Requirements Management

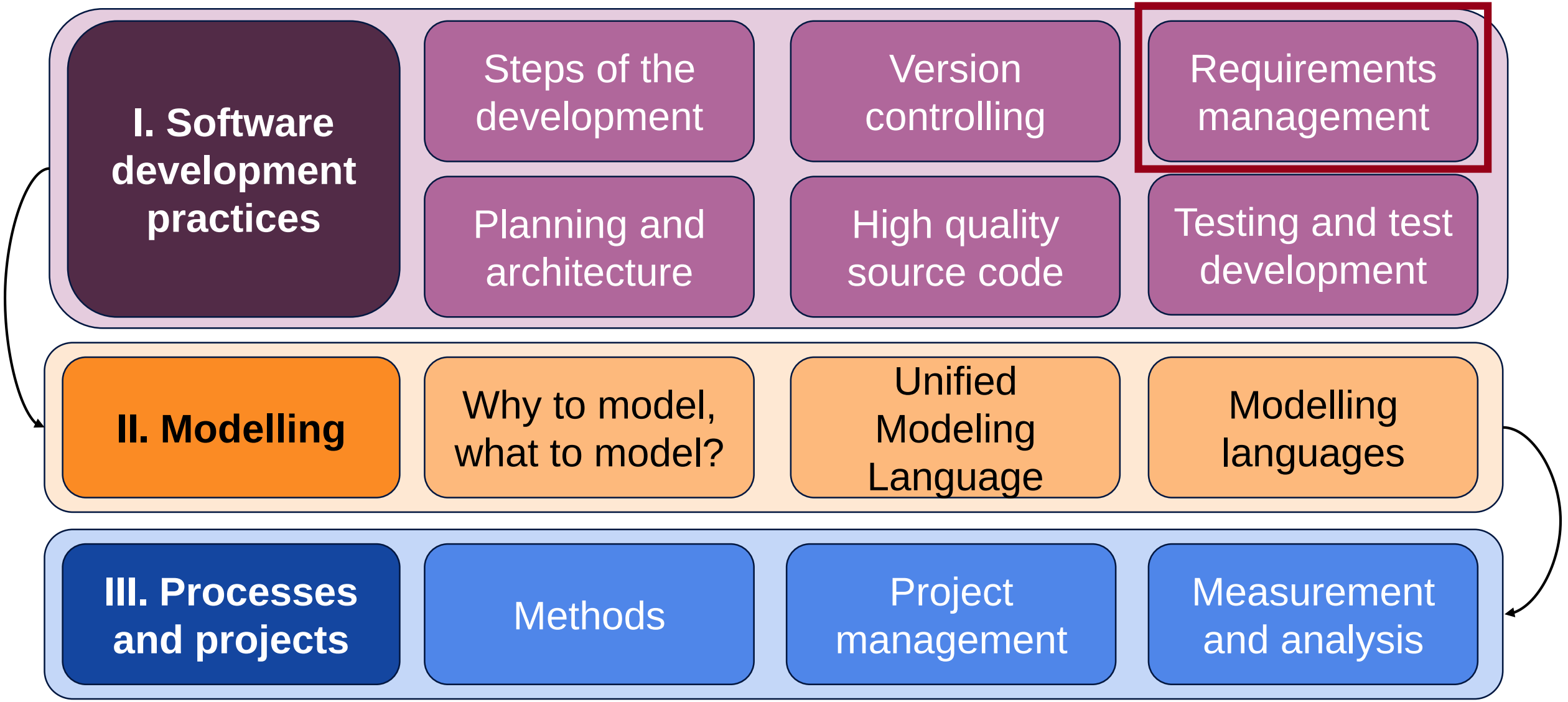
HUSZERL Gábor
huszerl@mit.bme.hu

MŰEGYETEM 1782
BME VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
Méréstechnika és Információs Rendszerek Tanszék
ftsrg Critical Systems Research Group

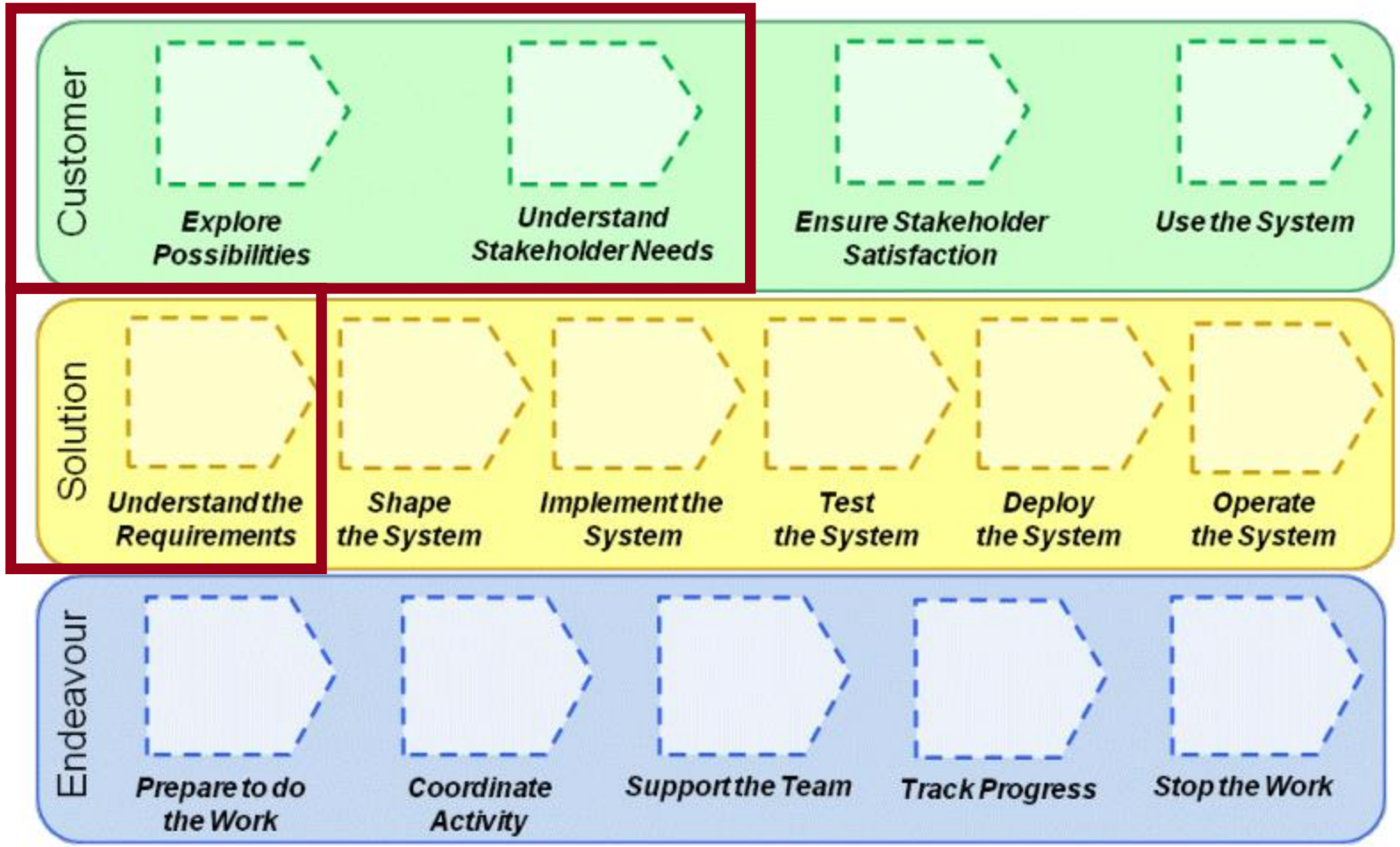
Learning Outcomes

- At the end of the lecture the students are expected to be able to
- (K1) list the possible roles and types of requirements
- (K2) summarize the possible ways of describing requirements,
- (K3) perform review of requirements.

Further Topics of the Subject



Essence: What Can Be Done on Them?



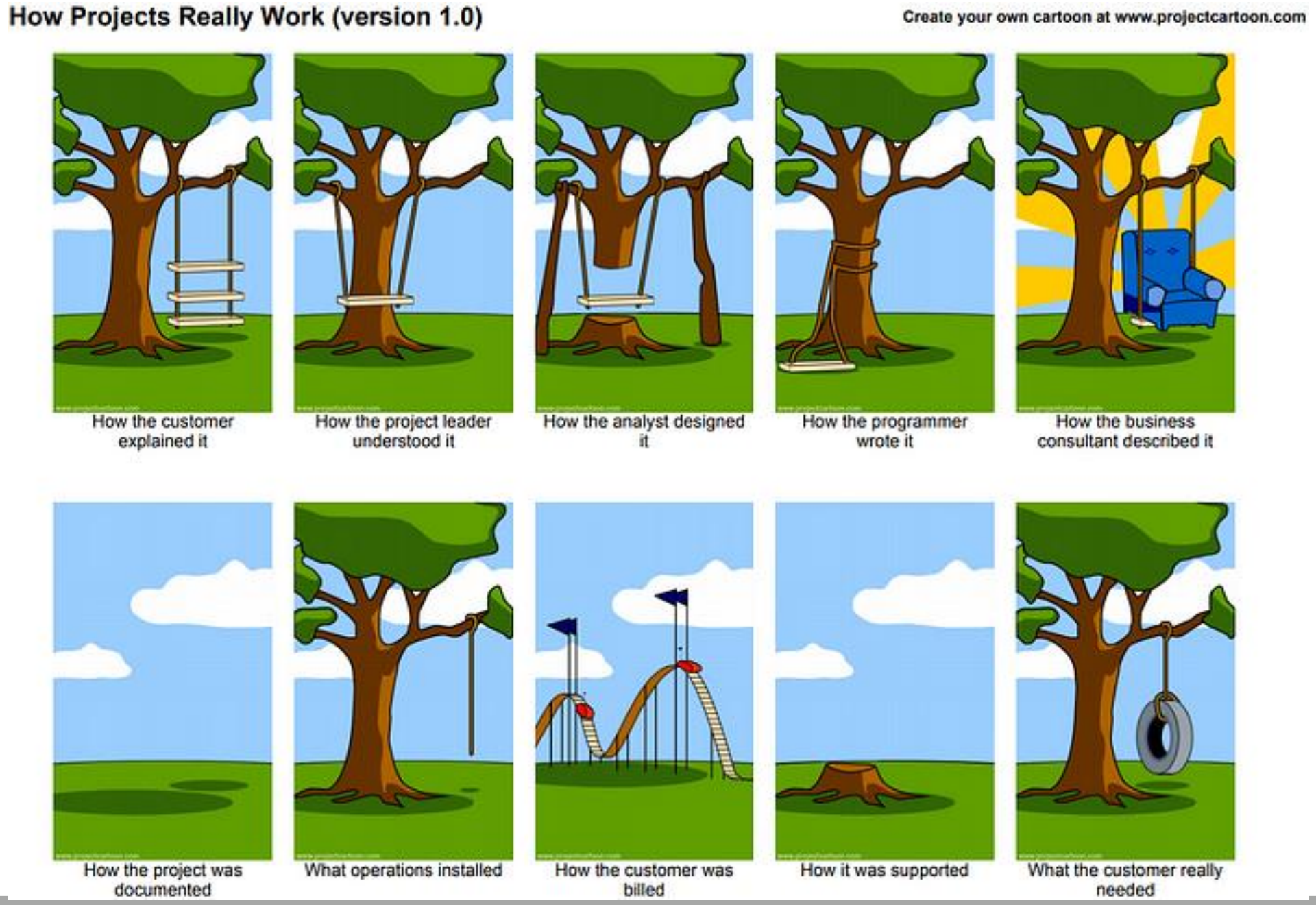
What Wants the Customer?

- He often doesn't even know exactly.

- What needs the customer?
  - And who knows that?

- Then what is to be delivered?

That is The Question!



See: tree swing picture

Requirements

Definitions, types

What Is a Requirement?

> “A condition or capability needed by a user to solve a problem or achieve an objective.” (IEEE)

> “A condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents.” (IEEE)

What Is a Requirement?

- The description of the **TARGET**, but **not of the ways and means**
- It defines the task, but not the solution

- Never ask the customer “What would you like?”
- Always ask “Why would you like that?”

- Example:
  - The customer does not want a web page with a table of the newest orders. He wants to be notified as early as possible about the new orders.
  - Maybe an app on his phone with push notifications would fit better. Collect further information first, and look after solutions only after that.

Examples for Simple Requirements

- “As a property owner, I want to place an ad in the system to sell my apartment.”
- “The monitoring system shall open the door within a maximum of 3 seconds after the emergency stop button is pushed.”
- “The new website must comply with the GDPR law.”

Principles of Requirement Management

Source: IREB, the International Requirements Engineering Board

Value Orientation	• Requirements are a means to an end, not an end in itself
Stakeholders	• RE is about satisfying the stakeholders' desires and needs
Shared Understanding	• Successful systems development is impossible without a common basis
Context	• Systems cannot be understood in isolation
Problem, requirement, solution	• An inevitably intertwined triple
Validation	• Non-validated requirements are useless
Evolution	• Changing requirements are no accident, but the normal case
Innovation	• More of the same is not enough
Systematic and disciplined work	• We can't do without in RE

Stakeholders

> “**stakeholder:** individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations”
>
> (IEEE)

Customers

Suppliers

Public authorities

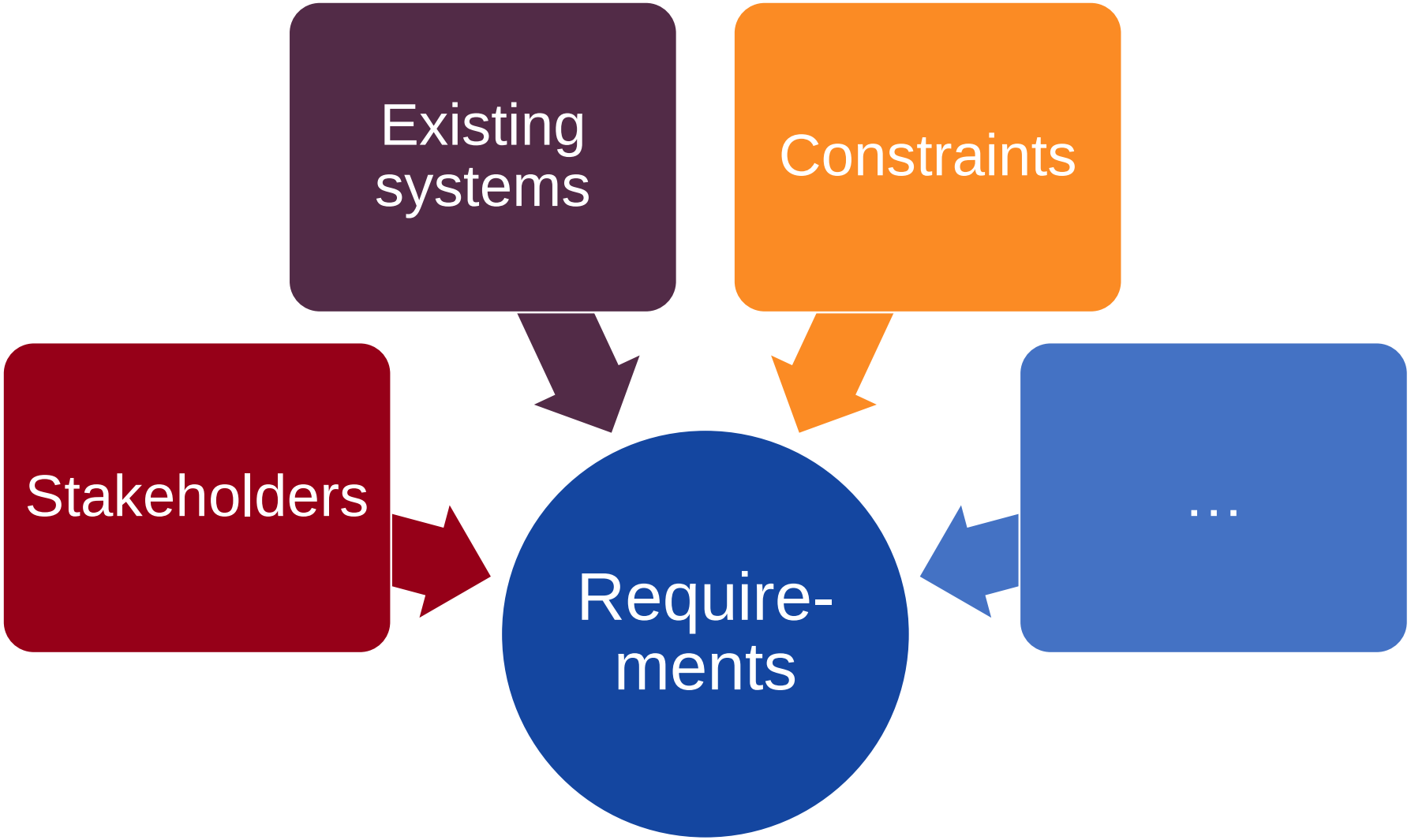
…

End users

Developers

Operators

Sources of Requirements



The System and Its Environment

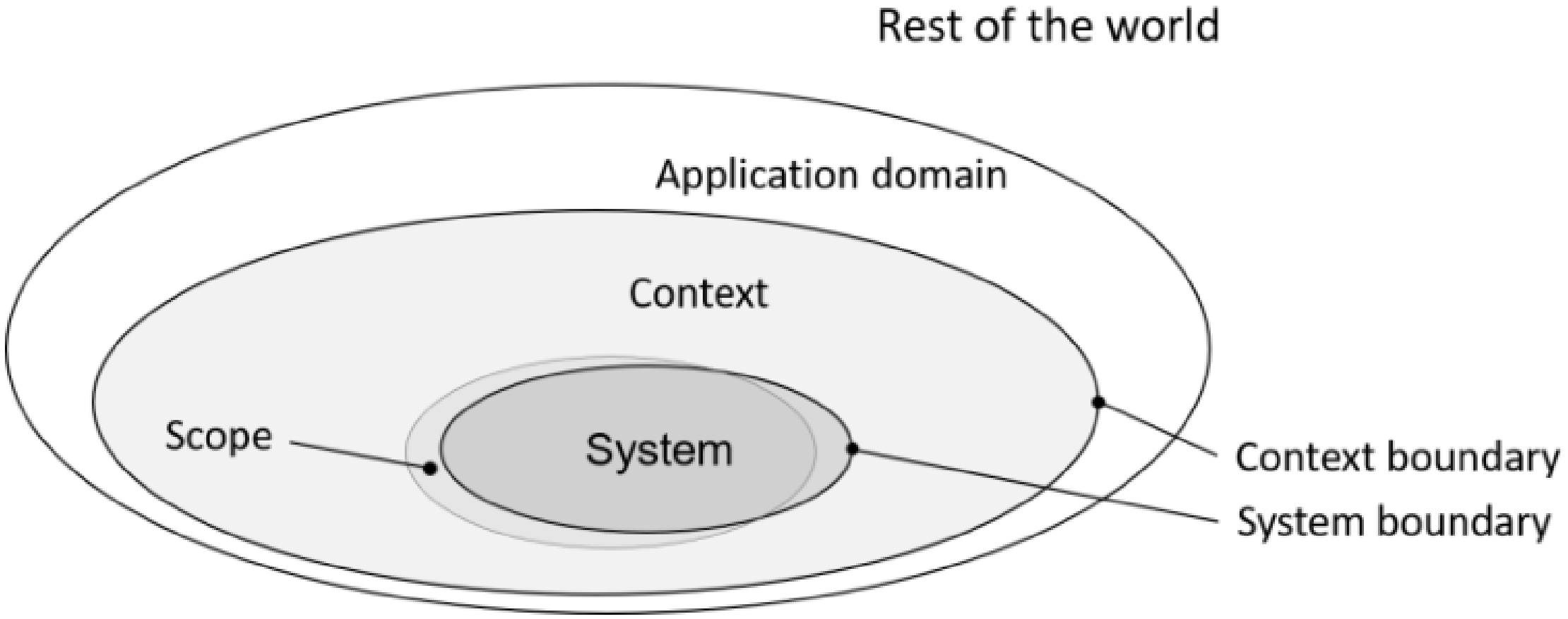


Figure 2.3 System, context, and scope

Sourcer: IREB CPRE Handbook

Types of Requirements

Functional

- **What** result or behaviour the system should produce
- That's why we make the system

Non-Functional

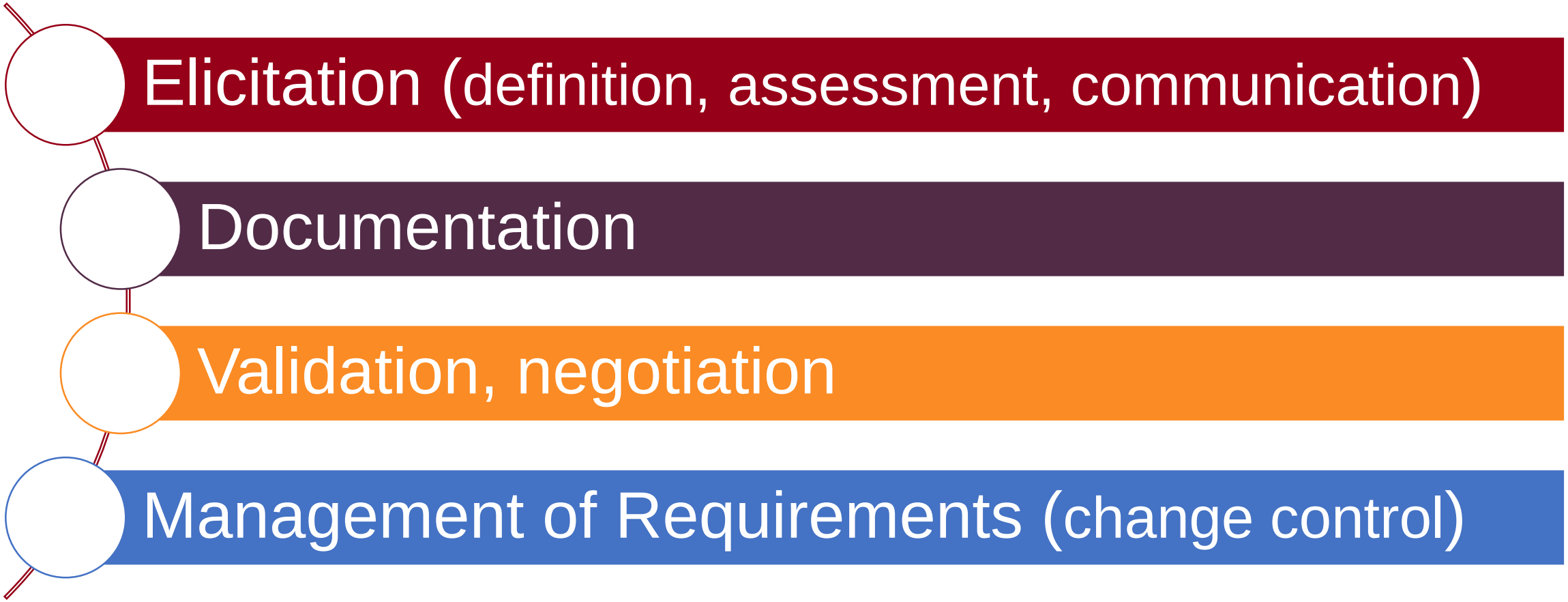
Extra-Functional

- **How**, in what quality the system should work
- Performance, reliability, ...

Examples for Requirements Types (FURPS^3)

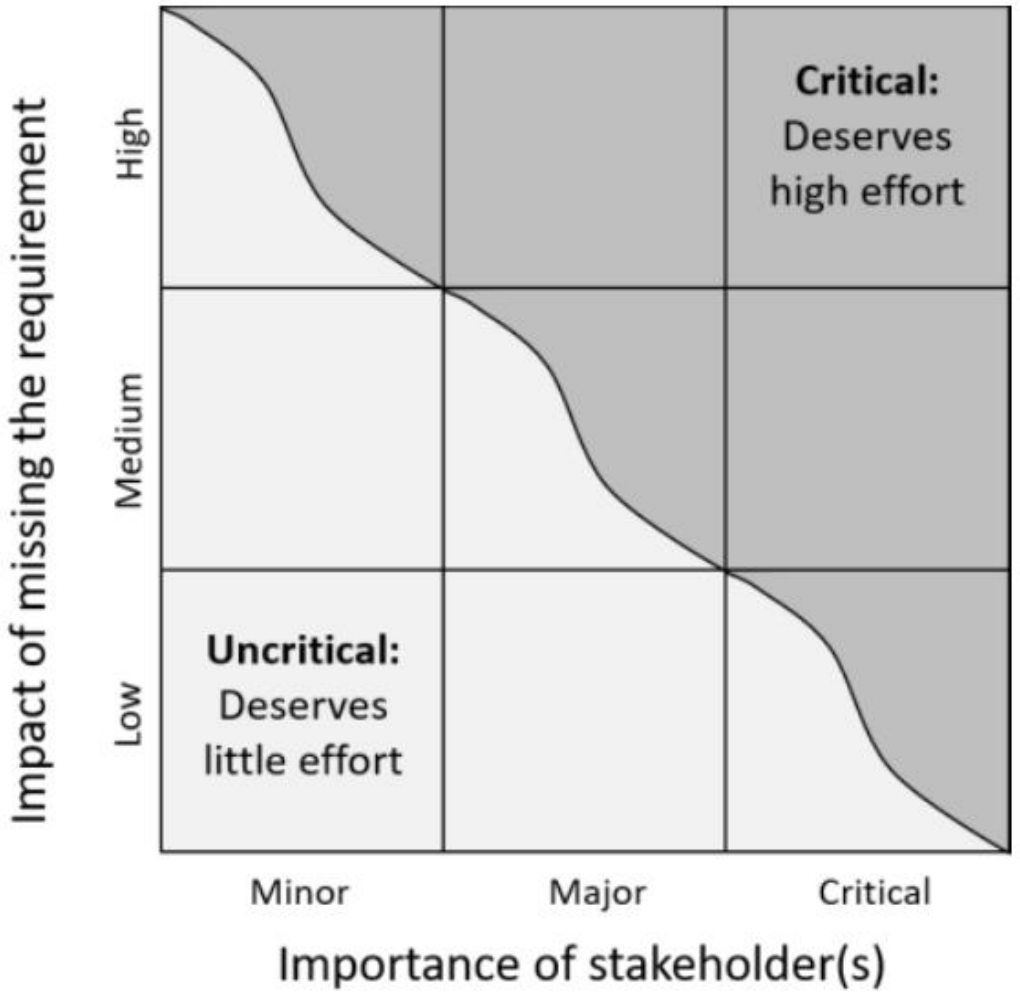
- **F**unctionality
- **U**sability: how easy/comfortable it is to use
- **R**eliability: probability of error-free operation
- **P**erformance: speed and efficiency of the system
- **S**afety: not endangering people/environment
- **S**ecurity: protection against deliberate attacks
- **S**upportability: the extent to which the system can be repaired, maintained or upgraded throughout its life cycle

Tasks of Requirements Management



Source: IREB Certified Professional for Requirements Engineering Syllabus

Prioritising Requirements



- **RISK**
- We cannot deliver everything
- What should we focus on first?

Source: IREB CPRE Handbook

Properties of Ideal Requirements

Identifiable + Unique (unique IDs)

Consistent (no contradiction)

Captured with special statements and vocabulary

Unambiguous (no different interpretations)

Verifiable (e.g. testable to decide if met)

Requirement and Specification

Requirement

- Vision, request, expectation from
  - users
  - stakeholders (authority, management, operator…)
- Basis for **validation**

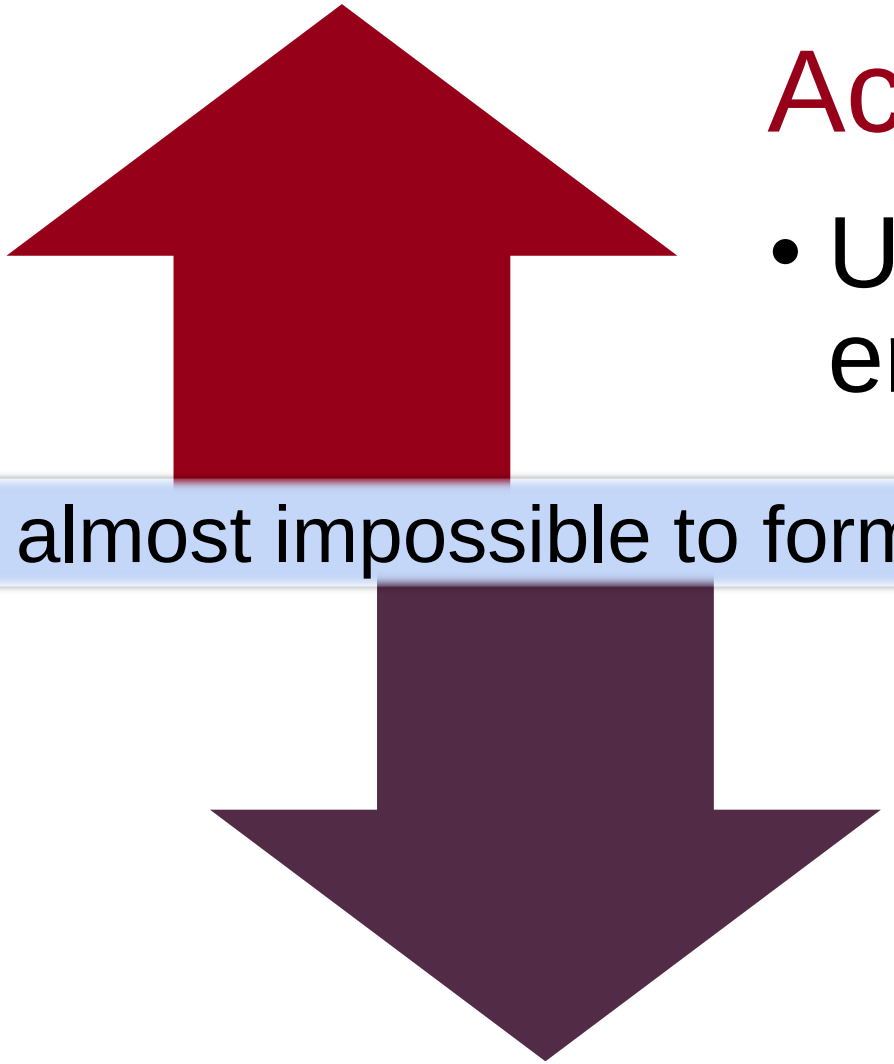
Specification

- Requests transformed for designers and developers
- Result of analysis (abstraction, structuring, …)
- Basis for **verification**

Defining Requirements

Methods, language, style

Basic Trade-Off



Accuracy

- Unambiguity, less errors, higher quality, …

It is almost impossible to formulate completely "perfect" requirements

Costs

- time, resources, efforts, …

Methods

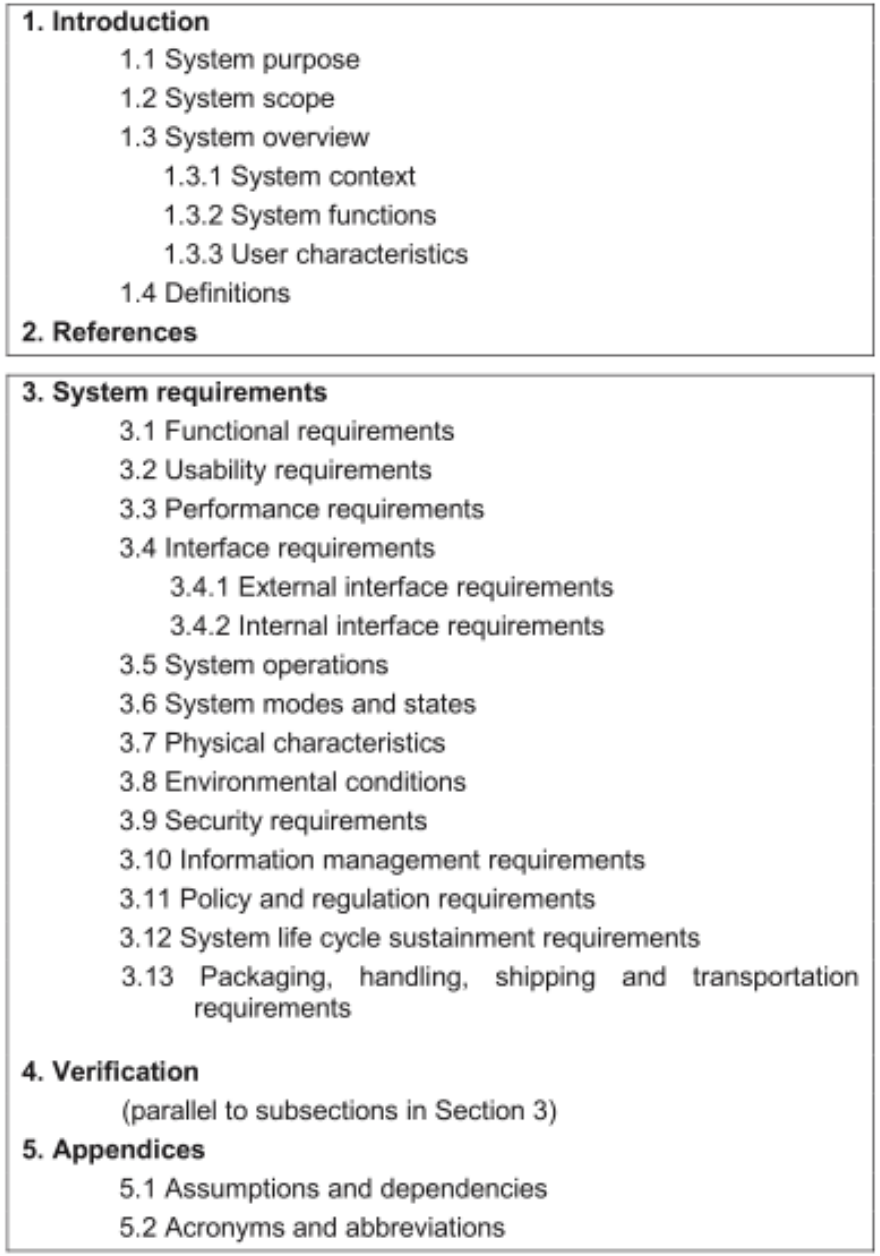
1. Informal textual description
2. User story (agile methods)
3. Requirements in critical systems

Natural Language Text

- Usually understandable,
  but not very precise

- It can be
  - informal text or
  - contract / technical specification

- Length and format environment dependent
  - From 1 page to 100(s) of pages
  - Docx / PDF / issue / wiki / …



Methods

1. Informal textual description
2. User story (agile methods)
3. Requirements in critical systems

Agile Requirements: User Stories

"As a <type of user>, I want <some goal>
so that <some reason>."

- Index card format (physical)
- Supports conversation
  (with customers/users/…)
- “Just-in-time requirements”, iterative
- Language rather business/customer than technical
- Connected to acceptance tests (→Behaviour-Driven Development)

(Many different further formats and templates possible)

Good User Story – INVEST

- **I**ndependent (from the other user stories)
- **N**egotiable (not [yet] a contract)
- **V**aluable (delivers value for the user)
- **E**stimable (with some approximation)
- **S**mall (fits into a single development iteration)
- **T**estable (also before implementation (prototyping?))

User Story – Examples

- „As a manager, I want to be able to understand my colleagues progress, so I can better report our success and failures.”

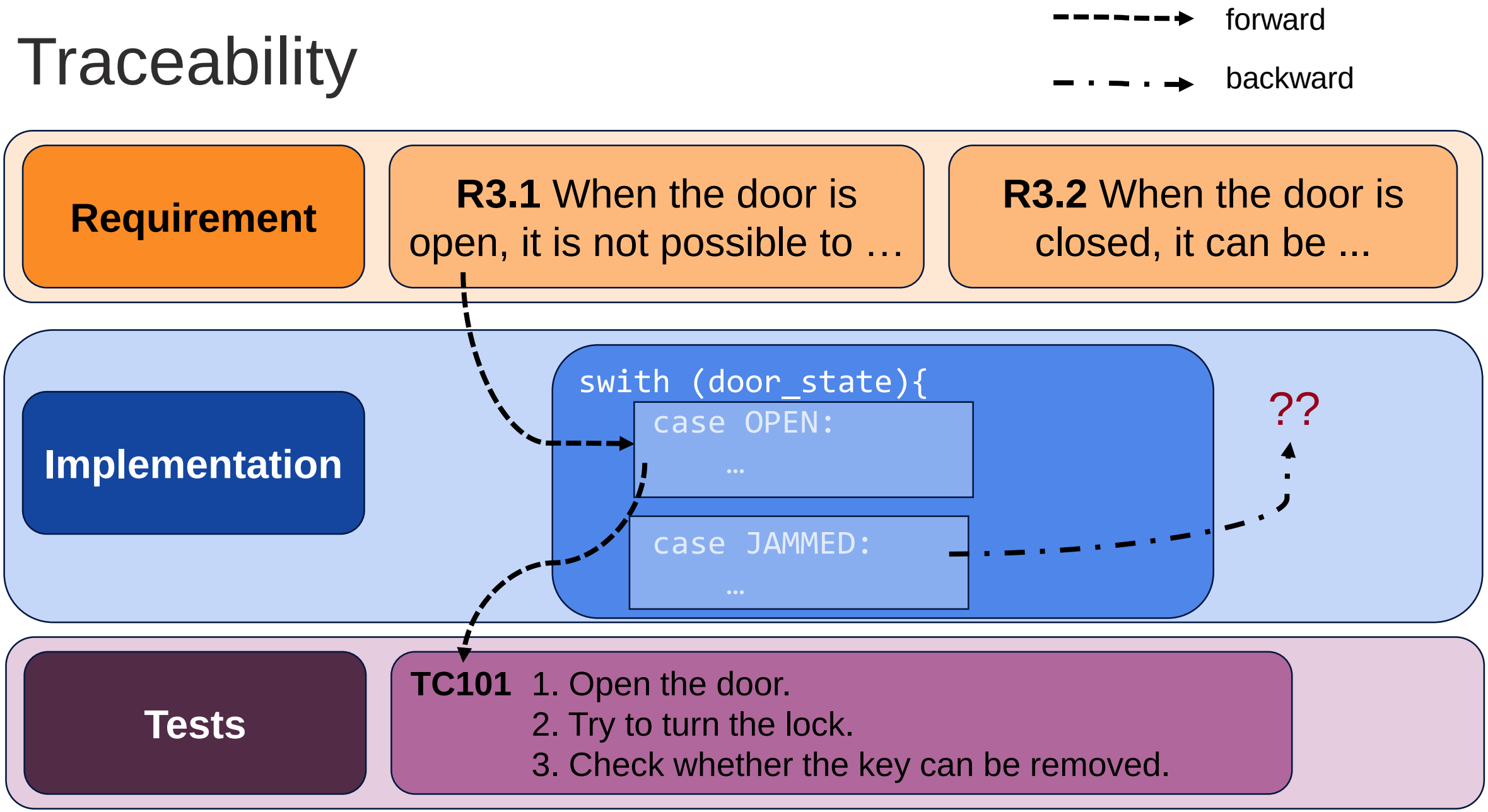
- As a trainer, I want my profile to list my upcoming classes and include a link to a detailed page about each, so that prospective attendees can find my courses.

- As a company, I can purchase a site license so that everyone in my company can take the course.

Methods

1. Informal textual description
2. User story (agile methods)
3. Requirements in critical systems

Traceability

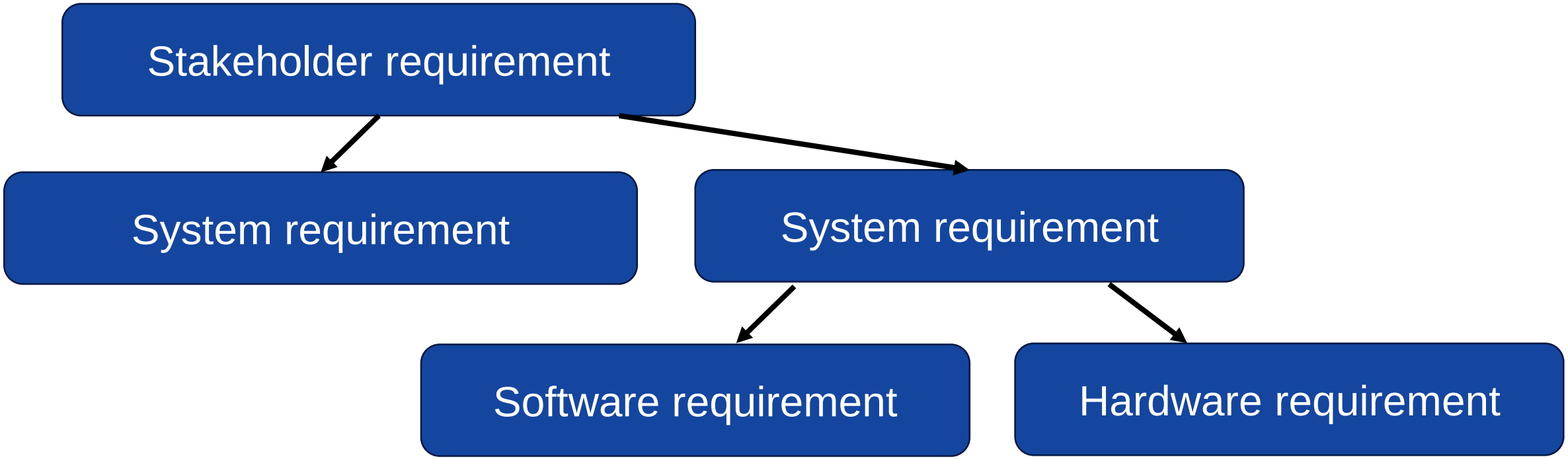


The Concept of Traceability

- Traceability is a core **certification concept**
  - For safety-critical systems
  - See safety standards: DO-178C, ISO 26262, EN 50126, …
- **Forward traceability**:
  - From each requirement to the corresponding lines of source code (and object code)
  - From each requirement to the corresponding tests and test results
  - Show responsibility
- **Backward traceability**:
  - From any lines of source code to one or more corresponding reqs
  - From any tests (results) to one or more corresponding reqs
  - No extra functionality

Hierarchy of Requirements

Often are the requirements themselves hierarchical



Requirements in Critical Systems

1. **Structured text (templates / patterns)**
2. (Graphical) semi-formal model (UML / SysML / …)
3. Formal (mathematical) description

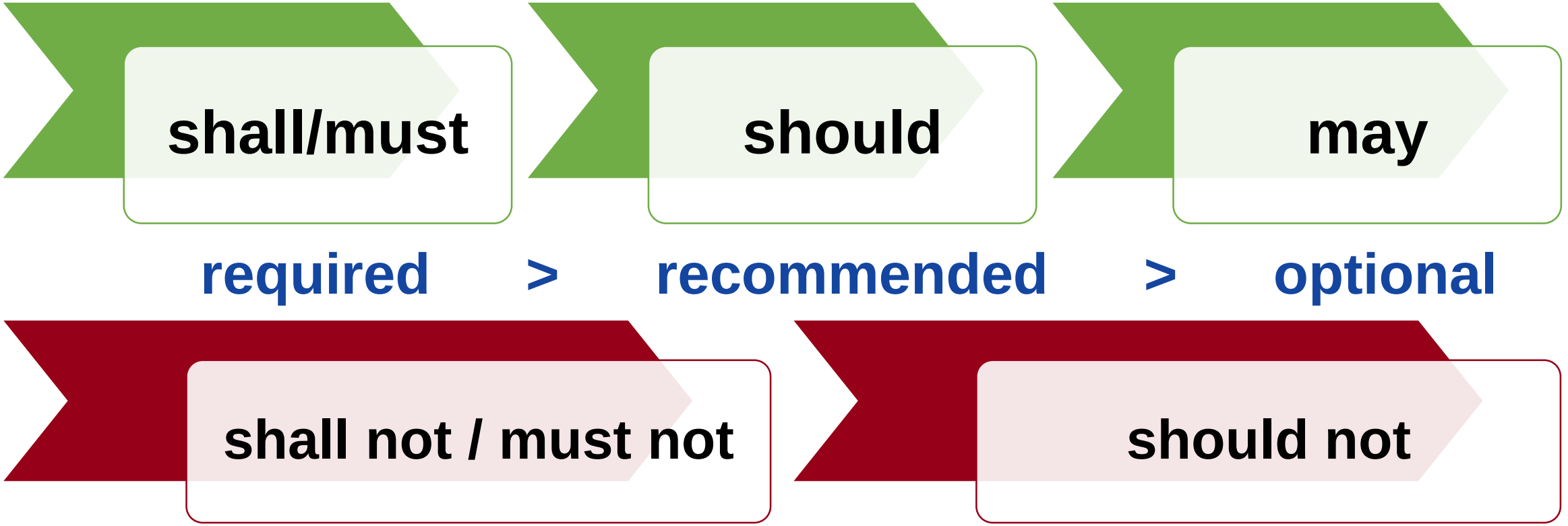
Structured Textual Requirements

A short description (stand-alone sentence / paragraph) of the problem and not the solution.

- Suggested template for (English) phrasing:
  - Pattern: Subject Auxiliary Verb Object Conditions
  - E.g.: The system shall monitor the room's temperature when turned on.

- Advices:
  - Avoid passive phrasing (lack of responsibility)
  - Use quantitative properties (measurability!)
    (e.g. the system must respond fast → in less than 5 sec)

Use of Auxiliaries (Priorities)



See: Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, IETF

Anti-patterns

1. The system should be safe
2. The system shall use Fast Fourier Transformation to calculate signal value.
3. The system shall continue normal operation soon after a failure.
4. Sensor data shall be logged by a timestamp
5. Unauthorized personnel could not access the system

Too general / high-level

Describes a solution
(and not the problem only)

Imprecise
(how to verify „soon”?)

Passive should be avoided!
(by whom?)

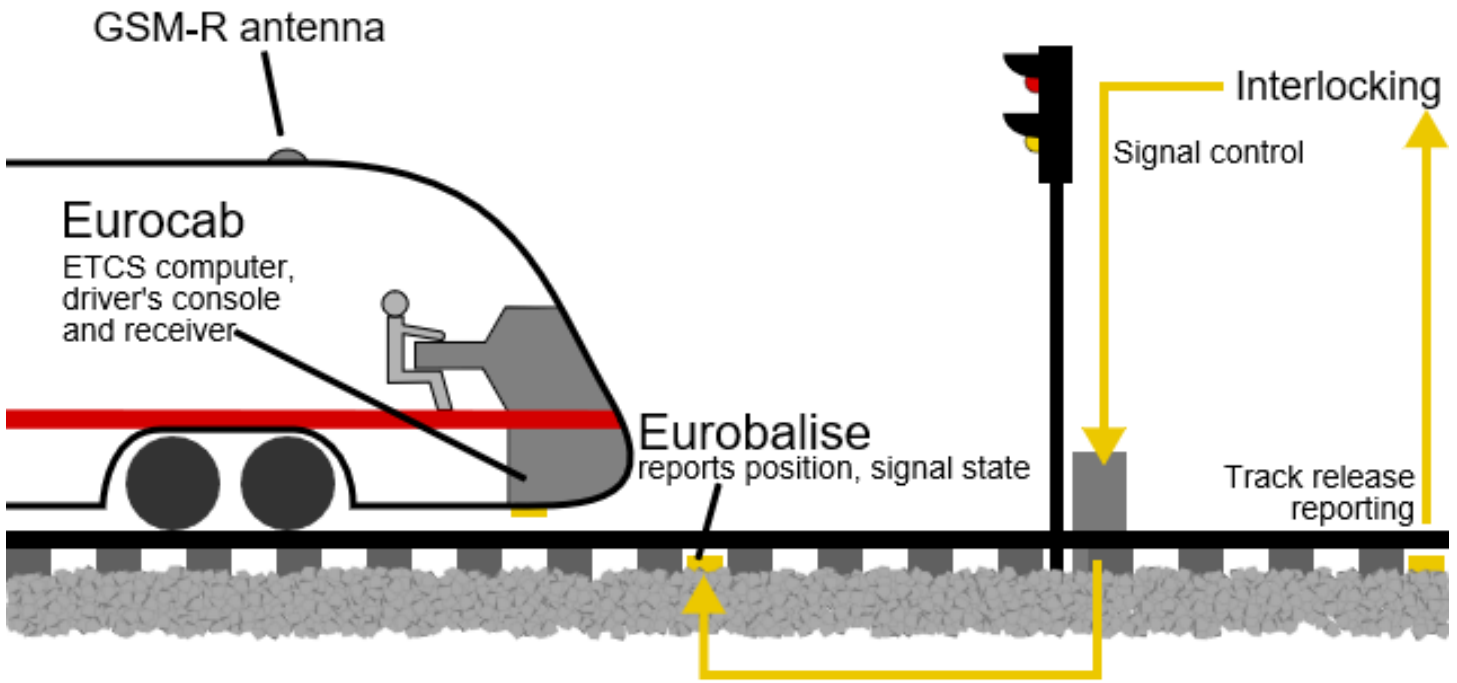
Use specific auxiliaries!

How to identify missing or inconsistent requirements?

Example Requirements: ETCS

- European Rail Traffic Management System (ERTMS)
  - European Train Control System (ETCS) + GSM-R

http://www.era.europa.eu/Core-Activities/ERTMS/Pages/Set-of-specifications-3.aspx



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

Example Requirements: ETCS

3.4.1 Balise Configurations – Balise Group Definition

3.4.1.1 A balise group shall consist of between one and eight balises.

3.4.1.2 In every balise shall at least be stored:

a) The internal number (from 1 to 8) of the balise

b) The number of balises inside the group

c) The balise group identity.

3.4.3.2 A balise may contain directional information, i.e. valid either for nominal or for reverse direction, or may contain information valid for both directions. In level 1, this information can be of the following type (please refer to section 3.8.5):

a) Non-infill

b) Intentionally deleted

c) Infill.

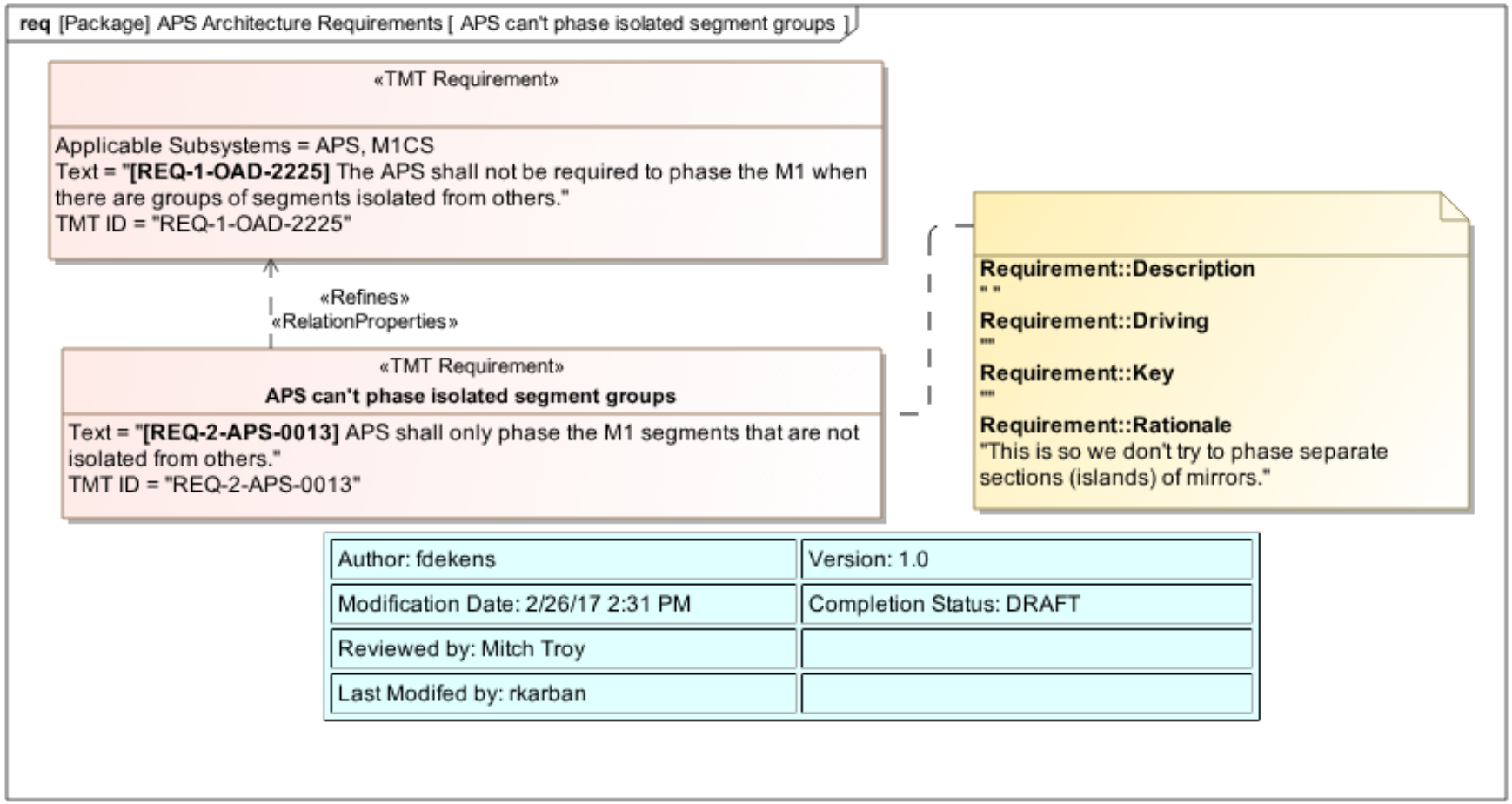
Requirements in Critical Systems

1. Structured text (templates / patterns)
2. **(Graphical) semi-formal model (UML / SysML / …)**
3. Formal (mathematical) description

Semi-Formal Description of Requirements

- More precise than natural languages
- Depicting predefined set of elements (req, actor, context, …) AND their relations AND
  formulating parameters and constraints
- See later in lectures about modelling

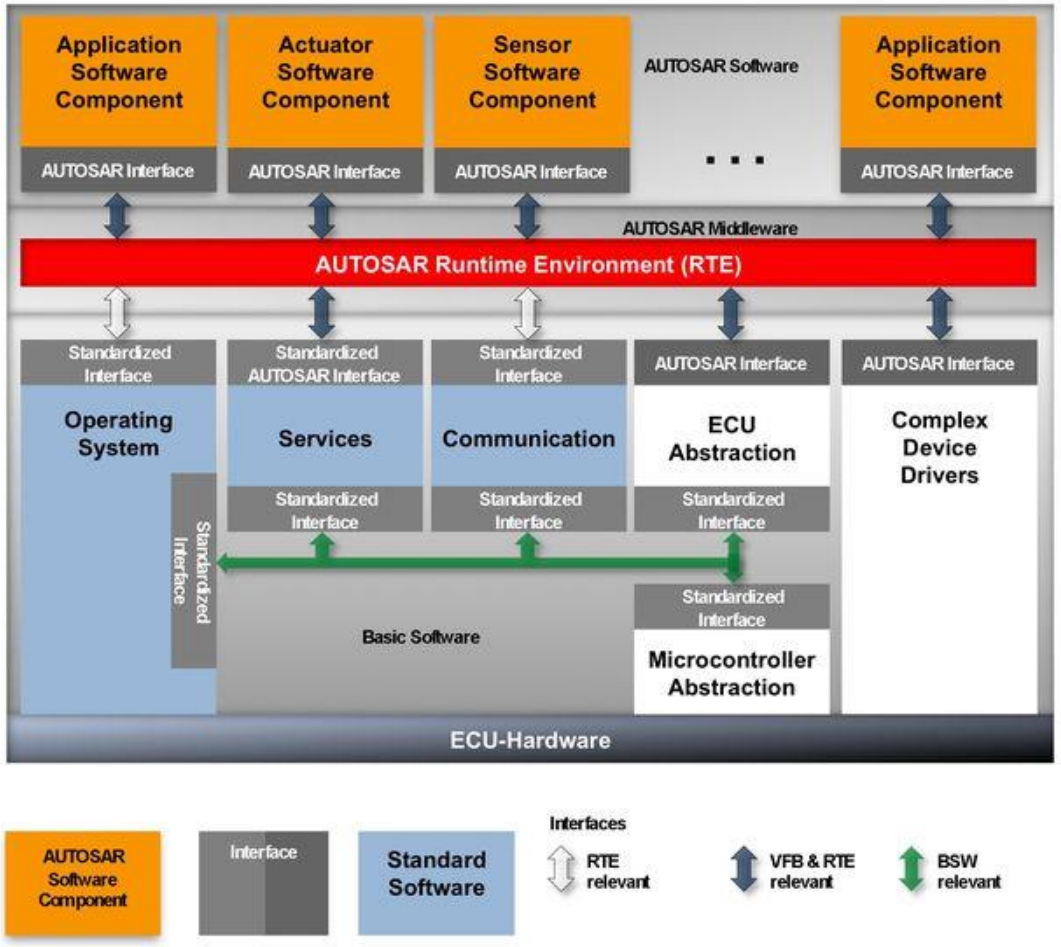
Example: Thirty Meter Telescope (TMT)



- SysML language
- Relations
  - Refinement
  - Usage
- Versioning
- Approval

See course „System Modelling”

Example Requirements: AUTOSAR



AUTomotive Open System Architecture

https://www.autosar.org/specifications/

Example Requirements: AUTOSAR

3.1 [RS_PO_00001] AUTOSAR shall support the transferability of software.

⌜

Type:	Valid
Description:	AUTOSAR shall enable OEMs and suppliers to transfer software across the vehicle network and to reuse software.
Rationale:	Transferring software across the vehicle network allows overall system scaling and optimization. Redevelopment of software is expensive and error prone.
Use Case:	Application software is reusable across different product lines and OEMs. Scaling and optimizing of vehicle networks by transferring application software. Basic software is reusable across different ECUs and domains.
Dependencies:	RS_PO_00003, RS_PO_00004, RS_PO_00007, RS_PO_00008
Supporting Material:	--

High-level requirement

3 Requirements Tracing

The following table references the requirements specified in **[RS_ProjectObjectives]** and links to the fulfilments of these.

Requirement	Description	Satisfied by
RS_PO_00001	AUTOSAR shall support the transferability of software.	RS_Main_00060, RS_Main_00100, RS_Main_00130, RS_Main_00140, RS_Main_00150, RS_Main_00270, RS_Main_00310, RS_Main_00400, RS_Main_00410, RS_Main_00440, RS_Main_00450, RS_Main_00460, RS_Main_00480

Traceability

[SWS_EcuM_03022]⌈The SHUTDOWN phase handles the controlled shutdown of basic software modules and finally results in the selected shutdown target OFF or RESET.⌋(SRS_ModeMgm_09072)

Low-level requirement

Requirements in Critical Systems

1. Structured text (templates / patterns)
2. (Graphical) semi-formal model (UML / SysML / …)
3. **Formal (mathematical) description**

Using Precise, Logical Languages

“Automatic cruise control must not be activated if the speed is less than 50 km/h or if it is raining and not on a motorway.”

What did we want to say? Are you sure?

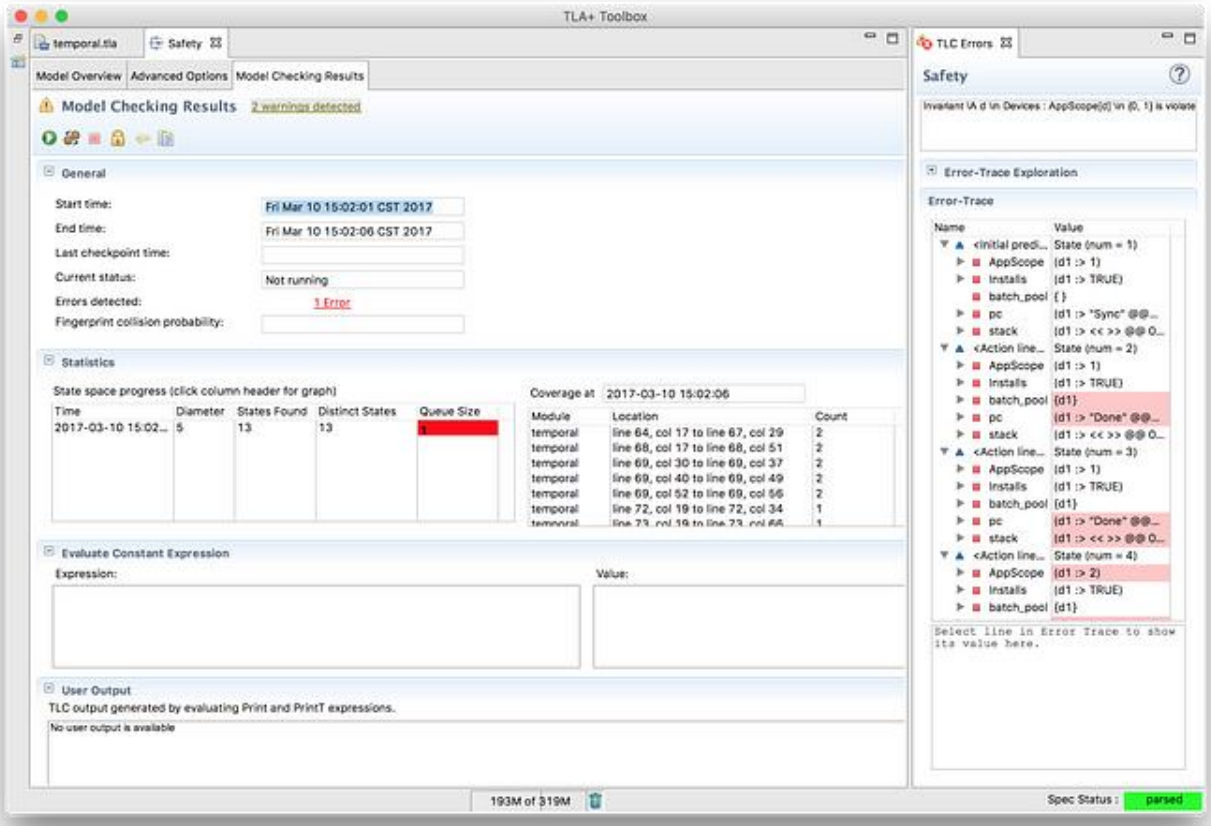
```
(SPEED < 50) OR (RAINING) AND (! HIGHWAY)
```

E.g. in a city, speed over 100 km/h, dry weather, it can be activated?

Using Precise, Logical Languages

```
1   EXTENDS Integers, TLC, Sequences
2   CONSTANTS Devices
3
4   (* --algorithm BatchInstall
5   variables
6     AppScope \in [Devices -> {0, 1}];
7     Installs \in [Devices -> BOOLEAN];
8     batch_pool = {};
9
10  define
11    PoolNotEmpty == batch_pool # {}
12  end define;
13
14  procedure ChangeAppScope()
15  begin
16    Add:
17      AppScope := [d \in Devices |->
18          IF d \in batch_pool THEN AppScope[d] + 1
19          ELSE AppScope[d]
20       ];
21    Clean:
22      batch_pool := {};
23    return;
24  end procedure;
```

Automatic checking of requirements:
Proof or generating counterexample



Source: [Formal Methods in Practice: Using TLA+ at eSpark Learning]

See MSc course „Formal Methods”

Checking Requirements

Why Is Checking Requirements Important?

The cost of correcting a mistake later is several times higher!

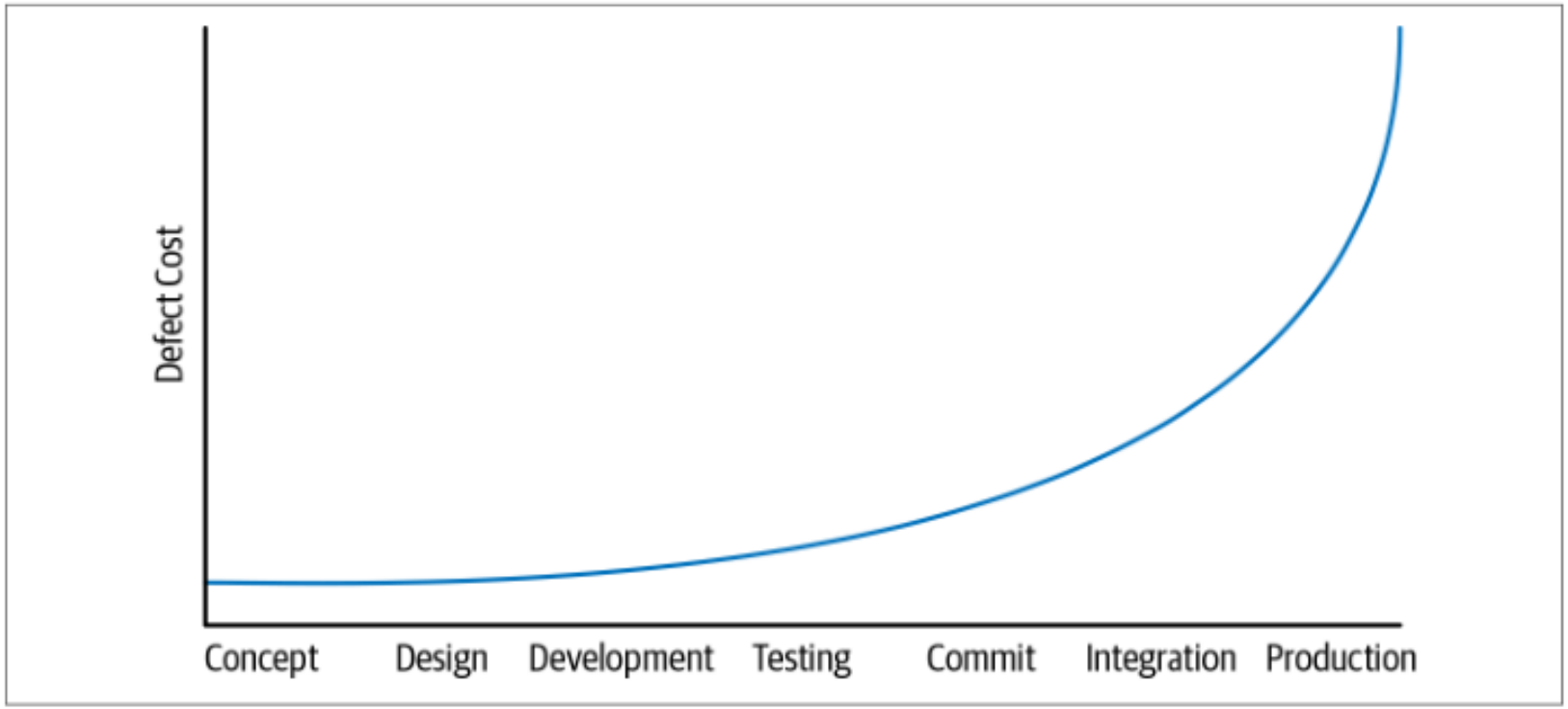


Figure 1-2. Timeline of the developer workflow

Source: Software Engineering at Google, O'Reilly, 2020

Usual Checking Methods

Review

- Manual method, executed by humans
- Reading, reviewing, analysing documents
- Generally based on structured check lists

Specification by example

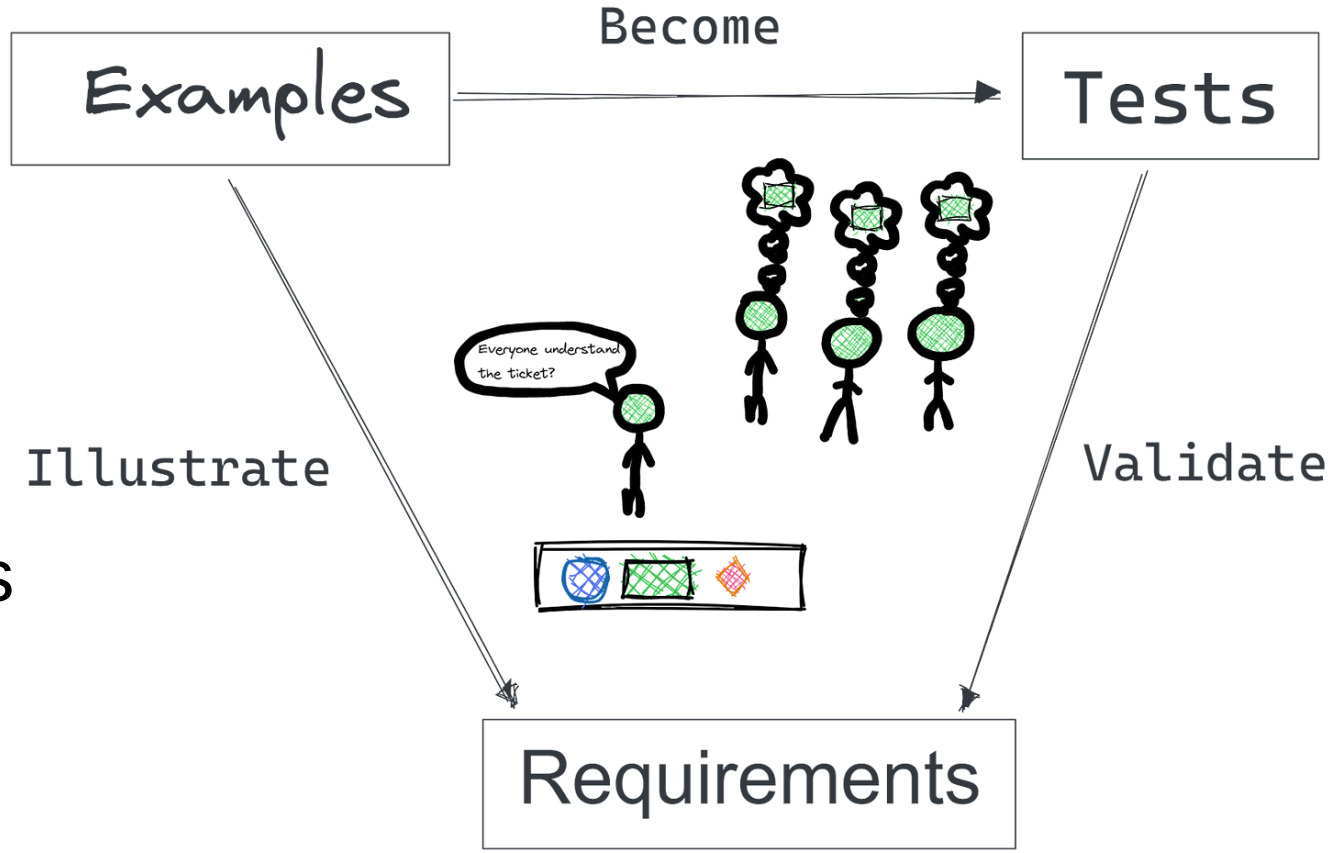
- Better understanding of requirements through examples
- Connecting requirements – examples – tests

Prototypes

- Facilitating common understanding
- Checking feasibility
- Varied levels of detail: diagrams, mock-ups, small features, ...

The Idea of “Specification by Example”

- Abstract requirements are easy to misunderstand
- Examples can help build common understanding
- Linking requirements and tests
- Collaborative process



Levels of Formality in Review

Informal review	• No formal process • Peer or technical lead reviewing
Walkthrough	• Meeting led by author • May be quite informal
Technical review	• Documented process • Review meeting with experts • Pre-meeting preparations for reviewers
Inspection	• Formal, documented process with preparation • Led by a trained moderator

Source: ISTQB CTFL

Activities of a Formal Review

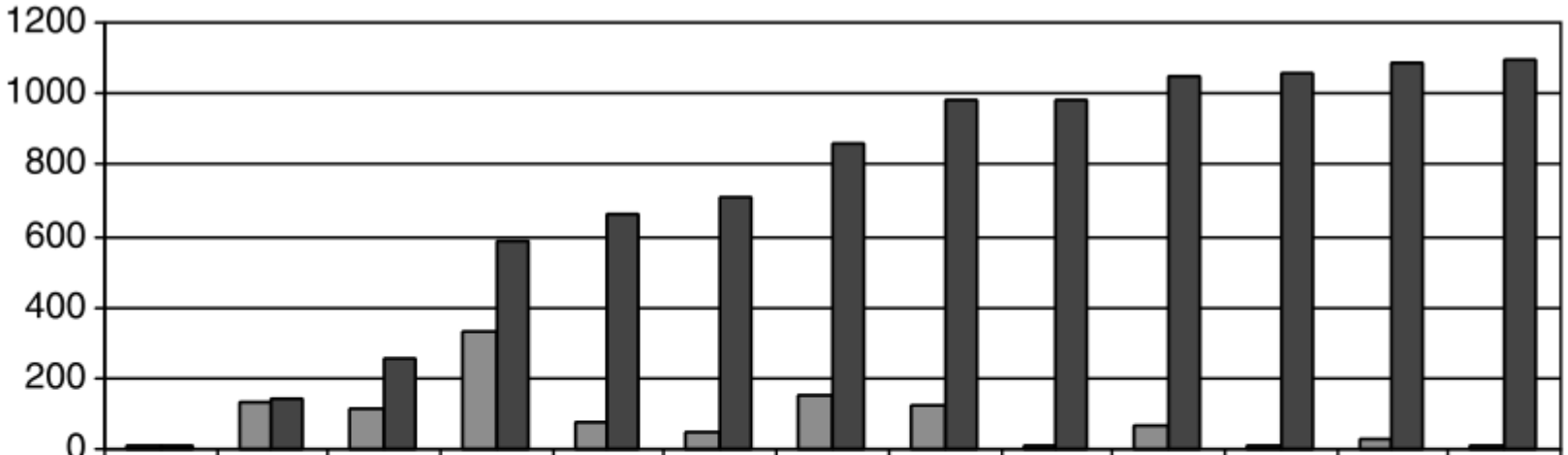
Planning	• Defining review criteria • Allocating roles
Kick-off	• Distributing documents • Explaining objectives
Individual preparation	• Reviewing artefacts • Noting potential defects, questions and comments
Review meeting	• Discussing and logging results • Noting defects, making decisions
Rework	• Fixing defects • Recording updated status
Follow-up	• Checking fixes • Checking on exit criteria

http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html

Recommendations for Reviewers

- Thorough review is time consuming
  - Usually 5-10 pages / hour
  - Can be 1 page / hour
- Increasing the number of pages to review can greatly reduce the defects found
  - Practical limits: meeting is 2 hours, max 40 pages
- (Think on this when planning your homework tasks!)

Data on Safety-critical Projects (based on a NASA project)



	fs	fs rev	des	des rev	code	code rev	ut des	ut run	int des	int run	sys des	sys run	engine
correction	10	133	109	332	78	46	149	123	7	64	7	24	13
cumulative	10	143	252	584	662	708	857	980	987	1051	1058	1082	1095

Fig. 2 Corrections found at each phase and cumulative totals

fs – functional specification
fs rev – fs review
des – design
des rev – review
ut des – unit test design
ut run – ut execution
int – integration test
sys – system test

Source: The Economics of Unit Testing, ESE 11: 5–31, 2006

Typical Review Criteria

Completeness	• Functions • References
Consistency	• Internal and external • Traceability
Implementability	• Resources • Usability, Maintainability • Risks: budget, technical, environmental
Verifiability	• Specific • Unambiguous • Measurable

Criteria from IEEE Standard 830-1998

Correct

- Every requirement stated therein is one that the software shall meet
- Consistent with external sources (e.g. standards)

Unambiguous

- Every requirement has only one interpretation
- Formal or semi-formal specification languages can help

Complete

- For every (valid, invalid) input there is specifies behavior
- TBD only possible resolution

Consistent

- No internal contradiction, terminology

Ranked for importance and/or stability

- Necessity of requirements

Verifiable

- Can be checked whether the requirement is met

Modifiable

- Not redundant, structured

Traceable

- Source is clear, effect can be referenced

Criteria from IEEE Standard 29148-2011

Necessary

- If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities

Implementation Free

- Avoids placing unnecessary constraints on the design

Unambiguous

- It can be interpreted in only one way; is simple and easy to understand

Consistent

- Is free of conflicts with other requirements

Complete

- Needs no further amplification (measurable and sufficiently describes the capability)

Singular

- Includes only one requirement with no use of conjunctions

Feasible

- Technically achievable, fits within system constraints (cost, schedule, regulatory…)

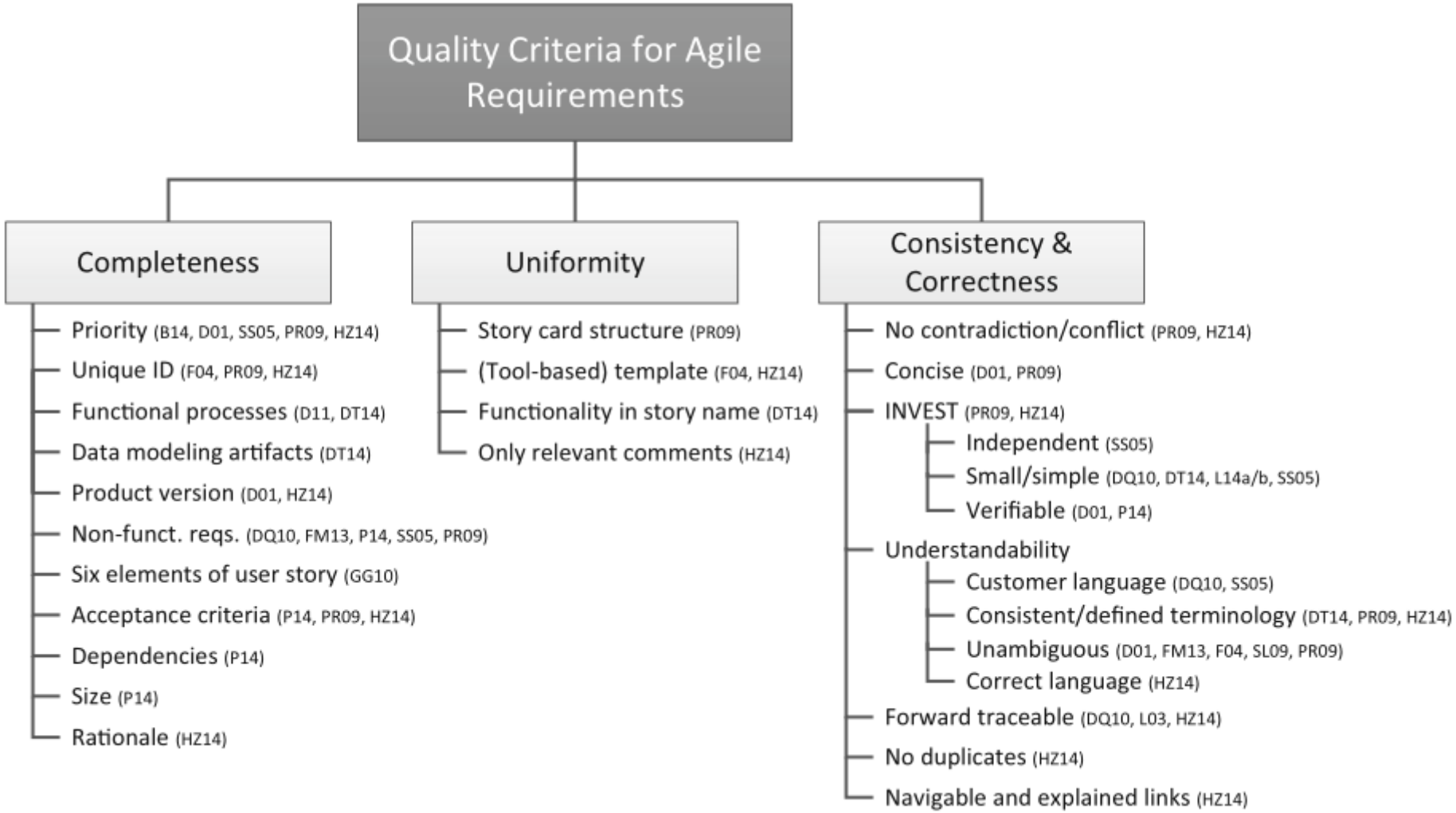
Traceable

- Upwards traceable to the stakeholder statements; downwards traceable to other documents

Verifiable

- Has the means to prove that the system satisfies the specified requirement

Quality Criteria for Agile Requirements



Source: Heck, P. & Zaidman, A. A systematic literature review on quality criteria for agile requirements specifications. Software Qual J (2016). DOI: 10.1007/s11219-016-9336-4

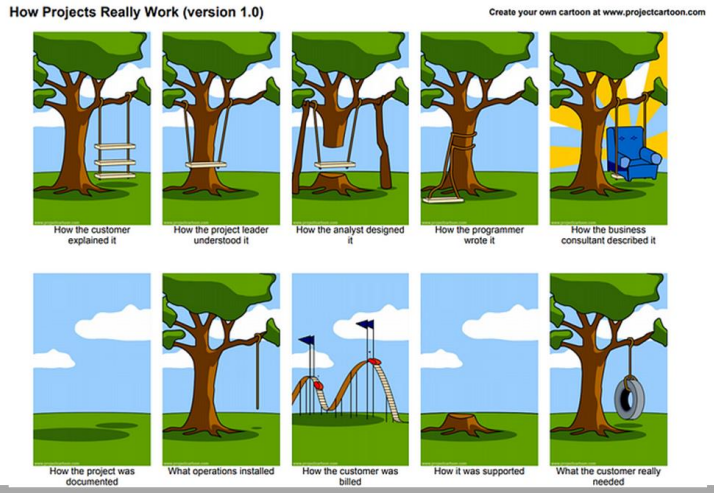
Example: Specification by Example (Gherkin)

```
Feature: Searching for books
  As a potential customer
  I want to search for books by a simple phrase
  So that I can easily allocate books by something I remember from them.

Background:
  Given the following books
    | Author         | Title                          |
    | Martin Fowler  | Analysis Patterns              |
    | Eric Evans     | Domain Driven Design           |
    | Ted Pattison   | Inside Windows SharePoint      |
    | Gojko Adzic    | Bridging the Communication Gap |

Scenario: Space should be treated as multiple OR search
  When I search for books by the phrase 'Windows Communication’
  Then the list of found books should contain only: 'Bridging the Communication Gap', 'Inside Windows
SharePoint'

Scenario: Search result should be ordered by book title
  When I search for books by the phrase 'id’
  Then the list of found books should be:
    | Title                          |
    | Bridging the Communication Gap |
    | Inside Windows SharePoint Services |
```

Feature / Scenario:
Given + When + Then

Summary

Summary



Properties of Ideal Requirements

Identifiable + Unique (unique IDs)

Consistent (no contradiction)

Captured with special statements and vocabulary

Unambiguous (no different interpretations)

Verifiable (e.g. testable to decide if met)

Basic Trade-Off

Accuracy

- Unambiguity, less errors, higher quality, …

It is almost impossible to formulate completely "perfect" requirements

Costs

- time, resources, efforts, …

Why Is Checking Requirements Important?

The cost of correcting a mistake later is several times higher!

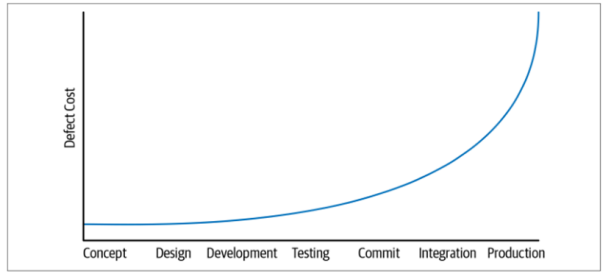


Figure 1-2. Timeline of the developer workflow

Source: Software Engineering at Google, O'Reilly, 2020