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 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)* **(2)** 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)* **(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) 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

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

software design description (SDD). (1) representation of software created to facilitate analysis, planning, implementation, and decision-making *(IEEE 1012-2024 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.

SDD. (1) software design description *(IEEE 1012-2024 IEEE Standard for System, Software, and Hardware Verification and Validation)*

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) See also:* data structure-centered design, input-process-output, modular decomposition, object-oriented design, rapid prototyping.

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)* **(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 TR 25060:2023 Systems and software engineering--Systems and software Quality Requirements and Evaluation (SQuaRE)--General Framework for Common Industry Format (CIF) 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)* **(6)** set of activities undertaken in order to achieve a specific goal *(ISO/IEC 25062:2025 Systems and software engineering -- Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Common Industry Format (CIF) for reporting usability evaluations, 3.13) Note:* While goals are independent of the means used to achieve them, tasks describe particular means of achieving goals. Related tasks are usually grouped to form activities.

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

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

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 deals *(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, 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 description. (1) model or information item that specifies the design of a system or component *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary) Note:* Typical contents include system or component architecture, control logic, data structures, input/output formats, interface descriptions, and algorithms. *Syn:* design document, design specification *See also:* product specification, requirements specification, software design description

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 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

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

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

order clash. (1) in software design, a type of structure clash in which a program 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

structure clash. (1) in software design, a situation in which a module 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

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

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

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

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)

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)*

formal design. (1) use of rigorous mathematical methods to specify a software design *(ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary)*

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)*

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)*

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)*

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, software engineering environment

nth of a kind. (1) re-manufacturing or re-installation of a previously verified and validated hardware or software design *(IEEE 1012-2024 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.

vulnerability. (1) potential flaw or weakness in software design or implementation that can 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)*

model pattern. (1) general, reusable model or model part that can be used as a solution to a commonly occurring problem within a given context in system or software design *(ISO/IEC/IEEE 24641:2023. Systems and Software*

engineering--Methods and tools for model-based systems and software engineering, 3.1.21)