Requirements Engineering in Rail Transit Production: an Experience Report

Fernanda Buonanno*, Domenico Di Leo †, Paolo Di Paolo*, Roberto Pietrantuono†, Stefano Russo‡

*AnsaldoBreda s.p.a.
† Università di Napoli Federico II
‡ Critiware s.r.l.
Outline

♦ The public-private cooperation
♦ Requirements engineering and standards
♦ Case Study
♦ Issues in the requirements engineering process
♦ Implemented actions
♦ Lessons learned
Context

♦ Ansaldo Breda is a Finmeccanica group’s company in the market of rail transit systems, with a product line (MLA) of technologically advanced rolling stocks
  ▪ High speed train, Metro, Tram
♦ The MLA platform has been deployed in several sites all over the world, such as Riyadh (SA), Milan and Brescia (I), Saloniki (GR), Taipei (RC).
Context

♦ Companies like AB are called to build high-quality control software, compliant to reference standards of both the railway and software industry (such as CENELEC 50128, IEEE 830), with contained cost and within stringent time to market.

♦ This pushes to explore a modernization of the internal software development life cycle to strengthen key aspects like reusability (across the product line) while assuring software integrity levels.
Requirements engineering as innovation driver

♦ Requirements engineering as a driver to improve product and process quality in the given context
  ▪ better support of software quality and safety assurance activities, with a proper cost/quality trade-off
  ▪ higher quality costs are compensated through reuse over a product line

♦ Note: AB is “traditional” company manufacturing trains, not a software company
  ▪ software is part of the train control systems
Requirements

- **Requirement** *(from IEEE-STD-1220-1998 “Standard for application and management of the systems engineering process”):* a statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)

- **Requirements engineering (RE)** *(from Wikipedia):* RE refers to the process of formulating, documenting and maintaining software requirements

- **RE activities include:**
  - Req elicitation - Req analysis and negotiation – Req specification – Req validation – Req management

“RE is common sense but it is perceived difficult and not well understood. For this reasons it is generally not well done” – Hull, Jackson, Dick, 2002
Requirements management

Maintenance requests (corrective/evolutionary)

Agreement

Analysis and design models

Requirement

Test cases

Define test cases (tester)

Map/trace requirements to analysis/design models

Change management

(Baseline)

(Customer)
Requirements in critical software

♦ Requirements have “a crucial importance … in critical software systems engineering” [3]

♦ Their importance is however underestimated, if we consider that “most tools in common use today still represent a requirement as a simple, unadorned string” [3]

Requirements and certification

Two true Neapolitans in Pasadena (certified Neapolitans)

Looking for a true Neapolitan pizza

Wow! There exist a pizza certification! Vera Pizza Napoletana!

Look! They must make a good pizza here

Ehm, will it really taste good?

♦ Roberto: It’s certified! So what, Domenico?
♦ Domenico: What were the requirements specified for a good pizza?
btw ... in fact:

♦ asked for a pizza with peperoni

♦ got a pepperoni pizza!
What standards tell about requirements – 1/3

**CENELEC EN 50128** *(Railway applications - Communications, signalling and processing systems)*

♦ 16 statements in Section 7.2 describe what is expected from requirements specification

♦ Table A.2 suggests techniques (e.g., Modeling, Formal Methods) to specify requirements

♦ Traceability is required as quality assurance technique/measure

♦ The role of the “requirement manager” is also foreseen
What standards tell about requirements – 2/3

**DO178C** *(Software Considerations in Airborne Systems and Equipment Certification)*

- Provides a guidance on how to structure them *(high-level and low-level requirements)*, and how to specify *(10 statements in section 5.1)*
- Requirements-based testing is explicitly required with several methods suggested *(6.4.3)*
- Emphasis on compliance, accuracy, consistency, traceability *(Table A-3)*
What standards tell about requirements – 3/3

ISO 26262 *(Road vehicles - Functional safety)*

- Reports several clauses in Section 6 about how to specify software safety requirements similarly to CENELEC in the reference V-Model
- Suggests method for verifying requirements (walkthrough, inspection. Semi-formal and formal verification)
Case Study

♦ A pilot project focused on a driverless Metro product line, namely what UNI 8369 defines as Light Rail Transit

♦ Main software systems
  - Train Control System (TCS)
  - Traction Control Units (TCUs)
  - Monitoring and Diagnostic System (MDS)
The MLA Product Line
Ansaldo Breda development process

- The company software development process adheres to the **V-Model** prescribed by the **CENELEC EN 50128** standard
  - adopted in many critical industrial domains
  - accounts for (V&V) at early stages, as soon as requirements are elicited
  - supports model-based V&V
Scope of Requirements Engineering in AB

Contractual document → System Requirements → Software Requirements → Design Documents

Customer → System Engineer → System Engineer → Project Engineer → Software Requirements Engineer → Software Developers
Main issues

- Insufficient **quality** and high **heterogeneity** of requirements
  - writing style, structuring, classification, granularity, relations, semantics
  - poor separation of *what* from *how*
- **Heterogeneity** leads frequently to misinterpretation
- Missing **traceability** and poor **testability**
- Lack of proper **abstraction level**
  - to foster comprehension, maintenance, reuse, ...
An example

The master ATO (in ATO or ATO+ATP modes) will request a braking effort by means of MVB signals, `AtoAtp[x].AtoCmd.Clk11._BrkMode_`, datasets 707(ATO A), 708 (ATO B), and the related effort (signal `AtoAtp[x].AtoCmd.ucPrpBrkEfRq`, `x = 0 => ATO A, x = 1 => ATO B`).

TCMS receives the requested effort rates and re-sends this information on MVB to TCUs and BCUs. *(Datasets 500, TcuBcu.ucRifMan).*

In every running mode (except for ATC BYPASS) at the start-up the ATC system will define which of the two on board equipments (ATC A or ATC B) will be the Master.
Main actions

♦ We defined:
  - Requirements structure
  - Abstraction levels
  - Guidelines for writing requirements
  - Relationships among requirements
  - Requirements traceability rules
  - Use of support tools
## Structure of a requirement

<table>
<thead>
<tr>
<th>Attribute</th>
<th>HLR</th>
<th>LLR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
<td>SHR-TCMS-01</td>
<td>SLR-TCMS-01</td>
</tr>
<tr>
<td><strong>TYPE</strong></td>
<td>Functional</td>
<td>Functional</td>
</tr>
<tr>
<td><strong>LINK TO</strong></td>
<td>SYS-REQ01</td>
<td>SHR-TCMS-01</td>
</tr>
<tr>
<td><strong>NAME</strong></td>
<td>Traction release by TCS</td>
<td>Traction release by TCS</td>
</tr>
<tr>
<td><strong>PRECONDITIONS</strong></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>INPUT</strong></td>
<td>Coasting and Propulsion</td>
<td>Coasting and Propulsion</td>
</tr>
<tr>
<td></td>
<td>commands Reset commands on</td>
<td>commands on MVB</td>
</tr>
<tr>
<td></td>
<td>by ATC</td>
<td>by ATC through dataset 707/708</td>
</tr>
<tr>
<td><strong>OUTPUT</strong></td>
<td>Coasting commanded,</td>
<td>Coasting -&gt; 1 on MVB,</td>
</tr>
<tr>
<td></td>
<td>service brake released,</td>
<td>Service Brake -&gt; 0,</td>
</tr>
<tr>
<td></td>
<td>and breaking train line</td>
<td>Train Line -&gt; 1.</td>
</tr>
<tr>
<td></td>
<td>Train Line set to off. BCU</td>
<td>set to off. BCU shall se-</td>
</tr>
<tr>
<td></td>
<td>shall send back the status of</td>
<td>BCU shall send back the status</td>
</tr>
<tr>
<td></td>
<td>service and holding brake.</td>
<td>of status of service and held</td>
</tr>
<tr>
<td>**POST</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CONDITION</strong></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Requirement restructuring

The master ATO (in ATO or ATO+ATP modes) will request a braking effort by means of MVB signals, AtoAtp[x].AtoCmd.Clk11._BrkMode_, datasets 707 (ATO A), 708 (ATO B), and the related effort (signal AtoAtp[x].AtoCmd.ucPrpBrkEfRq, x = 0 => ATO A, x = 1 => ATO B).

TCMS receives the requested effort rates and re-sends this information on MVB to TCUs and BCUs. (Datasets 500 TcuBcu.ucRifMan).

In every running mode (except to ATC BYPASS) at startup the ATC system will define which of the two on board equipments (ATC A or ATC B) will be the Master. The master ATO (in ATO or ATO+ATP modes) will request a braking effort by means of MVB signals, and the related effort. TCMS receive the requested effort rates and re-send this information on MVB to TCUs and BCUs.

Master ATO selection, signal (AtoAtp[x].AtpCntr.Cntr8._MstATC_) is managed in Mat_OpModes.mdl model, function “AtcMaster” providing signal OpMod.Sts1._MasterATC_.

Effort management in case of manual mode or automatic mode is performed in Effort function, model Mat_EDC.mdl.

**Input:** MVB signals

**Description:** In every running mode (except to ATC BYPASS) at the start-up ATC system will define which of the two on board equipments (ATC A or ATC B) will be the Master. The master ATO (in ATO or ATO+ATP modes) will request a braking effort by means of MVB signals, and the related effort. TCMS receive the requested effort rates and re-send this information on MVB to TCUs and BCUs.

**Output:** BCUs and TCUs receive the requested effort rates

**Type:** Functional

**Verification criteria:** check that the effort rate received by TCU is exactly the one required for braking
Abstraction

♦ Distinguish low-level from high-level requirements (HLR, LLR)
  ▪ like in DO-178C
♦ Separate the platform-level (more abstract) and project/product-dependent (more specific) sets of requirements.
  ▪ Separate concerns and responsibilities
  ▪ Favours reuse
Relationships among requirements

- We defined relations among elements at different level of abstraction, both vertically and horizontally:
  - Each system functional requirement with the interface requirements it references
  - Each high-level software requirement with the corresponding system requirements that it implements
  - Each high-level software requirement with other HLRs of other subsystems that it mentions in the text
  - Each low-level software requirement with the corresponding HLR that it implements
  - Each low-level software requirement with other LLRs of other subsystems that it references in the text.
We defined hierarchical relationships between requirements also at the same level of abstraction, for instance indicating a normal and an abnormal scenario.
Traceability

♦ Traceability establishes the (bidirectional) relationships among artifacts of the development process
♦ Traceability is crucial for many quality control and assurance activities
  ▪ Traceability *starts* from requirements
♦ The clear definition of items subject to traceability is fundamental
♦ Defining a homogenous form for requirements (in size, style, structure, classification, level of abstraction), and vertical and horizontal relations among requirements, makes tracing trees clear to understand and much less prone to mistakes
♦ Traceability implemented and verified up to source code files and test cases for each requirement
Tool support

♦ The usage of a requirements management tool (Rational DOORS), by all actors involved in the company and across all development phases, was promoted

but ...

♦ consolidated use and common practices require
  ▪ knowledge transfer
  ▪ technology transfer
  ▪ training

♦ and
  ▪ commitment by all internal stakeholders
Main benefits

♦ The major observed benefits from the implemented actions concern:
  ▪ Testability
  ▪ Traceability
  ▪ Reusability
  ▪ Maintainability
  ▪ Automation
Lessons learned – 1/2

♦ For critical embedded systems, requirements engineering tremendously impacts all development phases, and QA and V&V activities

♦ Badly specified requirements not only lead to unintended software, but also to uneffective testing

♦ RE is becoming more important in safety standards (and increasingly perceived as important in industry)

♦ Sound requirements engineering practices are essential if the embedded software is meant to undergo a certification process

♦ RE provides valuable benefits also in terms of
  ▪ Reusability – Maintainability – Automation
Lessons learned – 2/2

♦ In a traditional industry, where software is only part of complex systems (although an increasing part) sound RE practices require:
  ▪ long time
  ▪ strong commitment of all actors
  ▪ training, knowledge transfer, technology transfer
♦ Tool support and tools integration are important – but beware: no panacea, “hidden” costs
♦ The V&V team plays an important role
  ▪ it can devise testing activities as soon as the requirements are available, and can give valuable feedback to proper requirements specification practices in the application field
♦ Benefits are “guaranteed”, but in the long term
Questions