Standard glossary of terms used in

Requirements Engineering

Version 1.4

Change History

Version	Date	Remarks
1.0	8/19/2011	First version.
1,1	20/4/2013	Updated as part of "REQB Approach to Requirements Engineering".
1.2	24/10/2013	Update following changes in REQB Foundation Level syllabus.
1.3	22/01/2014	Internal review and updating terms according to standards
1.4	22/05/2015	Addition the following Agile-specific terms: business value, epic, theme, project stakeholde, system stakeholder, 3C concept, 4C+R criteria, analysis workshop, conditions of satisfaction, implementation story, infrastructure story, spike, emerging requirements, parking lot These terms are marked with [A], standing for Agile

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. 2.2

5. Trademarks

In this document the following trademarks are used:

- BABOK is a registered trademark of IIBA
- 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

0 to 9

3C concept [A]: The three elements that make a User Story: Card, Conversation and Confirmation

4C+R criteria [A]: The goal of the analysis, that at the end of the activity each analyzed requirement will acquire the properties to be:

- Correct: accurate, reflects actual needs
- Clear: only one interpretation, unambiguous
- Consistent: no conflict with other requirements
- Complete: no missing elements
- Relevant: necessary to achieve business goals and objectives

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 product to provide the right or agreed results or effects with the needed degree of precision [after 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 product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the product considered [after 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*.

Analysis Workshop [A]: In agile requirements engineering the key aim of the analysis workshop is to clarify the requirements to the point where they can be accurately estimated during the release planning workshop or the grooming workshops. See also *Workshop*

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

Artifact: One of outcomes produced during the development of software. Some artifacts (e.g., use cases, class diagrams, and other UML models, requirements and design documents) help describe the function, architecture, and design of software. Other artifacts 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 product to be attractive to the user [after ISO/IEC 25000]. See also *Usabilit*y.

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].

BPMN: see *Business Process Modeling Notation*.

Business Analysis (BA): Business Analysis is a discipline providing a set of activities, tools and methods aiming to establish business needs, problems and goals and determine proper solutions allowing to satisfy those needs and resolve given business problems. See also *System Analysis.*

Business Analyst: A person responsible for determining the business needs, expectations and goals of stakeholders, in order to determine solutions to business problems [after BABOK]. See also *System Analyst.*

Business constraint: The limitations on the project's flexibility to implement the requested solution [BABOK].

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.

Business need: see *Need.*

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].

Business Value [A]: The business value is the first criterion for requirement evaluation in the Agile Requirements Engineering Workflow. Business stakeholders evaluate the advantage to the business that will be gained by the implementation of the requirement. This evaluation will be based on the stakeholder's own vision and visibility into the business.

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.

Changeability: The capability of the product to enable specified modifications to be implemented [after 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 product to adhere to standards, conventions or regulations in laws and similar prescriptions [after 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 that represents concepts (entities) and relationships between them. The aim of conceptual modeling is clarifying and expressing the meaning of terms and concepts used by domain experts to address the business problem, and establishing the correct relationships between different concepts.

Condition of Satisfaction [A]: The conditions of satisfaction anticipate and enable the definition of the acceptance criteria, providing the conditions according to which the user story can be considered satisfied from the point of view of the user.

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. See also *Business Constraints, Technical Constraints*

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

Contractor: see *Supplier.*

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.

Customer Requirement: The result of eliciting, consolidating, and resolving conflicts among the needs, expectations, constraints, and interfaces of the product's relevant stakeholders in a way that is acceptable to the customer. [after CMMI]

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 [ISTQB].

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 product to provide appropriate performance, relative to the amount of resources used under stated conditions [after 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.

Emerging Requirement [A]: Requirements, functional and non-functional, that emerge during the development of a product. See also *Requirement*

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.*

Epic [A]: In agile terminology, epics are significantly larger bodies of work. Epics are feature-level work that encompasses many user stories. Epics are almost always delivered over a set of sprints. As a team learns more about an epic through development and customer feedback, user stories will be added and removed as refinement of the epic content.

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 [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.

FPA: See *Function Point Analysis.*

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 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 product to provide functions which meet stated and implied needs when the software is used under specified conditions [after 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.

GUI: Graphical User Interface. A type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.

GUI prototyping: The activity of creating prototypes of GUI of software applications or other products. See also *GUI.*

H

Horizontal prototype: A prototype that provides a broad view of an entire system or subsystem, focusing on user interaction more than low-level system functionality.

Horizontal traceability: Tracing between artifacts on the same level of abstraction (for example, tracing from a requirement to corresponding risk item).

I

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

Implementation Story [A]: In agile terminology, an implementation story is an additional item, conveniently written in the form of a user story, which may be required when new capabilities must be added in order to be able to implement feature items.

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.

Infrastructure Story [A]: In agile terminology, an infrastructure story is an item in the product backlog that is required to create the necessary infrastructure in order for the sprint to achieve the status of "done".

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 product to be installed in a specified environment [after ISO/IEC 25000]. See also *Portability*.

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

Interoperability: The capability of the product to interact with one or more specified components or systems [after 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. 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.

K

Kano model: A model of customer satisfaction dividing product attributes into three categories: threshold, performance, and excitement.

L

Learnability: The capability of the product to enable the user to learn its application [after 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 product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment [after ISO/IEC 25000].

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

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 Will not have this time but will have in the future.

N

Need: Something desired by a defined stakeholder [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.

Objective: see *Goal.*

Object-oriented analysis and design (OOAD): 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 product to enable the user to operate and control it [after ISO/IEC 25000]. See also *Usability*.

P

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].

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.

Parking Lot [A]: In agile terminology, the parking lot is a temporary repository where the emerging requirements are "parked" waiting to be discussed, approved and incorporated into the product backlog.

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 product can be transferred from one hardware or software environment to another [after ISO/IEC 25000].

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

Priority: Is an evaluation of the importance/urgency of a requirement

Problem: Is the description of what a customer whishes to do in order to realize its business processes. The problem describes or helps to describe the needs of a customer.

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 development improvement model. (2) A method-independent process description of development processes.

Product: A product is defined as a composition of software, hardware and other outputs of the production process.

Product Backlog: A set of user stories, requirements or features that have been identified as candidates for potential implementation, prioritized, and estimated [BABOK].

Product requirement: A refinement of customer requirements into the developers' language, making implicit requirements into explicit derived requirements. [after CMMI]

Product risk: Product risks are risks directly related to the quality of the product - potential failure areas (adverse future events or hazards) in the software or system. 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 that surround the project's capability to deliver its objectives.. See also *risk*.

Project Stakeholder [A]: Project stakeholders are those who may have substantial, vested interest in the project that is developing the system. They have a vested interest in the budget and schedule, in the developed product/system/solution and will be involved in marketing, selling, installing, or maintaining the system. See also *Stakeholder*

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 Control (QC): Part of Quality Management focused on fulfilling of quality requirements [ISO 9000].

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 product to re-establish a specified level of performance and recover the data directly affected in case of failure [after 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 product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations [after ISO/IEC 25000].

Replaceability: The capability of the product to be used in place of another specified software product for the same purpose in the same environment [after 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 product, service, product component, or service component to satisfy a supplier agreement, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2) [after CMMI]. See also *Emerging Requirement*

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): A collection of activities, tasks, techniques and tools to identify, analyze, document and validate customer, product and product component/system requirements.

Requirements Elicitation: see *Elicitation.*

Requirements Engineering (RE): Is a sub-discipline of Software Engineering, focused on determining and managing the requirements of hardware and software systems..

Requirements Management (RM): The management of all requirements received by or generated by the project or work group, including both technical and nontechnical requirements as well as those requirements levied on the project or work group by the organization. [after CMMI]

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: A specification describing the business problem area. 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.

Requirement Validation: confirmation by examination that requirements (individually and as a set) define the right system as intended by stakeholders [after ISO/IEC/IEEE 29148]

Requirement Verification: confirmation by examination that a requirement or a set of requirements has been reviewed to ensure the characteristics of good requirements are achieved [after ISO/IEC/IEEE 29148]

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 (assessment): 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. 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 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.*

SMART: A mnemonic, giving criteria to guide in the setting of objectives. SMART stands for:

- Specific – target a specific area for improvement.
- Measurable – quantify or at least suggest an indicator of progress.
- Assignable – specify who will do it.
- Realistic – state what results can realistically be achieved, given available resources.
- Time-related – specify when the result(s) can be achieved.

SME: See *Subject Matter Expert.*

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: A solution is the answer to the needs of a customer (business requirements).

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

Solution scope: All the capabilities, features and qualities that must be delivered by a solution in order to meet the stated business needs or objectives. See also: *Scope.*

Solution specification: A specification describing the business solution area from different points of view. Specific types of a solution specification are: Functional Specification, System Requirement Specification or Software Requirements Specification.

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].

Spike [A]: In agile terminology spikes are a special type of user story that is used to drive out risk and uncertainty in a user story or other project facet.

Sponsor: A stakeholder responsible for contracting or paying for the project.

Stability: The capability of the product to avoid unexpected effects from modifications in the software [after 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. [TGilb]. See also *Project Stakeholder*

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.

Subject Matter Expert (SME): A type of a stakeholder possessing specific experience, knowledge and expertise in a given aspect of the problem domain.

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

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

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.

System Stakeholder [A]: System stakeholders are those that directly use the system, work with the results from those who use the system and/or will be impacted by the deployment and operation of the system. See also *Stakeholder*

T

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

Testability of a requirementa: 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].

Technical constraint: Any restrictions that are related to the architecture of the solution such as hardware and software platforms, programming language or technology, and software that must be used [BABOK].

Theme [A]: In agile terminology, themes may be thought of as groups of related stories. Often the stories all contribute to a common goal or are related in some obvious way, such as all focusing on a single customer. However, while some stories in a theme may be dependent on one another, they do not need to encapsulate a specific work flow or be delivered together. Themes are often related to business goals.

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

UCP: See *Use Case Points.*

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

Usability: The capability of the product to be understood, learned, used and attractive to the user when used under specified conditions [after 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].

Use Case Points (UCP): A software estimation technique used to calculate the estimated effort for a software development project. UCP is best applicable when the Unified Modeling Language (UML) and Rational Unified Process (RUP) methodologies are being used for the software design and development. UCP requires writing requirements for the system in a form of use cases. The software size (UCP) is calculated based on elements of the system use cases with factoring to account for technical and environmental considerations.

User: A person who uses a software product.

User story: User stories are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template: As a <type of user>, I want <some goal> so that <some reason> [after Cohn]. See also: *Implementation Story, Infrastructure Story*

V

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: see *Supplier.*

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 prototype: A prototype that is more complete elaboration of a single subsystem or function.

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.

Vision Statement: A short text stating the why, what, and who of the desired product from a business point of view [after BABOK].

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.

X

XP: See *Extreme Programming*.

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. See also A*nalysis Workshop*

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

[BABOK] A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide), http://www.iiba.org/BABOK-Guide/BABOK-Guide-Online/appendix-a-glossary.aspx

[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

[Cohn] http://www.mountaingoatsoftware.com/agile/user-stories

[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