system considered at a particular level of analysis *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.36)* **(2)** functionally or logically distinct part of a software configuration item (SCI), distinguished for the purpose of convenience in designing and specifying a complex SCI as an assembly of subordinate elements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(3)** software system or element *(ISO/IEC TR 29110-1:2016 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 1: Overview, 3.57) Note:* Software component refers to a part of a whole, such as a component of a software product or a component of a software identification tag.

software configuration item (SCI). (1) software entity that has been established as a configuration item *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The SCI exists where functional allocations have been made that clearly distinguish equipment functions from software functions and where the software has been established as a configurable item. *See also:* computer software component, computer software configuration item, hardware configuration item, software item

software design. (1) use of scientific principles, technical information, and imagination in the definition of a software system to perform pre-specified functions with maximum economy and efficiency *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software design audit. (1) review of a software product to determine compliance with requirements, standards, and contractual documents *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software design concept. (1) fundamental idea (such as information hiding) that can be applied to designing a system *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software design notation. (1) means of describing a software design *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* It can be diagrammatic, symbolic, or textual. *Syn:* software design representation

software design verification. (1) evaluation of a design to determine correctness with respect to stated requirements, conformance to design standards, system efficiency, and other criteria *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software release management. (1) management of the activities surrounding the release of one or more versions of software to one or more customers, including identifying, packaging, and delivering the elements of a product *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1) See also:* software configuration management, version

software requirement. (1) software capability needed by a user to solve a problem or to achieve an objective *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software requirements analysis. (1) process of studying user needs to arrive at a definition of system, hardware, or software requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software requirements engineering. (1) science and discipline concerned with analyzing and documenting software requirements *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** software requirements elicitation, analysis, specification, verification, and management *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* It involves transforming system requirements into a description of software requirements, performance parameters, and a software configuration using an iterative process of definition, analysis, trade-off studies, and prototyping. *See also:* requirements engineering

software requirements management. (1) process of planning and controlling the identification, allocation, and flow-down of requirements from the system level to the module or part level, including interfaces, verification, modifications, and status monitoring *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* requirements management

software requirements phase. (1) software development life-cycle phase during which the requirements for a software product, such as functional and performance capabilities, are defined, documented, and reviewed *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software requirements verification. (1) process of ensuring that the software requirements specification complies with the system requirements, conforms to document standards of the requirements phase, and is an adequate basis for the architectural (preliminary) design phase *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* requirements verification, requirements validation

tag. (1) symbolic name assigned to a specific release or a branch *(ISO/IEC/IEEE 24765:2017 Systems and software* *engineering-Vocabulary)* **(2)** information structure that provides authoritative information about a software asset in order to facilitate its management *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* provides developers and end users with a unique reference to the code base they are working with.

task structuring. (1) software design stage with the objective of structuring a concurrent application into concurrent tasks and defining the task interfaces *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

trunk. (1) software's main line of development; the main starting point of most branches *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* One can often distinguish the trunk from other branches by the version numbers used for identifying its files, which are shorter than those of all other branches.

Unified Modeling Language (UML). (1) graphical language for visualizing, specifying, constructing, and documenting an object-oriented software- intensive system's artifacts *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

unit test. (1) testing of individual routines and modules by the developer or an independent tester *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** test of individual programs or modules in order to ensure that there are no analysis or programming errors *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(3)** test of individual hardware or software units or groups of related units *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

vendor branch. (1) branch for keeping track of versions of imported software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Differences between successive versions can then be readily applied to the locally modified import.

software asset management (SAM). (1) control and protection of software and related assets within an organization, and control and protection of information about related assets which are needed in order to control and protect software assets *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.35)* **(2)** coordinated activity of an organization to realize value from software assets *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.51) Note:* infrastructure and processes
necessary for the effective management, control and protection of the software assets within an organization, throughout all stages of their lifecycle

conformity evaluation report. (1) document that describes the conduct and results of the evaluation carried out for a Ready to Use software product (RUSP) *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.4)*

test environment. (1) environment containing facilities, hardware, software, firmware, and procedures needed to conduct a test *(ISO/IEC/IEEE 29119-2:2021, Software and systems engineering--Software testing--Part 2: Test processes, 3.34) Note:* A test environment can contain multiple environments to accommodate specific test levels or types, e.g., a unit test environment, a performance test environment. A test environment can comprise several interconnected systems or virtual environments.

assistive technologies. (1) hardware or software that is added to or incorporated within a system that increases accessibility for an individual *(ISO/IEC 25062:2006 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Common Industry Format (CIF) for usability test reports, 4.11)*

disaster recovery. (1) in computer system operations, the return to normal operation after a hardware or software failure *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** ability of the information and communications technology elements of an organization to support its critical business functions to an acceptable level within a predetermined period following a disaster *(ISO/IEC/IEEE 26511:2018 Systems and software engineering--Requirements for managers of information for users of systems, software, and services, 3.1.11)*

post-closure activities. (1) activities that occur after a software system has been formally accepted by its customer *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* These activities include, but are not limited to, lessons-learned reviews and archiving project materials.

process infrastructure. (1) internal structure of the software life-cycle process, to include lifecycle phases, documentation, baselines, reviews, and products *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

computer center. (1) facility that includes personnel, hardware, and software, organized to provide information processing services *(ISO/IEC 2382:2015 Information technology -- Vocabulary) Syn:* data processing center

data processing system. (1) one or more computers, peripheral equipment, and software that perform data processing *(ISO/IEC 2382:2015 Information technology -- Vocabulary) Syn:* computer system, computing system

information system. (1) information processing system, together with associated organizational resources such as human, technical, and financial resources, which provides and distributes information *(ISO/IEC 2382:2015 Information technology -- Vocabulary)* **(2)** all of the functions (input, output, transport, processing, and storage) of an application, databases, technical facilities, and manual procedures which support business processes *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.21)* **(3)** one or more computer systems and communication systems together with associated organizational resources such as human, technical, and financial resources that provide and distribute information *(ISO/IEC 25024:2015 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of data, 4.24)* **(4)** system that is comprised of software, hardware, communication facility, data, and the people who use it in a given environment to satisfy their information processing needs *(ISO/IEC 25030:2019 Systems and software engineering--Systems and software quality requirements and evaluation (SQuaRE)--Quality requirements framework, 3.10)* **(5)** system which is designated to collect, organize, store, communicate, and process data *(IEEE 7005 2021, IEEE Standard for Transparent Employer Data Governance, 3.1) See also:* application

functional unit. (1) entity of hardware or software, or both, capable of accomplishing a specified purpose *(ISO/IEC 2382:2015 Information technology -- Vocabulary)*

software package. (1) complete and documented set of software supplied for a specific application or function *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.44) Note:* the set of files associated with a specific set of business functionalities that can be installed on a computing device and has a set of specific licensing requirements *See also:* software product

computer crime. (1) crime committed through the use, modification, or destruction of hardware, software, or data

(ISO/IEC 2382:2015 Information technology -- Vocabulary)

software piracy. (1) illegal use or copying of software products *(ISO/IEC 2382:2015 Information technology -- Vocabulary)*

system design. (1) process of defining the hardware and Software architecture, components, modules, interfaces and data for a system to satisfy specified requirements *(ISO/IEC 2382:2015 Information technology -- Vocabulary)*

system description. (1) documentation that results from system design defining the organization, essential characteristics and the hardware and software requirements of the system *(ISO/IEC 2382:2015 Information technology -- Vocabulary)*

SPI. (1) schedule performance index *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(2)** serial peripheral interface *(ISO/IEC/IEEE 24765d:2015)* **(3)** system/software process improvement *(ISO/IEC TR 29110-3-4:2015 Systems and software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 3-4: Autonomy-based improvement method, 4.2)*

technique. (1) a defined systematic procedure employed by a human resource to perform an activity to produce a product or result or deliver a service, and that may employ one or more tools. *(A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Sixth Edition)* **(2)** methods and skills required to carry out a specific activity *(ISO/IEC 25001:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Planning and management, 4.5)* **(3)** technical or managerial procedure that aids in the evaluation and improvement of the software development process *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

platform. (1) type of computer or hardware device and/or associated operating system, or a virtual environment, on which software can be installed or run **(2)** combination of an operating system and hardware that makes up the operating environment in which a program runs *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.30)* **(3)** type of computer or hardware device and/or associated operating system, or a virtual environment on which software can be installed or run *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.23) Note:* A platform is distinct from the unique instances of that platform, which are typically referred to as devices or instances. *See also:* device

critical design review (CDR). (1) review conducted to verify that the detailed design of one or more configuration items satisfy specified requirements; to establish the compatibility among the configuration items and other items of equipment, facilities, software, and personnel; to assess risk areas for each configuration item; and, as applicable, to assess the results of producibility analyses, review preliminary hardware product specifications, evaluate preliminary test planning, and evaluate the adequacy of preliminary operation and support documents *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)* **(2)** review as in (1) of any hardware or software component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

software reliability management. (1) process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources (cost), schedule, and performance *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

safety-critical software. (1) software that falls into one or more of the following categories: a) software whose inadvertent response to stimuli, failure to respond when required, response out-of-sequence, or response in combination with other responses can result in an accident b) software that is intended to mitigate the result of an accident c) software that is intended to recover from the result of an accident *(IEEE 1228-1994 (R2002) IEEE Standard for Software Safety Plans, 3.1.4)*

software development process. (1) process by which user needs are translated into a software product *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use. These activities can overlap or be performed iteratively.

software diversity. (1) software development technique in which two or more functionally identical variants of a program are developed from the same specification by different programmers or programming teams with the intent of providing error detection, increased reliability, additional documentation, or reduced probability that programming or compiler errors will influence the end results *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

detailed design phase. (1) software development lifecycle phase during which the detailed design process takes place, using the software system design and software architecture from the previous phase (architectural design) to produce the detailed logic for each unit such that it is ready for coding *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

external interface requirement. (1) system or software requirement that specifies a hardware, software, or database element with which a system/software system or system/software component must interface, or that sets forth constraints on formats, timing, or other factors caused by such an interface *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

management review. (1) systematic evaluation of a software acquisition, supply, development, operation, or maintenance process performed by or on behalf of management that monitors progress, determines the status of plans and schedules, confirms requirements and their system allocation, or evaluates the effectiveness of management approaches used to achieve fitness for purpose *(ISO/IEC/IEEE 24765i:2020)*

project management software. (1) applications specifically designed to aid the project management team with planning, monitoring, and controlling the project, including cost estimating, scheduling, collaboration, and risk analysis *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

SEMDM. (1) software engineering metamodel for development methodologies *(ISO/IEC 24744:2014 Software Engineering--Metamodel for development methodologies, 4.2)*

evaluation activity. (1) assessment of systems or software product against targeted values of identified and applicable quality characteristics performed using applicable techniques or methods *(ISO/IEC 25001:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Planning and management, 4.1)*

evaluation group. (1) organization responsible for specifying the systems and software quality requirements as well as managing and implementing the quality evaluation activities through the provision of technology, tools, experiences, and management skills *(ISO/IEC 25001:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Planning and management, 4.3) Note:* Software quality requirements could be specified previously by the requestor of the evaluation, while the evaluation group would verify presence and value of the software quality requirements.

hardware engineering. (1) application of a systematic, disciplined, and quantifiable approach to design, implement, and maintain a tangible product by transforming a set of requirements that represent the collection of stakeholder needs, expectations, and constraints; using documented techniques and technology *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* software engineering, systems engineering

SAM. (1) software asset management *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.2)*

functional service. (1) base functional component (BFC) *(ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, 3.6)* **(2)** service that must be implemented in the piece of software in order to fulfill functional user requirements *(ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, A.9)*

reading reference. (1) data storage entity or record, or interface record from another software or system containing data retrieved in a BFC *(ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, 3.8) Note:* The number of reading references is equal to 0 for all BFC types where it is applicable.

writing reference. (1) data storage entity or other record, or interface record to another software or system to which data is written in a BFC *(ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, 3.10) Note:* The number of writing references is greater than 0 with all BFC types where it is applicable.

FiSMA. (1) Finnish Software Measurement Association *(ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, A.10) Note:* a network of Finnish companies, which share interest in developing software measurement and/or software processes.

multi-component system. (1) measured system consisting of more than one piece of software *(ISO/IEC 20968:2002 Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, A.13) (ISO/IEC 29881:2010 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, A.10)*

convention. (1) requirement employed to prescribe a disciplined, uniform approach to providing consistency in a software product, that is, a uniform pattern or form for arranging data *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) See also:* practice, standard

integrity level scheme. (1) set of system characteristics (such as complexity, risk, safety level, security level, desired performance, reliability, or cost) selected as important to stakeholders, and arranged into discrete levels of performance or compliance (integrity levels), to help define the level of quality control to be applied in developing or delivering the software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

reusable product. (1) system, software, or hardware product developed for one use but having other uses, or one developed specifically to be usable on multiple projects or in multiple roles on one project *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1) Note:* Each use can include all or part of the product and can involve its modification. This term can be applied to any software or system product (for example, requirements or architectures), not just to software or system itself.

software-based system. (1) computer system controlled by software *(ISO/IEC/IEEE 24765:2017 Systems and software engineering--Vocabulary)*

context-sensitive help. (1) type of onscreen information for users in which the material that is displayed depends upon the user's view of the software current status of the software and the progress of the users task *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.14) See also:* embedded documentation, printed documentation

customization. (1) adaptation of a software or information product to the needs of a particular audience *(ISO/IEC/IEEE 26512:2018 Systems and software engineering--Requirements for acquirers and suppliers of information for users, 3.7)* **(2)** modification of a document type definition to add new structures or change the document type definition in a way that is not compatible with a previous structure *(ISO/IEC/IEEE 26531:2023 Systems and software engineering -- Content management for product lifecycle, user and service management information for users, 3.1.9)*

embedded documentation. (1) information that is delivered as an integral part of a piece of software *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.14) See also:* separate documentation

on-screen documentation. (1) information that is intended to be read on the screen by the user while using the software *(ISO/IEC/IEEE 26512:2018 Systems and software engineering--Requirements for acquirers and suppliers of information for users, 3.15) Syn:* on-screen information *See also:* printed documentation, embedded documentation

real-world object. (1) entity that exists in a three-dimensional form and, by association, implies similar properties or behavior to software functions *(ISO/IEC/IEEE 24765k:2022)*

separate documentation. (1) documentation that can be used independently of the software *(ISO/IEC/IEEE 24765g:2018) See also:* embedded documentation

window. (1) area with visible boundaries that presents a view of a software object or through which a user conducts a dialog with a computer system *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.57)*

Government-off-the-Shelf. (1) software supplied by the government for reuse in another project *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Syn:* GOTS, Government-Off-The-Shelf, Government off the Shelf, Government Off The Shelf *See also:* COTS

information item. (1) separately identifiable body of information that is produced, stored, and delivered for human use *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.24) (ISO/IEC/IEEE 15289:2019 Systems and software engineering--Content of life-cycle information items (documentation), 5.11) (ISO/IEC/IEEE 15288:2023 Systems and software engineering--System life cycle processes, 3.18) (ISO/IEC 30103:2015 Software and Systems Engineering - Lifecycle Processes - Framework for Product Quality Achievement, 3.3)* **(2)** separately identifiable body of information that is produced and stored for human use during a system or software life

cycle *(ISO/IEC 25063:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) Common Industry Format (CIF) for usability: Context of use description) Note:* The term information product is often used as a synonym.
An information item can be produced in several versions during a system, software, or service life cycle. *See also:* document, information product
SE. (1) systems engineering *(ISO/IEC/IEEE 24765i:2020)* **(2)** software engineering *(ISO/IEC/IEEE 24765i:2020)*
SWEBOK. (1) Software Engineering Body of Knowledge *(ISO/IEC/IEEE 16326:2019 Systems and software engineering -- Life cycle processes -- Project management, 3)*
legacy software. (1) software originally created without information structures *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.17)*
SAM practitioner. (1) individual involved in the practice or role of managing software assets *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.31) Note:* A SAM practitioner is often involved in the collection or reconciliation of software inventory and software entitlements.
development project functional size. (1) measure of the functionality provided to the users with the first release of the software, as measured by the development project function point count *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.19) Note:* The functional size of a development project can include the size of conversion functionality.
user-recognizable. (1) of requirements for processes or data, agreed upon and understood by both the user and the software developer *(ISO/IEC 20926:2009 Software and systems engineering -- Software measurement -- IFPUG functional size measurement method 2009, 3.51) Syn:* user recognizable
adaptability. (1) degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.8.1)*

(2) ability of a system to react to changes in its environment in order to continue meeting both functional and non-functional requirements *(ISO/IEC TR 29119-11:2020, Software and systems engineering--Software testing--Part 11: Guidelines on the testing of AI-based systems, 3.1.5) Note:* Adaptability includes the scalability of internal capacity, such as screen fields, tables, transaction volumes, and report formats. Adaptations include those carried out by specialized support staff, business or
operational staff, or end users. If the system is to be adapted by the end user, adaptability corresponds to suitability for individualization as defined in ISO 9241-110. *See also:* flexibility
replaceability. (1) degree to which a product can replace another specified software product for the same purpose in the same environment *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.2.8.3) Note:* Replaceability of a new version of a software product is important to the user when upgrading. Replaceability will reduce lock-in risk, so that other software products can be used in place of the present one, *See also:* adaptability, installability
external measure of software quality. (1) measure of the degree to which a software product enables the behavior of a system to satisfy stated and implied needs for the system including the software to be used under specified

conditions *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.3.5) Note:* Attributes of the behavior can be verified or validated by executing the software product during testing and operation. *See also:* external software quality, internal measure of software quality

internal measure of software quality. (1) measure of the degree to which a set of static attributes of a software product satisfies stated and implied needs for the software product to be used under specified conditions *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.16) (ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.3.7) Note:* Static attributes include those that relate to the software architecture, structure and its components. Static attributes can be verified by review, inspection, simulation, or automated tools. *See also:* external measure of software quality

quality measure. (1) measure that is defined as a measurement function of two or more values of quality measure elements *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.3.10)* **(2)** derived measure that is defined as a measurement function of two or more values of quality measure elements *(ISO/IEC 25021:2012 Software engineering--Software product Quality Requirements and Evaluation (SQuaRE)--Quality measure elements, 4.13) Syn:* QM *See also:* software quality measure

software quality requirement. (1) requirement that a software quality attribute be present in software *(ISO/IEC 25010:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--System and software quality models, 4.3.14)*

commercial-off-the-shelf software product. (1) software product defined by a market-driven need, commercially available, and whose fitness for use has been demonstrated by a broad spectrum of commercial users *(ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.6) Syn:* COTS software product

evaluation coverage. (1) degree to which the evaluation covers the specified software product quality requirements *(ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.17)*

evaluation stringency. (1) degree required for the software product quality characteristics and subcharacteristics to fulfill the expected use criticality of the software product *(ISO/IEC 25040:2011 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation process, 4.24)*

sprint. (1) short time frame, in which a set of software features is developed, leading to a working product that can be demonstrated to stakeholders *(ISO/IEC/IEEE 24765h:2019) Note:* In some organizations, a sprint is known as an iteration.

user story. (1) simple narrative illustrating a user requirement from the perspective of a persona *(ISO/IEC/IEEE 26515: 2018 Systems and software engineering: Developing information for users in an agile environment, 3.16)* **(2)** a narrative description of a software requirement, function, feature, or quality attribute, presented as a narrative of desired user interactions with a software system *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

screen capture. (1) representation of what the user will see while using the software *(ISO/IEC/IEEE 24765a:2011)* *Syn:* screen dump

requirements management. (1) activities that ensure requirements are identified, documented, maintained, communicated and traced throughout the life cycle of a system, product, or service *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-Requirements engineering, 4.1.20)* **(2)** provision of storing and editing capabilities, tracking history of edition, versioning, author identification, change management, time stamping, user notification for content changes, security rights control *(ISO/IEC TR 24766:2009 Information technology--Systems and software engineering--Guide for requirements engineering tool capabilities, 3.3) See also:* software requirements management

requirements validation. (1) confirmation that requirements (individually and as a set) define the right system as intended by the stakeholders *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-Requirements engineering, 4.1.22) See also:* requirements verification, software requirements validation

requirements verification. (1) confirmation by examination that requirements (individually and as a set) are well formed *(ISO/IEC/IEEE 29148:2018 Systems and software engineering-Life cycle processes-Requirements engineering, 4.1.23) Note:* This means that a requirement or a set of requirements has been reviewed to ensure the characteristics of good requirements are achieved. *See also:* requirements validation, software requirements verification

generic profile group. (1) profile group applicable to very small entities (VSEs) that do not develop critical systems or software products and have typical situational factors *(ISO/IEC TR 29110-1:2016 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 1: Overview, 2.9) (ISO/IEC 29110-2-1:2015 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 2-1: Framework and taxonomy, 4.28)*

CM tool. (1) software product that can assist software engineers by providing automated support for configuration management activities *(ISO/IEC TR 18018:2010 Information technology--Systems and software engineering--Guide for configuration management tool capabilities, 3.9) Syn:* configuration management tool

software/system element. (1) element that defines and prescribes what a software or system is composed of *(ISO/IEC TR 18018:2010 Information technology--Systems and software engineering--Guide for configuration management tool capabilities, 3.12) Note:* An element can contain sub elements or other software/system elements that are dependent on the top-level element. *See also:* software element

user interface element. (1) entity of the user interface that is presented to the user by the software *(ISO/IEC TR 25060:2010 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Common Industry Format (CIF) for usability: General framework for usability-related information, 2.24) Note:* User interface elements can be interactive or not, and either entities relevant to the task or entities of the user interface *Syn:* user interface object

SCMP. (1) software configuration management plan *(ISO/IEC/IEEE 24765d:2015)*

SQAP. (1) software quality assurance plan *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.3)*

software baseline. (1) set (one or more) of software configuration items formally designated and fixed at a specific time during the software life cycle *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and*

Software Engineering, 2.1)

constituent configuration item. (1) individual item to be controlled that is a constituent (part) of a larger configuration item, such as a reference model, hardware prototype or software build *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)*

software version ID. (1) explicit and immutable version identifier (name or number) inserted into each configuration item, including each individual release, that can be used to identify the exact version of the configuration item in any instance or repository *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1) Syn:* software version identification

versioning. (1) assignment of either unique version names or unique version numbers to unique states of software configuration items, usually for a specific purpose, such as a release of the software product to an external group or the identification of a specific baseline *(IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1)*

software identification tag. (1) information structure containing identification information about a software configuration item, which can be authoritative if provided by a software creator *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.40)* **(2)** set of structured data elements containing authoritative identification information about a software configuration item *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.31) Syn:* SWID tag, SWID

software consumer. (1) entity that uses an entitlement of a software package *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.37)*

software creator. (1) person or organization that creates a software product or package *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.38) Note:* This entity might or might not own the rights to sell or distribute the software.

software developer. (1) person who creates software *(ISO/IEC/IEEE 24765e:2015) Note:* Often a software developer works with other developers for a software manufacturer to create commercial applications. A software developer can also often work as an in-house developer of software for use by the software developer's own organization.

software entitlement. (1) software license use rights as defined through agreements between a software licensor and a software consumer *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.39) Note:* Effective use rights take into account any contracts and all applicable licenses, including full licenses, upgrade licenses, and maintenance agreements.

software licensor. (1) person or organization who owns or holds the rights to issue a software license for a specific software package *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.43)*

software packager. (1) entity that packages or bundles software created by others *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.45)*

value-added reseller (VAR). (1) company licensed to repackage and support existing products, such as combined software packages *(ISO/IEC/IEEE 24765g:2018) Syn:* value added reseller

software license. (1) legal rights to use software in accordance with terms and conditions specified by the software licensor *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.41) Note:* Using a software product can include accessing, copying, distributing, installing and executing the software product, depending on the license's terms and conditions

architecture-driven modernization (ADM). (1) process of understanding and evolving existing software assets of a system of interest *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4) Note:* ADM does not preclude source-to-source migrations (where appropriate), but encourages user organizations to consider modernization from an analysis and design perspective.

runtime platform. (1) set of hardware and software components that implement the services utilized by the application software *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)*

software artifact. (1) tangible machine-readable document created during software development *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4) Syn:* software artefact

software asset. (1) software that has potential or actual value to an organization *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.50)* **(2)** description of a partial solution (such as a component or design document) or knowledge (such as requirements database or test procedures) that engineers use to build or modify software products *(ISO/IEC 19506:2012 Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM), 4)*

deliverable product. (1) unique and verifiable system or software product to perform a service, that is subject to approval by the project sponsor or customer *(ISO/IEC 25041: 2012 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation guide for developers, acquirers and independent evaluators, 4.1)*

dynamic product. (1) system or software product that is measurable during execution in a testing or an operational environment *(ISO/IEC 25041: 2012 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation guide for developers, acquirers and independent evaluators, 4.2)*

intermediate product. (1) system or software product of the development process that is used as inputs to other stages of the development process *(ISO/IEC 25041: 2012 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation guide for developers, acquirers and independent evaluators, 4.10) See also:* intermediate system or software product

product quality. (1) degree to which the product satisfies stated and implied needs when used under specified conditions *(ISO/IEC 25041: 2012 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation guide for developers, acquirers and independent evaluators, 4.11)* **(2)** capability of a system and/or software to satisfy stated and implied needs when used under specified conditions *(ISO/IEC 25020:2019 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Quality measurement framework, 3.17) Note:* This definition differs from the ISO 9000:2015 quality definition, because this definition refers to the satisfaction of stated and implied needs, while the ISO 9000 quality definition refers to the satisfaction of requirements. *Syn:* system and software product quality

static product. (1) non-executable system or software product for reviewing *(ISO/IEC 25041: 2012 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--Evaluation guide for developers, acquirers and independent evaluators, 4.12)*

functional user. (1) user that is a sender or an intended recipient of data in the Functional User Requirements of a piece of software *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.13)*

peer software. (1) piece of software that resides in the same layer as, and exchanges data with, another piece of software *(ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method, 2.21)*

SAM program scope. (1) clear statement listing of all parts of the organization and types of software, assets, and platforms covered by a SAM program *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.32)*

software usage. (1) consumption against a software entitlement measured as defined by the terms and conditions of that entitlement *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.47) Note:* Depending on the specific terms and conditions, usage can include accessing, copying, distributing, installing and executing software.

stock keeping unit (sku). (1) identification, usually alphanumeric, of a particular product that allows it to be tracked for inventory and software entitlement purposes *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.48) Note:* typically associated with unique products for sales purposes, such as software entitlements. It can correspond uniquely to specific software products, or represent packages of software, with specific terms and conditions, such as whether it relates to a full product, upgrade product, or maintenance on an existing product.

definitive software library (DSL). (1) secure storage environment, formed of physical media or of one or more electronic software repositories, capable of control and protection of definitive authorized versions of all software configuration items and masters of all software controlled by SAM *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.11)*

DSL. (1) definitive software library *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.11)*

SWID. (1) software identification (tag) *(ISO/IEC 19770-5:2015 Information technology--IT asset management--Overview and vocabulary, 3.40)*

specification-based testing. (1) testing in which the principal test basis is the external inputs and outputs of the test item, commonly based on a specification, rather than its implementation in source code or executable software *(ISO/IEC/IEEE 29119-1:2022, Software and systems engineering--Software testing--Part 1: General concepts, 3.75) Syn:* black-box testing, closed-box testing *See also:* functional testing

portability testing. (1) type of testing conducted to evaluate the ease with which a test item can be transferred from one hardware or software environment to another, including the level of modification needed for it to be executed in various types of environments *(ISO/IEC/IEEE 29119-2:2021, Software and systems engineering--Software testing--Part 2: Test processes, 3.59)*

test traceability matrix. (1) document, spreadsheet, or other automated tool used to identify related items in documentation and software, such as requirements with associated tests *(ISO/IEC/IEEE 29119-3:2021 Software and systems engineering--Software testing--Part 3: Test documentation, 3.26) Note:* Different test traceability matrices can have different information, formats, and levels of detail. *Syn:* verification cross reference matrix, requirements test matrix, requirements verification table *See also:* traceability matrix

ready to use software product (RUSP). (1) software product available to any user, at cost or not, to use without the need to conduct development activities *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.6) Note:* includes the product description (including cover information, data sheet, and website information; the user documentation necessary to install and use the software, including any configuration of the operating system or target computer required to operate the product; the software contained on a computer sensible media (e.g., disk or CD-ROM) or internet downloadable. Includes software produced and supported without typical commercial fees and licensing considerations. *Syn:* ready-to-use software product *See also:* commercial off the shelf (COTS)

RUSP. (1) ready-to-use software product *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.6)*

product identification. (1) software product name, version, variant, and date information *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.12)*

software function. (1) implementation of an algorithm in the software with which the end user or the software can perform part or all of a work task *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.1.14) Note:* a function does not need to be callable by the end user (e.g., automatic backup or data saving).

SQC. (1) software quality control *(ISO/IEC 25051:2014 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing, 4.2)*

backlog. (1) list of product requirements and deliverables not part of current work, to be prioritized and completed *(ISO/IEC/IEEE 24765h:2019)* **(2)** a set of software features awaiting development in a subsequent iteration *(Software Extension to the PMBOK(R) Guide Fifth Edition)* **(3)** collection of agile features or stories of both functional and nonfunctional requirements that are typically sorted in an order based on value priority *(ISO/IEC/IEEE 26515: 2018 Systems and software engineering: Developing information for users in an agile environment, 3.4)*

anchor point. (1) a milestone in software scheduling at which a major project life cycle transition occurs *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

burndown rate. (1) the number of software story points, features, functions, user stories, use cases, or requirements completed per work unit (week or iteration) *(Software Extension to the PMBOK(R) Guide Fifth Edition) See also:* velocity

increment. (1) tested, deliverable version of a software product that provides new or modified capabilities *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

refactor. (1) to restructure software code without altering its behavior for the purpose of improving quality attributes, easing future extension or adaptation, or adhering to an architectural style *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

release map. (1) a displayed forecast of when software features will be released and how they will be grouped into releases *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

retrospective meeting. (1) a team meeting at the end of an iterative cycle or at the end of a software project to reflect on what went well, what was learned, and what should be done differently next time *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

software quality assurance (SQA). (1) a set of activities that assess adherence to, and the adequacy of the software processes used to develop and modify software products. SQA also determines the degree to which the desired results from software quality control are being obtained. *(Software Extension to the PMBOK(R) Guide Fifth Edition)* **(2)** set of activities that define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended purposes *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2) Note:* A key attribute of SQA is the objectivity
of the SQA function with respect to the project. The SQA function can also be organizationally
independent of the project; that is, free from technical, managerial, and financial pressures from the project.

software quality control (SQC). (1) a set of activities that measure, evaluate and report on the quality of software project artifacts throughout the project life cycle *(Software Extension to the PMBOK(R) Guide Fifth Edition)*

workflow board. (1) in software development, a visual representation of work for developers who pull tasks from the task backlog; used for on-demand or resource-bound scheduling *(Software Extension to the PMBOK(R) Guide Fifth Edition) Syn:* kanban board

external measure of system or software quality. (1) measure of the degree to which a system or software product enables the behavior to satisfy stated and implied needs for the system, including the software to be used under specified conditions *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.11) Note:* Attributes of the behavior can be verified or validated by executing the system or software product during testing and operation.

SW. (1) software *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.2)*

development environment. (1) hardware, software, platform and tools for designers and developers of computer solutions *(ISO/IEC/IEEE 24765c:2014)*

development tool. (1) hardware and software for developing or modifying applications *(ISO/IEC/IEEE 24765c:2014)*

infrastructure. (1) hardware and software environment to support computer system and software design, development, and modification *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.25)* **(2)** facilities such as power, cooling, and physical security of the data center, networking, hardware, and software needed to support the systems life cycle and maintain information technology (IT) services *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1) Note:* Does not include the associated people or processes. In DevOps, software-defined infrastructure enables elasticity. *Syn:* ecosystem *See also:* IT infrastructure

critical system. (1) system having the potential for serious impact on the users or environment, due to factors including safety, performance, and security *(ISO/IEC 29110-2-1:2015 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 2-1: Framework and taxonomy, 4.22)* **(2)** those items (e.g. functions, parts, software, characteristics, processes) having significant effect on the product realization and use of the product -- including safety, performance, form, fit, function, producibility, service life, etc. -- that require specific actions to ensure they are adequately managed *(ISO/IEC TR 29110-5-6-2:2014 Systems and software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 5-6-2: Systems engineering--Management and engineering guide: Generic profile group: Basic profile, 3.2)*
SEP. (1) Systems Engineering Plan *(ISO/IEC TR 29110-5-6-2:2014 Systems and software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 5-6-2: Systems engineering--Management and engineering guide: Generic profile group: Basic profile, 3.5)* **(2)** systems and software engineering process *(ISO/IEC 29110-3-2:2018 Systems and software engineering-Lifecycle Profiles for Very Small Entities (VSEs)-Part 3-2: Conformity certification scheme, 4.1) Syn:* System Engineering Plan
electronic design automation (EDA). (1) software-driven design and development of electronic components such as microcomputer units and circuit boards *(ISO/IEC/IEEE 24765d:2015)*
middleware. (1) software layer between an operating system and the software applications *(ISO/IEC/IEEE 24765d:2015)*
embedded operating system. (1) operating system software for an embedded computer system *(ISO/IEC/IEEE 24765d:2015)*
evaluation module (EVM). (1) microcomputer module used in application development, e.g., to benchmark software, prototype applications, and debug algorithms for computer systems *(ISO/IEC/IEEE 24765d:2015)*
in-circuit emulator (ICE). (1) hardware device used to debug the software of an embedded system *(ISO/IEC/IEEE 24765d:2015)*
integrated development environment (IDE). (1) set of software tools or applications to provide comprehensive facilities for software development *(ISO/IEC/IEEE 24765d:2015)*
independence. (1) of software quality assurance (SQA), situation in which SQA is free from technical, managerial, and financial influences, intentional or unintentional *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)*
financial independence. (1) of software quality assurance (SQA), situation in which control of the SQA budget is vested in an organization independent of the development organization *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)*
managerial independence. (1) of software quality assurance (SQA), situation in which the responsibility of the SQA effort is vested in an organization separate from the development and project management organizations *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)*
technical independence. (1) of software quality assurance (SQA), situation in which the SQA effort uses personnel who are not involved in the development of the system or its elements *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)*
software quality management. (1) coordinated activities to direct and control an organization with regard to

software quality *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.2)*

software testing environment. (1) facilities, hardware, software, firmware, procedures, and documentation needed to perform testing of software *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes)*

SLC. (1) software life cycle *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.3)*

STE. (1) software test environment *(IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes, 3.3)* **(2)** Standard Technical English *(ISO/IEC/IEEE 26511:2018 Systems and software engineering--Requirements for managers of information for users of systems, software, and services, 3.2)*

interactive system. (1) combination of hardware, software and/or services that receives input from and communicates output to users *(ISO/IEC 25063:2014 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) Common Industry Format (CIF) for usability: Context of use description)* *Note:* This includes, where appropriate, packaging, branding, user documentation, online help, support and training.

hardware description language (HDL). (1) software programming language used to design and model hardware, especially digital logic circuits *(ISO/IEC/IEEE 24765d:2015)* *Syn:* hardware design language

IT system. (1) system which uses information technologies *(ISO/IEC TR 12182:2015 Systems and software engineering--Framework for categorization of IT systems and software, and guide for applying it, 3.3)* **(2)** set of one or more computers, associated software, peripherals, terminals, human operations, physical processes, information transfer means, that form an autonomous whole, capable of performing information processing and/or information transfer *(ISO/IEC TS 25011:2017 Information technology--Systems and software Quality Requirements and Evaluation (SQuaRE)--Service quality models, 3.3.5)*

target system. (1) complete computing platform capable of running the target software *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.28)* **(2)** system to be categorized, which can be an IT system and software, including service provided by IT system *(ISO/IEC TR 12182:2015 Systems and software engineering--Framework for categorization of IT systems and software, and guide for applying it, 3.4)* *Note:* consists of hardware resources and software resources installed on the hardware.

categorization space. (1) universal set of systems and software which has one or more classification axes as its individual dimension, by which stakeholder's concerns on categorization are expressed *(ISO/IEC TR 12182:2015 Systems and software engineering--Framework for categorization of IT systems and software, and guide for applying it, 3.6)*

classification axis. (1) total range of a mapping of systems and software for categorizing them from a particular perspective *(ISO/IEC TR 12182:2015 Systems and software engineering--Framework for categorization of IT systems and software, and guide for applying it, 3.7)*

SAD. (1) software architecture description *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.2)*

SAR. (1) software requirements and architecture review *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.2)* *See also:* SRR

STP. (1) software test plan *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.2)*

SVD. (1) software version description *(IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense*

Programs, 3.2)

functional system design. (1) specification of the functions of the components of a software system and of the working relationships between them *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.19)*

information provisioning. (1) collection of all the infrastructure tools, software applications, non-automated elements, data sets, user documentation, and organizational structures which serve to supply information to the business *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.20)*

IT infrastructure. (1) all the technical components, system software, databases and data files and deployed application software, technical procedures, and technical documentation used to make the information available *(ISO/IEC 16350-2015 Information technology--Systems and software engineering--Application management, 4.22)* **(2)** combined set of IT assets for developing, maintaining, and using IT services *(ISO/IEC 19770-1:2017 Information technology -- IT asset management -- Part 1: IT asset management systems--Requirements, 3.30)*

embedded middleware. (1) software that communicates between an embedded operating system and an embedded application or firmware *(ISO/IEC/IEEE 24765e:2015)*

SEI. (1) serial expansion interface *(ISO/IEC/IEEE 24765e:2015)* **(2)** Software Engineering Institute *(ISO/IEC/IEEE 24765e:2015)*

SIM. (1) [mobile telecommunications] subscriber identity module *(ISO/IEC/IEEE 24765e:2015)* **(2)** [embedded software] system integration module *(ISO/IEC/IEEE 24765e:2015)*

software element. (1) system element that is software *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.51)* **(2)** identifiable part of a software product. *(ISO/IEC/IEEE 90003:2018 Software engineering -- Guidelines for the application of ISO 9001:2015 to computer software, 3.8) See also:* system element, software/system element

FOSS. (1) free and open source software *(ISO/IEC/IEEE 24765f:2016)*

channel partner. (1) person or entity working with a software licensor or another person/entity within the channel who facilitates the sale of software to the end-user *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.4)*

downgrade right. (1) right granted to receive, install, or use an installation of a previous version of software than the currently granted entitlement *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.7)*

entitlement schema. (1) information structure containing a digital encapsulation of a licensing transaction and its associated entitlement information *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.11) Note:* A single transaction does not necessarily encapsulate a full (or effective) entitlement. An effective entitlement can be determined by an analysis of multiple licensing transactions, of a full license and then of upgrades and/or maintenance transactions assessed together with it. *Syn:* software entitlement schema, Ent

limit. (1) restriction on rights or privileges granted by a software entitlement *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.16)*

perpetual license. (1) license for a software entitlement granted in perpetuity *(ISO/IEC 19770-3:2016 Information* *technology--IT asset management--Part 3: Entitlement schema, 3.1.18) Note:* The alternative to a perpetual license is a term or subscription-based license.

right. (1) privilege or benefit granted by a software entitlement *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.22)*

SAM tool. (1) software used to assist in and automate parts of the process of management of software assets *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.25) Syn:* software asset management tool

software entitlement reconciliation. (1) process of comparing software entitlements owned with those required (granted and deployed), usually to determine compliance with software license agreements *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.30)*

software licensee. (1) person or organization granted a license to use a specific software product *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.1.33)*

Ent. (1) [software] entitlement schema *(ISO/IEC 19770-3:2016 Information technology--IT asset management--Part 3: Entitlement schema, 3.2)*

advanced profile. (1) profile targeted at very small enterprises (VSEs) which want to sustain and grow as an independent competitive system or software development business *(ISO/IEC 29110-2-1:2015 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 2-1: Framework and taxonomy, 4.3)*

service delivery profile. (1) profile targeted at very small enterprises (VSEs) that need to perform and manage service delivery processes, either for systems or software products that they have developed or that were developed by others *(ISO/IEC TR 29110-1:2016 Software engineering--Lifecycle profiles for Very Small Entities (VSEs)--Part 1: Overview, 3.54)*

automated systems process. (1) systems or software process that is performed either fully or partially supported by CASE tools *(ISO/IEC 15940:2013 Systems and software engineering--Software Engineering Environment Services, 2.2.3, 2.9) Syn:* assisted process, assisted software process, assisted systems process, automated process, automated software process

intermediate system or software product. (1) product of the system or software development process that is used as input to another stage of its
development process *(ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.15) Syn:* intermediate software product, intermediate system product *See also:* intermediate product

SSPL. (1) software and systems product line *(ISO/IEC 26559:2017 Software and systems engineering -- Methods and tools for variability traceability in software and systems product line, 4)*

software system element. (1) member of a set of elements that constitute a software system *(ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.56) Note:* A software system element can include one or more software units, software elements, hardware units, hardware elements, services, and other system elements and systems. A software system element can be viewed as a system element.

software development environment. (1) facilities, hardware, software, procedures, and documentation needed to perform software development *(ISO/IEC/IEEE 24748-5:2017 Systems and software engineering--Life cycle management--Part 5: Software development planning, 3.14) Note:* Plans for software development environments can include where the specified environment is to be constructed, when sites provide different environments or facilities. For example, different testing environments can be requested to be constructed at the acquirer's site and the supplier's site. *See also:* enabling system

online help. (1) information about the software that is intended to be read on the screen by the user while using the software *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.28) Note:* Online help can be displayed in a variety of forms (contextual help, screen tips, and examples).

test participant. (1) person who provides feedback and allows data collection to test that the information in the software documentation is sufficient to accomplish tasks correctly and form a conceptual understanding of the system *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.40)*

usability analyst. (1) person who observes users performing tasks using the software and documentation and records the actions the user took, problems the user encountered, and comments the user made during the test; and interprets these records to evaluate the results of the testing *(ISO/IEC/IEEE 26513:2017 Systems and software engineering--Requirements for testers and reviewers of information for users, 3.45)*

test execution engine. (1) tool implemented in software and sometimes in hardware that can manipulate the test item to execute test cases *(ISO/IEC/IEEE 29119-5:2016 Software and systems engineering--Software testing--Part 5: Keyword-driven testing, 4.11) Note:* A typical test execution engine includes unit test tool frameworks, stimulation-command systems, capture and playback tools or hardware robots, along with the software to control them.

IT service reliability. (1) degree to which an IT service provides consistent and stable IT service outcomes *(ISO/IEC TS 25011:2017 Information technology--Systems and software Quality Requirements and Evaluation (SQuaRE)--Service quality models, 3.2.4) See also:* reliability, software reliability

nth of a kind. (1) re-manufacturing or re-installation of a previously verified and validated hardware or software design *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1) Note:* The nth of a kind component or system is equivalent to the first
application in all relevant aspects, including functional and performance requirements, design documentation, environment, and regulatory constraints.

application programming interface (API). (1) set of functions, protocols, parameters, and objects of different formats, used to create software that interfaces with the features or data of an external system or service *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.6) Note:* APIs can take several forms. In general terms, an API is a set of clearly defined methods of communication among various components. An API specifies the information and methods that are needed to communicate with another application. Information for users of an API is of two main types: reference information, which contains information about all elements of the API) and developer guide (which explains how to use the API).

embedded help system. (1) information for users that is delivered as an integral part of a piece of software

(ISO/IEC/IEEE 26511:2018 Systems and software engineering--Requirements for managers of information for users of systems, software, and services, 3.1.14)

terminology management system. (1) software tool specifically designed for collecting, maintaining, and accessing terminological data *(ISO/IEC/IEEE 26511:2018 Systems and software engineering--Requirements for managers of information for users of systems, software, and services, 3.1.34)*

critical software. (1) software having the potential for serious impact on the users or environment, due to factors including safety, performance, and security *(ISO/IEC TR 29110-5-1-4:2018 Software and systems engineering-Lifecycle profiles for very small entities (VSEs)-Part 5-1-4: Software engineering: Management and engineering guidelines: Generic profile group: Advanced profile, 3.7)*

code data. (1) type of data entities used for software sizing (in addition to business data and reference data) *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* Code data usually exists to satisfy non-functional requirements (NFR) from the user (for quality requirements, physical implementation or a technical reason). Code data provides a list of valid values that a descriptive attribute may have. Typically, the attributes of the code data are Code, Description or other standard attributes describing the code; e.g., standard abbreviation, effective date, termination date, audit trail data.

family of platforms. (1) software platforms that are serving the same purpose *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1)*

non-functional size. (1) size of the software derived by quantifying the non-functional requirements (NFR), defined by a set of rules *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Syn:* nonfunctional size

partition. (1) set of software functions within an application boundary that share homogeneous criteria and values *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* A partition requires development effort, which may not be reflected when sizing the functional aspect of the project/product, using FPA. Partition is designed to improve non-functional perspective of the users, such as maintainability, portability, or installability, (and is not based on technical or physical considerations). Partitions do not overlap.

software non-functional assessment process (SNAP) category. (1) group of components, processes or activities that are used in order to meet the non-functional requirement *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* Categories classify the non-functional requirements (NFR), and are generic enough to allow for future technologies. Categories are divided into subcategories, so that the SNAP category groups the sub-categories based on the same level of operations or similar type of sizing activities. *Syn:* SNAP category

NFSSM. (1) non-functional software size measurement *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.2)*

SCU. (1) SNAP (software non-functional assessment process) counting unit *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.2)*

SNAP. (1) software non-functional assessment process *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.2)*

SP. (1) SNAP (software non-functional assessment process) point *(IEEE 2430-2019 Trial-Use Standard for Software*

Non-Functional Sizing Measurements, 3.2)

quality measure on external property. (1) measure of the degree to which a system or software product enables its behavior to satisfy stated and implied needs for the system including the software to be used under specified conditions *(ISO/IEC 25020:2019 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Quality measurement framework, 3.15) Syn:* QM on external property

quality measure on internal property. (1) measure of the degree to which a set of static attributes of a software product satisfies stated and implied needs for the software product to be used under specified conditions *(ISO/IEC 25020:2019 Systems and software engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE)--Quality measurement framework, 3.16) Note:* Quality measures on internal property are typically associated with quality requirements on static properties and attributes that can be specified in or derived from requirements. *Syn:* QM on internal property

non-functional requirement. (1) requirement for a software-intensive system or for a software product, including how it should be developed and maintained, and how it should perform in operation, except any functional user requirement for the software *(IEEE 2430-2019 Trial-Use Standard for Software Non-Functional Sizing Measurements, 3.1) Note:* Non-functional requirements (NFR) concern the software
system or software product quality, the environment in which the software system or software product must be implemented and which it must serve, and the processes and technology to be used to develop and maintain the software system or software product and the technology to be used for their execution. *See also:* functional user requirement

integration definition for functional modeling (IDEF). (1) family of modeling languages in the fields of systems and software engineering that provide a multiple&#8208;page (view) model of a system that depicts functions and information or product flow *(ISO/IEC/IEEE 24765l:2024) Note:* Boxes illustrate functions and arrows illustrate information and product flow. Alphanumeric coding is used to denote the view.

dynamic program analysis. (1) process of evaluating a software system or component based on its behavior during execution *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.5) Note:* The software is compiled and run on a certain number of input data test cases. The physical response from the system is then examined and compared to expected results. Dynamic program analysis can be done manually or using an automated process.

software security. (1) ability of software to protect its assets from a malicious attacker *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.22) Note:* Software security applies to software assets and is decomposed into the following set of properties: confidentiality, integrity, availability, authentication, authorization, and non-repudiation.

static program analysis. (1) sub-field of formal methods concerned with analyzing the properties of software code without executing this code in the target (binary) format *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.24)*

target of verification (TOV). (1) software or a set of software items or units to be verified, e.g. in terms of safety and security *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.26)*

vulnerability. (1) potential flaw or weakness in software design or implementation that could be exercised (accidentally triggered or intentionally exploited) and result in harm to the system *(ISO/IEC 23643:2020, Software and systems engineering--Capabilities of software safety and security verification tools, 3.36)*

fuzz testing. (1) software testing approach in which high volumes of random (or near-random) data, called fuzz, are used to generate inputs to the test item *(ISO/IEC TR 29119-11:2020, Software and systems engineering--Software testing--Part 11: Guidelines on the testing of AI-based systems, 3.1.38)*

software agent. (1) digital entity that perceives its environment and takes actions that maximize its chance of successfully achieving its goals *(ISO/IEC TR 29119-11:2020, Software and systems engineering--Software testing--Part 11: Guidelines on the testing of AI-based systems, 3.1.73)*

cryptographic hash. (1) Method to verify the authenticity of a system element or software via the production of a checksum *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1)*

DevOps. (1) set of principles and practices which enable better communication and collaboration between relevant stakeholders for the purpose of specifying, developing, and operating software and systems products and services, and continuous improvements in all aspects of the life cycle *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1)*

infrastructure as code (IaC). (1) definition, management, and provision of infrastructure components using software *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1) Note:* In DevOps, infrastructure as code facilitates the automation of the systems life cycle, enabling consistency, performance, and security across the system and resources

software as a service (SaaS). (1) access to resources from client devices through thin client interface or program interface *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1) Note:* The level of control over the resource provided to the consumer can vary with the service provider. In DevOps, SaaS can be automated as part of the DevOps pipeline. *See also:* IaaS, PaaS

SaaS. (1) software as a service *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1)*

continuous delivery. (1) software engineering practices that allow for frequent releases of new systems (including software) to staging or various test environments through the use of automated tools *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.1)*

OSS. (1) open source software *(IEEE 2675-2021, IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment, 3.2) See also:* FOSS

shared asset. (1) software and systems engineering lifecycle digital artifacts that compose a part of a delivered member product or support the engineering process to create and maintain a member product *(ISO/IEC 26580:2021, Software and systems engineering Methods and tools for the feature-based approach to software and systems product line engineering, 3.17) See also:* domain asset

customize. (1) adapt a software or information product to the needs of a particular audience *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.15)*

embedded information for users. (1) information for users that is accessed as an integral part of software *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.22) See also:* onscreen information for users, printed information for users

main library. (1) software library containing controlled versions of software and documentation from which working copies can be made for distribution and use *(ISO/IEC/IEEE 24765k:2022) Note:* replaces the deprecated term master library

additive maintenance. (1) modification of a software product performed after delivery to add functionality or features to enhance the usage of the product *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.2) Note:* Additive maintenance can be excluded from the definition of maintenance in the context of dependability that addresses recovery of a system to previous operational, functional and performance level, e.g. definition, monitor or measurement of availability, recoverability, or MTBF (mean time between failure). Additive maintenance provides additional new functions or features to improve software usability, performance, maintainability, or other software attributes for the future. It adds functionality or features with relatively large additions or changes on software for improving software attributes after delivery with identified opportunities to negotiate any of additions or changes on maintenance strategy, methods, resources, agreements, or service levels between suppliers and acquirers. Additions or enhancements can be handled through the maintenance process, while larger changes can involve a new development effort. *See also:* perfective maintenance

correction. (1) change that addresses and implements problem resolutions to recover gaps and to make software operational enough to meet defined operational requirements *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.3) Note:* The term correction is mainly used as a maintenance type and to classify modification requests (MR). *See also:* enhancement

software sustainment. (1) activities to support, maintain, and operate a software system to help ensure that the software system or service remains operational *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.13) Note:* Software sustainment includes processes, procedures, people, material, and information.

SQuaRE. (1) Software product Quality Requirements and Evaluation *(ISO/IEC TS 25052-1:2022, Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE): cloud services--Part 1: Quality model, 4)*

onscreen information for users. (1) information for users that is intended to be read on the screen by the user while using the software *(ISO/IEC/IEEE 26514:2022, Systems and software engineering -- Design and development of information for users, 3.1.36) See also:* embedded information for users, printed information for users

software correction. (1) change that addresses and implements problem resolutions to recover gaps and to make software operational enough to meet defined operational requirements *(ISO/IEC/IEEE 14764:2021, Software engineering -- Software life cycle processes -- Maintenance, 3.1.3) Note:* The term "correction" is mainly used as a maintenance type and to classify modification requests (MR). *See also:* enhancement