SYS 5460: Writing Requirements

Contents:

- Writing Requirements
- Requirement Characteristics
- Requirement Checks
- Examples

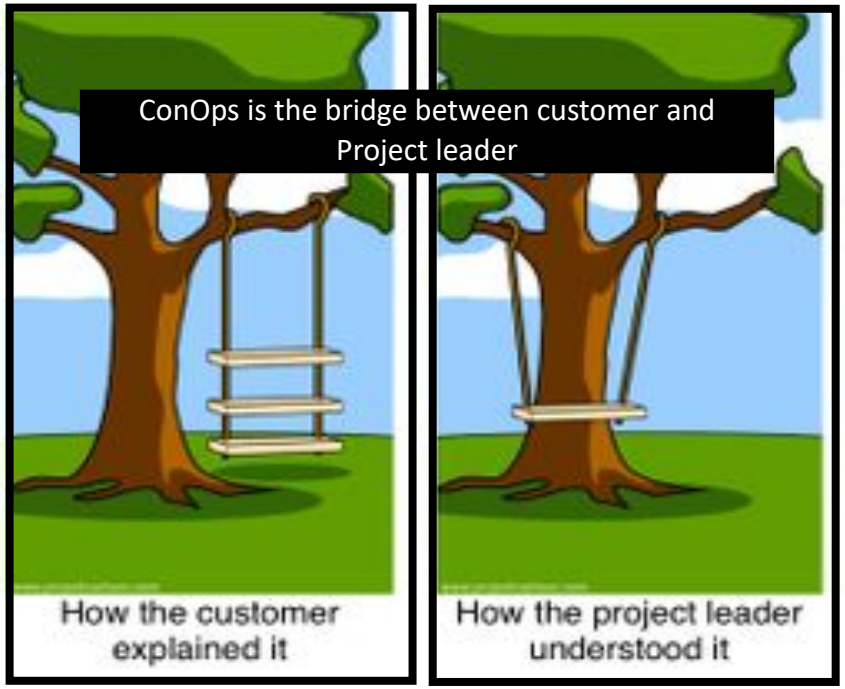
Department of Computer & Engineering Sciences

College of Engineering

Florida Institute of Technology

FLORIDA TECH

ConOps is the bridge between customer and Project leader

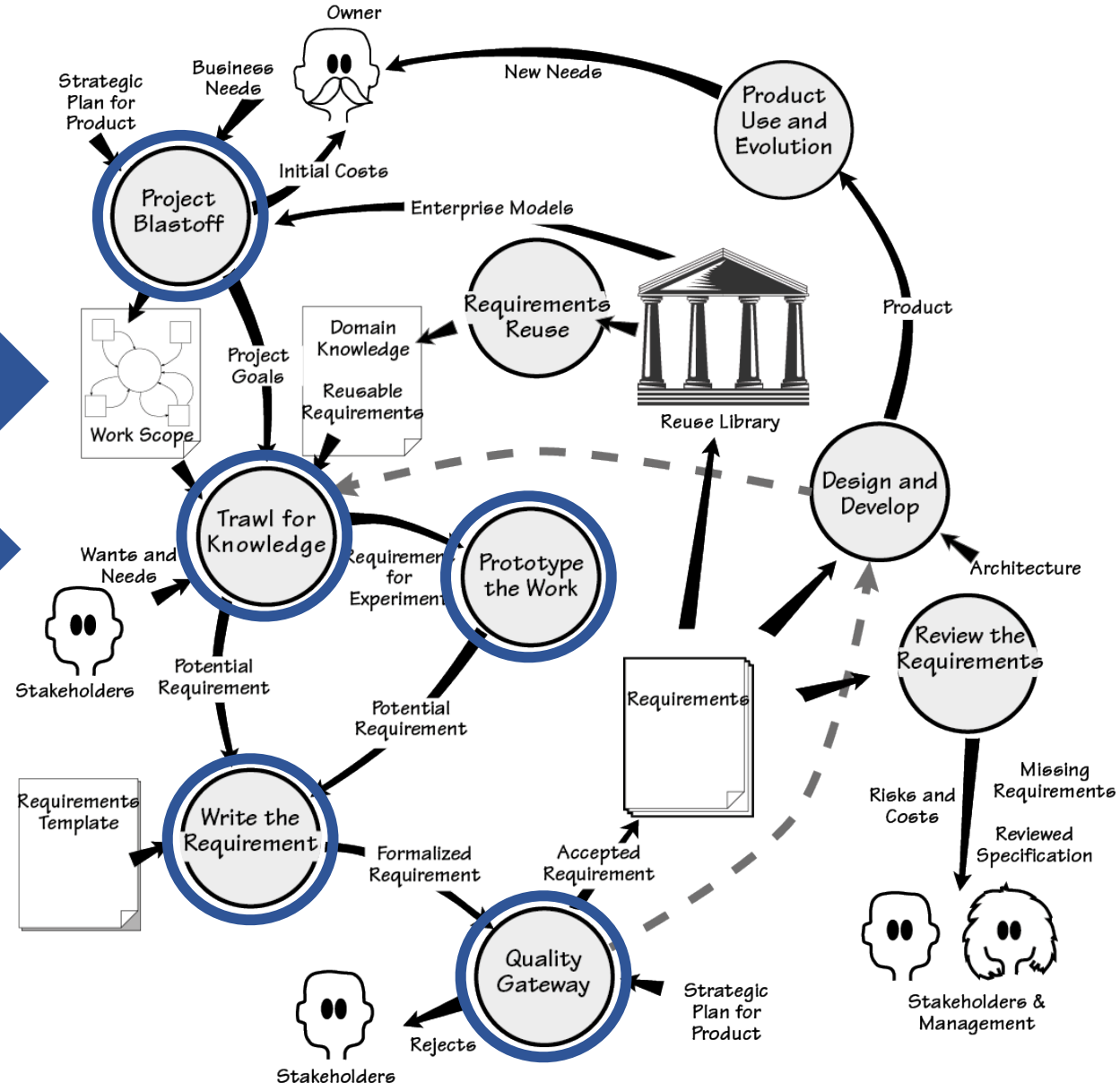


Volere Requirements Process (Robertson)

Model system functionality

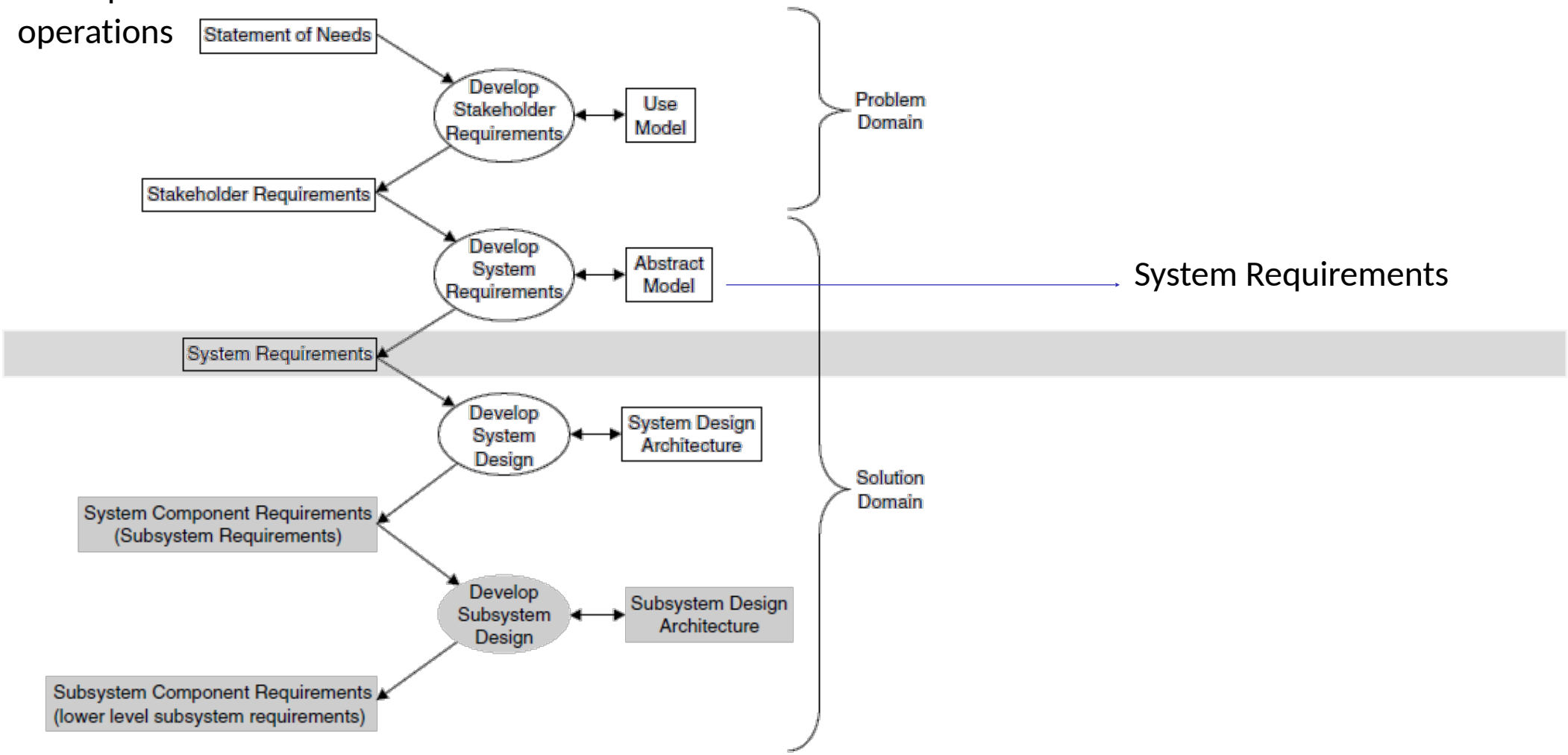
Concept of Operations

Functional Requirements



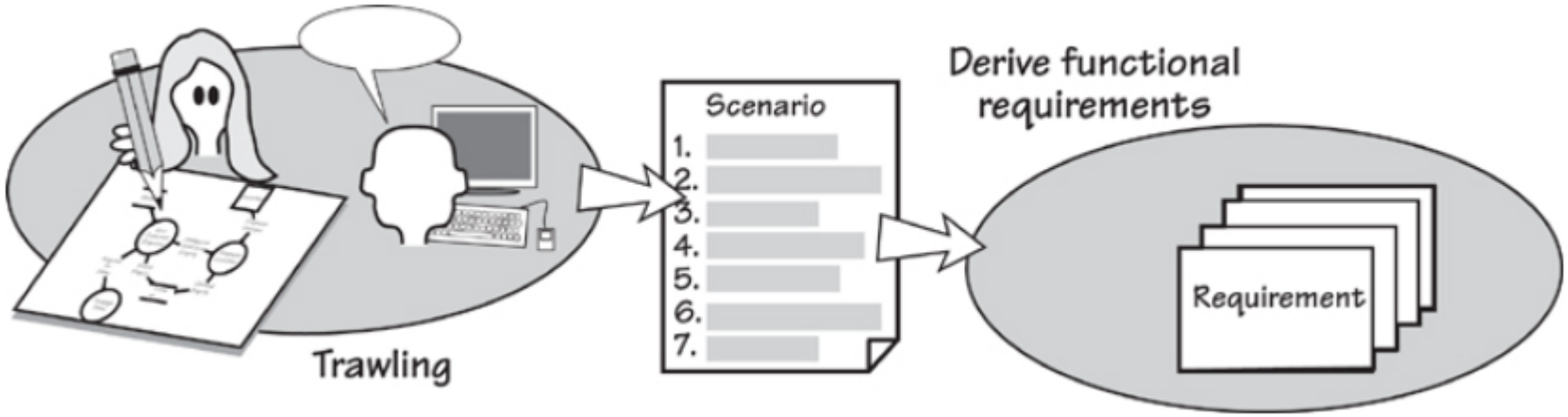
Stakeholders->Concept of Operations->Model

Concept of operations



Prerequisites

- System in the problem space is defined



Chapter 10 Robertson

Functional Requirement

- Functional requirements describe what the product has to do to support and enable the owner's work. They should be, as far as possible, independent of the technology used by the eventual product (Robertson ch 10)

- A statement that identifies what a product or process must accomplish to produce required behavior and/or results. 2. A requirement that specifies a function that a system or system component must be able to perform. (IEEE and ISO/IEC 2010, p. 153)
- Functional requirements have the following essential characteristics:
  - Define what the system should do
  - Be action oriented
  - Describe tasks or activities
  - Are associated with the transformation of inputs to outputs

Writing Functional Requirements

- Requirements are written as a single sentence with a single verb.
- It is common to use shall, must, will, might, could etc. to indicate the priority of the requirement. This may result in semantic confusion.
- Use one consistent form (shall) and use a separate attribute to indicate its priority
- Other practices include
  - Shall: Mandatory
  - Should: Optional
  - Will: Deferred compliance
- When in doubt IEEE
- For requirement modeling additional attributes with priority can be implemented

Requirements Style and Interpretation

- It is common to use shall, must, will, might, could etc. to indicate the priority of the requirement. This may result in semantic confusion.
- Use one consistent form (shall) and use a separate attribute to indicate its priority
- Other practices include
  - **Shall: Mandatory**
  - Should: Optional
  - Will: Deferred compliance
- When in doubt IEEE

Characteristics of Individual Requirements (SEBoK)

Characteristic	Description
Necessary	The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is not included in the set of requirements, a deficiency in capability or characteristic will exist, which cannot be fulfilled by implementing other requirements
Appropriate	The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction). This includes avoiding unnecessary constraints on the architecture or design to help ensure implementation independence to the extent possible
Unambiguous	The requirement is stated in such a way so that it can be interpreted in only one way.
Complete	The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity need without needing other information to understand the requirement
Singular	The requirement should state a single capability, characteristic, constraint, or quality factor.
Feasible	The requirement can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk
Verifiable	The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exists.
Correct	The requirement must be an accurate representation of the entity need from which it was transformed.
Conforming	The individual requirements should conform to an approved standard template and style for writing requirements, when applicable.

Attributes of the Requirements Document

Ability

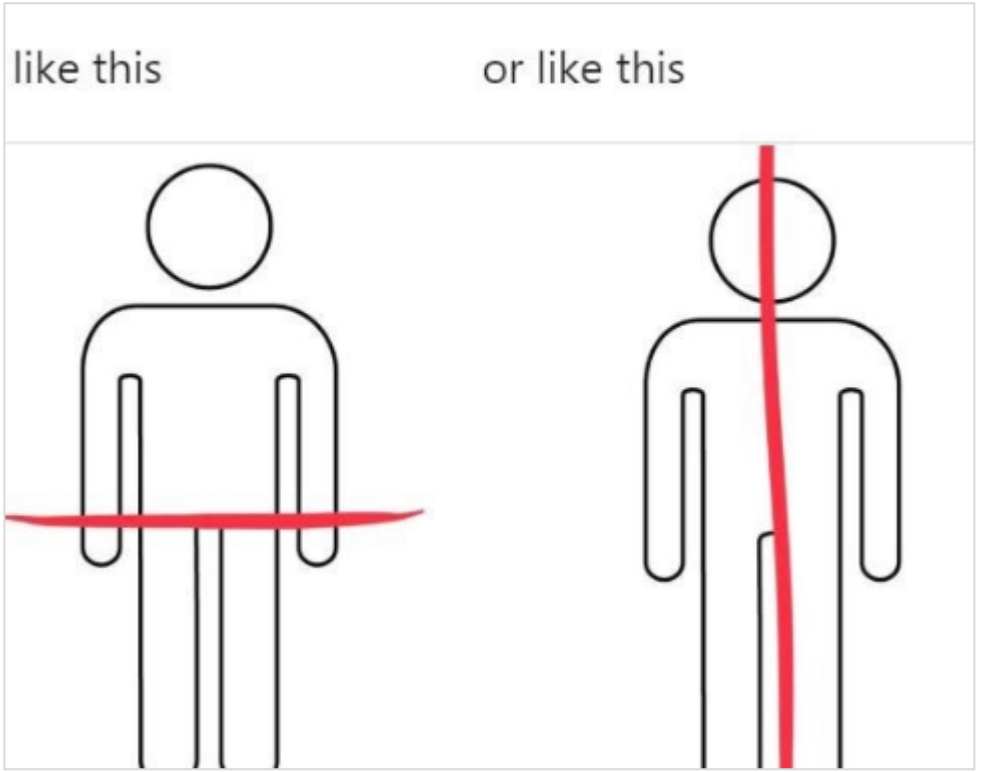
- Ability uniquely to identify every statement of requirement
- Ability to classify every statement of requirement in multiple ways, such as:
  By importance
  By type (*e.g.* functional, performance, constraint, safety)
  By urgency (when it has to be provided)
- Ability to track the status of every statement of requirement, in support of multiple processes, such as:
  Review status
  Satisfaction status
  Qualification status
- Ability to elaborate a requirement in multiple ways, such as by providing:
  Performance information
  Quantification
  Test criteria
  Rationale
  Comments
- Ability to view a statement of requirement in the document context, i.e. alongside its surrounding statements
- Ability to navigate through a requirements document to find requirements according to a particular classification or context
- Ability to trace to any individual statement of requirement

Helps to perform QA on requirements

minimize the number of requirements
understand large amounts of information
find sets of requirements relating to particular topics
detect omissions and duplications
eliminate conflicts between requirements
manage iteration (*e.g.* delayed requirements)
reject poor requirements
evaluate requirements
reuse requirements across projects

Ambiguity…


We plan to cut all homeless people in half by 2025.
Conservatives


like this
or like this

Key objectives when writing requirements

- Make the requirements document readable
- Make the set of requirements processable

Characteristic of a Set of Requirements (

Characteristic	Description
Complete	The requirement set stands alone such that it sufficiently describes the necessary capabilities, characteristics, constraints, and/or quality factors to meet the entity needs without needing other information. In addition, the set does not contain any to be defined (TBD), to be specified (TBS), or to be resolved (TBR) clauses.
Consistent	The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems they use are homogeneous. The language used within the set of requirements is consistent, i.e., the same word is used throughout the set to mean the same thing.
Feasible	The requirement set can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk. (Note: Feasible includes the concept of "affordable".)
Comprehensible	The set of requirements must be written such that it is clear as to what is expected by the entity and its relation to the system of which it is a part.
Able to be validated	It must be able to be proven the requirement set will lead to the achievement of the entity needs within the constraints (such as cost, schedule, technical, legal and regulatory compliance).

More on Requirements Language

- A typical English word has 10 synonyms, so even a 4 word requirement statement has 10,000 possible interpretations

- Bellagamba* suggest the following ambiguity check:

  - Compliance level
  - Completeness
  - Precision
  - Comprehension
  - Referencing
  - Vague words
  - Functional requirement
  - Acronyms
  - English unit usage
  - Word emphasis

*Bellagamba Larry, Systems Engineering and Architecting Ch4 , CRC Press 2012

Example of Requirement Check

- Compliance check (Bellagamba). Check for the following words:
  - Type 1. also, anticipate, apply, applies, are to, aspire, can, could, crave, demand, desire, expect, force, forcing, ideally, goal, got to, has to, is to, might, must, necessary, necessitates, need, needed, needs, obligate, obligation, require, prefer, preference, should, stipulate, want, wants, will, would
  - Type 2. can't, don't, mustn't, needn't, not, shouldn't, won't

Suggested Form

- For type 1:
  - If feature M is mandatory for entity E, use: *The E shall M*
  - If feature O is optional for entity E, use: *The E should O*
  - If feature D is deferred for entity E, use: *The E will D*
- For type 2:
  - Phrase as a statement of inclusion rather than exclusion

Example of Compliance Check (Cont'd)

- If entity E may exhibit feature V with values between V1 and V2, use:
  - *The E shall V between V1 and V2 with equal preference over the range.*

*Bellagamba Larry, Systems Engineering and Architecting Ch4 , CRC Press 2012

Completeness

- Completeness ambiguity is caused when missing information is included in the requirement statement.
  - not known, tbd, tbr, tbs, tbx, to be determined, to be provided, to be reviewed,to be specied, to be
  - Delete the requirement until complete, or provide the incomplete information supplied, unknown

Precision Ambiguity

- Precision ambiguity is uncertainty in how to interpret numerical information
  - Type 1: is above, at least, minimum of, no less than, not less than, not to be less
  - than, exceed.
  - Type 2: is at most, below, maximum, no greater than, not greater than, not to be
  - greater than, under, up to, within.
  - Type 3: is about, almost, approximately, at, between, close to, exactly, give or
  - take, more or less, near, of, or so, plus or minus, roughly, tolerance, use, +/–.
  - Type 4: is average.

- For type 1 (feature **F** to be >= **L)**:
  - The probability **F** is greater than (or equal to) **L** shall be **p**
- For type 2 (feature **F** to be <= **U)**:
  - The probability **F** is less than (or equal to) **U**, shall be **p**
- For type 3 (feature **F** to be C|feature **F** >(=) **L** and F<(=) **U)**:
  - The **F** shall be **C**
  - The probability **F** is greater than (or equal) to **L** and less than (or equal) to **U** shall be **p**

- For average:
  - use mean, median, mode, and arithmetic or geometric, or explain how average is to be determined

Comprehension Ambiguity

- Type 1. Using more than two instances of *shall*, a semicolon, or a colon.
- Type 2. Using more than two instances of *and* or *or*.
- Type 3. Using more than one *but*.

- Sugestions:
- If (C1 then) the E1 (shall) be A1 (else if C2 then) the E2 (shall) be A2
  - Ci: conditions
  - Ei: entities
  - Ai: Attrbutes

ISO/IEC/IEEE 29148:2011

- Preferences or goals are desired, non-mandatory, non-binding provisions and use 'should‘
- Suggestions or allowances are non-mandatory, non-binding provisions and use 'may‘
- Non-requirements, such as descriptive text, use verbs such as ‘are', ‘is', and ‘was'. It is best to avoid using the term ‘must', due to potential misinterpretation as a requirement
- Use positive statements and avoid negative requirements such as ‘shall not‘
- Use active voice: avoid using passive voice, such as 'shall be able to select'

ISO/IEC/IEEE 29148:2011 Recommendations for a Set of Requirements

- Complete. The set of requirements needs no further amplification because it contains everything pertinent to the definition of the system or system element being specified. In addition, the set contains no To Be Defined (TBD), To Be Specified (TBS), or To Be Resolved (TBR) clauses

- Consistent. The set of requirements does not have individual requirements which are contradictory. Requirements are not duplicated. The same term is used for the same item in all requirements

- Affordable. The complete set of requirements can be satisfied by a solution that is obtainable/feasible within life cycle constraints (e.g., cost, schedule, technical, legal, regulatory)

- Bounded. The set of requirements maintains the identified scope for the intended solution without increasing beyond what is needed to satisfy user needs

ISO/IEC/IEEE 29148:2011 Recommendations for Requirement Language (Avoid)

- Superlatives (such as 'best', 'most')
- Subjective language (such as 'user friendly', 'easy to use', 'cost effective')
- Vague pronouns (such as 'it', 'this', 'that')
- Ambiguous adverbs and adjectives (such as 'almost always', 'significant', 'minimal')
- Open-ended, non-verifiable terms (such as 'provide support', 'but not limited to', 'as a minimum')
- Comparative phrases (such as 'better than', 'higher quality')
- Loopholes (such as 'if possible', 'as appropriate', 'as applicable')
- Incomplete references (not specifying the reference with its date and version number; not specifying just the applicable parts of the reference to restrict verification work)
- Negative statements (such as statements of system capability not to be provided) All as

Examples

1. The assignment is due on March 2
2. An aircraft that is non-friendly and has an unknown mission or the potential to enter restricted airspace within 5 minutes shall raise an alert.
3. All lights shall have their switch
4. The trucks shall treat the roads before they freeze
5. The system shall shut off the pumps if the water level remains above 100 meters for more than 4 seconds
6. The product shall show the weather for the next 24 hours
7. The product shall communicate all roads predicted to freeze.

Discussion

Provide creative implementations to the requirement

Suggestions to avoid ambiguity

- Requirements elicitation and requirements documentation
  - Eliminate all pronouns from your requirements and replace them with the subject or object to which the pronouns refer
  - Read it aloud. If possible, have a colleague read it aloud
  - Confirm with your stakeholder that you both reach the same understanding of the requirement
- Keep in mind that you are writing a *description* of the requirement. The real requirement is revealed when you write the full documentation (fit criterion+ rationale)
- Establish a context, because language is interpreted always in context, and if this context is not made explicit and agreed to by all the stakeholders in an elicitation session, misinterpretations are likely

> Requirement engineer's paraphrasing of customers' and users' statements in her own words is an effective way for the requirements engineer to get the customers and users to spot their own ambiguities

Suggestions to avoid ambiguity (Continued)

- Increase precision of natural language
- Provide contextual information (rationale)
- Provide context and data definitions
- Look for patterns of ambiguity in the existing documentation
- Compare (and share) interpretation from different stakeholders

Using Requirements Templates or Boiler Plates

- [when?] [under what conditions?] THE SYSTEM SHALL | SHOULD | WILL <process> <thing to be processed>[<process detail>*]

[Condition] [Subject] [Action] [Object] [Constraint]

EXAMPLE: When signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** the signal x received bit **[Object]** within 2 seconds **[Constraint].**

Or

[Condition] [Action or Constraint] [Value]

EXAMPLE: At sea state 1 **[Condition]**, the Radar System shall detect targets at ranges out to **[Action or Constraint]** 100 nautical miles **[Value]**.

Or

[Subject] [Action] [Value]

EXAMPLE: The Invoice System **[Subject]**, shall display pending customer invoices **[Action]** in ascending order **[Value]** in which invoices are to be paid.

Templates by Hull (Dick & Hull 2017)

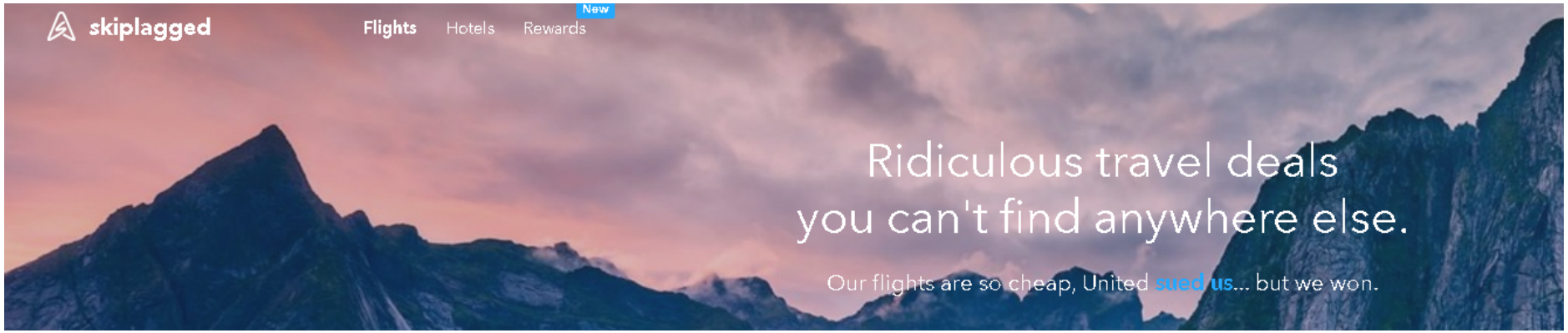
- For Stakeholder capabilities

The <stakeholder type> shall be able to <capability>
within <performance> of <event>
while <operational condition>.

The *weapons operator* **shall be able to** *fire a missile*
within 3 *seconds* **of** *radar sighting*
while *in severe sea conditions.*

The <stakeholder> shall not be placed
in breach of <applicable law>.

E.g. **The** *ambulance driver* **shall not be placed**
in breach of *national road regulations.*



Find flights the airlines don't want you to see.

We're exposing loopholes in airfare pricing to save you money.

Templates by Hull (Dick & Hull 2017)

- For System capabilities

The <system> shall <function>
not less than <quantity> <object>
while <operational condition>.

E.g. **The** *communications system* **shall** *sustain telephone contact*
with not less than *10 callers*
while *in the absence of external power.*

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

E.g. **The** *coffee machine* **shall** *produce a hot drink*
every *10 seconds.*

Requirement Expressions

- Requirement Expression= Requirement statement + attributes

Category	Example values
Identification	
• Identifier	Unique reference
• Name	Unique name summarising the subject of the requirement
Intrinsic characteristics	
• Basic type	Functional, performance, quality factor, environment, interface, constraint, non-requirement
• Quality factor sub-type	Availability, flexibility, integrity, maintainability, portability, reliability, safety, security, supportability, sustainability, usability, workmanship
• Product/process type	Product, process, data, service
• Quantitative/ qualitative type	Quantitative, qualitative
• Life-cycle phase	Pre-concept, concept, development, manufacturing, integration/test, deployment/delivery/installation, operation, support, disposal
Priority and importance	
• Priority (compliance level)	Key, mandatory, optional, desirable *or* Must, should, could, wish (MoSCoW)
• Importance	1 to 10
Source and ownership	
• Derivation type	Allocation, decomposition
• Source (origin)	Name of document or stakeholder

Category	Example values
• Owner	Name of stakeholder
• Approval authority	Name or person
Context	
• Requirements set/ document	(Best handled through positioning the requirement in a structured document)
• Subject	
• Scope	
Verification and validation	
• V&V method	Analysis, inspection, system test, component test
• V&V stage	(See life-cycle phase)
• V&V status	Pending, pass, failed, inconclusive
• Satisfaction argument	Rationale for choice of decomposition
• Validation argument	Rationale for choice of V&V methods
Process support	
• Agreement status	Proposed, being assessed, agreed
• Qualification status	Not qualified, qualified, suspect
• Satisfaction status	Not satisfied, satisfied, suspect
• Review status	To be reviewed, accepted, rejected
Elaboration	
• Rationale	Textual statement about why the requirement is present
• Comments	Textual comments of clarification
• Questions	Questions to be posed for clarification

Requirement Expressions (continued)

Elaboration	
• Rationale	Textual statement about why the requirement is present
• Comments	Textual comments of clarification
• Questions	Questions to be posed for clarification
• Responses	Responses received for clarification
Miscellaneous	
• Maturity (stability)	Number of changes/time
• Risk level	High, medium, low
• Estimated cost	
• Actual cost	
• Product release	Version(s) of product meeting the requirement

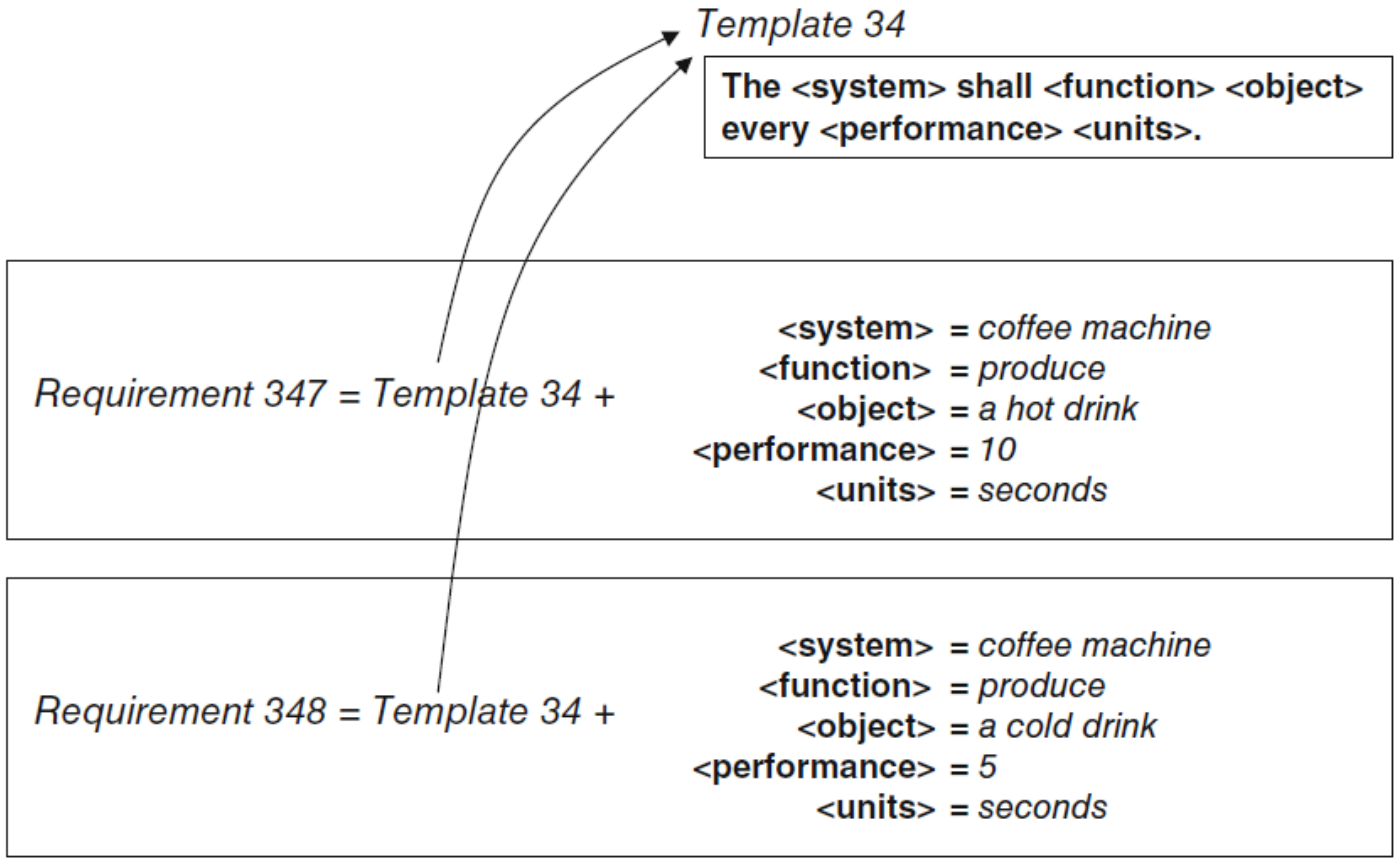
Extensions Attributes in SysML (as of SysML 1.6)

Table E-4: Requirement property enumeration types

Enumeration	Enumeration Literals	Example Description
RiskKind	High	High indicates an unacceptable level of risk
	Medium	Medium indicates an acceptable level of risk
	Low	Low indicates a minimal level of risk or no risk
VerificationMethodKind	Analysis	Analysis indicates that verification will be performed by technical evaluation using mathematical representations, charts, graphs, circuit diagrams, data reduction, or representative data. Analysis also includes the verification of requirements under conditions, which are simulated or modeled; where the results are derived from the analysis of the results produced by the model.
	Demonstration	Demonstration indicates that verification will be performed by operation, movement or adjustment of the item under specific conditions to perform the design functions without recording of quantitative data. Demonstration is typically considered the least restrictive of the verification types.
	Inspection	Inspection indicates that verification will be performed by examination of the item, reviewing descriptive documentation, and comparing the appropriate characteristics with a predetermined standard to determine conformance to requirements without the use of special laboratory equipment or procedures.
	Test	Test indicates that verification will be performed through systematic exercising of the applicable item under appropriate conditions with instrumentation to measure required parameters and the collection, analysis, and evaluation of quantitative data to show that measured parameters equal or exceed specified requirements.

Custom Templates or Boilerplates

- Collect patterns from the organization and type of product to create a collection of requirements

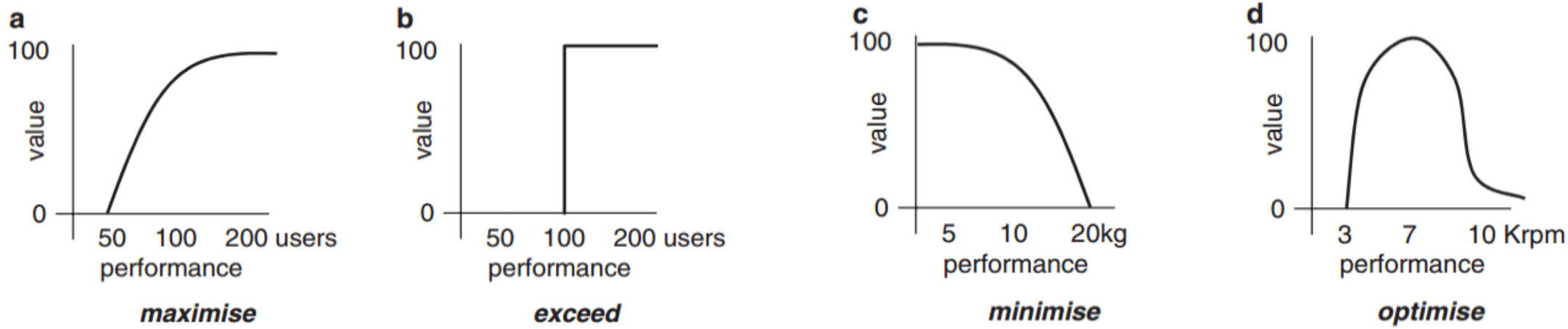


Examples (Hull Ch 4)

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>**.

Requirements Value

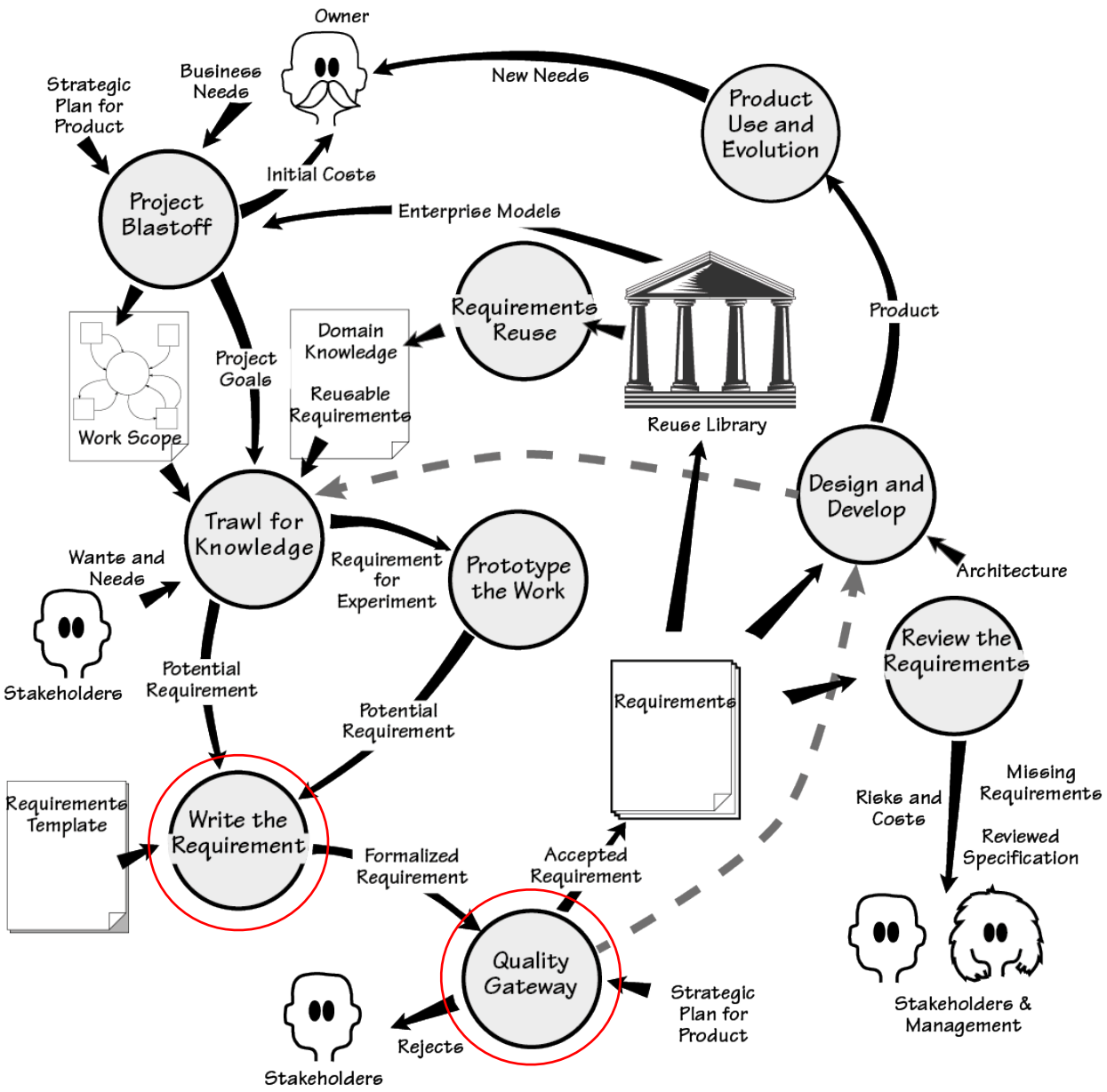
- Functional requirements are non-negiotiable. If they are not met , the product is of no use
- Other requirements may be negotiable
  - One approach is to provide performance values (Mandatory, Desired, Best)
  - Another proposed approach is to represent the value of a requirement value by supplying a function that maps performance to value

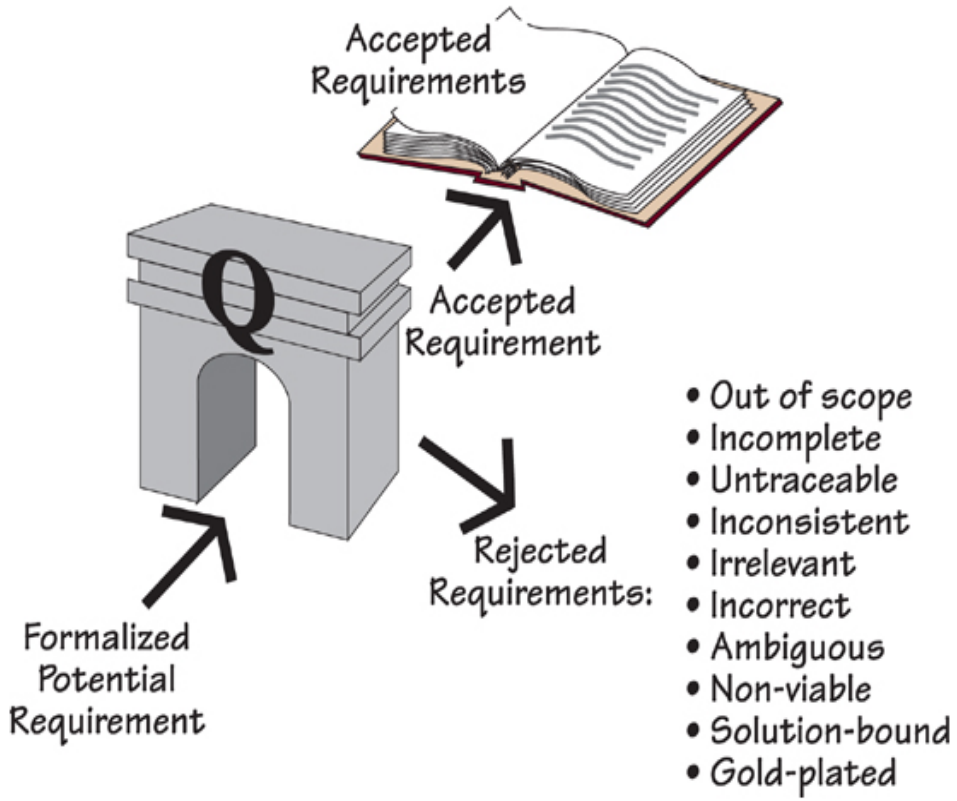


Summary

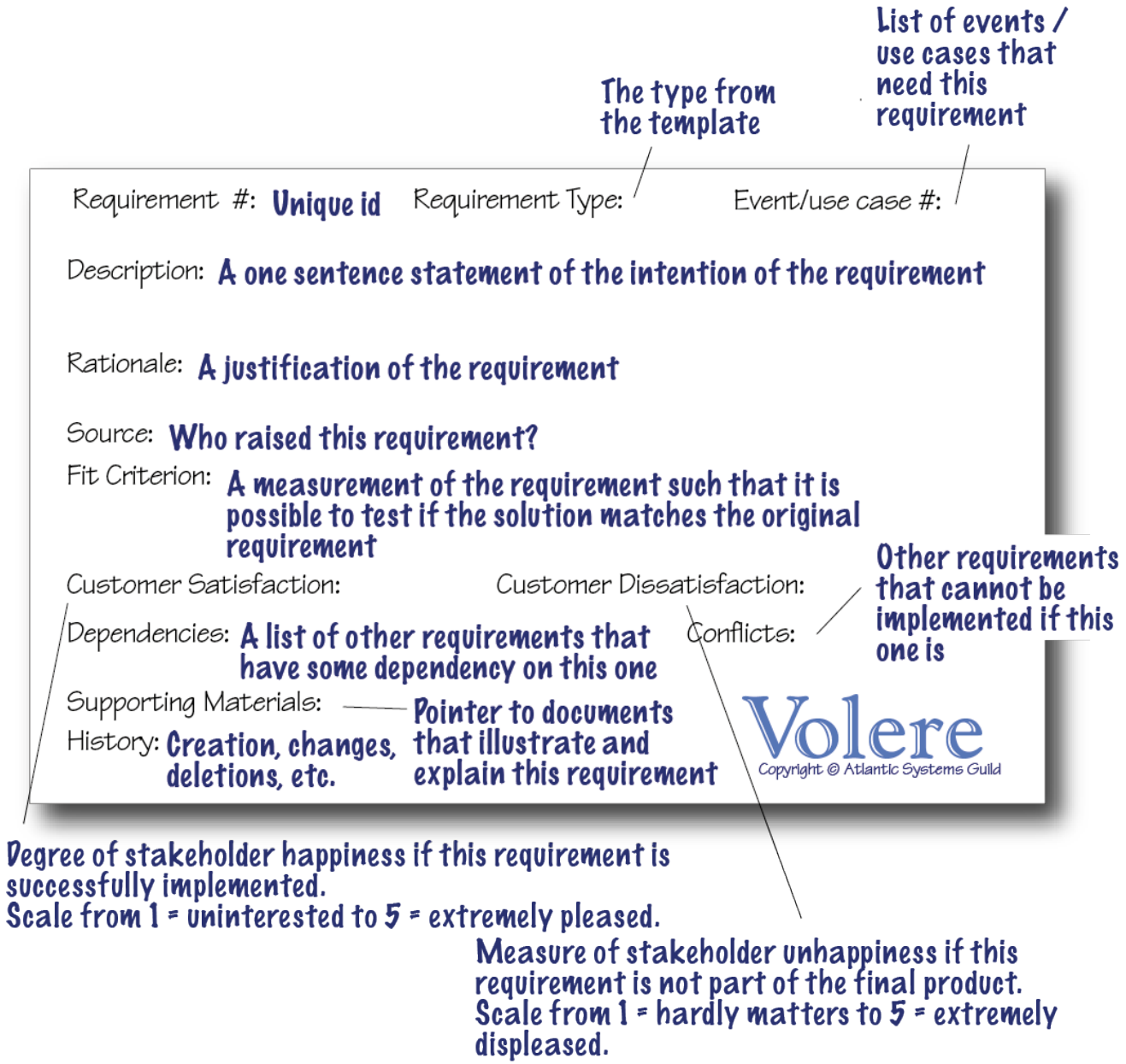
- Make requirements readable by applying existing rules
- Check INCOSE and IEEE for additional recommendations and up-to-date information
- Requirements are not only for humans, NLP engines can consume requirements too!!
- When gathering requirements paraphrase and see if it make sense
- Obtain agreement and consensus from stakeholders
- Try to create patterns for requirements

Where are we?


Owner
Strategic Plan for Product
Business Needs
New Needs
Product Use and Evolution
Project Blastoff
Initial Costs
Enterprise Models
Requirements Reuse
Domain Knowledge
Reusable Requirements
Reuse Library
Product
Project Goals
Work Scope
Design and Develop
Trawl for Knowledge
Wants and Needs
Requirement for Experiment
Prototype the Work
Architecture
Stakeholders
Potential Requirement
Review the Requirements
Requirements
Potential Requirement
Requirements Template
Write the Requirement
Risks and Costs
Missing Requirements
Reviewed Specification
Formalized Requirement
Accepted Requirement
Quality Gateway
Strategic Plan for Product
Stakeholders & Management
Rejects
Stakeholders


Accepted Requirements
Accepted Requirement
Q
Formalized Potential Requirement
Rejected Requirements:
• Out of scope
• Incomplete
• Untraceable
• Inconsistent
• Irrelevant
• Incorrect
• Ambiguous
• Non-viable
• Solution-bound
• Gold-plated

Volere Requirement Card



Example of a Finished Requirement

Requirement #: 75 Requirement Type: 9 Event/BUC/PUC #: 7, 9

Description: The product shall record all the roads that have been treated

Rationale: To be able to schedule untreated roads and highlight potential

Originator: Arnold Snow - Chief Engineer

Fit Criterion: The recorded treated roads shall agree with the drivers' road treatment logs and shall be up to date within 30 minutes of the completion of the road's treatment

Customer Satisfaction: 3 Customer Dissatisfaction: 5

Dependencies: All requirements using road and scheduling data Conflicts: 105

Supporting Materials: Work context diagram, terms definitions in section 5

History: Created February 29,2010

Volere