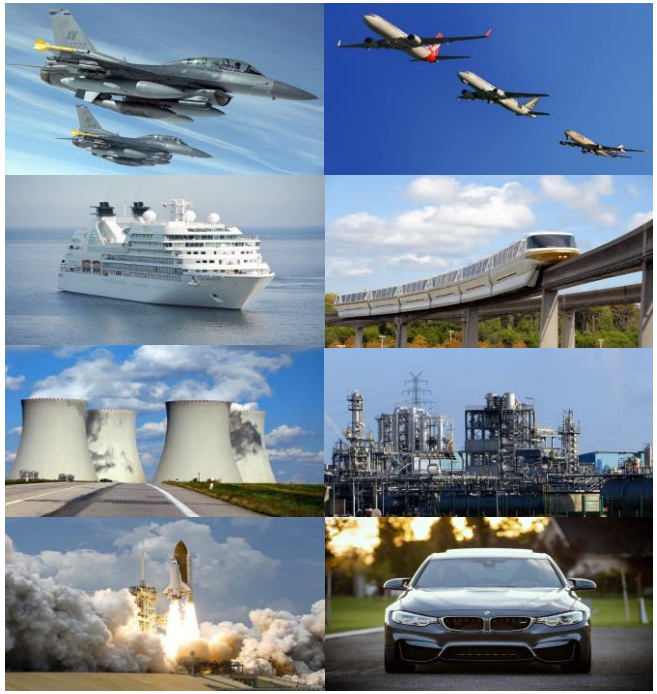
Elevating the meaning of data and operations within the development lifecycle through an interoperable toolchain

Jose María Alvarez, Roy Mendieta & Juan Llorens | Assoc. Prof. UC3M | josemaria.alvarez@uc3m.es

Introduction

Characteristics?



Aspect	Comment
Type of product	Complex (very complex!)
Development lifecycle	Multidisciplinary (software, mechanics, electronics, etc.)
→	Time and costs
Functionality	It is being increased over time
Lifetime	Long (+30 years)
Regulation (under)	High
Suppliers	Thousands
Engineers	Thousands
Customers	Hundreds
Scope	International
…	…

Introduction

Lifecycle processes

System Life Cycle Processes

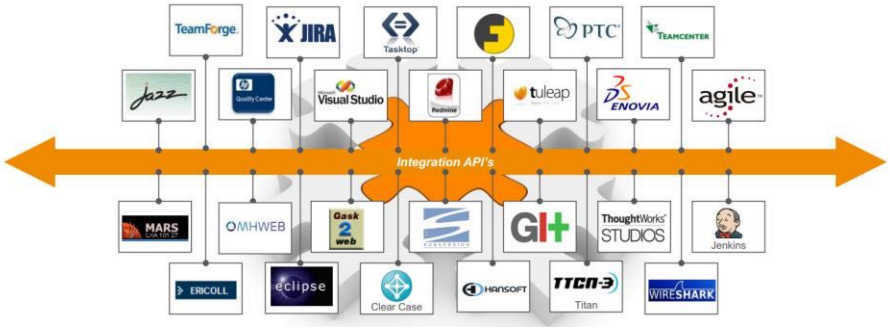
Agreement Processes
- Acquisition Process (Clause 6.1.1)
- Supply Process (Clause 6.1.2)

Organizational Project-Enabling Processes
- Life Cycle Model Management Process (Clause 6.2.1)
- Infrastructure Management Process (Clause 6.2.2)
- Portfolio Management Process (Clause 6.2.3)

Technical Management Processes
- Project Planning Process (Clause 6.3.1)
- Project Assessment and Control Process (Clause 6.3.2)
- Decisión Management Process (Clause 6.3.3)
- Risk Management Process (Clause 6.3.4)
- Configuration Management Process (Clause 6.3.5)
- Information Management Process (Clause 6.3.6)
- Measurement Process (Clause 6.3.7)

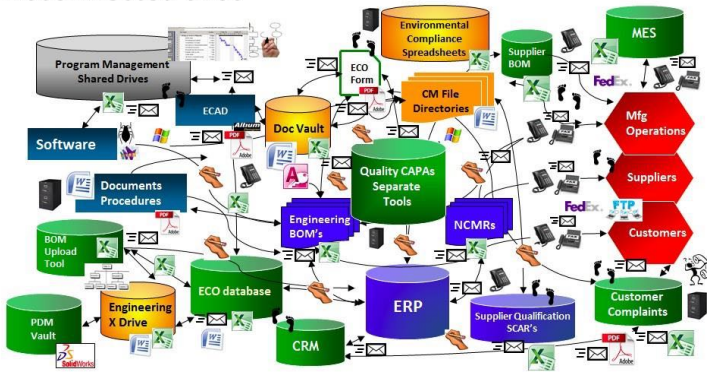
Technical Processes
- Bussiness or Mission Analysis Process (Clause 6.4.1)
- Stakeholder Needs & Requirements Definition Process (Clause 6.4.2)
- System Requirements Definition Process (Clause 6.4.3)
- Architecture Definition Process (Clause 6.4.4)
- Design Definition Process (Clause 6.4.5)
- System Analysis Process (Clause 6.4.6)
- Implentation Process (Clause 6.4.7)

Engineering (and corporate) environment



Mats Berglund (Ericsson)
http://www.ices.kth.se/upload/events/13/8440418gf85d41a6a7d1cafd0db4ee80.pdf

Disconnected Silos



Source: http://beyondplm.com/2014/07/22/plm-implementations-nuts-and-bolts-of-data-silos/

Some needs…

Knowledge

A **knowledge model** to drive the development lifecycle.

Discovery

A method to automatically **discovery and manage traces**.

Naming

A **common vocabulary** to standardize the naming of any system artefact.

Collaboration

An engineering environment to **ensure quality, save costs and enable team collaboration**.

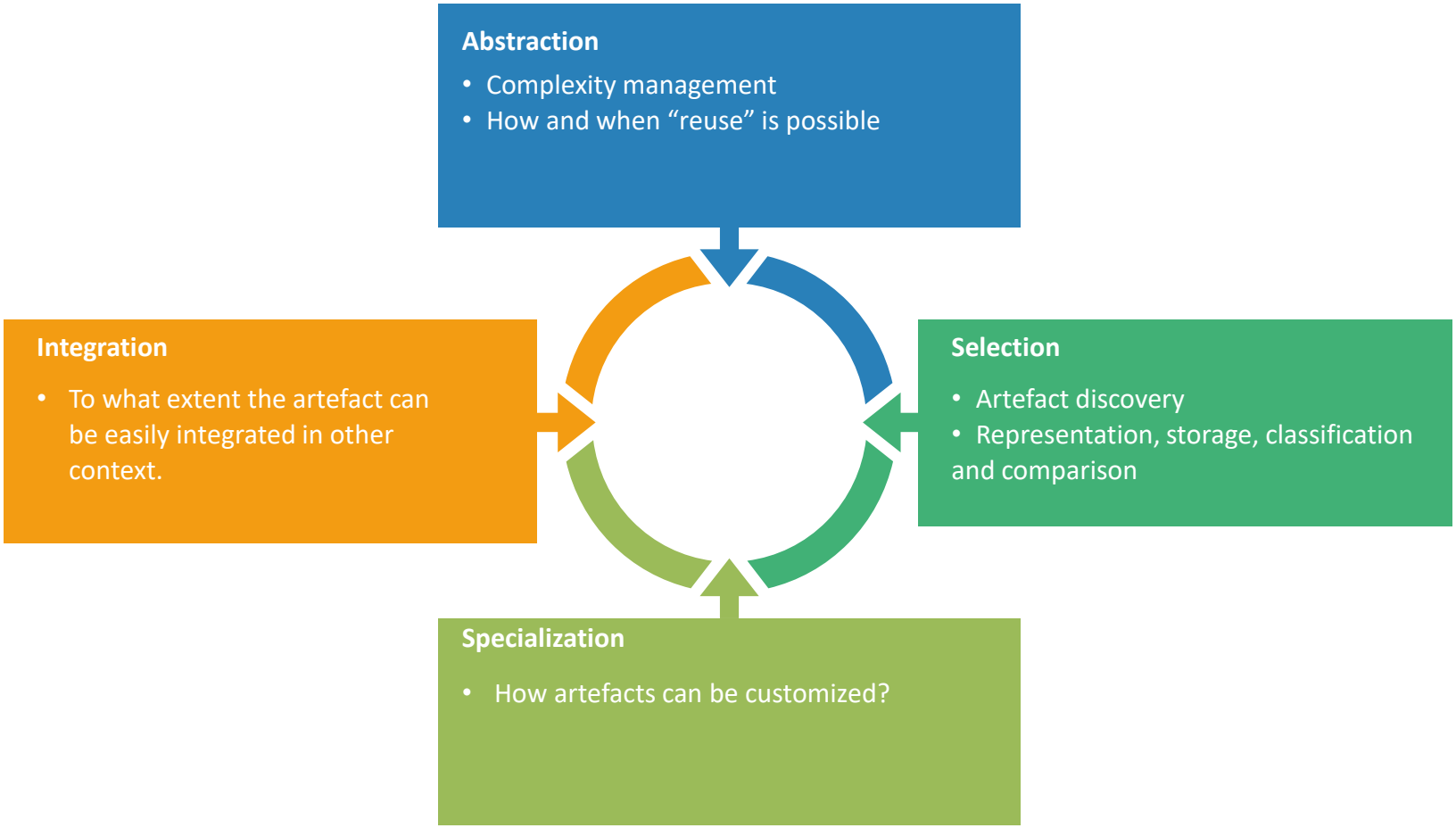
Ad-hoc integration

Integration of different tools.

Vendor Lock-in

A method to avoid the vendor lock-in ensuring **compatibility** in terms of models, formats, access protocols, etc.

Reuse principles



Main question

Is it possible to:

-improve the degree of reuse of any system artefact and
-deliver added-value services

through a common representation and an interoperable access model?

Related work: common needs

Data representation
- Common data model (vocabulary)
- Language

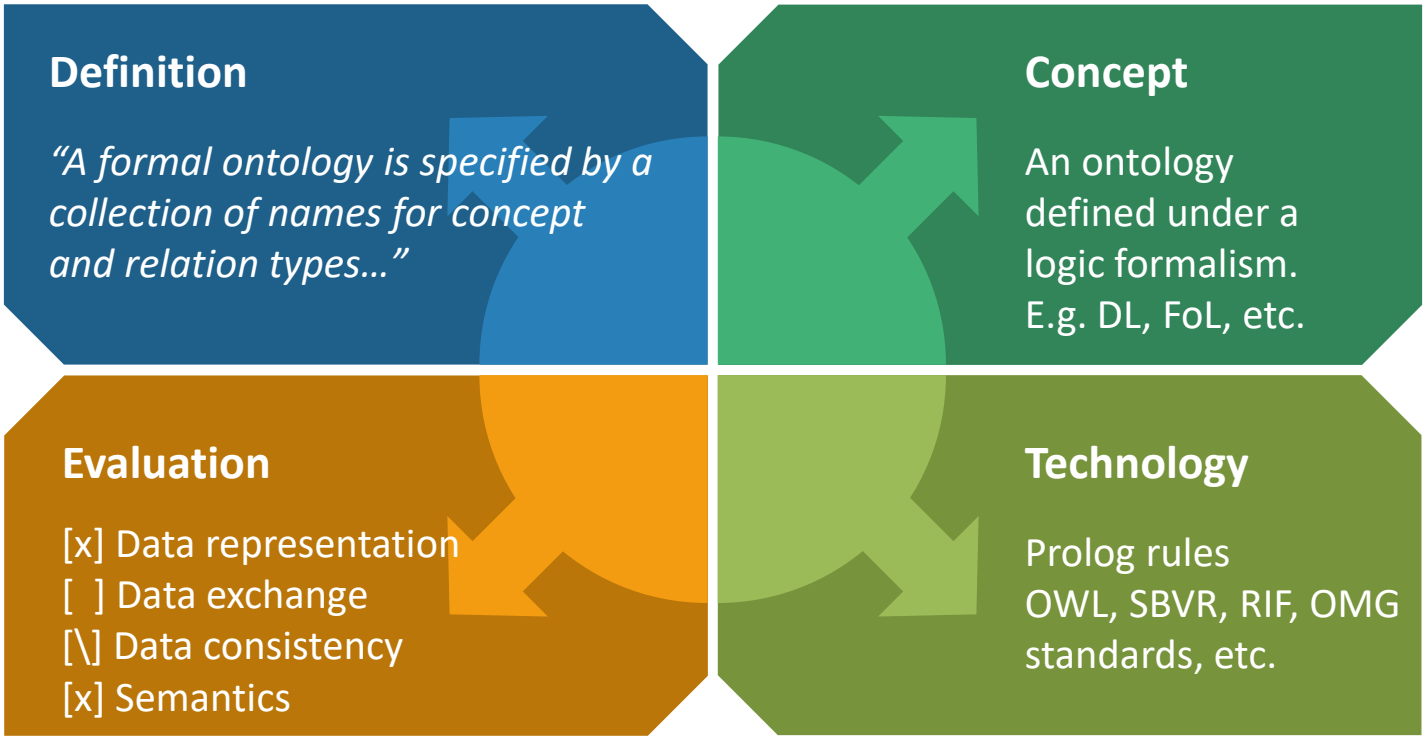
Data consistency & conformity
- Check integrity

As a service
- Methods to ease reuse

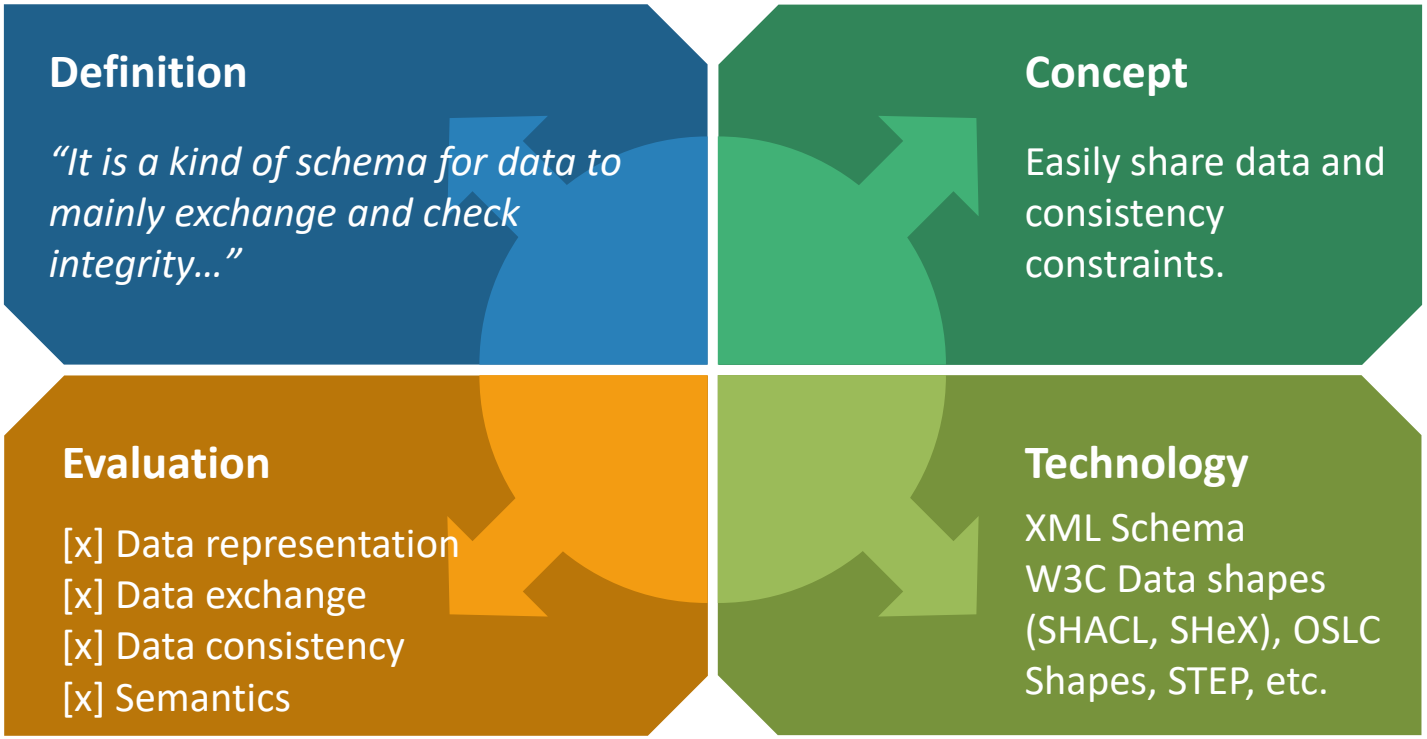
Data exchange (and sharing)
- Standardized formats and access protocols

Data interpretation
- Semantics

Formal ontology



Data Shape


Definition
"It is a kind of schema for data to mainly exchange and check integrity…"
Concept
Easily share data and consistency constraints.
Evaluation
[x] Data representation
[x] Data exchange
[x] Data consistency
[x] Semantics
Technology
XML Schema
W3C Data shapes (SHACL, SHeX), OSLC Shapes, STEP, etc.

Summary: Formal ontology vs Data shape

Formal ontologies

Main use:

- To create a knowledge base of the system: **knowledge creation (collaborative)**
- To perform reasoning processes for **knowledge inference**

How to use:

- Local and/or distributed reasoning
- **Not all ontologies are formal ontologies**

Warning:

- Do NOT use ontologies to perform data validation (consistency checking, etc.)→time consuming process
- Make ontologies "runnable" not just a document
- Avoid transformations from different paradigms but boost cooperation between paradigms
  - e.g. SysML←Transformation or cooperation?→OWL

Data Shapes

Main use:

- Data representation, exchange and consistency.
- Lightweight semantics→"The Shape"

How to use:

- Data as a Service: create standard-based APIs (technology is NOT relevant, FOUNDATIONS ARE)
  - OSLC
  - Swagger (Open API Specification)
- REST architectural style (JSON format)

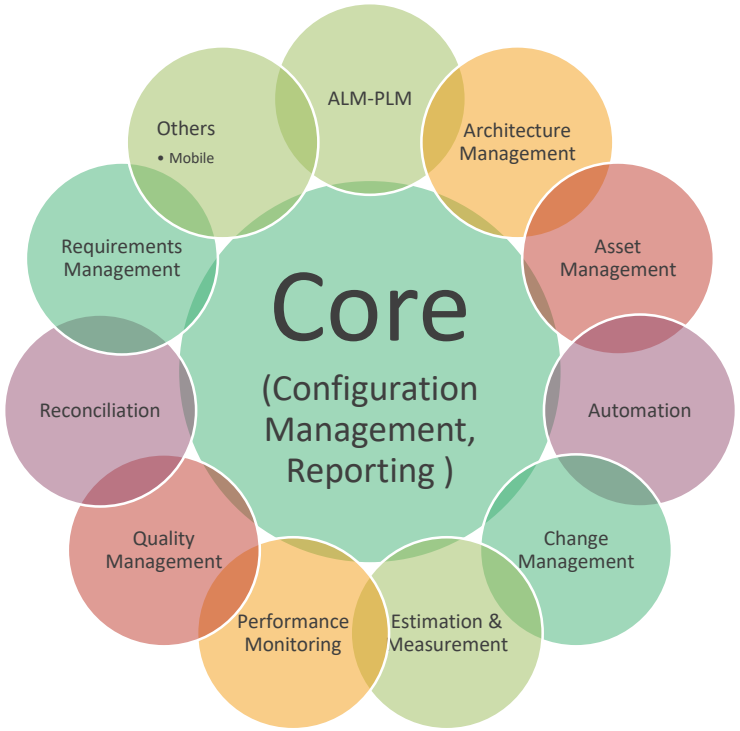
Warning:

- Define your URIs and methods properly
- Expose both: data and operations
- Document the use of the API

  →Swagger a good example

Related work: representation and data exchange

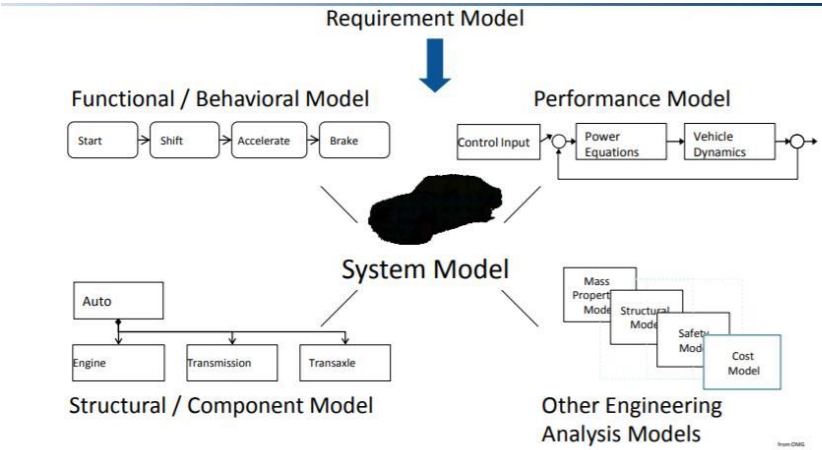
Open Services for Lifecycle Collaboration (OSLC)



REST services + Linked Data + Resource Shape

W3C Recommendation SHACL and Shape Expressions

Model-based Systems Engineering (MBSE) → SysML



ISO STEP 10303
(STandard for the Exchange of Product model data)

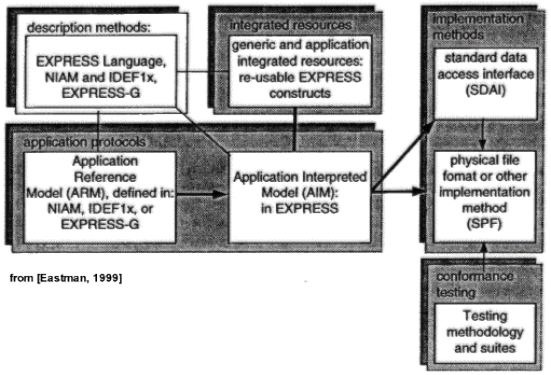
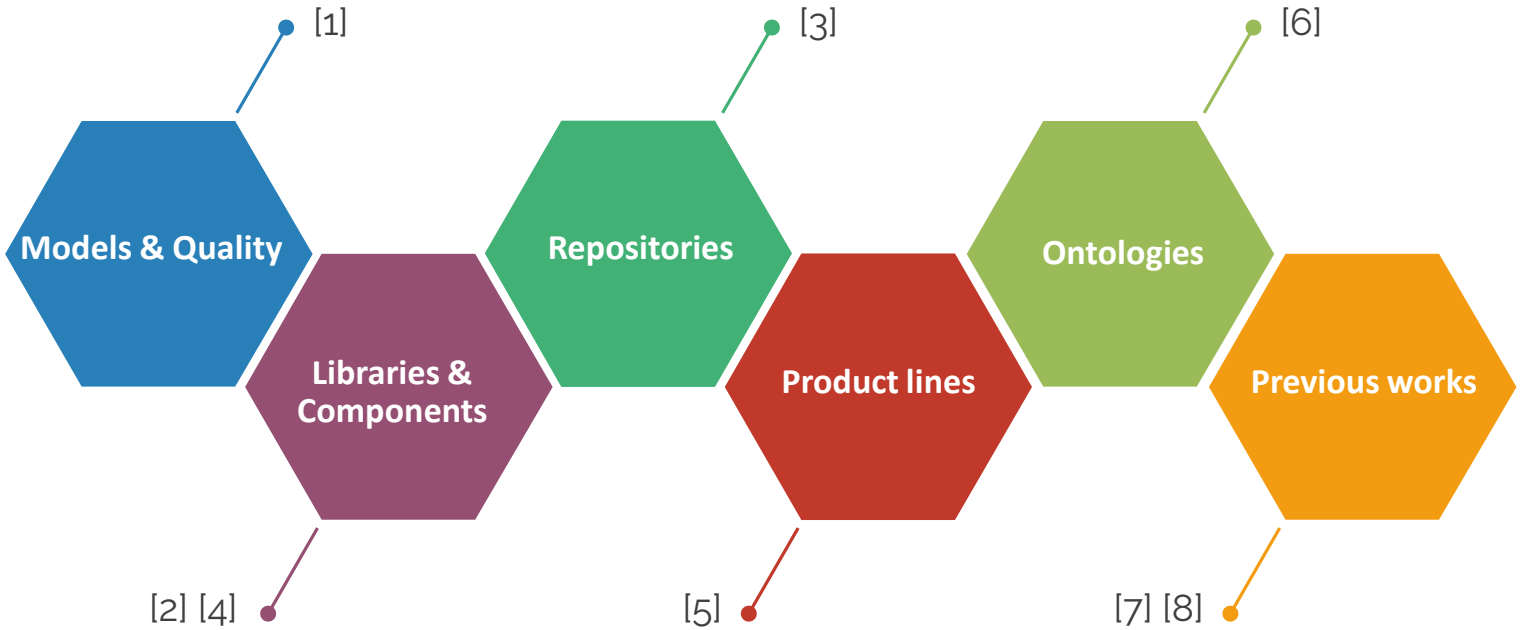


Figure 5.1: A diagrammatic representation of the different Parts of STEP, giving their names and how they are used. The thin lines designate language use, while the heavier arrows indicate a mapping realized by a translator. The one heavy line without an arrow indicates reuse of existing models.

Related work: system artefact reuse



Preliminary evaluation

OSLC/ STEP

- Some types of artefacts can not be represented (and lack of connectors for any X)
- Linked Data and RDF suits well mainly for data exchanging
- STEP is not service oriented→ making integration more difficult

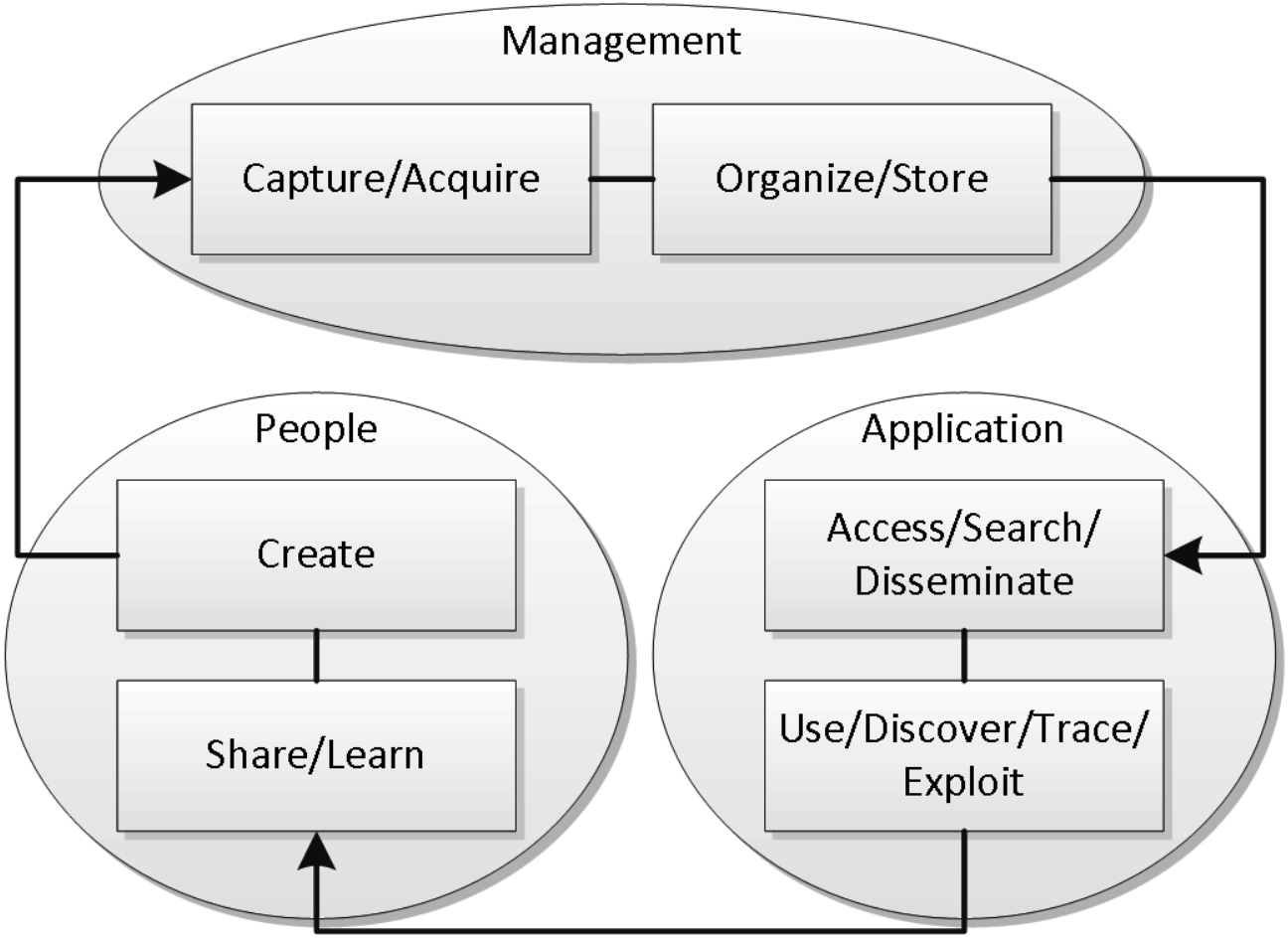
MBSE

- Not everything is a model
- Not every model is a SysML model
- Different SysML interpretations

Reuse

- Approaches focused on software artefacts (components and product lines)
- Component models and web services (operations)
- Common data models (data)

Concept: a knowledge management strategy



Concept: a *winning* strategy

Visualization
Integrated view of system artefacts.

Human interface
Query artefacts using natural language.

Automation of tasks
Support to tasks that require a whole view of the system:

-Test case description

-Change impact analysis

-Populate models

-Documentation

…

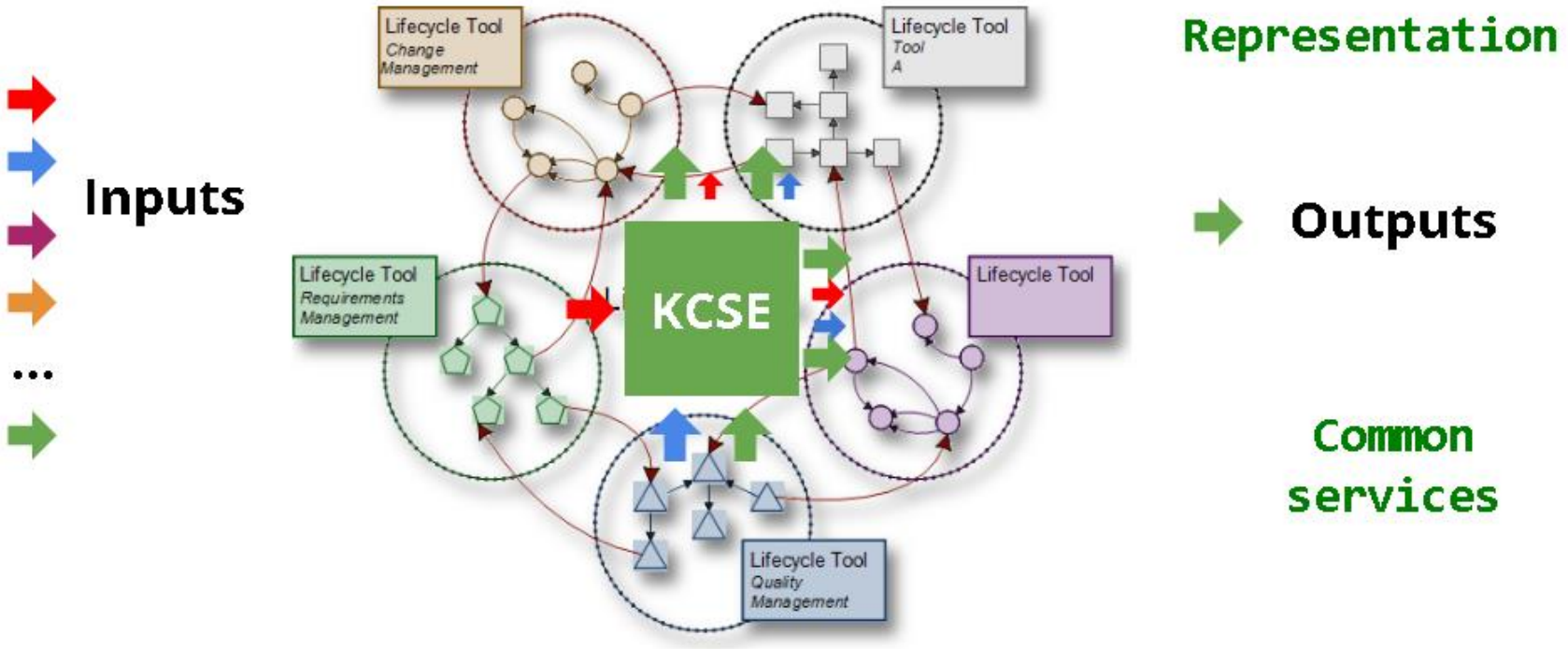
Quality
Ensure the quality of any system artefact

Language uniformity
Ensure consistency along the development lifecycle.

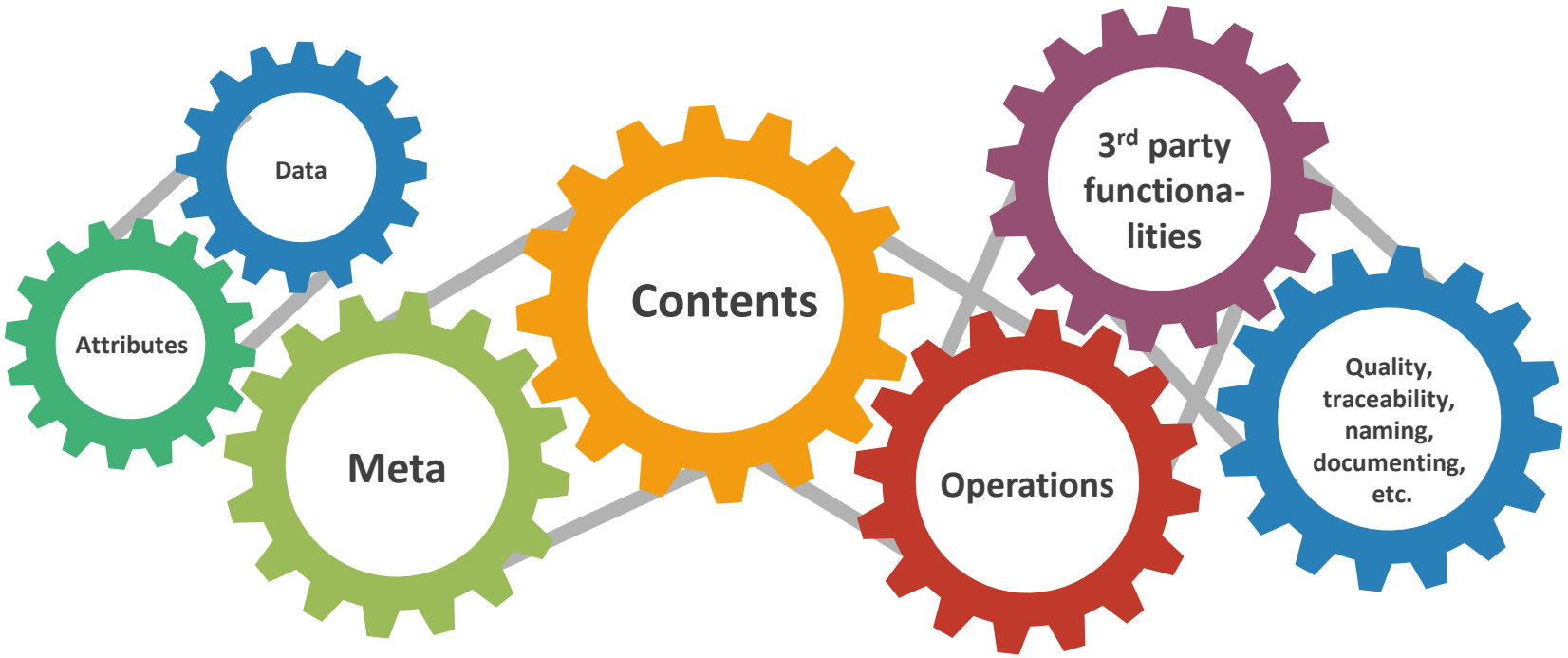
Traceability
Discover and manage links.

Concept: overview

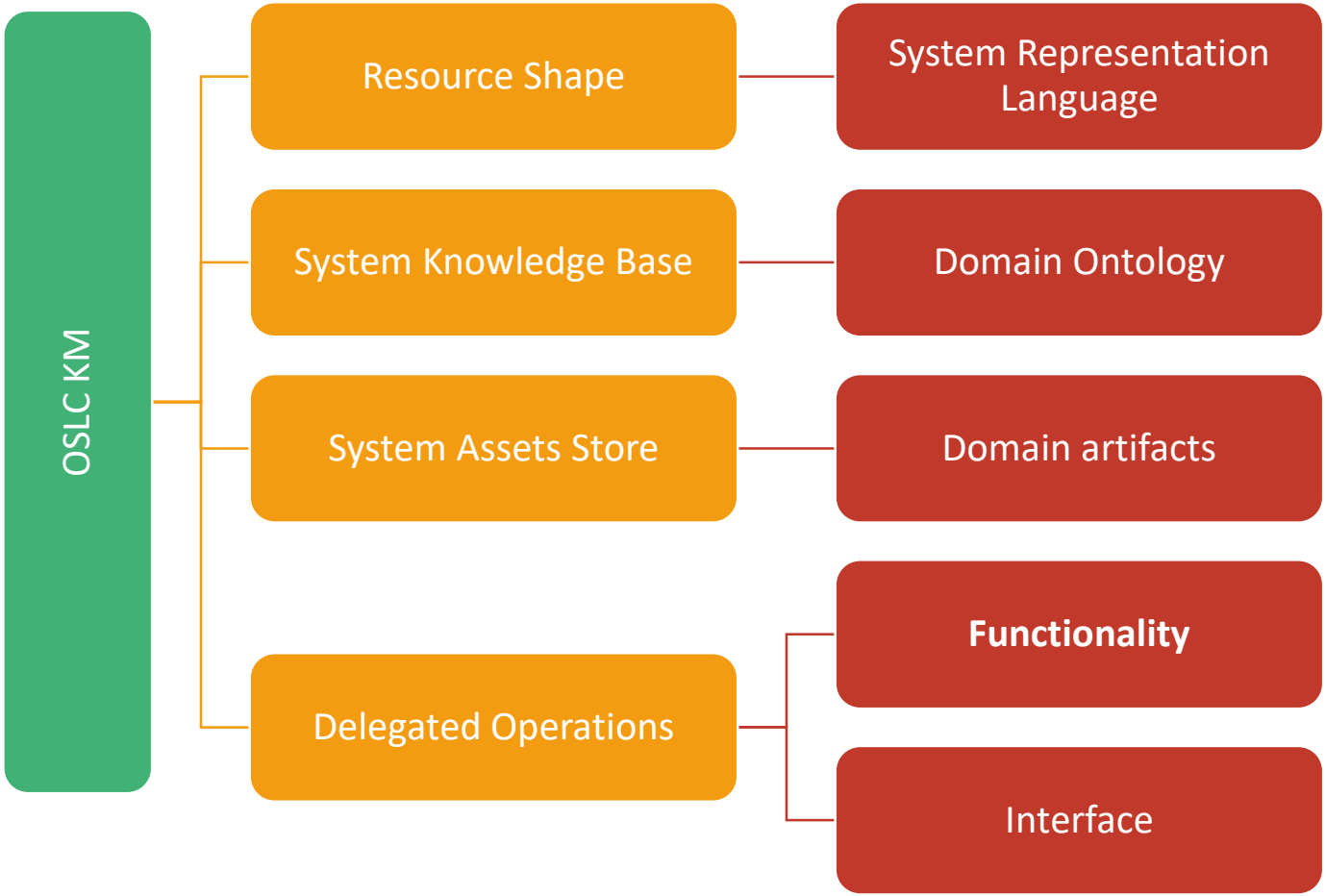
Knowledge-Centric Systems Engineering



Concept: metadata, data and operations

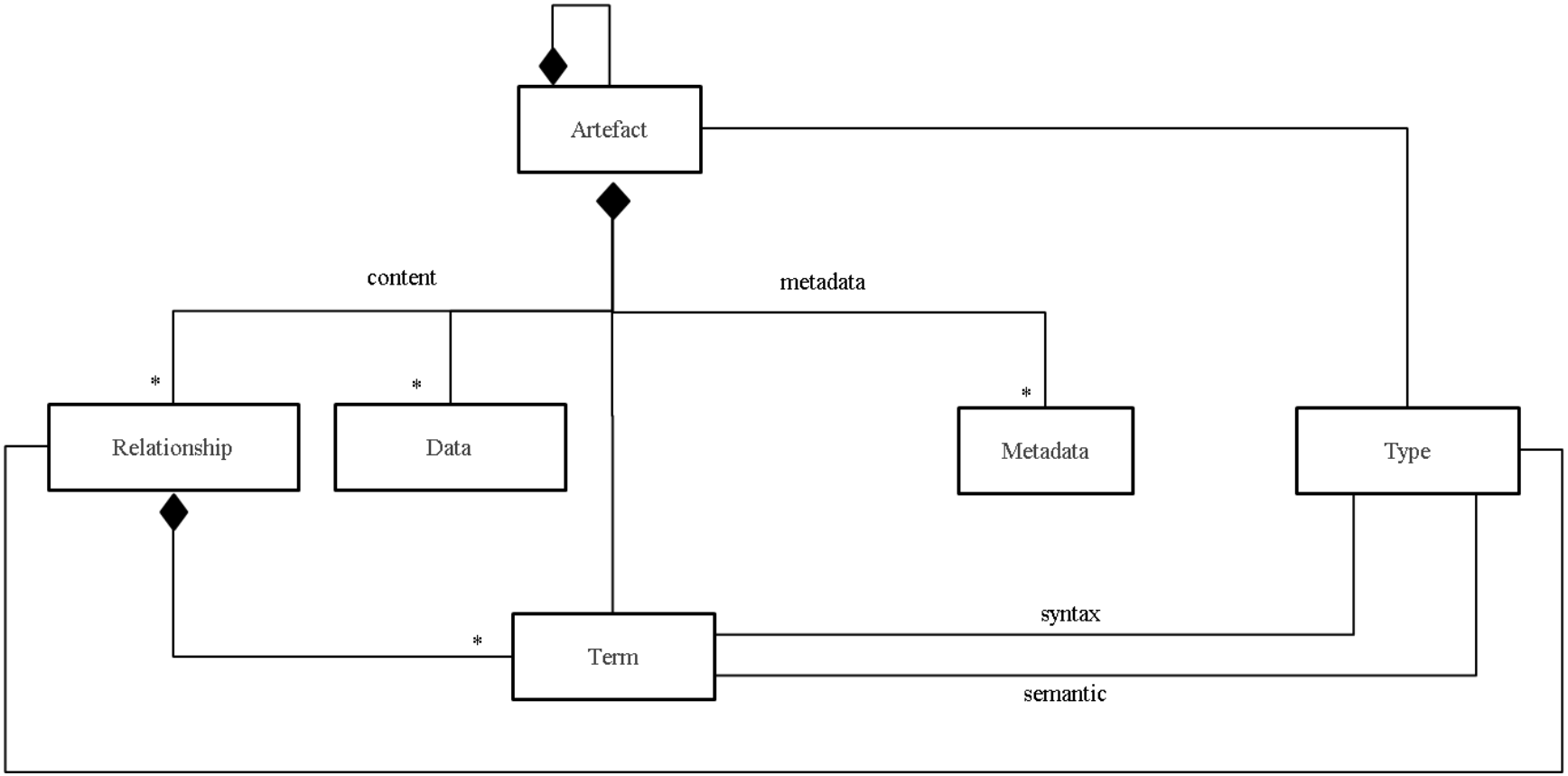


Concept: OSLC KM (*Knowledge Management*)



See specification: http://trc-research.github.io/spec/km/

OSLC KM: *System Representation Language*



OSLC KM: *Domain ontology*

Controlled vocabulary
Domain vocabulary

Taxonomy
Semantic relationships

Inference
Generation of new knowledge
Consistency
…

Patterns
Templates built on top of the domain vocabulary and semantic relationships.
E.g. requirements, design, etc.

E.g. Support smart artefact authoring (requirements)

Vocabulary

A380 | A350 | System | Operate | Temperature | Environment | Pressure

Architectures - Conceptual model

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

Environment
- Temperature
- Pressure

System
- Aircraft
  - A380
  - A350

Operation
- Operate
- Work

"Greater than (>) "

Patterns

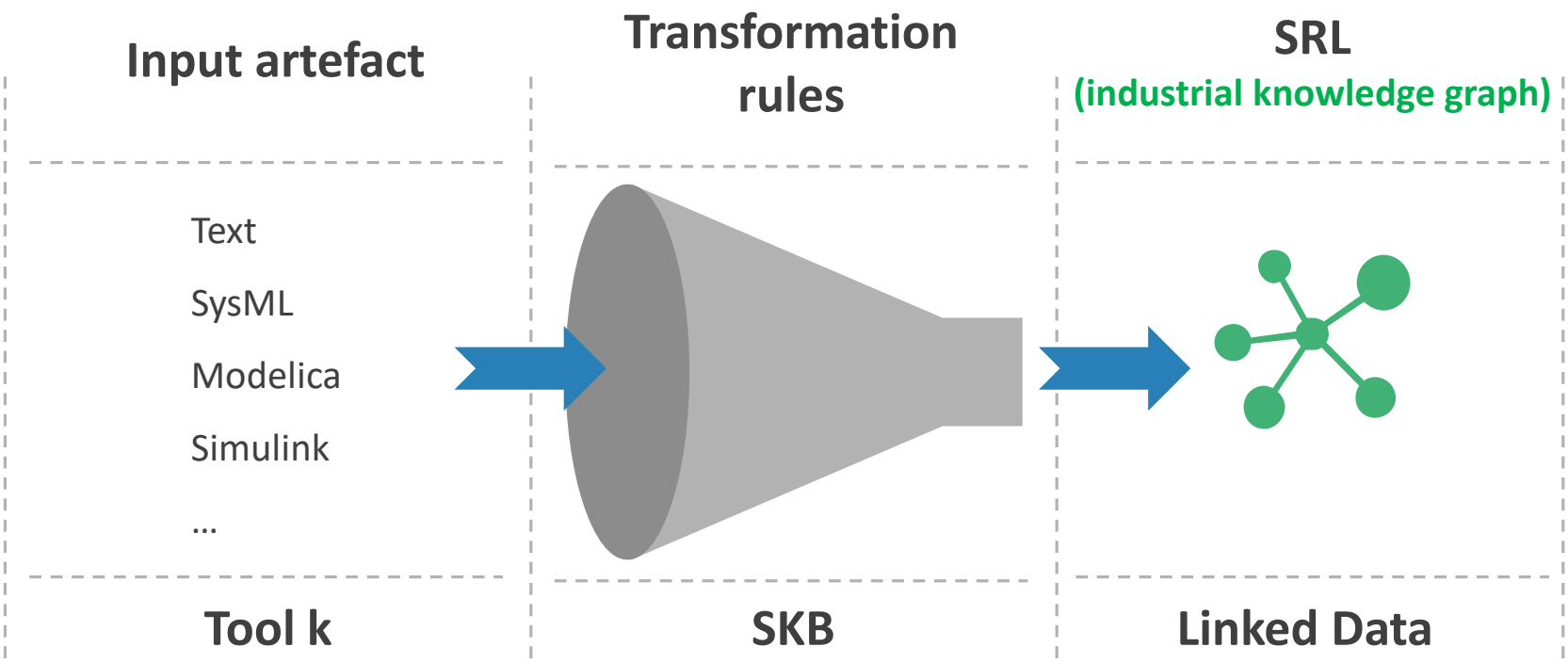
System (*) | Shall | Operation (*) | At | «Minimum» | Environment (*) | Of | NUMBER | MEASUREMENT UNIT

Formalization

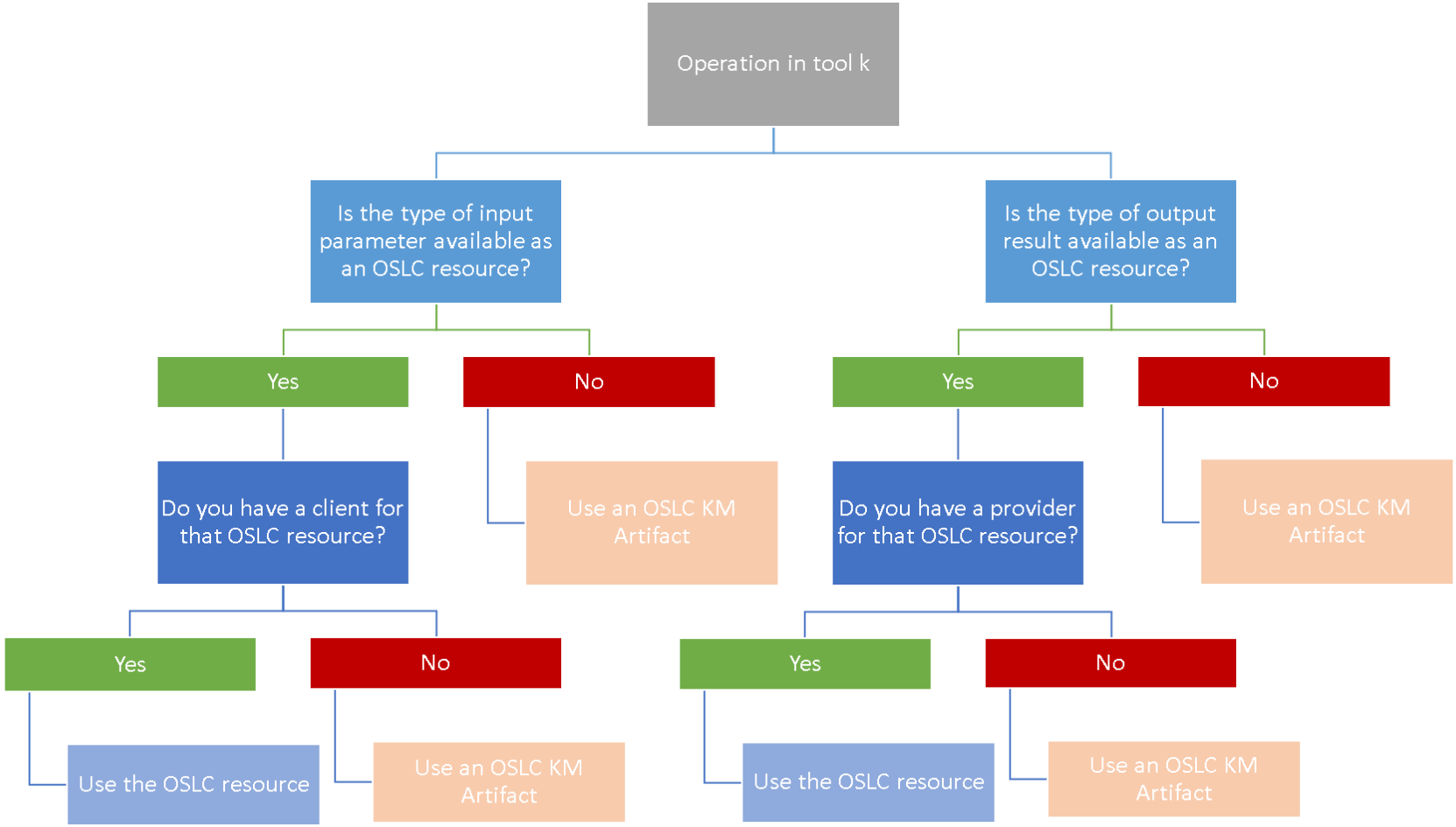
The aircraft shall be able to operate at a minimum temperature of -70º C → Temperature → "Greater than (>)" → -70 | ºC

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

OSLC KM: domain artefacts



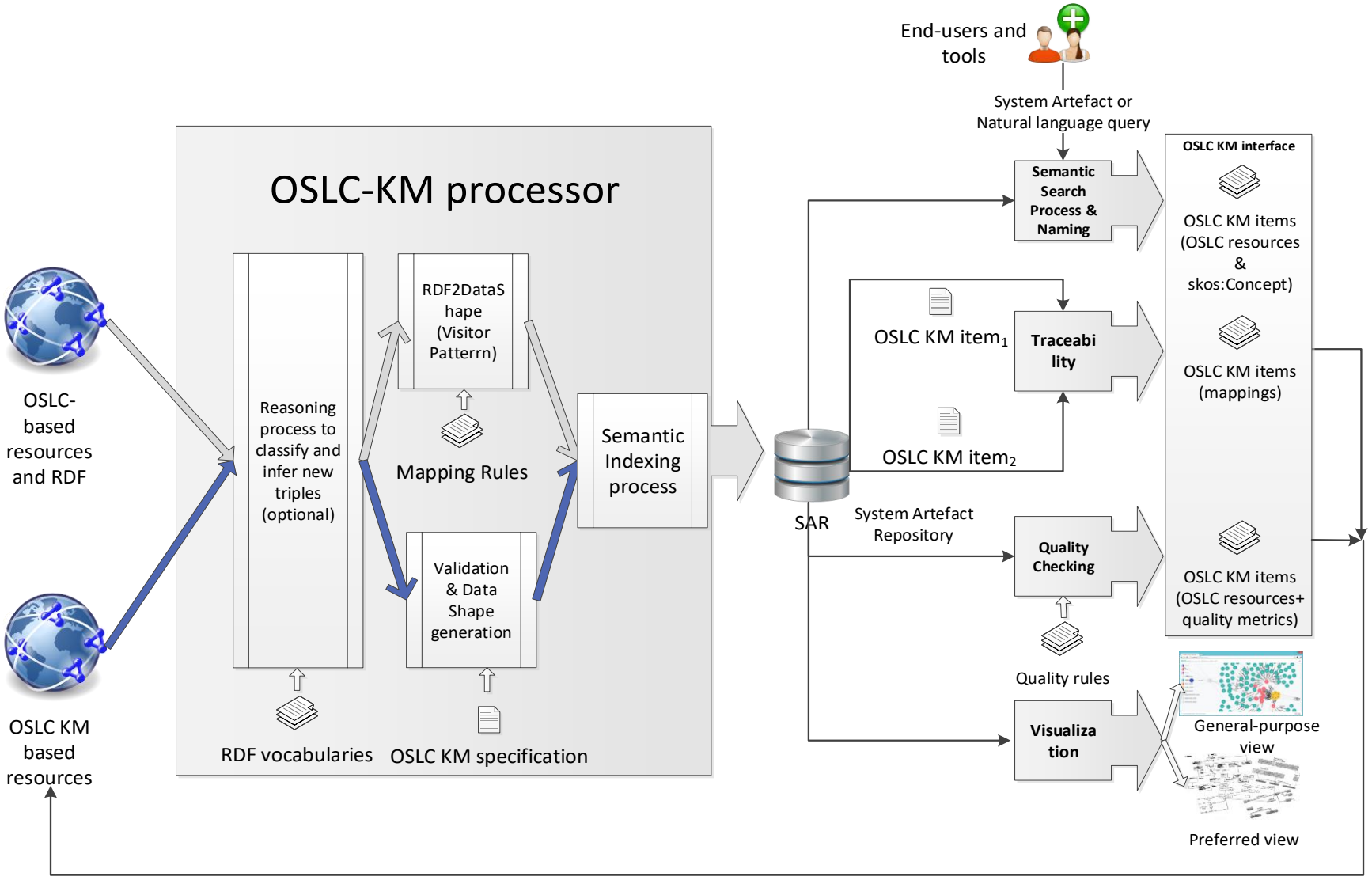
OSLC KM: delegated operation



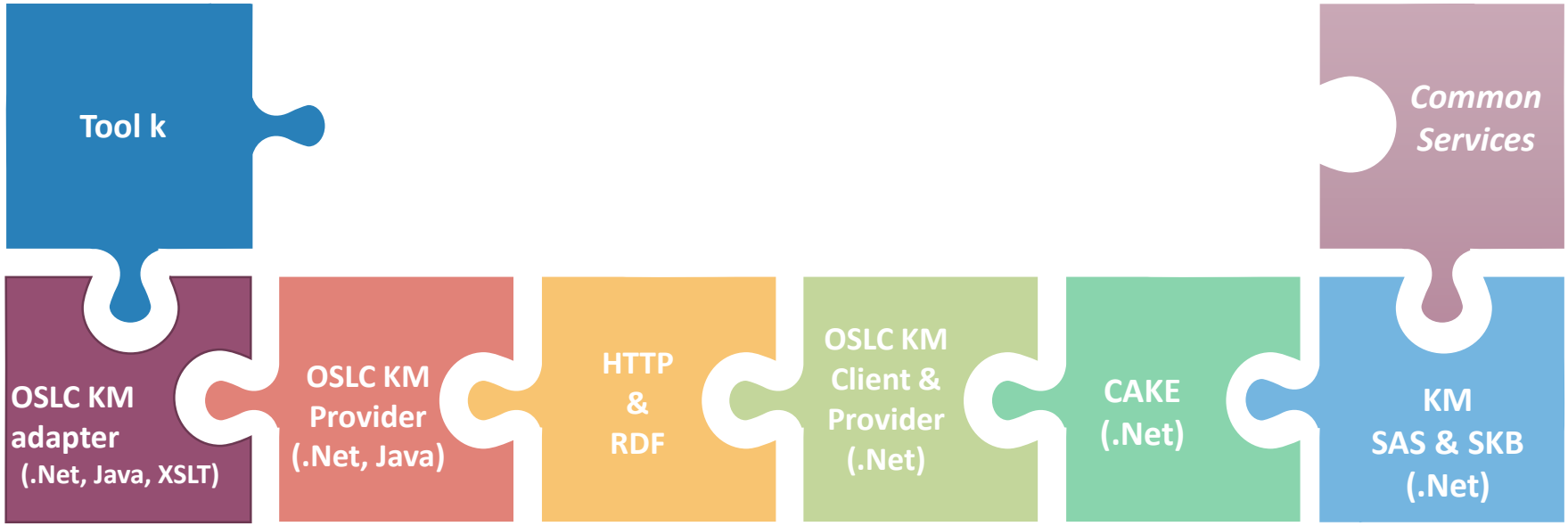
- *D6.3 Design of the AMASS tools and methods for cross/intra-domain reuse (b)*

- Mapping between WSDL and REST (and json-rpc)

OSLC KM: functional architecture

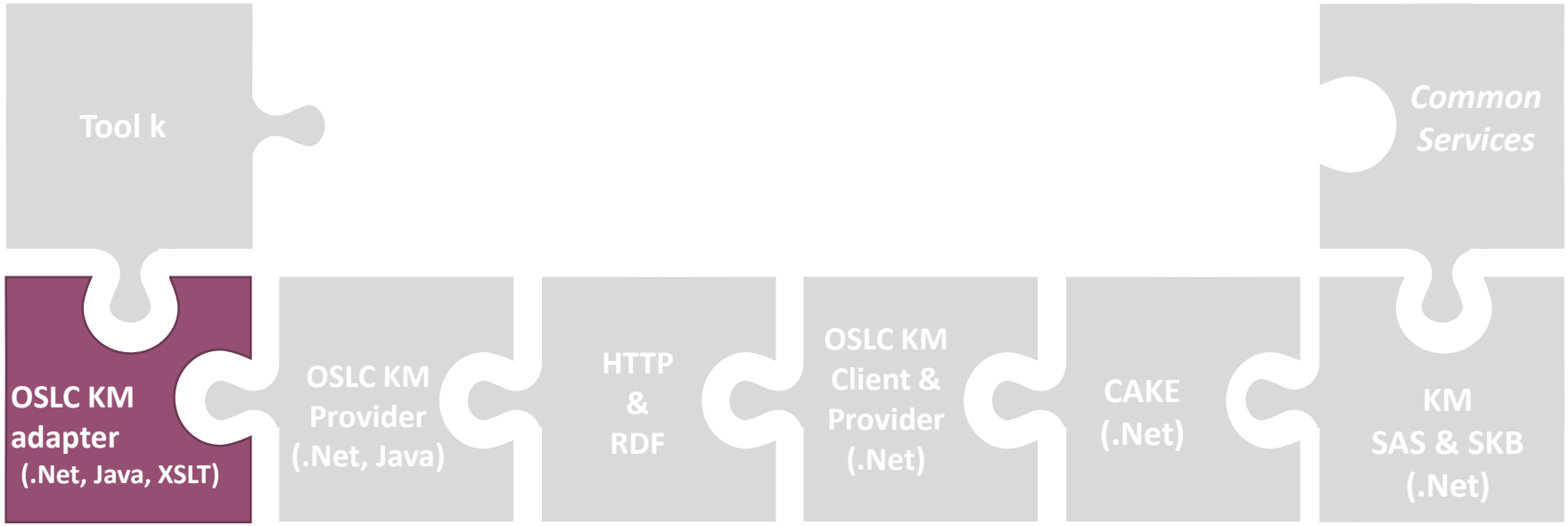


OSLC KM: technological environment



See libraries: https://github.com/trc-research/oslc-km

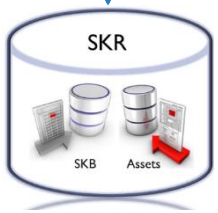
OSLC KM: technological environment

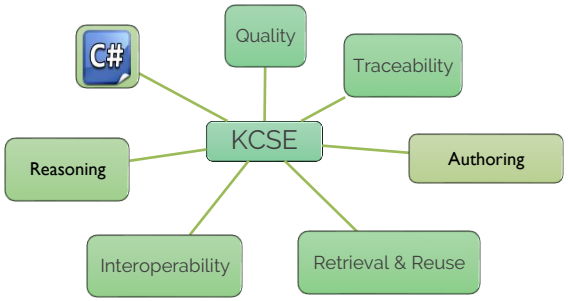


See libraries: https://github.com/trc-research/oslc-km

Implementation: A *world of knowledge* by The Reuse Company

SIM
System
Interoperability
Manager





the REUSE company

Scientific experimentation [9]

01 **Selection of tools and types of artefacts**

- Logical SysML models and two tools: Papyrus and IBM Rhapsody
- Physical models from Simulink

02 **Design of queries**

- 25 user-based quries for SyML models and 20 for Simulink models
- AMASS project

03 **Selection of performance metrics**

- Common information retrieval performance metrics:
- Precision, recall y F1 measure [10].

04 **Selection of acceptance ranges**

Based on [11]:
1) Precision > 20% acceptable, >30% good & > 50% Excellent
2) Recall: > 60% acceptable, > 70% good and > 80% Excellent

05 **Execution**

- Perform queries on top of the selected models to calculate the performance metrics

06 **Analisys of results and limitations**

- Analysis of results based on the acceptance ranges.

Data is available here: https://github.com/trc-research/oslc-km

Enabling system artefact exchange and selection through a Linked Datalayer. Jose María Álvarez-Rodríguez; Mendieta, R.; de la Vara, J. L.; Fraga, A.; and Llorens, J. UCS 24(11): 1536-1560 (2018)

Design of the experiment: user queries

Id	Query
Q1	System availability
Q2	Maximum rate of failure
Q3	Manage Traffic flow
Q4	System for purify water
Q5	System using remote control component
Q6	System use cameras
Q7	System with an statistical data component
Q8	System Performance Requirements
Q9	Requirements of System Usability
Q10	System with Simulation Component
Q11	Group Creation
Q12	System Restrictions Requirements
Q13	System that use Sensors
Q14	Gather and Interpret Information Module
Q15	Adaptive Control
Q16	Consistency in transaction
Q17	Manual Control
Q18	intruders detection
Q19	Time Validation
Q20	computer response time
Q21	System validation cards
Q22	tasks and scenarios
Q23	traffic management based in the region
Q24	semaphores automatic operation
Q25	Control standard

Logical models

Id	Consulta
Q1	A flow between a constant , product, block sum and a outport block.
Q2	A flow between an inport, product, an a block sum.
Q3	A flow between an inport, block sum and integrator.
Q4	A flow between a subsystem and outport block.
Q5	A flow between a subsystem and to Workspace block.
Q6	A flow between a Transport Delay and Subsystem block.
Q7	A flow between a Integrator block, Transport Delay and Subsystem block.
Q8	A flow between a Inport and constant blocks with a product block.
Q9	A flow between a Inport and constant blocks with a product block and the product block with outport block
Q10	A flow between a Integrator and Subsystem, Add block and subsystem and Subsystema with Subsystem
Q11	A flow between a Integrator and Subsystem, Add block and subsystem and Subsystema with Subsystem1 and subsystem2
Q12	A flow between a Integrator and Subsystem, Add block and subsystem and Subsystema with Subsystem1 and subsystem2 with to Workspace block
Q13	Model with no flows only inport block, outport block and product block
Q14	Two submodels of A flow between an inport, product, an a block sum and outport.
Q15	Two submodels of A flow between an inport, product, an a block sum and outport with two constants
Q16	A flow between inport and add block, and two inports nodes without flow
Q17	A flow between add bloc and constant with divide block.
Q18	A flow between divide block tro integrator nodes and tree outports block
Q19	A flow between integrator block and aoutport block and two outports block and one add block with no flows
Q20	A flow between 4 transfer delay with two subsystems.

Physical models

Design of the experiment: performance metrics

- **Precision**: fraction of relevant models among the retrieved models.
  - Value [0-1]

$$(P)recision = \frac{|\{relevant\ models\} \cap \{retrieved\ models\}|}{|\{retrieved\ models\}|}$$

- ***Recall***: fraction of relevant models that have been retrieved over the total amount of relevant models.
  - Value [0-1]

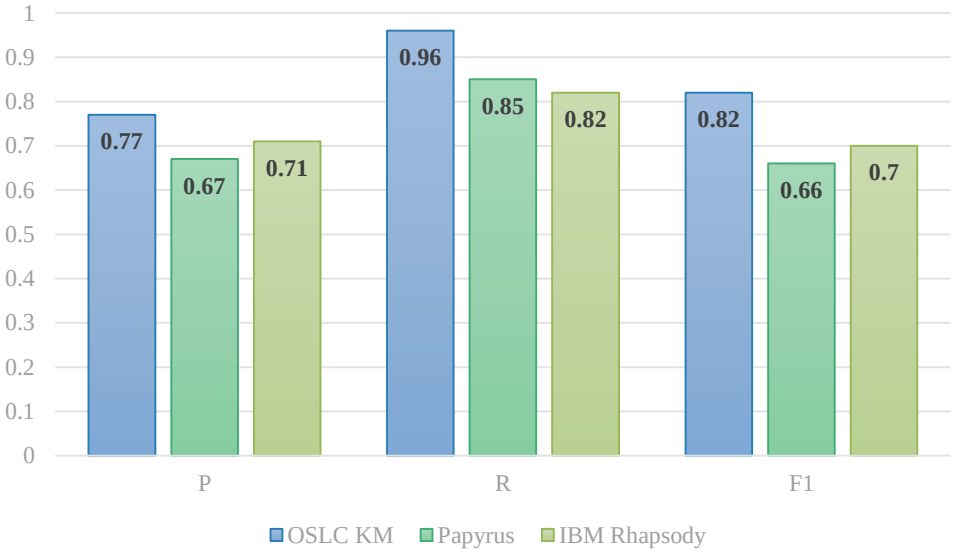
$$(R)ecall = \frac{|\{relevant\ models\} \cap \{retrieved\ models\}|}{|\{relevant\ models\}|}$$

- ***F1-measure***: harmonic mean of precision and recall.
  - Value [0-1]

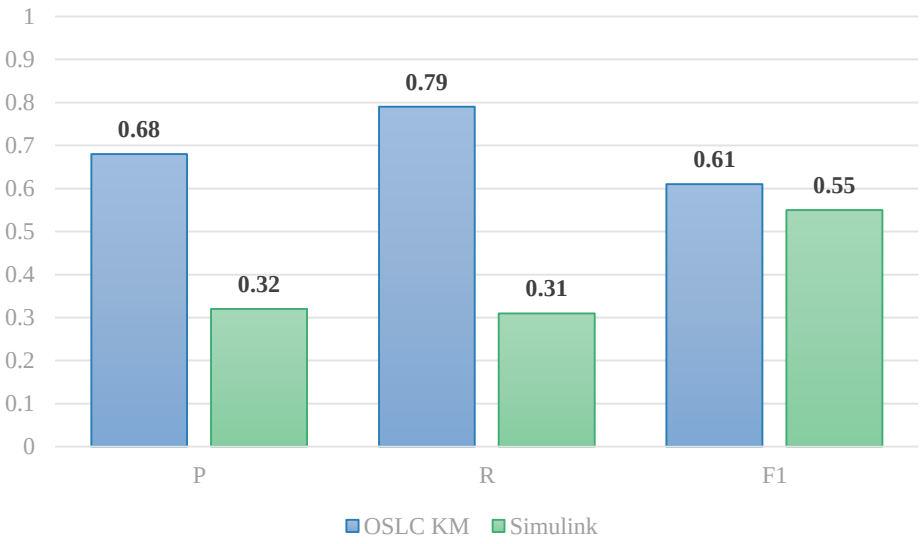
$$F1 = 2 * \frac{P * R}{P + R}$$

Analysis: aggregated values

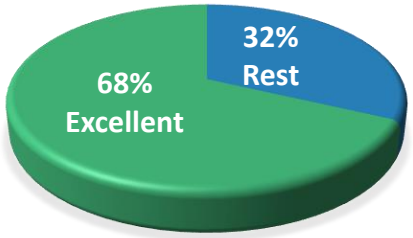
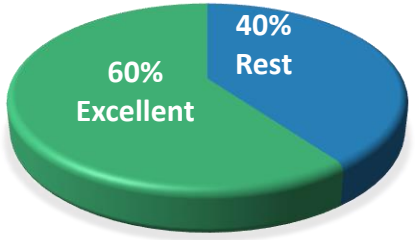
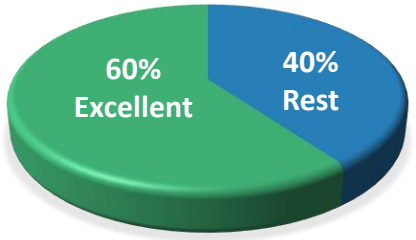
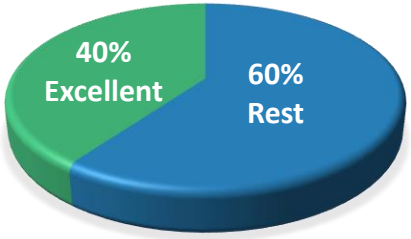
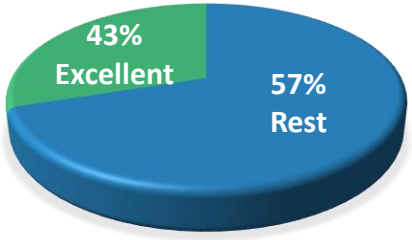
Logical models-SysML



Physical models-Simulink



Analysis of results: OSLC KM

	Logical models-SysML	Physical models-Simulink
Precision		
Recall		
F1		

Scientific experimentation: limitations

Data

- User queries are restricted to the AMASS use cases.
- Models are restricted to the AMASS use cases and those part of the common libraries.
- Only two types of models are considered

Process

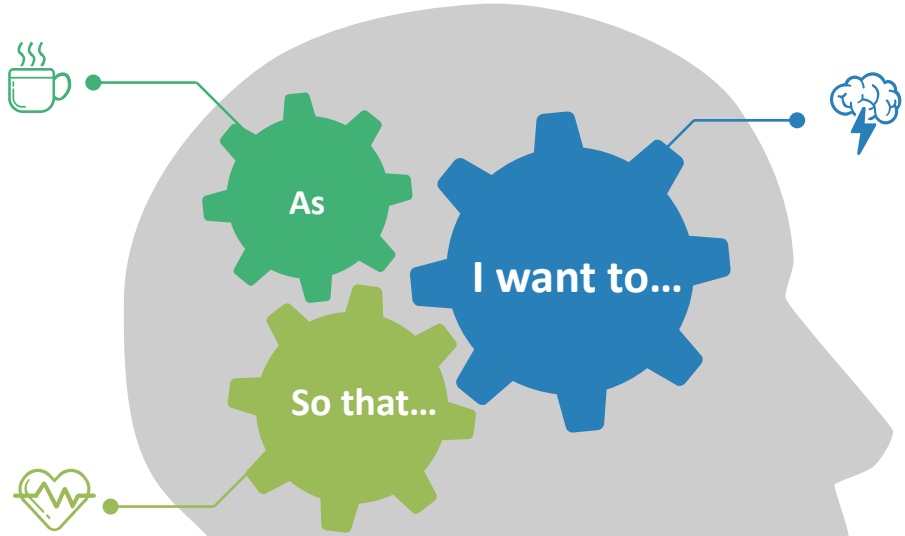
- Continuous calculation and improvement of the performance metrics, create a kind of "*benchmark*".
- Measure the impact of quality in the degree of reuse.

Analysis

- Robustness analysis to measure the impact of the different representations of the same data.

User story *: Extract information from legacy documents

Requirements Engineer



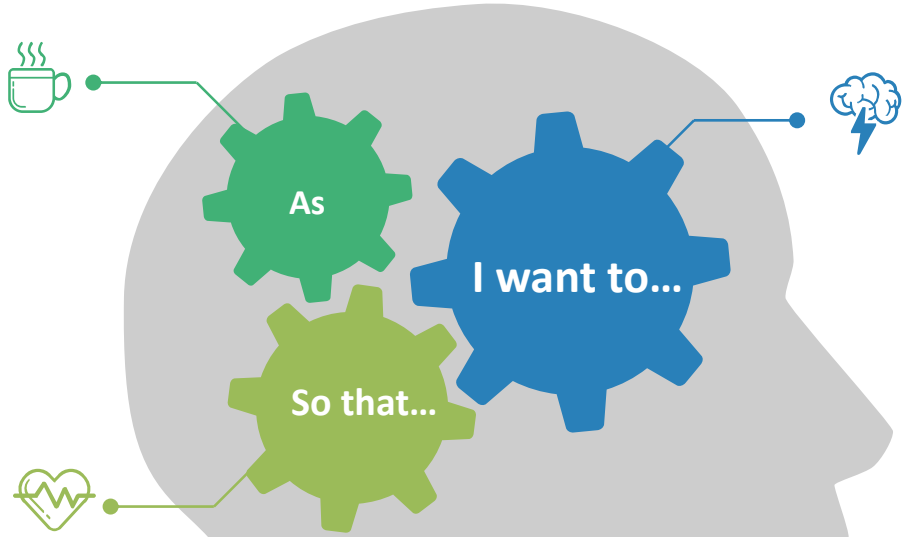
Import requirements

Already available in Word/PDF documents

Check quality, find similar requirements, recovery traces, etc.

User story I: Reuse (and find similar) logical & physical models

Domain Engineer



Access and search logical and physical models

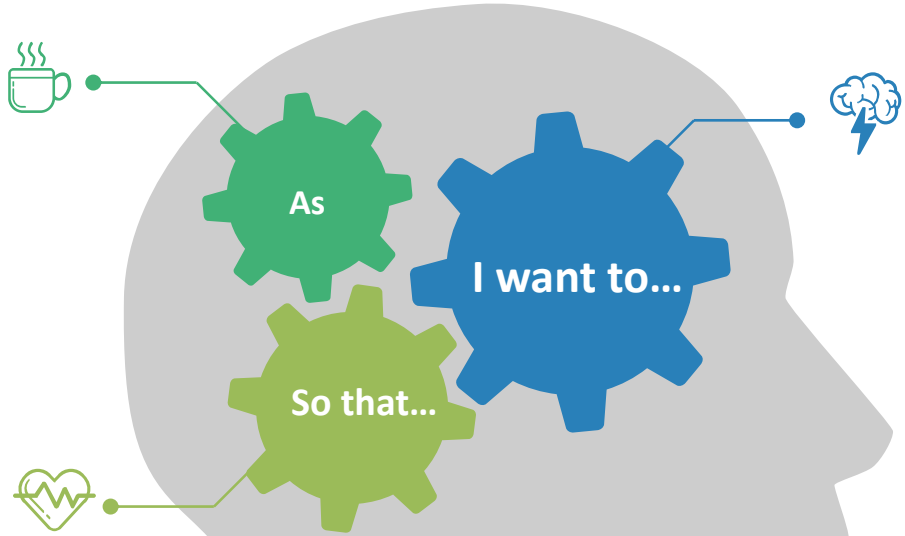
I am using different tools: Modelica, Papyrus, IBM Rhapsody and Magic Draw

Reuse of existing system artefacts

-Recovery traces

User story II: Check quality of logical models

Quality/Domain Engineer



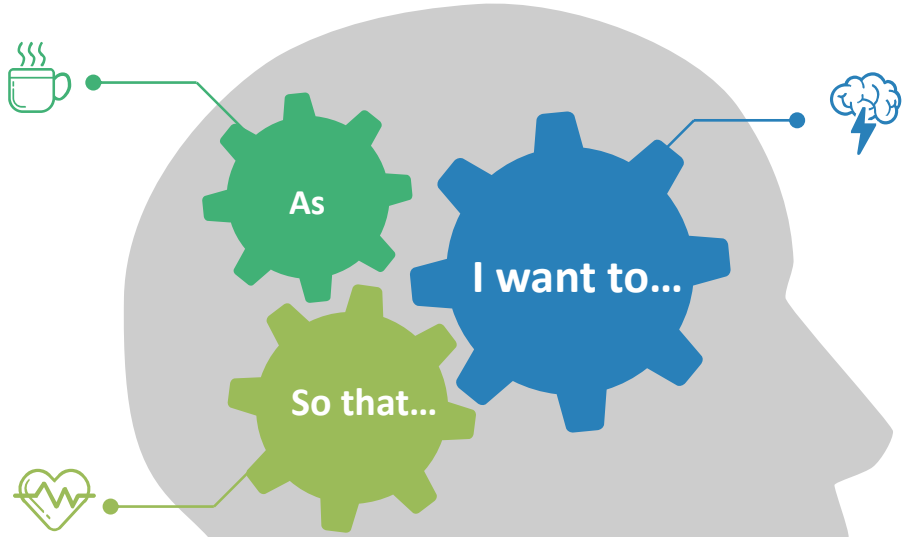
Ensure that everything is CCC

Check the quality of my models

I am using different tools: Modelica, Papyrus, IBM Rhapsody and Magic Draw

User story III: Generate documentation

Domain Engineer



Report documentation

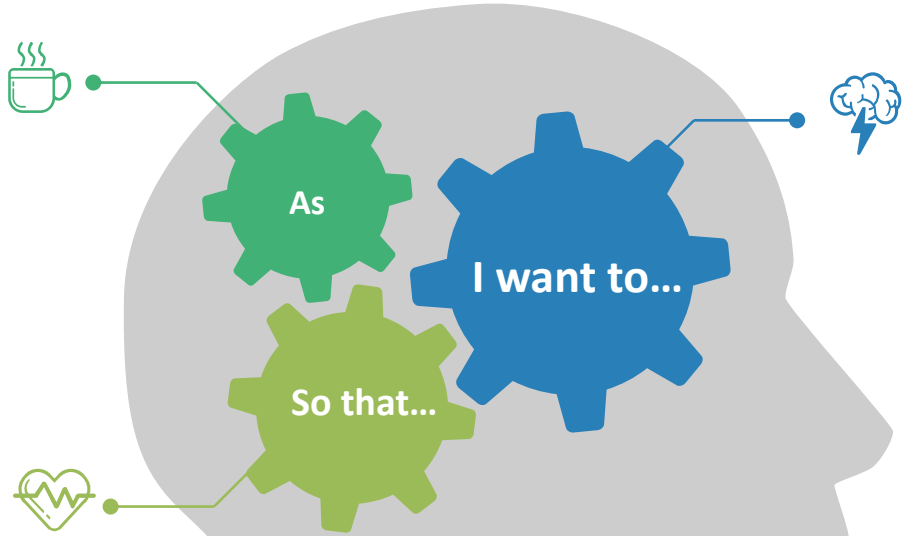
Reuse of my system artefacts

Create consistent and up-to-date documentation

Automation

User story IV: Populate models from Simulink (e.g. an ontology)

Domain Engineer

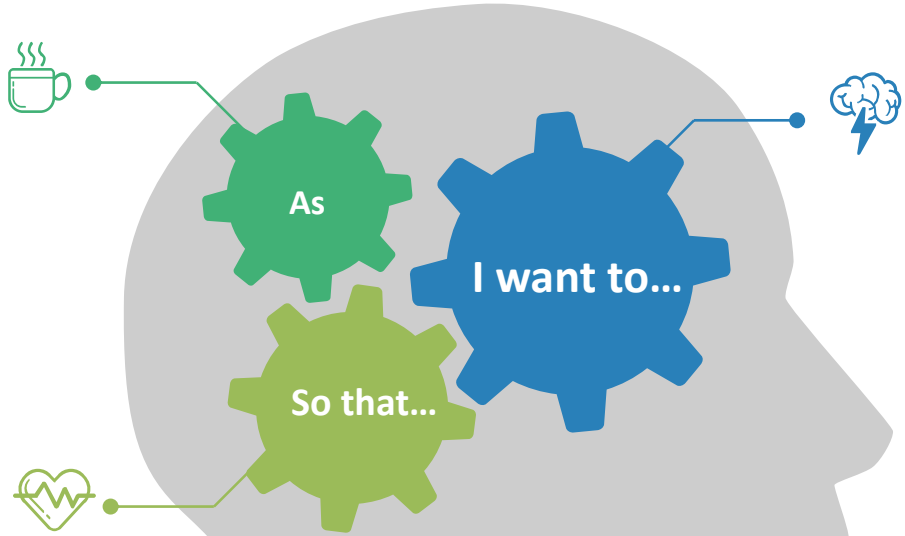


Reuse my physical models to populate an ontology

Create my web ontology based on OWL

User story V: Populate logical models from requirements

Systems Engineer

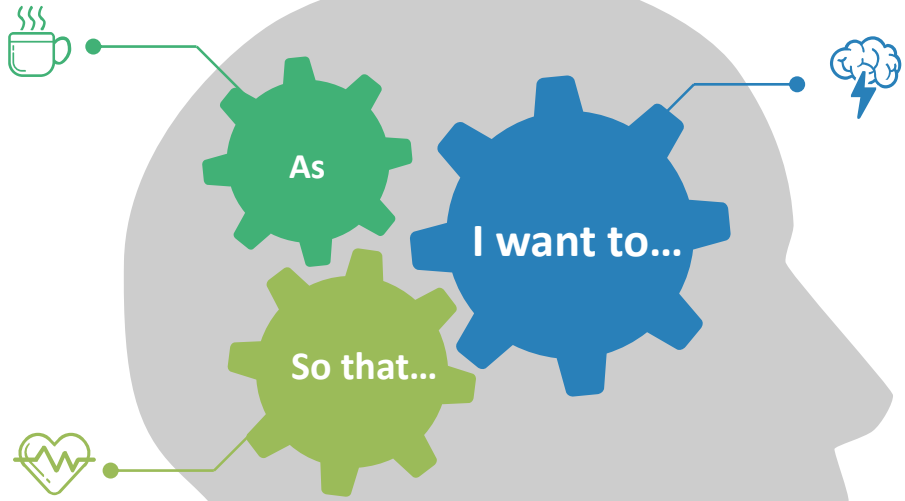


Reuse my requirements to populate an ontology

Avoid to start from scratch the logical modelling

User story VI: Consistency between descriptive and analytical models

Systems Engineer



Ensure consistency between models

I do not need to re-work between models.

Conclusions and Future work

Data exchange

-OSLC and Linked Data suits well for data exchange.
-Define methodology to reuse vocabularies, etc.

Coverage

-Increase the number of tools that are supported.
-API-economy: OSLC & SWAGGER

Represen-tation

SRL is a **language and a model repository** to ease the reuse of existing data and operations.

Experiment & User stories

-Extend the existing experiments and user stories.
-Take advantage of **the industrial knowledge graph**.

Reuse

Existing tools should **improve** its support to **interoperability** mechanisms in both : data and operations.

OSLC KM

-Release new versions of the source code.
-Reach a higher TRL (8-9)
-Promote the approach to OASIS OSLC

Acknowledgements

AMASS
Assurance and Certification of CPS

The research leading to these results has received funding from the AMASS project (H2020-ECSEL grant agreement no 692474; Spain's MINECO ref. PCIN-2015-262) and the CRYSTAL project (ARTEMIS FP7-CRitical sYSTem engineering AcceLeration project no 332830-CRYSTAL and the Spanish Ministry of Industry).

Learn more: https://www.amass-ecsel.eu/

References

1. W. Frakes and C. Terry, “Software reuse: metrics and models,” ACM Comput. Surv. CSUR, vol. 28, no. 2, pp. 415–435, 1996.
2. A. Mili, R. Mili, and R. T. Mittermeir, “A survey of software reuse libraries,” Ann. Softw. Eng., vol. 5, pp. 349–414, 1998.
3. J. Guo and others, “A survey of software reuse repositories,” in Engineering of Computer-Based Systems, IEEE International Conference on the, 2000, pp. 92–92.
4. R. Land, D. Sundmark, F. Lüders, I. Krasteva, and A. Causevic, “Reuse with software components-a survey of industrial state of practice,” in Formal Foundations of Reuse and Domain Engineering, Springer, 2009, pp. 150–159.
5. T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, “A Classification and Survey of Analysis Strategies for Software Product Lines,” ACM Comput. Surv., vol. 47, no. 1, pp. 1–45, Jun. 2014.
6. V. Castañeda, L. Ballejos, L. Caliusco, and R. Galli, “The Use of Ontologies in Requirements Engineering,” GJRE, vol. 10, no. 6, 2010.
7. R. Mendieta, J. L. de la Vara, J. Llorens, and J. M. Alvarez-Rodríguez, “Towards Effective SysML Model Reuse,” in Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD, 2017, pp. 536–541.
8. Elena Gallego, J. M. Alvarez-Rodríguez and J. Llorens, “Reuse of Physical System Models by means of Semantic Knowledge Representation: A Case Study applied to Modelica,” in Proceedings of the 11th International Modelica Conference 2015, 2015, vol. 1.
9. N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*, vol. 5/6. Springer Science & Business Media, 2001.
10. W. B. Croft, D. Metzler, and T. Strohman, Search Engines: Information Retrieval in Practice. Pearson Education, 2010.
11. J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, “Improving after-the-fact tracing and mapping: Supporting software quality predictions,” IEEE Softw., vol. 22, pp. 30–37, 2005.