Standard glossary of terms used in

Requirements Engineering

Version 1.0

Change History

Version	Date	Remarks
1.0	8/19/2011 6:06:00 PM	First version.

Table of Contents

1. Purpose

The purpose of this document is to provide standardized glossary to be used by IT professionals involved in Requirements Engineering to ensure common understanding of basic terms and activities.

2. Scope

This document presents concepts, terms and definitions related to Requirements Engineering, Business and System Analysis, general Software Engineering and related disciplines.

3. Arrangement

The glossary has been arranged in a single section of definitions ordered alphabetically. Some terms are preferred to other synonymous ones, in which case, the definition of the preferred term appears, with the synonymous ones referring to that.

4. Normative references

At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based upon this Standard are encouraged to investigate the possibility of applying the most recent edition of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

- IEEE Standard 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology
- IEEE Standard 829-1998 IEEE Standard for Software Test Documentation
- IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications
- IEEE Standard 1012-2004: IEEE Standard for Software Verification and Validation
- IEEE Standard 1059-1993: IEEE guide for software verification and validation plans

- IEEE Standard 1220-1998: IEEE Standard for Application and Management of Systems Engineering Process
- IEEE Standard 1233-1998 IEEE Guide for Developing System Requirements Specifications
- IEEE Standard 1362-1998 IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document
- ISO 9000:2005. Quality Management Systems – Fundamentals and Vocabulary.
- ISO/IEC 12207:1995. Information Technology – Software Lifecycle Processes.
- ISO/IEC 14598-1:1999. Information Technology – Software Product Evaluation - Part 1: General Overview.
- ISO 15504-9: 1998. Information Technology – Software Process Assessment – Part 9: Vocabulary
- ISO 31000: Risk Management - Principles and Guidelines on Implementation
- IEC 31010: Risk Management - Risk Assessment Techniques
- ISO/IEC 73: Risk Management – Vocabulary
- ISTQB Glossary of testing terms ver. 2.1

5. Trademarks

In this document the following trademarks are used:

- CMM and CMMI are registered trademarks of Carnegie Mellon University
- BPMN is a registered trademark of Business Process Management Initiative (BPMI), currently merged with Object Management Group
- RUP is a registered trademark of Rational Software Corporation
- TMMI is a registered trademark of TMMi Foundation
- UML is a registered trademark of Object Management Group
- SysML is a registered trademark of Object Management Group

6. Definitions

A

Acceptance criteria: The exit criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity [IEEE 610].

Accuracy: The capability of the software product to provide the right or agreed results or effects with the needed degree of precision [ISO/IEC 25000].

Activity diagram: A graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

Actor: A type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject [OMG].

Ad hoc review: See *Informal review*.

Adaptability: The capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered [ISO/IEC 25000]. See also *Portability*.

Agile Manifesto: A statement on the values that underpin agile software development. The values are:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile software development: A group of software development methodologies based on iterative incremental development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

Agreeing on requirements: see *Requirements acceptance.*

Apprenticing: A process of learning from the customer about his job. The customer teaches the Requirement Engineer – like a master and a student.

Artefact: One of outcomes produced during the development of software. Some artefacts (e.g., use cases, class diagrams, and other UML models, requirements and design documents) help describe the function, architecture, and design of software. Other artefacts are concerned with the process of development itself - such as project plans, business cases, and risk assessments.

Assessment: Activity of determination of quantitative or qualitative value of a product, service, activity, process in regard to given quality or acceptance criteria.

Attractiveness: The capability of the software product to be attractive to the user [ISO/IEC 25000]. See also *Usability*.

Attribute: A characteristic of an object.

Audit: An independent evaluation of software products or processes to ascertain compliance to standards, guidelines, specifications, and/or procedures based on objective criteria, including documents that specify [IEEE 1028]:

- The form or content of the products to be produced.
- The process by which the products shall be produced.
- How compliance to standards or guidelines shall be measured.

Availability: The degree to which a component or system is operational and accessible when required for use. Often expressed as a percentage [IEEE 610].

B

BA: see *Business Analysis, Business Analyst.*

Baseline: A specification or software product that has been formally reviewed or agreed upon, that thereafter serves as the basis for further development, and that can be changed only through a formal change control process [IEEE 610].

Behavioral diagram: In UML a type of diagram that depicts behavioral features of a system or business process. This includes activity, state machine, and use case diagrams as well as the four interaction diagrams. See also *Interaction diagrams.*

Benefit: Value delivered to stakeholders [TGilb].

Business Analysis (BA): The set of tasks, knowledge, tools and techniques required to identify business needs and determine solutions to business problems [BABOK]. See also *System Analysis.*

Business Analyst: A person responsible for identifying the business needs of their clients and stakeholders, to determine solutions to business problems [BABOK]. See also *System Analyst.*

Business domain: (1)The set of classes that represent objects in the business model being implemented. (2) In general – an area of the business being a subject of or impacting the planned solution.

BPMN: see *Business Process Modeling Notation*.

Business Process: A collection of activities designed to produce a specific output for a particular customer or market.

Business Process Modeling Notation (BPMN): A graphical notation that depicts the steps in a business process. BPMN depicts the end to end flow of a business process. The notation has been specifically designed to coordinate the sequence of processes and the messages that flow between different process participants in a related set of activities [BPMN.ORG].

C

Capability Maturity Model (CMM): A five level staged framework that describes the key elements of an effective software process. The Capability Maturity Model covers best practices for planning, engineering and managing software development and maintenance [CMM]. See also: *Capability Maturity Model Integration (CMMI).*

Capability Maturity Model Integration (CMMI): A framework that describes the key elements of an effective product development and maintenance process. The Capability Maturity Model Integration covers best-practices for planning, engineering and managing product development and maintenance. CMMI is the designated successor of the CMM [CMMI]. See also: *Capability Maturity Model (CMM).*

Change Control Board (CCB): See *Configuration Control Board.*

Change Management: (1) A structured approach to transitioning individuals, teams, and organizations from a current state to a desired future state. (2) Controlled way to effect a change, or a proposed change, to a product or service.

Change Request: An official document requesting modification of existing features, requirements or functions or new ones. Change Request should contain description of the current solution, justification for a change and suggested (desired) solution.

Changeability: The capability of the software product to enable specified modifications to be implemented [ISO/IEC 25000]. See also *Maintainability*.

Class: A class describes a set of objects that share the same specifications of features, constraints, and semantics. Class is a kind of classifier whose features are attributes and operations.

Class diagram: A type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

Client: see *Customer.*

Commitment: The degree of obligation of meeting the requirement.

Completeness of a requirement: The degree to which a requirement contains all necessary information.

Complexity: The degree to which a component or system has a design and/or internal structure that is difficult to understand, maintain and verify.

Compliance: The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions [ISO/IEC 25000].

Communication diagram: In UML a diagram that shows instances of classes, their interrelationships, and the message flow between them. For details refer to UML specification [OMG].

Component: A minimal software item that e.g. can be tested in isolation.

Component diagram: In UML a diagram that depicts the components that compose an application, system, or enterprise. For details refer to UML specification [OMG].

Composite Structure diagram: In UML a diagram that depicts the internal structure of a classifier (such as a class, component, or use case), including the interaction points of the classifier to other parts of the system. For details refer to UML specification [OMG].

Conceptual model: A model describing technological software/hardware specifications.

Configuration Control Board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes [IEEE 610].

Consistency: The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a component or system [IEEE 610].

Constraint: A statement of restriction that modifies a requirement or set of requirements by limiting the range of acceptable solutions.

Context: System view from any useful perspective [TGilb].

Context diagram: A diagram that represents the actors outside a system that could interact with that system.

Contractor: see *Vendor.*

Coverage: The degree, expressed as a percentage, to which a specified coverage item has been exercised by a test suite.

Criticality of requirements: Evaluation of the risk of a requirement by evaluating the damage in case of non-fulfillment of a requirement.

Customer: Current or potential buyer or user of the products or service of an individual or organization, called the supplier, seller, or vendor.

D

Data flow diagram: A graphical representation of the sequence and possible changes of the state of data objects, where the state of an object is any of: creation, usage, or destruction.

Decision table: A table showing combinations of inputs and/or stimuli (causes) with their associated outputs and/or actions (effects), which can be used to design test cases..

Defect: A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system.

Defect Management: The process of recognizing, investigating, taking action and disposing of defects. It involves recording defects, classifying them and identifying the impact [IEEE 1044].

Defect Management tool: A tool that facilitates the recording and status tracking of defects and changes. They often have workflow-oriented facilities to track and control the allocation, correction and re-testing of defects and provide reporting facilities.

Deliverable: Any (work) product that must be delivered to someone other than the (work) product's author.

Delphi method: A structured communication technique used to conduct interactive forecasting. It involves a panel of experts [Linstone75].

Dependency: A dependency is a reliance of some kind, of one set of components on another set of components, or one set of requirements or other artifacts on another set [TGilb].

Deployment diagram: In UML a diagram that shows the execution architecture of systems. For details refer to UML specification [OMG].

E

Efficiency: The capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions [ISO/IEC 25000].

Elicitation: The act of obtaining information from other people. In the context of Requirements Engineering, elicitation is the process of gathering requirements from stakeholders.

End user: see *User.*

Entity: (1) An element or set of elements that has a distinct, separate existence, although it need not be a material existence. (2) An abstraction from the complexities of some domain.

Entity-relationship diagram: see *Entity-relationship model.*

ERD: see *Entity-relationship diagram.*

Entity-relationship model: An abstract and conceptual representation of data. Entity-relationship model consists of a set of entities, characterized by attributes and linked by relationships.

ERM: see *Entity-relationship model.*

Error*:* A human action that produces an incorrect result [After IEEE 610].

Estimate: A numeric judgment about a future, present or past level of a scalar system attribute. This includes all performance and cost attributes. Estimates are usually made where direct measurement is: impossible (future), or impractical (past), or uneconomic (current levels) [TGilb].

Extreme Programming: A software engineering methodology used within agile software development whereby core practices are programming in pairs, doing extensive code review, unit testing of all code, and simplicity and clarity in code. See also *Agile software development.*

F

Failure: Deviation of the component or system from its expected delivery, service or result [Fenton].

Failure mode: The physical or functional manifestation of a failure. For example, a system in failure mode may be characterized by slow operation, incorrect outputs, or complete termination of execution [IEEE 610].

Failure Mode and Effect Analysis (FMEA): A systematic approach to risk identification and analysis of identifying possible modes of failure and attempting to prevent their occurrence.

Fault: see *Defect.*

Feature: An attribute of a component or system specified or implied by requirements documentation (for example reliability, usability or design constraints) [IEEE 1008].

FMEA: see *Failure Mode and Effect Analysis.*

Formal review: A review characterized by documented procedures and requirements, e.g. inspection.

Function: A description of "what" a system does. A function has a corresponding implied purpose and is a fundamental part of a system description: a system consists of function attributes, performance attributes, resource (cost) attributes and design attributes. All attributes exist with respect to defined specified conditions. A function can often be decomposed into a hierarchical set of sub-functions [TGilb].

Function point: A unit of measurement to express the amount of business functionality provided by an information system to a user.

Function Point Analysis (FPA): Method aiming to measure the size of the functionality of an information system. The measurement is independent of the technology. This measurement may be

used as a basis for the measurement of productivity, the estimation of the needed resources, and project control.

Functional requirement: A requirement that specifies a function that a component or system must perform [IEEE 610].

Functionality: The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions [ISO/IEC 25000].

G

Goal: A desired state or result of an undertaken. Goals should be measurable and defined in time so that the progress can be monitored.

H

High-level: A position in a hierarchy of defined system components, which is closer to the top than the bottom, relative to the total defined set of those components [TGilb].

Horizontal traceability: The tracing of requirements for a test level through the layers of test documentation (e.g. test plan, test design specification, test case specification and test procedure specification or test script).

I

Impact: Estimated or actual numeric effect of a design idea on a requirement attribute under given conditions.

Incremental development model: A development lifecycle where a project is broken into a series of increments, each of which delivers a portion of the functionality in the overall project requirements. The requirements are prioritized and delivered in priority order in the appropriate increment. In some (but not all) versions of this lifecycle model, each subproject follows a 'mini V-model' with its own design, coding and testing phases.

Informal review: A review not based on a formal (documented) procedure.

Interaction diagram: A subset of behavioral diagrams in UML which emphasize object interactions. This includes communication, interaction overview, sequence, and timing diagrams. See also *Behavioral diagram.*

Interaction overview diagram: A variant of an activity diagram which overviews the control flow within a system or business process.

Inspection: A type of peer review that relies on visual examination of documents to detect defects, e.g. violations of development standards and non-conformance to higher level documentation. The most formal review technique and therefore always based on a documented procedure [IEEE 610, IEEE 1028]. See also *peer review*.

Installability: The capability of the software product to be installed in a specified environment [ISO/IEC 25000]. See also *Portability*.

Integration: The process of combining components or systems into larger assemblies.

Interoperability: The capability of the software product to interact with one or more specified components or systems [ISO/IEC 25000]. See also *Functionality*.

Interview: A conversational technique where the interviewer is asking the responder to obtain information on specified topic.

Iterative development model: A development lifecycle where a project is broken into a usually large number of iterations. An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows from iteration to iteration to become the final product.

L

Learnability: The capability of the software product to enable the user to learn its application [ISO/IEC 25000]. See also *Usability*.

Lifecycle model: A partitioning of the life of a product or project into phases [CMMI].

M

Maintainability: The ease with which a software product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment [ISO/IEC 25000].

Maintenance: Modification of a software product after delivery to correct defects, to improve performance or other attributes, or to adapt the product to a modified environment [IEEE 1219].

Maturity: (1) The capability of an organization with respect to the effectiveness and efficiency of its processes and work practices. See also *Capability Maturity Model* (2) The capability of the software product to avoid failure as a result of defects in the software. [ISO 9126] See also *reliability*.

Maturity level: Degree of process improvement across a predefined set of process areas in which all goals in the set are attained [TMMi].

Maturity model: A structured collection of elements that describe certain aspects of maturity in an organization, and aid in the definition and understanding of an organization's processes. A maturity model often provides a common language, shared vision and framework for prioritizing improvement actions.

Measure: The number or category assigned to an attribute of an entity by making a measurement [ISO 14598].

Measurement: The process of assigning a number or category to an entity to describe an attribute of that entity [ISO 14598].

Metric: A measurement scale and the method used for measurement [ISO 14598].

Milestone: A point in time in a project at which defined (intermediate) deliverables and results should be ready.

Mind-map: A diagram used to represent words, ideas, tasks, or other items linked to and arranged around a central key word or idea. Mind maps are used to generate, visualize, structure, and classify ideas, and as an aid in study, organization, problem solving, decision making, and writing.

Model: A system of assumptions, concepts and relationships between them allowing to describe (model) in an approximate way a specific aspect of reality.

Modeling language: Any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules.

Modeling tool: A tool that supports the creation, amendment and verification of models of the software or system [Graham].

Moderator: The leader and main person responsible for an inspection or other review process.

Module: See *Component*.

MoSCoW: A prioritization technique allowing to prioritize requirements by allocating an appropriate priority expressed in the following terms: Must have, Should have, Could have and Would like to have in the future.

N

Need: Something desired by a defined stakeholder. Satisfying that need would have some value for some stakeholder. A need might not be agreed as a formal requirement, and it might not be prioritized such that it is actually acted upon (designed and implemented). Need is a term often used as a stakeholder view of a problem before requirements specification is carried out [TGilb].

Non-functional requirement: A requirement that does not relate to functionality, but to attributes such as reliability, efficiency, usability, maintainability and portability.

O

Object: In OOAD an instance of a class. See also: *Class, Object-oriented analysis and design.*

Object diagram: In UML a diagram that depicts objects and their relationships at a point in time, typically a special case of either a class diagram or a communication diagram.

Object-oriented analysis and design: A software engineering approach that models a system as a group of interacting objects. Each object represents some entity of interest in the system being modeled, and is characterized by its class, its state (data elements), and its behavior. OOAD encompasses Object-oriented analysis (OOA) and Object-oriented design (OOD). OOA applies object-modeling techniques to analyze the functional requirements for a system. OOD elaborates the analysis models to produce implementation specifications.

OOA: see *Object-oriented analysis and design.*

OOAD: see *Object-oriented analysis and design.*

OOD: see *Object-oriented analysis and design.*

Operability: The capability of the software product to enable the user to operate and control it [ISO/IEC 25000]. See also *Usability*.

P

Pair Programming: A software development approach whereby lines of code (production and/or test) of a component are written by two programmers sitting at a single computer. This implicitly means ongoing real-time code reviews are performed.

Package diagram: In UML a diagram that shows how model elements are organized into packages as well as the dependencies between packages. For details refer to UML specification [OMG].

Peer review: A review of a software work product by colleagues of the producer of the product for the purpose of identifying defects and improvements. Examples are inspection, technical review and walkthrough.

Performance: The degree to which a system or component accomplishes its designated functions within given constraints regarding processing time and throughput rate [IEEE 610]. See also *Efficiency*.

Point of view: A certain perspective on the system or requirements.

Portability: The ease with which the software product can be transferred from one hardware or software environment to another [ISO/IEC 25000].

Prioritization: A process of establishing requirement's implementation relative importance.

Priority: The level of (business) importance assigned to an item.

Process: A set of interrelated activities, which transform inputs into outputs [ISO 12207].

Process assessment: A disciplined evaluation of an organization's software processes against a reference model [ISO 15504].

Process improvement: A program of activities designed to improve the performance and maturity of the organization's processes, and the result of such a program [CMMI].

Process model: (1) A framework wherein processes of the same nature are classified into an overall model, e.g. a test improvement model. (2) A method-independent process description of development processes.

Process requirement: A requirement related to the development process.

Product: An output of a process.

Product requirement: A requirement related to the product of the development process. They affect quality of the product.

Product risk: A risk directly related to the quality of the product. See also *Risk*.

Project: A project is a unique set of coordinated and controlled activities with start and finish dates undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost and resources [ISO 9000].

Project risk: A risk related to management and control of the project. See also *risk*.

Prototype: An early sample or model built to test a concept or process or to act as a thing to be replicated or learned from. In Requirements Engineering prototypes can be used for requirements elicitation and validation.

Q

QA: see *Quality Assurance.*

Quality: The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations [IEEE 610].

Quality Assurance (QA): Part of quality management focused on providing confidence that quality requirements will be fulfilled [ISO 9000].

Quality attribute: A feature or characteristic that affects an item's quality [IEEE 610].

Quality characteristic: See *Quality attribute*.

Quality Management: Coordinated activities to direct and control an organization with regard to quality. Direction and control with regard to quality generally includes the establishment of the quality policy and quality objectives, quality planning, quality control, quality assurance and quality improvement [ISO 9000].

R

Rational Unified Process (RUP): A proprietary adaptable iterative software development process framework consisting of four project lifecycle phases: inception, elaboration, construction and transition.

Recoverability: The capability of the software product to re-establish a specified level of performance and recover the data directly affected in case of failure [ISO/IEC 25000]. See also *Reliability*.

Redundancy: Multiple occurrence of the same information in different places.

Release: A version of the solution released for installation and use by the customer/end users.

Reliability: The ability of the software product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations [ISO/IEC 25000].

Replaceability: The capability of the software product to be used in place of another specified software product for the same purpose in the same environment [ISO/IEC 25000]. See also *Portability*.

Requirement: (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2) [IEEE 610].

RD: see *Requirements Development.*

RE: see *Requirements Engineering.*

RM: see *Requirements Management.*

Requirements acceptance: A process of formal agreement that the content and scope of the requirements are accurate and complete between all relevant stakeholders [BABOK].

Requirements analysis: A set of tasks, activities and tools to determine whether the stated (elicited) requirements are unclear, incomplete, ambiguous, or contradictory, and then documenting the requirements in a form of consistent model.

Requirement attribute: Descriptive information about a requirement that enriches its definition beyond the statement of intended functionality. Examples include origin, rationale, priority, owner, release number, and version number [Wiegers].

Requirements Development (RD): Collection of activities, tasks, techniques and tools to identify, analyze and validate requirements. Includes the process of transforming needs into requirements. In CMMI model, Requirements Development is an engineering process area at Maturity Level 3.

Requirements elicitation: see *Elicitation.*

Requirements Engineering (RE): A sub-discipline of systems engineering and software engineering that is concerned with determining the goals, functions, and constraints of hardware and software systems [Laplante]. Requirements Engineering discipline involves the following sub-processes: requirements elicitation, analysis and negotiation, specification, system modeling, requirements validation and requirements management.

Requirements Management (RM): A continuous process of documenting, analyzing, tracing, prioritizing, communicating, agreeing on requirements and managing requirements' changes. In CMMI model, Requirements Management is a Project Management process area at Maturity Level 2

Requirements Management tool: A tool that supports the recording of requirements, requirements attributes (e.g. priority, knowledge responsible) and annotation, and facilitates traceability through layers of requirements and requirements change management. Some requirements management tools also provide facilities for static analysis, such as consistency checking and violations to pre-defined requirements rules.

Requirements model: A representation of user requirements using text and diagrams. Requirements models can also be called user requirements models or analysis models and can supplement textual requirements specifications.

Requirements phase: The period of time in the software lifecycle during which the requirements for a software product are defined and documented [IEEE 610].

Requirements source: The source from which requirements have been derived. Requirements sources can be stakeholders, documents, business processes, existing systems, market etc.

Requirements specification (customer): A specification describing the problem area. (Customer requirements specification is usually provided by the customer and contains a description of the required capabilities of a solution from the customer's point of view.)

Requirements traceability: The ability to define, capture and follow the traces left by requirements on other elements of the software development environment and the trace left by those elements on requirements [Pinheiro F.A.C. and Goguen J.A].

Requirements Traceability Matrix (RTM): A document, usually in the form of a table, which correlates any two baselined documents that require a many to many relationship to determine the completeness of the relationship.

Review: An evaluation of a product or project status to ascertain discrepancies from planned results and to recommend improvements. Examples include management review, informal review, technical review, inspection, and walkthrough [IEEE 1028].

Reviewer: The person involved in the review that identifies and describes anomalies in the product or project under review. Reviewers can be chosen to represent different viewpoints and roles in the review process.

Risk: (1) The effect of uncertainty on objectives, whether positive or negative [ISO 31000]. (2) A factor that could result in future negative consequences; usually expressed as impact and likelihood.) [ISTQB].

Risk analysis: The process of assessing identified risks to estimate their impact and probability of occurrence (likelihood).

Risk control: The process through which decisions are reached and protective measures are implemented for reducing risks to, or maintaining risks within, specified levels.

Risk identification: The process of identifying risks using techniques such as brainstorming, checklists and failure history.

Risk level: The importance of a risk as defined by its characteristics impact and likelihood. The level of risk can be used to determine the intensity of testing to be performed. A risk level can be expressed either qualitatively (e.g. high, medium, low) or quantitatively.

Risk Management: Systematic application of procedures and practices to the tasks of identifying, analyzing, prioritizing, and controlling risk.

Risk mitigation: See *Risk control*.

RTM: See *Requirements Traceability Matrix.*

RUP: See *Rational Unified Process.*

S

SA: see *System Analysis, System Analyst.*

Safety: The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use [ISO/IEC 25000].

Scalability: The capability of the software product to be upgraded to accommodate increased loads [Gerrard].

Scenario: (1) A projected course of action, events or situations leading to specified result. (2) An ordered sequence of interactions between specified entities (e.g. a system and an actor). (3) In UML: an execution trace of a use case.

Scope: The extent of influence of something. Scope can apply to anything, like a specification, or a specified system or project [TGilb].

Scrum: An iterative incremental framework for managing projects commonly used with agile software development. See also *Agile software development.*

Security: Attributes of software products that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data [ISO/IEC 25000]. See also *Functionality.*

Sequence diagram: In UML it is a structured representation of behavior as a series of sequential steps over time. Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. For details refer to UML specification [OMG].

Signoff: see *Requirements acceptance.*

Software lifecycle: The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software lifecycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. Note these phases may overlap or be performed iteratively.

Software quality: The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs [ISO/IEC 25000].

Software quality characteristic: See *Quality attribute*.

Solution: (1) Solution is the implementation of the requirement. (2) A design idea which, if implemented, is expected to lead to the partial or full satisfaction of a set of attribute requirements; to solve a (defined) problem [TGilb].

Solution model: A model describing the solution area from different views on the system.

Solution specification: also called Functional Specification, System Requirement Specification or Software Requirements Specification. Describes the solution area.

Specification: A document that specifies, ideally in a complete, precise and verifiable manner, the requirements, design, behavior, or other characteristics of a component or system, and, often, the procedures for determining whether these provisions have been satisfied [IEEE 610].

Stability: The capability of the software product to avoid unexpected effects from modifications in the software [ISO/IEC 25000]. See also *Maintainability*.

Stakeholder: Any person who has an interest in an IT project. Project stakeholders are individuals and organizations that are actively involved in the project, or whose interests may be affected as a result of project execution or project completion. Stakeholders can exercise control over both the immediate system operational characteristics, as well as over long-term system lifecycle

considerations (such as portability, lifecycle costs, environmental considerations, and decommissioning of the system) [TGilb].

Standard: Formal, possibly mandatory, set of requirements developed and used to prescribe consistent approaches to the way of working or to provide guidelines (e.g., ISO/IEC standards, IEEE standards, and organizational standards) [CMMI].

State machine: A behavioral model composed of a finite number of states, transitions between those states, and actions, similar to a flow graph.

State transition: A transition between two states of a component or system.

State machine diagram: see *State machine.*

Structure diagram: A type of UML diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object, and package diagrams.

Suitability: The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives [ISO/IEC 25000]. See also *Functionality*.

SysML: see *Systems Modeling Language.*

Systems Modeling Language (SysML): A general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.

System: A collection of components organized to accomplish a specific function or set of functions [IEEE 610].

System Analysis (SA): A set of activities, methods, techniques, tools focused on the translation of the business requirements into systems requirements. It describes a system and its limitations to the environment and provides a well-founded understanding of the environment and the system requirements.

System Analyst: A technically-oriented person, who researches given business problem, plans software solutions, recommends software and systems, and coordinates development to meet business or other requirements. The task of System Analyst is to develop business requirements into system requirements (expresses as technical specifications).

System boundary: The boundary between a system and its context.

T

Testability: The capability of the software product to enable modified software to be tested [ISO/IEC 25000]. See also *Maintainability*.

Testable requirements: The degree to which a requirement is stated in terms that permit establishment of test designs (and subsequently test cases) and execution of tests to determine whether the requirements have been met [IEEE 610].

Timing diagram: In UML a diagram that depicts the change in state or condition of a classifier instance or role over time. For details refer to UML specification [OMG]

Traceability: The ability to identify related items in documentation and software, such as requirements with associated tests. *See also* horizontal traceability, vertical traceability.

U

UML: see *Unified Modeling Language.*

Unified Modeling Language (UML): A standardized general-purpose modeling language in the field of software engineering. UML includes a set of graphic notation techniques to create visual models of software-intensive systems like use case diagrams, activity diagrams, class diagrams and many more.

Understandability: The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use [ISO/IEC 25000]. See also *Usability*.

Usability: The capability of the software to be understood, learned, used and attractive to the user when used under specified conditions [ISO/IEC 25000].

Use case: A sequence of transactions in a dialogue between an actor and a component or system with a tangible result, where an actor can be a user or anything that can exchange information with the system.

Use Case diagram: In UML a diagram that shows use cases, actors, and their interrelationships. For details refer to UML specification [OMG].

User: A person who uses a software product.

V

Value: Perceived benefit , it is the potential consequence of system attributes, for one or more stakeholders. Value is not linearly related to a system improvement: for example, a small change in an attribute level could add immense perceived value for one group of stakeholders for relatively low cost. Value is the perceived usefulness, worth, utility, or importance of a defined system component or system state, for defined stakeholders, under specified conditions. Value is relative to a stakeholder: it is not absolute [TGilb].

Validation: Confirmation by examination and through provision of objective evidence that the requirements for a specific intended use or application have been fulfilled [ISO 9000].

Vendor: A person, group or organization providing the solution.

Verification: Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled [ISO 9000].

Version: A specific form or variation of something.

Vertical traceability: The tracing of requirements through the layers of development documentation to components.

Vision: An image of the project's deliverable as the solution to the stated need or problem.

V-model: A framework to describe the software development lifecycle activities from requirements specification to maintenance. The V-model illustrates how testing activities can be integrated into each phase of the software development lifecycle.

W

Walkthrough: A step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content [Freedman and Weinberg, IEEE 1028]. See also *Peer review*.

Workshop: A kind of meeting focused on specific (previously defined and announced to the participants) topic, usually involving stakeholders representing different areas or/and domains for a short, intensive period.

7. Annex A (Informative)

Index of sources; the following non-normative sources were used in constructing this glossary:

[Beizer] B. Beizer (1990), *Software Testing Techniques*, van Nostrand Reinhold, ISBN 0-442- 20672-0

[BPMN.ORG] Object Management Group/Business Process Management Initiative

[CMM] M. Paulk, C. Weber, B. Curtis and M.B. Chrissis (1995), *The Capability Maturity Model, Guidelines for Improving the Software Process*, Addison-Wesley, ISBN 0-201-54664-7

[CMMI] M.B. Chrissis, M. Konrad and S. Shrum (2004), *CMMI, Guidelines for Process Integration and Product Improvement*, Addison Wesley, ISBN 0-321-15496-7

[Fenton] N. Fenton (1991), *Software Metrics: a Rigorous Approach*, Chapman & Hall, ISBN 0-53249-425-1

[Freedman and Weinberg] D. Freedman and G. Weinberg (1990), *Walkthroughs, Inspections, and Technical Reviews,* Dorset House Publishing, ISBN 0-932633-19-6.

[Gerrard] P. Gerrard and N. Thompson (2002), *Risk-Based E-Business Testing,* Artech House Publishers, ISBN 1-58053-314-0.

[Gilb and Graham] T. Gilb and D. Graham (1993), *Software Inspection*, Addison-Wesley, ISBN 0-201-63181-4.

[Graham] D. Graham, E. van Veenendaal, I. Evans and R. Black (2007), *Foundations of Software Testing, Thomson Learning*, ISBN 978-1-84480-355-2

[Laplante] Laplante, Phil (2009). *Requirements Engineering for Software and Systems (1st ed.*). Redmond, WA: CRC Press. ISBN 1-42006-467-3.

[OMG] OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2.

[Pinheiro F.A.C. and Goguen J.A] Pinheiro F.A.C. and Goguen J.A., *An object-oriented tool for tracing requirements*, in: *IEEE Software* 1996, 13(2), pp. 52-64

[TGilb] see: http://gilb.com, Planguage Concept Glossary