1. Glossary

|  |  |
| --- | --- |
| ***acceptance test*** | An acceptance test verifies that the system meets the ***Requirements Specification*** and stipulates the conditions under which the customer will accept the system (Chapter 7). |
| ***activity on node*** | A form of a ***network diagram*** used in a project plan. In the Activity on Node (AON) form, activities are represented by nodes and the dependencies by arrows (Chapter 10). |
| ***activity*** | An activity is a combination of a ***task*** and its associated ***deliverables*** that is part of a project plan (Chapter 10). |
| ***activity view*** | The activity view is part of the ***Unified Modeling Language***. It is characterized by an activity diagram; its ***intention*** is to describe the sequencing of processes required to complete a task (Chapter 6). |
| ***Analytical Hierarchy Process (AHP)*** | A decision-making process that combines both quantitative and qualitative inputs. It is characterized by weighted criteria against which the decision is made, a numeric ranking of alternatives, and computation of a numerical score for each alternative (Appendix B and Chapters 2 and 4). |
| ***artifact*** | System, component, or process that is the end-result of a design (Chapter 2). |
| ***automated script test*** | An automated script test is a sequence of commands given to a unit under test. For example, a test may consist of a sequence of inputs that are provided to the unit, where the outputs for each input are then verified against pre-specified values (Chapter 7). |
| ***baseline requirements*** | The original set of requirements that are developed for a system (Chapter 3). |
| ***black box test*** | A test that is performed without any knowledge of internal workings of the unit under test (Chapter 7). |
| ***bottom-up design*** | An approach to system design where the designer starts with basic components and synthesizes them to achieve the design objectives. This is contrasted to ***top-down*** design (Chapter 5). |
| ***Bohrbug*** | Bohrbugs are reliable ***bugs***, in which the error is always in the same place. This is analogous to the electrons in the Bohr atomic model which assume a definite orbit (Chapter 7). |
| ***brainstorming*** | A freeform approach to concept generation that is often done in groups. This process employs five basic rules: 1) no criticism of ideas, 2) wild ideas are encouraged, 3) quantity is stressed over quality, 4) build upon the ideas of others, and 5) all ideas are recorded (Chapter 4). |
| ***Brainwriting*** | A variation of ***brainstorming*** where a group of people systematically generate ideas and write them down. Ideas are then passed to other team members who must build upon them. |
| ***break-even point*** | The break-even point is the point where the number of units sold is such that there is no profit or loss. It is determined from the total costs and revenue (Chapter 10). |
| ***bug*** | A problem or error in a system that causes it to operate incorrectly (Chapter 7). |
| ***cardinality ratio*** | The cardinality ratio describes the multiplicity of the entities in a relationship. It is applied to ***entity relationship diagrams*** and Unified Modeling Language ***static view diagrams*** (Chapter 6). |
| ***class*** | Classes are used in object-oriented system design. A class defines the attributes and methods (functions) of an ***object*** (Chapter 6). |
| ***cohesion*** | Refers to how focused a module is—highly cohesive systems do one or a few things very well. Also see ***coupling*** (Chapter 5). |
| ***component design specification*** | See ***subsystem design specification*** (Chapter 3). |
| ***concept fan*** | A graphical tree representation of design decisions and potential solutions to a problem. Also see ***concept table*** (Chapters 1 and 4). |
| ***concept generation*** | A phase in the ***design process*** where many potential solutions to solve the problem are identified (Chapter 1). |
| ***concept table*** | A tool for generating concepts to solve a problem. It allows systematic examination of different combinations, arrangements, and substitutions of different elements for a system. Also see ***concept fan*** (Chapter 4). |
| ***conditional rule-based ethics*** | An ethics system in which there are certain conditions under which an individual can break a rule. This is generally because it is believed that the moral good of the situation outweighs the rule. Also see ***rule-based ethics*** (Chapter 11). |
| ***constraint*** | A special type of requirement that encapsulates a design decision imposed by the environment or a stakeholder. Constraints often violate the abstractness property of engineering requirements (Chapter 3). |
| ***controllability*** | A principle that applies to testing. Controllability is the ability to set any node of the system to a prescribed value (Chapter 7). |
| ***copyright*** | Copyrightsprotect published works such as books, articles, music, and software. A copyright means that others cannot distribute copyrighted material without permission of the owner (Chapter 11). |
| ***coupling*** | Modules are coupled if they depend upon each other in some way to operate properly. Coupling is the extent to which modules or subsystems are connected. See also ***cohesion*** (Chapter 5). |
| ***creative design*** | A formal categorization of design projects. Creative designs represent new and innovative designs (Chapter 2). |
| ***critical path*** | The path with the longest duration in a project plan. It represents the minimum time required to complete the project (Chapter 10). |
| ***cross-functional team*** | Cross-functional teams are those that are composed of people from different organizational functions, such as engineering, marketing, and manufacturing. Also see ***multi-disciplinary team*** (Chapter 9). |
| ***data dictionary*** | A dictionary of data contained in a ***data flow diagram***. It contains specific information on the data flows and is defined using a formal language (Chapter 6). |
| ***data flow diagram*** | The ***intention*** of a data flow diagram (DFD) is to model the processing and flow of data inside a system (Chapter 6). |
| ***decision matrix*** | A matrix that is used to evaluate and rank concepts. It integrates both the user-needs and the technical merits of different concepts (Chapter 4). |
| ***derating*** | A decrease in the maximum amount of power that can be dissipated by a device. The amount of derating is based upon operating conditions, notably increases in temperature (Chapter 8). |
| ***deliverable*** | Deliverables are entities that are delivered to the project based upon completion of ***tasks.*** Also see ***activity*** (Chapter 10). |
| ***descriptive design process*** | Describes typical activities involved in realizing designs with less emphasis on exact sequencing than a ***prescriptive design process* (**Chapter 1). |
| ***design architecture*** | The main (Level 1) organization and interconnection of modules in a system (Chapter 5). |
| ***design phase*** | Phase in the ***design process*** where the technical solution is developed, ultimately producing a detailed system design. Upon its completion, all major systems and subsystems are identified and described using an appropriate model (Chapter 1). |
| ***design process*** | The steps required to take an idea from concept to realization of the final system. It is a problem-solving methodology that aims to develop a system that best meets the customer’s need within given constraints (Chapter 1). |
| ***design space*** | The space, or collection, of all possible solutions to a design problem (Chapter 2). |
| ***detailed design*** | A phase in the technical design where the problem can be decomposed no further and the identification of elements such as circuit components, logic gates, or software code takes place (Chapter 5). |
| ***engineering requirement*** | A requirement of the system that applies to the technical aspects of the design. An engineering requirement should be abstract, unambiguous, verifiable, traceable, and realistic (Chapter 3). |
| ***entity relationship diagram (ERD)*** | An ERD is used to model database systems. The ***intention*** of an ERD is to catalog a set of related objects (entities), their attributes, and the relationships between them (Chapter 6). |
| ***entity relationship matrix*** | A matrix that is used to identify relationships between entities in a database system (Chapter 6). |
| ***ethics*** | Philosophy that studies ***morality***, the nature of good and bad, and choices to be made (Chapter 11). |
| ***event*** | An event is an occurrence at a specific time and place that needs to be remembered and taken into consideration in the system design (Chapter 6). |
| ***event table*** | A table that is used to store information about ***events*** in the system. It includes information regarding the event trigger, the source of the event, and process triggered by the event (Chapter 6). |
| ***failure function*** | The failure function, *F(t),* is a mathematical function that provides the probability that a device has failed at time *t* (Chapter 8). |
| ***failure rate*** | The failure rate, *(t),* for a device is the expected number of device failures that will occur per unit time (Chapter 8). |
| ***fixed costs*** | Fixed costs are those that are constant regardless of the number of units produced and cannot be directly charged to a process or activity (Chapter 10). |
| ***float*** | The amount of ***slippage*** that an activity in a project plan can experience without it becoming part of a new ***critical path*** (Chapter 10). |
| ***flowchart*** | A modeling diagram whose intention is to visually describe a process or algorithm, including its steps and control (Chapter 6). |
| ***functional decomposition*** | A design technique in which a system is designed by de­termining its overall functionality and then iteratively decomposing it into component subsys­tems, each with its own functionality (Chapter 5). |
| ***functional requirement*** | A ***subsystem design specification*** that describes the inputs, outputs, and functionality of a system or component (Chapters 3 and 5). |
| ***Gantt chart*** | Gantt charts are a bar graph representation of a project plan where the activities are shown on a timeline (Chapter 10). |
| ***Heisenbugs*** | Heisenbugs are ***bugs*** that are not always reproducible with the same input. This is analogous to the Heisenberg Uncertainty Principle, in which the position of an electron is uncertain (Chapter 7). |
| ***high-performance team*** | A team that significantly outperforms all similar teams. Part of the Katzenbach and Smith team model (Chapter 9). |
| ***integration test*** | An integration test is performed after the units of a system have been constructed and tested. The integration test verifies the operation of the integrated system behavior (Chapter 7). |
| ***intention*** | The intention of a model is the target behavior that it aims to describe (Chapter 6). |
| ***interaction view*** | The interaction view is part of the ***Unified Modeling Language***. Its ***intention*** is to show the interaction between objects. It is characterized by collaboration and sequence diagrams (Chapter 6). |
| ***key attribute*** | An attribute for an entity in a database system that uniquely identifies an instance of the entity (Chapter 6). |
| ***lateral thinking*** | A thought process that attempts to identify creative solutions to a problem. It is not concerned with developing the solution for the problem, or right or wrong solutions. It encourages jumping around be­tween ideas. It is contrasted to ***vertical thinking*** (Chapter 4). |
| ***liable*** | Required to pay monetary damages according to law (Chapter 11). |
| ***marketing requirement (specifications)*** | A statement that describe the needs of the customer or end-user of a system. They are typically stated in language that the customer would use (Chapters 2 and 3). |
| ***maintenance phase*** | Phase in the ***design process*** where the system is maintained, upgraded to add new functionality, or design problems are corrected (Chapter 1). |
| ***matrix test*** | A matrix test is a test that is suited to cases where the inputs submitted are structurally the same and differ only in their values (Chapter 7). |
| ***mean time to failure*** | The mean time to failure (MTTF) is a mathematical quantity which answers the question, “*On average how long does it take for a device to fail?*” (Chapter 8). |
| ***module*** | A block, or subsystem, in a design that performs a function (Chapter 5). |
| ***morals*** | The ***principles*** of right and wrong and the decisions that derive from those principles (Chapter 11). |
| ***multi-disciplinary team*** | In general, a multi-disciplinary team is one in which the members have complementary skills and the team may have representation from multiple technical disciplines. Also see ***cross-functional team*** (Chapter 9). |
| ***negligence*** | Failure to exercise caution, which in the case of design could be in not following reasonable standards and rules that apply to the situation (Chapter 11). |
| ***network diagram*** | A network diagram is a directed graph representation of the activities and dependencies between them for a project (Chapter 10). |
| ***Nominal Group Technique (NGT)*** | A formal approach to brainstorming and meeting facilitation. In NGT, each team member silently generates ideas that are reported out in a round-robin fashion so that all members have an opportunity to present their ideas. Concepts are selected by a multi-voting scheme with each member casting a predetermined number of votes for the ideas. The ideas are then ranked and discussed (Chapters 4 and 9). |
| ***non-disclosure agreement*** | An agreement that prevents the signer from disseminating information about a company’s products, services, and trade secrets (Chapter 11). |
| ***object*** | Objects represent both data (attributes) and the methods (functions) that can act upon data. An object represents a particular instance of a ***class***, which defines the attributes and methods (Chapter 6). |
| ***object type*** | Characteristic of a model used in design. The object type is capable of encapsulating the actual components used to construct the system (Chapter 6). |
| ***objective tree*** | A hierarchical tree representation of the customer’s needs. The branches of the tree are organized based upon functional similarity of the needs (Chapter 2). |
| ***observability*** | This principle applies to testing. Observabilityis the ability to observe any node of a system (Chapter 7). |
| ***over-specificity*** | This refers to applying targets for ***engineering requirements*** that go beyond what is necessary for the system. Over-specificity limits the size of the ***design space*** (Chapter 3). |
| ***pairwise comparison*** | A method of systematically comparing all customer needs against each other. A comparison matrix is used for the comparison and the output is a scoring of each of the needs (Appendix B, Chapter 2, and Chapter 4). |
| ***parallel system*** | A system that contains multiple modules performing the same function where a single module would suffice. The overall system functions correctly when any one of the submodules is functioning (Chapter 8). |
| ***patent*** | A patent is a legal device for protecting a design or invention. If a patent is held for a technology, others cannot use it without permission of the owner (Chapter 11). |
| ***path-complete coverage*** | Path-complete coverage is where every possible ***processing path*** is tested (Chapter 7). |
| ***performance requirement*** | A particular type of ***engineering requirement*** that specifies performance related measures (Chapter 3). |
| ***physical view*** | The physical view is part of the ***Unified Modeling Language***. Its ***intention*** is to demonstrate the physical components of a system and how the logical views map to them. It is characterized by a component and deployment diagram (Chapter 6). |
| ***potential team*** | A team where the sum effort of the team equals that of the individuals working in isolation. Part of the Katzenbach and Smith team model (Chapter 9). |
| ***prescriptive design process*** | An exact process, or systematic recipe, for realizing a system. Prescriptive design processes are often algorithmic in nature and expressed using flowcharts with decision logic (Chapter 1). |
| ***principle*** | Fundamental rules or beliefs that govern behavior, such as the Golden Rule (Chapter 11). |
| ***problem identification*** | The first phase in the design process where the problem is identified, the customer needs identified, and the project feasibility determined (Chapter 1). |
| ***processing path*** | A processing path is a sequence of consecutive instructions or states encountered while performing a computation. They are used to develop test cases (Chapter 7). |
| ***prototyping and construction phase*** | Phase in the ***design process*** in which different elements of the system are constructed and tested. The objective is to model some aspect of the system, demonstrating functionality to be employed in the final realization (Chapter 1). |
| ***pseudo-team*** | An under-performing team where the sum effort of the team is below that of the individuals working in isolation. Part of the Katzenbach and Smith team model (Chapter 9). |
| ***Pugh Concept Selection*** | A technique for comparing design concepts to the user needs. It is an iterative process where concepts are scored relative to the needs. Each concept is combined, improved, or removed from consideration in each iteration of the process (Chapter 4). |
| ***real team*** | A team where the sum effort of the team exceeds that of the individuals working in isolation. Part of the Katzenbach and Smith team model (Chapter 9). |
| ***redundancy*** | A design has redundancy if it contains multiple modules performing the same function where a single module would suffice. Redundancy is used to increase ***reliability*** (Chapter 8). |
| ***reliability*** | Reliability, *R(t)*, is the probability that a device is functioning properly (has not failed) at time *t* (Chapter 8). |
| ***research phase*** | Phase in the ***design process*** where research on the basic engineering and scientific principles, related technologies, and existing solutions for the problem are explored (Chapter 1). |
| ***Requirements Specification*** | A collection of engineering and marketing requirements that a system must satisfy in order for it to meet the needs of the customer or end-user. Alternate terms that are used for the Requirements Specification are the *Product Design**Specification* and the *Systems Requirements Specification* (Chapter 1 and 3). |
| ***reverse-engineering*** | Process where a device or process is taken apart to understand how it works (Chapter 11). |
| ***routine design*** | A formal categorization of design projects. They represent the design of artifacts for which theory and practice are well-developed (Chapter 2). |
| ***rule-based ethics*** | Rule-based ethicsare based upon a set of rules that can be applied to make decisions. In the strictest form, they are considered to be absolute in terms of governing behavior (Chapter 11). |
| ***satisfice*** | Satisfice means that a solution may meet the design requirements, but not be the optimal solution (Chapter 11). |
| ***series system*** | A system in which the failure of a single component (or subsystem) leads to failure of the overall system (Chapter 8). |
| ***situational ethics*** | Situational ethics are where decisions are made based on whether they produce the highest good for the person (Chapter 11). |
| ***slippage*** | Refers to an activity in a project plan taking longer than its planned time to complete. See also ***critical path*** and ***float*** (Chapter 10). |
| ***standards*** | A standard or established way of doing things. Standards ensure that products work together, from home plumbing fixtures to the modules in a modern computer. They ensure the health and safety of products (Chapter 3). |
| ***state*** | The state of a system represents the net effect of all the previous inputs to the system. Since the state characterizes the history of previous inputs, it is often synonymous with the word memory (Chapter 6). |
| ***state diagram (machine)*** | Diagram used to describe systems with memory. It consists of states and transitions between states (Chapter 6). |
| ***static view*** | The static view is part of the ***Unified Modeling Language***. The ***intention*** of the static view is to show the classes in a system and their relationships. The static view is characterized by a class diagram (Chapter 6). |
| ***step-by-step test*** | A step-by-step test case is a prescription for generating a test and checking the results. It is most effective when the test consists of a complex sequence of steps (Chapter 7). |
| ***strengths and weakness analysis*** | A technique for the evaluation of potential solutions to a design problem where the strengths and weaknesses are identified (Chapter 4). |
| ***structure charts*** | Specialized block diagrams for visualizing functional software designs. They employ input, output, transform, coordinate, and composite modules (Chapter 5). |
| ***strict liability*** | A form of ***liability*** that focuses only on the product itself—if the product contains a defect that caused harm, the manufacturer is liable (Chapter 11). |
| ***stub*** | A stub is a device that is used to simulate a subcomponent of a system during testing. Stubs simulate inputs or monitor outputs from the unit under test (Chapter 7). |
| ***subsystem design specification*** | Engineering requirements for subsystems that are constituents of a larger, more complex system (Chapter 3). |
| ***system integration*** | Phase in the ***design process*** where all of the subsystems are brought together to produce a complete working system (Chapter 1). |
| ***task*** | Tasks are actions that accomplish a job as part of a project plan. Also see ***activity*** and ***deliverable*** (Chapter 10). |
| ***Team Process Guidelines*** | Guidelines developed by a team that govern their behavior and identify expectations for performance (Chapter 9). |
| ***technical specification*** | A list of the technical details for a given system, such as operating voltages, processor architecture, and types of memory. The technical specification is fundamentally different from a requirement in that it indicates what was achieved in the end versus what a system needs to achieve from the outset. (Chapter 3). |
| ***test coverage*** | Test coverage is the extent to which the test cases cover all possible ***processing paths*** (Chapter 7). |
| ***test phase*** | Phase in the design process where the system is tested to demonstrate that it meets the requirements (Chapters 1 and 7). |
| ***testable*** | A design is testable when a failure of a component or subsystem can be quickly located. A testable design is easier to debug, manufacture, and service in the field (Chapter 7). |
| ***top-down design*** | An approach to design in which the designer has an overall vision of what the final system must do, and the problem is parti­tioned into components, or subsystems that work together to achieve the overall goal. Then each subsystem is successively refined and partitioned as necessary. This is contrasted to ***bottom-up*** design (Chapter 5). |
| ***tort*** | The basis for which a lawsuit is brought forth (Chapter 11). |
| ***trade secret*** | An approach to protecting intellectual property where the information is held secretly, without ***patent*** protection, so that a competitor cannot access it (Chapter 11). |
| ***under-specificity*** | This refers to a state of the ***Requirements Specification***. When it is under-specified, requirements do not meet the needs of the user and/or embody all of the requirements needed to implement the system (Chapter 3). |
| ***Unified Modeling Language (UML)*** | A modeling language that captures the best practices of object-oriented system design. It encompasses six different system views that can be used to model electrical and computer systems (Chapter 6). |
| ***unit test*** | A unit test is a test of the functionality of a system module in isolation. It establishes that a subsystem performs a single unit of functionality to some specification (Chapter 7). |
| ***use-case view*** | The use-case view is part of the ***Unified Modeling Language***. Its ***intention*** is to capture the overall behavior of the system from the user’s point of view and to describe cases in which the system will be used (Chapter 6). |
| ***utilitarian ethics*** | In utilitarian ethics, decisions are made based upon the decision that brings about the highest good for all, relative to all other decisions (Chapter 11). |
| ***validation*** | The process of determining whether the requirements meet the needs of the user (Chapter 3). |
| ***value*** | A value is something that a person or group believes to be valuable or worthwhile. Also see ***principles*** and ***morals*** (Chapter 11). |
| ***variable costs*** | Variable costs vary depending upon the process or items being produced, and fluctuate directly with the number of units produced (Chapter 10). |
| ***variant design*** | A formal categorization of design projects. They represent the design of existing systems, where the intent is to improve performance or add features (Chapter 2). |
| ***verifiable*** | Refers to a property of an engineering requirement. It means that there should be a way to measure or demonstrate that the requirement is met in the final system realization (Chapter 3). |
| ***vertical thinking*** | A linear, or sequential, thought process that proceeds logically towards the solution of a problem. It seeks to eliminate incorrect solutions. It is contrasted to ***lateral thinking*** (Chapter 4). |
| ***whistleblower*** | A person who goes outside of their company or organization to report an ethical or safety problem (Chapter 11). |
| ***white box test*** | White box tests are those that are conducted with knowledge of the internal working of the unit under test (Chapter 7). |
| ***work breakdown structure*** | The work breakdown structure (WBS) is a hierarchical breakdown of the tasks and deliverables that need to be completed in order to accomplish a project (Chapter 10). |
| ***working group*** | A group of individuals working in isolation, who come together occasionally to share information. Part of the Katzenbach and Smith team model (Chapter 9). |

**Concept Generation**