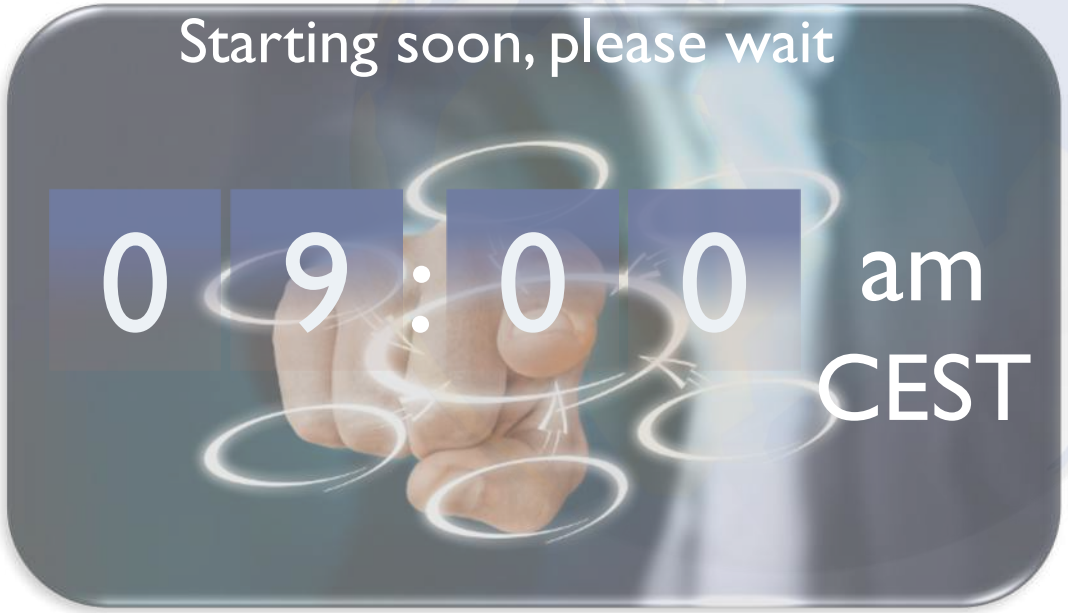


- Webinar rules:
  - You'll be muted all along the Webinar
  - There's a *Question* section to ask your questions whenever you want, you don't need to wait until the end. All questions will be addressed at the end of the webinar
  - If you have any technical issue please use this chatting box, or mail us at: support@reusecompany.com
  - You might receive a survey either after the webinar or by mail. Your opinion is very valuable
  - The Webinar will be recorded. A link to the recording will be sent to you in few days

Raising the ante: high-quality models as the only way forward after high-quality requirements

José M. Fuentes
Chief Sales Manager
The REUSE Company
jose.fuentes@reusecompany.com

Cecilia Karlsson
Marketing & Communication
The REUSE Company
cecilia.karlsson@reusecompany.com

THE REUSE COMPANY

Contents

WHEN

WHO

WHERE

WHY

01 The company was established in **1999**

As a spin-off of a University in Madrid

02 **System + Software Engineers**

Smart combination between Company staff and R&D from Academia

03 **Headquarters:**
Madrid (Spain)

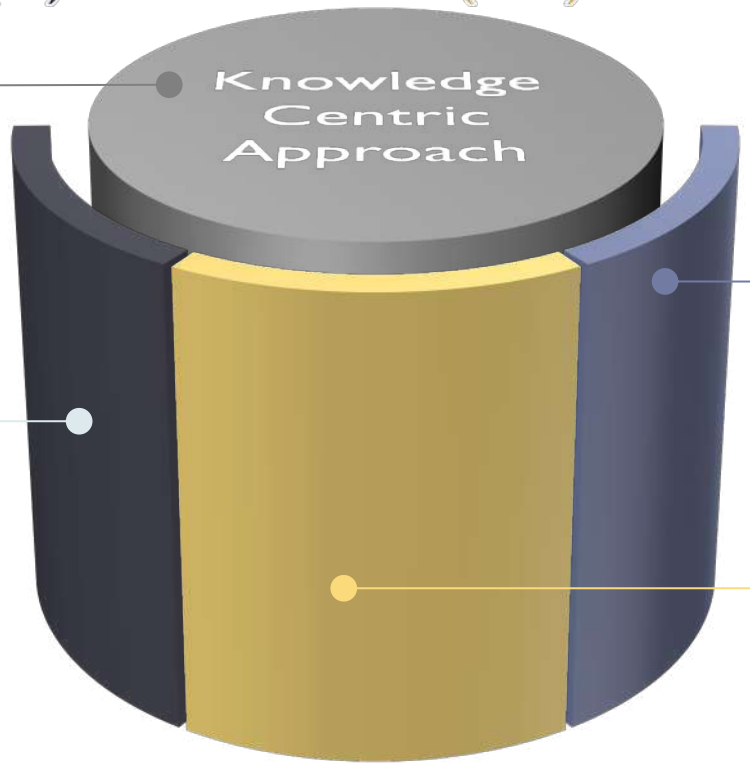
International offices:
Stockholm (Sweden)
Tokyo (Japan) Delegation

2022:
USA
Chicago/Detroit/Miami

04 To promote a **reusable**, **scalable** and global solution to a **smart** and **interoperable** Systems Engineering environment, by offering a **semantic knowledge centric** approach.

THE REUSE COMPANY

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



Knowledge Centric Systems Engineering (KCSE)

Global Repositories –
Archiving –
Configuration Management –

TRACEABILITY

Links and Interoperability –
Transformations –
Change management –

QUALITY (ies)

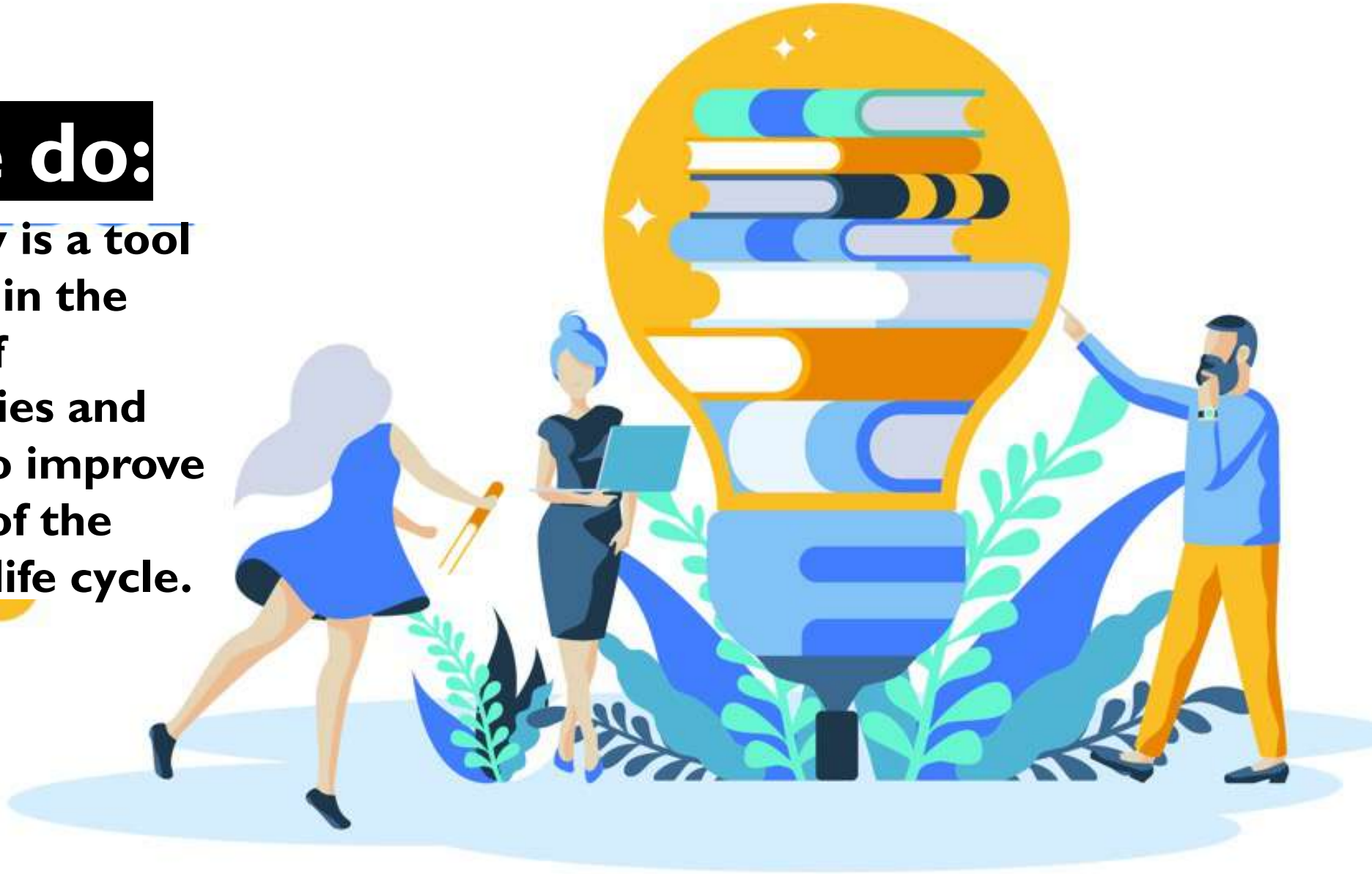
- Quality Management
- Verification & Validation management
- Risks Management
- Smart Authoring

REUSABILITY

- Interoperability
- Retrieval & Archiving
- Adaptability to Existing Toolsets

What we do:

The REUSE Company is a tool vendor specialized in the application of semantic technologies and artificial intelligence to improve the digitalization of the Systems Engineering life cycle.



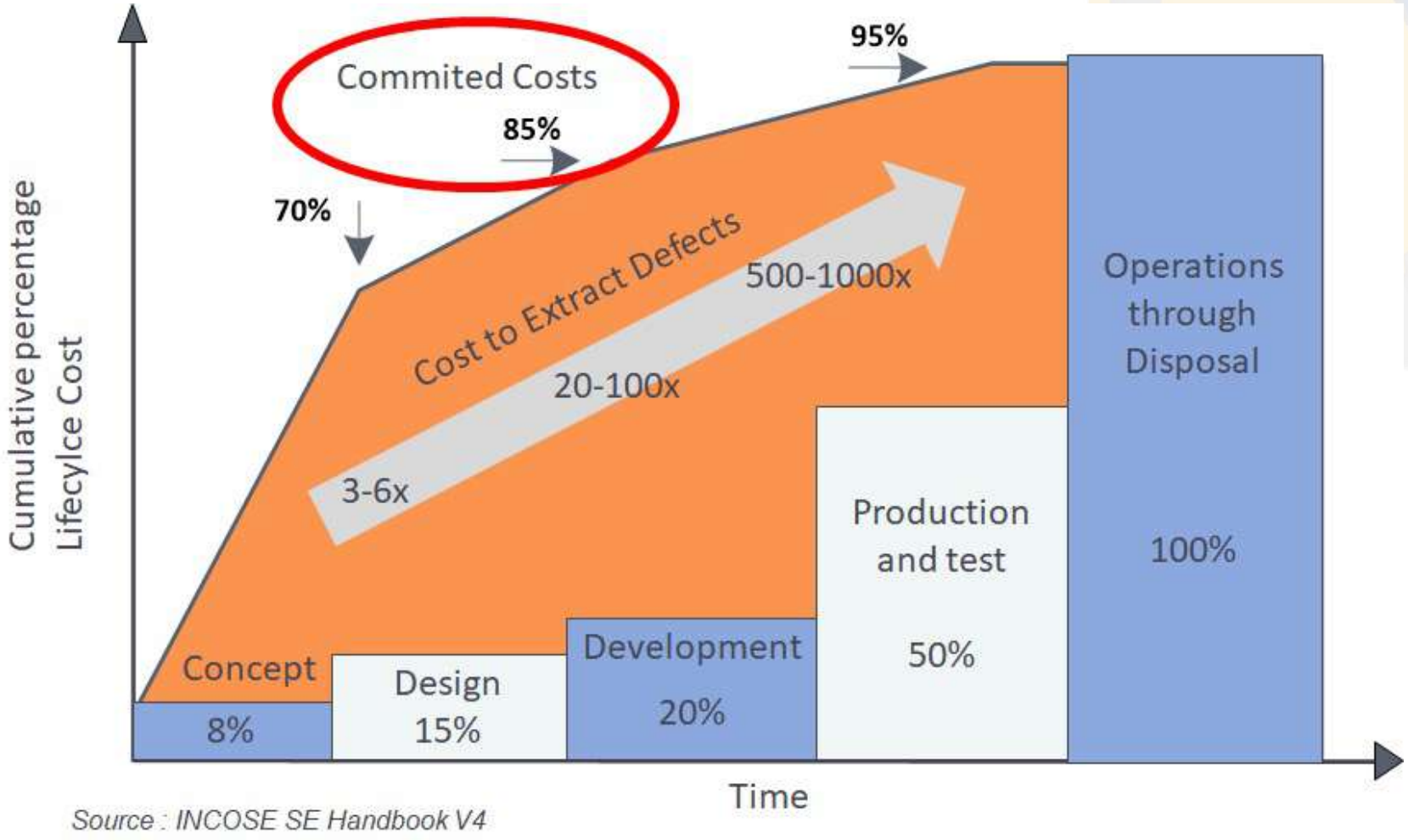
José Fuentes

- **Current position:** Chief Sales Manager of The REUSE Company
- Former Product Manager of RQA and the Systems Engineering Suite
- INCOSE CSEP Certified
- Graduated in the INCOSE Institute for Technical Leadership
- Active contributor to the INCOSE Guide for Writing Requirements
- Other certifications: ITIL
- Other interests: Project Management, Business Analysis, Risk Management

The need for early Verification

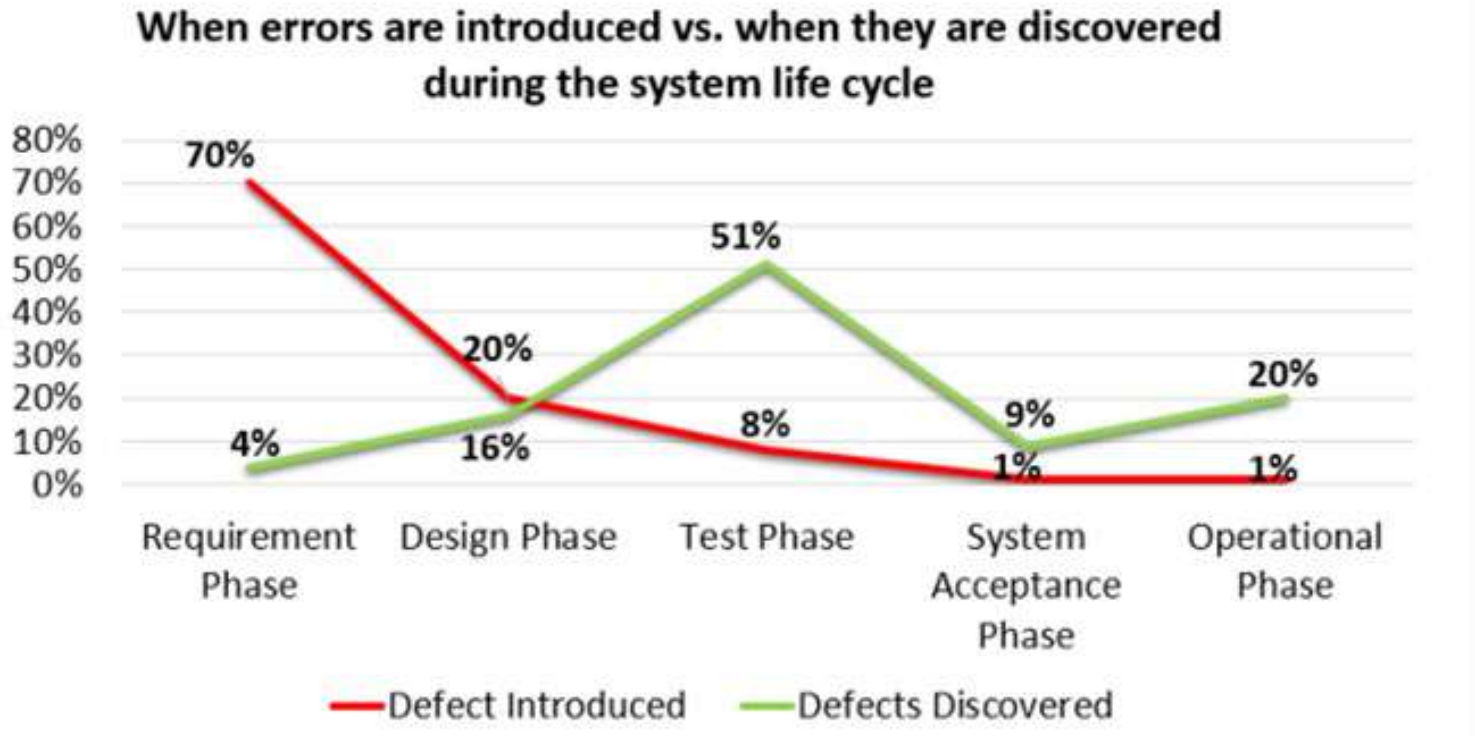
ECSS - Drafting Rules

The need for Early Verification

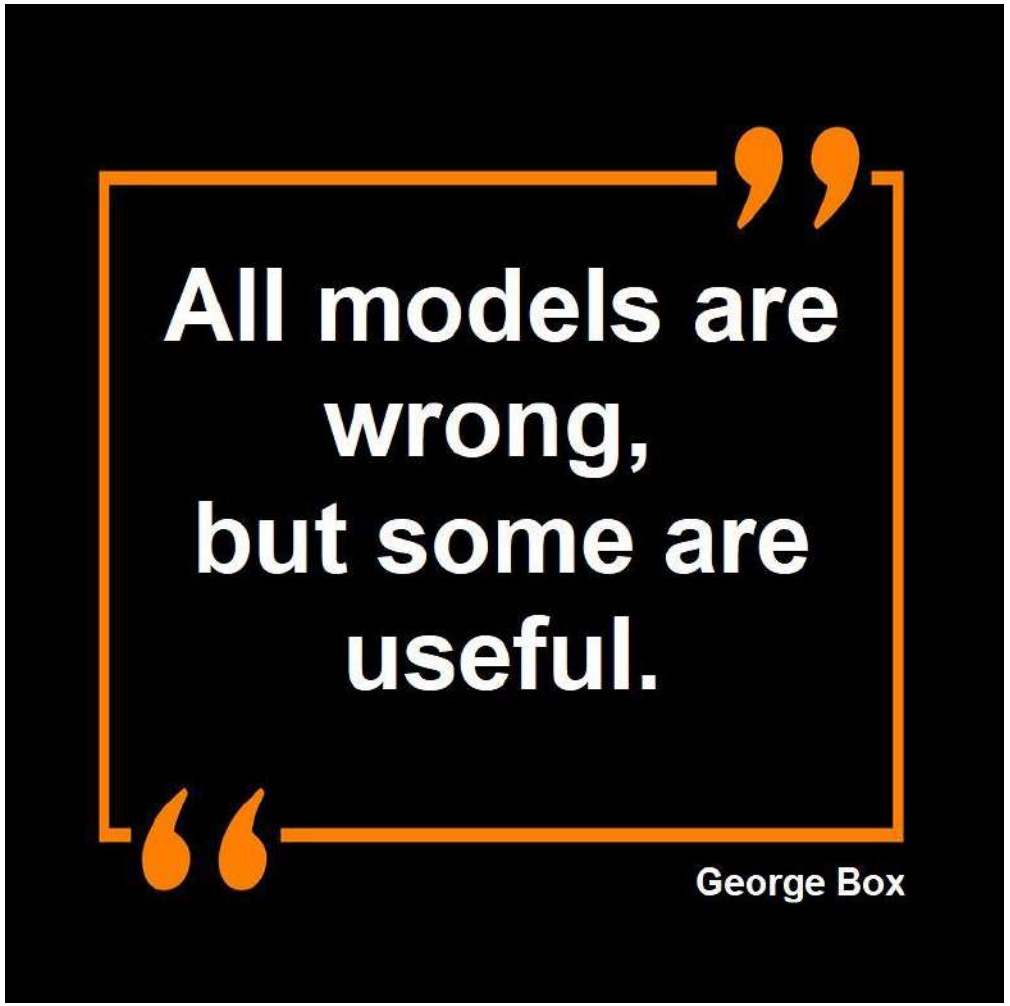


Source : INCOSE SE Handbook V4

REQUIREMENTS are the reason for FAILURE



Source: IBM Business research 2017



- "The practical question is how wrong do they have to be not to be useful."

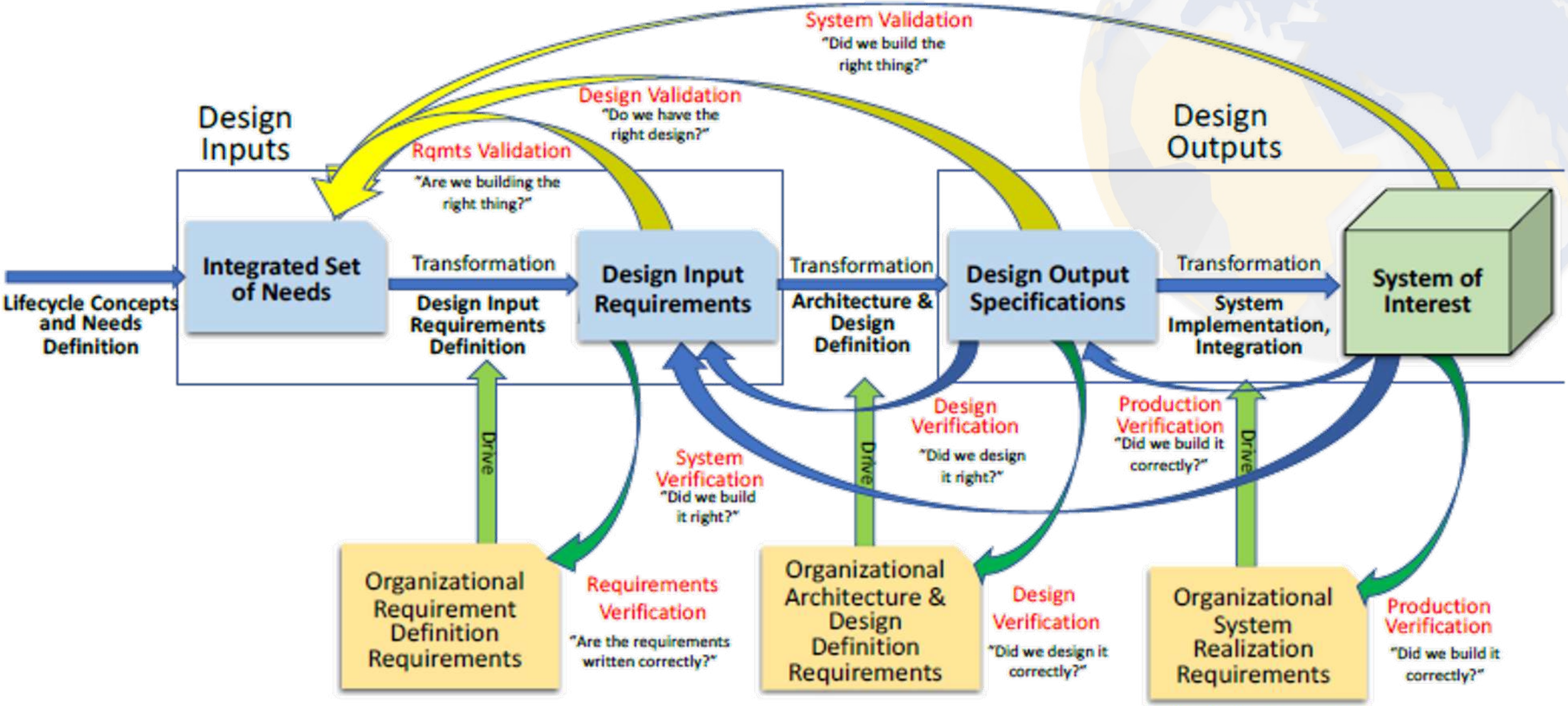
  George E.P. Box

- "Investing on the quality of models (through early inspections) is key for ensuring useful models, and thus, the success of our systems."

  José M. Fuentes

- **The notion of *quality* and verification in MBSE**
- Different stand-points
- 1. Conformance with Requirements:
  - Since a design model comes from the transformation of the textual requirements
  - This kind of conformance is already implemented in RQA: consistency and completeness between requirements and models
- 2. Fit for purpose:
  - Different models can implement the same set of requirements with different degrees of quality
- But all these models must satisfy several rules agreed in the organization

The need for Early Verification



Derived from Ryan, M. J.; Wheatcraft, L.S., "On the Use of the Terms Verification and Validation", February 2017

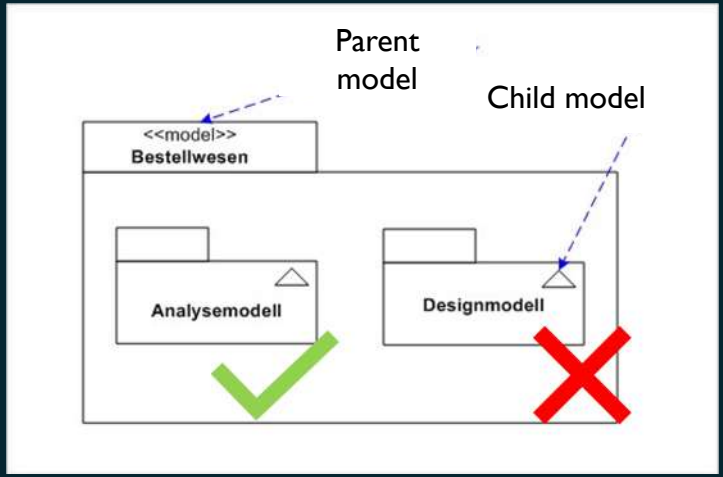
- **The notion of *quality* and verification in MBSE**
- Different degrees of inspection:
  - Pass-arounds
  - Peer-reviews
  - Walkthroughs
  - Inspections: PDR, CDR
  - Audits
- Regardless the level of the inspection, every inspection is "expensive"
- Early and frequent informal inspections is key to success; then formal ones later
- Using a tool performing some easy yet time-consuming checks is another key to success

The need for Early Verification

"The **verification** process can be applied to any engineering element that has contributed to the definition
and realization of the system itself" (e.g., verification of a system requirement, a function, an input/output flow, a system element, an interface, a design property, a verification procedure)."
INCOSE SEH v4

"The purpose of the verification process is to provide evidence that no error/defect/fault has been introduced at the time of any transformation of inputs into outputs; i is used to confirm that this transformation has been made "right" according to the requirements and selected methods, techniques, standards, or **rules**".
INCOSE SEH v4

"When using **models and simulations** it is important that they go through a **verification process** to ensure they are formed correctly as well as a **validation process** to ensure they correctly represent the entity they are modeling."
INCOSE NRVVLM



Rules for MBSE

Rules for MBSE

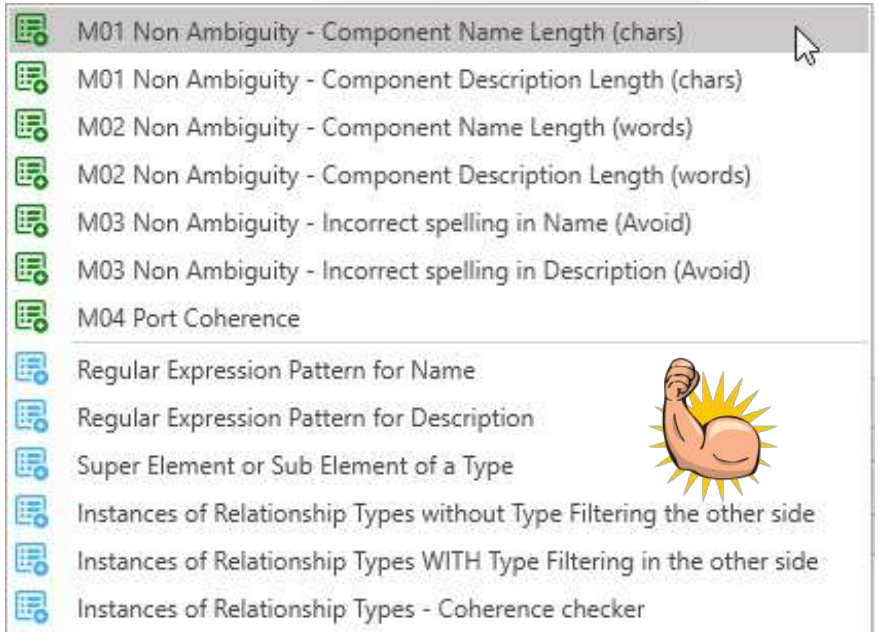
- Name of most kind of model elements should not be empty
- Name of some kind of elements should follow a given template (e.g. interfaces)
- Name of model elements should not include misspelling errors
- Description of some elements should not be empty
- Description of some elements should follow a specific template (e.g. use cases)
- Descriptions should not include misspelling errors
- Actors, logical systems and many other elements should be contained in at least one diagram
- Follow the 7±2 rules when decomposing systems or functions
- Stakeholder should be associated with at least one Use Case
- Actors or Stakeholders cannot be associate with other actors
- States must have at least one incoming and one outgoing transitions
- Some specific types of model elements must have a stereotype assigned
- Check the correct direction of a port
- Check the correct stereotype of a port
- …


THE REUSE COMPANY
RQA
QUALITY STUDIO
By The REUSE Company
Quality
Correcteness
Completeness
Consistency
Full License

RQA for MBSE

- **Non-parameterized metrics:**
  Do not require any parameter (configuration) to be executed.
  - E.g.: Incorrect spelling in any component description

- **Parameterized metrics:**
  Require some kind of parameter (configuration) to be executed.
  I.e., these metrics are algorithms that need some parameters to provide the output.
  - E.g.: Regular expression pattern check on the component name
    - It's mandatory to provide a regular expression to execute the metric
- There are optional parameters in **all** MBSE metrics (parameterized or non-parameterized):
  - This is the ***Filter***
  - To restrict the analysis only on those components of a given type
  - E.g.:
  - Description is not empty of the components of type Use_Case
    - Regular expression pattern on the description of the components of type Use_Case



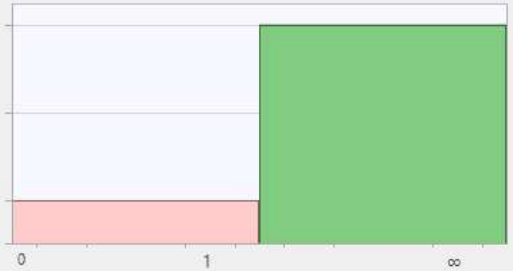
Quantitative vs Qualitative result for a metric

- **Quality function:**
  Every metric always returns a numerical result (i.e. quantitative value)
- This result (number) is transformed into a qualitative value through its ***quality function***

Quality function for selected metric:

Range	Mandatory	Quality Level	Summary	Description
[0 , 1)	No	★☆☆	An actor must be connected to, at least, one use case	
[1 , ∞)	No	★★★		

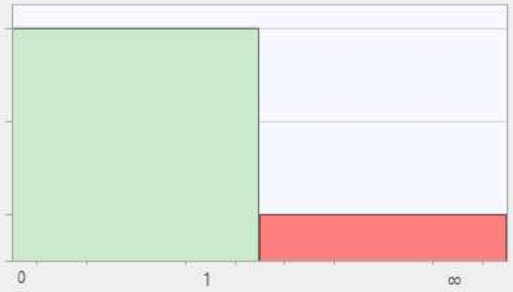
No. of ranges: 2



Quality function for selected metric:

Range	Mandatory	Quality Level	Summary	Description
[0 , 1)	No	★★★		
[1 , ∞)	No	★☆☆	Avoid associations among stakeholders	

No. of ranges: 2



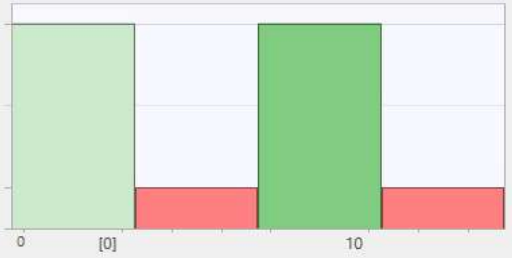
Quantitative vs Qualitative result for a metric

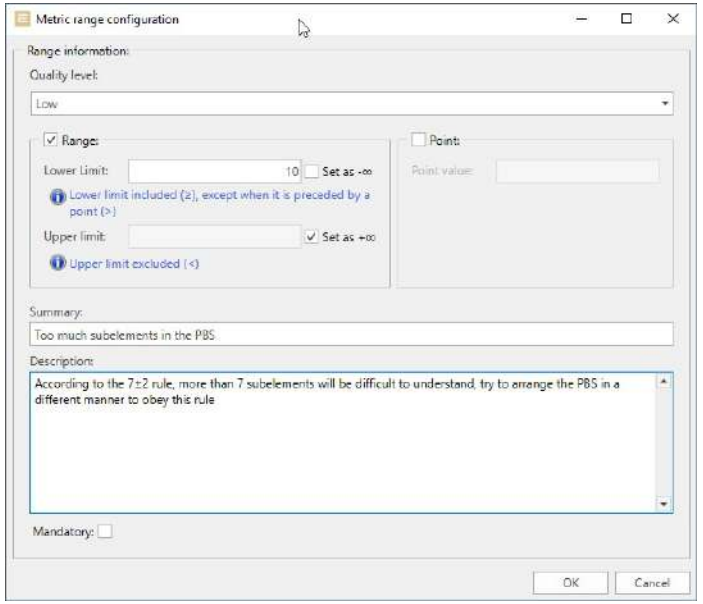
- **Quality function:**
  Every metric always returns a numerical result (i.e. quantitative value)
- This result (number) is transformed into a qualitative value through its ***quality function***

Quality function for selected metric:

Range	Mandatory	Quality Level	Summary	Description
[0]	No	★★★		
(2 , 5)	No	★☆☆	Too few subelements in the PBS	In order to follow the 7±2 rule, try to decompose this LogicalSystem into a larger nu...
[5 , 10)	No	★★★		
[10 , ∞)	No	★☆☆	Too many subelements in the PBS	According to the 7±2 rule, more than 7 subelements will be difficult to understand, tr...

No. of ranges: 4





Non-parameterized metrics

- **M01 – Text length for Name / Description (characters)**

  Computes the number of characters included in the Name or Description property of a model element.

- **M02 – Text length for Name / Description (words)**

  Computes the number of words included in the Name or Description property of a model element.

- **M03 – Misspelled words for Name / Description**

  Computes the number of misspelled words included in the Name or Description property of a model element.

Non-parameterized metrics

- **M04 – Missing port type**

  Only applicable to *Ports*, checks that every port always has a *Type* assigned to it.

- **M05 – Stereotype check**

  Counts the number of stereotypes assigned to a model element.

Regular Expression Pattern for Name / Description

- **Description:**

  Computes the **Name** or the **Description** property of the model element with a regular expression pattern defined by the user.

  The return value is the number of matches of the regular expression pattern inside the text of the property.

- **Example of rules based on this metric:**
  - The name of every *interface* starts with a lower *i*, followed by a capital letter
  - The description of every *use case* includes the typical pattern:

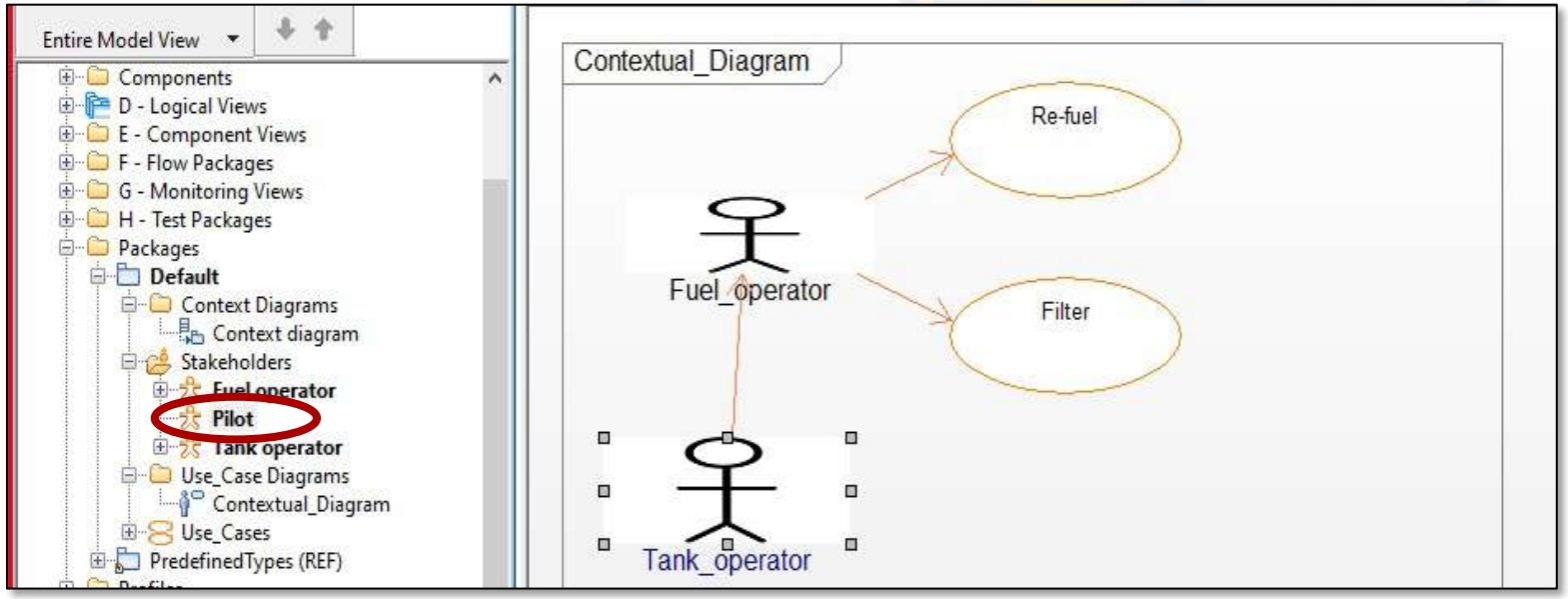
Name	Name of the UC
Pre-condition	To be true for the UC to be launched
Main flow	Actions...
Alternative flow	Actions when something doesn't go as expected
Post-condition	True once the UC is executed

Owner element of a type

- **Description:**

  Checks that model elements of a given type are always owned (contained) by elements of another given type

- **Example of rules based on this metric:**
  - A *Logical System* is contained in at least one Diagram
  - An *Actor* should be contained in at least one Diagram



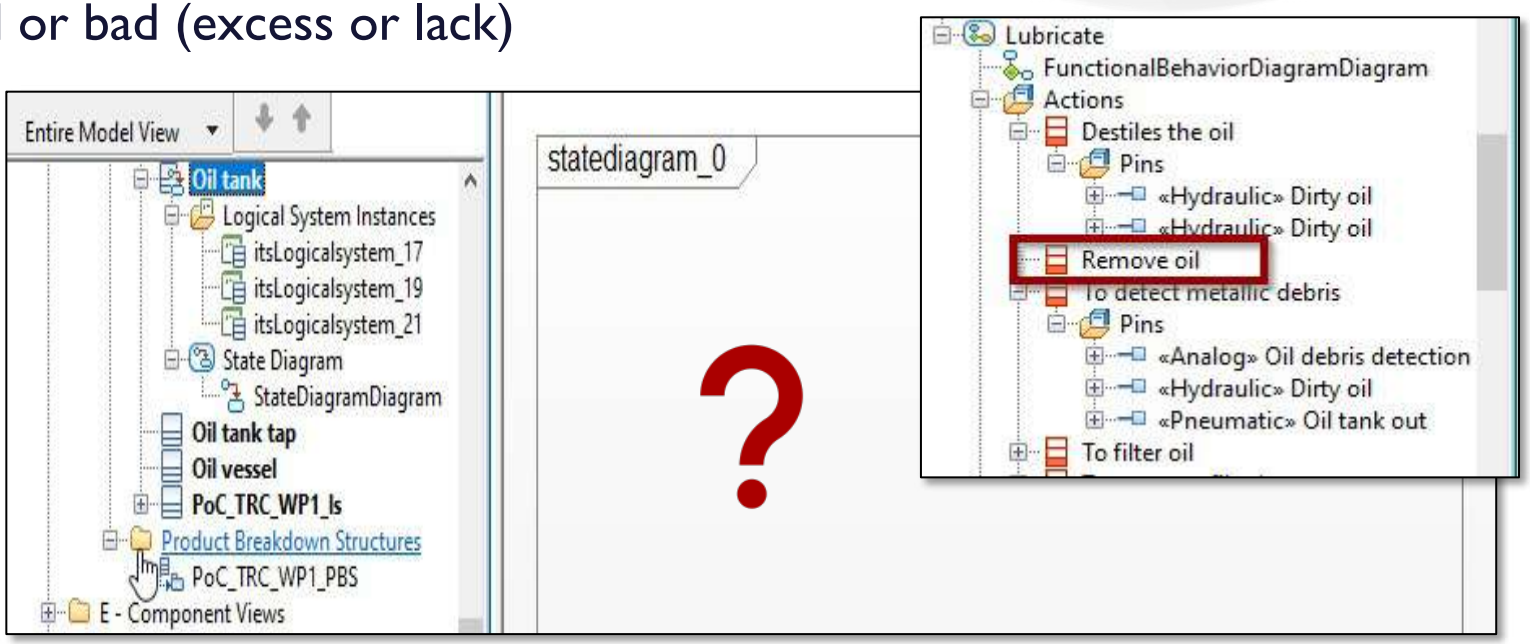
Owned elements

- **Description:**

  Checks that model elements of a given type always own (contain) elements of another given type

  In fact, the metric counts how many of those elements are owned then, the quality function will determine if this number is good or bad (excess or lack)

- **Example of rules based on this metric:**
  - A *state machine* without any *state*
  - An empty diagram
  - An *interface* without *operations*
  - A *Function* without *Pins*

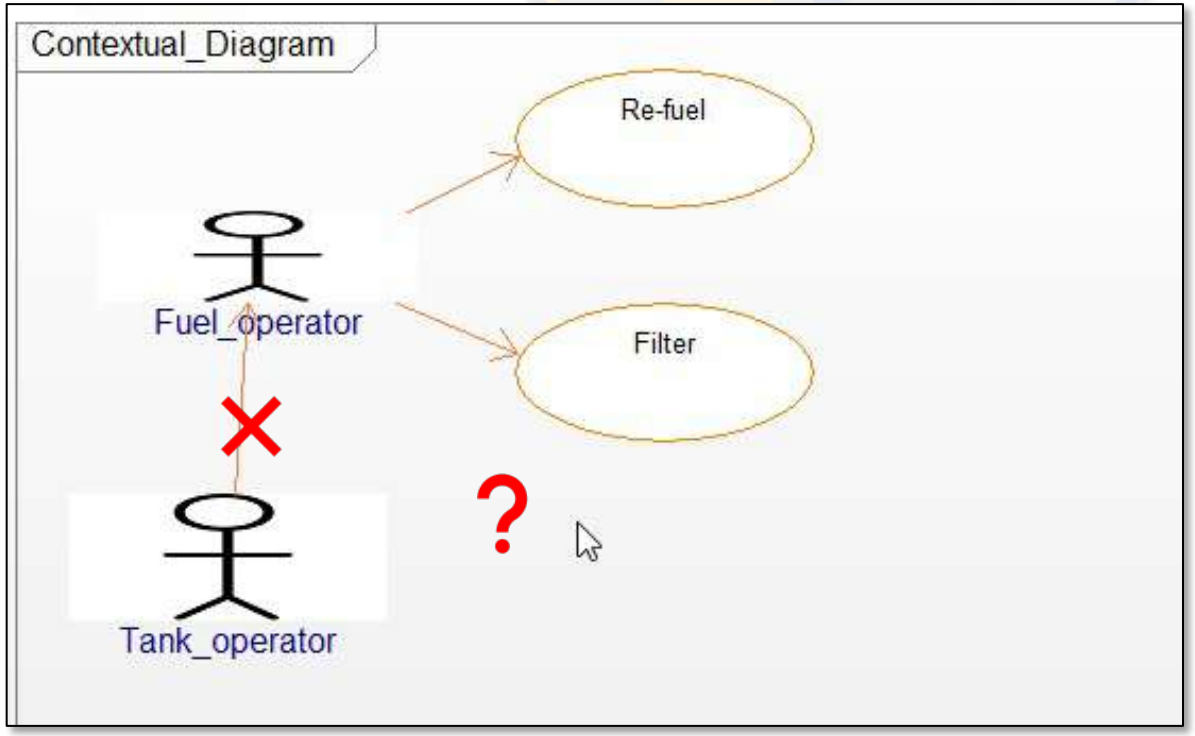


Relationship instances

- **Description**:

  Given an object (X), computes the number of the relationship instances of a given type (Y) where X is involved.

- **Optional configurations:**
  - It can be established if X shall play the role of source (from) in the relationship, target (to), or any of them
  - Additionally, the other element in the relationship (Z) can be checked to have a type or stereotype included in a list of valid ones.



- **Example of rules based on this metric:**
  - An *Actor* should be associated with, at least, one use case
  - An *Actor* cannot be associated with another *Actor*
  - A *Pin* should not be connected directly to a function by means of an *ObjectFlow*, it should be connected through another *Pin*

Coherence of relationships

- **Description:**

  Given an object (X) of a given type, computes the number of relationships of a given type (Y) that fails to fulfil a set of conditions among the following ones:

  - Same type
  - Same stereotype
  - Valid direction: the inputs and outputs must be coherent (see column on the right)
  - Valid name: several matching possibilities, same name, synonymy or belonging to same Conceptual Model View (ontology check)

- Direction coherence checks:
  - Among elements in the same level of the hierarchy

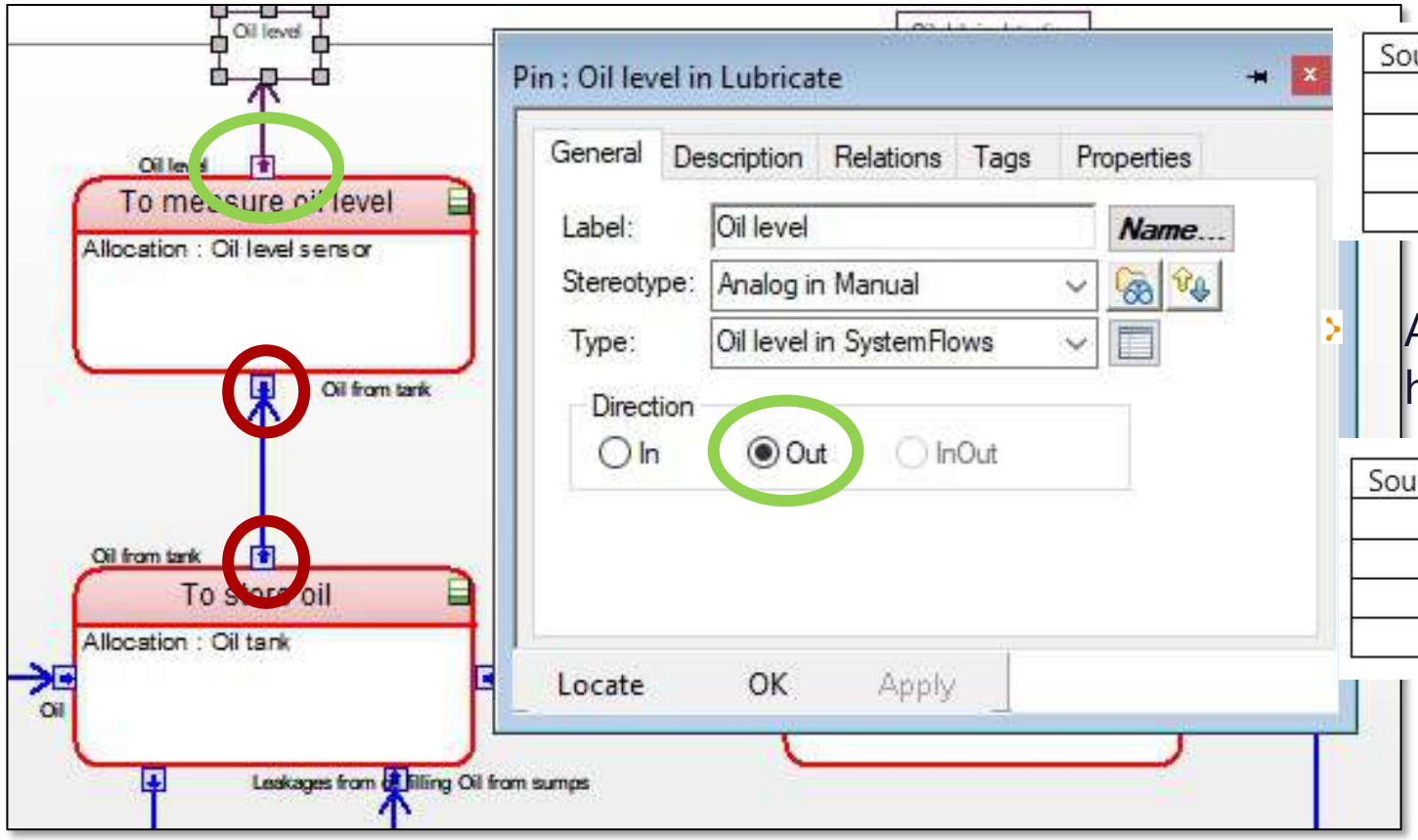
Source Port/Pin direction	Destination Port/Pin 2	IsValid
Input	Output	✓
Output	Input	✓
Output	Output	--
Input	Input	--

  - Among elements in different levels of the hierarchy

Source Port/Pin direction	Destination Port/Pin 2	IsValid
Input	Output	--
Output	Input	--
Output	Output	✓
Input	Input	✓

Coherence of relationships

- **Example of rules based on this metric:**



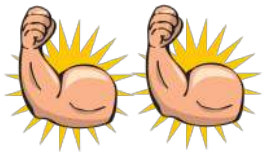
- Direction coherence checks:
  - Among elements in the same level of the hierarchy

Source Port/Pin direction	Destination Port/Pin 2	IsValid
Input	Output	✓
Output	Input	✓
Output	Output	--
Input	Input	--

  - Among elements in different levels of the hierarchy

Source Port/Pin direction	Destination Port/Pin 2	IsValid
Input	Output	--
Output	Input	--
Output	Output	✓
Input	Input	✓

- Custom-code metrics



Parameterized custom-code metric configuration

Information | Computation

Custom-code metric configuration:

Type:

Type 7 : ArtifactAndEvaluation TYPE7(CustomCodeCorrectnessInputInformation inputInformation, & Custom Parameters)

External library file:

Assembly:

Class:

Method:

Built-in code editor:

Code the metric in C# language using the programming environment built-in in SES ENGINEERING Studio: Code

Configure the input values for the parameters at the metric evaluation. Configure

Calculate this metric on the text matching:

Pattern group: Pattern:

Negative: Calculate this metric on the text NOT matching

Test custom-code | OK | Cancel

Source Code Editor

File Edit Build

Type 7, expected return value {ArtifactAndEvaluation}

```
ArtifactAndEvaluation metricEvaluation = new ArtifactAndEvaluation();
            Artifact indexingArtifact = null;
            bool destroyArtifact = false;
            try {

                Meta metaAbsoluteValue = null;
                Meta metaTolerance = null;
                Meta metaAbsoluteValueQualifier = null;
                Meta metaToleranceQualifier = null;
                bool evaluate = false;

                List<Tuple<Meta, Meta, Meta, Meta>> evaluateSet = new List<Tuple<Meta, Meta, Meta, Meta>>();

                HashSet<double> results = new HashSet<double>();

                if (inputInformation != null && inputInformation.Formalization != null) {
                    const string AbsolutevalueConstant = "AbsoluteValue";
                    const string ToleranceConstant = "Tolerance";

                    if (inputInformation.FilteringPattern != null) {
                        destroyArtifact = true;
                        indexingArtifact = new Artifact(inputInformation.Indexer.Repository, "Indexing", inpu
                        inputInformation.Indexer.RefreshPatternsTree(inputInformation.FilteringPattern);
                        inputInformation.Indexer.IndexText(inputInformation.RequirementText, indexingArtifact
                    }
                    else {
                        destroyArtifact = false;
                        indexingArtifact = inputInformation.Formalization;
                    }
                    List<List<Meta>> complexMetas = indexingArtifact.ComplexMetaProperties;

                    bool continueLoop = true;
```

Using / Imports

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Cake.Engine;
using Cake.Indexer;
using Cake.RBS.Compiler.Rule;
using Rqa.IndividualMetrics;
```

Using / Imports | Assemblies

Parameters

Cardinality	Type	Name	R.
Element	Rqa.IndividualMet...	inputInformation	

Compiler Errors

Save and close | Cancel

Ready

Compatibility

CAMEO SYSTEMS MODELER™

R

Capella

ENTERPRISE ARCHITECT

papyrus

MATLAB® & SIMULINK®

OpenModelica

ALTIUM DESIGNER

How to use it

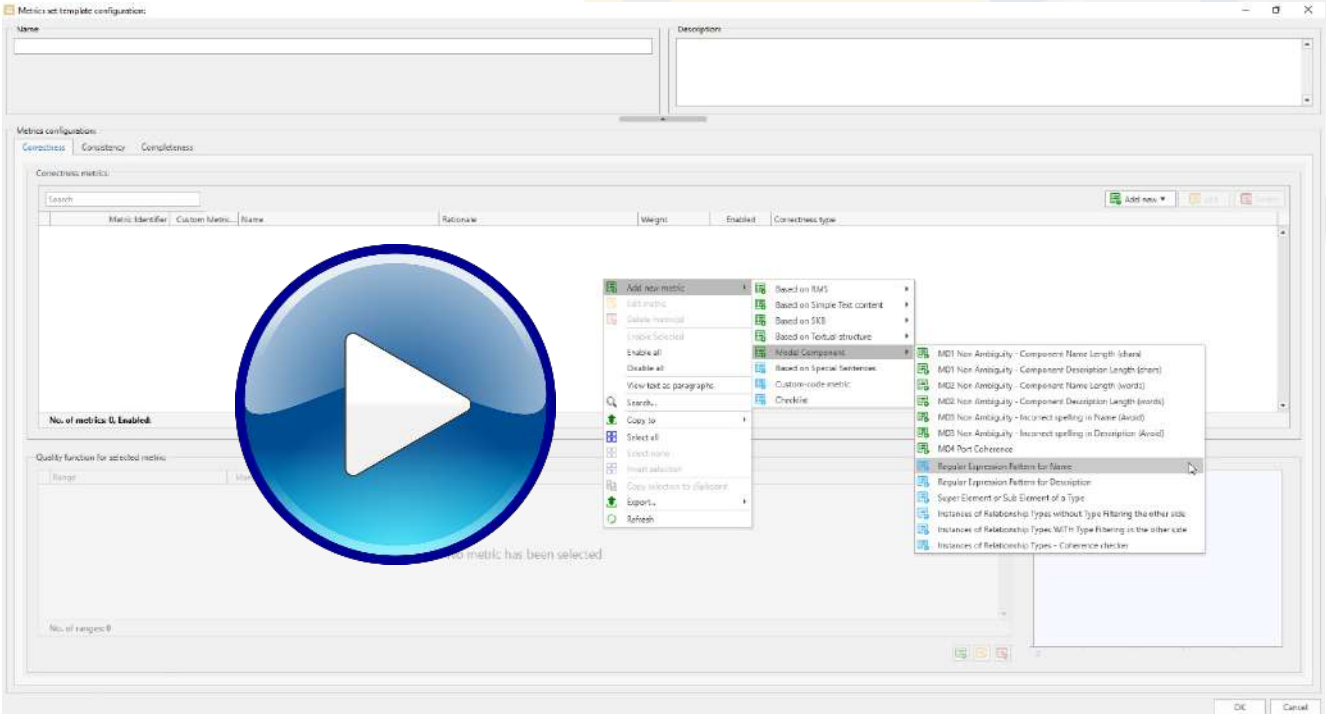
- The MBSE Quality metrics will be available with the Systems Engineering Suite v22
- To be released: June 2022
- How to enable these metrics:
  - Some metrics will be available as a Library: visit our website https://www.reusecompany.com/libraries
    - Use KM – KNOWLEDGE Manager to import the library
    - Use RQA to assign the catalogue of rules to your model
  - Start from scratch creating your own catalogue of rules for models: check out use-case #1

Demonstration

MBSE Metrics: Live Demo: Use case #1

- **Use case #1:** Prepare your own catalogue of MBSE metrics
- **Steps:**

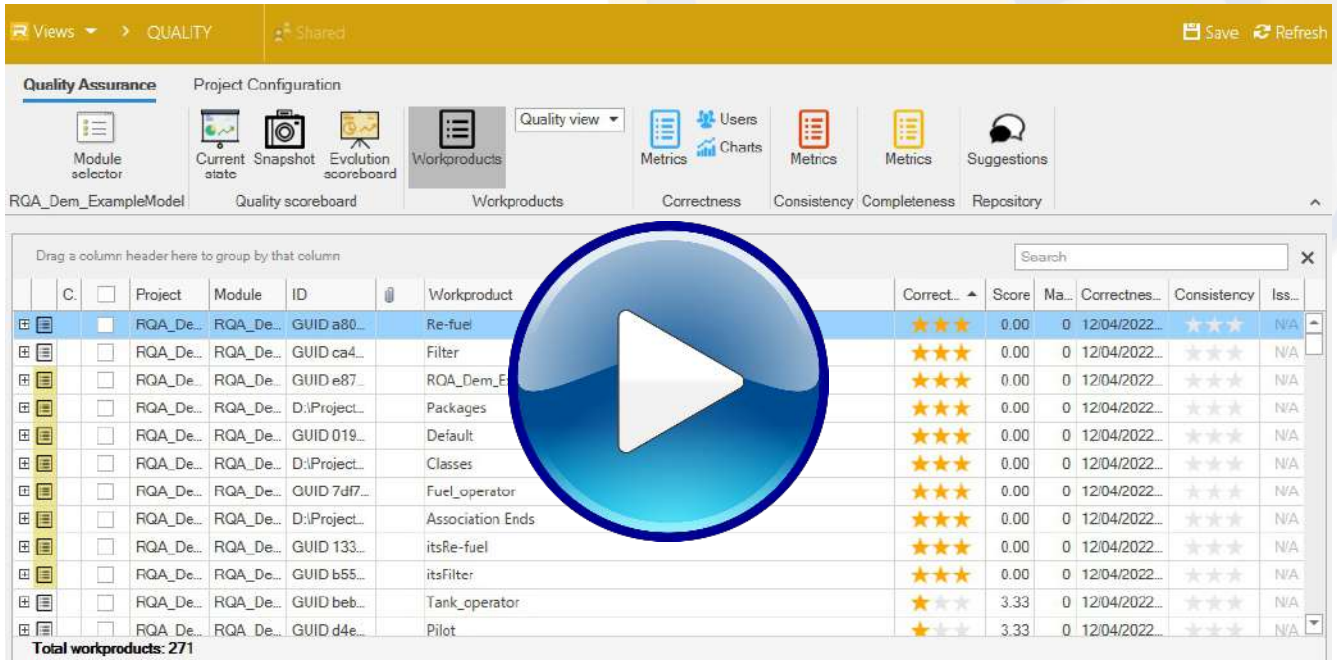
1. Open the SES ENGINEERING Studio
2. Go to *Quality/Metric set template*
3. Create your own template
4. Add non-parameterized metrics
5. Add some parameterized metrics



MBSE Metrics: Live Demo: Use case #2

- **Use case #2:** Detailed quality analysis
- **Steps:**

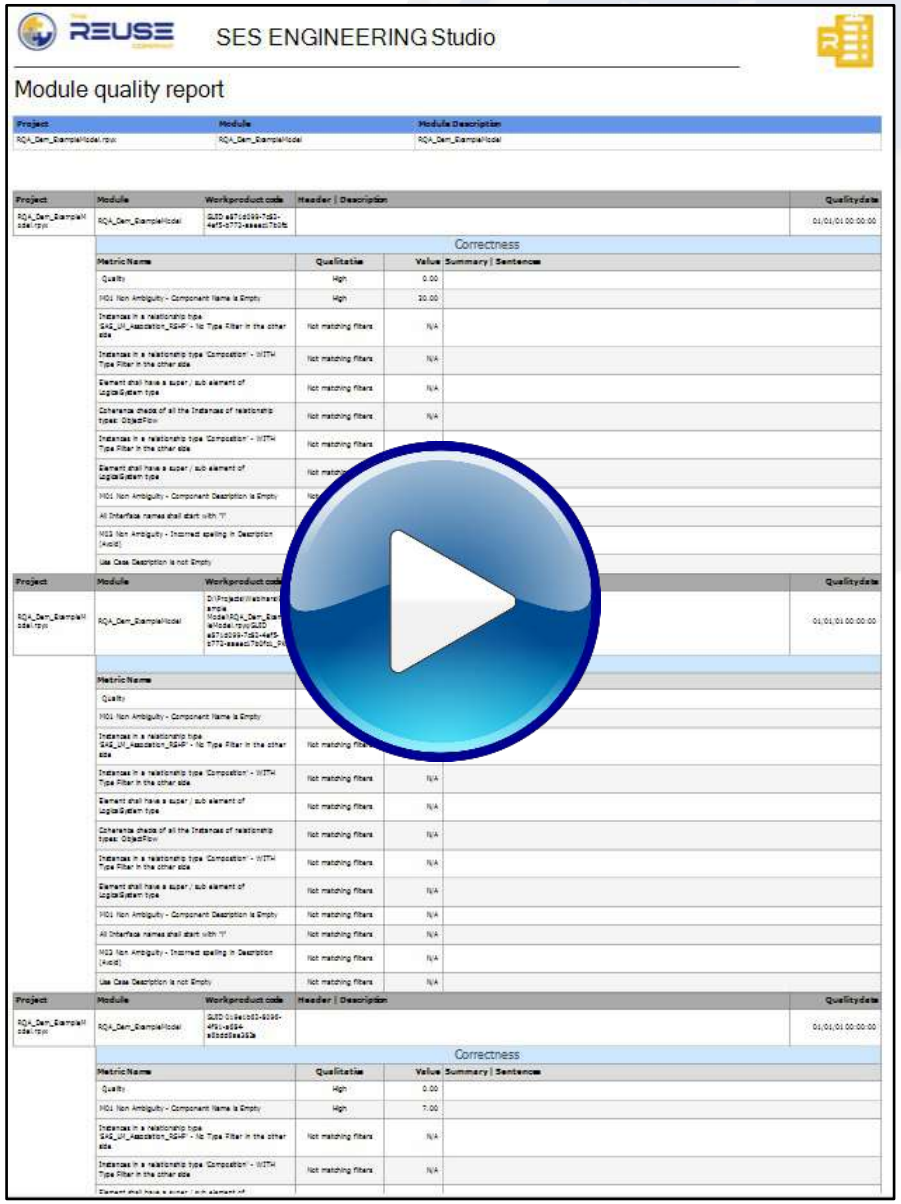
1. Open the SES ENGINEERING Studio
2. Connect to your model
3. Assign a catalogue to your model
4. Analyze the quality of your model
5. Show details
6. Open in the source tool the model elements involved in the different quality issues



MBSE Metrics: Live Demo: Use case #3

- **Use case #3:** Quality Reporting
- **Steps:**

1. Open the SES ENGINEERING Studio
2. Connect to your model
3. Create a baseline of the quality of your model
4. Change the rules assigned to your model
5. Analyze again the quality of your model
6. Show the quality evolution among different baselines
7. Generate a report




Q&A

Next webinar

- **Taming the System Engineering Life cycle using Connectivity and Interoperability: the SES ENGINEERING Studio**
  - The SES ENGINEERING Studio is a Software Tool designed to orchestrate the development of all kinds of systems (hardware, hybrid, software) by connecting and enabling interoperability between an unlimited number of existing Systems Engineering Tools (Requirements Management, MBSE tools, Simulation Tools, Risks Management, RAMS Management, MS Office, etc.).
  - This new SW tool promotes lifecycle management methodologies guided by REUSE, based on a knowledge-centric approach, supporting the notion of authoritative source of truth, offering connectivity to everything, unlimited interoperability, and providing full support to technical management as in ISO 15288
  - This webinar will present the core concepts of this approach, introduce the SES ENGINEERING Studio tool, and will show some connectivity, interoperability and technical management examples applied to existing market tools. (IBM DOORS, DS Cameo, MW Simulink, Microsoft Office, etc.)

- **Dates:** May 10th and 12th, 2022


Q&A

Contact information

José M. Fuentes

jose.fuentes@reusecompany.com

+34 912 17 25 96

@ReuseCompany

https://www.linkedin.com/in/josemiguelfuentes/

THE REUSE COMPANY