railwaysignalling.eu
walk the rail talk

Requirements Management for Safety Critical Systems

Maurizio Palumbo
July 2015
railwaysignalling.eu, UK
maur.palumbo@railwaysignalling.eu

ABSTRACT

A *Requirement* is a statement that identifies a product or operational, functional, or design characteristic or constraint, which is unambiguous, testable, or measurable and necessary for product or process acceptability. [3]

Requirements Engineering is the branch of systems engineering which address the process of identifying and then monitoring the stakeholders needs and systems required functionalities.

In this framework, *Requirements Management* is about organising these information in a form that will support the system implementation. In particular, it ensures the preservation of the information integrity during the whole system life-cycle, with respect to changes in the system and its environment. [19]

Requirements play a key role during the whole system development life-cycle, but since a typical medium-sized systems development project may yield some 2500 distinct statements of requirements, a number of Requirements Management tools (i.e: IBM DOORS) is available to support the product development and automate process.

Why is Requirements Management so important in the context of a structured organization which aims to deliver complex/safety critical systems?

Requirements Management offer to any project contractor the possibility to incrementally evaluate the efficiency of the system solution, by tracking its progress from the earliest phases (Concept Design) of a project, to System Testing, Validation and Operation.

KEYWORDS: Systems Engineering, Requirements Management, Requirements life-Cycle, V-Cycle for railway application, IBM Rational DOORS

1. INTRODUCTION

Systems Engineering (SE) is an interdisciplinary approach for the realization of successful systems, by developing a final product that meets the customer needs, goals and objectives.

Requirements Engineering is the branch of systems engineering which address the process of identifying and then monitoring the stakeholders needs and systems required functionalities.

In this framework, *Requirements Management* is about organizing these information in a form that will support the system implementation. In particular, it ensures the preservation of the information integrity during the whole system life-cycle, with respect to changes in the system and its environment. [19]

The issue is "how" is this information tracking is actually realized and what kind of tangible advantages it can guarantee to the systems development.

This paper will drive you across the different phases of the V-cycle for railway application, by explaining the key role of requirements during each phase and presenting the Systems Engineering activities required to support and control product development.

2. SYSTEMS ENGINEERING FUNDAMENTALS

2.1 Definitions

A *System* is a combination of interacting and integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products, processes, people, facilities, services, and other support elements. [3]

A *Safety Critical System* is a system whose failure or malfunction may result in:

- death or serious injury to people
- loss or severe damage to equipment/property
- environmental harm

Such risks are usually managed by methods and tools of Safety Engineering, and in particular through a safety performance measurement index to assess the required safety of a system, called *Safety Integrity Level (SIL).*

For more details about Safety, please refer to Appendix A.

Systems Engineering (SE) is an interdisciplinary approach for the realization of successful systems, by developing a final product that meets the customer needs, goals and objectives.

The *Systems Engineer* usually plays the key role in leading the development of a system, defining and allocating requirements, evaluating design tradeoffs, balancing technical risk between systems, defining and assessing interfaces, providing oversight of verification and validation activities, as well as many other tasks.

2.1 Systems Development Life-cycle

The systems development lifecycle is graphically well represented by the V-model. It presents and describes the activities to be performed (and the results that have to be produced) during each phase of the product development.

The term “V” comes from the shape of the diagram (see Figure 1 below), which is firstly split in two main macro-phases (phases 1,3) , connected by an intermediate one (phase 2):

1) System Decomposition and Definition (Design)

The left side of the "V" represents the design of the solution, during which the main concept will be specified and translated into system requirements, to be allocated to the different subsystems in the next stage. It is a top-down approach.

Final purpose of the systems development wil be ensuring that the whole system meets the requirements defined in this stage.

2) Implementation (Hw/Sw development)

This process is represented by the junction between the two sides of the “V”.

It transforms specified behaviour, interfaces and implementation constraints into fabrication actions that create a system element according to the practices of the selected implementation technology, regarding both Hardware and Software.

3) System Integration and Re-Composition (IVVQ)

The right side of the "V" represents the integration of the system parts (decomposed in the previous stage in order to be developed independently to each other) and correctness checking of static and dynamic aspects of interfaces between the implemented elements. It is a bottom-up approach.

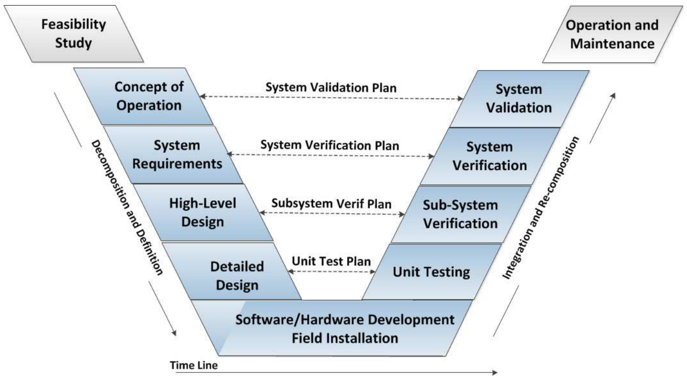


Figure 1: V-model diagram

Figure 1 highlights how the two macro-phases are linked to each other by means of bidirectionals arrows. Preparing the Verification and Validation plans is indeed a task to be accomplished whitin the design phase, whereas the activities addressed by these plans will actually take place during the integration and re-composition phase.

The full detailed description of each sub-phase of the system life-cycle is out of the scope of this paper, but for further details please refer to [1], [9] and [14].

3. REQUIREMENTS MANAGEMENT

3.1 What is a Requirement?

A *Requirement* is a statement that identifies a product or processes operational, functional, or design characteristic or constraint, which is unambiguous, testable, or measurable and necessary for product or process acceptability. [3]

3.2 Characteristic of good Requirements

A *well written Requirement* shall comply with the rules addressed by the following table:

Necessary
The stated requirement shall be an essential capability, physical characteristic, or quality factor of the product or process
Concise
The requirement statement shall include only one requirement simple and clear. To be concise, the requirement statements shall not contain any explanations, rationale, definitions or descriptions of system use
Complete (standalone)
The stated requirement is complete and does not need further amplification. The stated requirement will provide sufficient capability
Consistent
The stated requirement shall not contradict other requirements. It is not a duplicate of another requirement
Unequivocal
Each requirement shall have one and only one interpretation
Feasible
It can be implemented despite any possible limitations of the system and its environment
Verifiable/Testable
The stated requirement is not vague or general but is quantified in a manner that can be verified. The verifiability of a requirement should be considered at the same time that a requirement is being defined
Traceable
Each requirement shall be tagged with a unique name or reference number

Table 1: characteristics of good requirements

3.3 Types of Project Requrements

A complete set of project (technical) requirements includes the following types of requirements specifications:

- ***Functional Requirments (FR)***

Functional Requirements describe what the system shall do in terms of functionalities:

“System A shall send the message <MSG 1> to System B”

- ***Non-Functional Requirements (NFR)***

Non-Functional Requirements describe how these functions shall be performed. Some typical NFR follow:

Performance (e.g: Response Time, Throughput, Utilization, Static Volumetric), Scalability, Capacity, Availability, Reliability, Maintainability, Security, Environmental, Data Integrity

“System A shall connect into the Network within 10 seconds from its activation”

- ***Interface Requirements (IR)***

Interface requirements describe the interfaces between the design elements of the system.

“System A and System B shall be phyisically connected by means of an 100baseT Ethernet cable”

3.4 Requirements Engineering and Management

Requirements Engineering is the branch of systems engineering which address the process of identifying and then monitoring the stakeholders needs and systems required functionalities, organizing these information in a form that will support the system implementation.

As part of requirements engineering, *Requirements Management* is the activity concerned with the effective control of information related to systems requiremens, and in particular the preservation of the information integrity during the whole system life-cycle, with respect to changes in the system and its environment. [19]

Why is Requirements Management so important in the context of a structured organization which amis to deliver complex systems? An interesting cost-analysis is mentioned in [9] and summarized in Figure 2.

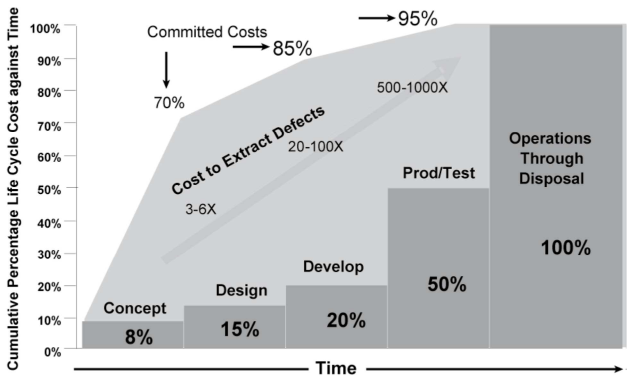


Figure 2: Committed Life-Cycle cost against Time

Requirements Management offer to any project contractor the possibility to incrementally evaluate the efficiency of the system solution, by tracking its progress from Concept to Operation.
Such approach allows to extract any defect in the early stages of the development, reducing risks and the associated costs.

3.5 Requirements Mangement along systems life-cycle

3.5.1 System Design

Requirements play a key role since in the earliest phases of a project. The following activities, listed in a chronological order, have to be taken in account:

1) Stakeholders Requirements Elicitation

Requirements elicitation is the practice of collecting the requirements of a system from users, customers and other involved stakeholders (any entity with a legitimate interest in the system).
It establishes the foundation from which the system is designed.

OUTPUT

At the end of the process, two formally documents will be delivered:
- A *Customer/Stakeholders Requirements Specification* will govern the development of the system, as the final product shall be compliant to it.
- A *Concept of Operations* will describe the way the system works from the operator's perspective. It will summarize needs, goals, and characteristics of the system's user community.

2) Formalise System Requirements

The purpose of this phase is to transform the stakeholders requirements (a set of desired services) into technical requirements for a specific product that could deliver those services.

OUTPUT

- A *System Requirements Specification (SRS)* shall state the functionalities that a system is required to achieve.

3) Develop Architectural Design

Developing an architectural design is about synthesizing a system component organisation which shall satisfy the system requirements.

OUTPUT

- A *System Architecture Description (SyAD)* shall be delivered as a formal description and representation of the system, in terms of its structure, behaviour and its interfaces with the external systems. It shall include a description of the system components (subsystems) and the interfaces (internal and external) among each other. A block diagram (System Breakdown Structure, see Figure 3) showing the major components, interconnections, and external interfaces of the system shall belong to this document.

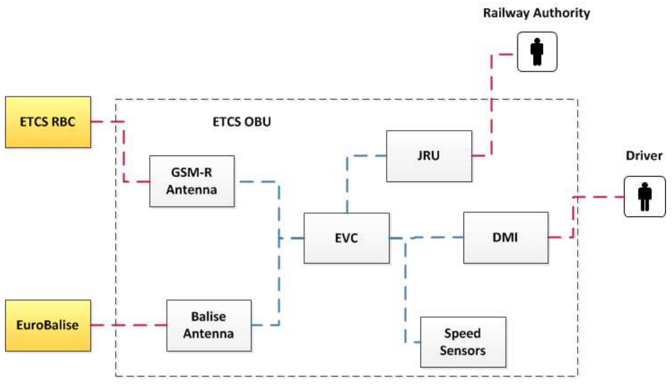


Figure 3: Simplified ETCS OBU SyAD

4) Formalise Subsystem Requirements

Once the System Architecture is defined, it's time to evaluate the System Requirements against the component structure, that is analysing the nature of these requirements and understanding which functionalities shall be achieved by a subsystem A rather than a subsystem B.

"The Onboard ETCS system shall automatically inform the driver upon start-up if the ETCS is unfit for service."

Talking about railway ERTMS/ETCS, the system requirement above shall be addressed to the subsystem "DMI", which is the on-board display, the main means of interaction between the driver and the system.

This relation between a source requirement and its natural evolution into a more detailed requirement is part of a requirements control process known as "Requirement

Traceability". A detailed description about traceability is available in §4.1 of this paper.

OUTPUT

For each defined component, a Sub-*System Requirements Specification (SSRS)* shall state the functionalities that a subsystem is required to achieve.

3.5.2 System IVVQ

The purpose of this phase can be summarized as below:

- Completely assemble the implemented elements to make sure that the they are compatible with each other.
- Demonstrate that the aggregates of implemented elements perform the expected functions and meet measures of performance/effectiveness.
- Detect defects/faults related to design and assembly activities by submitting the aggregates to focused V&V actions. [14]

The assembly activity joins together, and physically links, the implemented elements. Each implemented element is individually verified and validated prior to entering integration.

Regarding requirements, the key activity is each design requirement with a related test procedure which must be able to ensure the effectiveness of the addressed functionalities. Further details about that are covered in this paper in §4.1 (Conformance Traceability).

However, what's the difference between the concepts of Verification and Validation? Engineers usually struggle to collocate the two process and same terms often refer to different meanings. For further details about Verification and Validation and their application from a system point of view, please refer to Appendix B.

4. SYSTEMS ENGINEERING SUPPORT PROCESSES

The efficiency of Requirements Management activities is related to the implementation of a number of SE support processes, which help to control the product under development during its life-cycle.

4.1 REQUIREMENTS TRACEABILITY

Requirements Traceability is concerned with providing links between various associated requirements. It's common to differentiate Compliance and Conformance Traceability.

Compliance Tracebility

Compliance Traceability provides links between source and destination requirements.

One of the SE most common practices is indeed building a requirements hierarchy which will drive the evolution of each requirement from a high concept level to a more detailed specification of the functionalities to be satisfied.

Figure **4** below is a generic example of Requirements Breakdown Structure. The hierarchy is composed as follows:

- Level 0: Stakeholders Requirements
- Level 1: System Requirements
- Level 2: Subsystem Requirements
- Level 3: Unit Requirements

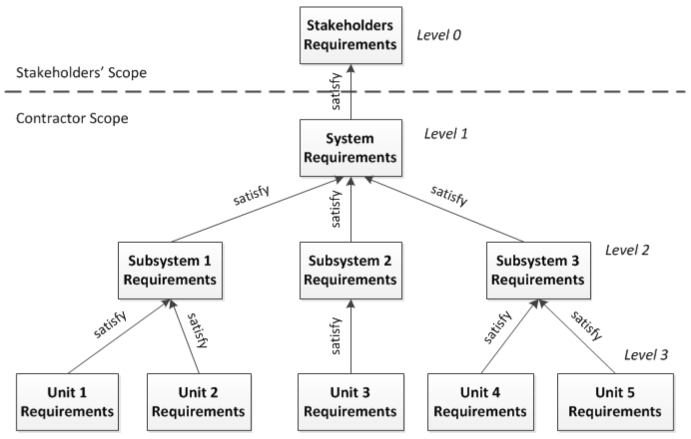


Figure 4: Requirements Breakdown Structure

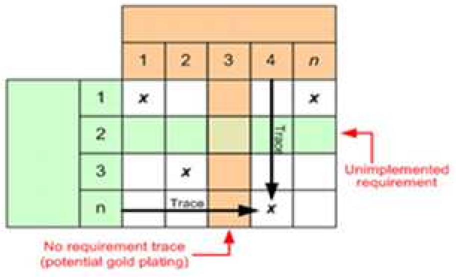
As expected, the route (Level 0) of the structure is represented by the Stakeholders Requirements Specification, the starting point of the product development.

A Stakeholders Requirement will be *satisfied* by a lower level (Level 1) system requirement, which again will be linked to a Level 2 requirement and so on, until the leaves of the tree (in this case Level 3 requirements) will be reached.

OUTPUT

The *Requirements Coverage Matrix* helps to trace and keep updated the source of each requirements.

Each requirement (unless it is a loaf requirement) shall be linked to a lower level requirement on the requirement tree, therefore the Matrix is also useful to identify any lack of compliance (unimplemented requirement) between a Requirement Specification and its linked destination.



1
2
3
4
n
1
2
3
n
x
x
x
x
Trace
Trace
Unimplemented requirement
No requirement trace (potential gold plating)

Figure 5: Identifying unimplemented requirements

Conformance Tracebility

Once requirements are written, methods for ensuring that the system contains the functionalities specified must be developed. There are a number of methods for verifying and validating functionalities, by determining if the system reacts in the expected way: testing, inspections, analysis, and demonstrations.

To verify and validate the requirements, test plans are written that contain multiple test cases; each test case is based on one system state and tests some functions that is based on a related set of requirements.

The conformity traceability guarantees the ability of the elements of the Solution to meet the allocated requirements, through one of the validation methods just mentioned.

OUTPUT

The *Requirements Conformance Matrix* associates each requirement to one or more test cases (or other Verification and Validation methods) in order to ensure the full conformance of any Requirements Specification.

4.2 CONFIGURATION AND CHANGE MANAGEMENT

Configuration management (CM) is about "tagging" a frozen version of a system, subsystem, document or any Configuration Item with a unique configuration identifier, which serves as a basis for defining and implementing changes during the product development lyfe-cycle.

Configuration management (CM) is about "tagging" a frozen version of a system, subsystem, document or any *Configuration Item* with a unique *Configuration identifier,* which serves as a basis for defining and implementing changes during the product development lyfe-cycle.

In configuration management, a *System Baseline* is a frozen version of the system under development.

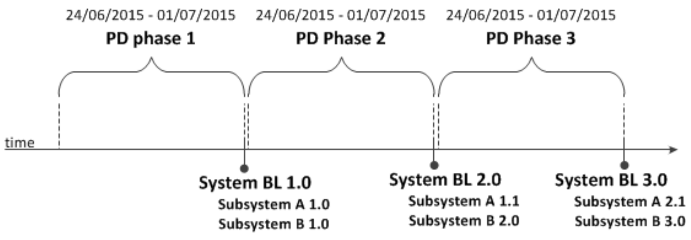


Figure 6: Baseline releasing along the project life-cycle

Figure 6 above presents a system composed by the two subsystems *A* and *B*. Each of them is to be considered a configuration item, therefore the system configuration will be:

- Sysem Baseline ***x.y***
  - Subsystem A ***z.w***
  - Subsystem B ***i.q***

For each released system baseline, it's possible to propose (open) a Change Request (CR).

Changes are analysed to determine the impact that the change will have on the work product, related work products information, schedule and cost.

Then, during the Change Control Board (CCB) the CR is discussed and, if considered as valid, it is assigned to an implementer, who realizes the modification. The change will actually take place and applied to the next system baseline.

Moreover, in this way it's possible to evaluate the detailed gap between two adjiacent systems baselines.

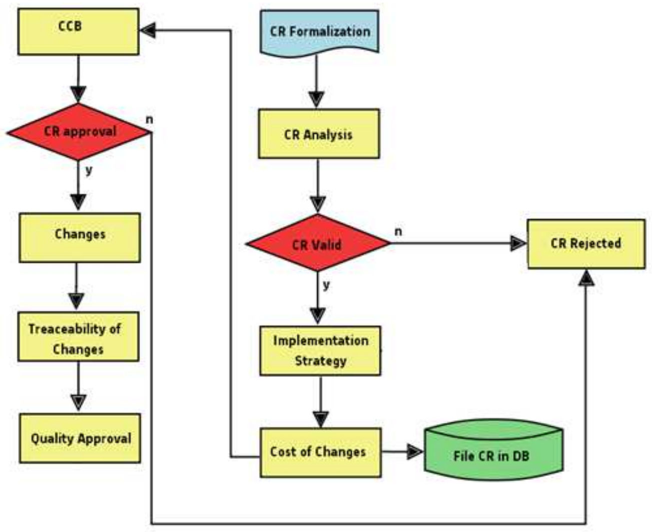


Figure 7: Change Request Process

5. REQUIREMENTS ENGINEERING TOOLS

For any realistically sized systems engineering project, requirements management is a clerically intensive task.

It entails being able to:

- relate many different documents;
- obtain a synoptic view of these document relations;
- retrieve information from within those documents;
- handle changes made across the set of documents in a consistent manner;
- accommodate diverse document structuring requirements and document types.

In order to understand the scale of Requirements Management it should be kept in mind that a typical medium-sized systems development project may yield some 2500 distinct statements of requirements, each of which may in turn result in a variety of development documents and associated rationale. [20]

Managing manually these huge amounts of data can still be an option, but it comes naturally that nowadays many organization are adopting a Requirements Management tool, to support the system development process.

A requirements management tool allows to:

- organizing requirements in a shared database, providing access to it to all of the project stakeholders and ensuring the control of requirements for the whole project life-cycle;
- creating linkages between requirements relevant to systems and subsystems areas;
- developing gap analysis, identifying potential inconsistencies between what is required and what is or will be satisfied;
- Track progressive informed changes to the product development, linking any update of requirements and its related impacts to the overall design and deliverability;
- Verify that the proposed design will meet the target performance;
- Enable evidence of compliance (through analysis, drawings, calculations, prototypes and test results) to be traced to requirements;

5.1 IBM DOORS

Among the different tools available, it's worth to spend some words about *DOORS* (Dynamic Object Orientated Requirements System), the most used tool in the world for the development of Safety Critical systems.
DOORS is a Requirements Management tool based on a Client-Server architecture that contains features for capturing, tracking, and managing project requirements.
Requirements can be entered into a database, tracked and managed throughout their life cycle using a variety of features, such as views, links and traceability analyses.

A full description of this tool is out of the scope of this paper. For any detail, please refer to [11].

6. NOMENCLATURE

CCB	Change Control Board
CM	Configuration Control
DOORS	Dynamic Object Oriented Requirements System
DMI	Display Machine Intefrace
ERTMS	European Railway Traffic Management System
ETCS	European Traffic Control System
EVC	European Vital Computer
FR	Functional Requirements
GSM-R	Global System for Mobile Communications - Railway
ICD	Interface Control Document
IR	Interface Requirements
IVVQ	Integration Verification Validation and Qualification
JRU	Juridical Recording Unit
NFR	Non Functional Requirements
OBU	On-Board Unit
RM	Requirements Management/Manager
RMP	Requirements Management Plan
RSM	Requirements Status Metrics
RTM	Requirements Traceability Matrix
RE	Requirements Engineering/Engineer
SE	System Engineering/Engineer
SIL	Safety Integrity Level
SRS	System Requirements Specification
SSRS	Subsystem Requirements Specification
SyAD	System Architecture Description
SysML	System Modelling Language
UML	Unified Modelling Language

7. REFERENCES

[1] National Aeronautics and Space Adrminstration (NASA) – "NASA Systems Engineering Handbook", 2007
[2] Atkins Rail Ltd – "Railway Systems Engineering in Action", 2006
[3] ISO/IEC 15939:2007 – "Systems and software engineering - Measurement process", 2007
[4] ISO – "ISO 900:2005, Quality Management Systems – Fundamentals and vocabulary", 2015
[5] CDC Unified Process Practices Guide – "Requiremets Management", 2007
[6] Dr. Steve Easterbrook (Institute for Software Research) –" Verification and Validation of Requriements for Mission Critical Systesms", 2004
[7] Holt, "UML for Systems Engineering: watching the wheels", Second edition, 2004
[8] Alexander & Stevens - "Writing Better Requirements" - Addison-Wesley 2002
[9] INCOSE – "System Engineering Handbook, a guide for system life-cycle processes and activity" - 2010
[10] Andrew Bourne (Tube Lines Ltd) – "System Requrirements Management", 2007
[11] IBM – "Software lifecycle management" available: http://www.ibm.com/developerworks/rational/slmoverview.html
[12] Larry A. Fellows (Compliance Automation Inc.) - "10 Steps to Better Requirements", 2003
[13] Department of Computer Science University of Missouri-Rolla - "Analysis of Conflicts among Non-Functional Requirements Using Integrated Analysis of Functional and Non-Functional Requirements", 2007
[14] INCOSE –Guide to the Systems Engineering body of knowledge available: http://sebokwiki.org/wiki/System_Requirements
[15] IBM – Rational DOORS getting started, 2013
[16] Ivy Hooks – "What Happens with Good Requirements Practices" - 2001
[17] T. Hammer, L. Rosenberg, L. Huffman, L. Hyatt (IEEE) – "Measuring Requirements Testing", 2005
[18] Requirements Experts – Checklist for Common Requirements Risk Factors, 2012
[19] A. Finkelstein (UCL) – "Requirements Management", available at: (http://www0.cs.ucl.ac.uk/staff/A.Finkelstein/talks/rmfsetut.pdf)
[20] A. Finkelstein (UCL) – "The future of Requirements Management Tools ", 2010.
[21] EN 50120 – Railway Applications – Communication, signaling and processing systems – Safety related electronic systems for signalling

8. BIOGRAPHY

Maurizio Palumbo is the founder of railwaysignalling.eu, where he's also known as "Vesuvius". He was born in Naples (Italy) on 26th September 1986. He's a young, curious, smiling and enthusiastic computer systems engineer with specific expertise in railway signalling and ERTMS/ETCS (European Railway Traffic Management System / European Train Control System). He has been involved in Systems and Application Engineering activities on three ERTMS schemes (Italy, Denmark and UK) and on a metro project (London tube, UK).

APPENDIX A: Safety Integrity Level

Safety engineering is a discipline which assures that engineered systems provide acceptable levels of safety.
During the system design phase, the System Requirements Specification is submitted to the safety engineering team, which process it to assess risk by means of a *Hazard Analysis*.
A *Hazard* is defined as a condition, event, or circumstance that could lead to or contribute to an unplanned or undesirable event.
A *Hazard Log* shall be created and maintained throughout the safety life-cycle. It shall include a list of identified hazards, toghether with associated risk classification and risk control information for each hazard. [21]
Safety integrity level (SIL) is defined as a relative level of risk-reduction provided by a safety function, or to specify a target level of risk reduction. In simple terms, SIL is a measurement of performance required for a safety instrumented function.

The safety standards for railway applications are fully specified in [21].

APPENDIX B: Verification and Validation

Verification proves that a realized product for any system model within the system structure conforms to the build-to requirements. In other words, the purpose of the Verification Process is to confirm that the specified ***design requirements*** are fulfilled by the system.

This process also provides the information required to effect the remedial actions that correct the non-conformances in the realized system.

Validation is the confirmation, through the provision of objective evidence, that the ***stakeholders requirements*** for a specific intended use or application have been fulfilled.

A validated system is therefore able to accomplish its intended final use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment.

A validation action applied to an engineering element includes the following:

- Identification of the element on which the validation action will be performed.
- Identification of the reference that defines the expected result of the validation action.
- Any engineering element can be validated using a specific reference for comparison (see Figure 8 below).

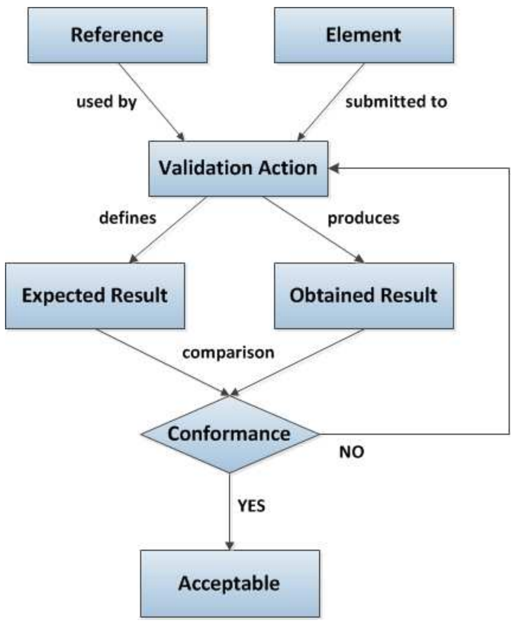


Figure 8: Validation Process

According to these definitions, from a process perspective, the two activities may be similar in nature, but the objectives are fundamentally different.

From an engineering point of view, it is essential to confirm that the realized product is in conformance with its design specifications (***did we build it right?***), whereas from a customer point of view, the interest is in whether the end product provided will do what the customer intended (***did we build the right thing?***). [1]

Moreover, it is important to understand the iterative nature of these activites, which can be repeated in each stage of the system life-cycle, in order to provide a progressive evidence of requirements satisfaction (and remedial actions against non-conformities) during the product development (see Figure 9 below).

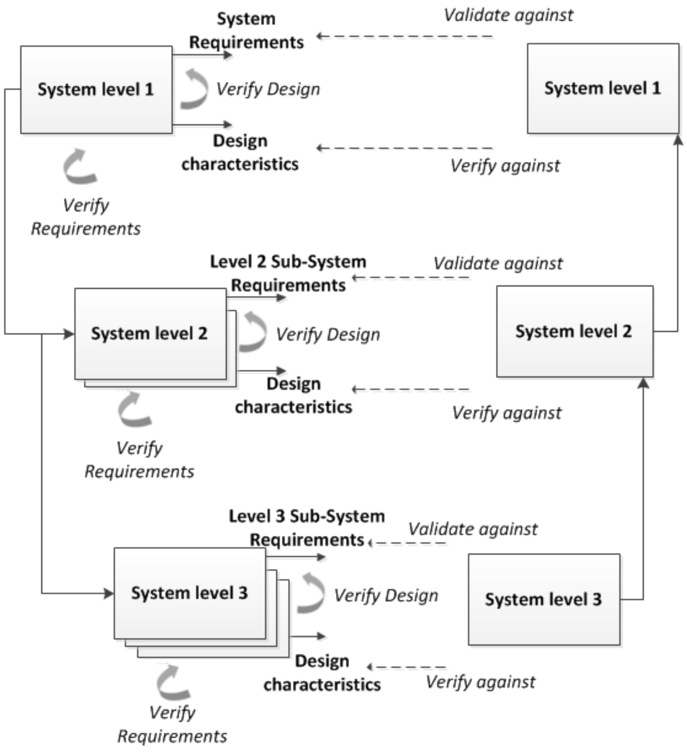


Figure 9: Iteractions of Validation and Verification process