IREB International Requirements Engineering Board

rockynook

> COMPUTING

Klaus Pohl · Chris Rupp

Requirements Engineering Fundamentals

A Study Guide for the
Certified Professional for Requirements Engineering Exam
Foundation Level / IREB compliant

❻ Estimate the relative cost and calculate the cost percentage for each requirement.
❼ Estimate the relative risks and calculate the risk percentage for each requirement.
❽ Calculate the individual requirement priorities:

$$Priority(R_i) = Value\%(R_i)/(Cost\%(R_i) \times WeightCost + Risk\%(R_i) \times WeightRisk)$$

❾ Assert the rank of the individual requirements.

It became apparent in practice that analytical prioritization approaches such as the prioritization matrix according to Wiegers as sketched above demand considerably more time and effort than ad hoc approaches, so these ad hoc approaches are to be favored in many cases. However, analytical approaches have the advantage that the degree of subjectivity in the prioritization results can be significantly reduced so that they lead to more objective and comprehensible results in complex and critical prioritization situations.

8.4 Traceability of Requirements

An important aspect of requirements management is ensuring the traceability of requirements. The traceability of a requirement is the ability to trace the requirements over the course of the entire life cycle of the system (see section 4.5.5).

8.4.1 Advantages of Traceable Requirements

Advantages of requirements traceability

The use of traceability information supports system development in many aspects and is often the precondition for establishing and using certain techniques during the developmental process [Pohl 1996; Ramesh 1998]:

- *Verifiability:* Traceability of requirements allows verifying whether a requirement has been implemented in the system, i.e., if the requirement has been implemented through a system property.
- *Identification of gold-plated solutions in the system:* Traceability of requirements allows for the identification of so-called gold-plated solutions of the developed system and thereby allows identifying unneeded properties. In order to do that, for each system property (functional or

qualitative), a check is performed to determine whether it contributes to the implementation of a requirement of the system.

- *Identification of gold-plated solutions in the requirements:* Tracing requirements back to their origin allows identifying requirements that do not contribute to any system goal and are not associated with any source. Usually, there is no reason for these requirements to exist and hence these requirements do not have to be implemented.
- *Impact analysis:* Traceability of requirements allows for the analysis of effects during change management. For example, traceability of requirements allows identifying the requirements artifacts that must be changed when their underlying requirements undergo a change.
- *Reuse:* Traceability of requirements allows for the reuse of requirements artifacts in other projects. By comparing the requirements of a previous project to the requirements of a new project by means of trace links, development artifacts (e.g., components, test cases) can be identified that may be adapted and/or reused in the new development project.
- *Accountability:* Traceability of requirements allows for retroactive assignment of development efforts to a requirement. After the requirement is implemented, for example, all partial efforts for the associated development artifact can be summed up and associated with the requirement.
- *Maintenance:* Traceability of requirements allows for simplified system maintenance. For example, the cause and effect of failures can be identified, the system components that are affected by the failure can be determined, and the effort for removing the underlying error can be estimated.

8.4.2 Purpose-Driven Definition of Traceability

As resources are usually severely restricted during development projects, capturing all conceivable information that supports the traceability of requirements over the course of the system life cycle is almost never possible.

Purpose of traceability information

In order to establish requirements traceability effectively and efficiently, the information to be recorded should be chosen with respect to the purpose that it will serve. In other words, only the information which has a clear purpose for system development or system evolution [Dömges and Pohl 1998; Ramesh and Jarke 2001] ought to be recorded. Recording

of traceability information that is not purpose driven often results in the fact that the recorded information cannot be profitably used in the development project. Traceability information that is recorded in this fashion is often sketchy and incomplete, unstructured, and erroneous with regard to its further use.

8.4.3 Classification of Traceability Relations

Pre-RS traceability and post-RS traceability

The pertinent literature on the topic of requirements traceability suggests different kinds of traceability of requirements. A common differentiation is distinguishing between pre-requirements-specification (pre-RS) traceability and post-requirements-specification (post-RS) traceability of requirements [Gotel and Finkelstein 1994]. We thus distinguish between three kinds of traceability:

- *Pre-RS traceability*: Pre-RS traceability are traceability links between requirements and those artifacts that are the basis for the requirements, e.g., artifacts like the source or origin of a requirement (previous artifacts).
- *Post-RS traceability*: Post-RS traceability comprises traceability information between requirements and artifacts of subsequent development activities. For example, such artifacts could be components, implementation, or test cases that belong to a requirement (posterior artifacts).
- *Traceability between requirements*: The traceability between requirements is about mapping dependencies between requirements. An example of this kind of traceability is the information that a requirement refines another requirement, generalizes it, or replaces it.

Figure 8-5 shows the three types of traceability of requirements in requirements engineering.

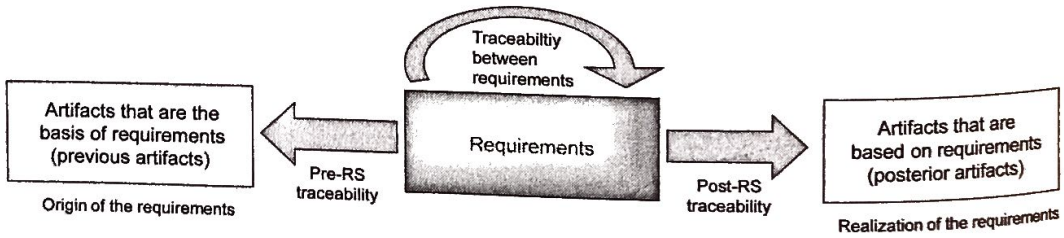

Figure 8-5 *Types of requirements traceability*

Figure 8-6 shows the three types of requirements traceability by means of requirement "R-14" in an example. The pre-RS traceability comprises the relations of requirement "R-14" to its origin. The origin of this requirement are the artifacts in the system context that influence the requirement. The post-RS traceability of requirement "R-14" consists of the relations to the components in the rough design, the refined design, and the respective implementation as well as test cases that are used during system testing and verify the implementation of the requirement in the developed system.

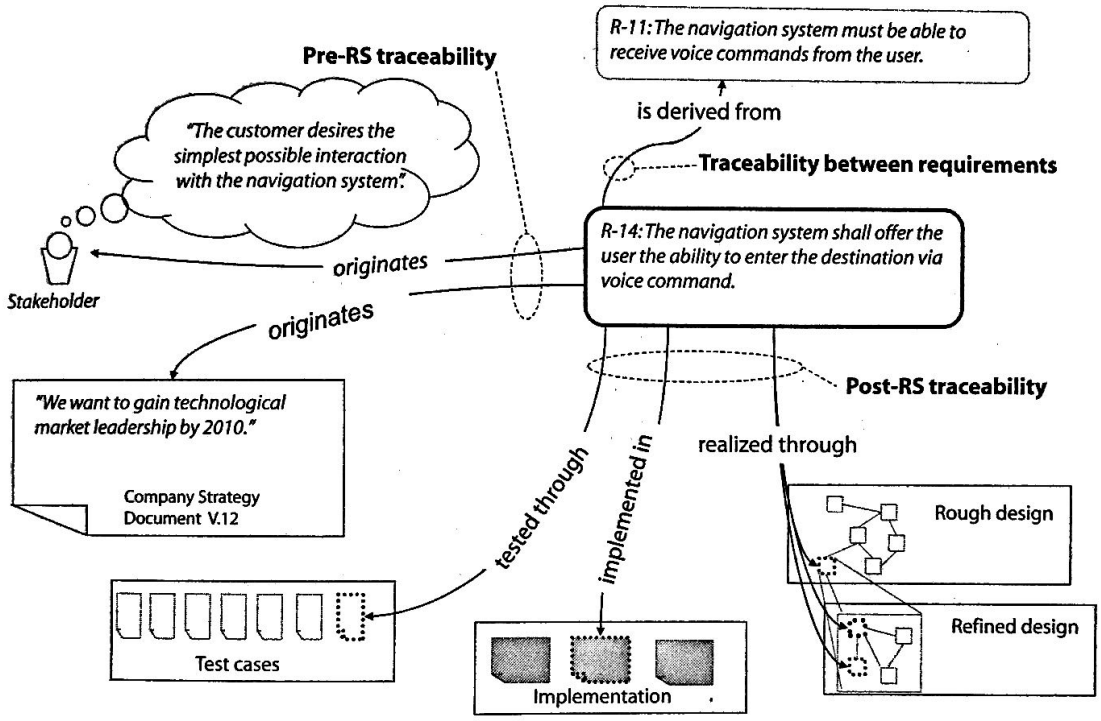

Figure 8-6 *Example of the three types of requirements traceability*

In addition, figure 8-6 shows the traceability between requirements. The traceability link between requirement "R-14" and "R-11" documents that requirement "R-14" was derived from requirement "R-11".

8.4.4 Representation of Requirements Traceability

Requirements traceability information can be represented in different ways. The most common approaches to representing traceability are simple textual references, hyperlinks, and trace matrices and trace graphs.

Text-Based References and Hyperlinks

This simple way to represent traceability information of a requirement consists of annotating the target artifact as a textual reference in the requirement (initial artifact) or to establish a hyperlink between the initial artifact and the target artifact. When linking artifacts, different types of hyperlinks with specific link semantics can be used.

Trace Matrices

Another common technique for representing and documenting traceability information between requirements as well as between requirements and previous and posterior artifacts in the development process are trace matrices. The rows in a trace matrix contain the initial artifacts (requirements). In the columns, the target artifacts (e.g., sources of requirements, development artifacts, requirements) are represented. If a trace link exists between an initial artifact in row *n* and a target artifact in column *m*, cell *(n, m)* is marked in the trace matrix.

Interpretation of a trace matrix

Figure 8-7 shows a simple trace matrix for the trace relation "derived" that exists between two requirements. An entry in the matrix specifies that a trace link of type "derived" exists from a requirement "*Req-n*" to another requirement "*Req-m*" such that "*Req-n*" was derived from "*Req-m*".

Target artifacts

Initial artifacts: derived	Req-1	Req-2	Req-3	Req-4	Req-5
Req-1		X			
Req-2			X		
Req-3					X
Req-4			X		
Req-5					

Figure 8-7 *Representation of traceability information in a trace matrix*

Maintainability of trace matrices

In practice, it became apparent that trace matrices are difficult to maintain as the number of requirements increases. A trace matrix that, for example, documents the refinement relations between merely 2,000 requirements contains over four million cells. In addition, many trace matrices must be created in order to be able to represent the available information cleanly (e.g., with regard to different types of traceability links).

Trace Graphs

A trace graph is a graph in which all nodes represent artifacts and all edges represent relationships between artifacts. The distinction between different artifacts and types of traceability can be realized by means of assigning different attributes to the nodes and edges of the graph.

Trace graph over different development artifacts.

Figure 8-8 shows the representation of traceability information in a simple example. In the trace graph, a node type is defined for each type of artifact (context information "*C*", requirements "*Req-n*", components "*Comp-n*"). In addition, three types of edges are defined to represent three types of traceability relations ("*realized through*", "*is origin*", "*refines*").

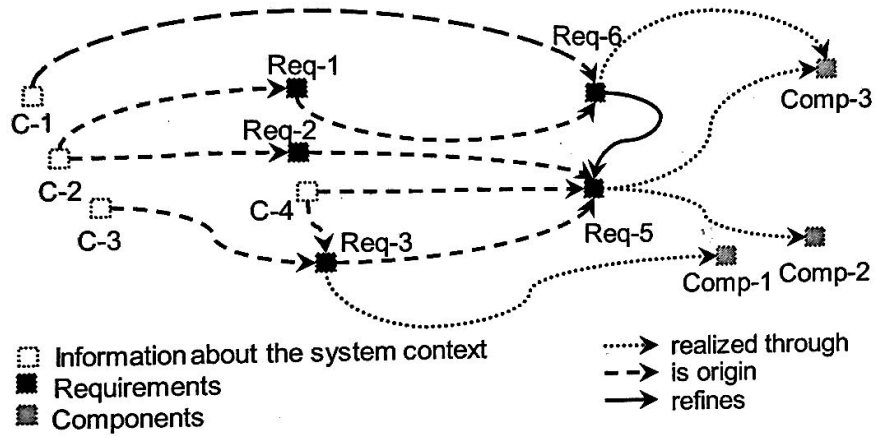

Figure 8-8 *Representation of traceability in a trace graph (extract)*

Traceability chains

If traceability information about previous artifacts (e.g., stakeholders and interview protocols) as well as posterior artifacts (e.g., test cases and components) must be managed, traceability chains for the respective requirement can be created at different levels, up to a trace of the requirement over the entire life cycle of the system. Common tools to maintain requirements allow for the definition of representation levels when creating traceability chains so that, depending on the selected level, only immediate relations of a requirement or entire traceability chains for the requirement can be generated and displayed. The traceability chains are the foundation for a comprehensive impact analysis during requirements change management.

8.5 Versioning of Requirements

During the life cycle of a system, the requirements of the system change as new requirements are added and existing requirements are removed or altered. The reasons for changes in requirements are diverse. One possible reason is, for instance, the fact that stakeholders learn more and more about the system as requirements engineering progresses. As a result, new and altered requirements come to their mind. Due to these changes, a suitable versioning of requirements is strongly advisable.

Subject of version control

Versioning of requirements aims at providing access to the specific change states of individual requirements over the course of the life cycle of the system. The version of a requirement is defined by its specific content of the change state and is marked by a unique version number. The information that is subject to version management can be single text-based requirements, sentences, sections of requirements documents, or entire requirements documents, but also requirements models and partial requirements models.

8.5.1 Requirements Versions

When versioning requirements, one can distinguish between the version and the increment of the version number. For example, the version number 1.2 references a requirement with version 1 and the increment 2.

Figure 8-9 illustrates the method of assigning version numbers. As shown in the figure, with smaller changes regarding the content, the increment is increased by one. If larger changes are performed, the version number is incremented. If the version number is increased, the increment is set to the initial value (0). A *v* can be added in front of the version number to make it more understandable and easier to identify as such.

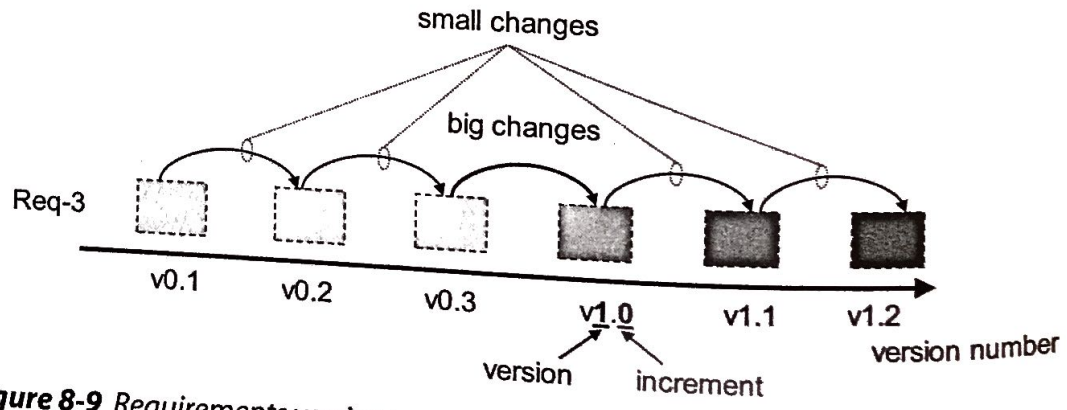

Figure 8-9 *Requirements versions*

Along with the rather simple structuring by means of version numbers, and the proposed method of versioning requirements, other methods of assigning version numbers are widely used. For example, it is possible to distinguish between the version identifier, the increment identifier, and the sub-increment identifier (v1.2.12).

8.5.2 Requirements Configurations

A requirements configuration consists of a set of requirements with the additional condition that each selected requirement is present in the requirements configuration with exactly one version, identified by the version number.

Dimensions of configuration management of requirements

Managing configurations of requirements can be described in two dimensions [Conradi and Westfechtel 1998]: In the product dimension, configuration management deals with individual requirements within the requirements foundation. In the version dimension, configuration management considers the various change states as part of version management within the product dimension. Figure 8-10 illustrates both dimensions of configuration management of requirements. On the requirements axis, requirements are represented. On the version axis, the different versions of the requirements are depicted.

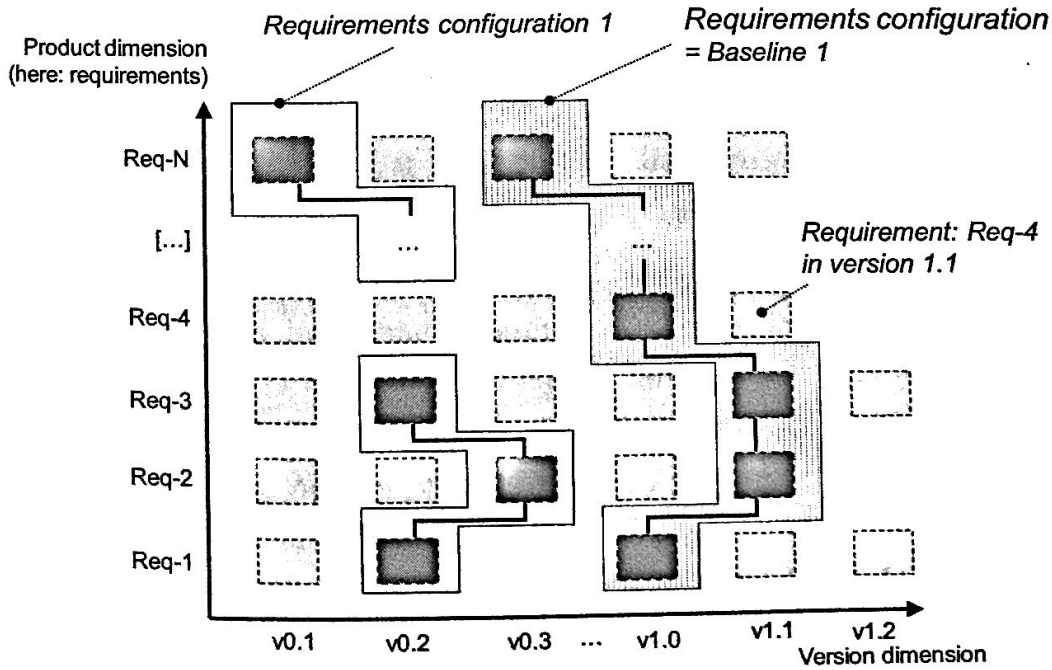

Figure 8-10 *Dimensions of configuration management of requirements (based on [Conradi and Westfechtel 1998])*