10ème édition
de la

JOURNÉE FRANCAISE DE L'INGÉNIERIE DES EXIGENCES

CFTL
Comité Français des Tests Logiciels

GABARITS D'EXIGENCES : COMMENT LES UTILISER ET QUELS SONT LES BÉNÉFICES ?

ALAIN RIBAULT

FRANÇOIS-XAVIER DE LAUNET

KEREVAL

Valeo
SMART TECHNOLOGY FOR SMARTER MOBILITY

Du 14 au 16 Novembre 2023

A partir de 11h30

En ligne

100% gratuit

IREB International® Requirements Engineering Board

GASQ

Agenda

- Requirements quality problems
- Solutions to improve requirements quality
- Requirements patterns…
- What are benefits and limits?

Requirement definition

- According to IEEE 610.12
  - “(1) A **condition or capability** needed by a **user** to solve a problem or achieve an objective
  - (2) A **condition or capability** that must be met or possessed by a **system** or **system component** to satisfy a contract, standard, specification, or other formally imposed documents
  - (3) a **documented representation** of a **condition of capability** as in (1) or (2)”

Quality of a requirement

INCOSE	IREB	AGILITE (User Story)
• VERIFIABLE	• ADEQUATE	• INDEPENDANT
• NECESSARY	• NECESSARY	• NEGOCIABLE
• IMPLEMENTATION INDEPENDANT	• UNAMBIGUOUS	• VALUABLE / VERTICAL
• UNAMBIGUOUS	• COMPLETE (SELF-CONTAINED)	• ESTIMABLE
• SINGULAR	• UNDERSTANDABLE	• SMALL
• FEASIBLE	• VERIFIABLE	• TESTABLE
• CORRECT		
• TRACEABLE		
• SELF SUPPORTING		

Exemple of requirement defects

Defect type	Defect description	Bad requirement	Good requirement
Not feasible	A requirement can be realized within defined constraints (cost, schedule, regulatory…).	The WIC_System shall have 100% availability	The WIC_System shall have 98% availability
Inconsistency	The Information contained in the requirement is inconsistent	All lights shall be green at 550nm < λ < 554nm. LED 1 shall have 634nm +/-2nm.	All lights shall be green at wavelength 550nm < λ < 554nm. LED 1 shall have 552nm +/-2nm to compensate lasering process.
Vagueness Ambiguity Unclarity Immeasurability	Ambiguous words, which do not have quantifiable meanings (Easy, careful, quickly, complete, if required, Ad hoc, …)	My Mercedes climate control rotary knob shall have similar noise and feeling as a bank safe deposit box	The operator shall turn the button with Fmax= 10N

Exemple of requirement defects

JOURNÉE FRANCAISE DE L'INGÉNIERIE DES EXIGENCES 2023

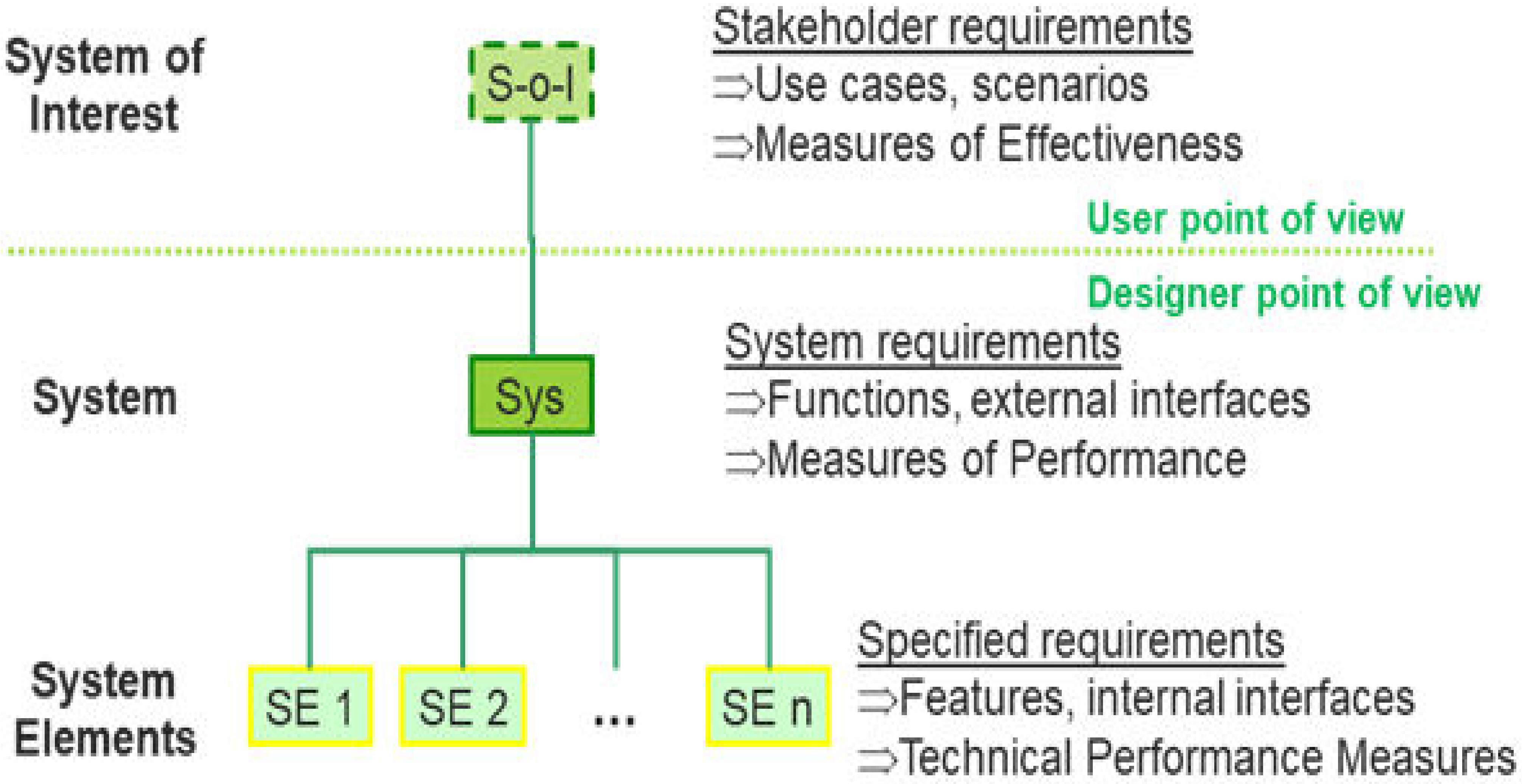
Defect type	Defect description	Bad requirement	Good requirement
Incompleteness	The requirement or relevant information is missing	The system shall allow authentication of authorized flashing system.	The system shall allow authentication by signature of authorized flashing system. The system shall send negative response in case of wrong signature.
Weakness	The requirement contains weak main verb such as: Can; Could; May; Might	The sensor might measure up to 80°C.	The sensor should measure up to 80°C.
Multiplicity	more than one main verb or more than one subject.	REQ_1: The motor runs on 1500 RPM+/-10% and is supplied by 5V+/-10%	REQ_1: The motor runs on 1500 RPM+/-10%. REQ_2: The motor is supplied by 5V+/-10%.
Typo or formatting error	Orthographic, semantic, multiple negation, passive	Training rewards and points will not be visible to users who cannot participate in training rewards.	Training rewards and points will only be shown to people participated the training

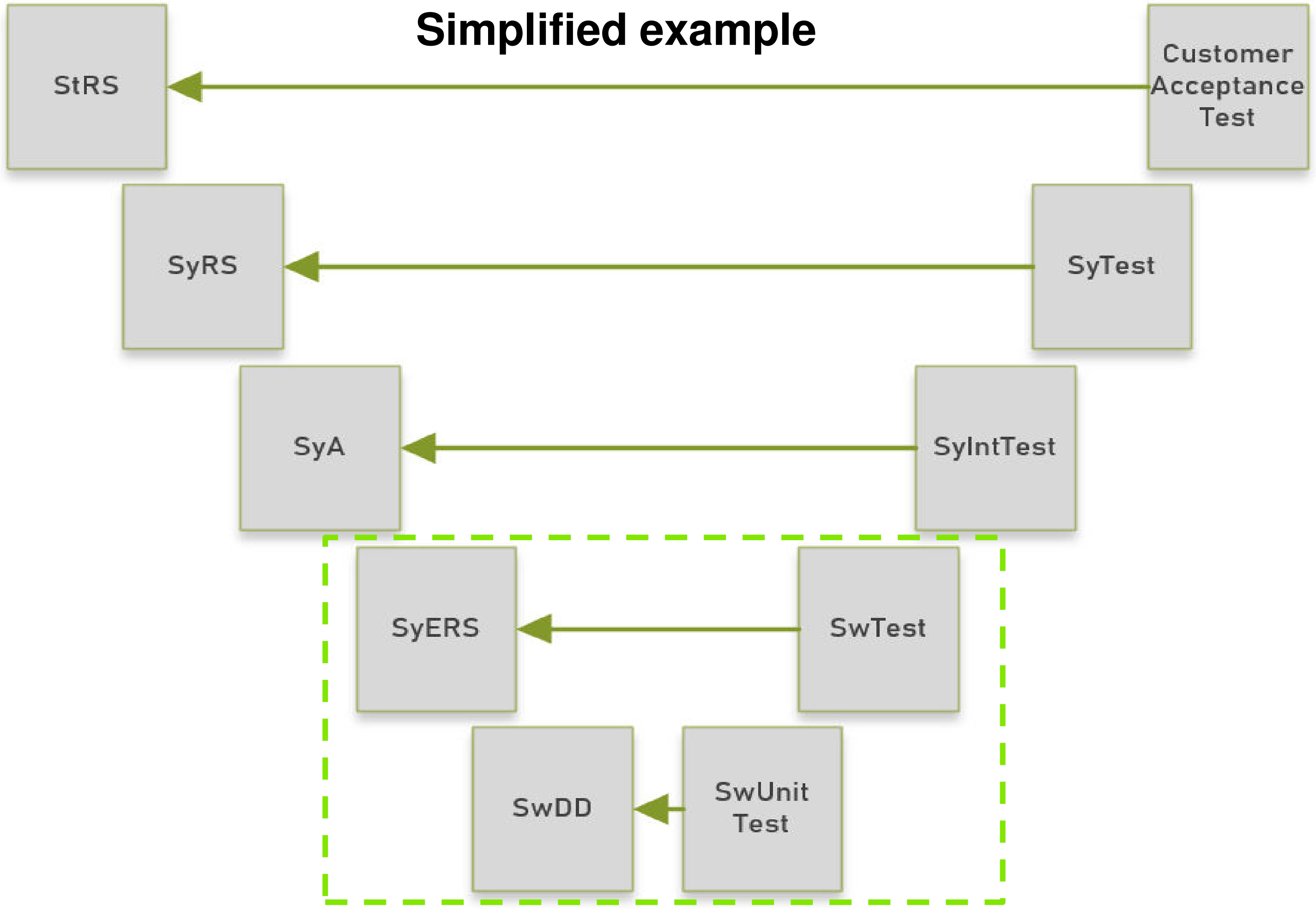
Solutions to improve requirements quality?

- Follow Requirements Development process / sub-process
  - Elicitation ➔ clarification of needs into requirements
  - Analysis ➔ refinement, consistency, completeness, use of models
  - Documentation
    - Levels of formalization (non-formal, semi-formal, formal)
    - ***Use of patterns***, tools
  - Verification and validation ➔
    - Quality criteria
      - Requirements quality criteria (ISO, INVEST, INCOSE)
      - Specifications quality criteria (ISO, INCOSE, SEBOK)

Requirement engineering in SE

JOURNÉE FRANCAISE DE L'INGÉNIERIE DES EXIGENCES 2023

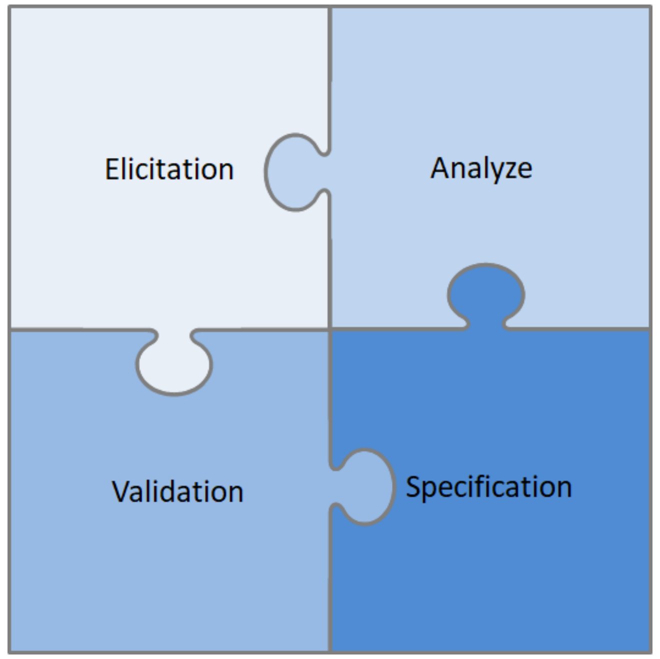
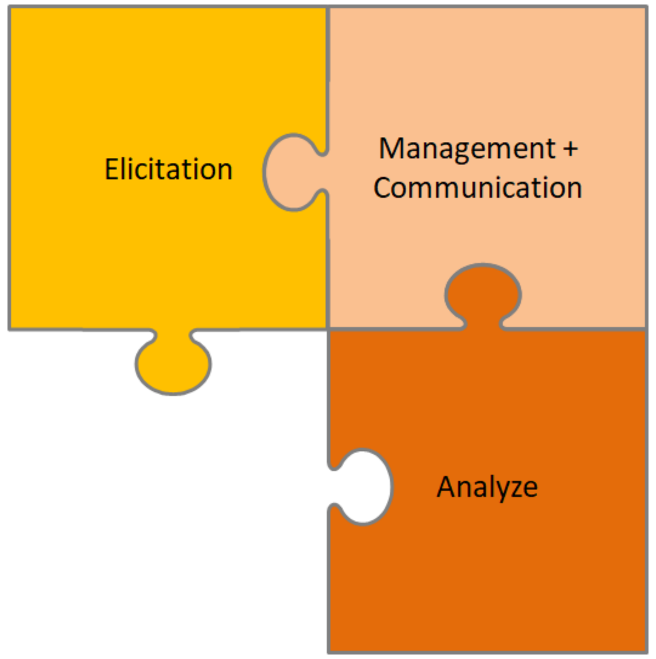
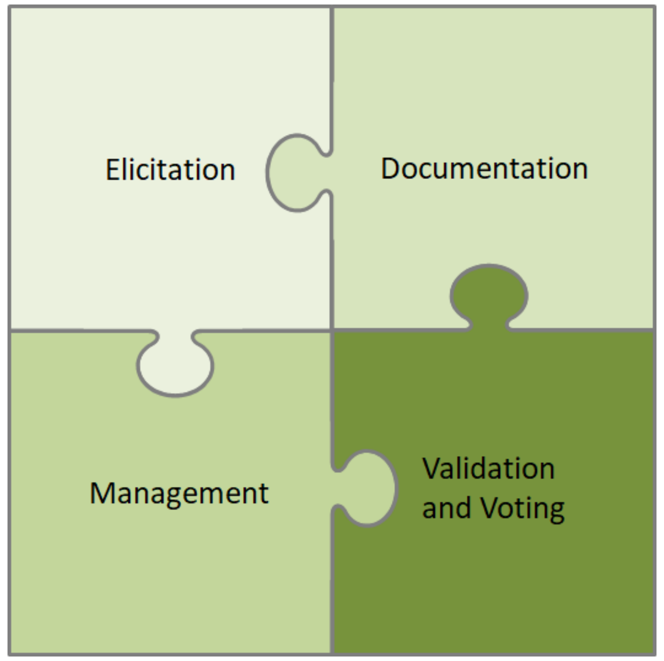
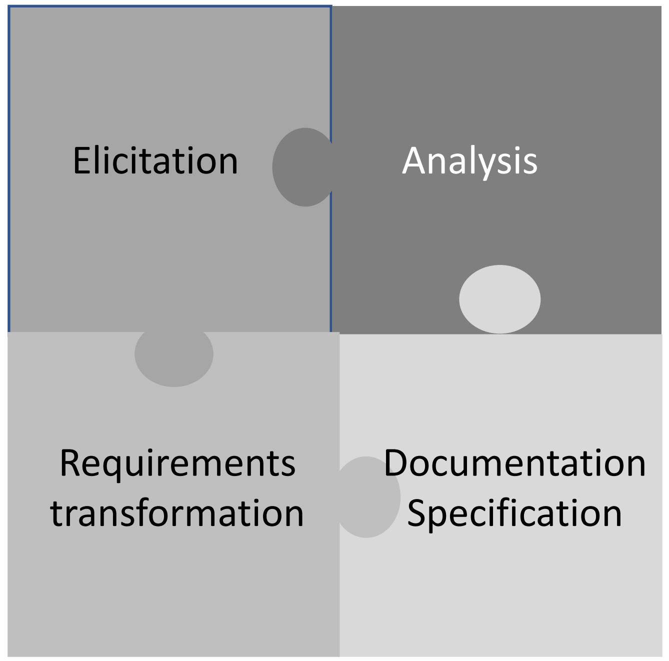




- StRS: Stakeholder Requirements Specification
- SyRS: System Requirements Specification
- SyA(D): System Architecture
- SyERS: System Element Requirement Specification
- SwDD: Software Detailed Design

Requirement engineering principles

Requirement Engineering is a systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreements between customer and the project team on the changing requirements of the system.

IEEE 24765:2017 Vocabulary for ISO/IEC/IEEE 15288	IIBA International Institute for Business Analysis	INCOSE, IREB International Requirements Engineering Board	IEEE 29148:2018 Requirements Engineering
 Elicitation Analyze Validation Specification	 Elicitation Management + Communication Analyze	 Elicitation Documentation Management Validation and Voting	 Elicitation Analysis Requirements transformation Documentation Specification

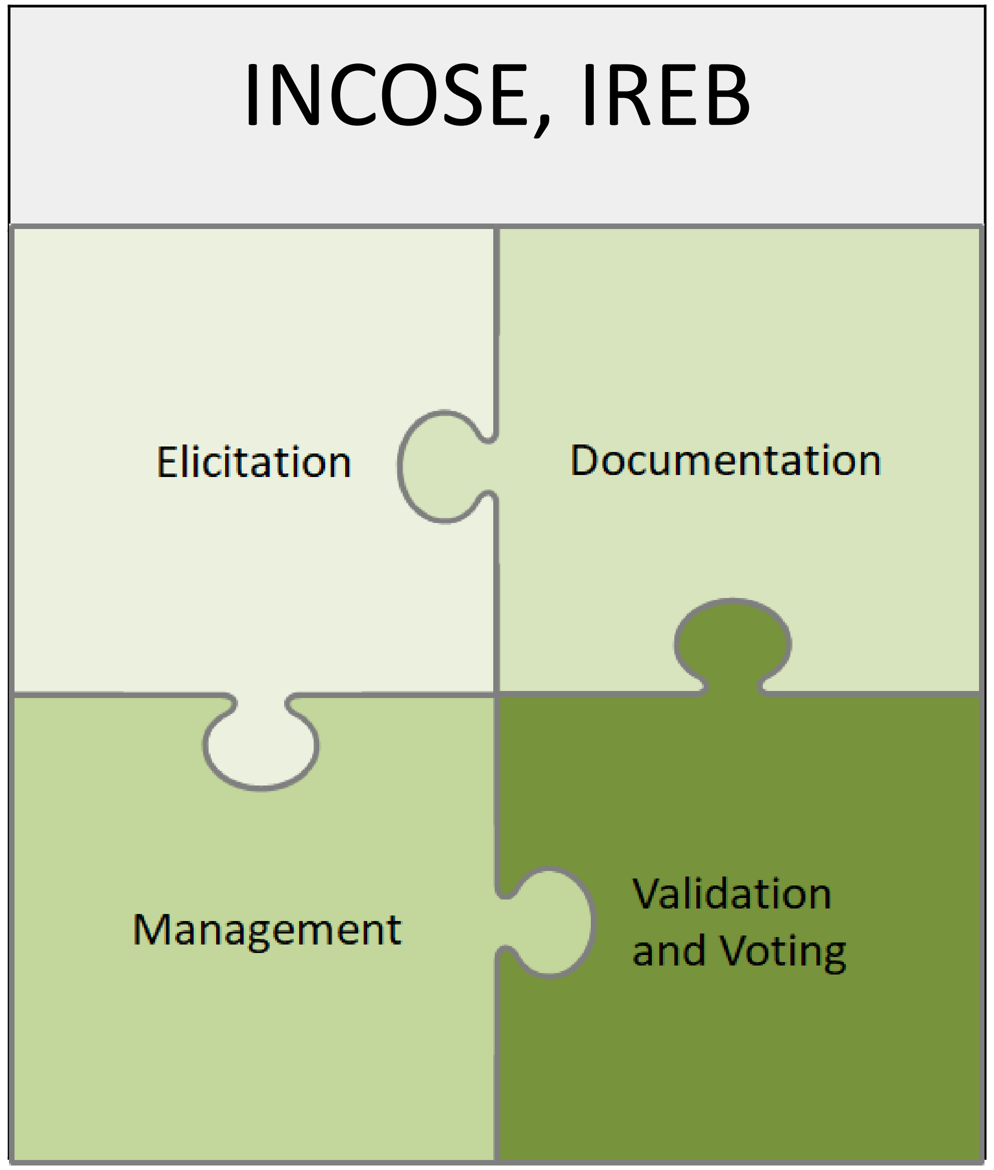
Well-formed requirement

According to ISO/IEC/IEEE 29148

- “A **well-formed specified requirement** contains one or more of the following:
  - it shall be met or possessed by a system to solve a problem, achieve an objective or address a stakeholder concern; (cf. requirements definition – IEEE 610.12)
  - it is **qualified by measurable conditions**; (testability)
  - it is **bounded by constraints**; (regulation, business rules, feasibility, testability)
  - it defines the **performance of the system…**; (non functional requirements) and
  - it **can be verified** (testability)

- It is important **to agree in advance** on the **specific keywords and terms…** A common approach is to stipulate the following.
  - Requirements are mandatory binding provisions and **use 'shall'**.
  - Non-requirements, use verbs such as ‘are’, ‘is’, and ‘was’. It is best to avoid using the term ‘must.
  - **Preferences or goals** are desired, non-mandatory, non-binding provisions and **use 'should'**. **They are not requirements**.
  - **Use positive statements** and **avoid negative requirements** such as ‘shall not’.
  - **Use active voice**: avoid using passive voice, such as ’it is required that’.

Requirement pattern usage; When?



Elicitation	Elicit, particularize and refine requirements. Interview techniques; creativity techniques; document-centered techniques; observation techniques
Documentation and Analyze ✓	Describe requirements adequately, e.g. in prose or model based. 3 perspectives: structure, function, behaviour; natural language, conceptual models; standard structure; quality criteria; glossary
Validation and Voting/ Negotiation ✓	The quality of requirements is ensured. Quality aspects: content; documentation; consensus; statement; inspection; walkthrough; perspective oriented reading
Management	Complies with requirements management; e.g. versioning requirements. Setting attributes; role specific views; prioritization; traceability; versioning; configuration management; change management; tools

Usage of patterns

✓ To describe requirements; Linguistic pattern (Eye of the requirement engineer)
✓ To validate requirement; Attributes pattern (Eye of the requirement consumer)

Requirement pattern usage; Why?

- Writing requirements in Natural language is a common way as it is easy to understand for everybody. Moreover it is suitable for the documentation of any kind of requirements.
- Problems linked to requirements are linked to requirement quality attributes not respected (ISO 25010, IEEE830, INCOSE….+ internal rules of requirement definition).
- Quality attribute violation involves requirement defects: “Incompleteness”, “Non-Verifiable”, “Ambiguous”, “Infeasibility”; “Vagueness”; “Ambiguity”;” Un-clarity”; “Subjectivity”;” Implicity”; “Typo”; “Formatting error”; “Weakness”; “Multiplicity”….)
- ***Usage of Patterns reduce linguistic defects.***

Minimum patterns – User Perspective

- For **user** requirement
  - Condition: initial state / context
  - Actor (user): human / persona
  - Action (condition / capability): intent of the actor
  - Objective (problem / objective) : new state / result of the action

- "Condition", as an "actor", I shall "action" so that "objective"
- Example
  - When I'm logged in, as a speaker, I shall upload my presentation proposal so that it can be reviewed by the program committee

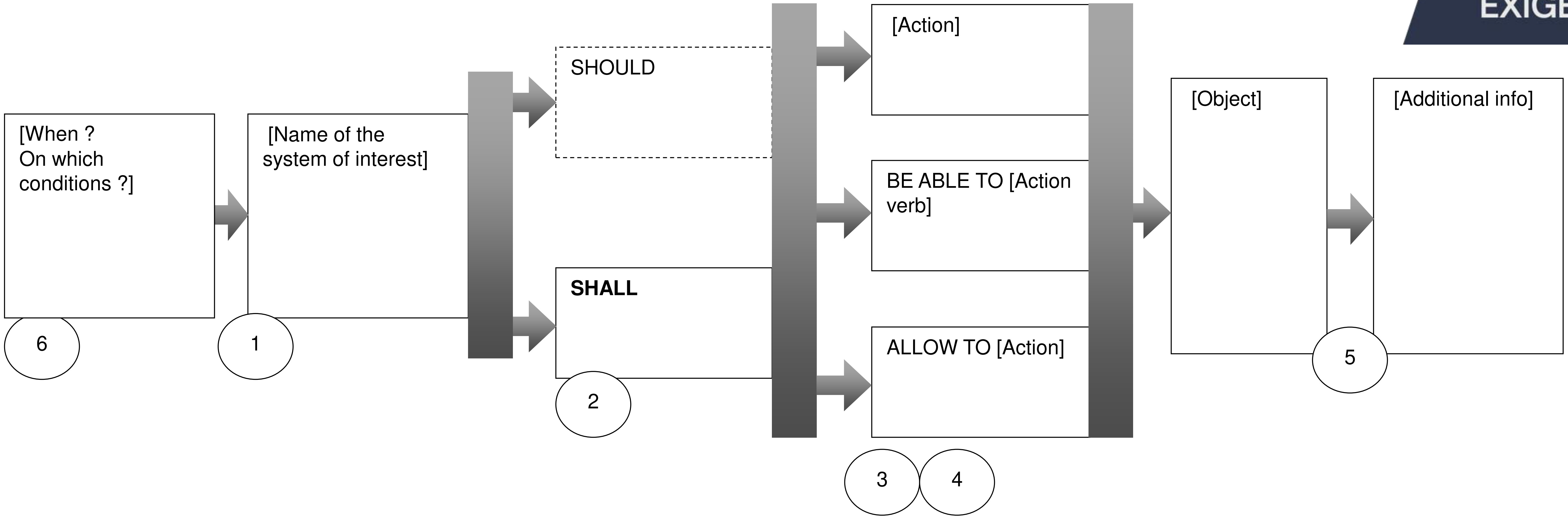
Minimum patterns – System / Component perspective

- For **functional** requirement
  - Condition: initial state / context
  - Actor: system / component
  - Action (condition / capability): intent of the actor
  - Objective (satisfy contract, standard, specification): new state / result of the action

- “Condition”, the “actor” shall “action” so that “objective”
- Examples
  - When a user is logged, the system shall grant this user to upload presentation proposals so that they can be stored in a shared space
  - For a user with a valid account, the “rights management” component shall verify that the user is logged so that it can grant to upload presentation proposals

Usage of pattern; Exemple

JOURNÉE FRANCAISE DE L'INGÉNIERIE DES EXIGENCES 2023



"shall" = requirement
"will" = facts or declaration of purpose
"should" = goal, a preference at early stage
"shall not" = safety / security properties

"Shall": Functional Requirement
"Shall be able to ": Interface requirement (Constraint)
"Shall allow to": User interaction requirements (stakeholders).

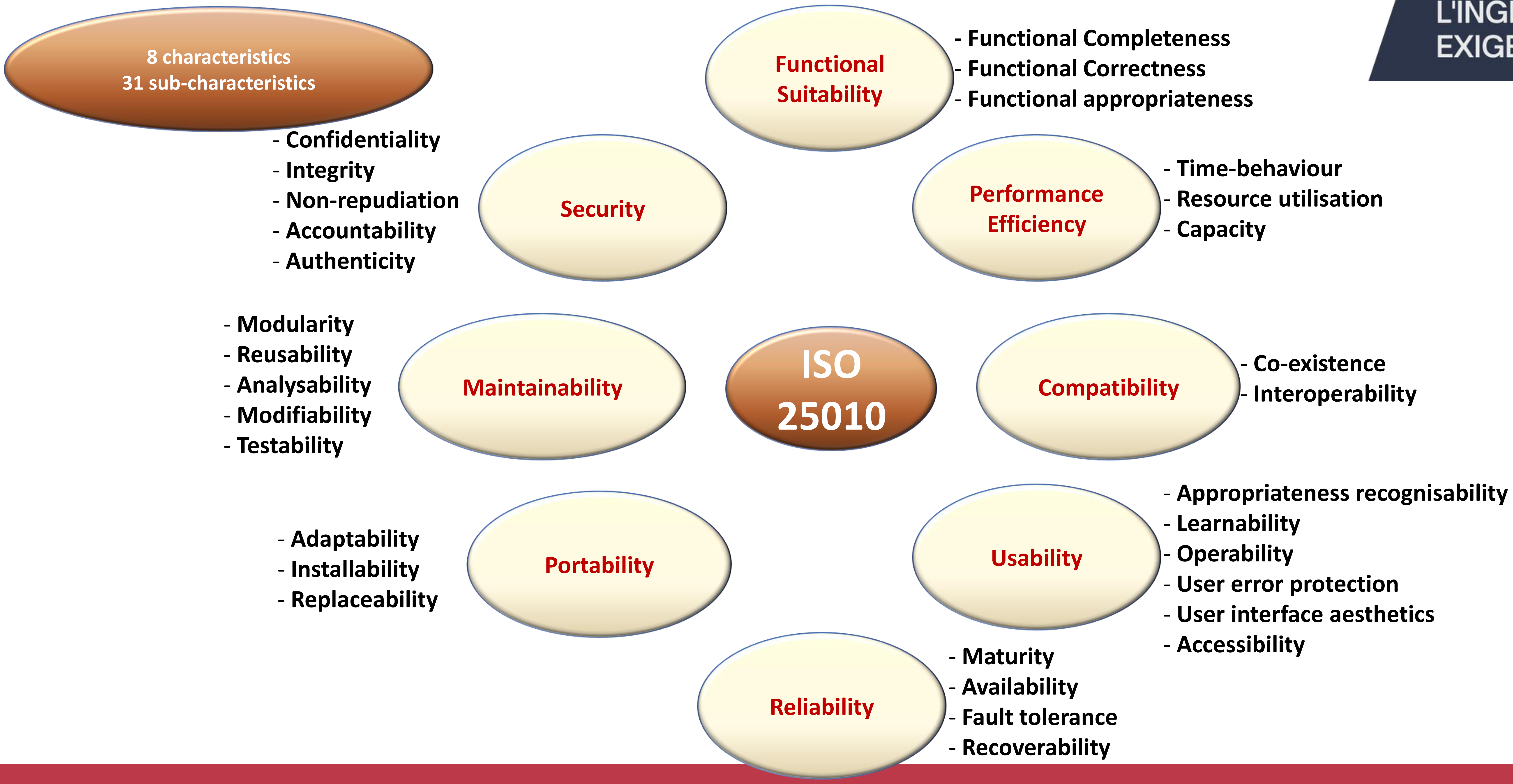
Example: If the Driver switches button DoorLock the system shall send Signal sDoorLock = true on LIN.

Minimum patterns

- For **non functional** requirement
  - Condition: initial state / context
  - Actor: system / system component
  - Action: intent of the actor
  - How: how system / system component shall behave
  - Objective: new state / result of the action

- "Condition", the "actor" shall "action" "how" so that "objective"
- Example:
  - *In operation mode, the system shall produce RPI4 electronic board with a minimal availability rate of 95% so that orders will be delivered in time (reliability)*

Non functional requirement (ISO 25010)



8 characteristics
31 sub-characteristics

ISO 25010

Functional Suitability

- Functional Completeness
- Functional Correctness
- Functional appropriateness

Performance Efficiency

- Time-behaviour
- Resource utilisation
- Capacity

Compatibility

- Co-existence
- Interoperability

Usability

- Appropriateness recognisability
- Learnability
- Operability
- User error protection
- User interface aesthetics
- Accessibility

Reliability

- Maturity
- Availability
- Fault tolerance
- Recoverability

Portability

- Adaptability
- Installability
- Replaceability

Maintainability

- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

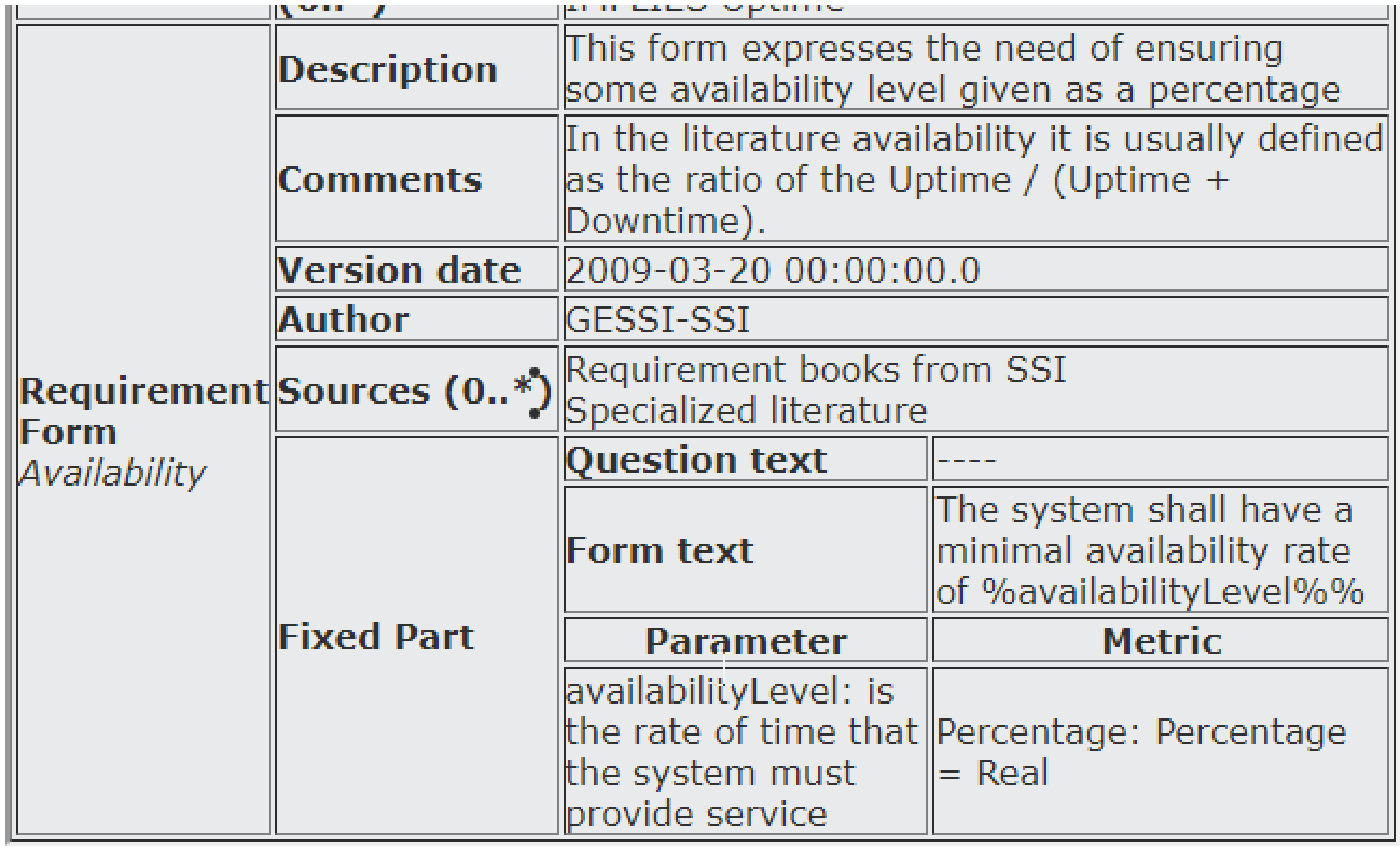
Security

- Confidentiality
- Integrity
- Non-repudiation
- Accountability
- Authenticity

Non functional requirement

Pros': good guidelines to specify non functional requirements
Cons': based on ISO 9126, not finalized

- PABRE (Non-Functional Requirements catalog…)
  - Framework
    - Requirements Pattern
      - Description
      - Comments
      - Pattern Goal
      - Keywords
    - Requirements Form
      - Description
      - Comments
      - « Fixed Part »
      - « Extended Part » (optional)

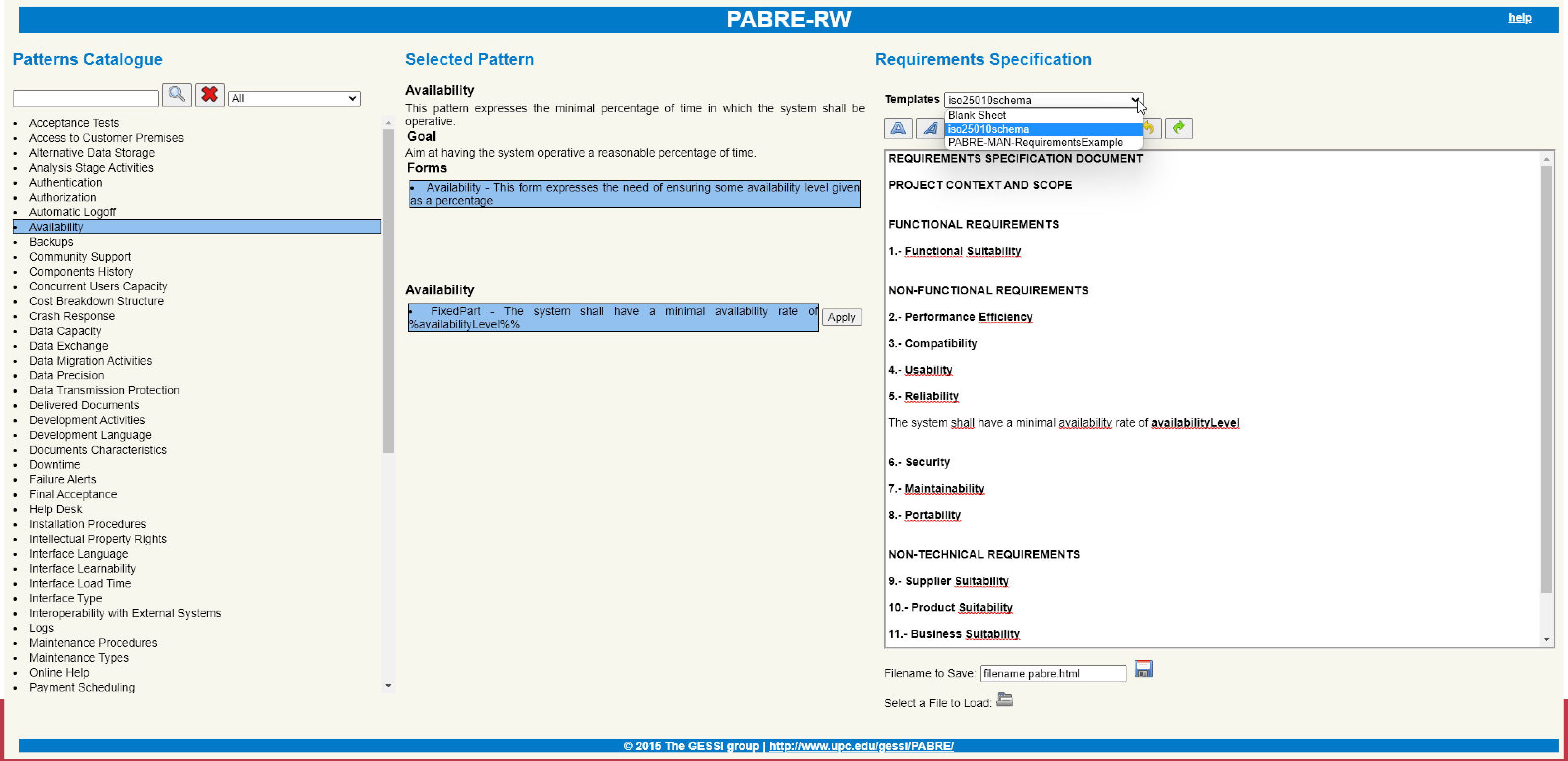


Requirement Form *Availability*	**Description**	This form expresses the need of ensuring some availability level given as a percentage	
	Comments	In the literature availability it is usually defined as the ratio of the Uptime / (Uptime + Downtime).	
	Version date	2009-03-20 00:00:00.0	
	Author	GESSI-SSI	
	Sources (0..*)	Requirement books from SSI Specialized literature	
	Fixed Part	**Question text**	----
		Form text	The system shall have a minimal availability rate of %availabilityLevel%%
		Parameter	**Metric**
		availabilityLevel: is the rate of time that the system must provide service	Percentage: Percentage = Real

Source: *https://www.upc.edu/gessi/tools/PABRE/PABRE-Patterns.html#*

Non functional requirement

- PABRE (…and tools) *Source: https://chrome.google.com/webstore/detail/pabre-requirements-specif/kafckkocfheeilpaokjjgiajcojapolg*



Patterns Catalogue

- Acceptance Tests
- Access to Customer Premises
- Alternative Data Storage
- Analysis Stage Activities
- Authentication
- Authorization
- Automatic Logoff
- Availability
- Backups
- Community Support
- Components History
- Concurrent Users Capacity
- Cost Breakdown Structure
- Crash Response
- Data Capacity
- Data Exchange
- Data Migration Activities
- Data Precision
- Data Transmission Protection
- Delivered Documents
- Development Activities
- Development Language
- Documents Characteristics
- Downtime
- Failure Alerts
- Final Acceptance
- Help Desk
- Installation Procedures
- Intellectual Property Rights
- Interface Language
- Interface Learnability
- Interface Load Time
- Interface Type
- Interoperability with External Systems
- Logs
- Maintenance Procedures
- Maintenance Types
- Online Help
- Payment Scheduling

Selected Pattern

Availability

This pattern expresses the minimal percentage of time in which the system shall be operative.

Goal

Aim at having the system operative a reasonable percentage of time.

Forms

- Availability - This form expresses the need of ensuring some availability level given as a percentage

Availability

- FixedPart - The system shall have a minimal availability rate of %availabilityLevel%%

Requirements Specification

Templates: iso25010schema

REQUIREMENTS SPECIFICATION DOCUMENT

PROJECT CONTEXT AND SCOPE

FUNCTIONAL REQUIREMENTS

1.- Functional Suitability

NON-FUNCTIONAL REQUIREMENTS

2.- Performance Efficiency

3.- Compatibility

4.- Usability

5.- Reliability

The system shall have a minimal availability rate of **availabilityLevel**

6.- Security

7.- Maintainability

8.- Portability

NON-TECHNICAL REQUIREMENTS

9.- Supplier Suitability

10.- Product Suitability

11.- Business Suitability

Filename to Save: filename.pabre.html

Select a File to Load:

© 2015 The GESSI group | http://www.upc.edu/gessi/PABRE/

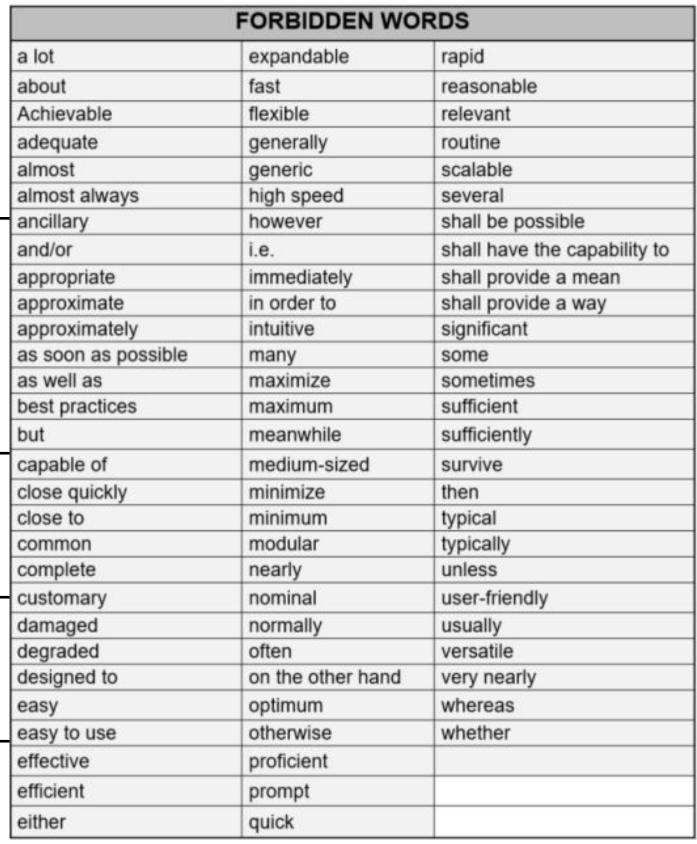
Requirement validation pattern

Following 6 principles improve the quality of the validation process [IREB]:

- Participation of right contributors – Use quality critera
- Separation of review and error
- Review from different views – disciplines, consumers, testers
- Change of documentation type - Requirements based on model or natural language have their individual strengths and weaknesses.
- Construction of development artefacts - E.g. trying to define test criteria
- Repetitive reviews – Especially on iterative, agile developments

Usage of verifiable writing rules

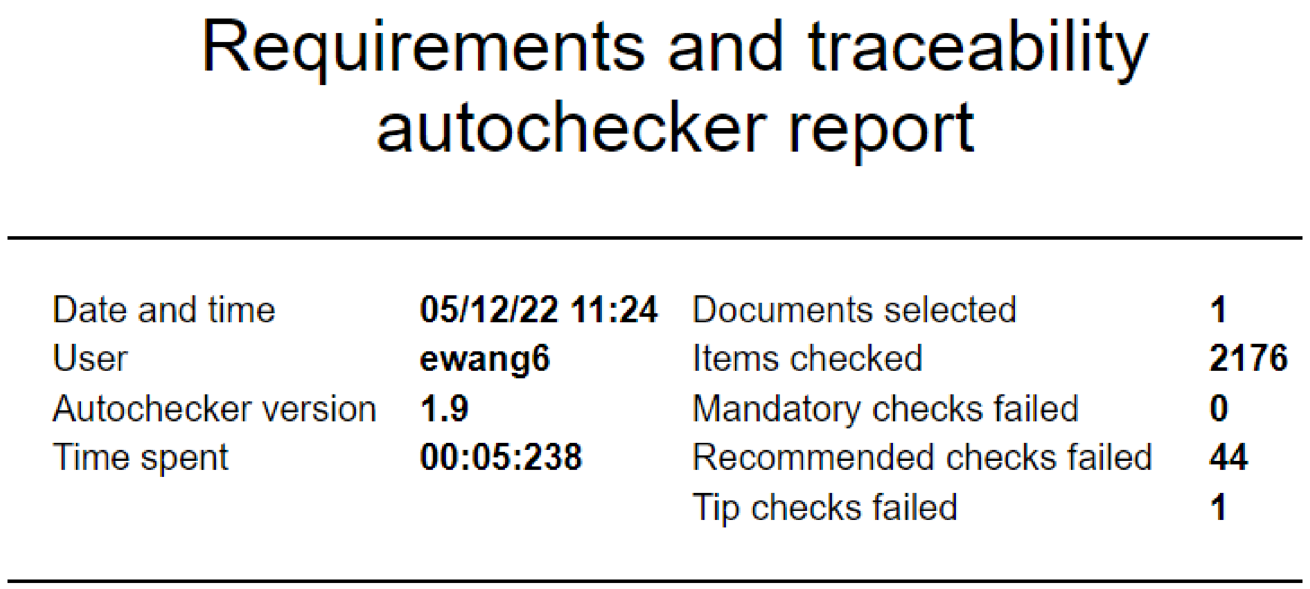
Rule	**Rule description**
Interface	Requirements description shall use identified interface of the item under construction
Tolerance	Requirements shall define the range of acceptable values associated with quantities.
Forbidden Words	Adverbs, Adjectives, Imprecise quantifiers, Combinators, pronouns Define glossary of forbidden words =>
Shall	Terms like « has to », « can », « may », « could » are forbidden in a requirement description.
Active Actor	Requirements shall use the active voice with the actor clearly identified.
Unit	When stating quantities, requirements shall use physical units.
Correct	Requirements shall use correct grammar, spelling and punctuation.
Oblique	Requirements shall avoid the use of the oblique "/" symbol except for the standard symbol “+/-“.



FORBIDDEN WORDS		
a lot	expandable	rapid
about	fast	reasonable
Achievable	flexible	relevant
adequate	generally	routine
almost	generic	scalable
almost always	high speed	several
ancillary	however	shall be possible
and/or	i.e.	shall have the capability to
appropriate	immediately	shall provide a mean
approximate	in order to	shall provide a way
approximately	intuitive	significant
as soon as possible	many	some
as well as	maximize	sometimes
best practices	maximum	sufficient
but	meanwhile	sufficiently
capable of	medium-sized	survive
close quickly	minimize	then
close to	minimum	typical
common	modular	typically
complete	nearly	unless
customary	nominal	user-friendly
damaged	normally	usually
degraded	often	versatile
designed to	on the other hand	very nearly
easy	optimum	whereas
easy to use	otherwise	whether
effective	proficient	
efficient	prompt	
either	quick	

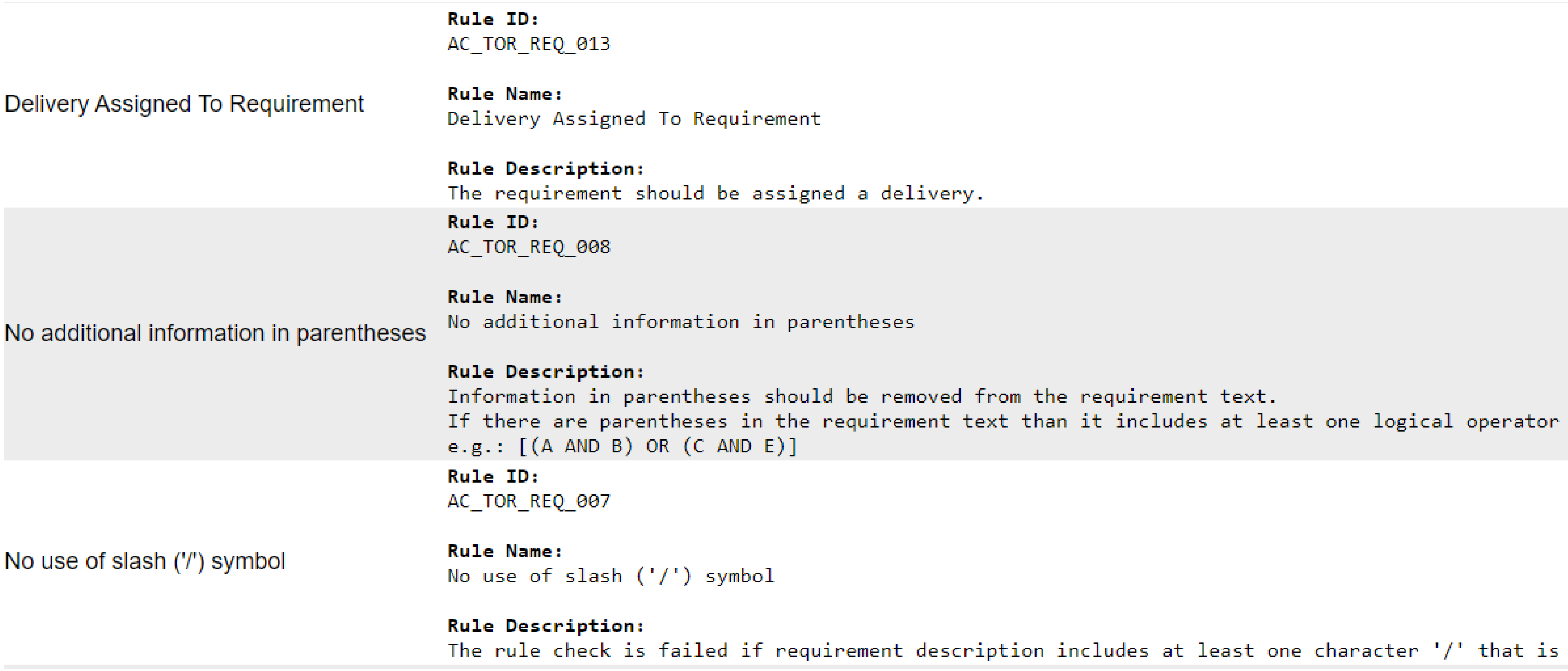
Example: Req pattern verification tool

- Tool implemented in DOORS for 50 defined rules
  Both of requirement writing pattern and attribute pattern
- Coverage: 19/50 fully, 17/50 partially; 14/50 not covered
- Example



Requirements and traceability autochecker report

Date and time	**05/12/22 11:24**	Documents selected	**1**
User	**ewang6**	Items checked	**2176**
Autochecker version	**1.9**	Mandatory checks failed	**0**
Time spent	**00:05:238**	Recommended checks failed	**44**
		Tip checks failed	**1**



Delivery Assigned To Requirement	**Rule ID:** AC_TOR_REQ_013 **Rule Name:** Delivery Assigned To Requirement **Rule Description:** The requirement should be assigned a delivery.
No additional information in parentheses	**Rule ID:** AC_TOR_REQ_008 **Rule Name:** No additional information in parentheses **Rule Description:** Information in parentheses should be removed from the requirement text. If there are parentheses in the requirement text than it includes at least one logical operator e.g.: [(A AND B) OR (C AND E)]
No use of slash ('/') symbol	**Rule ID:** AC_TOR_REQ_007 **Rule Name:** No use of slash ('/') symbol **Rule Description:** The rule check is failed if requirement description includes at least one character '/' that is

Example: Req pattern verification tool

- OpenReq-CT (Conformance to Template)
  - Check the compliance to requirement templates
  - Available set of templates
  - Possibility to add templates
  - An example of a correct ABNF grammar for Rupp's template (summarised in Figure 1) would be:
    - <main> ::= <conditions> <actor> <modal> <last_part> <np>
    - <conditions> ::= %IF | %AFTER | %AS %SOON %AS | %AS %LONG %AS
    - <modal> ::= %SHALL | %SHOULD | %WILL
    - <actor> ::= <np>
    - <infinitive-vp> ::= %TO <vp>
    - <last_part> ::= <vp> | %PROVIDE <np> %THE %ABILITY %TO <infinitive-vp> | %BE %ABLE <infinitive-vp>

OpenReq

Source: https://gessi.upc.edu/en/tools/openreq-upc/conformance-to-templates
Github: https://github.com/OpenReqEU/conformance-to-templates

Pros': open source
Cons': not maintained

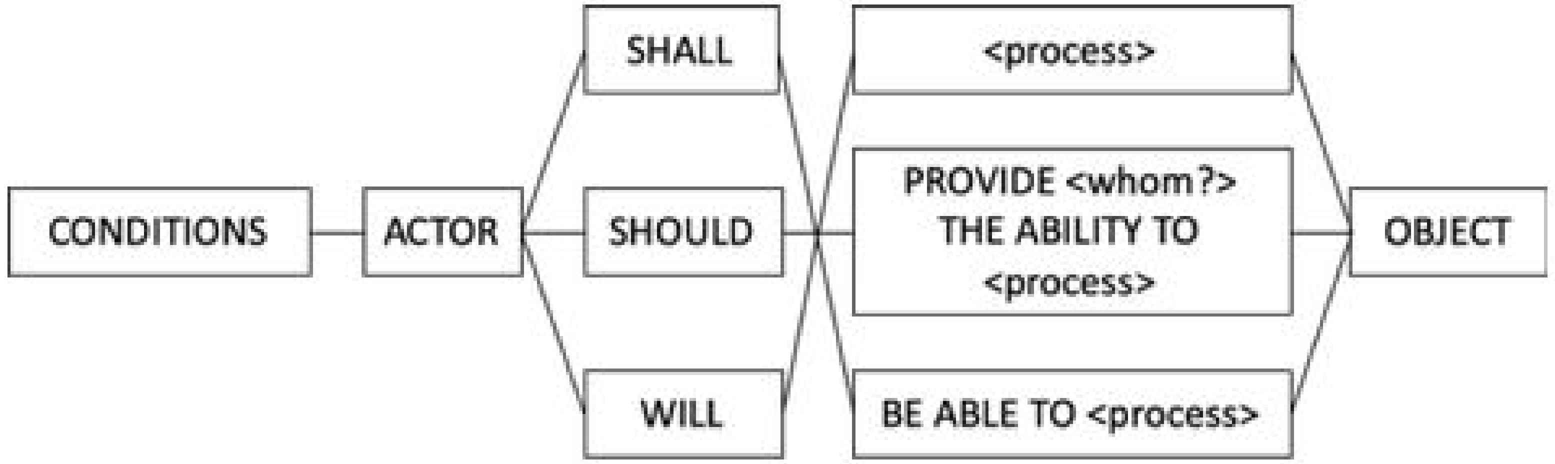


Figure 1: Rupp's templates excerpt

What are the benefits?

- Show how patterns answer/contribute to quality criteria

INCOSE	IREB	Agility
VERIFIABLE: Partially	ADEQUATE: Partially	I-Indépendance: Yes
NECESSARY: No	NECESSARY: No	N-Negociable: No
IMPLEMENTATION INDEPENDANT: Yes	UNAMBIGUOUS: Yes	V-Valuable: Partially
UNAMBIGUOUS: Yes	COMPLETE (SELF-CONTAINED): Partially	E-Estimable: Partially
SINGULAR: Yes	UNDERSTANDABLE: Yes	S-Small: Partially
FEASIBLE: Partially	VERIFIABLE: Partially	T-Testable: Partially
CORRECT: Partially		
TRACEABLE: No		
SELF SUPPORTING: Partially		

What are the benefits?

- Benefits for the development
  - Less implementation questions
- Benefits for the test
  - Acceptance / Test criteria
  - Patterns AAA/GWT: Arrange-Act-Assess / Given When Then
    - Arrange / Given (Test pre-condition) ➔ Requirement condition
    - Act / When (Test action) ➔ actor + action
    - Assess / Then (Expected test result) ➔ result of the actor + action

10ème édition
de la

JOURNÉE FRANCAISE DE L'INGÉNIERIE DES EXIGENCES

Du 14 au 16 Novembre 2023

A partir de 11h30

En ligne

100% gratuit

Le rendez-vous
incontournable
des experts du domaine

International®
Requirements
Engineering
Board

CFTL
Comité Français
des Tests Logiciels

Merci de votre écoute !