

Glossary

ability [N-COUNT-U1] An **ability** is the skill to do something. **zdolność, umiejętność**

abstract data type [N-COUNT-U6] An **abstract data type** is a software structure that matches the structure of the original data, and whose system components are designed to maintain themselves. **abstrakcyjny typ danych**

abstraction [N-COUNT-U7] An **abstraction** is a general system plan that ignores details that are irrelevant at a broad level. **abstrakcja**

adaptive maintenance [N-UNCOUNT-U14] **Adaptive maintenance** is the practice of updating software according to changes in environment, such as upgrades to hardware or operating systems, without changing the functionality of the software. **konserwacja adaptacyjna**

address [V-T-U2] To **address** something is to give it appropriate action or attention. **zajmować się (jakąś sprawą), rozwiązywać (problem)**

analysis [N-COUNT-U2] An **analysis** is an examination and report on the structure or functionality of something. **analiza**

anticomposition property [N-COUNT-U13] The **anticomposition property** is a property of test adequacy that states that components that were tested alone should still be tested when assembled as a whole. **aksjomat antykompozycji**

antidecomposition property [N-COUNT-U13] The **antidecomposition property** is a property of test adequacy that states that components that were tested as part of a whole should still be tested alone. **aksjomat antydekompozycji**

antiextensionality property [N-COUNT-U13] The **antiextensionality property** is a property of test adequacy that states that programs need different types of testing even if they have similar names or functions. **aksjomat antyekstensjonalności (nierozszerzalności)**

applicability property [N-COUNT-U13] The **applicability property** is a property of test adequacy that states that an adequate test set exists for every program. **aksjomat stosowalności**

application [N-COUNT-U2] An **application** is the action of putting something into operation or practice. **aplikacja, zastosowanie, użycie**

application framework [N-COUNT-U5] An **application framework** is a semi-finished system that describes the relations between a family of similar systems. **struktura aplikacji, ramy aplikacji**

approach [N-COUNT-U2] An **approach** is a way of dealing with or looking at something. **podejście, metoda, sposób (zrobienia czegoś)**

attribute [N-COUNT-U10] An **attribute** is a fundamental quality of something. **atrybut, cecha**

black-box testing [N-UNCOUNT-U12] **Black-box testing** (also referred as ‘functional or closed-box testing’) is a software testing technique that concentrates on the analysis of software functionality in contrast to internal system mechanisms. **testowanie czarnoskrzynkowe, metoda czarnej skrzynki**

Booch method [N-COUNT-U10] The **Booch method** is a software modeling language and process that is used in object-oriented software development. **metoda Boocha**

bottom-up design [N-COUNT-U8] A **bottom-up design** is a method of designing software in which the engineer begins with the software’s most basic functions and proceeds to more complicated functions, until the higher-order functions of the software have been created. **projektowanie oddolne**

call graph [N-COUNT-U7] A **call graph** is a graph depicting the outcome of a software design process. **graf wywołań**

class [N-COUNT-U10] A **class** is a group of objects that have the same set of attributes. **klasa (kategoria przedmiotów jednakowej jakości)**

class diagram [N-COUNT-U10] A **class diagram** is a graph that models the relationships between nodes and depicts the decomposition of a system. **diagram klas**

closed system [N-COUNT-U3] A **closed system** is a system that does not gain or lose mass. **układ zamknięty, system zamknięty**

cohesion [N-UNCOUNT-U7] **Cohesion** is the connection between the modules of a system. **spójność (oprogramowania)**

collaboration diagram [N-COUNT-U10] A **collaboration diagram** is a graph that depicts objects in relation to a particular interaction. **diagram komunikacji, diagram kolaboracji, diagram współdziałania**

commitment [N-UNCOUNT-U1] **Commitment** is an attitude of strong support of or loyalty to something. **zobowiązanie, poświadczenie, oddanie**

compare [V-T-U11] To **compare** two things is to note the similarities or differences between them. **porównywać**

complexity [N-UNCOUNT-U7] **Complexity** is the measure of the amount of time and other resources required to construct or change a system. **złożoność**

complexity property [N-COUNT-U13] The **complexity property** is a property of test adequacy that states that the complexity of a program directly relates to the complexity required of its test sets. **aksjomat złożoności**

component [N-COUNT-U6] A **component** is a computational element or data store used in software architecture structures. **komponent**

conceptual view [N-COUNT-U5] A **conceptual view** is a way to describe a software system in terms of major design elements and the interactions between those elements. **widok koncepcyjny**

connector [N-COUNT-U6] A **connector** is a computational element that determines how components interact. **łącznik**

consumption [N-UNCOUNT-U3] **Consumption** is the amount of an extensive quantity that is destroyed during a particular period of time. **zużycie**

control structure [N-COUNT-U6] A **control structure** is a component that shows and dictates the order of execution of components. **struktura sterująca**

corrective maintenance [N-UNCOUNT-U14] **Corrective maintenance** is the practice of repairing faults in software systems. **konserwacja naprawcza**

coupling [N-UNCOUNT-U7] **Coupling** is the measure of the strength of connections between modules in a system. **sprzężenie**

coverage-based testing [N-UNCOUNT-U12] **Coverage-based testing** is a model of software testing in which the adequacy of a test is determined by the percentage of the software that is examined. **testowanie oparte na pokryciu, testowanie w oparciu o pokrycie**

critical thinking [N-UNCOUNT-U1] **Critical thinking** is the ability to draw logical conclusions based on facts and evidence. **krytyczne myślenie**

curious [ADJ-U1] If someone is **curious**, he or she wants to know more about something. **ciekawy, zaciekawiony**

customer-driven [ADJ-U4] If software is **customer-driven**, it is developed in response to a clear, specific need of potential customers. **ukierunkowany na potrzeby klienta, zorientowany na klienta**

data flow design [N-COUNT-U9] A **data flow design** is a plan that shows the way data will move through a system. **projekt przepływu danych**

decompose [V-T-U8] To **decompose** a function is to split it into the subfunctions that make it up. **rozkładać, dekomponować**

dedicated [ADJ-U1] If someone is **dedicated**, he or she is enthusiastic about a task or cause. **oddany (czemuś), zaangażowany (w coś)**

demonstration model [N-COUNT-U11] The **demonstration model** is a type of software testing which ensures that software satisfies its intended purpose. **model demonstracyjny**

deployment view [N-COUNT-U5] A **deployment view** is a way to describe a software system in terms of the way software assigns tasks to physical nodes. **widok wdrożenia**

design method [N-COUNT-U8] A **design method** is a set of guidelines and procedures for designing a software system. **metoda projektowania**

design pattern [N-COUNT-U5] A **design pattern** is a reusable solution that can be applied to commonly occurring software design problems. **wzorzec projektowy**

design recovery [N-UNCOUNT-U15] **Design recovery** is the process of creating a program that is identical to an existing program in function but is better organized in abstraction. **odzyskiwanie projektu, odtwarzanie projektu**

destruction model [N-COUNT-U11] The **destruction model** is a type of software testing intended to detect implementation faults in a new piece of software. **model destrukcyjny**

DFD [N-COUNT-U9] A **DFD** (Data Flow Diagram) is a graphical representation of the route that data takes as it moves through a system. **diagram przepływu danych, DPD (lub z ang. DFD)**

DSSA [N-UNCOUNT-U6] **DSSA** (Domain-Specific Software Architecture) is any style of system architecture which includes a reference architecture, component library, and application configuration method. **architektura oprogramowania specyficzna dla domeny**

dynamic analysis [N-UNCOUNT-U12] **Dynamic analysis** is a type of software testing in which a program is executed and the results of this execution are examined. **analiza dynamiczna**

elicitation [N-UNCOUNT-U4] **Elicitation** is the process of causing something to become apparent or realized. **pozyskiwanie (np. wymagań)**

enhance [V-T-U14] To **enhance** something is to improve its function. **ulepszać**

error [N-COUNT-U11] An **error** is a human action that produces an incorrect result in software. **błąd (niezgodność pomiędzy dostarczonym przez funkcję, zaobserwowanym lub zmierzonym rezultatem jej wykonania, a oczekiwana wartością)**

Glossary

error-based testing [N-UNCOUNT-U12] Error-based testing is a software testing technique that detects common errors made by humans. **testowanie oparte na błędach**

evaluation model [N-COUNT-U11] The **evaluation model** is a type of software testing intended to detect requirement, design, and implementation faults. **model oceny, model ewaluacji**

expected [ADJ-U11] If something is **expected**, it is considered likely to happen. **spodziewany, oczekiwany**

expertise [N-UNCOUNT-U1] **Expertise** is extensive or advanced knowledge in a particular subject or area. **wiedza specjalistyczna**

extensive quantity [N-COUNT-U3] An **extensive quantity** is an amount that changes based on the size of a system and has distinct, countable units. **zmienna ekstensywna, wielkość ekstensywna, parametr ekstensywny (zależy od wielkości układu)**

Fagan inspection [N-COUNT-U12] A **Fagan inspection** is a software engineering process that is used to identify defects in ready-to-use software by a group of specialists other than the creator of the software as opposed to simply revealing symptoms.

inspekcja Fagana (proces polegający na próbie znalezienia defektów w dokumentach na różnych etapach procesu tworzenia oprogramowania)

failure [N-COUNT-U11] A **failure** is the observable results of a fault in software. **awaria (niezdolność komponentu lub systemu do wykonania operacji w np. określonym w wymaganiach czasie)**

fault [N-COUNT-U11] A **fault** is the result of an error made by an engineer. **usterka (wada modułu lub systemu, która może spowodować, że moduł lub system nie wykona zadanej czynności)**

fault detection [N-UNCOUNT-U11] **Fault detection** is the process of finding faults and exposing failures in software. **wykrywanie usterek**

fault prevention [N-UNCOUNT-U11] **Fault prevention** is the process of anticipating and stopping problems by testing software multiple times during the development phase. **zapobieganie usterekom**

fault-based testing [N-UNCOUNT-U12] **Fault-based testing** is a software testing technique that focuses primarily on testing for faults. **testowanie oparte na usterkach**

final [ADJ-U3] If something is **final**, it is related to the status of something at the end of a process or period of time. **końcowy**

focus [V-I-U1] To **focus** on something is to watch it closely or give full attention to it. **skupiać się**

functional decomposition [N-UNCOUNT-U8] **Functional decomposition** is a design philosophy in which a function is decomposed into a number of subfunctions. **dekompozycja funkcjonalna**

functional equivalence [N-UNCOUNT-U15] **Functional equivalence** is a measure of how similar two programs are in purpose and function, even though they may be coded differently. **równoważność funkcjonalna**

functional hierarchy [N-COUNT-U4] A **functional hierarchy** is an undefined system used to organize specifications in a requirements document. **hierarchia funkcjonalna**

Fusion method [N-COUNT-U10] The **Fusion method** is an object-oriented software development process that structures the process into analysis, design, and implementation phases. **metoda Fusion**

general multiple change property [N-COUNT-U13] The **general multiple change property** is a property of test adequacy that states that programs with the same structure and dataflow characteristics should still be tested on different criteria. **aksjomat zmiany semantycznej**

generation [N-UNCOUNT-U3] **Generation** is the amount of an extensive quantity that is created during a particular period of time. **generacja**

goal-oriented [ADJ-U1] If someone is **goal-oriented**, he or she works or acts towards a particular purpose. **ukierunkowany na cel, nastawiony na osiągnięcie celu**

idealistic [ADJ-U8] If something is **idealistic**, it assumes the best possible conditions and situations. **idealistyczny**

idiom [N-COUNT-U5] An **idiom** is a low level pattern that is specific to a programming language and can be used to perform a basic function. **idiom**

implementation stage [N-COUNT-U9] The **implementation stage** is a stage in JSD in which a system is transformed from a network of processes to a working design. **etap implementacji, etap wdrażania, etap realizacji**

implementation view [N-COUNT-U5] An **implementation view** is a way to describe a software system in terms of modules of packages and layers. **widok implementacji**

implicit invocation [N-COUNT-U6] An **implicit invocation** is a system in which computations are invoked by certain events rather than by interaction with the user. **wywołanie niejawne**

inadequate empty set property [N-COUNT-U13] The **inadequate empty set property** is a property of test adequacy that states that an empty set is not an adequate test set for any program. **aksjomat niewłaściwości zbioru pustego**

information hiding [N-UNCOUNT-U7] **Information hiding** is a system in which modules contain information that is not likely to change, thereby protecting parts of a program from extensive modifications. **enkapsulacja, ukrywanie informacji, przesłanianie informacji**

initial [ADJ-U3] If something is **initial**, it is related to the status of something at the beginning of a process or period of time. **początkowy, wstępny**

innovative [ADJ-U1] If something is **innovative**, it is new, creative, and advanced. **innowacyjny**

input [N-COUNT-U3] An **input** is the amount of an existing extensive quantity that is added to a system during a particular period of time. **wkład, substancje/materiały wejściowe**

insufficient [ADJ-U14] If something is **insufficient**, it is not suitable or strong enough for a particular purpose. **niewystarczający, niedostateczny**

intensive quantity [N-COUNT-U3] An **intensive quantity** is an amount that does not change based on the size of a system, which can be measured, but cannot be separated into distinct, countable units. **zmienna intensywna, wielkość intensywna, parametr intensywny (nie zależy od wielkości układu)**

interaction diagram [N-COUNT-U10] An **interaction diagram** is a graph that depicts the sequence of messages of which a typical graph is composed. **diagram interakcji**

inter-modular attribute [N-COUNT-U7] An **inter-modular attribute** is a feature or characteristic of an entire system of modules. **atribut międzymodułowy**

intra-modular attribute [N-COUNT-U7] An **intra-modular attribute** is a feature or characteristic of an individual module. **atribut wewnętrzmodułowy**

iteration [N-UNCOUNT-U2] An **iteration** is a new or updated version of a piece of hardware or software. **iteracja**

iterative [ADJ-U2] If something is **iterative**, it is intended to be updated in order to improve or perfect it. **iteracyjny**

JSD [N-UNCOUNT-U9] **JSD** (Jackson System Development) is a method of system development which contains three distinct phases in the development process. **liniowa metoda Jacksona tworzenia oprogramowania, metoda JSD, Jackson System Development**

JSP [N-UNCOUNT-U9] **JSP** (Jackson Structured Programming) is a method of system development that is based on data flow and program structure. **metoda Jacksona programowania strukturalnego, metoda JSP, Jackson Structured Programming**

law of continuing change [N-COUNT-U14] The **law of continuing change** is a principle that states that a system in use should undergo continuing change until it becomes more cost-effective to restructure the system. **prawo ciąglej zmiany**

law of increasing complexity [N-COUNT-U14] The **law of increasing complexity** is a principle that states that a structure becomes more complex with every change that is made to it. **prawo rosnącej złożoności**

layered [ADJ-U6] If an architectural style is **layered**, it is organized by ascending functionality. **warstwowy**

legacy system [N-COUNT-U15] A **legacy system** is a term used to refer to outdated software programs, computer systems or programming languages used rather than updated ones. **system zastany, system istniejący, stary, dotychczasowy system**

logical [ADJ-U1] If something is **logical**, it is based on evidence and reason. **logiczny**

main program with subroutines [N-COUNT-U6] A **main program with subroutines** is a hierarchical system in which a top-level module invokes other modules in a given order. **program główny z podprogramami**

market-driven [ADJ-U4] If software is **market-driven**, it is developed for a broad purpose rather than a specific need. **zorientowany na rynek, ukierunkowany na rynek**

mode [N-COUNT-U4] A **mode** is a changeable system of operation that dictates how software behaves. **tryb (działania)**

modeling stage [N-COUNT-U9] The **modeling stage** is a stage in JSD in which a description is made of the problem that the software needs to solve. **etap modelowania**

modernize [V-T-U15] To **modernize** something is to make it compatible with new technology or update its appearance and functionality. **zmodernizować**

modularity [N-UNCOUNT-U7] **Modularity** is a way of viewing a system as a series of smaller interconnected systems. **modułowość, modularność**

module [N-COUNT-U5] A **module** is a group of software functions that are bundled together. **moduł**

monotonicity property [N-COUNT-U13] The **monotonicity property** is a property of test adequacy criteria that states that additional testing can be performed even after a program has been adequately tested. **aksjomat monotoniczności**

Glossary

network stage [N-COUNT-U9] The **network stage** is a stage in JSD in which a system is shown as a network of communicating processes. **etap sieci**

non-exhausting applicability property [N-COUNT-U13] The **non-exhausting applicability property** is a property of test adequacy criteria that states that a criterion does not require exhaustive testing in all circumstances. **aksjomat niewyczerpującej stosowalności**

object [N-COUNT-U4] An **object** is a physical thing that can be touched and seen. **obiekt**

object-oriented [ADJ-U10] If a design is **object-oriented**, it uses objects, or data structures, as a basis for designing software. **(o programowaniu) zorientowany obiektywnie, obiektowy**

OMT [N-UNCOUNT-U10] The **OMT** (Object Modeling Technique) is an object-oriented approach to software development. **technika modelowania obiektowego**

open system [N-COUNT-U3] An **open system** is a system that allows mass to enter and leave it. **system otwarty, układ otwarty**

oracle [N-COUNT-U11] An **oracle** is a mechanism used to compare predicted results with the actual results of a software test. **wyrocznia**

output [N-COUNT-U3] An **output** is the amount of an extensive quantity that is removed from a system, but not destroyed, during a particular period of time. **uzysk, wydajność, substancje/materiały wyjściowe**

outside the box [ADV PHRASE-U1] If someone thinks **outside the box**, he or she has ideas that are creative or unusual for a particular situation. **twórczo, kreatywnie, niestandardowo**

peer review [N-UNCOUNT-U12] **Peer review** is a practice in which engineers read the programs of other engineers to identify faults or inadequacies in programs. **ocena przez współpracowników, recenzja współpracownika(-ów)**

perfective maintenance [N-UNCOUNT-U14] **Perfective maintenance** is the practice of updating software in order to accommodate new user requirements. **konserwacja udoskonalająca/doskonała/perfekcyjna**

philosophy [N-COUNT-U8] A **philosophy** is a way of understanding or viewing something. **filozofia**

pipes and filters [N-UNCOUNT-U6] **Pipes and filters** is a style that relies on input streams and system operations to process ordered data. **styl "potoki i filtry" ("potoki" służą jako łączniki dla strumienia przekształcanych danych; "filtry" wykonują transformacje danych i przetwarzają otrzymane dane wejściowe)**

prevention model [N-COUNT-U11] The **prevention model** is a type of software testing intended to prevent faults in design, requirements, and implementation. **model prewencyjny**

preventive maintenance [N-UNCOUNT-U14] **Preventive maintenance** is the practice of improving the structure of a system in order to make it easier to maintain. **konserwacja prewencyjna**

primitive [ADJ-U8] If something is **primitive**, it is simple or basic. **prosty, nieskomplikowany**

problem identification [N-UNCOUNT-U2] **Problem identification** is the act of describing and analyzing problems at the first stage of the problem solving process. **identyfikacja problemu**

problem solving [N-UNCOUNT-U2] **Problem solving** is the ability to identify problems, think of solutions, and enact those solutions. **rozwiązywanie problemów**

procedure [N-COUNT-U2] A **procedure** is an established series of actions that dictates how to do something. **procedura**

process view [N-COUNT-U5] A **process view** is a way to describe a software system in terms of the tasks and processes a system performs and the way those tasks and processes interact. **widok procesu**

programming plan [N-COUNT-U5] A **programming plan** is a program fragment that is used to describe a common action. **plan programowania**

proof of correctness [N-UNCOUNT-U12] **Proof of correctness** is a process which formally states a program's specification and proves that the program meets that specification. **dowód poprawności**

property [N-COUNT-U10] A **property** is an identifying and descriptive characteristic or attribute, and may apply to a single entity or a relationship between entities. **właściwość**

rational [ADJ-U8] If a design process is **rational**, it works according to a logical system. **racjonalny**

redefine [V-T-U2] To **redefine** something is to change its function or meaning. **przedefiniować**

redocumentation [N-UNCOUNT-U15] **Redocumentation** is the process of improving or simplifying a program's code without changing its function or level of abstraction. **redokumentacja**

- reengineering** [N-UNCOUNT-U15] **Reengineering**, also called renovation, is the process of making functional changes to a system.
reinżynieria, reengineering, przekonstruowanie, przeprojektowanie
- relationship** [N-COUNT-U10] A **relationship** is a property that depends on the way two entities interact. **relacja, stosunek**
- release** [N-COUNT-U14] A **release** is an updated version of an existing software program. **tu: wersja (oprogramowania)**
- renaming property** [N-COUNT-U13] The **renaming property** is a property of test adequacy that states that two programs that differ only in unimportant ways can be tested with the same test sets. **aksjomat przemianowania**
- renovation** [N-UNCOUNT-U15] **Renovation**, also called reengineering, is the process of making functional changes to a system.
reinżynieria, reengineering, przekonstruowanie, przeprojektowanie
- repair** [V-T-U14] To **repair** something is to fix parts of it that are not functioning correctly. **naprawiać**
- repository** [N-COUNT-U6] A **repository** is an architectural style designed for systems which manage a body of data with an inherent structure. **repozytorium**
- requirements engineering** [N-UNCOUNT-U4] **Requirements engineering** is the practice of creating and documenting requirements for software and other computer systems. **inżynieria wymagań**
- response** [N-COUNT-U4] A **response** is information provided by software upon search or request. **odpowiedź**
- restructuring** [N-UNCOUNT-U15] **Restructuring** is the process of updating a system while keeping the same functionality and level of abstraction. **restrukturyzacja**
- revamping** [N-UNCOUNT-U15] **Revamping** is the process of updating the user interface of a program without changing the program's structure. **modernizacja**
- reverse engineering** [N-UNCOUNT-U15] **Reverse engineering** is the process of analyzing an existing software system and creating a new version of the system at a higher level of abstraction. **inżynieria odwrotna**
- SA** [N-COUNT-U9] A(n) **SA** (Structured Analysis) is a method for converting real-life requirements into software that will fulfill a specific need. **analiza ustukturyzowana**
- satisfy** [V-T-U11] To **satisfy** a requirement is to complete the necessary tasks or meet all of the expectations involved in the requirement. **spełnić (wymaganie)**
- scenario-based evaluation** [N-UNCOUNT-U12] **Scenario-based evaluation** is a model of software testing which is guided by simulations of common use scenarios. **ocena oparta na scenariuszach**
- schematic logic** [N-UNCOUNT-U9] **Schematic logic** is a code that is used in a structure diagram. **schemat logiczny**
- SD** [N-COUNT-U9] A(n) **SD** (Structured Design) is the development of modules and module hierarchies with the goal of creating an optimal module structure. **projektowanie strukturalne**
- sequence diagram** [N-COUNT-U10] A **sequence diagram** is a graph that depicts the time ordering of events within an interaction. **diagram sekwencji, diagram przebiegu**
- simplify** [V-T-U7] To **simplify** something is to eliminate unnecessary elements. **upraszczać**
- software architecture** [N-UNCOUNT-U5] **Software architecture** is the practice of viewing systems in terms of their major components and characterizing the interaction between those components. **architektura oprogramowania**
- software maintenance** [N-UNCOUNT-U14] **Software maintenance** is the process of adapting or modifying a software system to correct faults or improve performance. **konserwacja oprogramowania**
- solution** [N-COUNT-U2] A **solution** is a way of solving or fixing a problem. **rozwiązywanie (np. problemu)**
- specification** [N-COUNT-U4] A **specification** is the precise definition or description of a problem. **specyfikacja, charakterystyka**
- state** [N-COUNT-U10] A **state** is the set of attributes of a particular object. **stan**
- state diagram** [N-COUNT-U10] A **state diagram** is a graph which depicts the dynamic behavior of single objects. **diagram stanów**
- statement coverage property** [N-COUNT-U13] The **statement coverage property** is a property of test adequacy that states that every possible action of a program should be executed by its test sets. **aksjomat pokrycia instrukcji**
- static analysis** [N-UNCOUNT-U12] **Static analysis** is a type of software testing in which a program's structure and components are examined without being executed. **analiza statyczna**

Glossary

- stepwise abstraction** [N-UNCOUNT-U12] **Stepwise abstraction** is a procedure in which a software specialist identifies the prime subprograms of a software, establishes their function and uses this as a basis to determine a function for the whole program. This derived procedure is then used to locate defects by comparing it to the intended one. **abstrakcja krokowa**
- stepwise refinement** [N-UNCOUNT-U8] **Stepwise refinement** is a problem-solving approach in software design in which a problem is divided into smaller, more manageable sections. **uściślanie stopniowe**
- stopping rule** [N-COUNT-U7] A **stopping rule** is an indication that the solution to a problem has been reached. **reguła zatrzymania**
- structure chart** [N-COUNT-U9] A **structure chart** is a chart that shows the functions of a system from the most complex to the most primitive. **schemat strukturalny**
- structure diagram** [N-COUNT-U9] A **structure diagram** is a diagram representing compound components in a structure. **diagram struktury**
- subfunction** [N-COUNT-U8] A **subfunction** is a component which combines with other subfunctions to make up a function. **podfunkcja**
- synthesis** [N-COUNT-U2] A **synthesis** is a combination of multiple items or elements. **syntezą**
- system** [N-COUNT-U3] A **system** is a set of connected things that work together to produce a result. **system**
- system model** [N-COUNT-U6] A **system model** is a description of the characterization of a system as defined by its components and connectors. **model systemu**
- system structure** [N-COUNT-U7] A **system structure** is the makeup of a system's modules and how they are connected. **struktura systemu**
- team player** [N-COUNT-U1] A **team player** is someone who takes actions that benefit a larger group rather than only his or her own interests. **osoba umiejąca pracować w zespole**
- test adequacy criteria** [N-COUNT-U13] **Test adequacy criteria** are sets of requirements that measure the effectiveness of a software testing process. **kryteria adekwatności testów**
- test criterion** [N-COUNT-U11] A **test criterion** is a set standard or qualification by which something is tested. **kryterium testu**
- top-down design** [N-COUNT-U8] A **top-down design** is a method of designing software in which the engineer begins by defining the main user functions and decomposes those functions into subfunctions, until the basic operations of the software are defined. **projektowanie odgórne**
- universal accounting equation** [N-UNCOUNT-U3] The **universal accounting equation** is an equation that is used to measure changes in extensive quantities over particular periods of time. **uniwersalne równanie bilansu**
- unstructured code** [N-UNCOUNT-U14] **Unstructured code** is the code for a system that is designed poorly or coded without a clear structure or order. **kod niestrukturalny**
- user class** [N-COUNT-U4] A **user class** is a distinction that changes the function of software according to the particular user of the software. **klasa użytkownika**
- user-friendly** [ADJ-U4] If something is **user-friendly**, it is easy for most people to understand or use. **przyjazny dla użytkownika**
- validation** [N-UNCOUNT-U4] **Validation** is the process of determining that the requirements of a problem are correct. **walidacja**
- verification** [N-UNCOUNT-U4] **Verification** is the process of determining that a problem's requirements are expressed correctly. **weryfikacja**
- web-based** [ADJ-U15] If something is **web-based**, it is used on the Internet. **internetowy**
- white-box testing** [N-UNCOUNT-U12] **White-box testing** (also referred to as ‘transparent box testing, structural testing, clear box testing or glass box testing’) is a software testing technique that focuses on validating the components of a software, its internal structure and its internal system mechanisms as opposed to the way it functions. **testowanie białoskrzynkowe, metoda białej skrzynki**
- wicked problem** [N-COUNT-U7] A **wicked problem** is a problem encountered in software design that has both a complicated cause and complicated solution, and may be the result of another problem. **zawiły problem (problem trudny lub niemożliwy do rozwiązania)**