International Requirements Engineering Board

Martin Glinz

A Glossary of Requirements Engineering Terminology

Caution: This glossary is aligned to the CPRE Foundation Level syllabus 3.0 only!

Version 2.0.0 October 2020

Standard Glossary for the Certified Professional for Requirements Engineering (CPRE) Studies and Exam

University of Zurich UZH

Department of Informatics

Requirements Engineering Research Group

About the Author

Martin Glinz is a full professor emeritus at the University of Zurich (UZH). From 1993 until 2017, he was a professor of Informatics at UZH's Department of Informatics. From 2007-2016, he was the department head. His interests include requirements and software engineering — in particular modeling, validation, quality, and evolution.

He received a diploma degree in Mathematics in 1977 and a Dr. rer. nat. in Computer Science in 1983, both from RWTH Aachen University. Before joining the University of Zurich, he worked in industry for ten years, where he was active in software engineering research, development, training, and consulting. He retired in summer 2017, but he is still active in Requirements Engineering research, education, and service.

Martin Glinz has over 35 years of experience in Requirements Engineering, both academic and industrial. He is on editorial boards and program committees of major journals and conferences in software and requirements engineering and served as general chair, program chair, steering committee chair and organizer for the top international conferences in his field. He is a full member of the International Requirements Engineering Board (IREB), where he chairs the IREB Council. He received the ACM SIGSOFT Distinguished Service Award and the IEEE International Requirements Engineering Conference Lifetime Service Award in 2016 and the IEEE International Requirements Engineering Conference Most Influential Paper Award in 2017.

Terms of Use

Acknowledgements

I gratefully acknowledge the contributions of several people to this glossary. Discussions and joint work with Klaus Pohl, Chris Rupp and Thorsten Weyer shaped several definitions in the first version of this glossary. Karol Frühauf carefully reviewed my drafts of all definitions in version 2.0. Karol's review comments and the subsequent discussions between him and me were valuable sources for improvement.

The alignment of terminology between the glossaries of IREB and ISTQB was achieved in intense discussions between Karol Frühauf and me for IREB and Matthias Hamburg and Armin Born for ISTQB.

Xavier Franch was the IREB Council shepherd for this glossary. He carefully reviewed the final draft and provided feedback that improved the final document in many places.

Many people contributed to the translations of the terminology into languages other than English. Only the translation into German was done by myself.

Release Notes

Version 2.0 of this glossary is aligned with the IREB CPRE Foundation Level version 3.0 and the CPRE Advanced Levels. Users of the CPRE Foundation Level version 2.2 should use the previous version 1.7 of this glossary.

The translations are no longer part of this document. Instead, every translation is published separately as a Dictionary of Requirements Engineering Terminology for the respective language.

Version History

Version 1.1	May 2011: Initial Document
Version 1.1-1	November 2011: French, Spanish, Portuguese (Brazil) dictionaries added
Version 1.2	May 2012: Dutch dictionary added. Version history moved to the top of the document. Credits moved to the top of the document.
Version 1.3	August 2012: Version error fixed. Polish and Swedish dictionaries added.
Version 1.4	September 2012: Italian dictionary added.
Version 1.5	May 2013: Hungarian dictionary added.
Version 1.6	May 2014: Russian dictionary added.
Version 1.7	May 2017: Swedish and Russian dictionaries updated.
Version 2.0.0	October 2020: Major revision and extension of terminology covered by this glossary, including important terms from the CPRE Advanced Levels. Aligned with the terminology used in the CPRE Foundation Level 3.0. Implemented the alignment between the IREB and ISTQB glossaries. Created independent dictionaries of RE terminology for languages other than English.

Table of Contents

Preface

In the preface to the first edition of this glossary, published in May 2011, I wrote:

> *When looking for definitions of terms in Requirements Engineering, one can find definitions for almost any term by searching the web. However, such searching requires effort and the quality of the results is unpredictable. Frequently, definitions found in different sources are inconsistent with each other. Existing glossaries in Requirements Engineering textbooks mostly focus on the topics covered in these books. Systematic translations of terminology into major languages other than English are missing completely.*
>
> *This glossary aims at collecting the existing knowledge on Requirements Engineering terminology and defining the core terminology carefully and consistently. In cases where more than one definition is in use or where terms are defined differently when viewed from different perspectives, multiple definitions or perspectives are included. For terms having both a general meaning and a specific meaning in a Requirements Engineering context, both meanings are defined. Important terms are annotated with hints and additional information.*

This glossary has closed the gap identified above. The principle of not just compiling existing definitions but defining the core Requirements Engineering terminology carefully and consistently, has also stood the test of time. Nevertheless, after almost ten years since its initial publication, it was time for a major revision.

A good glossary should be a stable work product: users need to rely on a common terminology — which is not possible when that terminology is constantly changing. On the other hand, it would be foolish to believe that terminology does not evolve over time. In particular, the major revision of the IREB CPRE Foundation Level syllabus required adaptations and extensions of the terminology. Doing a major revision was also an occasion to include important terms from the IREB CPRE Advanced Level syllabi (which did not yet exist when the glossary was initially published). Finally, IREB and ISTQB, the International Software Testing Qualification Board, had agreed in 2019 to harmonize the quality and testing terminology in their respective glossaries.

From the 128 terms defined in the first edition of the glossary, 42 (i.e., about one third) remained unchanged. 67 definitions underwent minor or merely syntactic changes. We re-wrote 17 definitions, deleted two ones, and added 85 new definitions. Major additions concern terminology about agile, modeling, prototyping, and product lines. We also added several basic terms such as *activity*, *method*, *process*, or *technique*.

Many major changes were due to the harmonization of terminology with ISTQB. However, we also modernized fundamental terms: for example, we simplified the definitions of *requirement* and *Requirements Engineering* and made major changes to the notes in the definition of *system*. The major revision of the glossary was also an occasion to mark explanatory notes clearly in all definitions, separating them from the main definition phrase.

The translations of the terminology into other languages, which were an integral part of the previous versions of this glossary, are now published as separate dictionaries of terminology. I gratefully acknowledge the work performed by all the translators.

Karol Frühauf owes my deepest thanks for carefully reviewing all my definition drafts and for fruitful discussions that led to major improvements of this glossary. I also thank Xavier Franch and Stan Bühne for many helpful comments. Most of all, I thank my wife Angelika. Without her love, patience and understanding, most of my professional work, including this one, would not have been possible.

Martin Glinz

Zurich, October 2020

Definitions of Terms

Terms marked with an asterisk (*) are key terms that have to be known on the IREB CPRE Foundation Level.

Acceptance	The process of assessing whether a ↑system satisfies all its ↑requirements.
Acceptance criteria*	In agile: Criteria that the implementation of a ↑user story must satisfy in order to be accepted by the ↑stakeholders. **Note:** Acceptance criteria may also be written for ↑backlog items other than user stories.
Acceptance test	A test that assesses whether a ↑system satisfies its ↑requirements. **Note:** Typically used by ↑customers to determine whether or not to accept a system.
Activity	An action or a set of actions that a person or group performs to accomplish a ↑task.
Activity model*	A ↑model of the flow of actions in some part of a ↑system.
Activity diagram*	A diagram type in ↑UML which models the flow of actions in some part of a ↑system, including ↑data flows and areas of responsibility where necessary.
Actor*	A person in some ↑role, a ↑system or a technical device in the context of a subject under consideration that interacts with that subject. **Note:** In RE, the subject under consideration typically is a ↑system. In testing, it may be a test ↑object.
Adequacy* (of a requirement)	The degree to which a ↑requirement expresses the ↑stakeholders' true and agreed desires and needs (i.e., those they had actually in mind when stating the requirement).
Agile*	**1.** In general: (a) Able to move quickly and easily. (b) Quick, smart, and clever. **2.** In software development: A development approach which builds a product ↑incrementally by dividing work into ↑iterations of fixed duration (↑timeboxes). **Note:** Agile development is characterized by focusing on delivering a working product in each iteration, collaboration with ↑stakeholders with frequent feedback and adaptation of plans after each iteration based on feedback and changed ↑requirements.
Ambiguity	The contrary of →unambiguity
Application domain*	Those parts of the real world that are relevant for determining the ↑context of a ↑system.

Artifact	Synonym for ↑work product.
Association	In UML: A relationship between two ↑classes in a ↑UML ↑class model.
Attribute*	A characteristic property of an ↑entity or an ↑object.
Backlog	→ Product backlog, → sprint backlog
Baseline*	A stable, change-controlled ↑configuration of ↑work products. **Note:** Baselines serve for ↑release planning and release definition as well as for project management purposes such as effort estimation.
Behavior*	The way in which a ↑system reacts to stimuli, changes its state and produces observable results. **Note:** Stimuli may be events or changes of conditions. Their origin may be external or system-internal.
Behavior model*	A ↑model describing the ↑behavior of a ↑system, e.g., by a ↑state machine.
Branch	A line of ↑configurations or ↑work product ↑versions that forks away from the main line (or from another branch) at some point in time. **Note:** A branch is created by making a copy of some configuration or work product version and making this copy the root of the branch. A branch may be merged with the main line or with another branch at some later point in time.
Bug	→ Defect
Burndown chart	A diagram plotting the work items that remain to accomplish on a time scale.
Business requirement*	A ↑requirement stating a business ↑goal, objective or need of an organization. **Note:** Business requirements typically state those business goals, objectives and needs that shall be achieved by employing a ↑system or a collection of systems.
Cardinality	**1.** In modeling: The minimum and maximum number of ↑objects in a relationship. **2.** In mathematics: The number of elements in a set. **Note:** In ↑UML, the term multiplicity is used for cardinality.
Change control board*	A committee of ↑customer and ↑supplier representatives that decides on ↑change requests. **Abbreviation:** CCB **Note:** The Change control board should not be confused with a *change advisory board*, which is a committee that evaluates change requests for a ↑system in operation and typically has no decision power.

Change management*	A controlled way to effect or deny a requested change of a ↑work product.
Change request*	In RE: A well-argued request for changing one or more ↑baselined ↑requirements.
Changeability	→ Modifiability
Class*	A representation of a set of ↑objects of the same kind by describing the structure of the objects, the ways they can be manipulated and how they behave.
Class diagram*	A diagrammatic representation of a ↑class model.
Class model*	A model consisting of a set of ↑classes and relationships between them.
Commonality	The parts of a ↑product line that are shared by all its members.
Completeness* (of requirements)	**1.** For a single ↑requirement: The degree to which the specification of a requirement is self-contained. **2.** For a ↑work product covering multiple requirements: The degree to which the work product contains all known requirements that are relevant in the scope of this work product.
Compliance*	The adherence of a ↑work product to ↑standards, conventions, regulations, laws, or similar prescriptions.
Component*	**1.** In general: A delimitable part of a ↑system. **2.** In software architecture: An encapsulated set of coherent ↑objects or ↑classes that jointly achieve some purpose. **3.** In testing: A part of a ↑system that can be tested in isolation. **Note:** When viewed in isolation, a component is a ↑system by itself.
Composition (in a technical context)	**1.** An ↑item that is composed of a set of items; forming a whole-part relationship. **2.** The act of composing a whole from a set of parts.
Configuration*	A consistent set of logically coherent ↑items. The items are individually identifiable ↑work products or parts of work products in at most one ↑version per item.
Conflict (about requirements)	→ Requirements conflict
Conformity*	The degree to which a ↑work product conforms to regulations given in some ↑standard.
Consistency* (of requirements)	The degree to which a set of ↑requirements is free of contradicting statements.
Constraint* (in RE)	A ↑requirement that limits the solution space beyond what is necessary for meeting the given ↑functional requirements and ↑quality requirements.

Context*	1. In general: The network of thoughts and meanings needed for understanding phenomena or utterances. 2. Especially in RE: The part of a ↑system's environment being relevant for understanding the system and its ↑requirements. **Note:** Context in the second meaning is also called the ↑system context.
Context boundary*	The boundary between the ↑context of a ↑system and those parts of the ↑application domain that are irrelevant for the ↑system and its ↑requirements. **Note:** The context boundary separates the relevant part of the environment of a system to be developed from the irrelevant part, i.e., the part that does not influence the system to be developed and, thus, does not have to be considered during Requirements Engineering.
Context diagram*	**1.** A diagrammatic representation of a ↑context model. 3. In ↑Structured Analysis, the context diagram is the root of the ↑dataflow diagram hierarchy.
Context model*	A ↑model describing a ↑system in its ↑context.
Control flow	The order in which a set of actions is executed.
Correctness	The degree to which the information contained in a ↑work product is provably true. **Note:** In RE, correctness is sometimes used as a synonym for ↑adequacy, particularly when validating a ↑requirement rigorously against formally stated properties in the ↑context of a ↑system.
Customer*	A person or organization who receives a ↑system, a ↑product or a ↑service. Also see ↑stakeholder.
Customer requirements specification*	A coarse description of the required capabilities of a ↑system from the ↑customer's perspective. **Note:** A customer requirements specification is usually supplied by the ↑customer.
Data flow	A sequence of data items flowing from a producer to a consumer.
Data flow model	A model that describes the ↑functionality of a ↑system by ↑activities, data stores and ↑data flows. **Note:** Incoming data flows trigger activities which then consume the received data, transform them, read/write persistent data held in data stores and then produce new data flows which may be intermediate results that trigger other activities or final results that leave the system.
Data flow diagram	A diagrammatic representation of a ↑data flow model. **Abbreviation:** DFD

Decision table	A tabular representation of a complex decision, specifying which actions to perform for the possible combinations of condition values.
Defect*	An imperfection or deficiency in a ↑work product that impairs its intended use. **Synonyms:** bug, fault
Design*	**1.** A plan or drawing produced to show how something will look, function or be structured before it is made. **2.** The activity of creating a design. **3.** A decorative pattern [This meaning does not apply in the software engineering ↑domain]. **Notes:** 1. In software product development, we distinguish between *creative design* which shapes the look and feel of the product, i.e., its perceivable form, function and quality, and *technical design* (also called software design) which determines the inner structure of the product, in particular the software architecture. 2. The creative design of products is also called *product design*. 3. The creative design of digital solutions is called *digital design*.
Document template	A template providing a predefined skeleton structure for a document. (→ requirements template) **Note:** In RE, document templates can be used to structure ↑requirements documents.
Domain*	A range of relevant things (for some given matter); for example, an ↑application domain.
Domain model*	A ↑model describing phenomena in an ↑application domain. **Notes:** 1. In RE, domain models are created with the intention to understand the ↑application domain in which a planned ↑system will be situated. 2. *Static domain models* specify (business) objects and their relationships in a ↑domain of interest. 3. Domain story models specify visual stories about how actors interact with devices, artifacts, and other items in a ↑domain.
Domain requirement*	A ↑domain property in the ↑context of a ↑system that is required to hold.
Effectiveness*	The degree to which an ↑item produces the intended results. **Note:** In RE, effectiveness frequently is the degree to which a ↑system enables its ↑users to achieve their ↑goals.
Efficiency*	The degree to which resources are expended in relation to results achieved.
Elaboration* (of requirements)	An umbrella term for requirements ↑elicitation, ↑negotiation and ↑validation.
Elicitation* (of requirements)	→ Requirements elicitation

End user	→ User
Entity	**1.** In general: Anything which is perceivable or conceivable (→ item). **2.** In entity-relationship-modeling: an individual ↑item which has an identity and does not depend on another item (→ object).
Entity-relationship diagram	A diagrammatic representation of an ↑entity-relationship model. **Abbreviation:** ERD
Entity-relationship model	A ↑model of data that are relevant for a ↑system or of the data of an ↑application domain, consisting of a set of entity types that are each characterized by ↑attributes and linked by relationships. **Abbreviation:** ER Model
Epic	In agile development: An abstract description of a ↑stakeholder need which is larger than what can be implemented in a single ↑iteration.
Error*	**1.** A human action that produces an incorrect result. **2.** A discrepancy between an observed ↑behavior or result and the specified behavior or result. **Note:** In practice, both meanings are used. Where needed, the meaning of error can be disambiguated by using human error and observed error or observed fault, respectively.
Evolutionary prototype*	A pilot system forming the core of a ↑system to be developed.
Exploratory prototype*	A throwaway ↑prototype used to create shared understanding, clarify ↑requirements or validate requirements.
Fault*	→ Defect
Fault tolerance*	The capability of a ↑system to operate as intended despite the presence of (hardware or software) ↑faults. **Note:** Fault tolerance may be stated as a ↑quality requirement.
Feasibility (of a requirement)	The degree to which a ↑requirement for a ↑system can be implemented under existing ↑constraints.
Feature*	A distinguishing characteristic of a ↑system that provides value for ↑stakeholders. **Note:** A feature typically comprises several ↑requirements and is used for communicating with ↑stakeholders on a higher level of abstraction and for expressing variable or optional characteristics.
Feature diagram	A diagrammatic representation of a ↑feature model.
Feature model	A ↑model describing the variable features of a ↑product line, including their relationships and dependencies.

Form template	A template providing a form with predefined fields to be filled-in. (→ requirements template) **Note:** In RE, form templates can be used to specify ↑use cases or ↑quality requirements.
Functional requirement*	A ↑requirement concerning a result or ↑behavior that shall be provided by a function of a ↑system.
Functionality*	The capabilities of a ↑system as stated by its ↑functional requirements.
Glossary*	A collection of definitions of terms that are relevant in some ↑domain. **Note:** Frequently, a glossary also contains cross-references, ↑synonyms, ↑homonyms, acronyms, and abbreviations.
Goal*	A desired state of affairs (that a ↑stakeholder wants to achieve). **Note:** Goals describe intentions of stakeholders. They may conflict with one another.
Goal model*	A ↑model representing a set ↑goals, sub-goals and the relationships between them. **Note:** Goal models may also include tasks and resources needed to achieve a goal, actors who want to achieve a goal, and obstacles that impede the achievement of a goal.
Homonym	A term looking identical to another term but having a different meaning. **Note:** For example, bill as a bank note and bill as a list (of materials) are homonyms.
Increment* (in software development)	An addition to a ↑system under development that extends, enhances or refactors (↑refactoring) the existing parts of the system. **Note:** In ↑agile development, every ↑iteration produces an increment.
Inspection*	A formal ↑review of a ↑work product by a group of experts according to given criteria, following a defined procedure.
Item	Anything which is perceivable or conceivable. **Synonyms:** entity, object
Iteration*	**1.** In general: The repetition of something, for example, a procedure, a process or a piece of program code. **2.** In agile development: A ↑timeboxed unit of work in which a development team implements an ↑increment to the ↑system under development. **Note:** In agile development, iteration and ↑sprint are frequently used as synonyms.

Kind of requirement*

A classification of requirements according to their kind into ↑*system* requirements (consisting of ↑*functional* requirements, ↑*quality* requirements and ↑*constraints*), *project* requirements, and *process* requirements.

Notes:
1. RE is primarily concerned with system requirements.
2. Quality requirements and constraints are also called ↑*non-functional* requirements.

Language

A structured set of signs for expressing and communicating information.

Note: Signs are any elements that are used for communication: spoken or written words or expressions, symbols, gestures, sounds, etc.

Maintainability

The ease with which a ↑system can be modified by the intended maintainers.

Note: Maintainability may be stated as a ↑quality requirement.

Method

The systematic application of a ↑technique (or a set of techniques) to achieve an objective or create a ↑work product.

Methodology

1. The systematic study of ↑methods in a particular field, in particular, how to select, apply or evaluate methods systematically in a given situation.
2. A set of ↑methods being applied in some combination.

Mock-up*
(of a digital system)

A medium-fidelity ↑prototype that demonstrates characteristics of a user interface without implementing any real ↑functionality.

Note: In RE, a mock-up primarily serves for specifying and validating user interfaces.

Model*

An abstract representation of an existing part of reality or a part of reality to be created.

Notes:
1. The notion of reality includes any conceivable set of elements, phenomena, or concepts, including other models.
2. Models are always built for *specific purposes* in a *specific context*.
3. With respect to a model, the modeled part of reality is called the *original*.
4. In RE, ↑requirements can be specified with models.

Modeling language*

A ↑language for expressing ↑models of a certain kind. May be textual, graphic, symbolic or some combination thereof.

Modifiability*

The degree to which a ↑work product or ↑system can be modified without degrading its ↑quality.

Multiplicity

→ Cardinality

Native prototype*

A high-fidelity ↑prototype that implements critical parts of a ↑system to an extent that ↑stakeholders can use the prototype to see whether the prototyped part of the system will work and behave as expected.

Natural language*	A ↑language that people use for speaking and writing in everyday life. **Note:** This is in contrast to *artificial languages* that people have deliberately created for specific purposes such as programming or specifying.
Necessity* (of a requirement)	The degree to which an individual ↑requirement is a necessary part of the ↑requirements specification of a ↑system.
Negotiation	→ Requirements negotiation
Non-functional requirement*	A ↑quality requirement or a ↑constraint. **Note:** ↑Performance requirements may be regarded as another category of non-functional requirements. In this glossary, performance requirements are considered to be a sub-category of ↑quality requirements.
Object*	**1.** In general: Anything which is perceivable or conceivable (→ item). **2.** In software engineering: an individual ↑item which has an identity, is characterized by the values of its ↑attributes and does not depend on another item (→ entity).
Object diagram	A diagrammatic representation of an ↑object model.
Object model*	A ↑model describing a set of ↑objects and relationships between them.
Performance requirement*	A ↑requirement describing a performance characteristic (timing, speed, volume, capacity, throughput, ...). **Note:** In this glossary, performance requirements are regarded as a sub-category of ↑quality requirements. However, they can also be considered as a ↑kind of requirements of its own.
Persona*	A fictitious character representing a group of ↑users with similar needs, values and habits who are expected to use a ↑system in a similar way.
Phrase template	A template for the syntactic structure of a phrase that expresses an individual ↑requirement or a ↑user story in ↑natural language. (→ requirements template)
Portability	The ease with which a ↑system can be transferred to another platform while preserving its characteristics.
Practice	A proven way of how to carry out certain types of ↑tasks or ↑activities.
Priority*	The level of importance assigned to an ↑item, e.g., a ↑requirement or a ↑defect, according to certain criteria.
Prioritization	The process of assigning priorities to a set of ↑items.
Problem	A difficulty, open question or undesirable condition that needs investigation, consideration, or solution.

Process*	A set of interrelated ↑activities performed in a given order to process information or materials. **Note:** The notion of process includes *business processes* (e.g., how to commission and send ordered goods to ↑customers), *information processes* (e.g., how to deliver records from a database that match a given query), and *technical processes* (e.g., cruise control in a car).
Process model*	A ↑model describing a ↑process or a set of related processes.
Process pattern	An abstract, reusable ↑model of a ↑process which can be used to configure and instantiate a concrete process for a given situation and ↑context.
Product* (in the context of software)	A software-based ↑system or a ↑service provided by a system which is developed and marketed by a ↑supplier and used by ↑customers.
Product backlog*	An ordered, typically prioritized collection of work items that a development team has to work on when developing or evolving a ↑system. **Note:** Items include ↑requirements, ↑defects to be fixed, or ↑refactorings to be done.
Product line	A jointly managed set of systems (provided as products or services) that share a common core and have a configurable set of ↑variants for satisfying needs of particular ↑customers or market segments. **Note:** The points in a product line where there is more than one ↑variant to select from are called ↑variation points. **Synonym:** Product family
Product owner	A person responsible for a ↑product in terms of ↑functionality, value and ↑risk. **Note:** The product owner maintains and prioritizes the ↑product backlog, makes sure that the ↑stakeholders' ↑requirements as well as market needs are elicited and adequately documented in the ↑product backlog and represents the stakeholders when communicating with the development team.
Prototype*	**1.** In manufacturing: A piece which is built prior to the start of mass production. **2.** In software and systems engineering: A preliminary, partial realization of certain characteristics of a ↑system. **3.** In design: A preliminary, partial instance of a design solution. **Notes:** 1. In RE, prototypes are used as a means for requirements ↑elicitation (see ↑specification by example) and ↑validation. 2. Prototypes in RE can be classified (a) with respect to their degree of fidelity into ↑native prototypes, ↑mock-ups and ↑wireframes; (b) with respect to their purpose into ↑exploratory prototypes and ↑evolutionary prototypes.

Prototyping	A ↑process that involves the creation and evaluation of ↑prototypes.
Quality*	**1.** In general: The degree to which a set of inherent characteristics of an item fulfills ↑requirements. **2.** In systems and software engineering: The degree to which a ↑system satisfies stated and implied needs of its ↑stakeholders. **Note:** Quality in this definition means fitness for intended use, as stated in the ↑requirements. This is in contrast to the colloquial notion of quality which is typically connoted with goodness or excellence.
Quality requirement*	A ↑requirement that pertains to a quality concern that is not covered by ↑functional requirements.
Refactoring	The improvement of the internal ↑quality of source code, particularly the structure of the code, without changing its observable behavior.
Redundancy*	Multiple occurrence of the same information or resource.
Release*	A ↑configuration that has been released for installation and use by ↑customers.
Reliability*	The degree to which a ↑system performs specified functions under specified conditions for a specified period of time. **Note:** Reliability may be stated as a ↑quality requirement.
Requirement*	**1.** A need perceived by a ↑stakeholder. **2.** A capability or property that a ↑system shall have. **3.** A documented representation of a need, capability or property.
Requirements analysis	**1.** Analysis of elicited ↑requirements in order to understand and document them. **2.** Synonym for ↑Requirements Engineering.
Requirements baseline	A ↑baseline for a set of ↑requirements.
Requirements branching	→ Branch
Requirements configuration	→ Configuration
Requirements conflict*	**1.** A situation where two or more ↑requirements cannot be satisfied together. **2.** A situation where two or more ↑stakeholders disagree about certain ↑requirements. **Note:** Requirements conflicts have to be solved by ↑requirements negotiation.
Requirements discovery	→ Requirements elicitation
Requirements document*	A document consisting of a ↑requirements specification. **Note:** Requirements document is frequently used as a synonym for requirements specification.

Requirements elicitation*	The process of seeking, capturing and consolidating ↑requirements from available ↑sources, potentially including the re-construction or creation of requirements.
Requirements Engineer*	A person who – in collaboration with ↑stakeholders – elicits, documents, validates, and manages ↑requirements. **Note:** In most cases, requirements engineer is a ↑role and not a job title.
Requirements Engineering*	The systematic and disciplined approach to the ↑specification and management of ↑requirements with the goal of understanding the ↑stakeholders' desires and needs and minimizing the risk of delivering a ↑system that does not meet these desires and needs. **Abbreviation:** RE
Requirements management*	The process of managing existing ↑requirements and requirements-related ↑work products, including the storing, changing and tracing of requirements (↑traceability).
Requirements model	A ↑model that has been created with the purpose of specifying ↑requirements.
Requirements negotiation*	A ↑process where ↑stakeholders are working toward reaching an agreement to resolve ↑requirements conflicts.
Requirements source*	The source from which a ↑requirement has been derived. **Note:** Typical sources are ↑stakeholders, documents, existing ↑systems and observations.
Requirements specification*	A systematically represented collection of ↑requirements, typically for a ↑system, that satisfies given criteria. **Notes:** 1. In some situations we distinguish between a ↑*customer* requirements specification (typically written by the ↑customer) and a ↑*system* requirements specification or ↑*software* requirements specification (written by the supplier). 2. Requirements specification may also denote the ↑*process* of specifying (↑eliciting, documenting and ↑validating) requirements.
Requirements template*	A template for specifying ↑requirements. **Note:** In RE, several forms of templates are used. ↑*Phrase templates* are used for specifying individual ↑requirements or ↑user stories. ↑*Form templates* can be used to specify ↑use cases or ↑quality requirements. ↑*Document templates* provide a predefined structure for ↑requirements documents.
Review*	An evaluation of a ↑work product by an individual or a group in order to find problems or suggest improvements. **Note:** Evaluation may be performed with respect to both contents and conformance.

Risk*	A possible event that threatens the success of an endeavor. **Note:** A risk is typically assessed in terms of its probability and potential damage.
Role*	**1.** A part played by a person in a given context. **2.** In ↑UML ↑class models: The parts played by the linked ↑objects in an ↑association.
Safety*	The capability of a ↑system to achieve an acceptable level of probability that the system, under defined conditions, will not reach a state in which human life, health, property, or the environment is endangered. **Note:** Safety ↑requirements may be stated as ↑quality requirements or in terms of ↑functional requirements.
Scenario	**1.** In general: A description of a potential sequence of events that lead to a desired (or unwanted) result. **2.** In RE: An ordered sequence of interactions between partners, in particular between a ↑system and external ↑actors. May be a concrete sequence (instance scenario) or a set of potential sequences (type scenario, ↑use case).
Scope* (of a system development)	The range of things that can be shaped and designed when developing a ↑system.
Scrum	A popular ↑process framework for ↑agile development of a ↑system.
Security*	The degree to which a ↑system protects its data and resources against unauthorized access or use and secures unobstructed access and use for its legitimate ↑users. **Note:** Security requirements may be stated as ↑quality requirements or in terms of ↑functional requirements.
Semantics*	The meaning of a sign or a set of signs in a ↑language.
Semi-formal	Something which is formal to some extent, but not completely. **Note:** A ↑work product is called semi-formal if it contains formal parts, but isn't formalized totally. Typically, a semi-formal work product has a defined ↑syntax, while the ↑semantics is partially defined only.
Sequence diagram*	A diagram type in ↑UML which models the interactions between a selected set of ↑objects and/or ↑actors in the sequential order in which those interactions occur.
Service*	The provision of some ↑functionality to a human or a ↑system by a provider (a system, organization, group or individual) that delivers value to the receiver. **Note:** In systems engineering, software engineering and Requirements Engineering, services are typically provided by a ↑system for a ↑user or another system.

Software requirements specification*	A ↑requirements specification pertaining to a software ↑system. **Abbreviation:** SRS
Source (of a requirement)	→ Requirements source
Specification*	**1.** As a work product: A systematically represented description of the properties of an ↑item (a ↑system, a device, etc.) that satisfies given criteria. **2.** As a process: the process of specifying (↑eliciting, documenting and ↑validating) the properties of an ↑item. **Note:** A specification may be about required properties (↑requirements specification) or implemented properties (e.g., a technical product specification).
Specification by example	A ↑technique that specifies test cases and ↑requirements for a ↑system by providing examples of how the system should behave.
Specification language	An artificial ↑language that has been created for expressing ↑specifications.
Spike	In agile development: A task aimed at gaining insight or gathering information, rather than at producing a ↑product ↑increment.
Sprint*	An ↑iteration in ↑agile development, particularly when using ↑Scrum.
Sprint backlog	A set of ↑product backlog items that have been selected to be implemented in the current ↑sprint.
Stakeholder*	A person or organization who influences a ↑system's ↑requirements or who is impacted by that system. **Note:** Influence can also be indirect. For example, some stakeholders may have to follow instructions issued by their managers or organizations.
Stakeholder requirement*	A ↑requirement expressing a ↑stakeholder desire or need. **Note:** Stakeholder requirements are typically written by stakeholders and express their desires and needs from their perspective.
Standard*	A formal, possibly mandatory set of regulations for how to interpret, develop, manufacture, or execute something. **Note:** In RE, there are RE-relevant standards issued by ISO/IEC and IEEE.
State machine*	A ↑model describing the behavior of a ↑system by a finite set of *states* and state *transitions*. State transitions are triggered by *events* and can in turn trigger *actions* and new events.
State machine diagram*	A diagrammatic representation of a ↑state machine.
State-transition diagram	→ State machine diagram
Statechart*	A ↑state machine having states that are hierarchically and/or orthogonally decomposed.

Steering committee	A committee that supervises a project.
Story (in an RE context)	→ User story
Storyboard	A series of sketches or pictures that visualize the execution of a ↑scenario.
Story map	A two-dimensional arrangement of ↑user stories. **Note:** A story map helps understand the ↑functionality of a ↑system, identify gaps and plan releases.
Structured Analysis	An approach for specifying the ↑functionality of a system based on a hierarchy of ↑data flow diagrams. Data flows as well as persistent data are defined in a data dictionary. A ↑context diagram models the sources of incoming and the destinations of outgoing ↑data flows.
Supplier	A person or organization who delivers a ↑product or ↑service to a ↑customer.
Synonym*	A word having the same meaning as another word.
Syntax*	The rules for constructing structured signs in a ↑language.
System*	**1.** In general: A principle for ordering and structuring. **2.** In engineering: A coherent, delimitable set of elements that – by coordinated action – achieve some purpose. **Notes:** 1. A system may comprise other systems or ↑components as sub-systems. 2. The purposes achieved by a system may be delivered by • deploying the system at the place(s) where it is used, • selling/providing the system as a ↑product to its ↑users, • having providers who offer the system's capabilities as ↑services to users. 3. Systems containing both software and physical ↑components are called *cyber-physical systems*. 4. Systems spanning software, hardware, people and organizational aspects are called *socio-technical systems*. **Important:** In all definitions referring to system in this glossary, system is an umbrella term which includes • ↑*Products* provided to ↑customers, • ↑*Services* made available to ↑customers, • Other work products such as *devices*, *procedures* or *tools* that help people or organizations achieve some goal, • System ↑*components* or ↑*compositions* of systems.

System boundary*	The boundary between a ↑system and its surrounding ↑context. **Notes:** 1. The system boundary delimits the system as it shall be after its implementation and deployment. 2. At the system boundary, the external interfaces between the ↑system and its ↑context have to be defined. 3. The system boundary frequently coincides with the ↑scope of a ↑system (which denotes the range of things that can be shaped and designed). However, this is not always the case: there may be components within the system boundary that have to be re-used as they are (i.e., cannot be shaped nor designed), while in the system context there may be things that can be re-designed when the system is developed (which means that they are in scope).
System context	The part of a ↑system's environment that is relevant for the definition as well as the understanding of the ↑requirements of a ↑system to be developed.
System requirement*	A ↑requirement pertaining to a ↑system.
System requirements specification*	A ↑requirements specification pertaining to a ↑system. **Note:** A system requirements specification is frequently considered to be a synonym for ↑requirements specification. **Abbreviation:** SyRS
Task	A coherent chunk of work to be done.
Technique	A documented set of coherent actions for accomplishing a ↑task or achieving an objective.
Theme	In agile development: A collection of related ↑user stories.
Timebox	A fixed, non-extendable amount of time for completing a set of ↑tasks.
Tool* (in software engineering)	A (software) ↑system that helps develop, operate and maintain systems. **Note:** In RE, tools support ↑requirements management as well as modeling, documenting, and validating ↑requirements.
Traceability*	**1.** In general: The ability to establish explicit relationships between related ↑work products or ↑items within work products. **2.** In RE: The ability to trace a ↑requirement (a) back to its origins, (b) forward to its implementation in design and code and its associated tests, (c) to requirements it depends on (and vice-versa).
UML*	Abbreviation for Unified Modeling Language, a standardized language for modeling problems or solutions.
Unambiguity* (of requirements)	The degree to which a ↑requirement is expressed such that it cannot be understood differently by different people.

Understandability*	The degree to which an ↑item is comprehensible to its intended users. **Note:** Typical items are: a ↑system, a ↑work product, or a part thereof.
Usability*	The degree to which a ↑system can be used by specified ↑users to achieve specified ↑goals in a specified context of use. **Note:** Usability particularly includes the capability of a ↑system to be understood, learned, used, and liked by its intended ↑users.
Use case*	A set of possible interactions between external ↑actors and a ↑system that provide a benefit for the actor(s) involved. **Note:** Use cases specify a system from a user's (or other external actor's) perspective: every use case describes some ↑functionality that the system must provide for the actors involved in the use case.
Use case diagram*	A diagram type in ↑UML that models the ↑actors and the ↑use cases of a ↑system. **Note:** The boundary between the actors and the use cases constitutes the ↑system boundary.
Use case model	A ↑model consisting of a set of ↑use cases, typically together with a ↑use case diagram.
User*	A person who uses the ↑functionality provided by a ↑system. **Note:** Users (also called end users) always are ↑stakeholders of a ↑system.
User requirement*	A ↑requirement expressing a ↑user need. **Note:** User requirements are typically about what a system should do for certain users and how they can interact with the system. User requirements are a subset of ↑stakeholder requirements.
User story*	A description of a need from a ↑user's perspective together with the expected benefit when this need is satisfied. **Notes:** 1. User stories are typically written in ↑natural language using a ↑phrase template and are accompanied by ↑acceptance criteria. 2. In ↑agile development, user stories are the main means for communicating needs between a ↑product owner and the development team.
Validation*	The ↑process of confirming that an ↑item (a ↑system, a ↑work product or a part thereof) matches its ↑stakeholders' needs. **Note:** In RE, validation is the process of confirming that the documented ↑requirements match their ↑stakeholders' needs; in other words: whether the right requirements have been specified.
Variability	**1.** The degree to which a ↑system can be changed or customized. **2.** In product lines: The ↑features that can differ among the members of the ↑product line.

Variant	One of the possible forms that an ↑item (e.g., a ↑requirement) may have.
Variation point	A point in a ↑product line where an element of the product line (typically a variable or a ↑feature) can be chosen from a set of ↑variants.
Verifiability* (of requirements)	The degree to which the fulfillment of a ↑requirement by an implemented ↑system can be verified. **Note:** Such ↑verification can be performed, for example, by defining ↑acceptance test cases, measurements or ↑inspection procedures.
Verification	The process of confirming that an ↑item (a system, a work product, or a part thereof) fulfills its ↑specification. **Note:** Requirements verification is the process of confirming that the ↑requirements have been documented properly and satisfy the ↑quality criteria for requirements; in other words, whether the requirements have been specified right.
Version*	An occurrence of an ↑item which exists in multiple, time-ordered occurrences where each occurrence has been created by modifying one of its previous occurrences.
View*	An excerpt from a ↑work product, containing only those parts one is currently interested in. **Note:** A view can abstract or aggregate parts of the work product.
Viewpoint	A certain perspective on the ↑requirements of a ↑system. **Note:** Typical viewpoints are perspectives that a ↑stakeholder or stakeholder group has (for example, an end user's perspective or an operator's perspective). However, there can also be topical viewpoints such as a security viewpoint.
Vision* (for a system or product)	A conceptual imagination of a future ↑system or ↑product, describing its key characteristics and how it will create value for its ↑users.
Walkthrough*	A ↑review in which the author of a ↑work product leads the reviewers systematically through the work product and the reviewers ask questions and make comments about possible issues.
Wireframe*	A low-fidelity ↑prototype built with simple materials that primarily serves for discussing and validating requirements, design ideas or user interface concepts. **Note:** When prototyping digital systems, wireframes are typically built with paper. Such prototypes are also called *paper prototypes*.
Work product*	A recorded, intermediate or final result generated in a work ↑process. **Synonym:** ↑Artifact

List of Abbreviations

CCB	**C**hange **c**ontrol **b**oard
CPRE	**C**ertified **P**rofessional for **R**equirements **E**ngineering
DFD	**D**ata **f**low **d**iagram
ER	**E**ntity-**r**elationship
ERD	**E**ntity-**r**elationship **d**iagram
IREB	**I**nternational **R**equirements **E**ngineering **B**oard
RE	**R**equirements **E**ngineering
SRS	**S**oftware **r**equirements **s**pecification
SyRS	**Sy**stem **r**equirements ***s***pecification
UML	**U**nified **M**odeling **L**anguage

Sources

I don't cite sources for individual definitions because I deliberately decided not to compile definitions from various existing sources just by copy-paste, but to carefully re-formulate all definitions consistently and according to today's use.

Several definitions are based on my own work [Gl07], [GlWi07], [Gl19]. Most definitions from the agile domain have been taken from or adapted from the IREB RE@Agile Glossary, which was joint work of the RE@Agile working group and me. The revision of the IREB CPRE Foundation Level syllabus [IREB20] also informed several new or changed definitions.

I consulted numerous international standards when writing the definitions [IEEE610], [IEEE730], [IEEE830], [IEEE1012], [IEEE1028], [ISO9000], [ISO12207], [ISO19770], [ISO20246], [ISO24765], [ISO25000], [ISO25010], [ISO26550], [ISO29148], [ISO42010]. However, as the terminology defined or used in these standards is frequently inconsistent or inadequate for a Requirements Engineering glossary, I did not copy any definitions verbatim from these standards.

Other sources that influenced some definitions are [GaWe89], [My06], [Po10], [St73], and [ZoCo05].

For cross-checking, I also consulted the Merriam-Webster online dictionary (https://www.merriam-webster.com) and Wikipedia (https://en.wikipedia.org).

Below I want to give credit for some definitions that I have taken more or less verbatim from a source or that are joint work with others. The copyright for cited definitions lies with the authors of the cited work. The copyright for joint work lies jointly with the author of this glossary and the persons mentioned.

Term	**Reference**
Context boundary	Joint work with Klaus Pohl, Chris Rupp, and Thorsten Weyer, based on [Po10], [PoRu11] and [We10]
Functional requirement	Joint work with Klaus Pohl, Chris Rupp, and Thorsten Weyer
Model	Joint work with Klaus Pohl and Chris Rupp, based on [PoRu11]
Quality requirement	Joint work with Klaus Pohl, Chris Rupp, and Thorsten Weyer, based on definitions in my course notes on Requirements Engineering I
Requirements Engineering	Definition is a simplification of a definition that was joint work with Klaus Pohl, Chris Rupp, and Thorsten Weyer
Requirements specification	Adapted from Pohl and Rupp [PoRu11]
System boundary	Joint work with Klaus Pohl, Chris Rupp, and Thorsten Weyer based on [Po10], [PoRu11]
System context	Joint work with Klaus Pohl, Chris Rupp, and Thorsten Weyer based on [Po10], [PoRu11], [We10]

References

[GaWe89] Donald C. Gause and Gerald M. Weinberg (1989). *Exploring Requirements: Quality before Design*. New York: Dorset House.

[Gl07] Martin Glinz (2007). On Non-Functional Requirements. *15th IEEE International Requirements Engineering Conference (RE'07)*, Delhi, India. 21-26.

[GlWi07] Martin Glinz and Roel Wieringa (2007). Stakeholders in Requirements Engineering (Guest Editors' Introduction). *IEEE Software* 24(2):18-20.

[Gl19] Martin Glinz (2019). *Requirements Engineering I*. Course Notes, University of Zurich. https://www.ifi.uzh.ch/en/rerg/courses/hs19/re-i.html#resources. Last visited August 2020.

[IEEE610] *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990.

[IEEE730] *IEEE Standard for Software Quality Assurance Processes*. IEEE Std 730-2014.

[IEEE830] *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std 830-1998.

[IEEE1012] *IEEE Standard for System, Software, and Hardware Verification and Validation*. IEEE Std 1012-2016.

[IEEE1028] *IEEE Standard for Software Reviews and Audits*. IEEE Std 1028-2008.

[IREB20] IREB (2020). *Certified Professional for Requirements Engineering – Foundation Level – Syllabus, Version 3.0*. https://www.ireb.org/en/downloads/#cpre-foundation-level-syllabus-3-0**.** Last visited September 2020.

[ISO9000] *Quality Management Systems — Fundamentals and Vocabulary*. ISO Standard 9000:2015.

[ISO12207] *Systems and Software Engineering — Software Life Cycle Processes*. ISO/IEC/IEEE Standard 12207:2017.

[ISO19770] *Information Technology — IT Asset Management — Part 1: IT Asset Management Systems — Requirements*. ISO/IEC Standard 19770-1:2017.

[ISO20246] Software and Systems Engineering — Work Product Reviews. ISO/IEC Standard 20246:2017

[ISO24765] *Systems and Software Engineering — Vocabulary*. ISO/IEC/IEEE Standard 24765:2017.

[ISO25000] *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*. ISO/IEC Standard 25000:2014.

[ISO25010] *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models*. ISO/IEC Standard 25010:2011.

[ISO26550] *Software and Systems Engineering — Reference Model for Product Line Engineering and Management*. ISO/IEC Standard 26550:2015.

[ISO29148]	*Systems and Software Engineering — Life Cycle Processes — Requirements Engineering*. ISO/IEC/IEEE Standard 29148:2018.
[ISO42010]	*Systems and Software Engineering — Recommended Practice for Architectural Description of Software-Intensive Systems*. ISO/IEC Standard 42010:2007.
[My06]	John Mylopoulos (2006). *Goal-Oriented Requirements Engineering: Part II.* Presentation slides of keynote talk at the 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis, USA.
[Po10]	Klaus Pohl (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques*. Berlin-Heidelberg: Springer.
[PoRu11]	Klaus Pohl, Chris Rupp (2011). *Requirements Engineering Fundamentals*. Santa Barbara, Ca.: RockyNook.
[St73]	Herbert Stachowiak (1973). *Allgemeine Modelltheorie*. (in German) Wien: Springer.
[We10]	Thorsten Weyer (2010). *Kohärenzprüfung von Verhaltensspezifikationen gegen spezifische Eigenschaften des operationellen Kontexts* (in German). PhD Dissertation, University of Duisburg-Essen.
[ZoCo05]	Didar Zowghi and Chad Coulin (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum, C. Wohlin (eds.): *Engineering and Managing Software Requirements*. Berlin: Springer. 19–46.