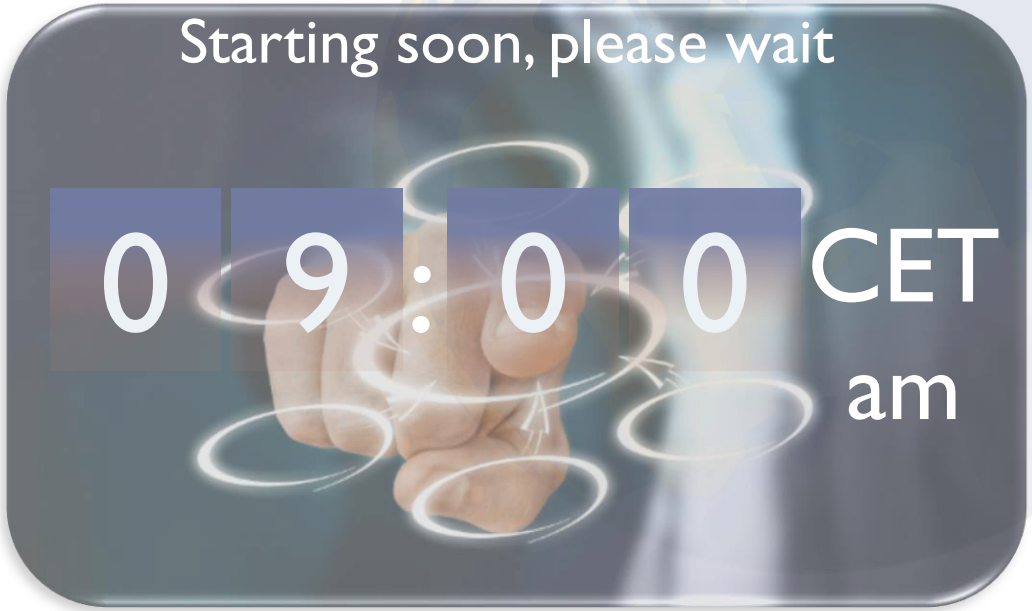
Introduction: Webinar rules



- Webinar rules:
  - You'll be muted all along the Webinar
  - There's a chatting box to ask your questions or send your comments when you want
  - Please address these comments and questions to the user "The REUSE Company" and not to the presenter directly
  - If you have any technical issue please use this chatting box, or mail us at: support@reusecompany.com
  - The Webinar will be recorded. A link to the recording will be sent to you in few days

TRC WEBINARS 2020

TRC

WEBINARS 2020

Writing requirements with a Knowledge Library based on the NASA Systems Engineering Handbook

Tuesday, 12 May, 2020

TRC WEBINARS 2020

Presenters' profile

- Luis Javier Muñoz
  - Sales and Consulting Engineer, The REUSE Company

Luis J. Muñoz
luis.munoz@reusecompany.com

- Cecilia Karlsson
  - Marketing & Communication

Cecilia Karlsson
cecilia.karlsson@reusecompany.com

Table of Contents

Providing a **knowledge centric** approach to leverage system engineering activities in our customers

contact@reusecompany.com

@reusecompany

Calle Margarita Salas, 16 2-D
28919 – Leganés (Madrid)
SPAIN

+34 912 172 596

About The REUSE Company (TRC)

01 The company was created in **1999**

As a spin-off of a local university in Madrid (Spain)

02 **System + Software Engineers**

Smart combination between Company staff and R&D from Academia

03 **Head Quarters:** Madrid (Spain)

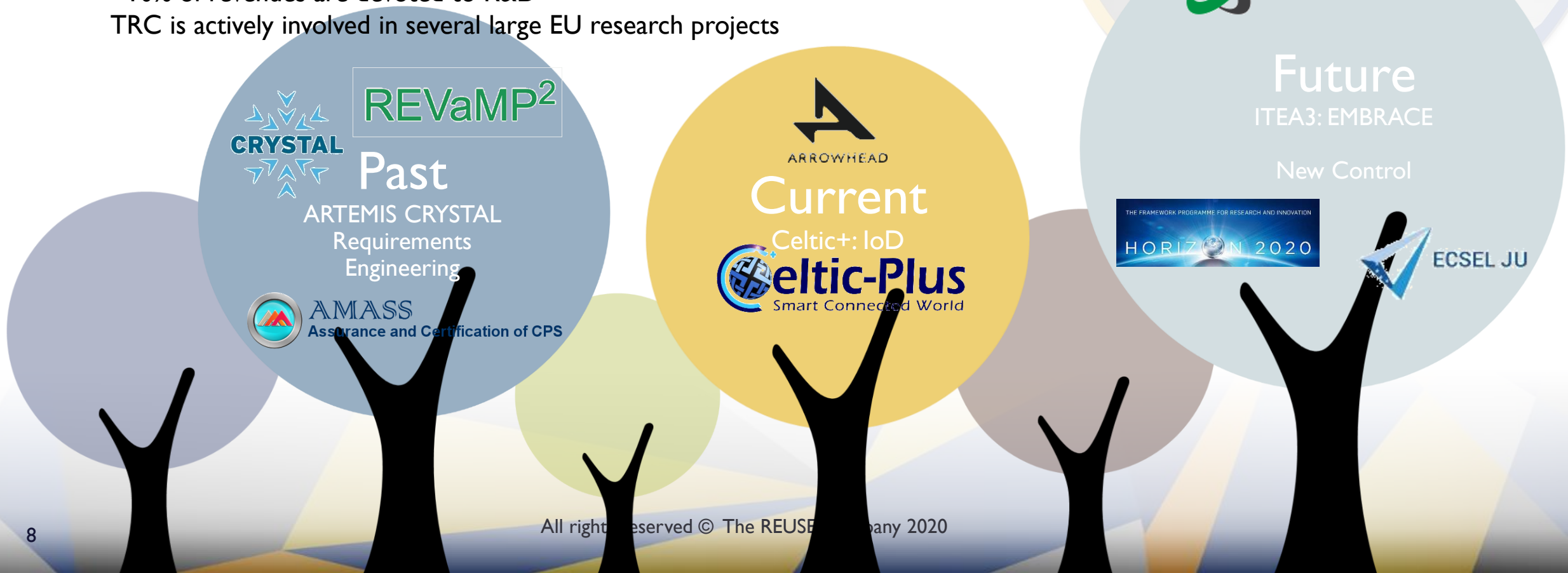
International offices: London (UK) Stockholm (Sweden)

04 Offering a **knowledge centric** approach to leverage system engineering activities in our customers

Research and innovation in our DNA. Public projects

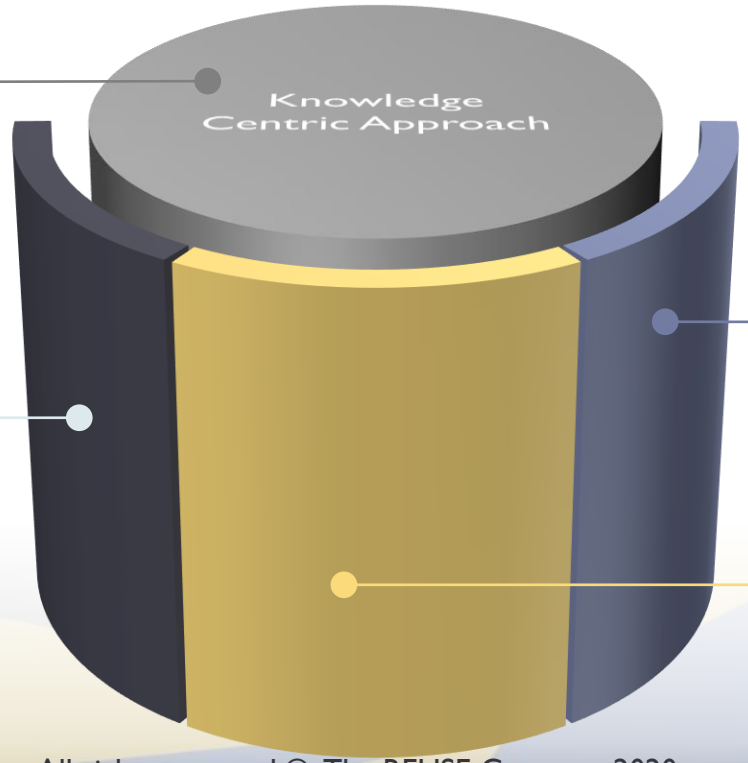
Research and Innovation in our DNA

- Spin-off of Carlos III University of Madrid
- TRC's headquarter is in the Legatec Technology Park of the University
- ≈10% of revenues are devoted to R&D
- TRC is actively involved in several large EU research projects



T (he) R (euse) Q (ompany)

Leveraging Systems Engineering activities

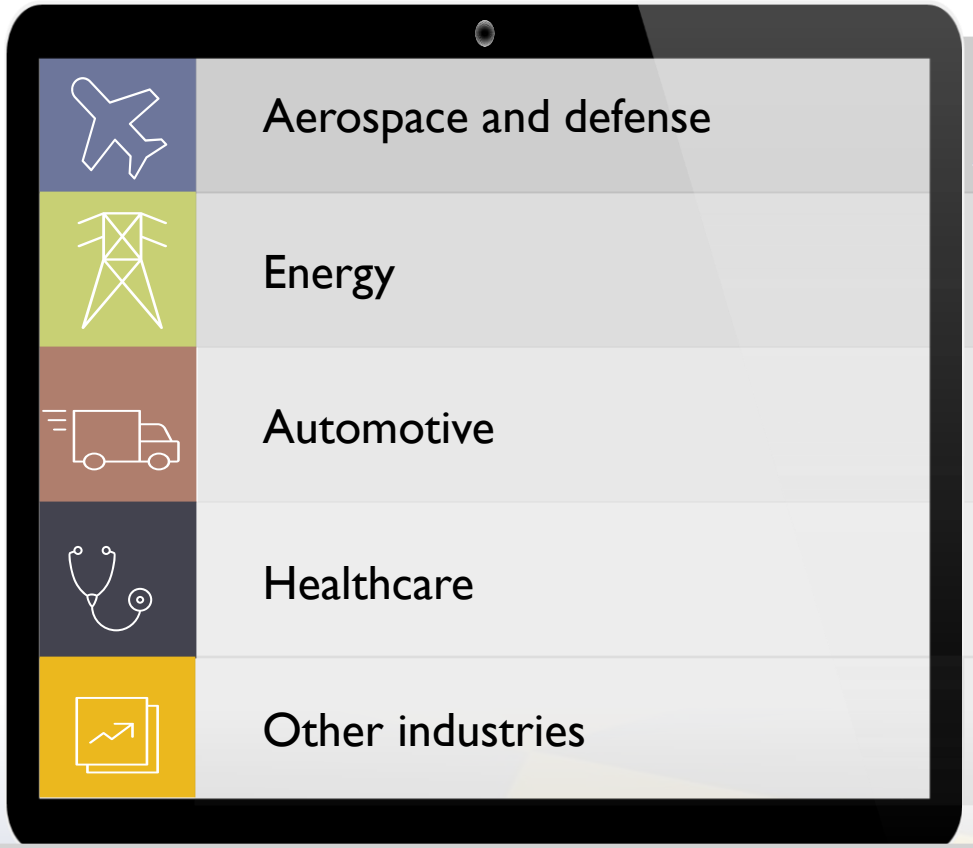


TRACEABILITY

QUALITY

REUSABILITY

Who is using our technology?



AIRBUS
AIRBUS DEFENCE & SPACE
AIRBUS GROUP INNOVATIONS
SAFRAN AIRCRAFT ENGINES
THALES
HENSOLDT
THALES Communications & Security S.A.S.
ariane GROUP
FUSION FOR ENERGY
iter the way to new energy
EDF
REPSOL
RENAULT
THINK SURGICAL
Health Net
SIEMENS
acciona Agua
orange
rtve
ELRA European Land Registry Association
tirant lo blanch

Luis Javier MUÑOZ

- **Sales & Consulting Engineer** at The REUSE Company.
- Luis Javier has experience in **Aeronautics Engineering** and systems design in different industrial sectors such as aeronautics, defense and space.
- Luis Javier's main missions are: **international sales** of our systems engineering solutions, **consulting** of our customers **and account management**.
- His main interests include **knowledge** management, **aerospace** engineering, **requirements** engineering, and **System Engineering processes**.

TRC WEBINARS 2020

TRC

WEBINARS 2020

Writing requirements with a Knowledge Library based on the NASA Systems Engineering Handbook

Tuesday, 12 May, 2020

NASA SYSTEMS ENGINEERING HANDBOOK

- After some events and reports **NASA Office of the Chief Engineer (OCE)** took the initiative to **improve the overall Agency systems engineering** infrastructure and capability for the efficient and effective engineering of NASA systems.
- As part of this initiative the **initial writing** of NASA/SP-6105 was in **1995**.
- **Latest** version is **NASA SP-2016-6105 Rev2** supersedes SP-2007-6105 Rev 1.
- **Objectives** highlighted in the handbook:
  - *"to provide **general guidance and information on systems engineering** that will be useful to the NASA community."*
  - *"to bring the **fundamental concepts and techniques of systems engineering** to NASA personnel in a way that recognized the nature of NASA systems and the NASA environment"*

- **NASA** defines **Systems Engineering** in the Handbook as:
  - ***"a methodical, multi-disciplinary approach for the design, realization, technical management, operations, and retirement of a system. A "system" is the combination of elements that function together to produce the capability required to meet a need."***

Requirements at the NASA design process

- Handbook section 4.0 System Design Processes **identified 4 design processes.**

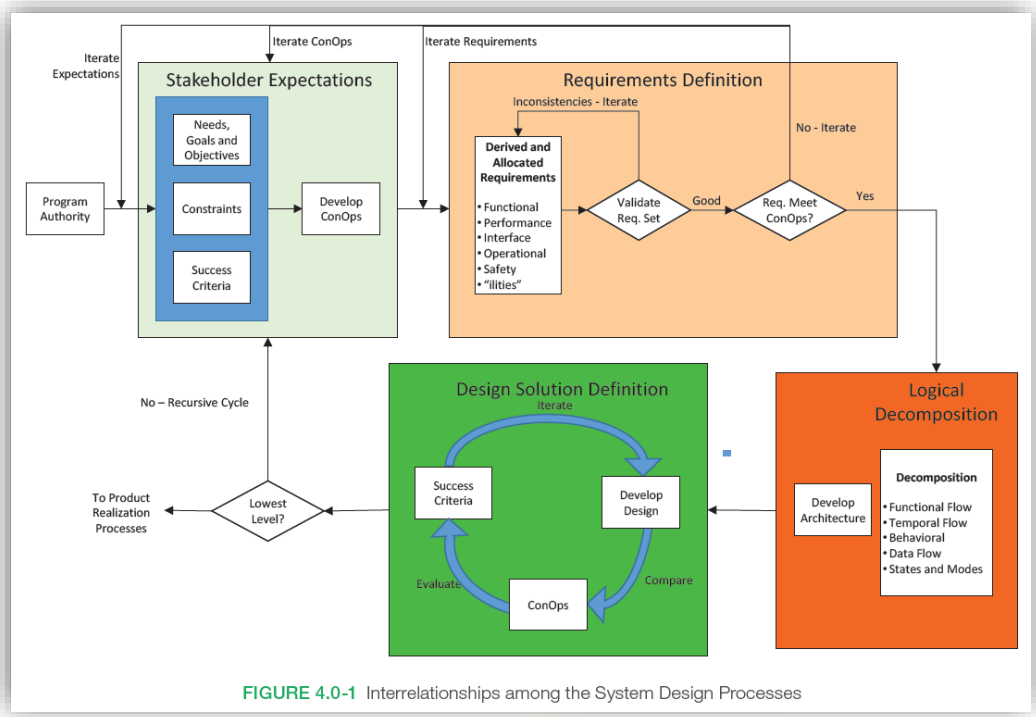


FIGURE 4.0-1 Interrelationships among the System Design Processes

Source: NASA Systems Engineering Handbook SP-2016-6105 Rev2

"The Technical Requirements Definition Process **transforms the stakeholder expectations** into a definition of the problem and then **into a complete set of validated technical requirements** expressed as "shall" statements that can be used for defining a design solution for the Product Breakdown Structure (PBS) and related enabling products."

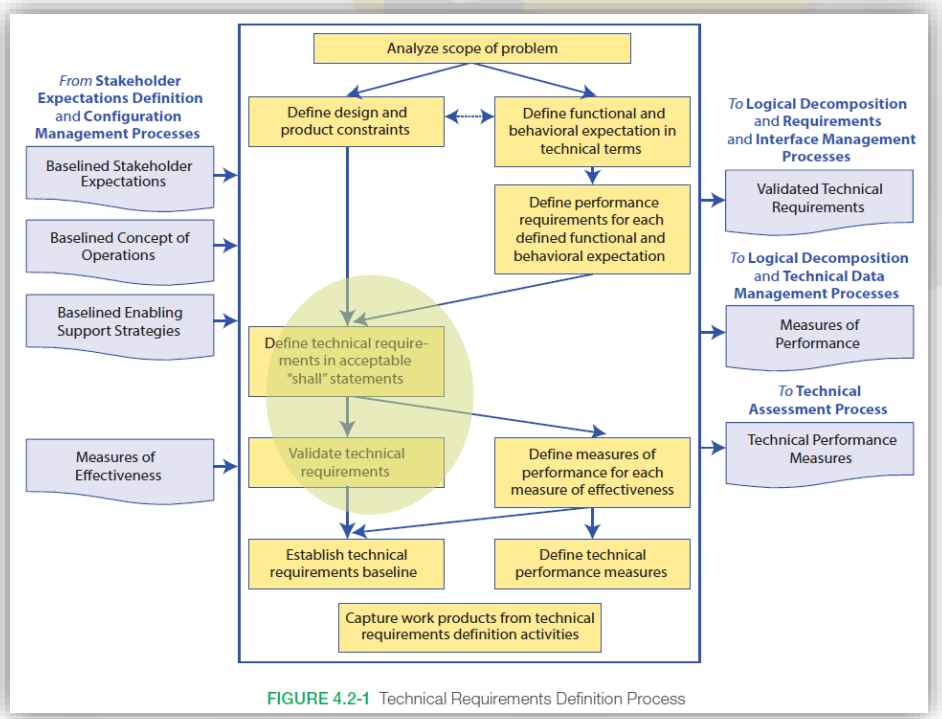
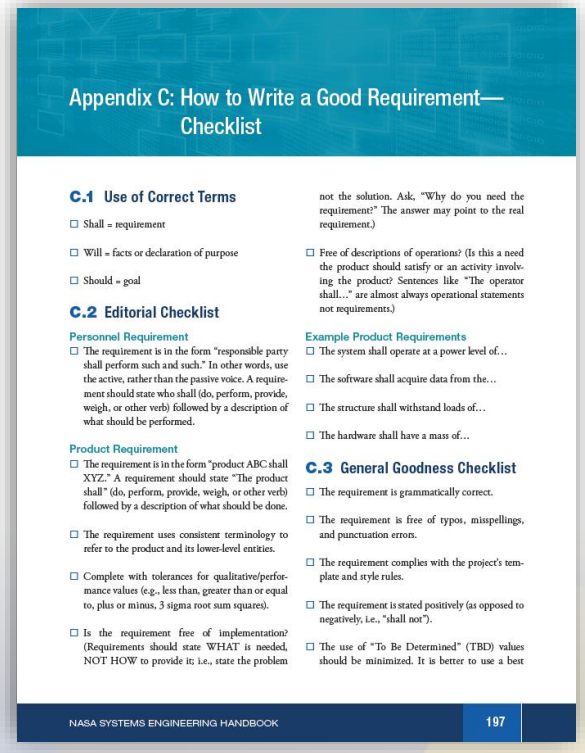


FIGURE 4.2-1 Technical Requirements Definition Process

Source: NASA Systems Engineering Handbook SP-2016-6105 Rev2

Requirements at the NASA design process

- **Handbook section 4.2.1.2.3 Define Requirements in Acceptable Statements**
- *"the requirements should be defined in acceptable "shall" statements, which are complete sentences with a single "shall" per statement. Rationale for the requirement should also be captured to ensure the reason and context of the requirement is understood."*
- **Appendix C: How to Write a Good Requirement— Checklist**



Appendix C: How to Write a Good Requirement—Checklist

C.1 Use of Correct Terms

- ☐ Shall = requirement
- ☐ Will = facts or declaration of purpose
- ☐ Should = goal

C.2 Editorial Checklist

Personnel Requirement

- ☐ The requirement is in the form "responsible party shall perform such and such." In other words, use the active, rather than the passive voice. A requirement should state who shall (do, perform, provide, weigh, or other verb) followed by a description of what should be performed.

Product Requirement

- ☐ The requirement is in the form "product ABC shall XYZ." A requirement should state "The product shall" (do, perform, provide, weigh, or other verb) followed by a description of what should be done.
- ☐ The requirement uses consistent terminology to refer to the product and its lower-level entities.
- ☐ Complete with tolerances for qualitative/performance values (e.g., less than, greater than or equal to, plus or minus, 3 sigma root sum squares).
- ☐ Is the requirement free of implementation? (Requirements should state WHAT is needed, NOT HOW to provide it; i.e., state the problem not the solution. Ask, "Why do you need the requirement?" The answer may point to the real requirement.)
- ☐ Free of descriptions of operations? (Is this a need the product should satisfy or an activity involving the product? Sentences like "The operator shall…" are almost always operational statements not requirements.)

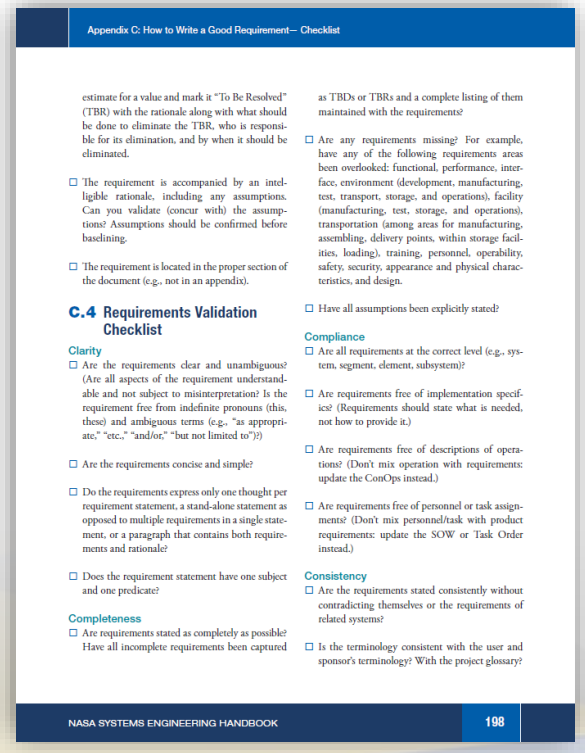
Example Product Requirements

- ☐ The system shall operate at a power level of…
- ☐ The software shall acquire data from the…
- ☐ The structure shall withstand loads of…
- ☐ The hardware shall have a mass of…

C.3 General Goodness Checklist

- ☐ The requirement is grammatically correct.
- ☐ The requirement is free of typos, misspellings, and punctuation errors.
- ☐ The requirement complies with the project's template and style rules.
- ☐ The requirement is stated positively (as opposed to negatively, i.e., "shall not").
- ☐ The use of "To Be Determined" (TBD) values should be minimized. It is better to use a best

NASA SYSTEMS ENGINEERING HANDBOOK 197



Appendix C: How to Write a Good Requirement— Checklist

estimate for a value and mark it "To Be Resolved" (TBR) with the rationale along with what should be done to eliminate the TBR, who is responsible for its elimination, and by when it should be eliminated.

- ☐ The requirement is accompanied by an intelligible rationale, including any assumptions. Can you validate (concur with) the assumptions? Assumptions should be confirmed before baselining.
- ☐ The requirement is located in the proper section of the document (e.g., not in an appendix).

C.4 Requirements Validation Checklist

Clarity

- ☐ Are the requirements clear and unambiguous? (Are all aspects of the requirement understandable and not subject to misinterpretation? Is the requirement free from indefinite pronouns (this, these) and ambiguous terms (e.g., "as appropriate," "etc.," "and/or," "but not limited to")?)
- ☐ Are the requirements concise and simple?
- ☐ Do the requirements express only one thought per requirement statement, a stand-alone statement as opposed to multiple requirements in a single statement, or a paragraph that contains both requirements and rationale?
- ☐ Does the requirement statement have one subject and one predicate?

Completeness

- ☐ Are requirements stated as completely as possible? Have all incomplete requirements been captured as TBDs or TBRs and a complete listing of them maintained with the requirements?
- ☐ Are any requirements missing? For example, have any of the following requirements areas been overlooked: functional, performance, interface, environment (development, manufacturing, test, transport, storage, and operations), facility (manufacturing, test, storage, and operations), transportation (among areas for manufacturing, assembling, delivery points, within storage facilities, loading), training, personnel, operability, safety, security, appearance and physical characteristics, and design.
- ☐ Have all assumptions been explicitly stated?

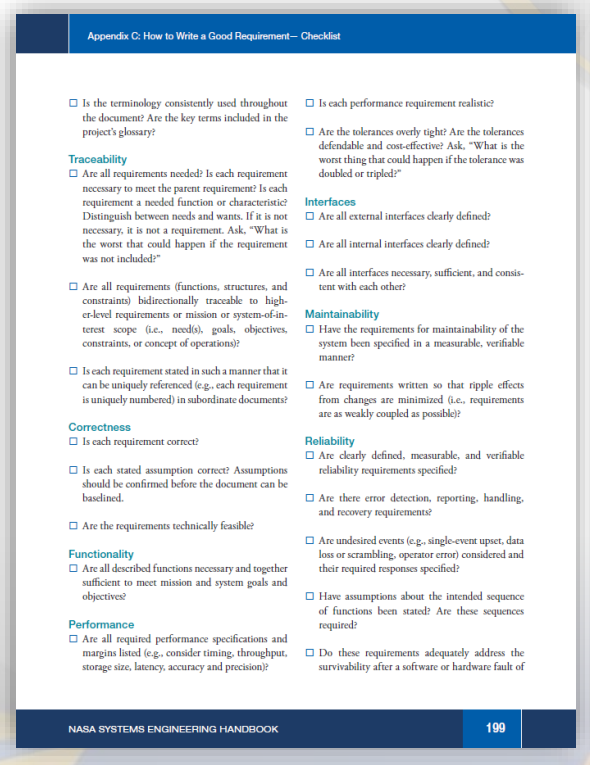
Compliance

- ☐ Are all requirements at the correct level (e.g., system, segment, element, subsystem)?
- ☐ Are requirements free of implementation specifics? (Requirements should state what is needed, not how to provide it.)
- ☐ Are requirements free of descriptions of operations? (Don't mix operation with requirements: update the ConOps instead.)
- ☐ Are requirements free of personnel or task assignments? (Don't mix personnel/task with product requirements: update the SOW or Task Order instead.)

Consistency

- ☐ Are the requirements stated consistently without contradicting themselves or the requirements of related systems?
- ☐ Is the terminology consistent with the user and sponsor's terminology? With the project glossary?

NASA SYSTEMS ENGINEERING HANDBOOK 198



Appendix C: How to Write a Good Requirement— Checklist

- ☐ Is the terminology consistently used throughout the document? Are the key terms included in the project's glossary?

Traceability

- ☐ Are all requirements needed? Is each requirement necessary to meet the parent requirement? Is each requirement a needed function or characteristic? Distinguish between needs and wants. If it is not necessary, it is not a requirement. Ask, "What is the worst that could happen if the requirement was not included?"
- ☐ Are all requirements (functions, structures, and constraints) bidirectionally traceable to higher-level requirements or mission or system-of-interest scope (i.e., need(s), goals, objectives, constraints, or concept of operations)?
- ☐ Is each requirement stated in such a manner that it can be uniquely referenced (e.g., each requirement is uniquely numbered) in subordinate documents?

Correctness

- ☐ Is each requirement correct?
- ☐ Is each stated assumption correct? Assumptions should be confirmed before the document can be baselined.
- ☐ Are the requirements technically feasible?

Functionality

- ☐ Are all described functions necessary and together sufficient to meet mission and system goals and objectives?

Performance

- ☐ Are all required performance specifications and margins listed (e.g., consider timing, throughput, storage size, latency, accuracy and precision)?
- ☐ Is each performance requirement realistic?
- ☐ Are the tolerances overly tight? Are the tolerances defendable and cost-effective? Ask, "What is the worst thing that could happen if the tolerance was doubled or tripled?"

Interfaces

- ☐ Are all external interfaces clearly defined?
- ☐ Are all internal interfaces clearly defined?
- ☐ Are all interfaces necessary, sufficient, and consistent with each other?

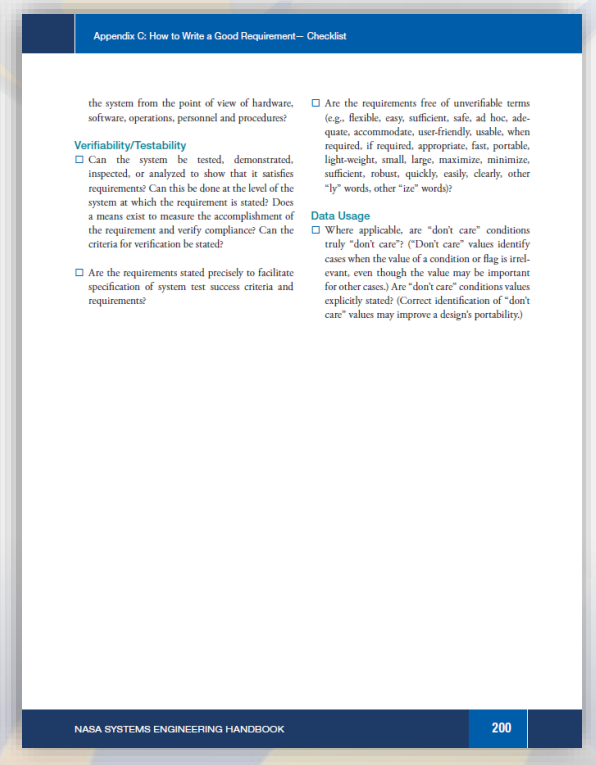
Maintainability

- ☐ Have the requirements for maintainability of the system been specified in a measurable, verifiable manner?
- ☐ Are requirements written so that ripple effects from changes are minimized (i.e., requirements are as weakly coupled as possible)?

Reliability

- ☐ Are clearly defined, measurable, and verifiable reliability requirements specified?
- ☐ Are there error detection, reporting, handling, and recovery requirements?
- ☐ Are undesired events (e.g., single-event upset, data loss or scrambling, operator error) considered and their required responses specified?
- ☐ Have assumptions about the intended sequence of functions been stated? Are these sequences required?
- ☐ Do these requirements adequately address the survivability after a software or hardware fault of

NASA SYSTEMS ENGINEERING HANDBOOK 199



Appendix C: How to Write a Good Requirement— Checklist

the system from the point of view of hardware, software, operations, personnel and procedures?

Verifiability/Testability

- ☐ Can the system be tested, demonstrated, inspected, or analyzed to show that it satisfies requirements? Can this be done at the level of the system at which the requirement is stated? Does a means exist to measure the accomplishment of the requirement and verify compliance? Can the criteria for verification be stated?
- ☐ Are the requirements stated precisely to facilitate specification of system test success criteria and requirements?
- ☐ Are the requirements free of unverifiable terms (e.g., flexible, easy, sufficient, safe, ad hoc, adequate, accommodate, user-friendly, usable, when required, if required, appropriate, fast, portable, light-weight, small, large, maximize, minimize, sufficient, robust, quickly, easily, clearly, other "ly" words, other "ize" words)?

Data Usage

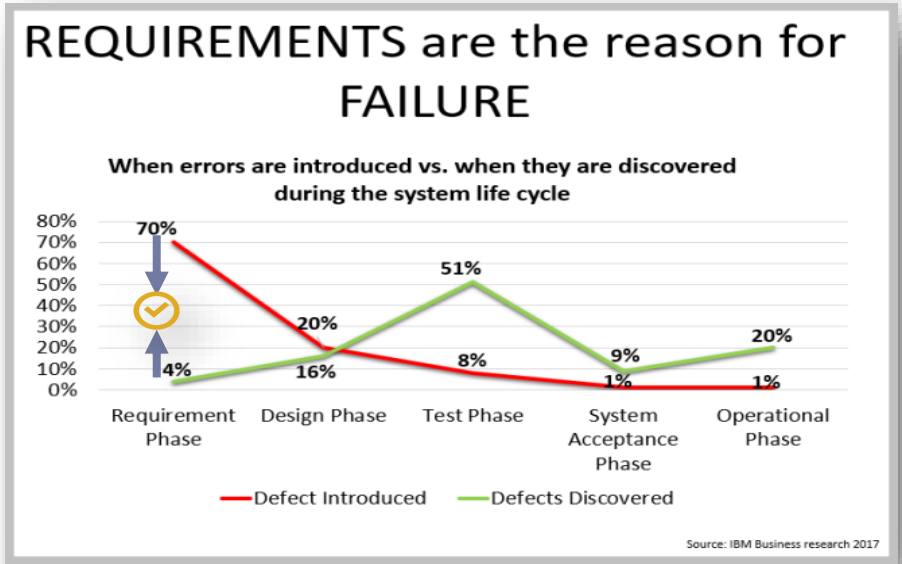
- ☐ Where applicable, are "don't care" conditions truly "don't care"? ("Don't care" values identify cases when the value of a condition or flag is irrelevant, even though the value may be important for other cases.) Are "don't care" conditions values explicitly stated? (Correct identification of "don't care" values may improve a design's portability.)

NASA SYSTEMS ENGINEERING HANDBOOK 200

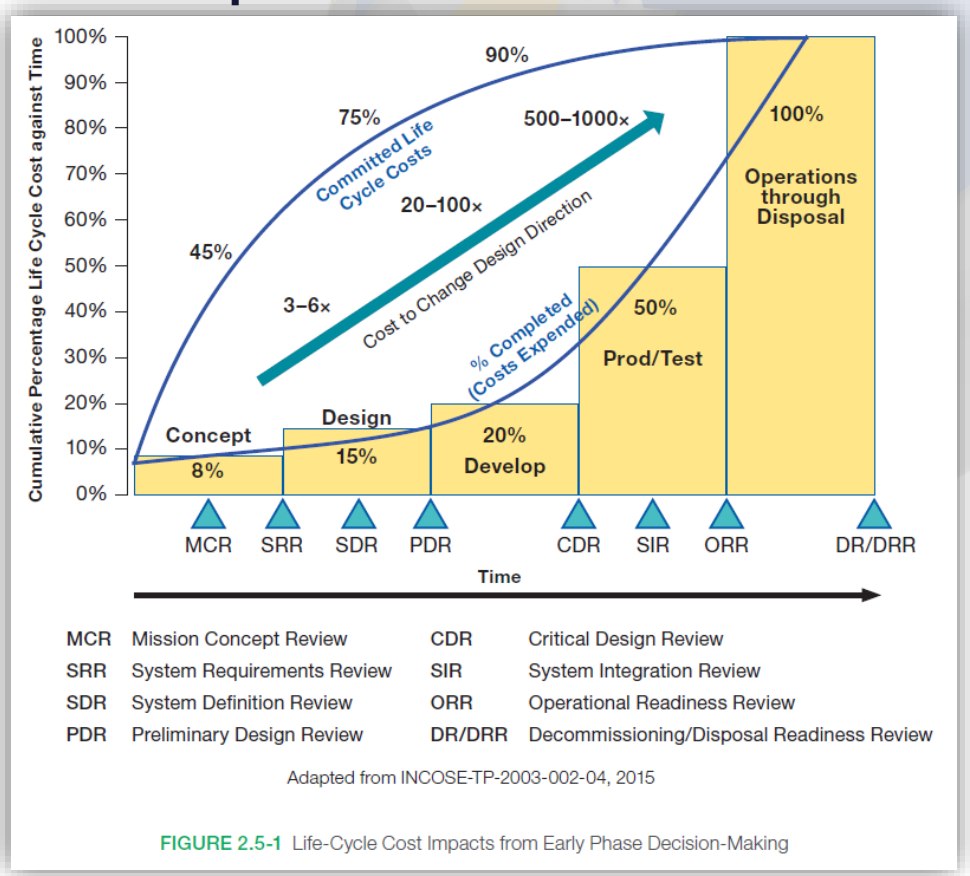
Cost Effectiveness Considerations

NASA Handbook points:

*"The **objective of systems engineering** is to see that the system is designed, built, and can be operated so that it accomplishes its purpose safely in **the most cost-effective** way possible considering performance, cost, schedule, and risk".*



Costs expended vs. cost committed



Source: NASA Systems Engineering Handbook SP-2016-6105 Rev2

According to J. Stecklein (NASA JSC.) *"The cost of fixing a requirement error discovered during the Operations phase **ranged from 29 to more than 1500 times** the cost for addressing that error at the Requirements phase"*

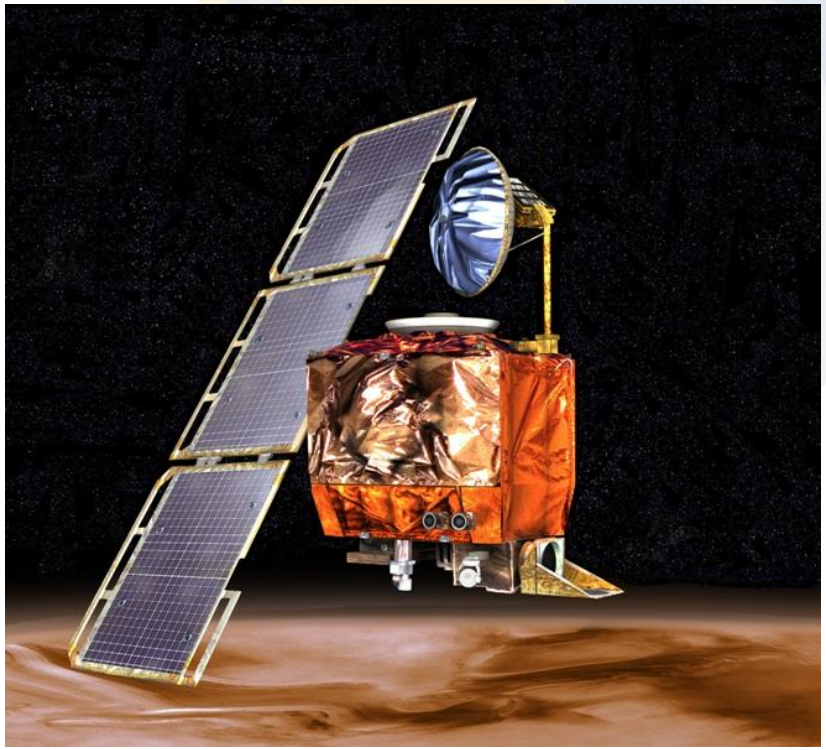
The consistency problem in systems engineering: NASA 1999

A disaster investigation board reports that NASA's **Mars Climate Orbiter** burned up in the Martian atmosphere because **engineers failed to convert units from English to metric**.

The $125 million satellite was supposed to be the first weather observer on another world.

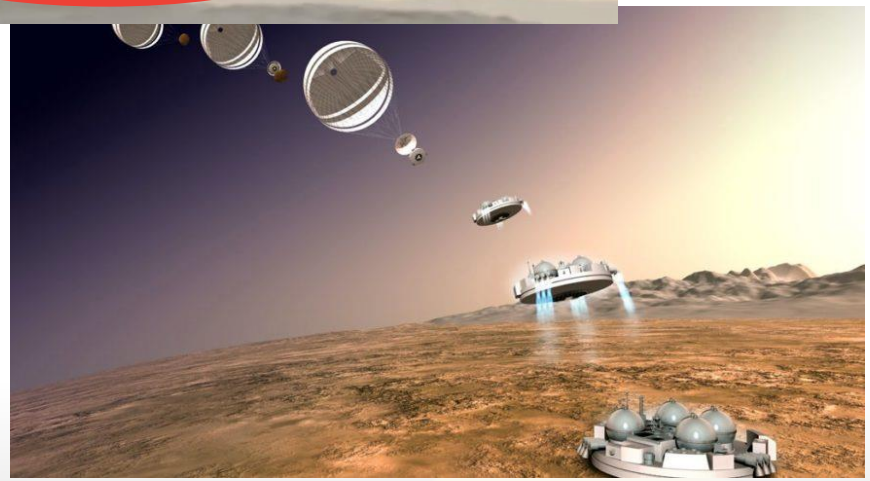
A NASA review board found that the problem was in the software controlling the orbiter's thrusters. The **software** calculated the force the thrusters needed to exert in ***pounds*** of force. A separate piece of software took in the data assuming it was in the metric unit: ***newtons***.

"*People make errors,*" Gavin said. "*The problem here was not the error. It was **the failure of us to look at it end-to-end and find it.** It's unfair to rely on any one person.*"



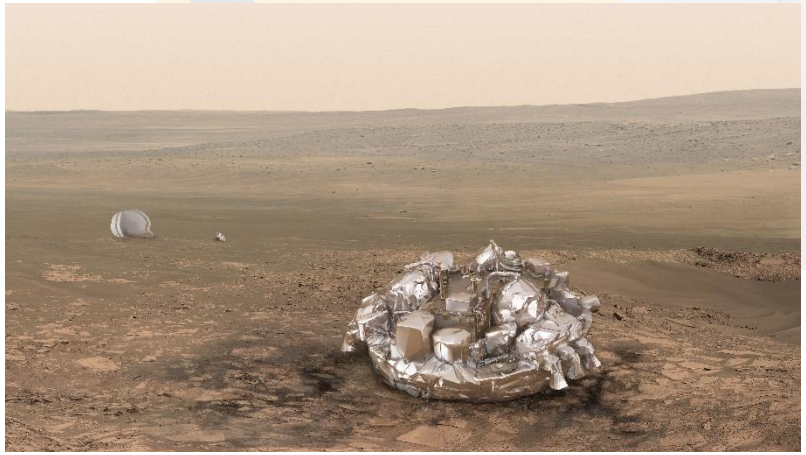
https://www.wired.com/2010/11/1110mars-climate-observer-report/
http://edition.cnn.com/TECH/space/9909/30/mars.metric.02/

The consistency problem in systems engineering: Schiaparelli lander



http://spacenews.com/esa-mars-lander-crash-caused-by-1-second-inertial-measurement-error/

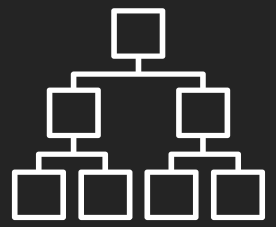
Vs.



400.000.000 € loss

Schiaparelli lander's crash landing on Mars on Oct. 19 2016 - ESA

http://spaceflight101.com/exomars/exomars-tgo-enters-orbit-lander-falls-silent/



What is an

Ontology and a knowledge library

What is an Ontology and a knowledge library

- A **combination of Knowledge items**,
  - of different nature,
  - at different levels of abstraction
- Representing a **specific business domain** or **area of knowledge**
- With the aim of **improving the way projects are managed**, including:
  - the promotion of the principle: **quality** *right the first time,*
  - enabling semantic search portals to **archive and retrieve assets**,
  - thus providing tools to **reuse assets** at different level,
  - and **reducing time** to market,
  - **improving** the way engineers generate (**author**) new assets,
  - **enhancing** the way items are **inspected and verified,**
  - **enabling** real **interoperability** mechanisms and services,
  - **reducing time** to elaborate documents, systems and projects

What is an Ontology and a knowledge library

01 Vocabulary/Glossary

Controlled Organizational and Project Vocabulary for a common understanding among stakeholders

02 SCM/Architectures

Capture the system architectures represented in views and models. Stablish relationships among system and system elements, and among other system entities. Classifying information by meaning, nature…

03 Patterns

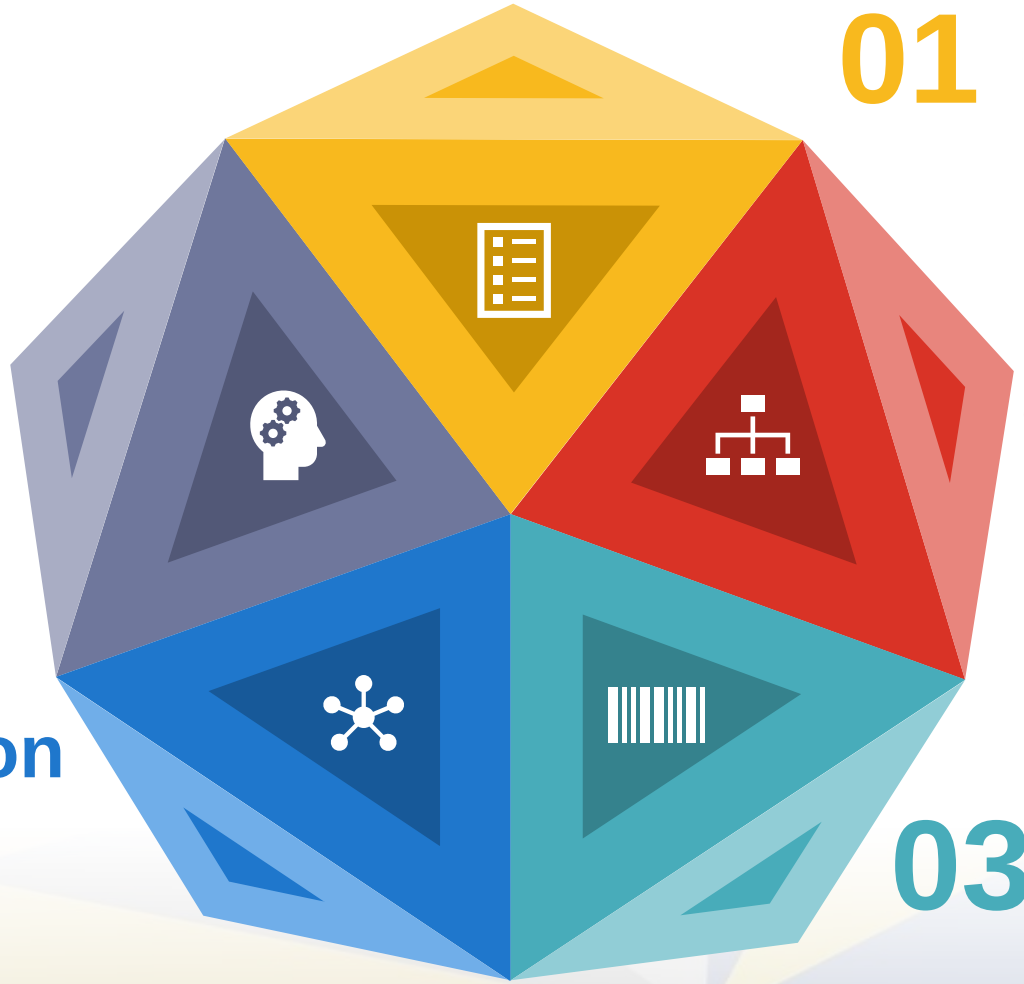
Representing a set of agreed-upon templates (grammars) to create and maintain consistent textual artifacts

04 Formalization

Representation of assets semantic through SRL – System Representation Language

05 Reasoning

A combination of rules, and actions to infer information from valuable assets and to control the behavioural part of the knowledge library



Example of Knowledge Library

Domain specific

Common English

Vocabulary

Aircraft | A380 | A350

System | Operate | Temperature | Environment | Pressure

shall | The | of | at | Lower

Architectures - Conceptual model

Temperature — "Operation Range" → [-60ºC , +60ºC]

System

Aircraft

Fuselage | Wing | Landing gear | A380 | A350

<Operation>

Work

Operate

Environment

Temperature | Pressure

"Greater than (>)"

Patterns

System (*) | Shall | Operation (*) | At | «Minimum» | Environment (*) | Of | NUMBER | MEASUREMENT UNIT

Formalization

The A380 shall be able to operate at a minimum temperature of -70º C → Temperature — "Greater than (>)" → -70 | ºC

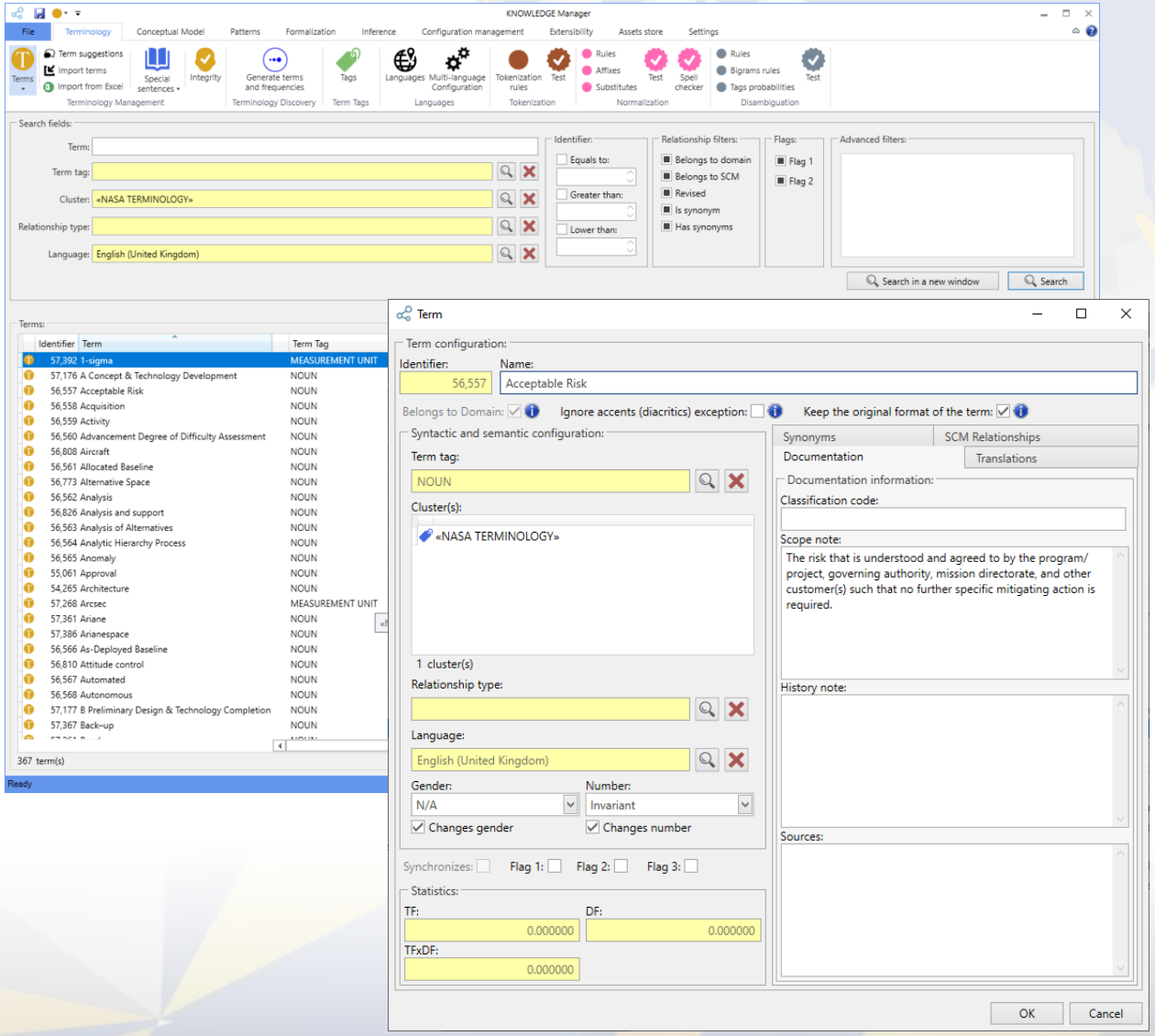
Reasoning

If NUMBER Lower than (<) -60º ºC Or NUMBER Greater than (>) +60º ºC → ✗

1. Vocabulary: NASA Handbook, Glossary, etc

- Terms from the Handbook glossary have been included in the library

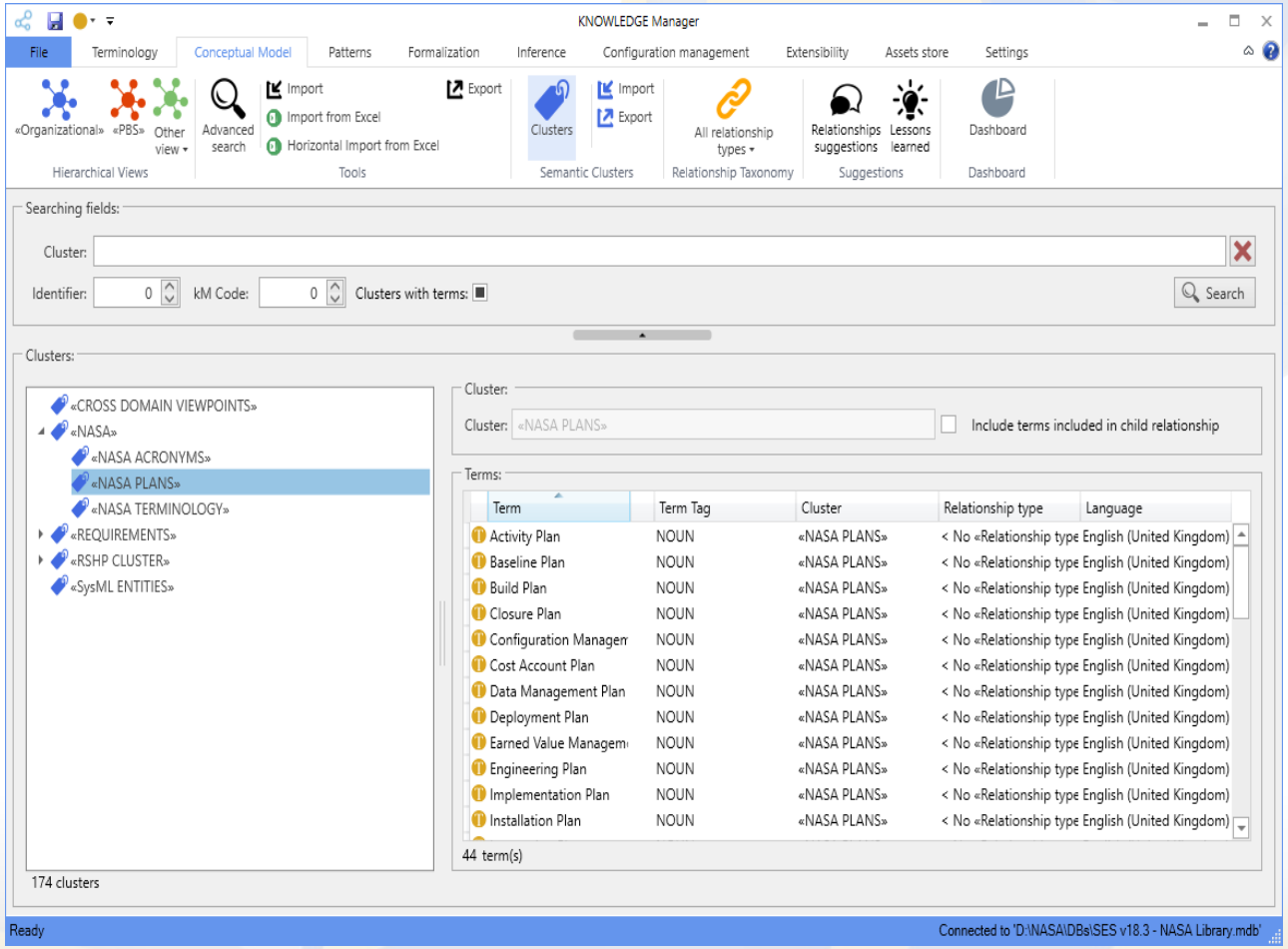
- **Provides a consistent way to name and understand all the concepts across the industry**
- **The system can highlight and link references to these entries in the body of the documents**



2. Conceptual Models: NASA Handbook, Clustering the terms of the glossary

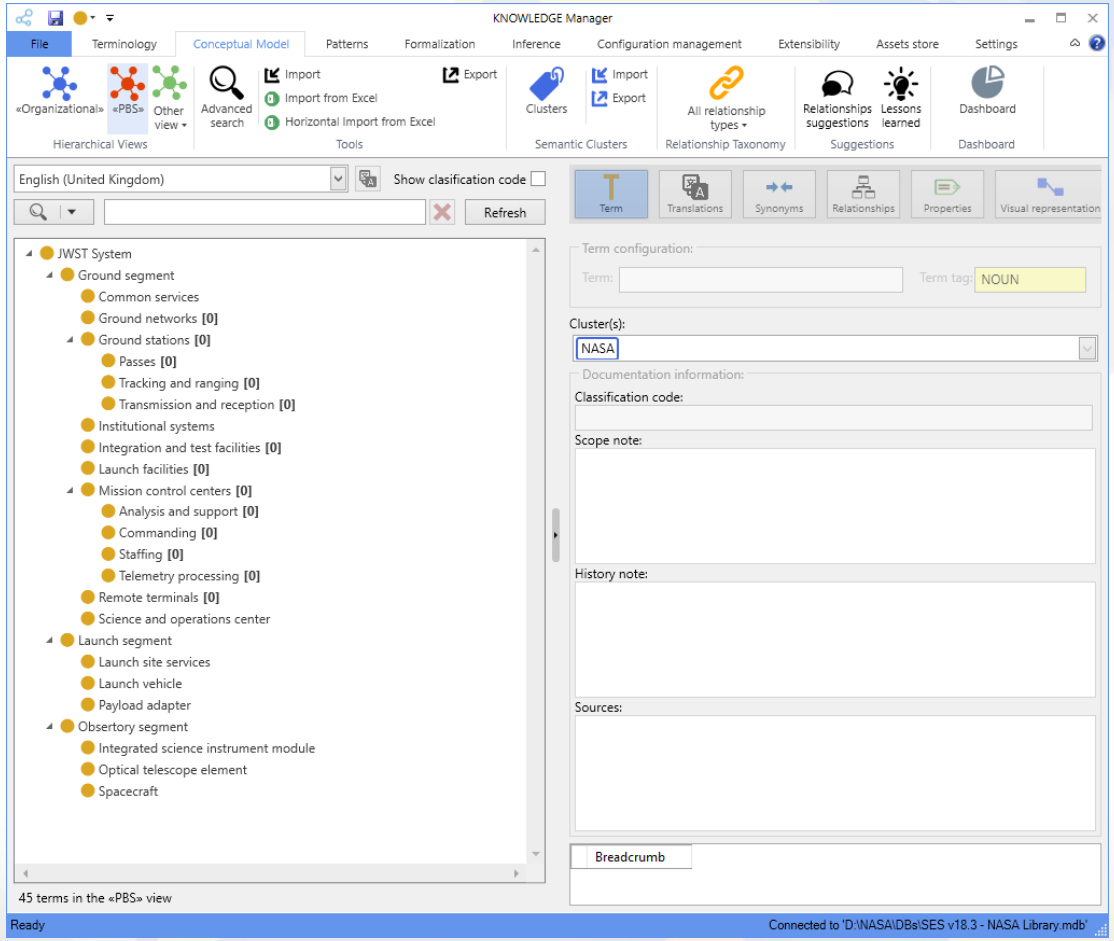
- **Clustering:** according to the semantic of the terms in the library

- **Provides means to fit the textual paterns and help authors while the write requirements or other types of textual assets**



2. Conceptual Models: NASA Handbook, Taxonomy of terms and other relationships

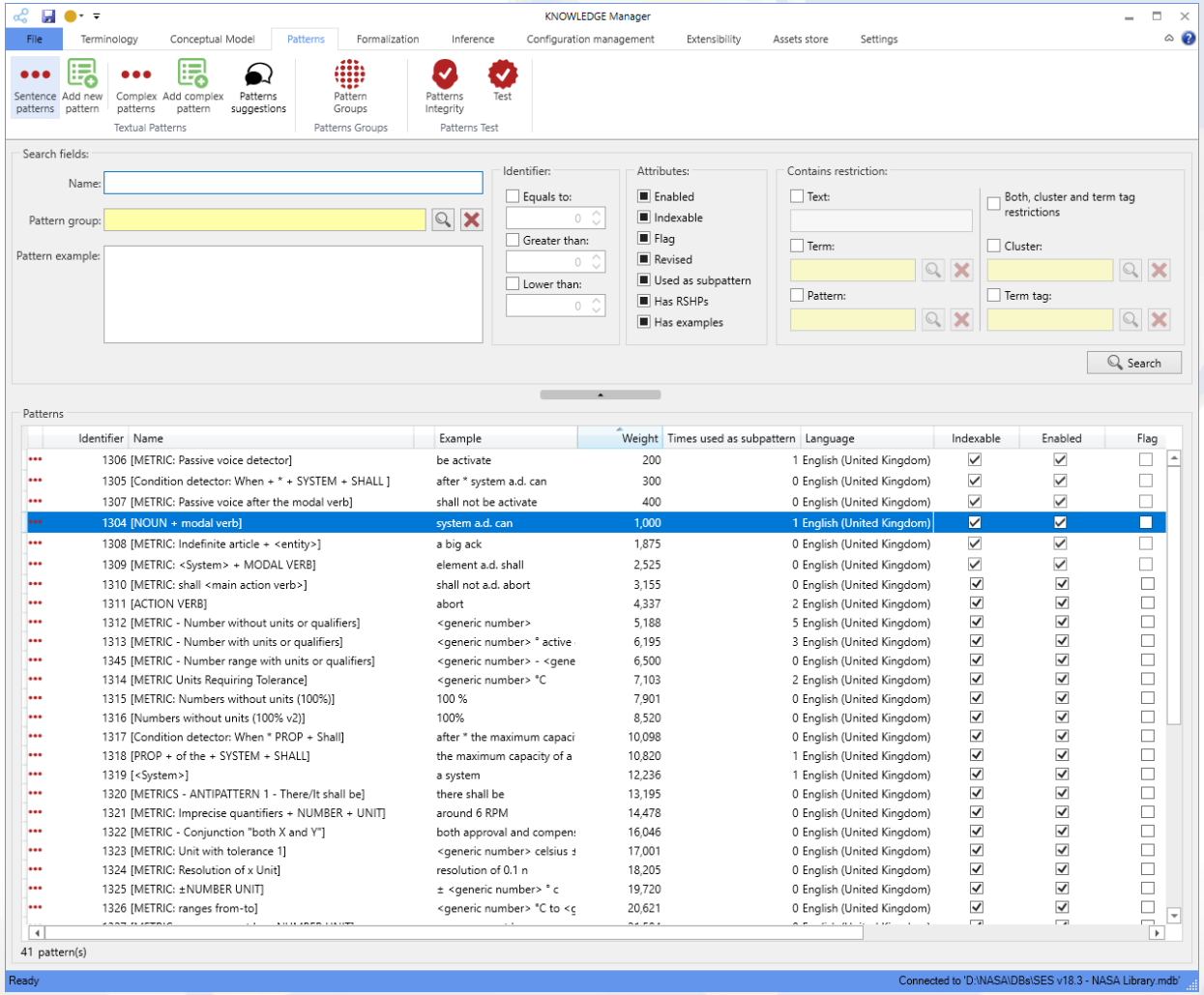
- Based on some relationships showed in the Handbook
- Including PBS views
- **Provides means to propagate queries in further reuse stages or just for information retrieval**



1. Patterns: NASA Handbook,

- Patters for enabling the Handbook mapping metrics have been included in the library

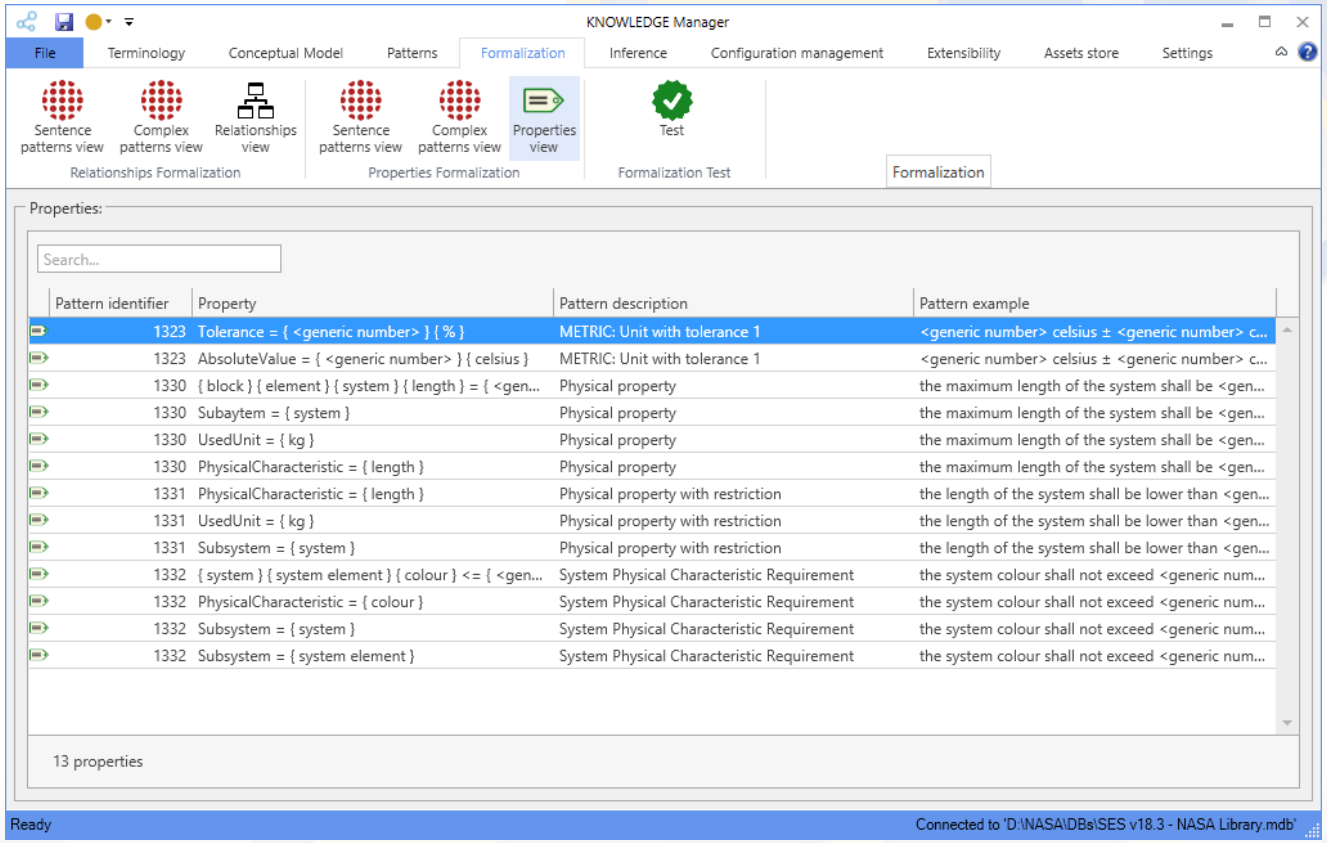
- **Represent requirements similarities and enable formal representation, automatic recognition and aid authors**



1. Formalization: NASA Handbook

- Formalization extracts relations and properties for enabling the metrics that have been included in the library

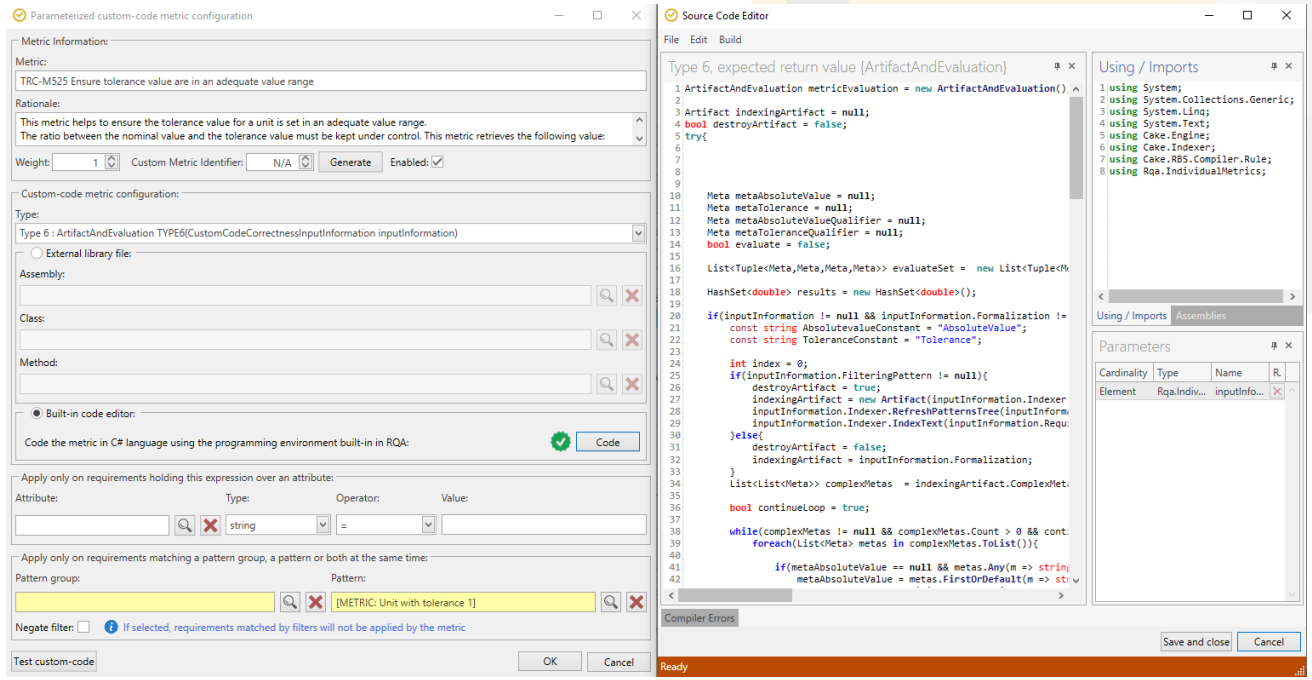
- **Representation of assets semantic through SRL – System Representation Language**



1. Reasoning: NASA Handbook, Glossary, etc

- Reasoning for enabling the advanced semantic features the Handbook mapping metrics included in the library

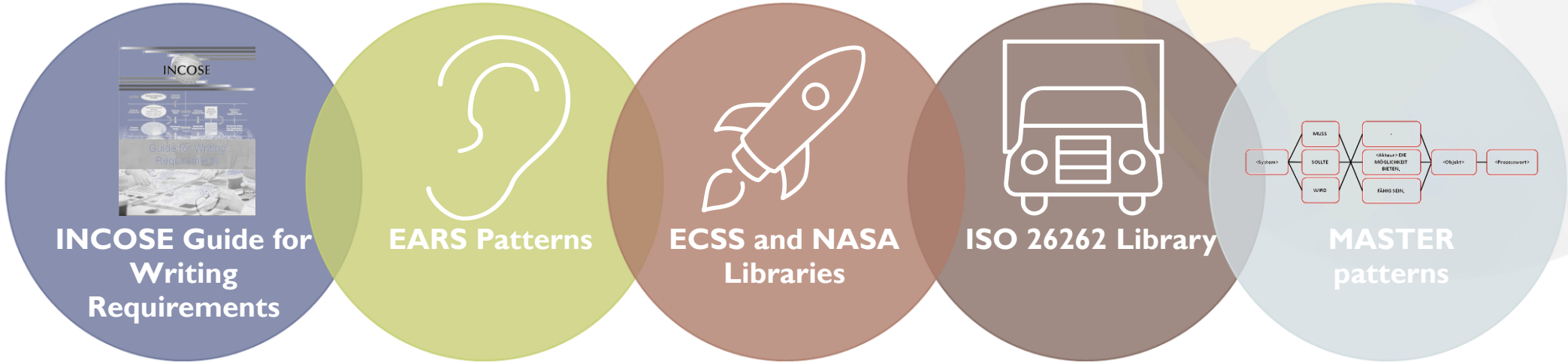
- **A combination of rules, tasks and groups to infer information from valuable assets**



Knowledge Libraries

ECSS and NASA

Glossary, patterns and rules



INCOSE

Quality rules for the analysis of textual requirements

EARS

Requirements patterns

ISO 26262

Glossary, patterns and rules

MASTER

Quality rules for requirements and requirements patterns

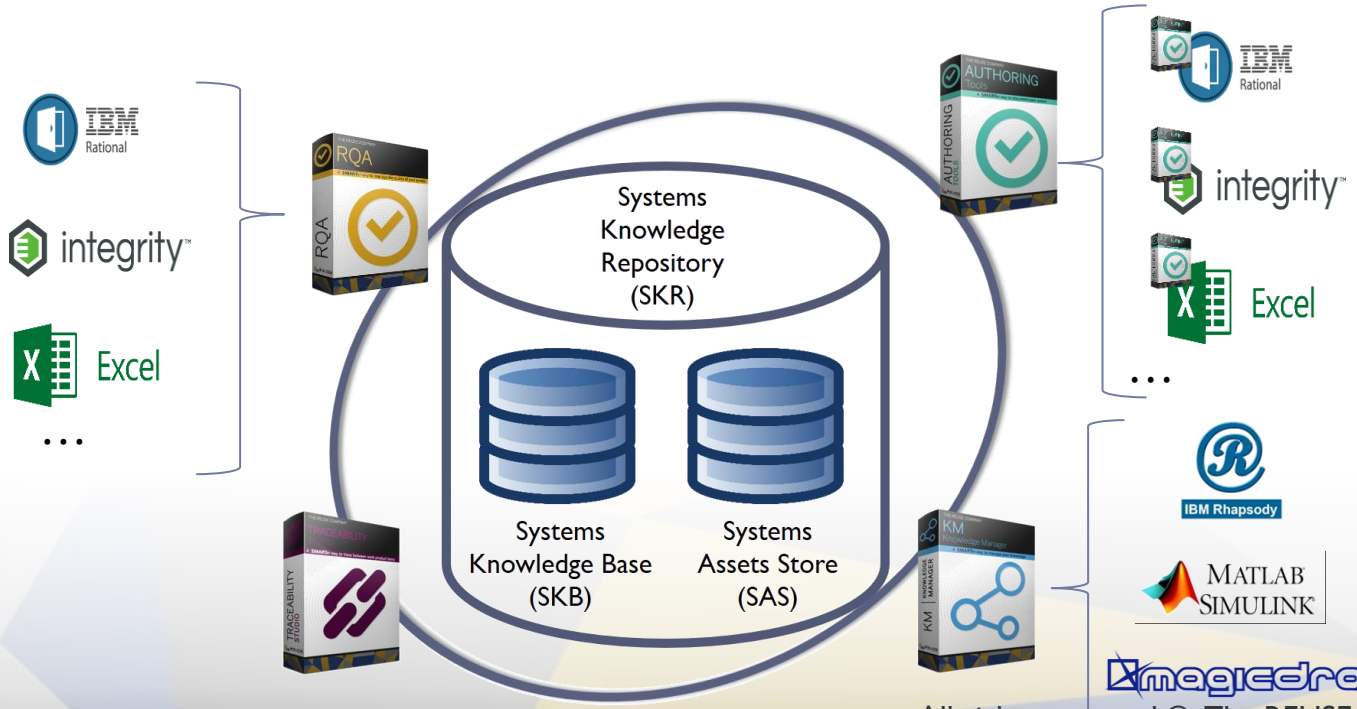
Knowledge Base

Users


The Systems
Engineering Suite

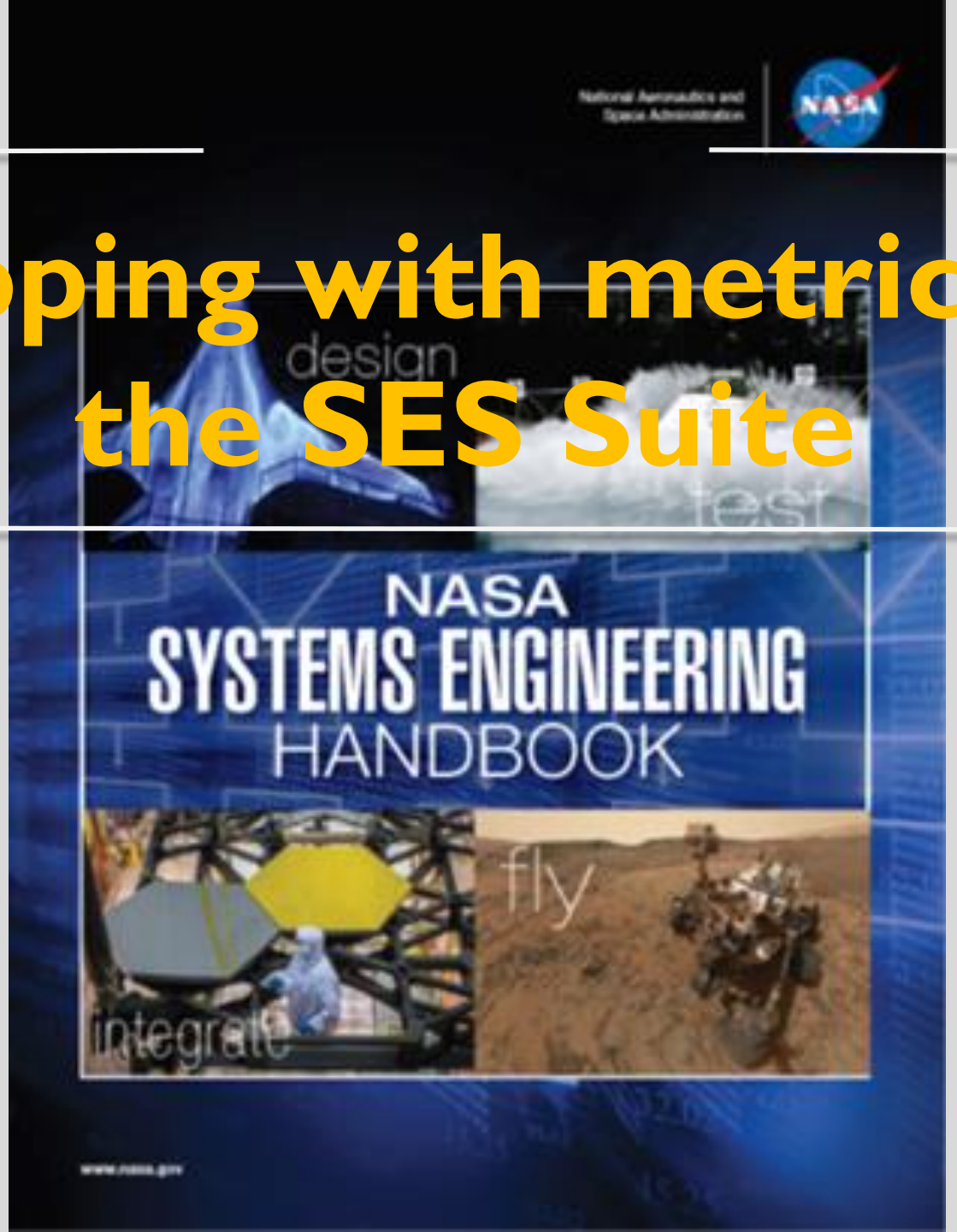
The TRC Systems Engineering Suite

- The Systems Engineering Suite intends to tackle **requirements quality management** by offering a set of tools and processes
- **Automatic measurement of requirements quality metric**
- **Support to Requirements Authoring**
- SES Suite models requirements quality metrics using the CCC approach (**Correctness, Consistency and Completeness**)



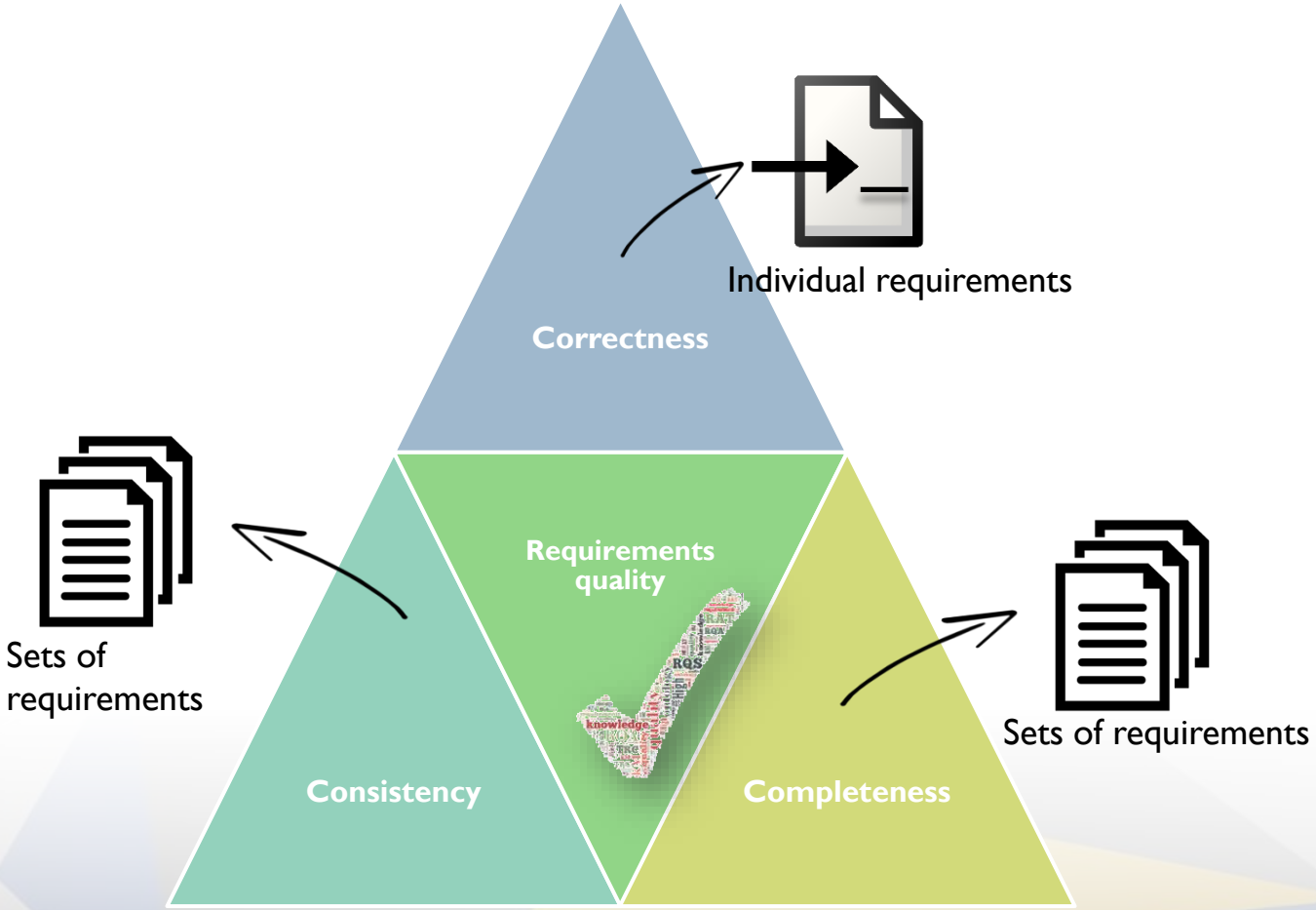
- **Quality Studio (RQA):** to setup, check and manage the quality of a requirements specification
- **Rich Authoring Tool (RAT):** to assist authors while they are creating or editing requirements
- **Knowledge Manager (KM):** to manage knowledge around a requirements specification: dictionaries, glossaries, concept maps, knowledge models, ontologies, patterns…

Mapping with metrics in the SES Suite



Requirements quality metrics: CCC Approach

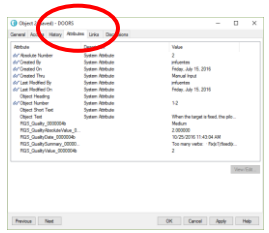
- CCC – Correctness, Consistency and Completeness



Requirement: The agreed-upon need, desire, want, capability, capacity, or demand for personnel, equipment, facilities, or other resources or services by specified quantities for specific periods of time or at a specified time expressed as **a "shall" statement**. Acceptable form for a requirement statement is individually **clear, correct, feasible to obtain, unambiguous in meaning, and can be validated at the level of the system structure at which it is stated**. In pairs of requirement statements or as a set, collectively, they are not redundant, are adequately related with respect to terms used, and are not in conflict with one another.

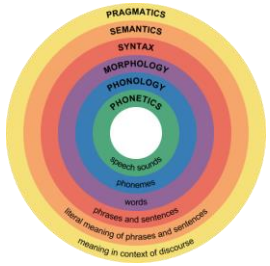
Source: NASA Systems Engineering Handbook SP-2016-6105 Rev2

Examples of requirements metrics: **Correctness**

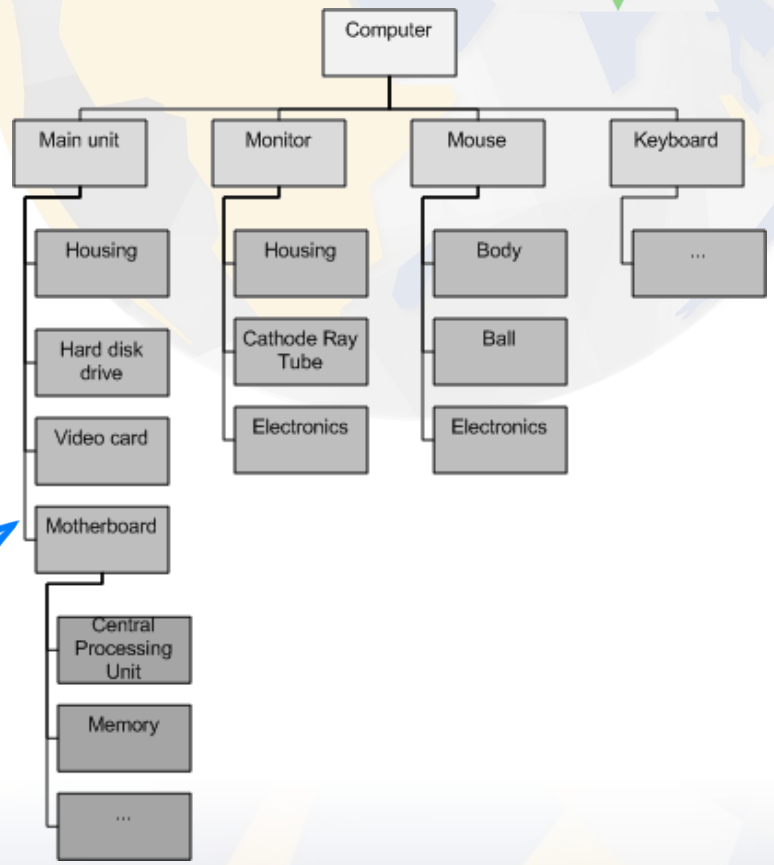


- Metrics based on information coming **from the RMS**:
  - Attributes, links, versions…
- Metrics based **on lists of terms**:
  - Forbidden: ambiguous, pronouns…
  - Restricted: negations…
  - Mandatory: 'shall', 'will', 'should'…

- Metrics based **on linguistic algorithms**:
  - Text length, misspelling, readability….
  - Detection of passive voice, imperative tense…



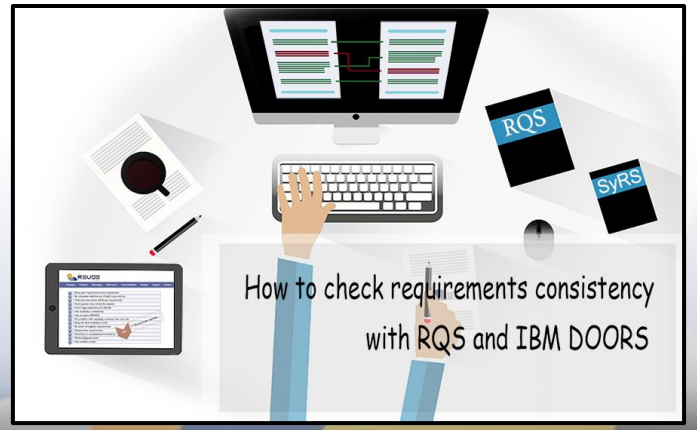
- Metrics based on the **conformance with models**:
  - Concepts in your requirements coming from PBS, FBS…



- Metrics **based on patterns**:
  - Compliance with different types of requirements patterns
  - Detection of specific structures within the requirements

Examples of requirements metrics: **Consistency**

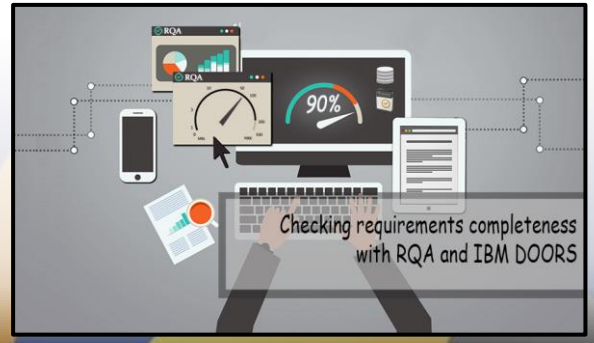
- Are your requirements **consistent with each other**?
- Are your requirements consistent **with the models** of your projects?
- Do you have **duplicated** requirements in your specifications?
- Are the values for the mentioned signals within the **expected ranges**?
- Are you using the proper **measurement units** in your requirements?
- Are all the properties **property allocated** along the system decomposition?
- Are your requirements describing **wrong transitions** in a state chart?



Examples of requirements metrics: **Completeness**

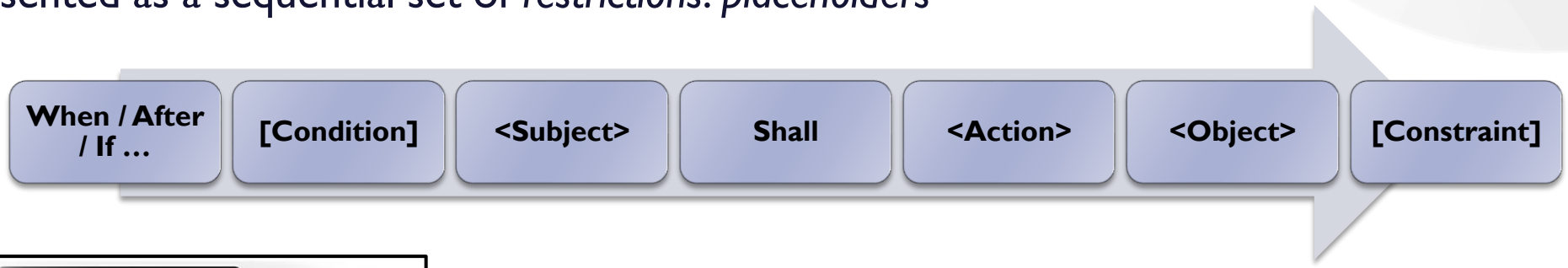
- Completeness at specification/project level:
  - Are all the expected **requirements types** involved in your specifications?
  - Are all the **key concepts** (from the ontology or from other models, e.g. blocks, states, signals, properties…) properly covered?
    - Does the whole set of requirements documents include requirements **for all the elements of the system** according to a block diagram (architecture)?
    - Does the spec. include requirements mentioning **all the signals**?
    - Does the spec. include requirements describing the behavior of the system elements in **any of their possible states and modes**?
  - Are your requirements properly **linked**? At the different levels?
  - Are all the properties stated **for every system element**?
    - For those properties in a model whose **value is to be provided** in the spec, is the value actually provided?
- Completeness at requirement level:
  - Does every requirement include **all the agreed parts** (condition, subject…):

    following patterns
  - Are you stating the values for the mentioned **properties with tolerances**: 12V±0.5V

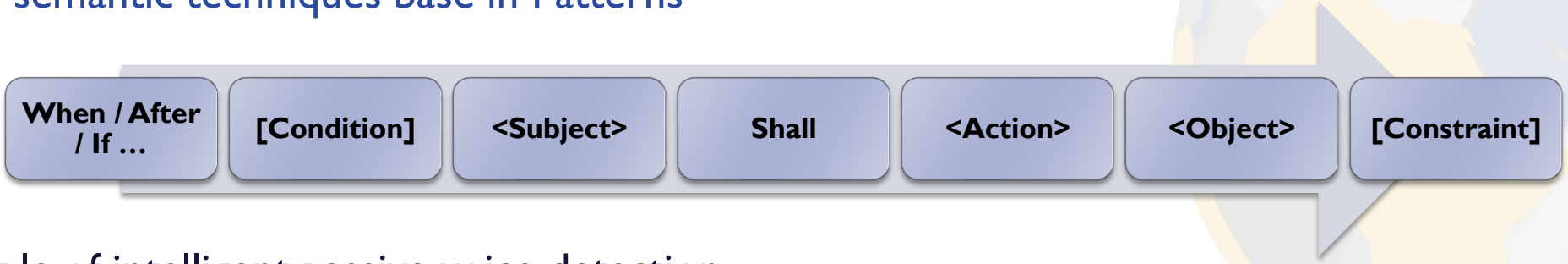


Patterns

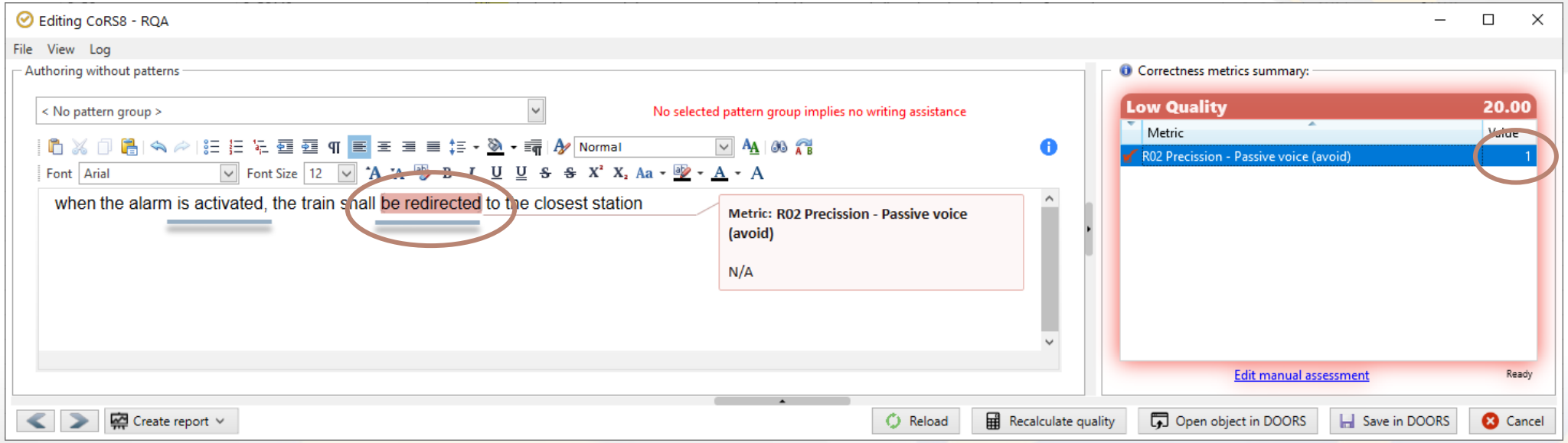
- Represents the structures every *correct* requirement should meet
- Different types of requirements → different patterns (templates)
- Customizable for every domain, customer and content of each requirements document
- Libraries with sets of patterns
- Represented as a sequential set of *restrictions*: *placeholders*



Advanced semantic techniques base in Patterns



- Example of intelligent passive voice detection



Advanced semantic techniques

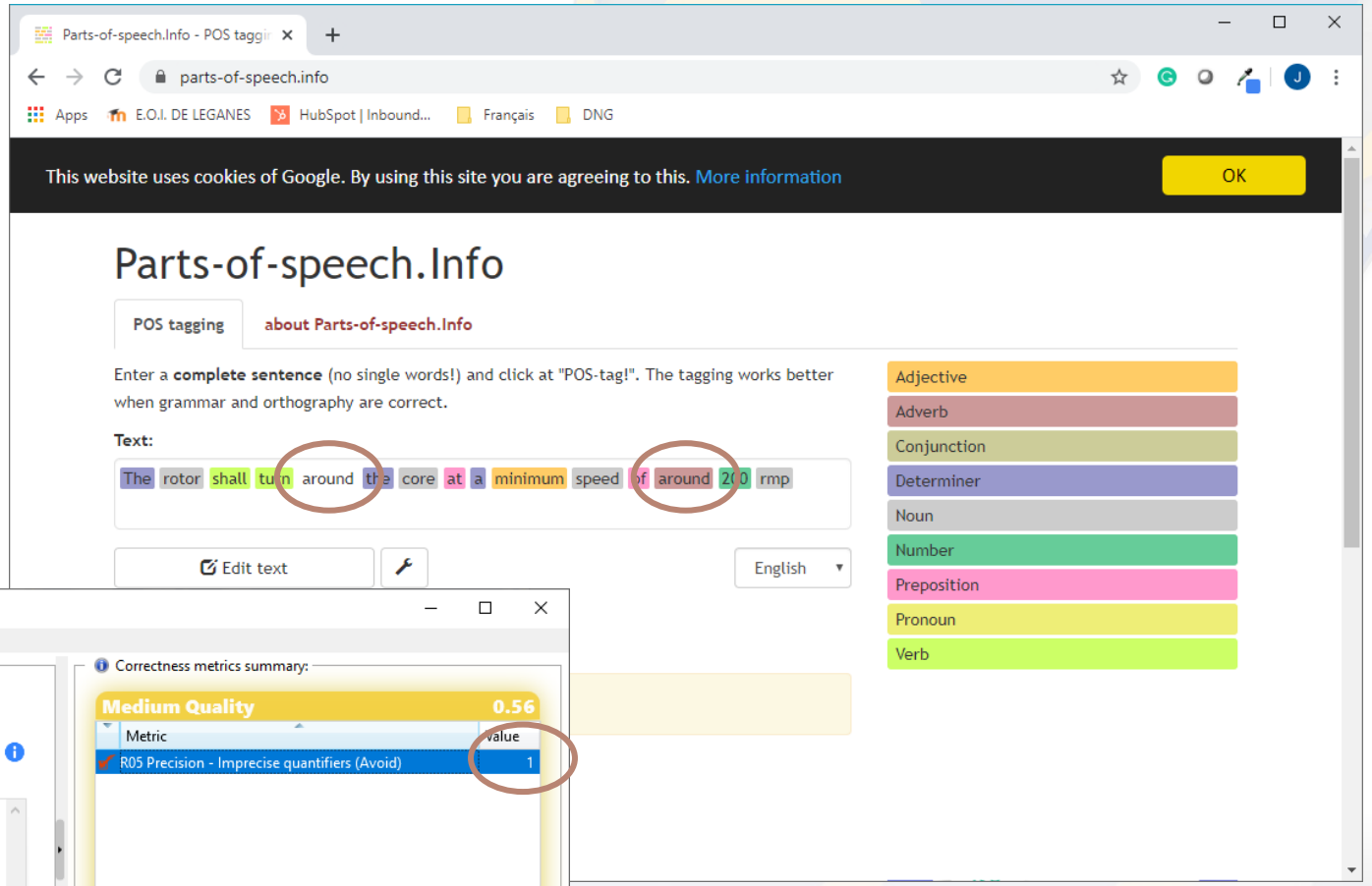
System Requirements Spec.

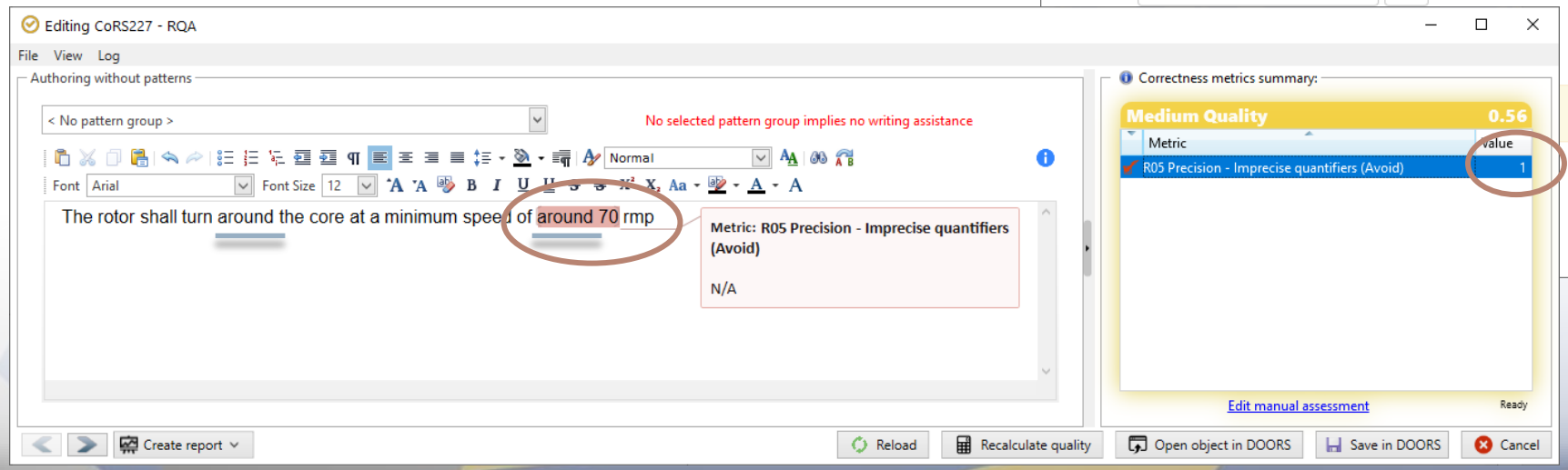
…

SyR-088: xxx

SyR-089: The rotor shall turn around the core at a minimum speed of around 70 rmp

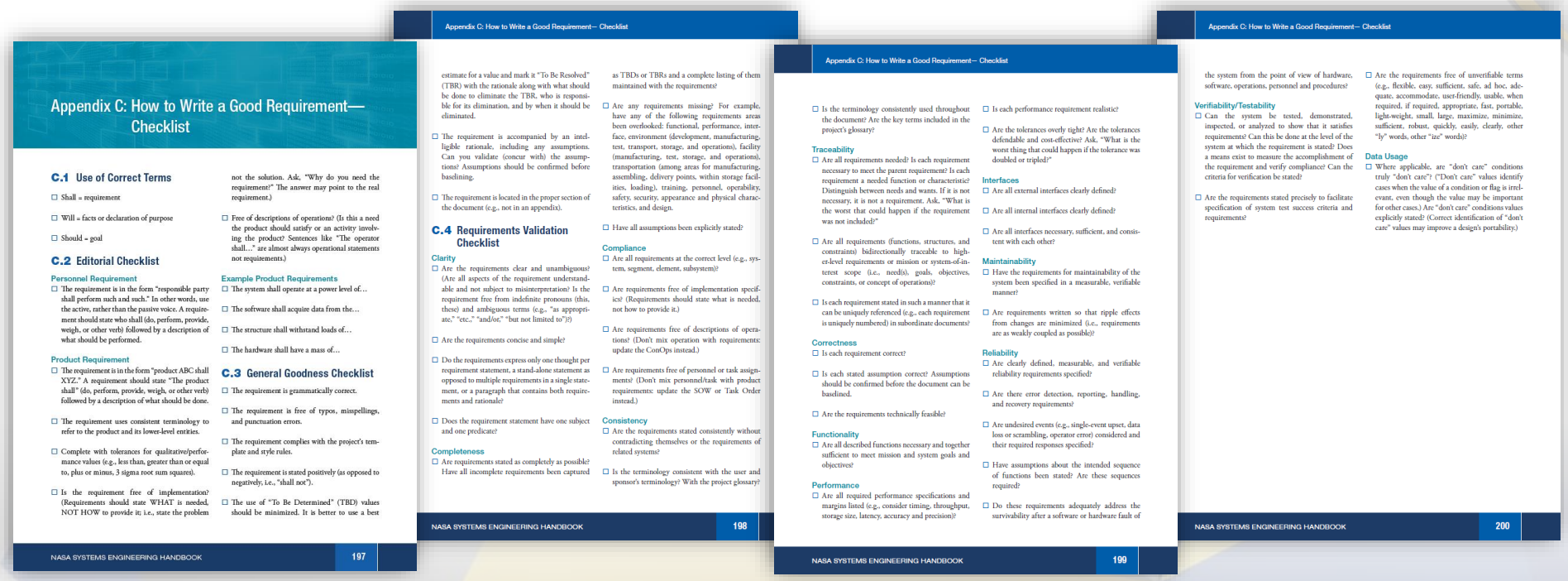
SyR-090: xxx





Requirements at the NASA design process

- **Handbook section 4.2.1.2.3 Define Requirements in Acceptable Statements**
- *"the requirements should be defined in acceptable "shall" statements, which are complete sentences with a single "shall" per statement. Rationale for the requirement should also be captured to ensure the reason and context of the requirement is understood."*
- **Appendix C: How to Write a Good Requirement— Checklist**



Appendix C: How to Write a Good Requirement— Checklist

C.1 Use of Correct Terms

- ☐ Shall = requirement
- ☐ Will = facts or declaration of purpose
- ☐ Should = goal

C.2 Editorial Checklist

Personnel Requirement

- ☐ The requirement is in the form "responsible party shall perform such and such." In other words, use the active, rather than the passive voice. A requirement should state who shall (do, perform, provide, weigh, or other verb) followed by a description of what should be performed.

Product Requirement

- ☐ The requirement is in the form "product ABC shall XYZ." A requirement should state "The product shall" (do, perform, provide, weigh, or other verb) followed by a description of what should be done.
- ☐ The requirement uses consistent terminology to refer to the product and its lower-level entities.
- ☐ Complete with tolerances for qualitative/performance values (e.g., less than, greater than or equal to, plus or minus, 3 sigma root sum squares).
- ☐ Is the requirement free of implementation? (Requirements should state WHAT is needed, NOT HOW to provide it; i.e., state the problem not the solution. Ask, "Why do you need the requirement?" The answer may point to the real requirement.)
- ☐ Free of descriptions of operations? (Is this a need the product should satisfy or an activity involving the product? Sentences like "The operator shall…" are almost always operational statements not requirements.)

Example Product Requirements

- ☐ The system shall operate at a power level of…
- ☐ The software shall acquire data from the…
- ☐ The structure shall withstand loads of…
- ☐ The hardware shall have a mass of…

C.3 General Goodness Checklist

- ☐ The requirement is grammatically correct.
- ☐ The requirement is free of typos, misspellings, and punctuation errors.
- ☐ The requirement complies with the project's template and style rules.
- ☐ The requirement is stated positively (as opposed to negatively, i.e., "shall not").
- ☐ The use of "To Be Determined" (TBD) values should be minimized. It is better to use a best estimate for a value and mark it "To Be Resolved" (TBR) with the rationale along with what should be done to eliminate the TBR, who is responsible for its elimination, and by when it should be eliminated.
- ☐ The requirement is accompanied by an intelligible rationale, including any assumptions. Can you validate (concur with) the assumptions? Assumptions should be confirmed before baselining.
- ☐ The requirement is located in the proper section of the document (e.g., not in an appendix).

C.4 Requirements Validation Checklist

Clarity

- ☐ Are the requirements clear and unambiguous? (Are all aspects of the requirement understandable and not subject to misinterpretation? Is the requirement free from indefinite pronouns (this, these) and ambiguous terms (e.g., "as appropriate," "etc.," "and/or," "but not limited to")?)
- ☐ Are the requirements concise and simple?
- ☐ Do the requirements express only one thought per requirement statement, a stand-alone statement as opposed to multiple requirements in a single statement, or a paragraph that contains both requirements and rationale?
- ☐ Does the requirement statement have one subject and one predicate?

Completeness

- ☐ Are requirements stated as completely as possible? Have all incomplete requirements been captured as TBDs or TBRs and a complete listing of them maintained with the requirements?
- ☐ Are any requirements missing? For example, have any of the following requirements areas been overlooked: functional, performance, interface, environment (development, manufacturing, test, transport, storage, and operations), facility (manufacturing, test, storage, and operations), transportation (among areas for manufacturing, assembling, delivery points, within storage facilities, loading), training, personnel, operability, safety, security, appearance and physical characteristics, and design.
- ☐ Have all assumptions been explicitly stated?

Compliance

- ☐ Are all requirements at the correct level (e.g., system, segment, element, subsystem)?
- ☐ Are requirements free of implementation specifics? (Requirements should state what is needed, not how to provide it.)
- ☐ Are requirements free of descriptions of operations? (Don't mix operation with requirements: update the ConOps instead.)
- ☐ Are requirements free of personnel or task assignments? (Don't mix personnel/task with product requirements: update the SOW or Task Order instead.)

Consistency

- ☐ Are the requirements stated consistently without contradicting themselves or the requirements of related systems?
- ☐ Is the terminology consistent with the user and sponsor's terminology? With the project glossary?
- ☐ Is the terminology consistently used throughout the document? Are the key terms included in the project's glossary?

Traceability

- ☐ Are all requirements needed? Is each requirement necessary to meet the parent requirement? Is each requirement a needed function or characteristic? Distinguish between needs and wants. If it is not necessary, it is not a requirement. Ask, "What is the worst that could happen if the requirement was not included?"
- ☐ Are all requirements (functions, structures, and constraints) bidirectionally traceable to higher-level requirements or mission or system-of-interest scope (i.e., need(s), goals, objectives, constraints, or concept of operations)?
- ☐ Is each requirement stated in such a manner that it can be uniquely referenced (e.g., each requirement is uniquely numbered) in subordinate documents?

Correctness

- ☐ Is each requirement correct?
- ☐ Is each stated assumption correct? Assumptions should be confirmed before the document can be baselined.
- ☐ Are the requirements technically feasible?

Functionality

- ☐ Are all described functions necessary and together sufficient to meet mission and system goals and objectives?

Performance

- ☐ Are all required performance specifications and margins listed (e.g., consider timing, throughput, storage size, latency, accuracy and precision)?
- ☐ Is each performance requirement realistic?
- ☐ Are the tolerances overly tight? Are the tolerances defendable and cost-effective? Ask, "What is the worst thing that could happen if the tolerance was doubled or tripled?"

Interfaces

- ☐ Are all external interfaces clearly defined?
- ☐ Are all internal interfaces clearly defined?
- ☐ Are all interfaces necessary, sufficient, and consistent with each other?

Maintainability

- ☐ Have the requirements for maintainability of the system been specified in a measurable, verifiable manner?
- ☐ Are requirements written so that ripple effects from changes are minimized (i.e., requirements are as weakly coupled as possible)?

Reliability

- ☐ Are clearly defined, measurable, and verifiable reliability requirements specified?
- ☐ Are there error detection, reporting, handling, and recovery requirements?
- ☐ Are undesired events (e.g., single-event upset, data loss or scrambling, operator error) considered and their required responses specified?
- ☐ Have assumptions about the intended sequence of functions been stated? Are these sequences required?
- ☐ Do these requirements adequately address the survivability after a software or hardware fault of the system from the point of view of hardware, software, operations, personnel and procedures?

Verifiability/Testability

- ☐ Can the system be tested, demonstrated, inspected, or analyzed to show that it satisfies requirements? Can this be done at the level of the system at which the requirement is stated? Does a means exist to measure the accomplishment of the requirement and verify compliance? Can the criteria for verification be stated?
- ☐ Are the requirements stated precisely to facilitate specification of system test success criteria and requirements?
- ☐ Are the requirements free of unverifiable terms (e.g., flexible, easy, sufficient, safe, ad hoc, adequate, accommodate, user-friendly, usable, when required, if required, appropriate, fast, portable, light-weight, small, large, maximize, minimize, sufficient, robust, quickly, easily, clearly, other "ly" words, other "ize" words)?

Data Usage

- ☐ Where applicable, are "don't care" conditions truly "don't care"? ("Don't care" values identify cases when the value of a condition or flag is irrelevant, even though the value may be important for other cases.) Are "don't care" conditions values explicitly stated? (Correct identification of "don't care" values may improve a design's portability.)

NASA SYSTEMS ENGINEERING HANDBOOK 197 | 198 | 199 | 200

C.1 Use of Correct Terms C.2 Editorial Checklist	
	Personnel Requirement
	Product Requirement
C.3 General Goodness Checklist C.4 Requirements Validation Checklist	
	Clarity
	Completeness
	Compliance
	Consistency
	Traceability
	Correctness
	Functionality
	Performance
	Interfaces
	Maintainability
	Reliability
	Verifiability/Testability
	Data Usage

Metrics for Quality Analysis & Authoring of Requirements

C.1 Use of Correct Terms		Tackle	TRC Metric	Metric Name
	Shall = requirement Will = facts or declaration of purpose Should = goal	Yes	M365	Avoid the use of Banned Modal Verbs
C.2 Editorial Checklist				
Personnel Requirement		**Tackle**	**TRC Metric**	**Metric Name**
	Use the active, rather than the passive voice. A requirement should state who shall (do, perform, provide, weigh, or other verb) followed by a description of what should be performed.	Yes	M040	Avoid the use of Passive Voice out of the condition part
Product Requirement		**Tackle**	**TRC Metric**	**Metric Name**
	The requirement is in the form "product ABC shall XYZ." A requirement should state "The product shall" (do, perform, provide, weigh, or other verb) followed by a description of what should be done.	Yes	M010	Enforce the use of a complete structure sentence
			M360	Check the number of Modal Verbs
	The requirement uses consistent terminology to refer to the product and its lower-level entities.	Yes	M220	Avoid Out of the Dictionary Nouns
			M630	Enforce the use of Define Terms by avoiding Synonyms
			M150	Detect inadequate Unit for a Characteristic
	Complete with tolerances for qualitative/performance values (e.g., less than, greater than or equal to, plus or minus, 3 sigma root sum squares).	Yes	M520	Force to include tolerance value for the units that required tolerance
			M525	Ensure tolerance value are in an adequate value range
	Is the requirement free of implementation? (Requirements should state WHAT is needed, NOT HOW to provide it; i.e., state the problem not the solution. Ask, "Why do you need the requirement?" The answer may point to the real requirement.)	Yes	M490	Avoid stating a solution
	Free of descriptions of operations? (Is this a need the product should satisfy or an activity involving the product? Sentences like "The operator shall..." are almost always operational statements not requirements.)	Partial	M500	Avoid the use of Flow sentences

C.3 General Goodness Checklist		Tackle	TRC Metric	Metric Name
	1. The requirement is grammatically correct.	Partial	M230	Avoid inadequate grammar structures
	2. The requirement is free of typos, misspellings, and punctuation errors.	Yes	M240	Avoid Incorrect spelling
			M260	Review incorrect punctuation
			M250	Facilitate readability
	3. The requirement complies with the project's template and style rules.	Yes	M010	Enforce the use of a complete structure sentence
	4. The requirement is stated positively (as opposed to negatively, i.e., "shall not").	Yes	M285	Avoid the use of Negative Expressions out of the condition part
	5. The use of "To Be Determined" (TBD) values should be minimized. It is better to use a best estimate for a value and mark it "To Be Resolved" (TBR) with the rationale along with what must be done to eliminate the TBR, who is responsible for its elimination, and by when it must be eliminated.	Partial	M900	Look for TBD expressions
	6. The requirement is accompanied by an intelligible rationale, including any assumptions. Can you validate (concur with) the assumptions? Assumptions must be confirmed before baselining.	Partial	M460	Enforce attribute type is not empty
	7. The requirement is located in the proper section of the document (e.g., not in an appendix).	No		

Mapping with metrics in the SES Suite

C.4 Requirements Validation Checklist				
Clarity		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are the requirements clear and unambiguous? (Are all aspects of the requirement understandable and not subject to misinterpretation? Is the requirement free from indefinite pronouns (this, these) and ambiguous terms (e.g., "as appropriate," "etc.," "and/or," "but not limited to")?)	Yes	M130	Avoid the use of Indefinite Articles before Entity
			M070	Avoid the use of Pronouns to refer to nouns
			M950	Avoid the use of Vague Terms
			M545	Avoid the usage of Imprecise Quantifiers apply to a property
			M560	Avoid the use of Temporal Indefinite keywords out of the condition part
	2. Are the requirements concise and simple?	Yes	M330	Check the text length by counting words
			M360	Check the number of Modal Verbs
			M320	Check the text length by counting paragraphs
			M340	Control the number of Action Verbs out of the condition part
	3. Do the requirements express only one thought per requirement statement, a standalone statement as opposed to multiple requirements in a single statement, or a paragraph that contains both requirements and rationale?	Yes	M340	Control the number of Action Verbs out of the condition part
			M360	Check the number of Modal Verbs
	4. Does the requirement statement have one subject and one predicate?	Yes	M370	Multiple subject detection
			M340	Control the number of Action Verbs out of the condition part

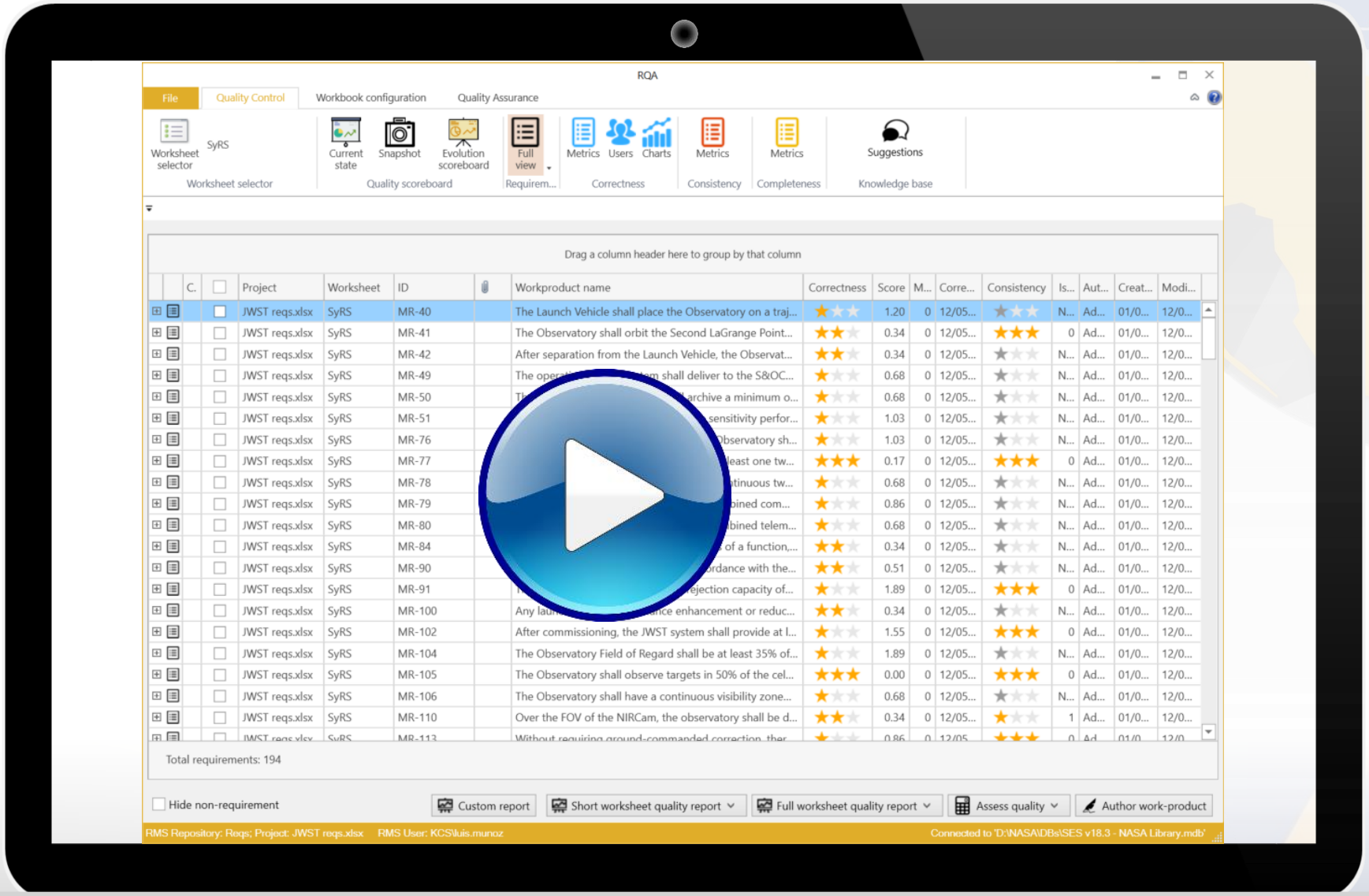
Completeness		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are requirements stated as completely as possible? Have all incomplete requirements been captured as TBDs or TBRs and a complete listing of them maintained with the requirements?	Partial	M900	Look for TBD expressions
	2. Are any requirements missing? For example have any of the following requirements areas been overlooked: functional, performance, interface, environment (development, manufacturing, test, transport, storage, operations), facility (manufacturing, test, storage, operations), transportation (among areas for manufacturing, assembling, delivery points, within storage facilities, loading), training, personnel, operability, safety, security, appearance and physical characteristics, and design.	Partial	M940	SCM organization completeness
	3. Have all assumptions been explicitly stated?	No		
Compliance		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are all requirements at the correct level (e.g., system, segment, element, subsystem)?	Partial	M055	Detect inappropriate subject to the document level
	2. Are requirements free of implementation specifics? (Requirements should state what is needed, not how to provide it.)		M490	Avoid stating a solution
	3. Are requirements free of descriptions of operations? (Don't mix operation with requirements: update the ConOps instead.)		M380	Avoid phrases that indicate the purpose
Consistency				
	1. Are the requirements stated consistently without contradicting themselves or the requirements of related systems?	Partial	M480	Avoid overlapping between the requirements
			M160	Avoid the use of different Unit systems for the same Characteristic
	2. Is the terminology consistent with the user and sponsor's terminology? With the project glossary?	Yes	M220	Avoid Out of the Dictionary Nouns
	3. Is the terminology consistently used through out the document?	Yes	M580	Avoid the use of unknown acronyms
	4. Are the key terms included in the project's glossary?	Yes	M590	Avoid the use of unknown abbreviations
Traceability		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are all requirements needed? Is each requirement necessary to meet the parent requirement? Is each requirement a needed function or characteristic? Distinguish between needs and wants. If it is not necessary, it is not a requirement. Ask, "What is the worst that could happen if the requirement was not included?"	No		
	2. Are all requirements (functions, structures, and constraints) bidirectionally traceable to higher level requirements or mission or system-of-interest scope (i.e., need(s), goals, objectives, constraints, or concept of operations)?	Partial	M910	R80 Traceability - TRC - Out-links (Enforce)
		Partial	M920	R80 Traceability - TRC - In-links (Enforce)
	3. Is each requirement stated in such a manner that it can be uniquely referenced (e.g., each requirement is uniquely numbered) in subordinate documents?	Yes	M930	Ensure requirement unique reference

Mapping with metrics in the SES Suite

Correctness		Tackle	TRC Metric	Metric Name
	1. Is each requirement correct?	Partial	Metric set	
	2. Is each stated assumption correct? Assumptions must be confirmed before the document can be baselined.			
	3. Are the requirements technically feasible?			
Functionality		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are all described functions necessary and together sufficient to meet mission and system goals and objectives?	No		
Performance		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are all required performance specifications and margins listed (e.g., consider timing, throughput, storage size, latency, accuracy and precision)?	Partial	M545	Avoid the usage of Imprecise Quantifiers apply to a property
			M530	Confirms the value for a property is in a controlled range
	2. Is each performance requirement realistic?		M430	Avoid unachievable Absolutes expressions impossible to verify
			M140	Ensure Numbers are followed by Units or noun qualifications
	3. Are the tolerances overly tight? Are the tolerances defendable and cost-effective? Ask, "What is the worst thing that could happen if the tolerance was doubled or tripled?"		M525	Ensure tolerance value are in an adequate value range
Interfaces		**Tackle**	**TRC Metric**	**Metric Name**
	1. Are all external interfaces clearly defined?	Partial	M945	SCM PBS completeness
	2. Are all internal interfaces clearly defined?		M945	SCM PBS completeness
	3. Are all interfaces necessary, sufficient, and consistent with each other?		M945	SCM PBS completeness
Maintainability		**Tackle**	**TRC Metric**	**Metric Name**
	1. Have the requirements for system maintainability been specified in a measurable, verifiable manner?	Partial	M140	Ensure Numbers are followed by Units or noun qualifications
			M540	Avoid the usage of Imprecise Quantifiers
			M430	Avoid unachievable Absolutes expressions impossible to verify
			M940	SCM organization completeness
	2. Are requirements written so that ripple effects from changes are minimized (i.e., requirements are as weakly coupled as possible)?		M200	Avoid the use of Open-Ended clauses

Reliability		Tackle	TRC Metric	Metric Name
	1. Are clearly defined, measurable, and verifiable reliability requirements specified?	Partial	M540	Avoid the usage of Imprecise Quantifiers
			M430	Avoid unachievable Absolutes expressions impossible to verify
	2. Are there error detection, reporting, handling, and recovery requirements?		M940	SCM organization completeness
	3. Are undesired events (e.g., single event upset, data loss or scrambling, operator error) considered and their required responses specified?			
	4. Have assumptions about the intended sequence of functions been stated? Are these sequences required?			
	5. Do these requirements adequately address the survivability after a software or hardware fault of the system from the point of view of hardware, software, operations, personnel and procedures?			
Verifiability/ Testability		**Tackle**	**TRC Metric**	**Metric Name**
	1. Can the system be tested, demonstrated, inspected, or analyzed to show that it satisfies requirements? Can this be done at the level of the system at which the requirement is stated? Does a means exist to measure the accomplishment of the requirement and verify compliance? Can the criteria for verification be stated?	Partial	M540	Avoid the usage of Imprecise Quantifiers
			M430	Avoid unachievable Absolutes expressions impossible to verify
	2. Are the requirements stated precisely to facilitate specification of system test success criteria and requirements?		M940	SCM organization completeness
	3. Are the requirements free of unverifiable terms (e.g., flexible, easy, sufficient, safe, ad hoc, adequate, accommodate, user-friendly, usable, when required, if required, appropriate, fast, portable, light-weight, small, large, maximize, minimize, sufficient, robust, quickly, easily, clearly, other "ly" words, other "ize" words)?		M430	Avoid unachievable Absolutes expressions impossible to verify
			M950	Avoid the use of Vague Terms
Data Usage		**Tackle**	**TRC Metric**	**Metric Name**
	1. Where applicable, are "don't care" conditions truly "don't care"? ("Don't care" values identify cases when the value of a condition or flag is irrelevant, even though the value may be important for other cases.) Are "don't care" conditions values explicitly stated? (Correct identification of "don't care" values may improve a design's portability.)	No		

TRC WEBINARS 2020




Q&A

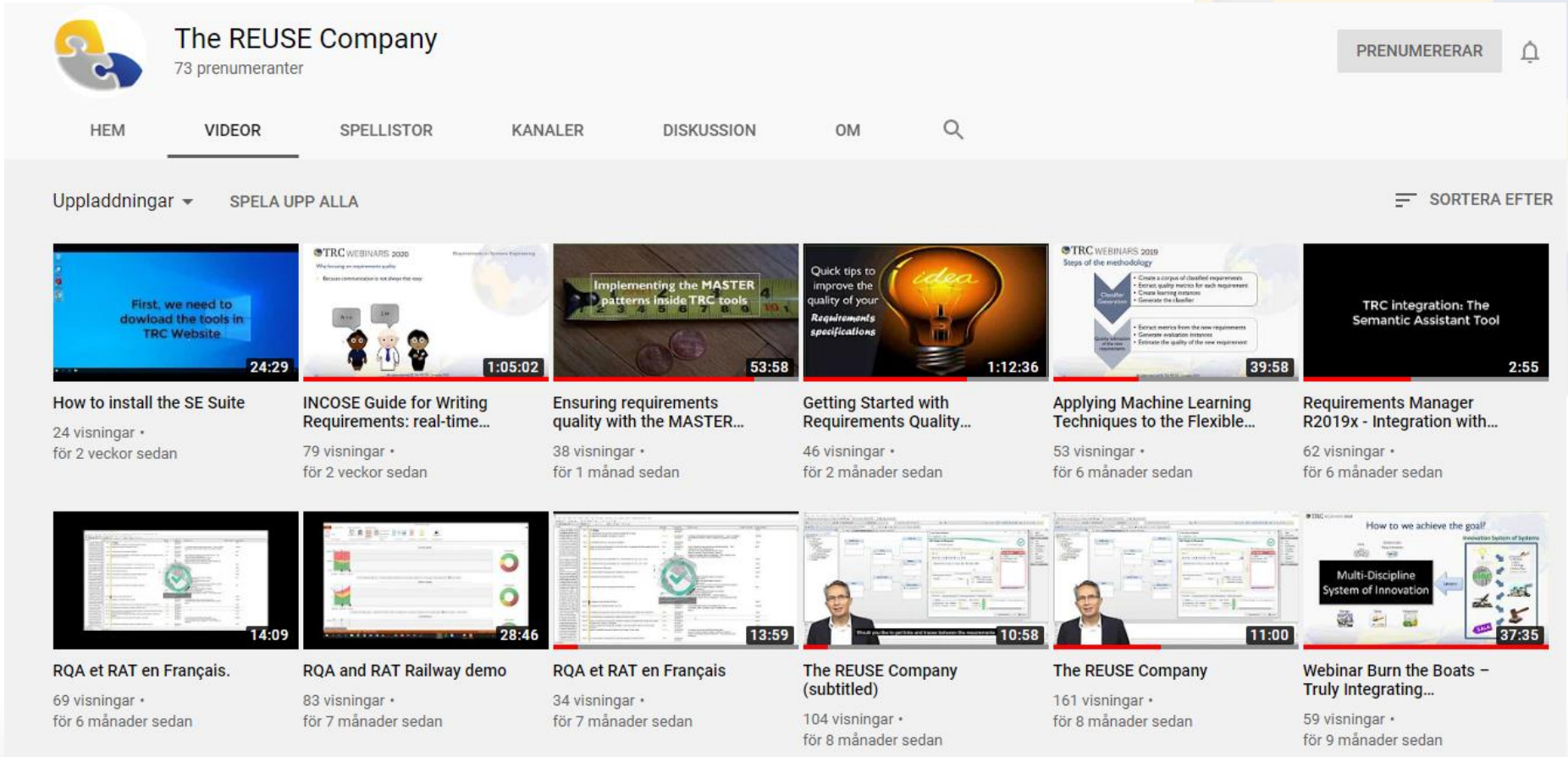
Next webinar

- **A practical way to implement ISO 15288 V&V processes: The V&V Studio**
- The **ISO 15288** clearly defines what must be done when performing **Verification and Validation processes**. We must use and manage verification actions and collect evidences. But how should we do it? How to integrate them all in one environment? How to delegate **V&V** to specialized tools for specific work-products? How to deal with interoperability? This webinar intends to provide insight for these kind of questions.
- Current systems engineering makes clear distinction between verification, validation and quality assurance processes. As part of its knowledge reuse approach, The Reuse Company has created the **V&V Studio** as a software tool ready to provide support to the **ISO15288 V&V processes** by using (and reusing) information from RQA - QUALITY Studio and the **Ontology**. The **V&V Studio** merges the three concepts (**Verification, Validation and Quality**) and offers **V&V** by managing the corresponding verification and validation actions through quality and other measures.
  Dates:
  - Jun 02 and 4


Q&A

TRC WEBINARS 2020

Reuse on YouTube



https://www.youtube.com/user/TheREUSECompany/videos

Contact information

Luis Javier MUÑOZ

luis.munoz@reusecompany.com

+34 912 17 25 96

@ReuseCompany

https://www.linkedin.com/in/lj-munoz/


the
REUSE
company