Electrical Engineering and Computer Science Dept
College of Engineering

REQUIREMENT ATTRIBUTES AND BOILERPLATES

Prepared by Prof. Dr. Christopher Shneider Cerqueira

SYS530 - 2025


YOUR USER REQUIRE-
MENTS INCLUDE FOUR
HUNDRED FEATURES.
www.dilbert.com scottadams@aol.com
DO YOU REALIZE THAT
NO HUMAN WOULD BE
ABLE TO USE A PRODUCT
WITH THAT LEVEL OF
COMPLEXITY?
4/4/01 © 2001 United Feature Syndicate, Inc.
GOOD POINT.
I'D BETTER ADD
"EASY TO USE"
TO THE LIST.

W	Dates	Topic	Ref	Activity
35	26-Aug	Course Introduction Systems Engineering Review Requirement Engineering (Generic Process)	[1][6] [7][9]	Groups & Names
36	2-Sep	Lego exercise 01 - First assessment	[2] [10]	Exercise Report
37	9-Sep	Definition and Structure of Requirements Functional and non-functional Requirements	[8][6] [10]	Exercise List
38	16-Sep	Elicitation Techniques (Interview / Rich Picture / use Cases)		Group Project Delivery
39	23-Sep	Rules for need and requirement statements	[2][3]	Exercise List
40	30-Sep	Attributes of need and requirement statements Boilerplates/patterns	[2]	Exercise List
41	7-Oct	[Carrer Explo]		* Reading activity
42 *	14-Oct	Midterm Session		Self-paced concept questions
		[Fall break]		

W	Dates	Topic	Ref	Activity
43	21-Oct	MBSE Trends	**	Group Project Delivery
44	28-Oct	Traceability - Organization	[5][	Group Project Delivery
45	4-Nov	Metadata: Rationale + Verification and Validation (Success Criteria)	[2][4]	Group Project Delivery
46	11-Nov	[Veterans Day]		* reading activity
47	18-Nov	Lego Exercise 02	[2] [10]	Exercise Report
48	25-Nov	System Requirement Review Presentation (G1-G4)		
		[Thanksgiving]		
48	2-Dec	System Requirement Review Presentation (G5-G8)		
49	9-Dec	Finals		Self-paced concept questions and a small case

Attributes

The reason for using attributes is to better manage your project

- Given that needs and requirements are the common threads that tie each of the systems engineering product development life-cycle process activities together, having insight into these activities is necessary to manage the project effectively.
- Hull, Jackson and Dick (2011) describe attributes in terms of how they support the engineering process, stating: *"Attributes allow the information associated with a single requirement to be structured for ease of processing, filtering, sorting, etc. Attributes can be used to support many of the abilities [associated with requirements engineering], enabling the requirements to be sorted or selected for further action, and enabling the requirements development process itself to be controlled."*

- **Attributes are used to manage projects more effectively**. To use attributes to do so, they need to be accurate and current. As the old saying goes, **"garbage in; garbage out."** Another applicable saying is **"A time saving tool should not take more time to maintain than the time it saves."**
- (Tailorable) list of attributes that can be associated with each need and requirement statement.
- The listed attributes (not exhaustive and not prescriptive) are organized within the following four broad categories: (45)
  - *Attributes to help define the need or requirement and its intent. (8)*
  - *Attributes associated with the SOI (System of Interest) and its verification or validation. (4)*
  - *Attributes to help manage the needs or requirements. (25)*
  - *Attributes to show applicability and enable reuse of the needs and requirements. (8)*

Attributes to help define the need or requirement and its intent

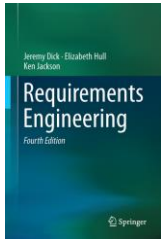
- **A1 – RATIONALE**
- **A2 – SOI PRIMARY VERIFICATION OR VALIDATION METHOD**
- **A3 – SOI VERIFICATION OR VALIDATION APPROACH**
- **A4 – TRACE TO PARENT**
- A5 – TRACE TO SOURCE
- A6 – CONDITION OF USE
- A7 – STATES AND MODES
- A8 – ALLOCATION

A1 – RATIONALE

- Rationale states the reason for the need or requirement statement's existence.
- Rationale defines why the need or requirement is needed and other information relevant to better understand the reason for and intent of the need or requirement.
- Rationale can also record assumptions that were made when writing the need or requirement, the source of numbers, and what design effort drove the requirement.

Rationale

- So often, the **real rationale for a particular design**, and the deeper understanding of how the components of a system work together to achieve an end result, remain in the **minds of the engineers**.
- Months or years later, when the original designers have long since moved on, or their memory has dimmed, the loss of that understanding may seriously impede the ability to evolve, maintain or reuse the system.



- In the absence of traceability to higher-level requirements (for example, when already in the top level of requirements), **many practitioners advocate that every requirement expression should contain an attribute called "Rationale",** which is used to explain the need for the requirement.
- For example, consider the requirement: ***"The boiler shall be able to provide a supply of water at temperatures between 90°C and 98°C at a rate of 20 litres per hour".***
- The question "Why is this requirement needed?" may lead us to capture the following rationale: **This requirement is necessary to allow the Baristas to make a sufficient quantity of hot drinks.**

Rationale

- Provides key information on how/why/when a requirement was developed or derived.
  - Can explain **why a requirement is needed** (including linking back to a User Need statement).
  - **What assumptions** the requirement may be based on.
  - **What trade study or conceptual design or architecture drove** the requirement.
  - Documenting this information in a rationale **statement helps maintain 'corporate knowledge' over time and personnel changes**.
  - Can be used to make the requirement statement **easier to understand**… **but don't use this as an excuse to write a poor Shall statement!**

Rationale information

- **Reason for the Requirement**: **Often the reason for the requirement is not obvious**, and it **may be lost** if not recorded as the requirement is being documented. The reason may point to a constraint or concept of operations. If there is a clear parent requirement or trade study that explains the reason, then it should be referenced.
- **Document Assumptions**: If a requirement was written assuming the **completion of a technology development program or a successful technology mission**, the assumption should be documented.
- **Document Relationships**: The relationships with the **product's expected operations** (e.g., expectations about how stakeholders will use a product) should be documented. This may be done with a link to the ConOps.
- **Document Design Constraints**: Constraints imposed by the results from **decisions made as the design evolves** should be documented. If the requirement states a method of implementation, the rationale should state why the decision was made to limit the solution to this one method of implementation.

Satisfaction rationale

- While the rationale attribute addresses necessity, there is the symmetrical concept of **sufficiency**.
- **The intention that the three system requirements are somehow sufficient to satisfy the user requirement**. It is difficult, however, for a non-expert to assess the validity of this assertion, because the reasoning has not been presented.
- What is better is to provide a “satisfaction statement”—sometimes called a“satisfaction argument” or “satisfaction rationale”—for each user requirement.
- The statement should **address the sufficiency of the set of linked requirements** in terms of satisfying the high-level requirement.

A2/A3 - A2 – SOI PRIMARY VERIFICATION OR VALIDATION METHOD // APPROACH

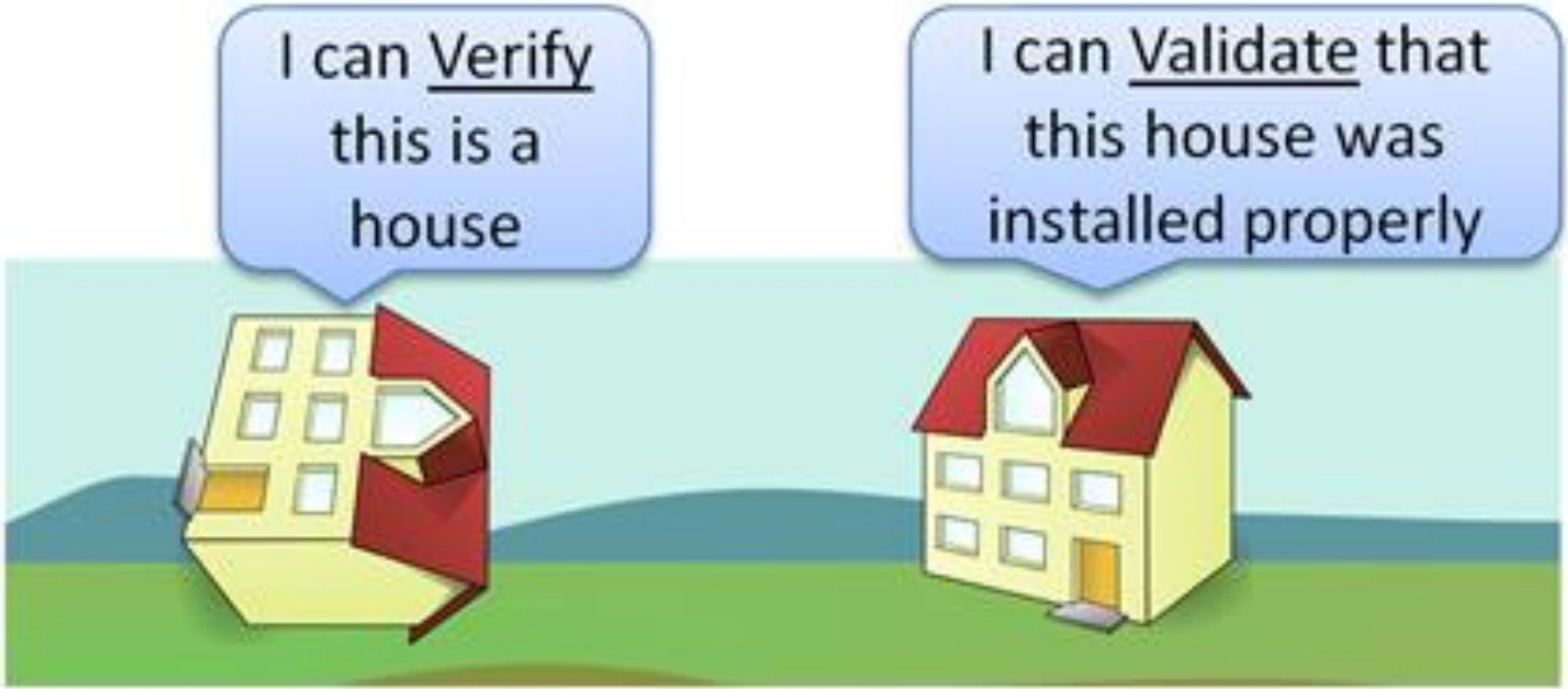
- The primary verification or validation method for each need or requirement states the planned primary method of verification or validation (test, demonstration, inspection, analysis) that is being proposed to provide proof the designed and built system meets the requirement (verification) or stakeholder need (validation).
- Approach or strategy suggested to verify the SOI meets a requirement or validate the SOI meets a need statement per the primary verification or validation method.
  - For example, for a requirement “The system shall do X.”, the primary method may be Test.
  - Then the verification approach or activity may be “The Test will consist of ….” with a definition of what the test comprises.
  - Other information may also be included: “The ability of the system to do X will be proven when the Test shows …” or “Verification will be considered successful when the Test shows …” This is a definition of the success criteria for verifying the system does meet the system requirement or validation that the system meets the need.

Verification

- The verification of a product shows **proof of compliance with requirements** – the product can meet each "must" claim as proven through the performance of a test, analysis, inspection or demonstration (or combination of these).
- **Verification is a process of confirming that a requirement or system is compliant**.
  - In other words, the system checker asks the question: **Does the system meet your requirements?**
  - The requirements checker asks the question: **Does the system really meet this specific requirement?**

Validation

- VALIDATION refers to the **ConOps document**.
- VALIDATION is conducted **under realistic conditions** (or simulated conditions) on end products for the purpose of determining the effectiveness and suitability of the product for use in mission operations by typical users.
- VALIDATION can be performed at each stage of development using phase products (e.g. models) and not just at delivery using end products.


I can Verify this is a house
I can Validate that this house was installed properly

Methods of V&V

4 FUNDAMENTAL METHODS	DESCRIPTION
INSPECTION	➢ **Visual examination** of drawings or data ➢ Examining a direct physical attribute of the item itself • dimensions • weight • physical characteristics • color or markings
ANALYSIS	➢ **Evaluation of data** by generally accepted analytical techniques • systems engineering analysis • Statistics • qualitative analysis • analog modeling • Similarity • computer and hardware simulation ➢ Can also be evaluation of data derived from lower-level verifications.
DEMONSTRATION	➢ **Showing that the use or operation** achieves the results without detailed data gathering. Utilize physical models or mockups. ➢ Often associated with the "ilities" (accessibility, transportability, serviceability, etc.)
TEST	➢ **Operation** of equipment during controlled conditions or when subjected to specified operational environments to evaluate performance ➢ Test is the preferred method of V&V

Other Methods of V&V

ADDITONAL METHODS	DESCRIPTION
SIMILARITY	➢ Assessing by **review of prior acceptance** data that a system is similar or identical in design & manufacturing process to another system that has been previously qualified ➢ previous and current system predicted, or actual environments are similar ➢ Usually considered a subset of Analysis ➢ Is not used when either, or both, of the following conditions exist: • The similar item used in the assessment was itself verified using similarity as the method • Items of criticality 1 or 1R (i.e., loss of vehicle, life, or serious injury)
VALIDATION OF RECORDS	➢ The use of vendor-furnished manufacturing or **processing records** to ensure requirements have been met ➢ Often used for COTS products or products purchased to standards

- There are differing definitions within the community of practice.
- The key is for your Project to establish agreed-to definitions, then be consistent in using the terminology!

Tasks	Steps
Define verification/validation requirements.	a. Define the method by which the requirement is to be verified. (1) Test. (2) Analysis. (3) Inspection. (4) Demonstration. (5) Validation of Records. (6) Similarity. b. Define the level at which the verification/validation will occur. (1) System. (2) Subsystem. (3) Component. c. Define the phase or purpose of the verification/validation activity to be performed. (1) Development. (2) Qualification. (3) Acceptance. (4) Pre-launch. (5) Flight/Mission. (6) Post-flight. (7) Disposal.
Manage and maintain the verification/validation requirements.	a. Manage and maintain verification/validation requirements. b. Update as necessary in accordance with established data management requirement.

Reference: MSFC-HDKB-2221, Volume I and Volume II

MoCs
Means of Compliance

Type of Compliance	Means of Compliance	Associated Compliance Documents
Engineering evaluation	MC0: -Compliance statement -Reference to Type Design documents -Election of methods, factors... -Definition	-Type Design documents -Recorded statements
	MC1: Design review	-Description -Drawings
	MC2: Calculation/Analysis	-Substantiation reports
	MC3: Safety assessment	-Safety analysis
Tests	MC4: Laboratory tests	-Test programmes -Test reports -Test interpretations
	MC5: Ground tests on related product	
	MC6: Flight tests	
	MC8: Simulation	
Inspection	MC7: Design inspection/audit	-Inspection or audit reports
Equipment qualification	MC9: Equipment qualification	-Note: Equipment qualification is a process which may include all previous means of compliance

https://www.faa.gov/aircraft/gen_av/light_sport/StandardsChart

Success Criteria

- The Product Verification/Product Validation success criteria provide **the detailed and specific criteria that determine the successful completion of the verification/validation planning activities**.

- Product Verification/Product Validation **success criteria are submitted as part of the PDR data package (ideally) and established (prepared) at least 90 days prior to the start of the verification/validation activity** to provide sufficient time to develop and publish the procedures.

- **VR-5T: Prove that the product's communications system is capable of communicating with the ground command team by performing an integrated system test within an integrated hardware/software environment. Testing will show that the product can transmit and receive to standard ground recovery forces audio at frequencies represented by communications devices defined in (TBD).**

Verification Objective

Verification Method

Environment

Note – there are no Special Conditions

Success Criteria

A4 – TRACE TO PARENT

- System level requirements are a transformation from one or more needs, so at the system level all requirements must trace to their parent stakeholder need(s).
- Requirements at one level are implemented at the next level of the system architecture via allocation (see A8, Allocation).
  - A child requirement is one that has been derived or decomposed from the allocated parent.
  - The achievement of each of the children's requirements will lead to the achievement of the parent requirement.
  - Each of the children's requirements must be traced to its parent requirement (and then to any antecedent requirement, and ultimately to the system need/mission.
- When managing requirements in a requirement management tool (RMT) or other SE tool, when a trace is established from the child to the parent, a reverse trace is also established between the parent and that child requirement. This is commonly referred to as “bi-directional” traceability.

Requirement tree

- Understand the Requirements Structure of your Project
  - Identify Parent Requirement Documents and Requirements Sources
  - Identify Peer and Interface Requirements
  - Identify Children Requirement Documents

- Note that at the ‘Document Level’, the above will start to identify ‘Stakeholders’

Decomposition and derivation of requirements

- **As requirements are developed at each level of the requirements hierarchy**, you start with the requirements allocated to you (your Parent requirements) but that is not where you stop!
  - The same process, identification of Stakeholders, a ConOps, etc., **must be applied at every level** of the requirements hierarchy.
  - The **size & depth** of those activities may diminish, and some products may be combined or informal, but you must go through the same thought process to ensure **each level's requirement set is complete and correct**.

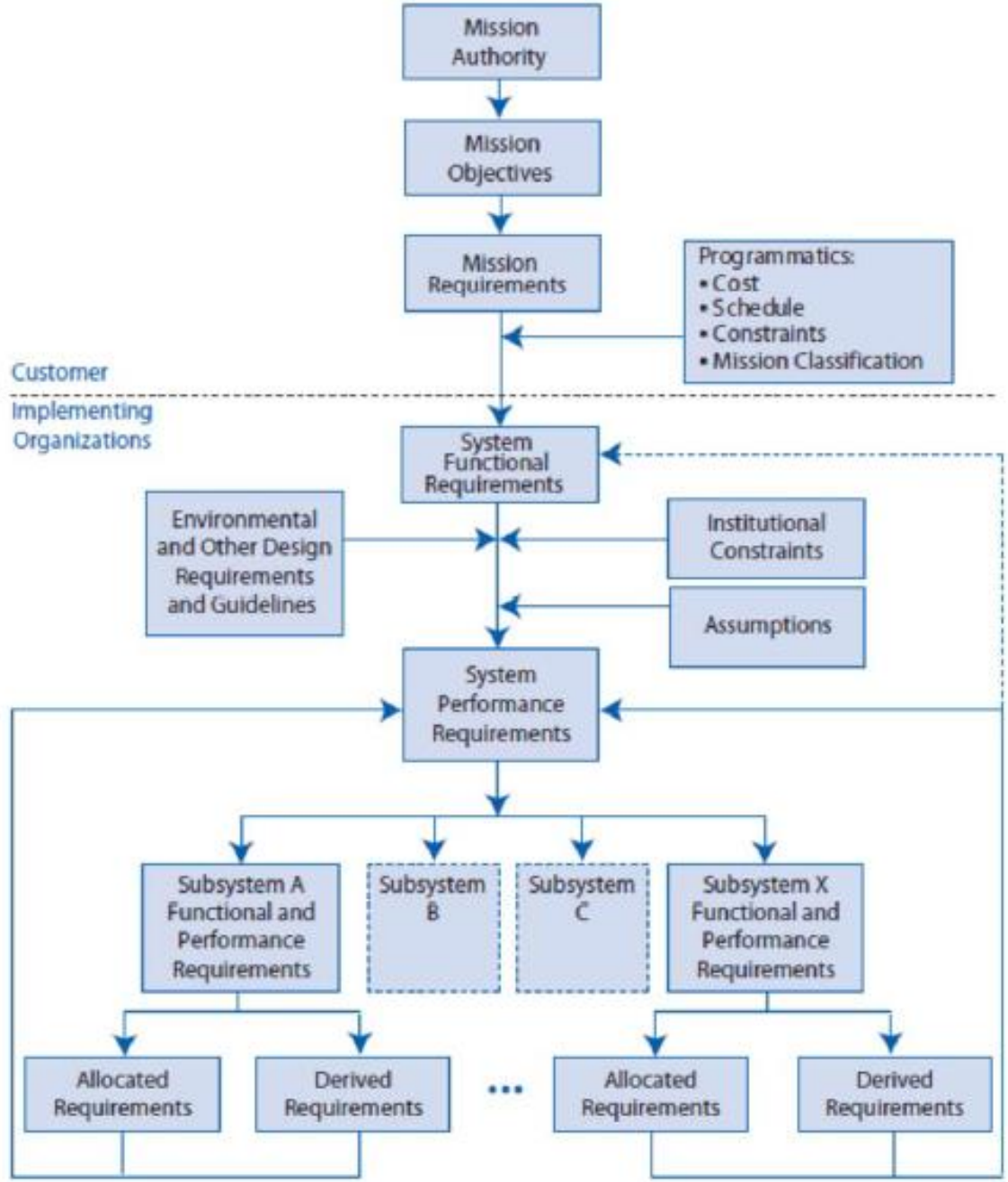
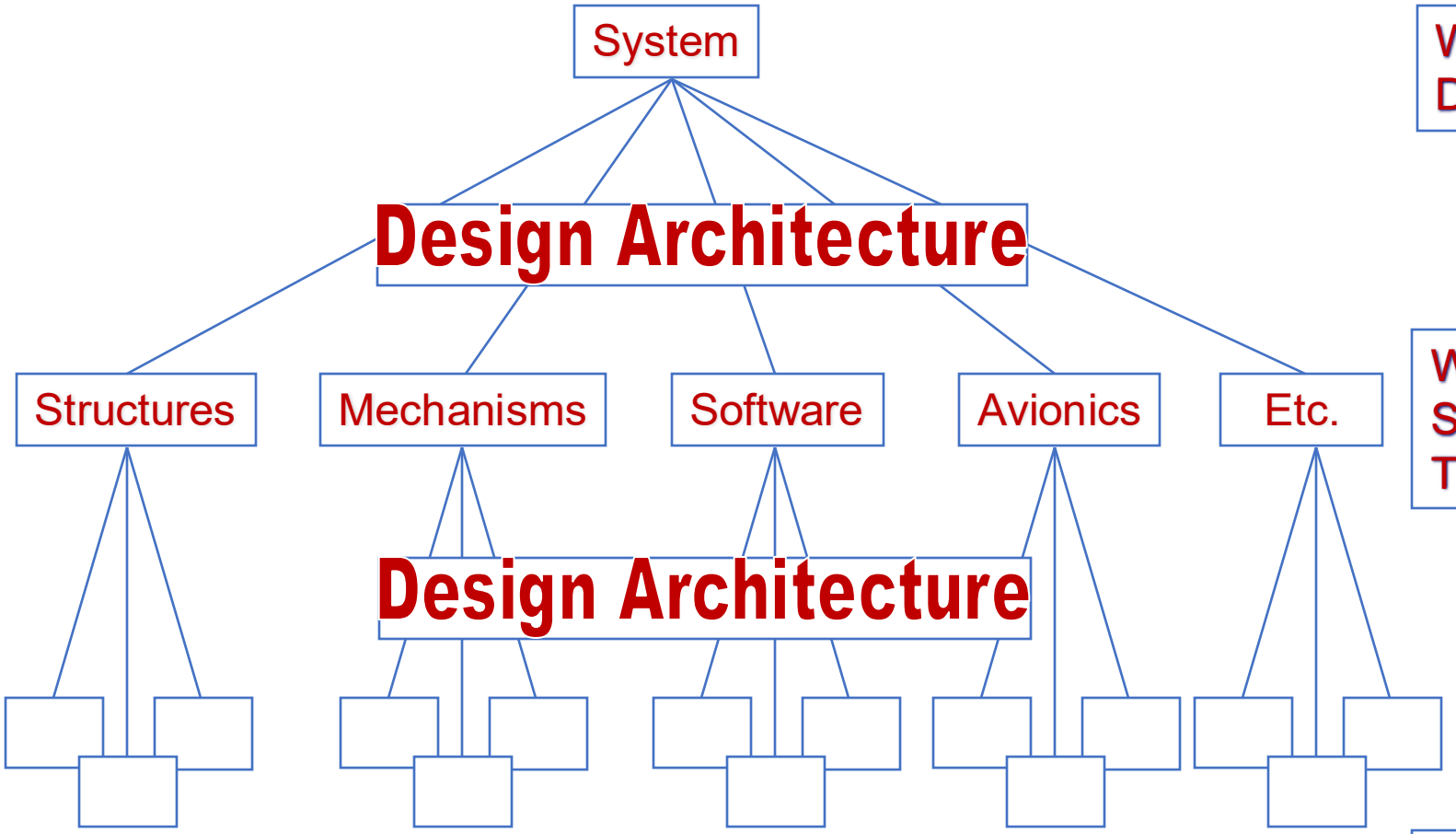


Figure 4.2-4 The Flowdown of Requirements

Requirements flowdown: one to many



What The System Shall Do

What The Subsystems Shall Do For The The System

What The Components Shall Do For The The Subsystem

Traceability

- Traceability is the relationship of requirements up & down the requirements hierarchy.
  - Typically this is Parent to Child (and the reverse) but can also be Peer to Peer.
  - Helps to confirm complete and accurate allocation of requirements
    - **Do all Parent Requirements have Children?**
    - **Do all Children Requirements have Parents (or are they Orphans)?**
  - Helps identify 'gold plating', the addition of un-needed requirements.
  - Vitale to **assessing impact of changes** in requirements.
  - *Obviously, a Requirements Management Tool (database) is very helpful.*
    - *But we've used Traceability Tables in Word documents as well.*

Attributes associated with the SOI and its verification or validation

- A9 – SOI VERIFICATION OR VALIDATION LEVEL
- A10 – SOI VERIFICATION OR VALIDATION PHASE
- A11 – SOI VERIFICATION OR VALIDATION RESULTS
- A12 – SOI VERIFICATION OR VALIDATION STATUS

Attributes to help maintain the requirements

- **A13 – UNIQUE IDENTIFIER**
- **A14 – UNIQUE NAME**
- **A15 – ORIGINATOR/AUTHOR**
- A16 – DATE REQUIREMENT ENTERED
- **A17 – OWNER**
- A18 – STAKEHOLDERS
- A19 – CHANGE BOARD
- A20 – CHANGE STATUS
- **A21 – VERSION NUMBER**
- **A22 – APPROVAL DATE**
- **A23 – DATE OF LAST CHANGE**
- A24 – STABILITY
- **A25 – RESPONSIBLE PERSON**
- A26 – NEED OR REQUIREMENT VERIFICATION STATUS
- A27 – NEED OR REQUIREMENT VALIDATION STATUS
- **A28 – STATUS (OF THE NEED OR REQUIREMENT)**
- **A29 – STATUS (OF IMPLEMENTATION)**

Attributes to help maintain the requirements

- A30 – TRACE TO INTERFACE DEFINITION
- A31 – TRACE TO PEER REQUIREMENTS
- A32 – PRIORITY
- A33 – CRITICALITY OR ESSENTIALITY
- A34 – RISK (OF IMPLEMENTATION)
- A35 – RISK (MITIGATION)
- A36 – KEY DRIVING NEED OR REQUIREMENT (KDN/KDR)
- A37 – ADDITIONAL COMMENTS
- A38 – TYPE/CATEGORY

Attributes to show applicability and enable reuse

- A39 – APPLICABILITY
- A40 – REGION
- A41 – COUNTRY
- A42 – STATE/PROVINCE
- A43 – APPLICATION
- A44 – MARKET SEGMENT
- A45 – BUSINESS UNIT
- A46 – BUSINESS (PRODUCT) LINE

Requirement Patterns

Introduction to Requirement Patterns

- The notion of requirement patterns (boilerplates or templates) was initially applied within the Future Surface Combatant (FSC) defense project in the United Kingdom in 1998 (Dick and Llorens, 2012) as an aid to **solve several difficulties when writing different types of textual requirements** (timeliness, etc.).
- Once requirement writers are shown a good example of how to form a requirement statement based on its type, the difficulty in writing a properly formed requirement was largely overcome.
- **The use of patterns also enables the development of natural language processing (NLP) tools or "digital assistants**". These tools, along with a project ontology, can aid writers when writing need and requirement statements and helping ensure the rule and characteristics defined within this Guide are met.

Benefits of using Patterns to form Requirement Statements

- Ensuring that requirements conform to agreed-to patterns contributes to the requirement verification activity and helps to ensure the requirements properly communicate needs.
- In addition, writing requirements following a set of agreed-upon patterns also makes it easier to:
  - *write concise, easy to read and atomic requirement statements;*
  - *find and classify requirements within a large document;*
  - *find missing requirements (completeness);*
  - *find inconsistences within a large set of requirements;*
  - *find duplicated requirements (and finding and reusing requirements in general); and*
  - *follow with other activities such as analysis and implementation.*

Choosing a Requirements Template

- A **Requirements Template / Boilerplate** is an outline or guideline for **how to organize** your requirements set.
  - Can be applied to a document or a database.
  - Can be tailored but best to be consistent in organization.
  - Helps to keep the focus on developing the requirements, not on re-inventing a template.
  - Makes information easy to find for all users and from Project to Project.
  - Can be used as a checklist to prevent any omissions.

Requirement text template

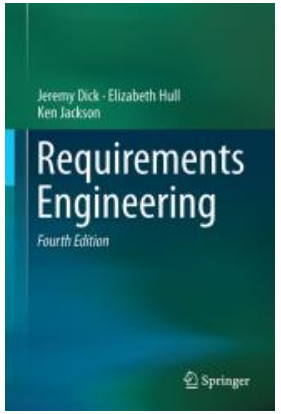


Robert Halligan

Element	Text
Actor:	The system,
Condition:	upon receipt of a message,
Action:	shall switch
Object of Action:	that message
Constraints of Action:	within 10 milliseconds of receipt
Refinement of Object:	for messages in ACP128 format having a valid routing indicator
Source of Object:	from the message input port,
Destination of Object:	to a message output port,
(Further) Refinement of Action:	corresponding to the routing indicator in the message.

Elements

- **Actor**. This is the subject of the sentence - the thing being specified. Examples (good and bad!) are: “the system”, “the interface”, “the function”, ...
- **Conditions of Action**. This defines the conditions during which the action is to take place, for example “in Replay Mode”, and/or the initiating conditions: “upon receipt of a message”, “power having been applied”, ...
- **Action**. This is a verb - the action to be taken by the actor (subject). Examples are “shall calculate”, “shall display”, “shall fly”, “shall be displayed”.....
- **Constraints of Action**. These qualify the action, for example “at a resolution of 400 x 1000 pixels”, “within limits imposed by vehicle speed”, ..., and include performance
- **Object of Action**. This is a noun, and is the thing acted upon in taking the action. Examples are: “the message”, “the input signal”, ...
- **Refinement/Source of Object**. These qualify the object, for example (refinement): “of flash priority”, for example (source): “from DISCON”.
- **Refinement/Destination of Action**. These further qualify the action and may be additional to Constraints of Action. Examples are “in accordance with IEEE 802.11g”, “to DISCON”.



Template 34

The <system> shall <function> <object> every <performance> <units>.

Requirement 347 = Template 34 +

<system> = *coffee machine*
<function> = *produce*
<object> = *a hot drink*
<performance> = *10*
<units> = *seconds*

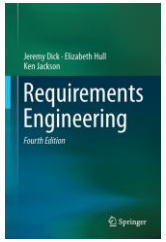
Requirement 348 = Template 34 +

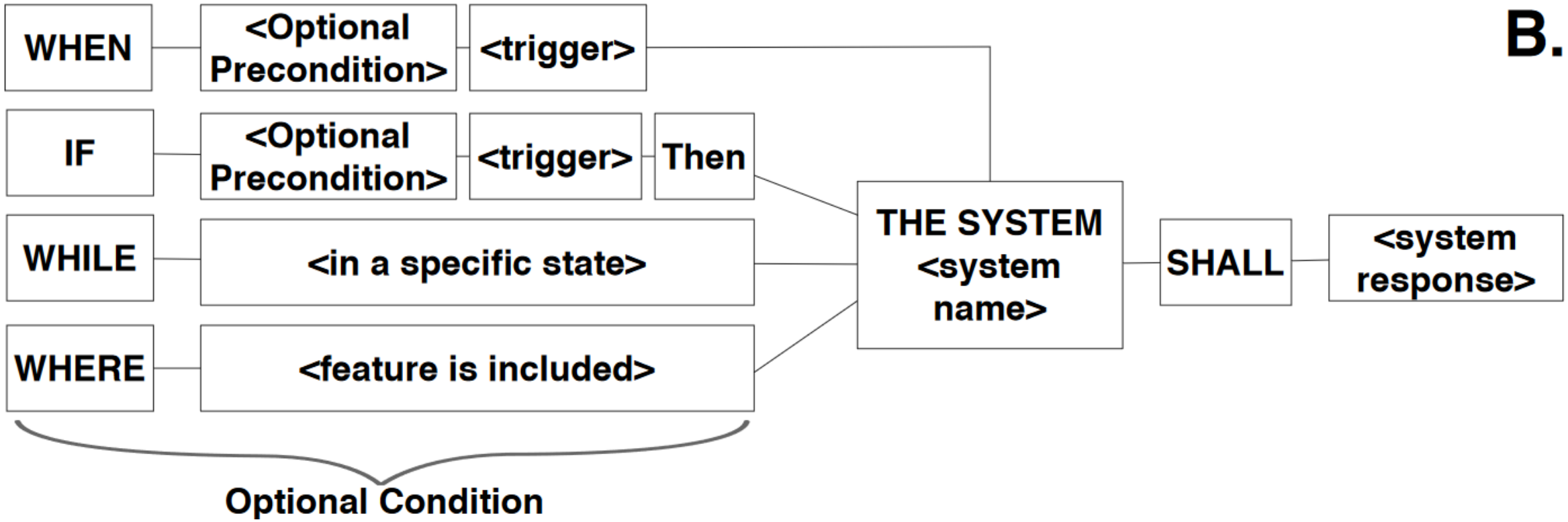
<system> = *coffee machine*
<function> = *produce*
<object> = *a cold drink*
<performance> = *5*
<units> = *seconds*

Fig. 4.3 Global templates

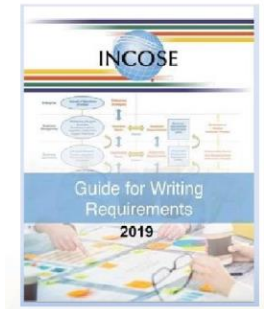
Table 4.4 Example boilerplates for constraint requirements

Type of constraint	Boiler-plate
Performance/ capability	The <system> shall be able to <function> <object> **not less than <performance> times per <units>**.
Performance/ capability	The <system> shall be able to <function> <object> **of type <qualification> within <performance> <units>**.
Performance/ capacity	The <system> shall be able to <function> **not less than <quantity>** <object>.
Performance/ timeliness	The <system> shall be able to <function> <object> **within <performance> <units> from <event>**.
Performance/ periodicity	The <system> shall be able to <function> **not less than <quantity>** <object> **within <performance> <units>**.
Interoperability/ capacity	The <system> shall be able to <function> <object> **composed of not less than <performance> <units> with <external entity>**.
Sustainability/ periodicity	The <system> shall be able to <function> <object> **for <performance> <units> every <performance> <units>**.
Environmental/ operability	The <system> shall be able to <function> <object> **while <operational condition>**.





Arora, C., Sabetzadeh, M., Briand, L.C., & Zimmer, F. (2014). Requirement boilerplates: Transition from manually-enforced to automatically-verifiable natural language patterns. 2014 IEEE 4th International Workshop on Requirements Patterns (RePa), 1-8.



TRC

INCOSE Guide for Writing Requirements: Real-Time Quality Assessment of the INCOSE Rules

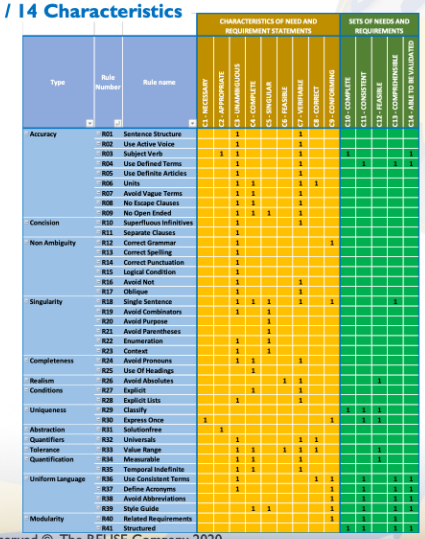
Tuesday, 24 March, 2020

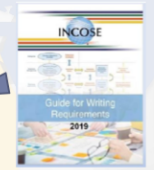
TRC

INCOSE GfWR

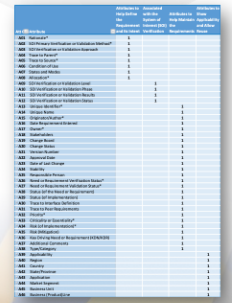
- **Characteristics** of individual and sets of needs and requirements, provides rationale and guidance for helping understand the characteristics.
- **Rules** for individual and sets of needs and requirements that help to formulate them. Included an explanation of the rule and examples of the application of the rule.
- **Attributes** that can be attached to a need or requirement statements to form need or requirement expressions. Also included is guidance on the use of attributes.

41 Rules / 14 Characteristics





46 Attributes



All rights reserved © The REUSE Company 2020

TRC

Patterns

When / After / If ... | [Condition] | <Subject> | Shall | <Action> | <Object> | [Constraint]

4.1.2 R2 - /ACCURACY/USEACTIVEVOICE

Use the active voice in the main sentence structure of the need or requirement statement with the responsible entity clearly identified as the subject of the sentence.

Elaboration:

The active voice requires that the entity performing the action is the subject of the sentence. This is important in writing needs and requirements since the onus for satisfying the requirement is on the subject, not the object of the statement. If the entity responsible for the action is not identified explicitly, it is unclear who or what should perform the action making verification of that requirement very difficult. Including the entity in the subject also helps ensure the requirement refers to the appropriate level consistent with the entity name (s

Often when the phrase "shall be" is used, the statement is in th



4.4.2 R19 - /SINGULARITY/AVOIDCOMBINATORS

Avoid combinators.

Elaboration:

Combinators are words that join clauses, such as "and", "or", "then", "unless", "but", "as well as", "but also", "however", "whether", "meanwhile", "whereas", "on the other hand", and "otherwise." Their presence in a requirement usually indicates that multiple requirements should be written.

Exception: AND, OR, NOT can be used in need and requirement statements as logical conditions and qualifiers as stated in R15.

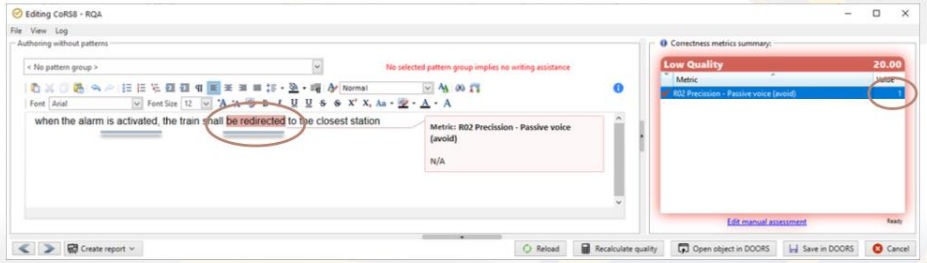
See also R16 and R17.

All rights r

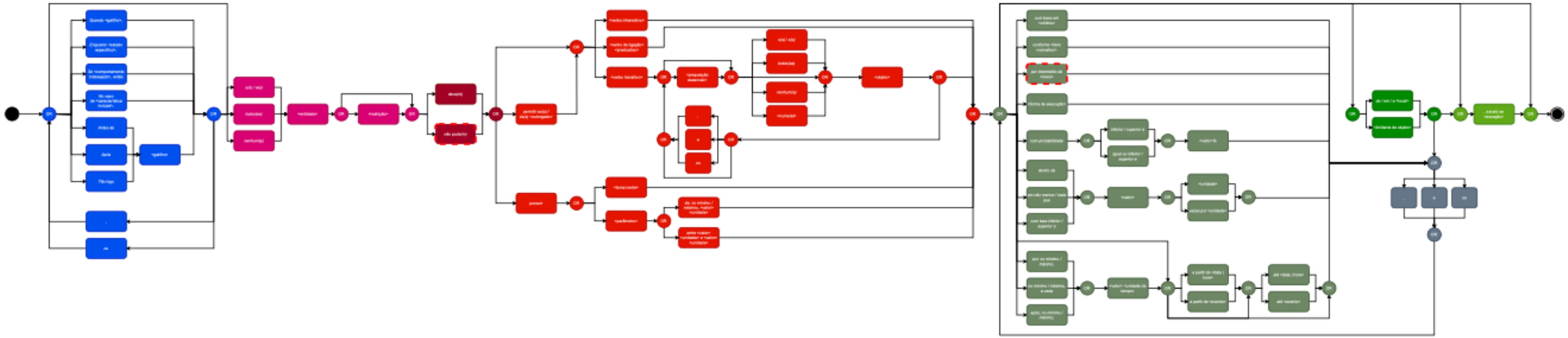
TRC

Patterns

When / After / If ... | [Condition] | <Subject> | Shall | <Action> | <Object> | [Constraint]



All rights reserved © The REUSE Company 2020



ABSTRACT

The thesis outlines a structured natural language for the documentation of requirements for military systems and employment materials, in the Portuguese language, for use in the Brazilian Army. Following the Design Science Research paradigm, the problem was precisely defined and justified in terms of its academic, industrial, and governmental relevance. Awareness of the problem arose through the exposition of the theoretical foundations of systems engineering and requirements engineering, and through the achievement of an integrative literature review. After establishing its objectives, the artifact was constructed using the design catalog of solutions method, resulting in a structure made up of 69 individual blocks. The artifact was artificially evaluated using a mathematical-linguistic model, based on dependency grammar. By processing 696 sentences, it was statistically proven that the use of the proposed structure is capable of reducing the textual complexity of requirements in three metrics relevant to the model. A naturalistic evaluation, conducted through the analysis of the perception of a group of typical users, demonstrated that it meets goals related to military use, sentence writing improvement, identification of structuring terms, and the Technology Acceptance Model. Evidence was found that, even though the ease of learning is uncertain, potential users are willing to employ the artifact in their professional daily routine.

Keywords: Requirements Engineering. Requirements documentation. Structured natural language.

NATURAL LANGUAGE STRUCTURING FOR DOCUMENTATION OF REQUIREMENTS FOR SYSTEMS AND MATERIALS FOR MILITARY USE IN THE PORTUGUESE LANGUAGE (2024)

Secretary of Defense // Brazilian Army
Department of Science and Technology
Ltn Cel Leonardo de Mello Barbosa (ASEP)

2009 17th IEEE International Requirements Engineering Conference

EARS (Easy Approach to Requirements Syntax)

Alistair Mavin Philip Wilkinson Adrian Harwood Mark Novak

Rolls-Royce PLC
PO Box 31
Derby DE24 8BJ, UK
alistair.mavin@rolls-royce.com

Generic requirements syntax

- The generic requirement syntax adopted during this study is as follows:

<optional preconditions> <optional trigger> the <system name> shall <system response>

- This simple structure forces the requirement author to separate the conditions in which the requirement can be invoked (preconditions), the event that initiates the requirement (trigger) and the necessary system behavior (system response).

Ubiquitous requirements

- A ubiquitous requirement has no preconditions or trigger. It is not invoked by an event detected at the system boundary or in response to a defined system state, but is always active. The general form of a ubiquitous requirement is:

The <system name> shall <system response>

- For example: "The control system shall prevent engine overspeed". This is a system behaviour that must be active at all times; hence this is a ubiquitous requirement.

Event-driven requirements

- An event-driven requirement is initiated only when triggering event is detected at the system boundary. The keyword **When** is used for event-driven requirements. The general form of an event-driven requirement is:

WHEN <optional preconditions> <trigger> the <system name> shall <system response>

- For example: "When continuous ignition is commanded by the aircraft, the control system shall switch on continuous ignition". This system response is required when and only when the stated event is detected at the boundary of the system.

Unwanted behaviours

- Requirements to handle unwanted behaviour are defined using a syntax derived from event-driven requirements.
- Requirements for unwanted behaviour are designated using the keywords **If** and **Then**. The general form of a requirement for unwanted behaviour is:

IF <optional preconditions> <trigger>, THEN the <system name> shall <system response>

- For example, "If the computed airspeed fault flag is set, then the control system shall use modelled airspeed". In this example, the unwanted event (computed air speed fault flag is set) triggers the system response, which allows continued safe operation of the system.

State-driven requirements

- A state-driven requirement is active while the system is in a defined state. The keyword **While** is used to denote state-driven requirements. The general form of a state-driven requirement is:

WHILE <in a specific state> the <system name> shall <system response>

- For example: "While the aircraft is in-flight, the control system shall maintain engine fuel flow above XXlbs/sec". The system response is required at all times whilst the system is in the defined state.
- To make requirements easier to read, the keyword During can be used instead of While for state-driven requirements.

Optional features

- An optional feature requirement is applicable only in systems that include a particular feature. An optional feature requirement is designated with the keyword **Where**. The general form of an optional feature requirement is:

  WHERE <feature is included> the <system name> shall <system response>

- For example, "Where the control system includes an overspeed protection function, the control system shall test the availability of the overspeed protection function prior to aircraft dispatch". This functionality only makes sense (and therefore is only required) for a system that includes the specified feature.

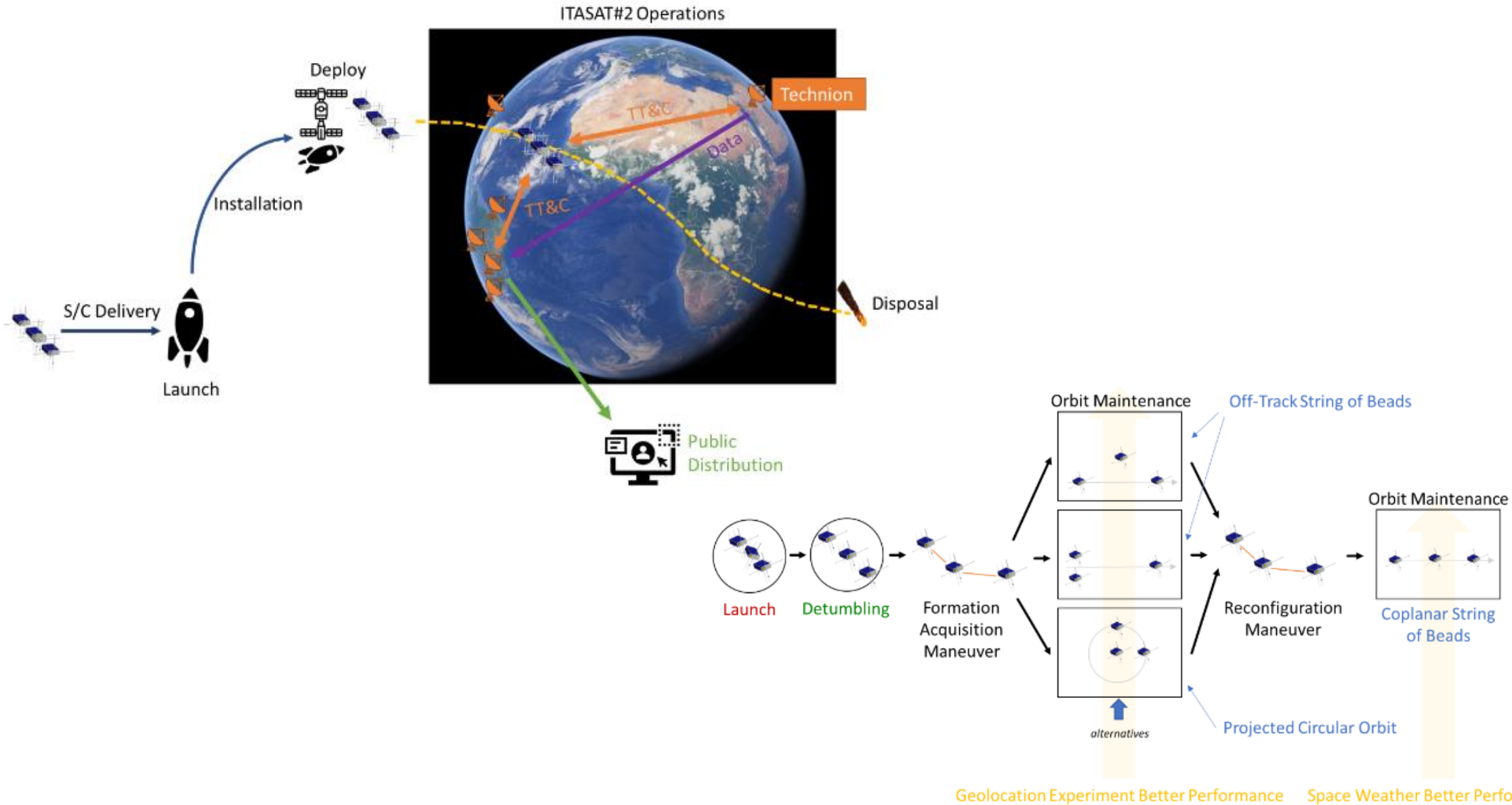
EARS (Easy Approach to Requirements Syntax) Summary

Generic requirements syntax	<optional preconditions> <optional trigger> the <system name> shall <system response>
Ubiquitous requirements	The <system name> shall <system response>
Event-driven requirements	When <optional preconditions> <trigger>, the <system name> shall <system response>
Unwanted behaviours	If <optional preconditions> <trigger>, then the <system name> shall <system response>
State-driven requirements	While <in a specific state> the <system name> shall<system response>
Optional features	Where <feature is included> the <system name> shall <system response>

*"The control system **shall** prevent engine overspeed"*
*"**When** continuous ignition is com-manded by the aircraft, **the** control system **shall** switch on continuous ignition"*
*"**If** the computed airspeed fault flag is set, **then the** control system **shall** use modelled airspeed"*
*"**When** selecting idle set-ting, if aircraft data is unavailable**, then the** control system **shall** select Approach Idle"*
*"**While** the aircraft is on-ground, when reverse thrust is commanded, **the** control system **shall** enable deployment of the thrust reverser"*

A. Mavin, P. Wilkinson, A. Harwood and M. Novak, "Easy Approach to Requirements Syntax (EARS)," *2009 17th IEEE International Requirements Engineering Conference*, Atlanta, GA, 2009, pp. 317-322, doi: 10.1109/RE.2009.9.

(my own) Example



(my own) Require Example

- **IT2_MIS_006 - Collaborative RF Triangulation**
  - **Text:** *The ITASAT#2 Mission shall triangulate the position of a collaborative S-Band radio-frequency emitting source.*
  - **Rationale:** The ITASAT#2 second mission, as proposed by the stakeholders, includes the capability to track the position of an S-Band collaborative RF emitting source. This enhances situational awareness, enables technology demonstration for RF localization in space, and validates the spacecraft's ability to support future collaborative operations.
  - **Verification:** Prove that the ITASAT#2 Mission can triangulate the position of a collaborative S-Band radio-frequency emitting source by conducting an analysis. The verification will be performed through simulations of the triangulation algorithm. Simulations will be executed in a mission-representative software environment with RF propagation and orbital geometry models. The requirement will be considered met when simulations show that the algorithm calculates the source position with accuracy within ±[TBD] km.
  - **Originator:** Stakeholder board – Date: xxxx-xx-xx
  - **Owner:** SE Team – IT#2 Project - **Version:** 1.0 - **Approval Date:** [tbd]
  - **Parent Requirement:** IT2_UC_002 (Mission Objective Definition Document)
  - **Allocation:** [tbd]