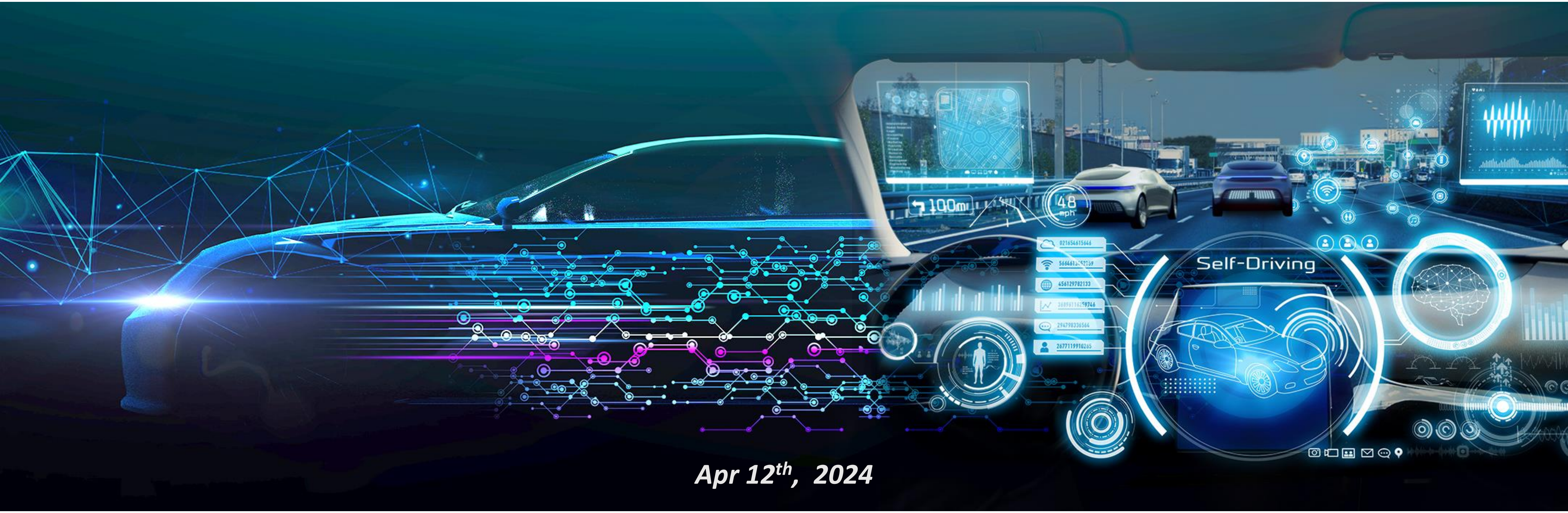
Requirements Engineering



Apr 12th, 2024

Marco Marazza
FuSa Mgr. & MBSE expert @ESKUBE
https://www.eskube.com/
marco.marazza@eskube.com

The Vital Role of Requirement Engineering in the Automotive Industry

THE CHAOS REPORT, 1995 [9]

The survey participants were also asked about the factors that cause projects to be challenged.

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

36.9%

Opinions about why projects are impaired and ultimately canceled ranked incomplete requirements and lack of user involvement at the top of the list.

Project Impaired Factors	% of Responses
1. Incomplete Requirements	13.1%
2. Lack of User Involvement	12.4%
3. Lack of Resources	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. Changing Requirements & Specifications	8.7%
7. Lack of Planning	8.1%
8. Didn't Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

25.5%

Requirements Engineering (2019) [11]

The impact of requirements on systems development speed: a multiple-case study in automotive

Abstract

Automotive manufacturers have historically adopted rigid requirements engineering processes. This allowed them to meet safety-critical requirements when producing a highly complex and differentiated product out of the integration of thousands of physical and software components. Nowadays, few software-related domains are as rapidly changing as the automotive industry. In particular, the needs of improving development speed are increasingly pushing companies in this domain toward new ways of developing software. In this paper, we investigate how the goal to increase development speed impacts

RE as the cause of project behind schedule

REFSQ 2023 [10]

Bringing Stakeholders Along for the Ride: Towards Supporting *Intentional* Decisions in Software Evolution

Abstract. [Context and Motivation] During elicitation, in addition to collecting requirements, analysts also collect stakeholders' goals and the present and historical interests that motivate their goals. This information can guide the resolution of requirements conflicts, support the evolution of requirements when changes occur (e.g., environmental constraints), and inform decisions in software design. [Problem] Unfortunately, this information is rarely explicitly represented and maintained.

RE challenged by long-standing, difficult issues

RE challenged by new technology trends

RE as the cause of project failure

REFSQ 2023 [10]

A Requirements Engineering Perspective to AI-Based Systems Development: A Vision Paper

Abstract. *Context and motivation*: AI-based systems (i.e., systems integrating some AI model or component) are becoming pervasive in society. A number of characteristics of AI-based systems challenge classical requirements engineering (RE) and raise questions yet to be answered. *Question*: This vision paper inquires

ESKUBE SYSTEMS SAFETY & SECURITY

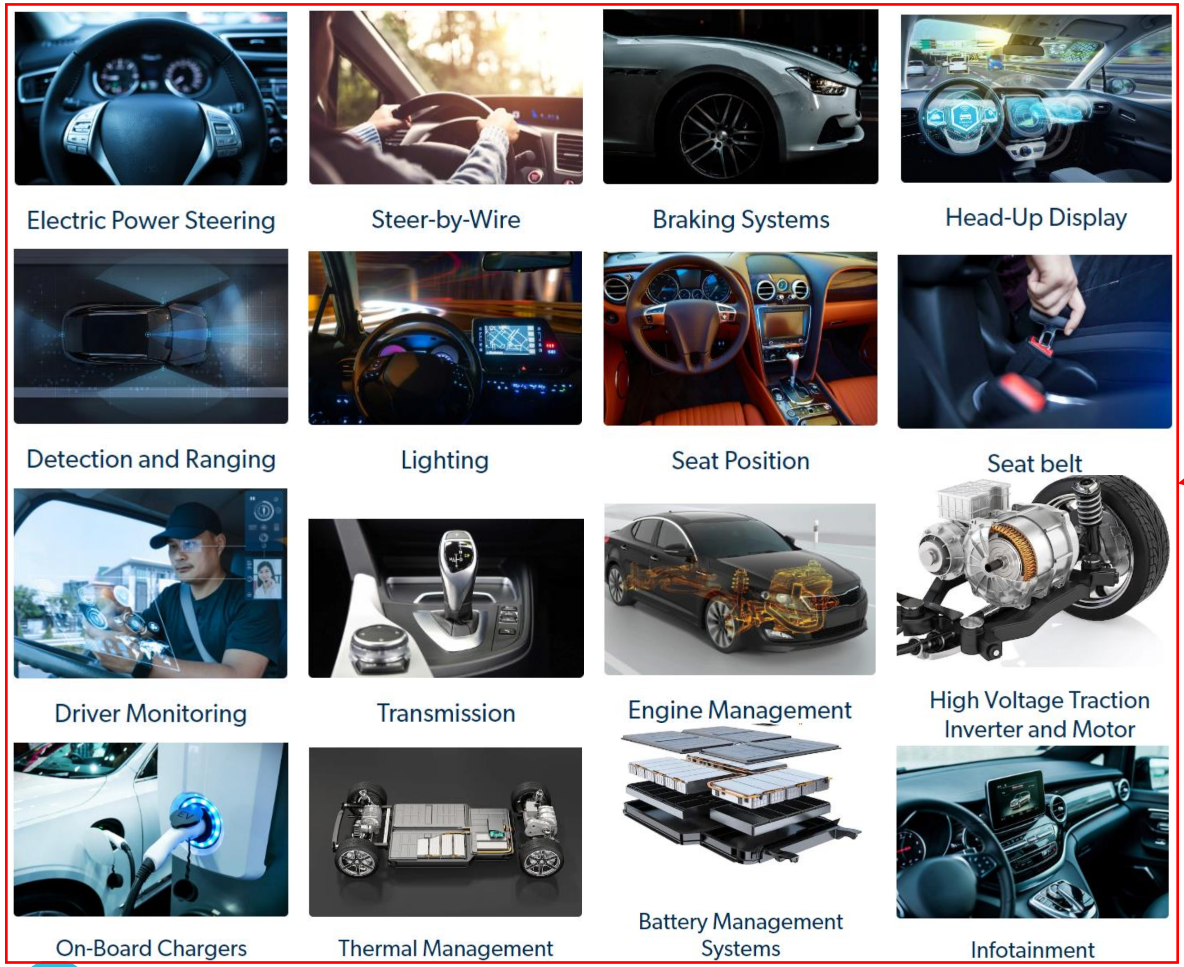
Outline

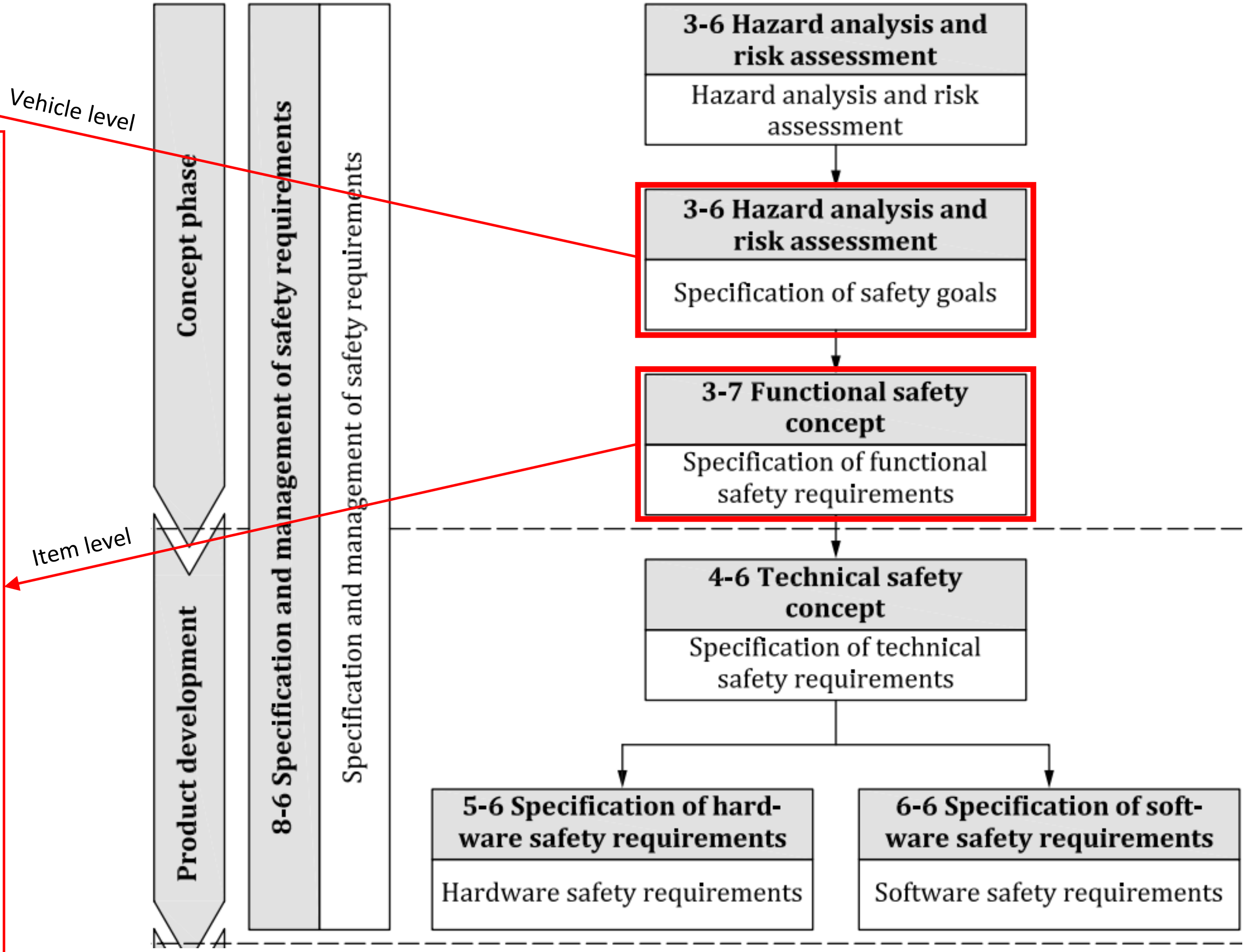
1. What is a requirement and why is it so important
2. Where do I find product requirements in a project
3. How many types of requirement do we need
4. What is Requirement Engineering
5. Example
6. Appendix A: Requirement analysis techniques

What is a requirement and why is it so important

1. Agreement between the **stakeholders** and the **supplier**
   1. **Supplier**
      1. Team, or department, or organization developing a **product** and delivering it to its customer(s)
   2. **Stakeholder**: anyone interested in some (all) aspect(s) of product's development:
      1. **Customer**(s): who is eventually buying the product (end user)
      2. Other **departments / teams** in the supplier's organization
      3. Other **organization**
2. Agreement about product's purpose, functionality, conditions of use, modes of operation, accuracy, efficiency, availability, safety, cyber-security, liveness, […]
3. **Product** = vehicle, ECU, hardware board, hardware IP, soft IP, software unit, software component, …
4. Requirements must be **understandable** to both stakeholders and supplier

Where do I find product requirements in a project

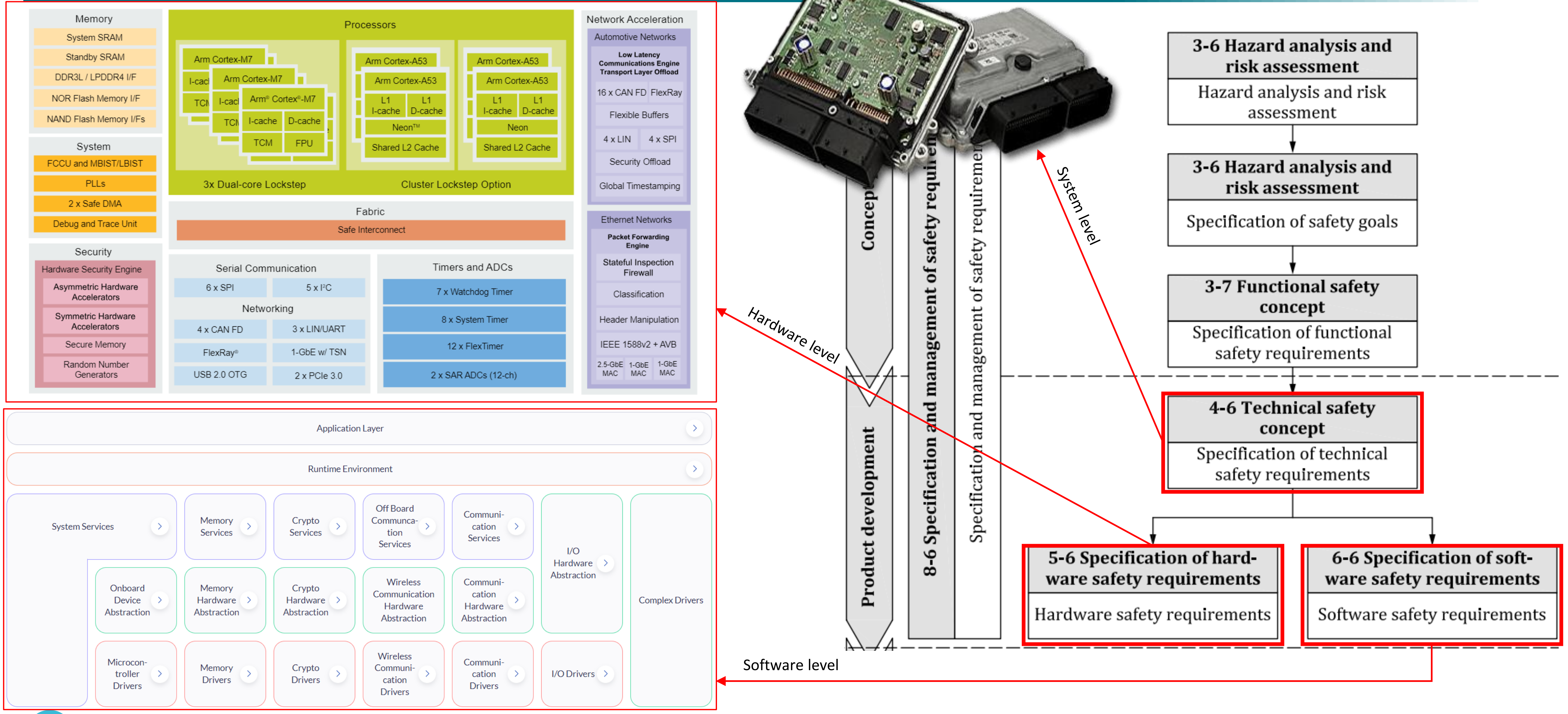




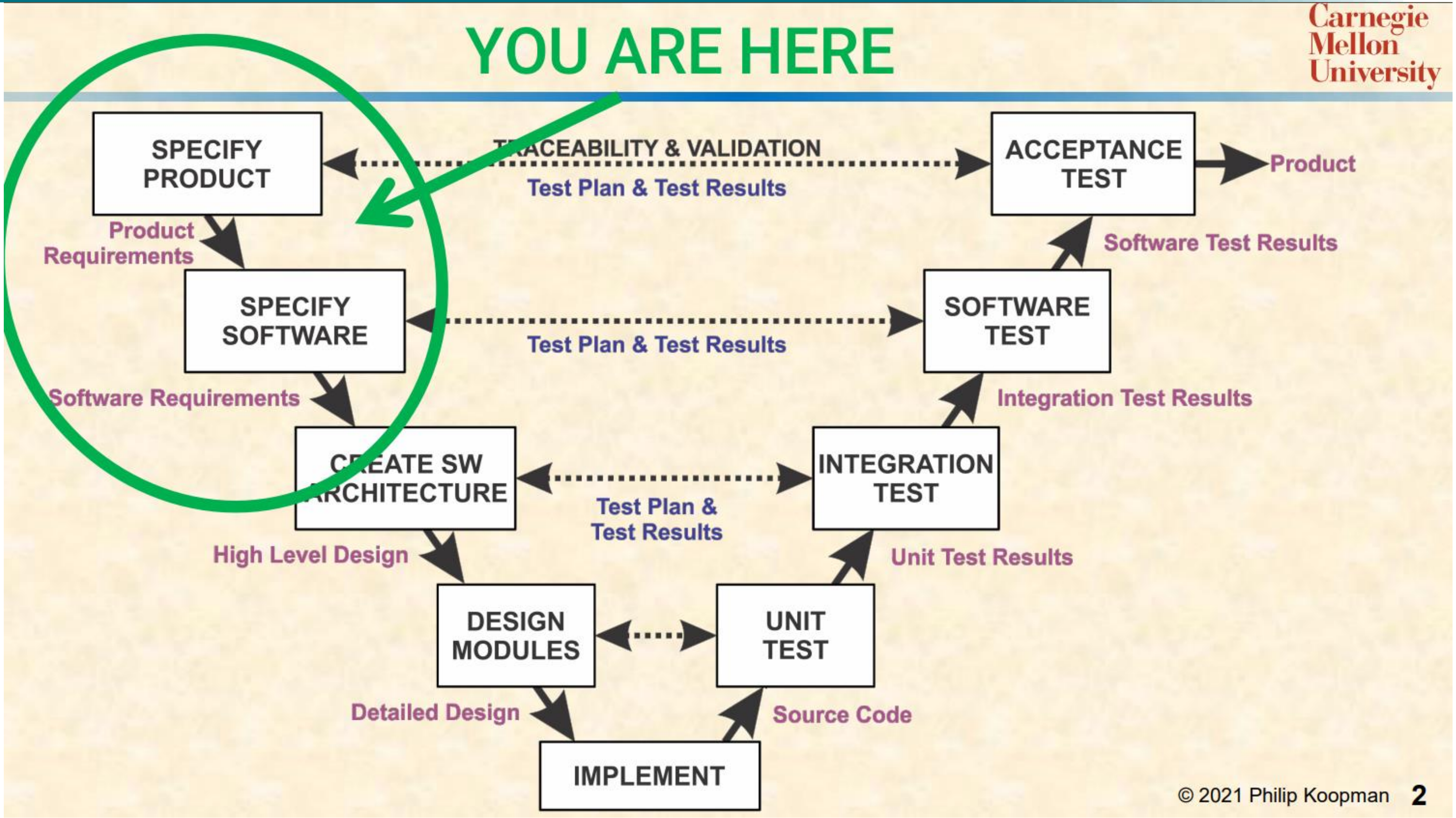
Road vehicles — Functional safety ISO 26262:2018
ISO 26262-8:2018 — Figure 2 — Structure of safety requirements

Source: https://www.iso.org search for “iso 26262:2018”

Where do I find product requirements in a project



Product requirements in the SW development V model



Source: [1]

Product requirements in the vehicle development V model

3. Concept phase

- 3-5 Item definition
- 3-6 Hazard analysis and risk assessment
- 3-7 Functional safety concept

4. Product development at the system level

- 4-5 General topics for the product development at the system level
- 4-6 Technical safety concept
- 4-7 System and item integration and testing
- 4-8 Safety validation

7. Production, operation, service and decommissioning

- 7-5 Planning for production, operation, service and decommissioning
- 7-6 Production
- 7-7 Operation, service and decommissioning

12. Adaptation of ISO 26262 for motorcycles

- 12-5 General topics for adaptation for motorcycles
- 12-6 Safety culture
- 12-7 Confirmation measures
- 12-8 Hazard analysis and risk assessment
- 12-9 Vehicle integration and testing
- 12-10 Safety validation

5. Product development at the hardware level

- 5-5 General topics for the product development at the hardware level
- 5-6 Specification of hardware safety requirements
- 5-7 Hardware design
- 5-8 Evaluation of the hardware architectural metrics
- 5-9 Evaluation of safety goal violations due to random hardware failures
- 5-10 Hardware integration and verification

6. Product development at the software level

- 6-5 General topics for the product development at the software level
- 6-6 Specification of software safety requirements
- 6-7 Software archtectural design
- 6-8 Software unit design and implementation
- 6-9 Software unit verification
- 6-10 Software integration and verification
- 6-11 Testing of the embedded software

Source: https://www.iso.org/standard/68383.html

Taxonomy of Requirements

External and Internal Quality

- **Functionality**
  - Functions
  - reaction to inputs
  - Exceptions
  - modes of operation
  - Functionality Compliance
- **Reliability**
  - Maturity
  - Fault Tolerance
  - Recoverability
  - Reliability Compliance
- **Usability**
  - Understandability
  - Learnability
  - Operability
  - Attractiveness
  - Usability Compliance
- **Efficiency**
  - Time Behavior
  - Resource Utilization
  - Efficiency Compliance
  - Performance
  - Space
- **Maintainability**
  - Analyzability
  - Changeability
  - Stability
  - Maintainability Compliance
- **Portability**
  - Adaptability
  - Installability
  - Co-existence
  - Replaceability
  - Portability Compliance
- **Organizational**
  - Delivery
  - Implementation
  - Internal standard
- **Legislative**
  - Privacy
  - Safety

Non-functional requirements

Functional requirements describe the requested functionality/behaviour of the system: services (functions), reactions to inputs, exceptions, modes of operation

Non-functional requirements represent constraints on the system and its functionality: performance constraints, compliance with standards, constraints on the development process

Source of text in black: [3], [4], [5], [6], [7]. **Source of text in blue** [8]

Requirement Engineering

WIKIPEDIA
The Free Encyclopedia

Requirements engineering (**RE**) is the process of defining, documenting, and maintaining requirements in the engineering design process
[...]
The activities involved in requirements engineering vary widely, depending on the type of system being developed and the organization's specific practice(s) involved.[6] These may include:

1. Requirements inception or requirements elicitation – Developers and stakeholders meet; the latter are inquired concerning their needs and wants regarding the software product.
2. Requirements analysis and negotiation – Requirements are identified (including new ones if the development is iterative), and conflicts with stakeholders are solved. Both written and graphical tools (the latter commonly used in the design phase, but some find them helpful at this stage, too) are successfully used as aids. Examples of written analysis tools: use cases and user stories. Examples of graphical tools: UML[7] and LML.
3. System modeling – Some engineering fields (or specific situations) require the product to be completely designed and modeled before its construction or fabrication starts. Therefore, the design phase must be performed in advance. For instance, blueprints for a building must be elaborated before any contract can be approved and signed. Many fields might derive models of the system with the Lifecycle Modeling Language, whereas others, might use UML. Note: In many fields, such as software engineering, most modeling activities are classified as design activities and not as requirement engineering activities.
4. Requirements specification – Requirements are documented in a formal artifact called a Requirements Specification (RS), which will become official only after validation. A RS can contain both written and graphical (models) information if necessary. Example: Software requirements specification (SRS).
5. Requirements validation – Checking that the documented requirements and models are consistent and meet the stakeholder's needs. Only if the final draft passes the validation process, the RS becomes official.
6. Requirements management – Managing all the activities related to the requirements since inception, supervising as the system is developed, and even until after it is put into use (e. g., changes, extensions, etc.)

These are sometimes presented as chronological stages although, in practice, there is considerable interleaving of these activities.

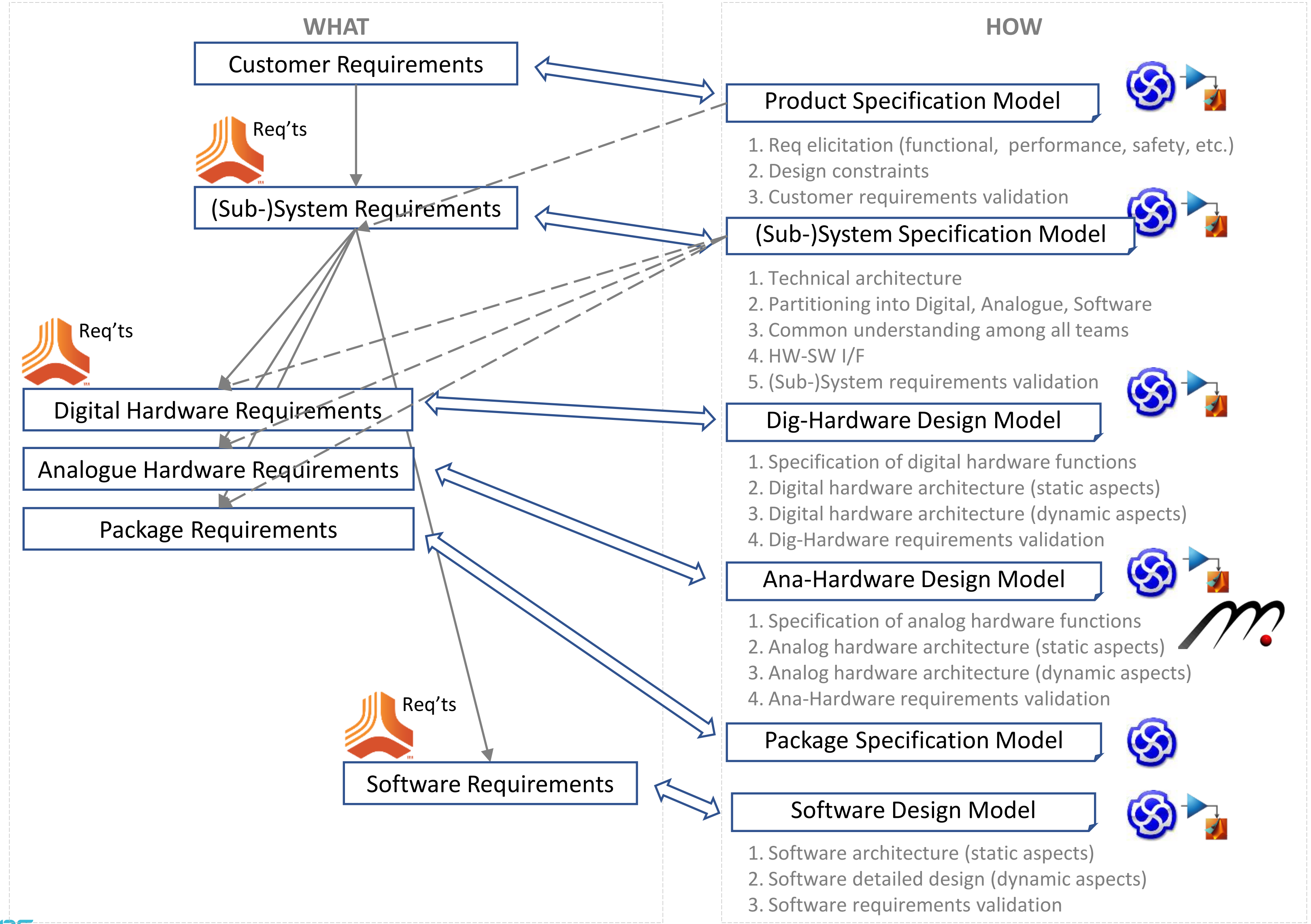
Source: https://en.wikipedia.org/wiki/Requirements_engineering

Requirement Engineering - What vs. How

Product devel. @ system level

Product devel. @ hardware level

Product devel. @ software level



Takeaway

Points #1, 2, 3 and 5 of previous slide are often successfully supported by a Model-Based System Engineering (MBSE) approach.
Models and formal languages enable automatic techniques to accomplish requirement validation and requirement analysis tasks very effectively.

Software Tools mentioned in this slide

JAMA Software. Cloud-based requirement management database.

Enterprise Architect. Semi-formal language tool and model database (SysML, UML, etc.).

MATLAB / Simulink. Block diagramming environment used to design systems with multidomain models, perform simulations before hardware, and deploy without writing code.

Modelica. Object oriented language to model cyber-physical systems. It supports acausal connection of reusable components governed by mathematical equations to facilitate modeling from first principles.

Thank you

Marco Marazza
FuSa Mgr. & MBSE expert @ESKUBE
https://www.eskube.com/
marco.marazza@eskube.com

References / Useful links

1. **Embedded Software Requirements**. Prof. Philip Koopman. Carnegie Mellon University. Lecture slides from fall 2023 (slides dated 2021). https://course.ece.cmu.edu/~ece642/lectures/07_Requirements.pdf
2. **Embedded Software Requirements**. Prof. Philip Koopman. Carnegie Mellon University. Youtube video of slides referred to at [1]. https://www.youtube.com/watch?v=P9P1iJBUDCI
3. ISO/IEC 9126-1, 2001. Information Technology - Software Engineering Product Quality. Part 1: **Quality Model**.
4. ISO/IEC 25030, 2006. Software Engineering. Software Product Quality Requirements and Evaluation (SQuaRE). **Quality Requirements**.
5. ISO/IEC CD 25010, 2007. Software Engineering. Software Product Quality Requirements and Evaluation (SQuaRE). **Quality Model and guide**.
6. **REquirements, Aspects and Software Quality: the REASQ model**. Jørgen Bøegh et. Al. Journal of Object Technology. Published by ETH Zurich, Chair of Software Engineering, ©JOT 2010 Online at http://www.jot.fm.
7. **A New Standard for Quality Requirements**. Jørgen Bøegh. March/April 2008 IEEE Software – quality requirements. https://www.researchgate.net/profile/Jorgen-Boegh/publication/3249481_A_New_Standard_for_Quality_Requirements/links/54f8ae620cf2ccffe9df5aa9/A-New-Standard-for-Quality-Requirements.pdf
8. Ian Sommerville, **Software Engineering**, 8th Ed., Addison-Wesley, 2006 and on Ch6 PowerPoint presentation from the book's web-site. https://www.cse.unr.edu/~dascalus/SE2007_08.ppt
9. The Standish Group International. **The CHAOS Report**. 1995. https://www.csus.edu/indiv/v/velianitis/161/chaosreport.pdf
10. **Requirements Engineering: Foundation for Software Quality**. Proceedings of the 29th International Working Conference, REFSQ 2023 Barcelona, Spain, April 17–20, 2023. https://2023.refsq.org/getImage/orig/978-3-031-29786-1.pdf
11. Ågren, S.M., Knauss, E., Heldal, R. et al. **The impact of requirements on systems development speed: a multiple-case study in automotive**. Requirements Eng 24, 315–340 (2019). https://doi.org/10.1007/s00766-019-00319-8

Appendix A

Why Requirements Analysis?

1. One of the primary reasons why system projects fail is because requirements of the project were not captured properly. Often during the project product development life-cycle the demands keep varying and this can also have an impact in eliciting proper requirements
2. Requirement analysis covers those tasks to determine the needs of a proposed system solution or product, often involving requirements of various stakeholders associated with the solution. Requirement analysis is a key component in the system development lifecycle and is usually the initial step before the project commences

Requirement Analysis

1. Requirements analysis is critical to the success of a development project
2. Requirements must be documented, actionable, measurable, **testable**, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design
3. Requirements can be architectural, structural, behavioral, functional, and non-functional

Common errors in requirements analysis

Problem 1: Customers don't (really) know what they want

Problem 2: Requirements change during the project lifespan

Problem 3: Customers have unreasonable timelines

Problem 4: Communication gaps exist between customers, engineers and project managers

Problem 5: The development team doesn't understand the politics of the customer's organization

Requirements Analysis and Validation

- "***Are we building the right product***"?
- i.e. will we build the product the customer truly wants to have? (at least at some point in time!)
- **Paradox**: only the customer can determine this … but the customer is non-technical!

Requirement Analysis

Requirement analysis involves several types of checks and tests that can be carried out:

- Validity
- Consistency
- Completeness
- Realism (Feasibility)
- Verifiability

Validity

- **Problem**: User may have incorrectly defined a functional requirement. All requirements must be checked for functional correctness
- **Methods**:
  - Rapid prototyping
  - Paper model
  - Animation/simulation
  - Check against existing/historic data
  - Test case generation
  - User Requirements Document review
  - System User Manual creation

Consistency

- **Problem**: User may state requirements that contradict each other (Common with many end-users!)
- E.g. year + 1 > year

  <u>year is a 2-digit number</u>

  99 + 1 = 00 > 99 contradiction!

  Simplified model of the *Year 2000 Problem*
- **Methods**:
  - If requirements are formal use constraint solvers and/or CASE tools for automatic check
  - Manual check (unclear, error prone, combinatorial explosion!)
  - Note: problem might not be solved by prototyping

Completeness

- **Problem**: user may have forgotten some requirements, leaving holes in the requirements document. These may possibly be solved arbitrarily … but possibly not!
- **Methods**:
  - Rapid prototyping
  - User Requirements Document reviews
  - Test case generation
  - Use cases analysis
  - Tables
  - Fault/ decision trees

Are Your Requirements Complete?
Donald Firesmith, Software Engineering Institute, U.S.A.
https://www.jot.fm/issues/issue_2005_01/column3/

Realism (Feasibility)

- **Problem**: User may express requirements that are either not technically feasible (e.g., performance) or violate some non-functional requirement (e.g., legislative)
- **Methods**:
  - Prototyping
  - Mathematical model/simulation (e.g., queue theory)
  - User Requirements Document reviews
  - External advice (e.g., lawyers)

Verifiability

- **Problem**: Users may state requirements which can never be checked/verified, e.g., "user interface must be user friendly and easy to use"
  - Contractual disputes may emerge
- **Methods**:
  - Test case generation, especially acceptance tests
  - Mathematical model/simulation and automatic test case generation