AIRBUS

CoCoVaD

Requirements' Early V&V

McMaster University -- 2023/06/19

Jean-Michel Bruel

https://bit.ly/jmbruel

@SmartModelTeam

https://github.com/smart-researchteam

UNIVERSITÉ TOULOUSE Jean Jaurès

SM@RT

IRIT

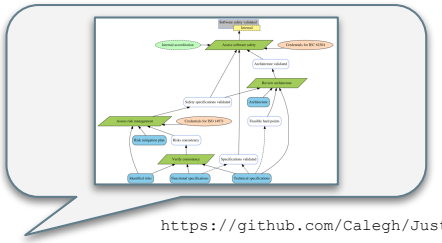
https://www.linkedin.com/posts/daniel-abrahams_reminder-people-dont-buy-products-they-ugcPost-7010015948820680704-CTJD?utm_source=share&utm_medium=member_android

People don’t buy products

They buy solutions to their problem

[...] they buy solutions to their problem

- **Play** with the product
  - Not so easy with an airplane...
- Don't need details
  - **Early** V&V
- Validation => **Rational**



https://github.com/Calegh/JustificationDiagram

Outline

- Concrete examples
- Context: the “CoCoVaD Airbus chair”
- Formal Requirements
- Requirements concepts Ontology
- Business Analysis

Outline

- Concrete examples
- Context: the “CoCoVaD Airbus chair”
- Formal Requirements
- Requirements concepts Ontology
- Business Analysis

What do we mean by “early V&V”?

- “What-if scenarios”
  - What are the consequences of being able now to cross Atlantic with 2 engines instead of 4?
  - Is my early design compatible with manufacturing or operations?
- Requirement mining
  - What are the requirements that have quantities in their description?
- Contextualization of requirements
  - Section titles
  - Illustrations and details

From text to formal specification: a too **big step**

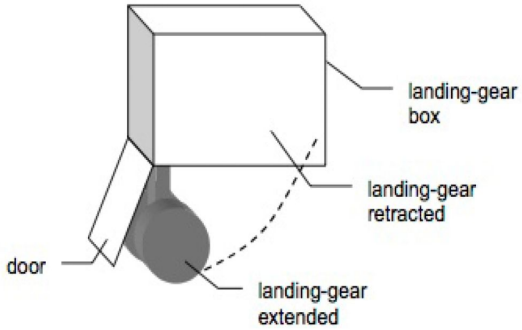


Fig. 1. Landing set

The system is controlled digitally in nominal mode and analogically in emergency mode. In this case study, we do not consider the emergency mode. However, in order to allow the pilot to activate the emergency command, the system has to elaborate health parameters for all the equipments involved in the landing gear function. This health monitoring part is in the scope of the case study.

In nominal mode, the landing sequence is: open the doors of the landing gear boxes, extend the landing gears and close the doors. This sequence is illustrated

Spec

MACHINE Landing_System

...

EVENTS

Event extend_gears

Event retract_gears

Event open_doors

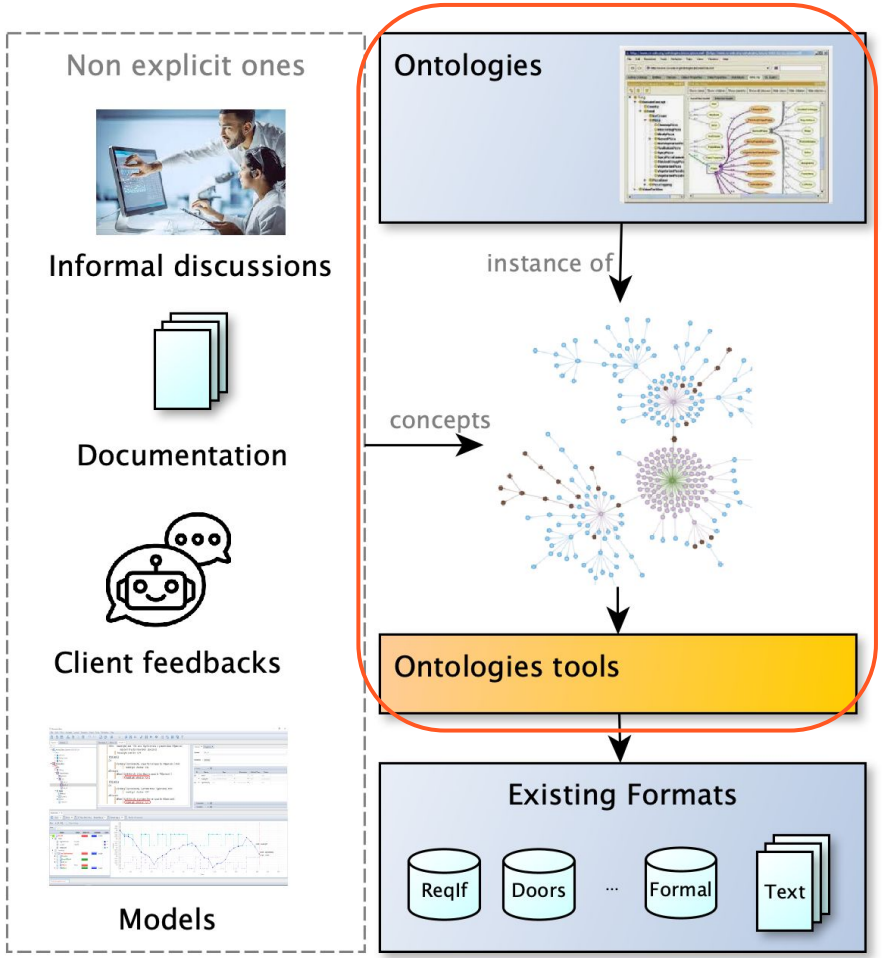
Event close_doors

...

END

Intermediate representation

- “Single source of truth”
- “Data lake”
- “Conceptual Models”



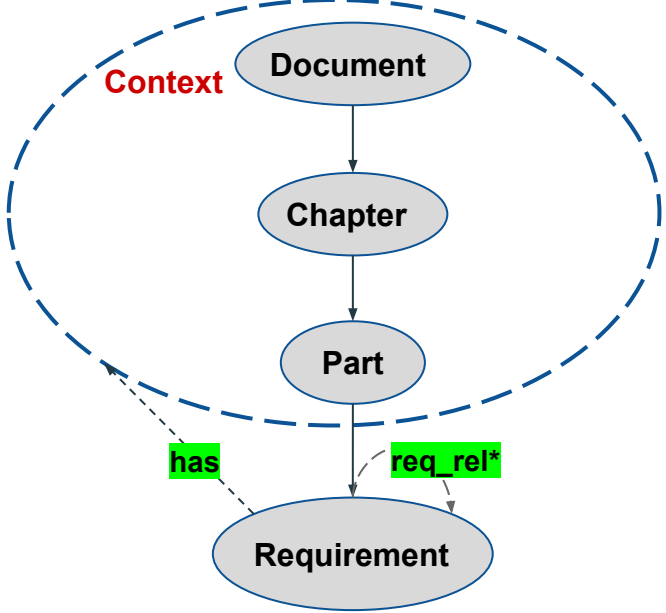
Example: Informal Requirements (LGS)

- **R1:** In nominal mode, the landing sequence is: open the doors of the landing gear boxes, extend the landing gears and close the doors.

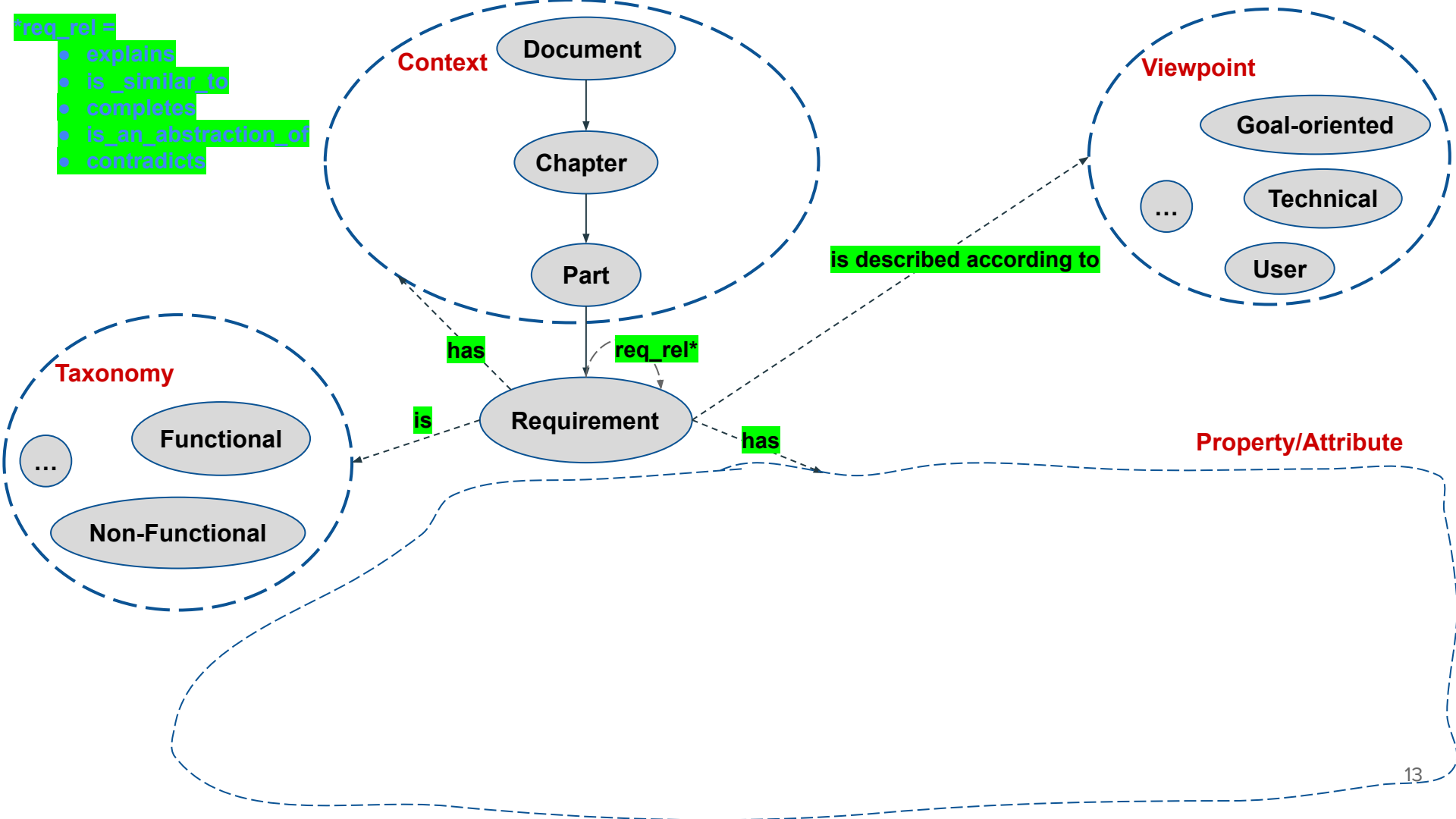
- **R2:** The landing system is in charge of maneuvering landing gears and associated doors. The landing system is composed of 3 landing sets: front, left and right. Each landing set contains a door, a landing-gear and associated hydraulic cylinders.

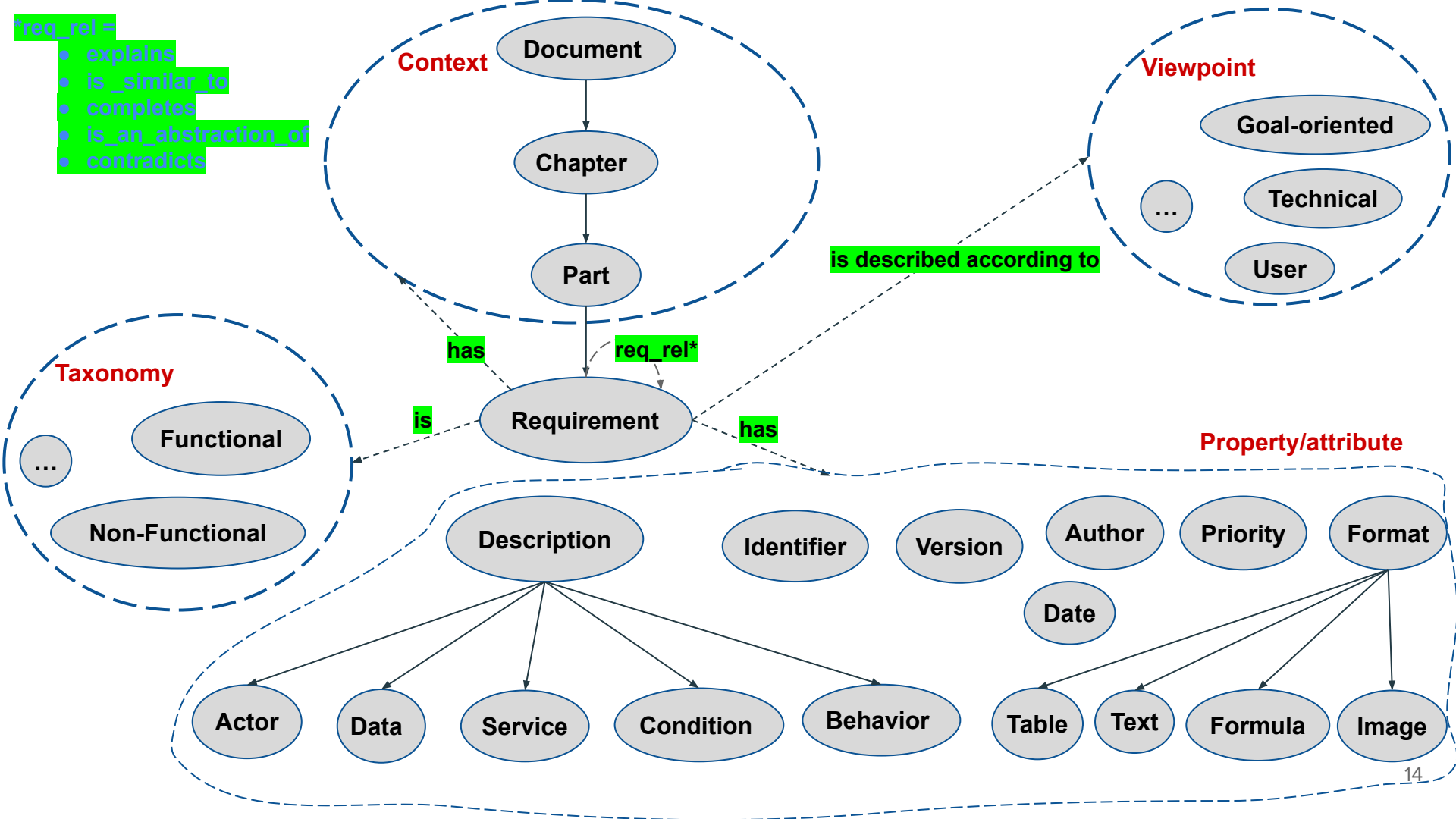
*req_rel =

- explains
- is_similar_to
- completes
- is_an_abstraction_of
- contradicts



- **R1:** In nominal mode, the landing sequence is: open the doors of the landing gear boxes, extend the landing gears and close the doors.
- **R2:** The landing system is in charge of maneuvering landing gears and associated doors. The landing system is composed of 3 landing sets: front, left and right. Each landing set contains a door, a landing-gear and associated hydraulic cylinders.


*req_rel =
explains
is_similar_to
completes
is_an_abstraction_of
contradicts
Context
Document
Chapter
Part
Viewpoint
Goal-oriented
Technical
User
...
is described according to
has
req_rel*
Taxonomy
Functional
Non-Functional
...
is
Requirement
has
Property/Attribute


*req_rel =
explains
is_similar_to
completes
is_an_abstraction_of
contradicts
Context
Document
Chapter
Part
Viewpoint
Goal-oriented
Technical
User
...
is described according to
has
req_rel*
Taxonomy
Functional
Non-Functional
...
is
Requirement
has
Property/attribute
Description
Identifier
Version
Author
Priority
Format
Date
Actor
Data
Service
Condition
Behavior
Table
Text
Formula
Image

Outline

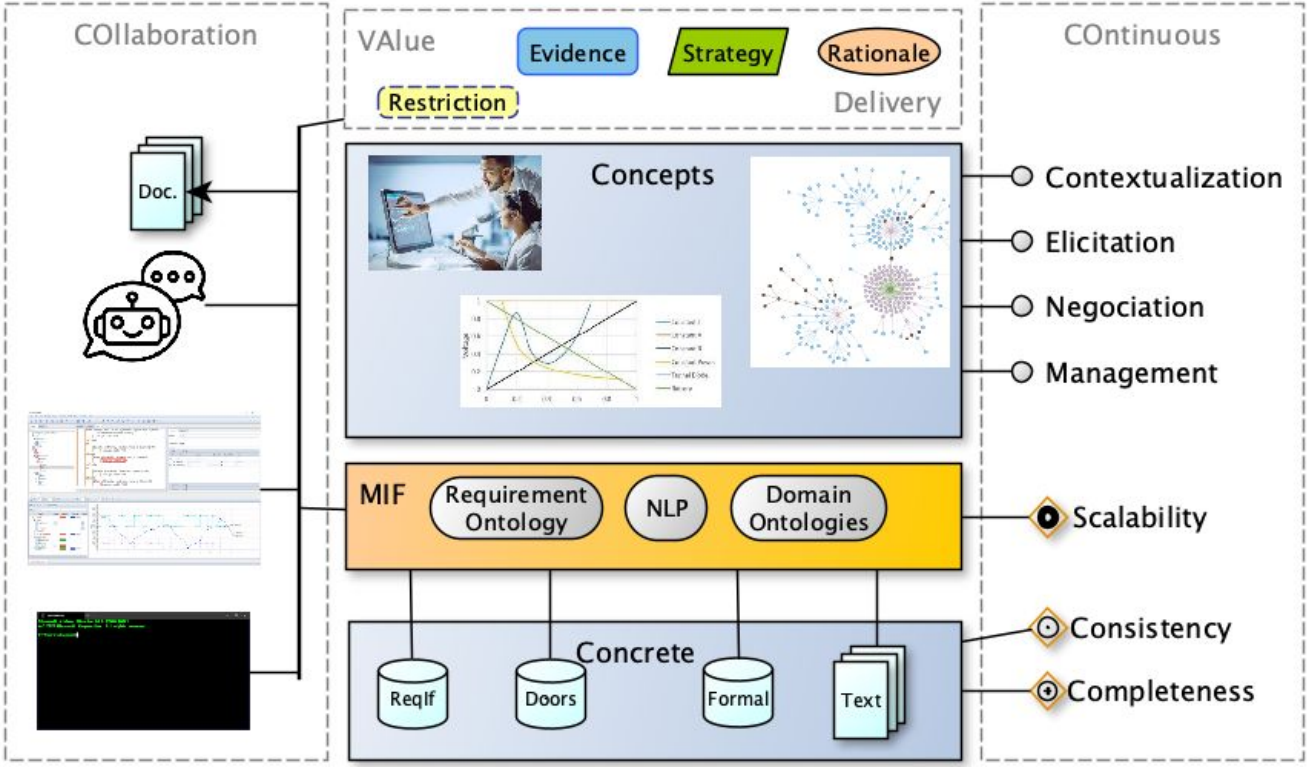
- Concrete examples
- Context: the “CoCoVaD Airbus Chair”
- Formal Requirements
- Requirements concepts Ontology
- Business Analysis

IRIT effort...

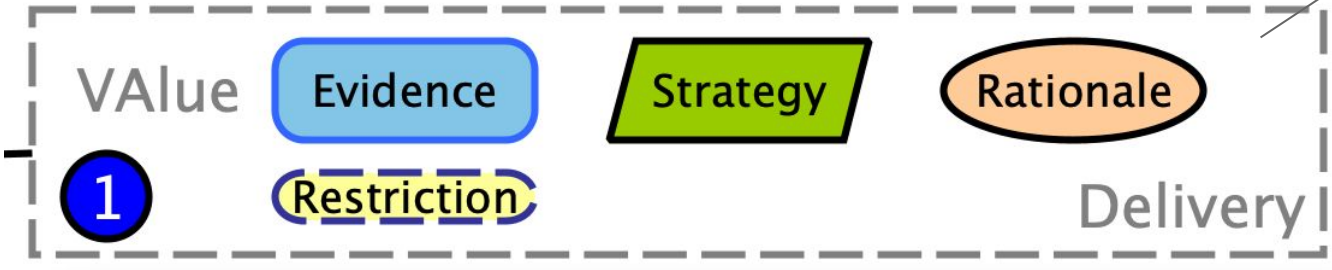
- IRIT/SM@RT team
  - Florian 
  - Sophie 
  - JMB 

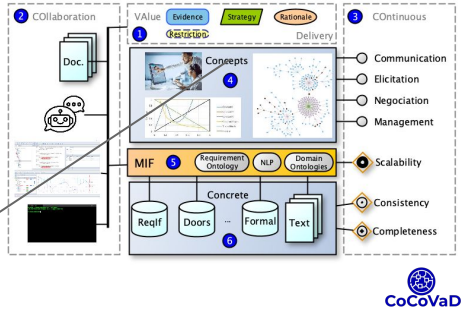
- CoCoVaD
  - Imen 
  - Thuy 
  - Mrunmayi 
  - Nathalie 
  - Marc 

Requirements as first-class citizens



Value Delivery

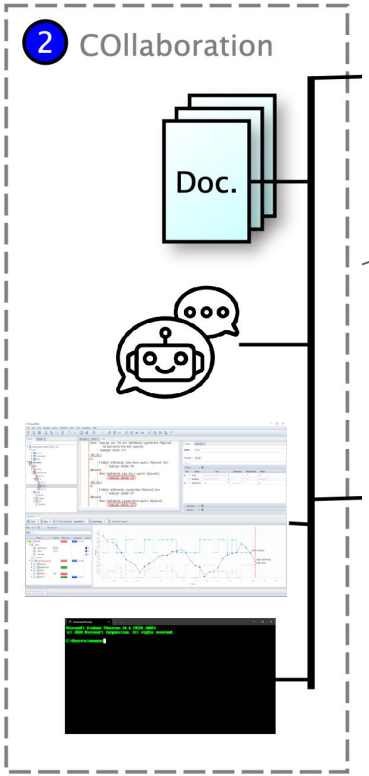




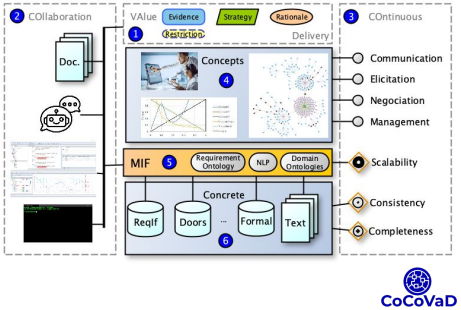
- **Who** are the clients
- **Why** do they need this product

Not the primary target of our research, but crucial CoCoVaD

Collaborative effort



- Different **Viewpoints**
- Different **Knowledge**
- Different **Purposes**
- **Collaborative** effort

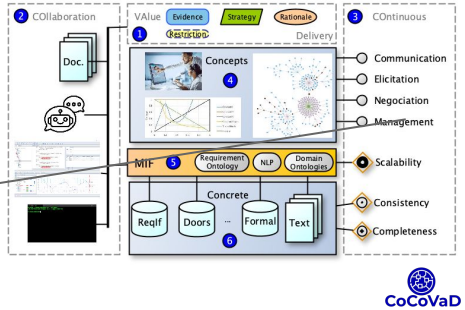


Ph.D. #1

- Model Alignment
- DSLs
- Abstraction/Ontologies

CoCoVaD

Continuous effort



3 COntinuous

- Communication
- Elicitation
- Negociation
- Management
- Scalability
- Consistency
- Completeness

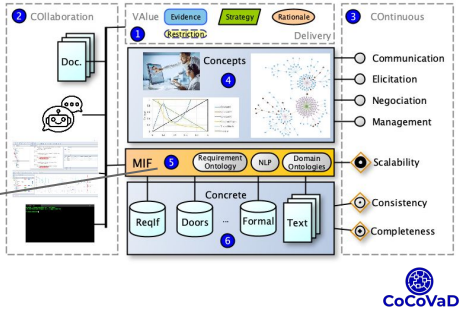
- Check **Q&A**
- **Check Value** Delivery
- **Allow** Collaboration
- **Support** activities

Ph.D. #2

- DevOPS
- Incremental V&V

CoCoVaD

Federation of models



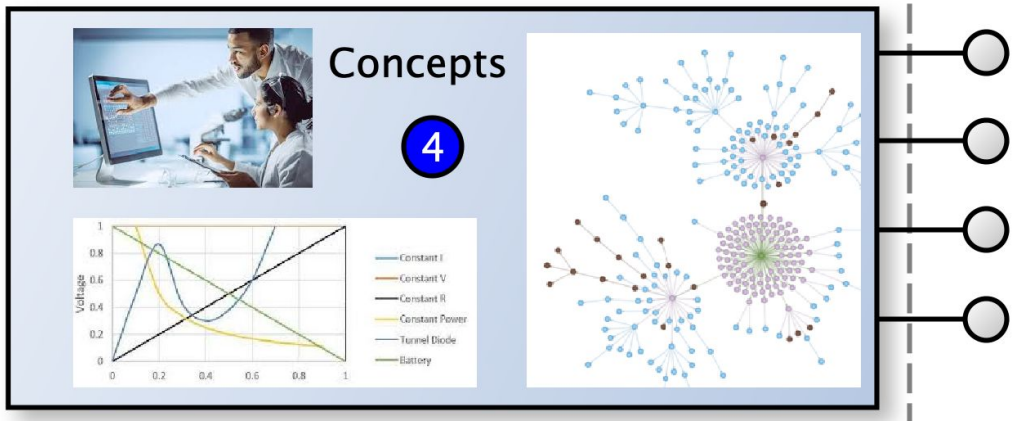
- Model As A Service
- Semantic alignment
- NoSQL
- Model mining

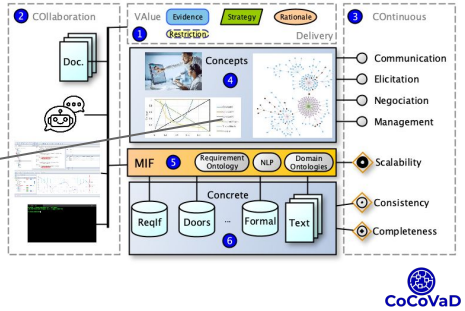
Ph.D. #3

- Abstraction
- Ontologies
- OO concepts

CoCoVaD

Single source of truth (data lake)



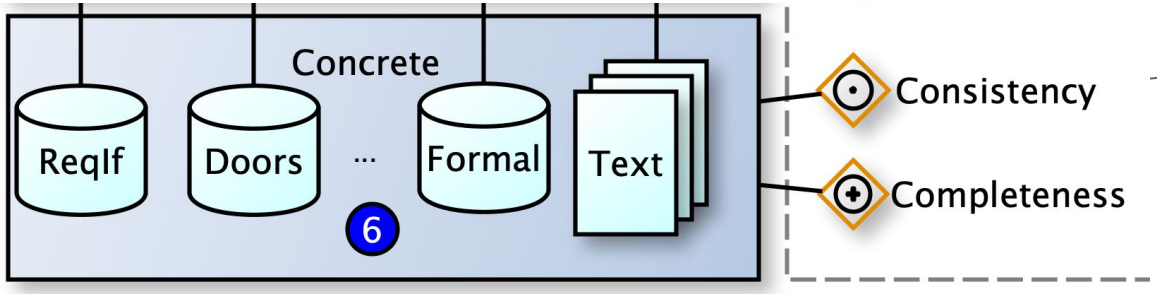


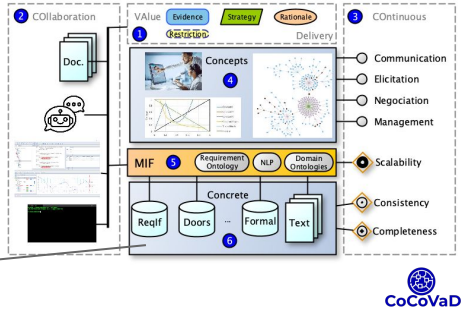
- Model As A Service
- Model & Data collaboration
- Massive data sets

- Abstraction
- Data extraction
- MBSE framework

CoCoVaD

Digitalization





- DDMS
- Zillions of licences and formats

Out of **our** scope!

Outline

- Concrete examples
- Context: the “CoCoVaD Airbus chair”
  - Formal Requirements
  - Requirements concepts Ontology
  - Business Analysis

Joint effort…

- Innopolis University
  - Alexandr 
  - Bertrand 
  - Manuel M. 
  - Maria 

- IRIT/SM@RT team
  - Florian 
  - Sophie 
  - JMB 

- CoCoVaD
  - Imen 
  - Thuy 
  - Mrunmayi 
  - Nathalie 
  - Marc 

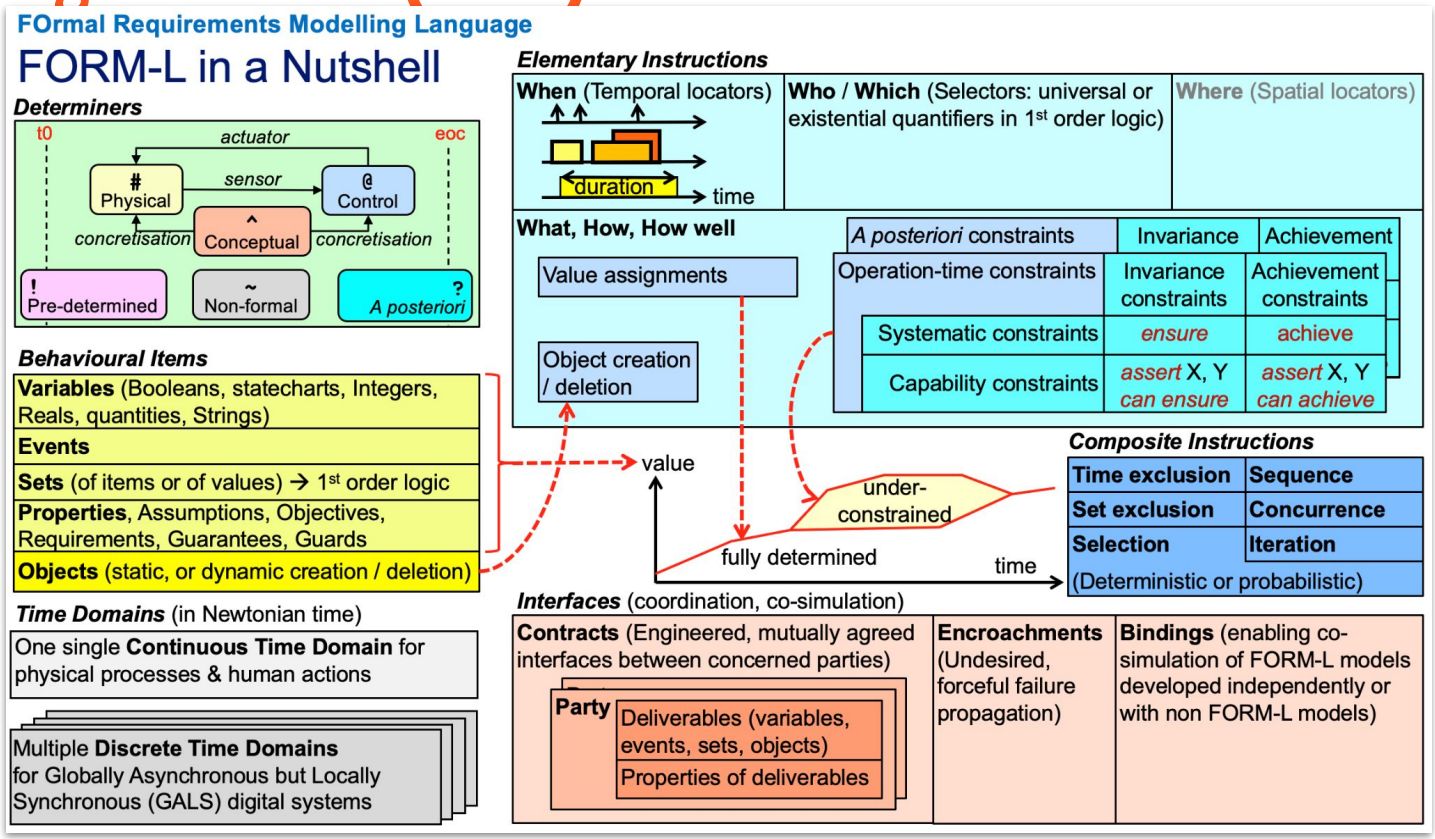
- Constructor University
  - Bertrand 
  - Manuel O. 
  - Li Huang 

Outline

- Concrete examples
- Context: the “CoCoVaD Airbus chair”
- **Formal Requirements**
- Requirements concepts Ontology
- Business Analysis

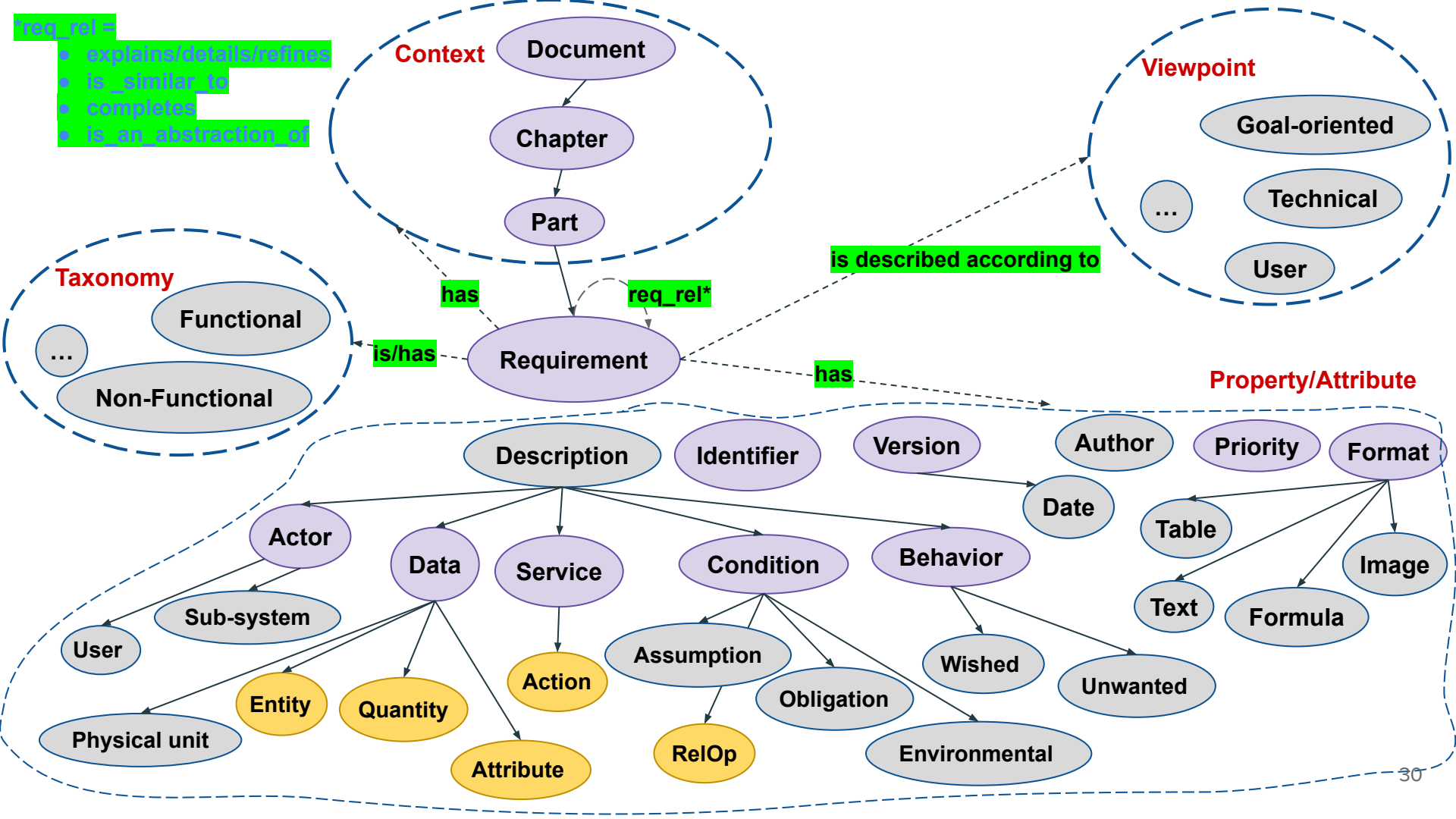
Identifying **ambiguities**

Working on Form-L (CRML)

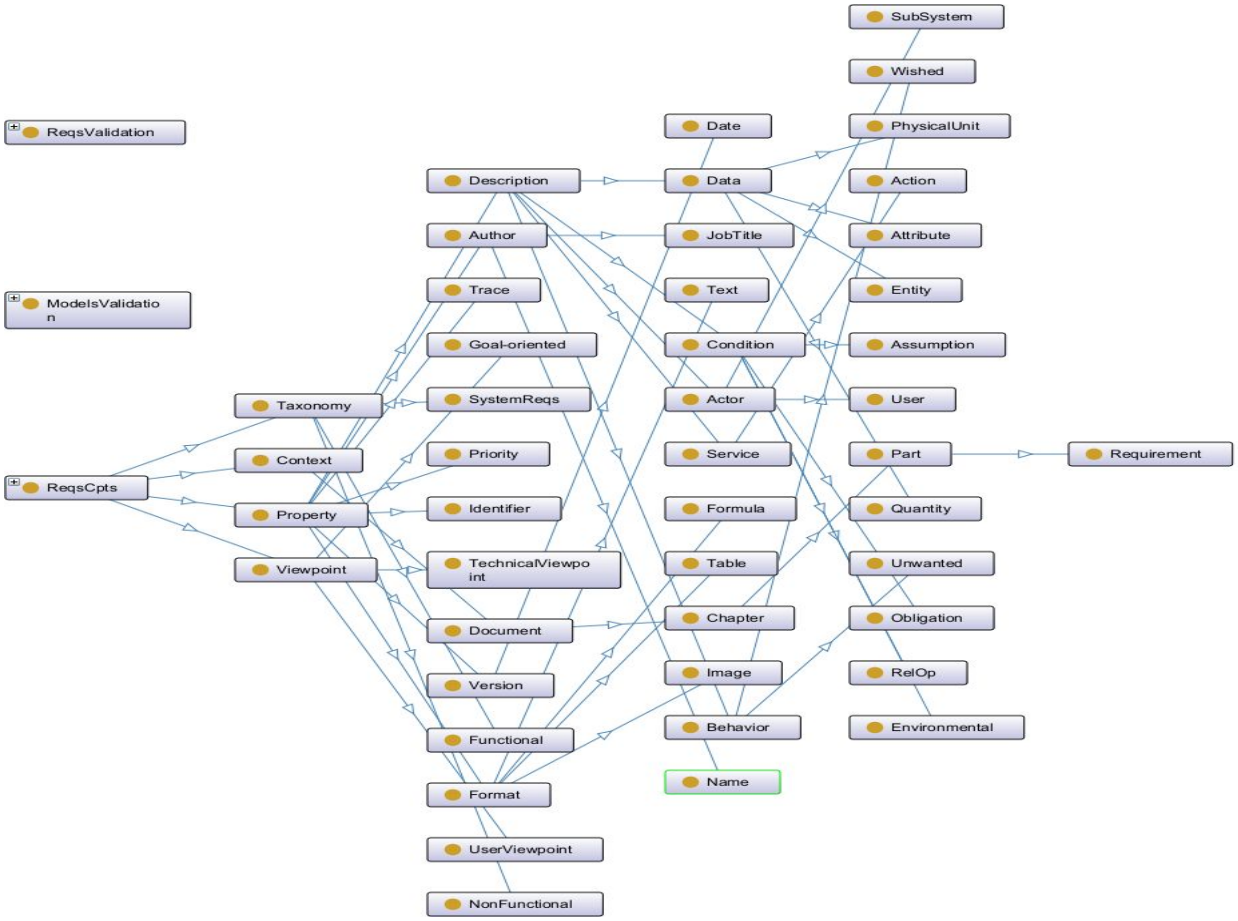


Outline

- Concrete examples
- Context: the “CoCoVaD Airbus chair”
  - Formal Requirements
  - Requirements concepts Ontology
  - Business Analysis

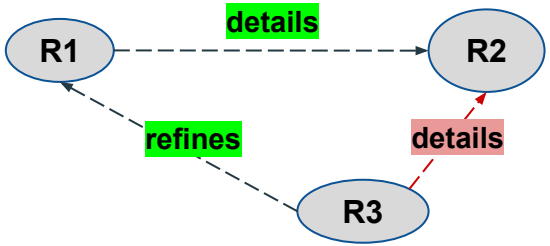

*req_rel =
explains/details/refines
is_similar_to
completes
is_an_abstraction_of
Context
Document
Chapter
Part
Viewpoint
Goal-oriented
Technical
User
...
Taxonomy
Functional
Non-Functional
...
has
req_rel*
is described according to
is/has
Requirement
has
Property/Attribute
Description
Identifier
Version
Author
Priority
Format
Date
Actor
Data
Service
Condition
Behavior
Table
Image
Text
Formula
User
Sub-system
Entity
Quantity
Physical unit
Attribute
Action
Assumption
Obligation
RelOp
Wished
Unwanted
Environmental

Protégé implementation (PoC)

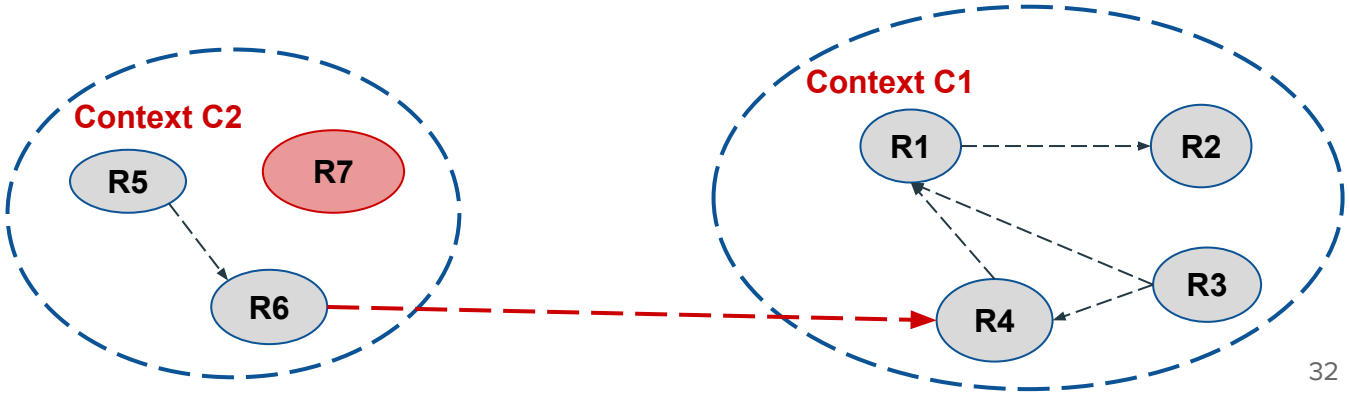


Benefit examples

- Deduction

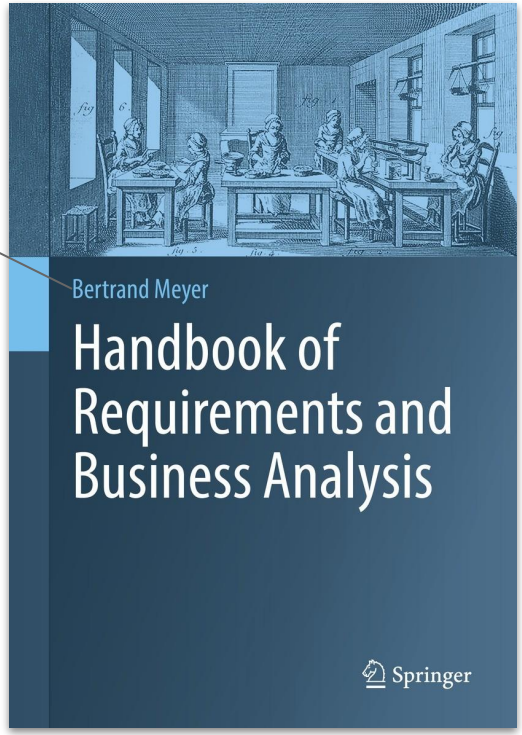


- Detection of holes/cracks in the requirements



Outline

- Concrete examples
- Context: the “CoCoVaD Airbus chair”
  - Formal Requirements
  - Requirements concepts Ontology
  - Business Analysis



https://se.inf.ethz.ch/requirements/

IEEE/SWEBOK/ISO definition of a Requirement

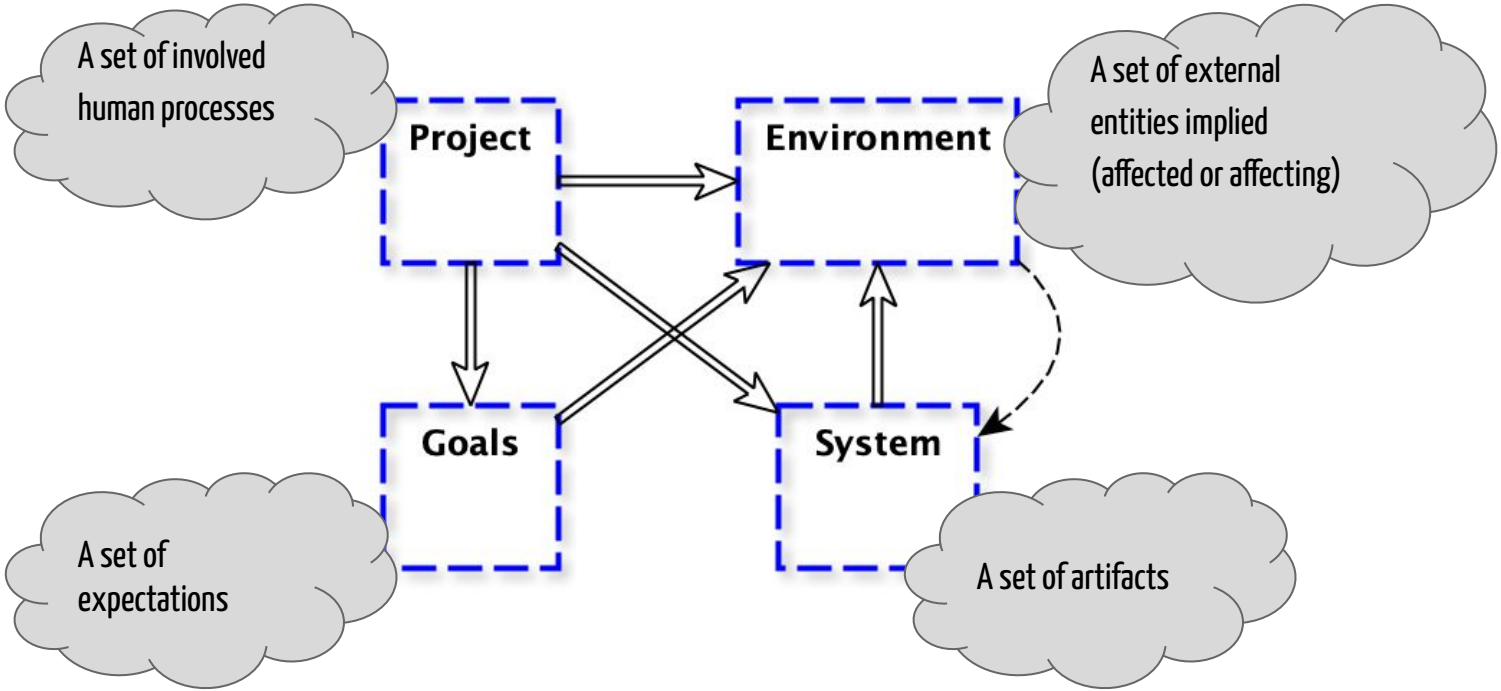
“A **1.1 Definition of a Software Requirement**

At its most basic, a software requirement is a property that must be exhibited by something in order to solve some problem in the real world. It may aim to automate part of a task for someone to support the business processes of an organization, to correct shortcomings of existing software, or to control a device—to name just a few of the many problems for which software solutions are possible. The ways in which users, business processes, and devices function are typically complex. By extension, therefore, the requirements on particular software are typically a complex combination from various people at different levels of an organization, and who are in one way or another involved or connected with this feature from the environment in which the software will operate.
”

http://swebokwiki.org/Chapter_1:_Software_Requirements

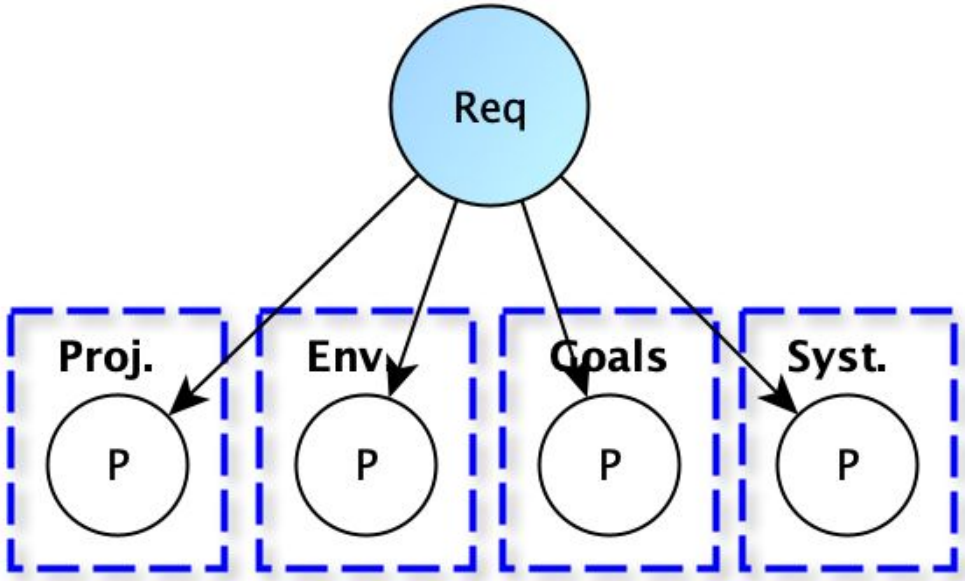
Context (universe of discourse)

"a **project** to develop a **system**, in a certain **environment**, to satisfy a set of **goals**"



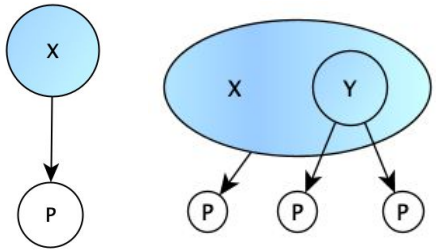
(our) General definition of a Requirement

"A requirement is a (relevant) **statement** about a **p**roject, **e**nvironment, **g**oals or **s**ystem **property**"



Elements of graphical representation

A requirement can be **Atomic** or **Composite**



The **type** of a requirement is the notation in which it is expressed (English, SysML, Eiffel, ...)

"The LGS has three components."

Some basic concepts

Property: boolean predicate (on a project, system or environment)

Statement: human-readable expression of a property

Kind of requirements (overview)

Kind of requirements (common to all PEGS)

- Component
- Responsability
  - *Role*
- Limit

Kind of requirements (Goals)

- Goal
  - *Obstacle*

Kind of requirements (Projects)

- Task
- Product

Kind of requirements (System)

- Behaviour
  - *Functional*
  - *Non-functional*
  - *Example*

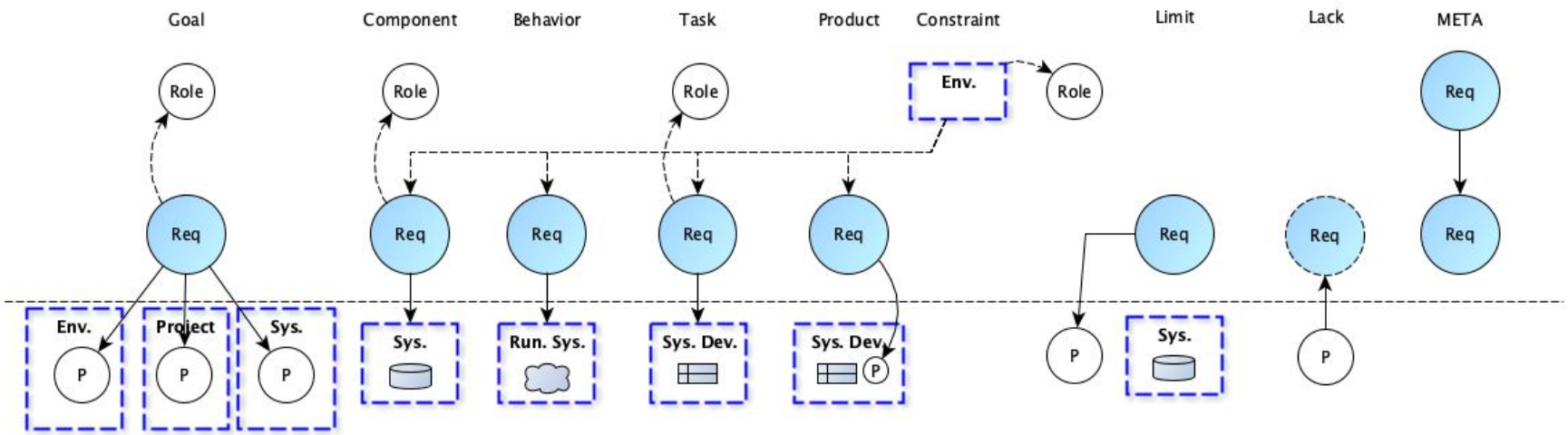
Kind of requirements (Environment)

- Constraint
  - *Business rule*
  - *Physical rule*
  - *Engineering decision*
- Assumption
- Effect
- Invariant

Kind of requirements (Document description)

- Silence
- Noise
  - *Hint*
- Meta-requirement
  - *Justification*

Classification (overview)



Categories of requirements (derived)

- Actor (from Component)
- Justification (from Meta)
- Role (from Responsibility)
- Obstacle (from Goal)
- Hint (from Noise)
- Obligation (from Constraint)
- Functional (from Behavior)
- Non-Functional (from Behavior)
- Example (from Behavior)

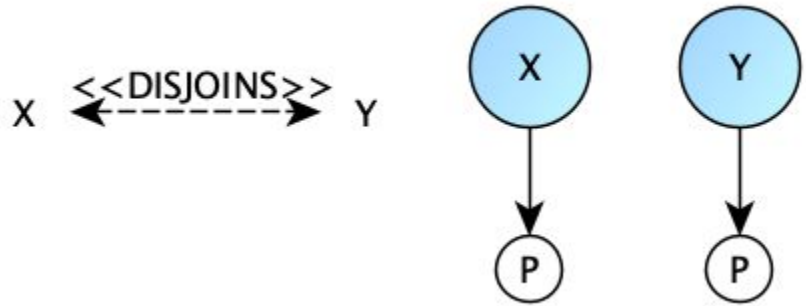
Categories of **relations**

Relations between requirements

- Disjoins (X || Y)
- Belongs (X ⊆ Y)
- Repeats (X ⇔ Y)
- Contradicts (X ⊕ Y)
- Extends (X > Y)
- Excepts (X \\ Y)
- Constrains (X ▸ Y)
- Characterizes (X ⟶ Y)

X and Y are unrelated

X || Y

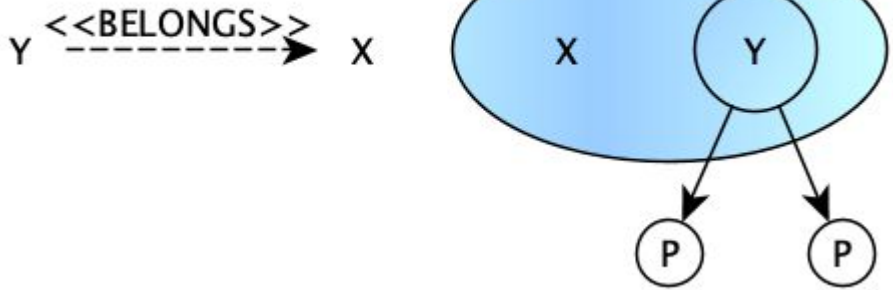


"The system is composed of three components."

"The car should be as economic in fuel consumption as possible."

Y is a sub-requirement of X

$Y \subseteq X$

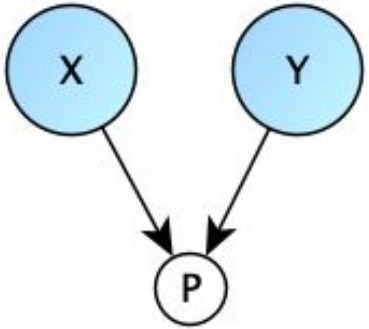


"4.3. System Externals"

"A customer is any user of the system that has not identified himself as an SBE employee."

X specifies the same property as Y

X ⇔ Y

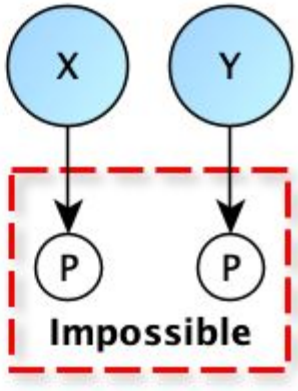


"The system is composed of three components."

"Here are the descriptions of the three parts of the system:"

X specifies a property in a way not compatible with Y

$X \oplus Y$

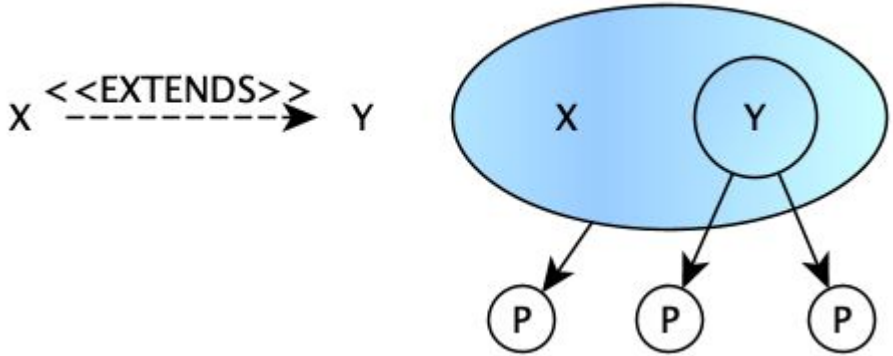


"The system has no interaction with human."

"The user should login interactively with the system."

X > Y

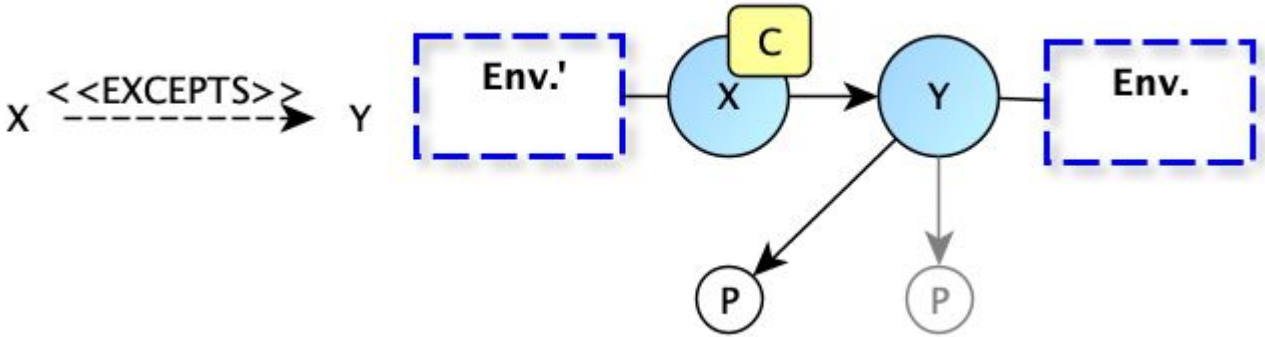
X assumes Y and specifies a property not specified by Y



"The online product ordering should allow direct access to the confirmation page."

"The system shall allow for online product ordering by either the customer or the sales agent."

X changes or removes, for a specified case, a property specified by Y

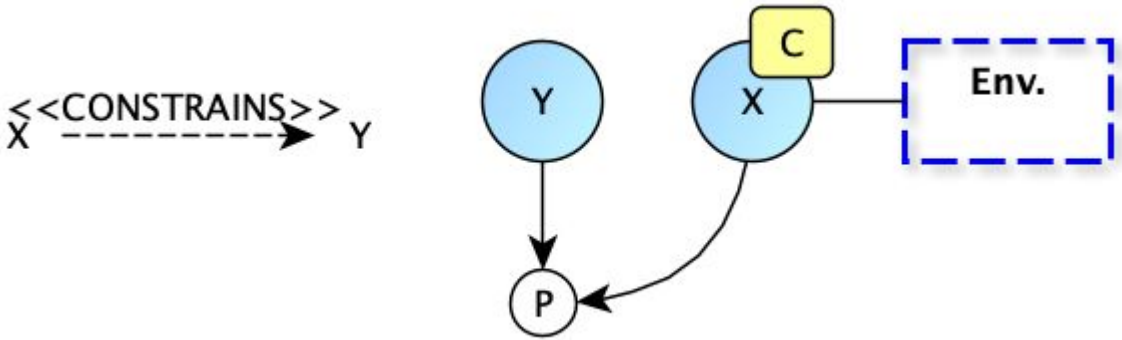


"In case of emergency braking, the system should prevent the wheels from frozing."

"The wheel can be frozen by braking."

X specifies a constraint on a property specified by X

X ▶ Y

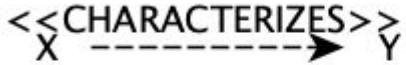


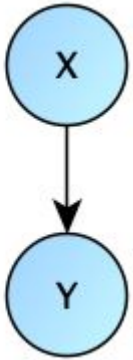
"The user is registered."

"In order to get personalized or restricted information, place orders or do other specialized transactions a user must login so that that the system can determine his access level."

X is a meta-requirement involving Y

$X \longrightarrow Y$





"The following requirement is optional:"

"The car should looks like a Ferrari."

What are the **benefits** ?

Examples of possible prescriptions

No **Duplicates**

Few **Excepts**

...

Contributions

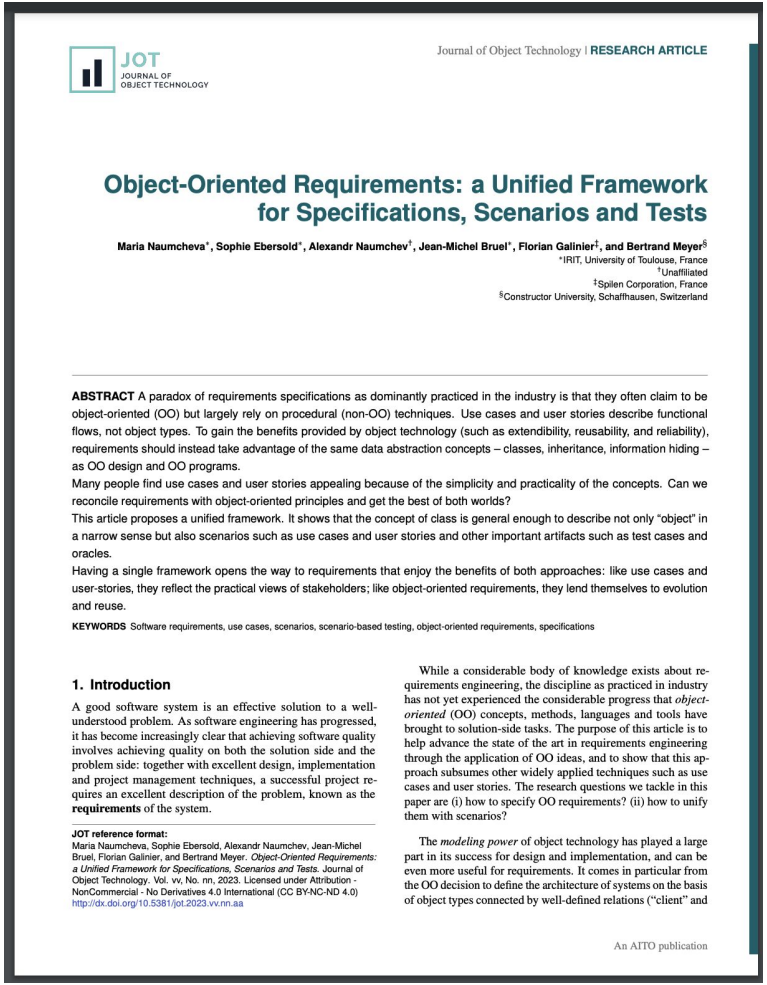
Clarification of reqs concepts

Basis for reqs methodology

Basis for critical analysis of reqs docs

Basis for NLP

...



JOT
JOURNAL OF OBJECT TECHNOLOGY

Journal of Object Technology | RESEARCH ARTICLE

Object-Oriented Requirements: a Unified Framework for Specifications, Scenarios and Tests

Maria Naumcheva*, Sophie Ebersold*, Alexandr Naumchev†, Jean-Michel Bruel*, Florian Galinier‡, and Bertrand Meyer§
*IRIT, University of Toulouse, France
†Unaffiliated
‡Spilen Corporation, France
§Constructor University, Schaffhausen, Switzerland

ABSTRACT A paradox of requirements specifications as dominantly practiced in the industry is that they often claim to be object-oriented (OO) but largely rely on procedural (non-OO) techniques. Use cases and user stories describe functional flows, not object types. To gain the benefits provided by object technology (such as extendibility, reusability, and reliability), requirements should instead take advantage of the same data abstraction concepts – classes, inheritance, information hiding – as OO design and OO programs.

Many people find use cases and user stories appealing because of the simplicity and practicality of the concepts. Can we reconcile requirements with object-oriented principles and get the best of both worlds?

This article proposes a unified framework. It shows that the concept of class is general enough to describe not only "object" in a narrow sense but also scenarios such as use cases and user stories and other important artifacts such as test cases and oracles.

Having a single framework opens the way to requirements that enjoy the benefits of both approaches: like use cases and user-stories, they reflect the practical views of stakeholders; like object-oriented requirements, they lend themselves to evolution and reuse.

KEYWORDS Software requirements, use cases, scenarios, scenario-based testing, object-oriented requirements, specifications

1. Introduction

A good software system is an effective solution to a well-understood problem. As software engineering has progressed, it has become increasingly clear that achieving software quality involves achieving quality on both the solution side and the problem side: together with excellent design, implementation and project management techniques, a successful project requires an excellent description of the problem, known as the **requirements** of the system.

While a considerable body of knowledge exists about requirements engineering, the discipline as practiced in industry has not yet experienced the considerable progress that *object-oriented* (OO) concepts, methods, languages and tools have brought to solution-side tasks. The purpose of this article is to help advance the state of the art in requirements engineering through the application of OO ideas, and to show that this approach subsumes other widely applied techniques such as use cases and user stories. The research questions we tackle in this paper are (i) how to specify OO requirements? (ii) how to unify them with scenarios?

The *modeling power* of object technology has played a large part in its success for design and implementation, and can be even more useful for requirements. It comes in particular from the OO decision to define the architecture of systems on the basis of object types connected by well-defined relations ("client" and

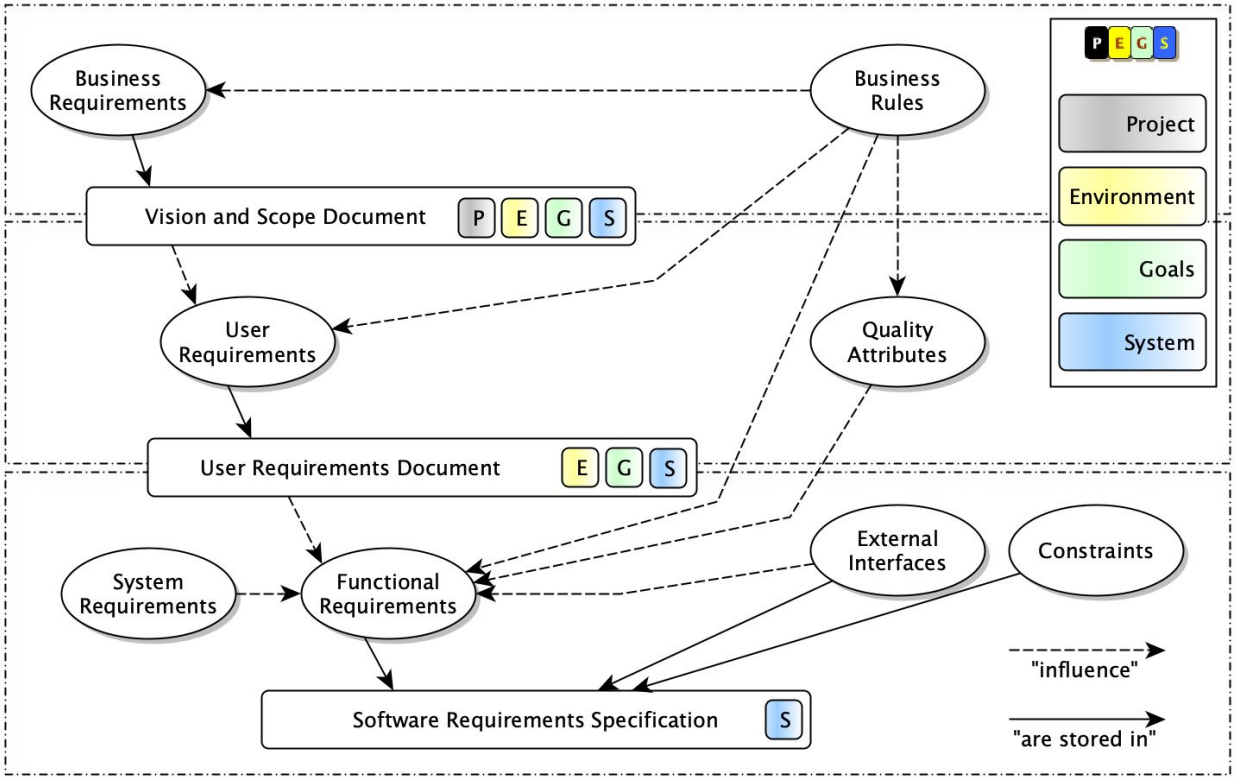
JOT reference format:
Maria Naumcheva, Sophie Ebersold, Alexandr Naumchev, Jean-Michel Bruel, Florian Galinier, and Bertrand Meyer. *Object-Oriented Requirements: a Unified Framework for Specifications, Scenarios and Tests.* Journal of Object Technology. Vol. vv, No. nn, 2023.
http://dx.doi.org/10.5381/jot.2023.vv.nn.aa

An AITO publication

http://dx.doi.org/10.5381/jot.2023.22.1.a3

Ongoing effort

Alignment with “Classics”



E.g., Wiergers & Beatty

Effective business analysis

- Companion material for an upcoming book… (https://requirements.university)
- Tutorial at RE'23

More than Word & Excel

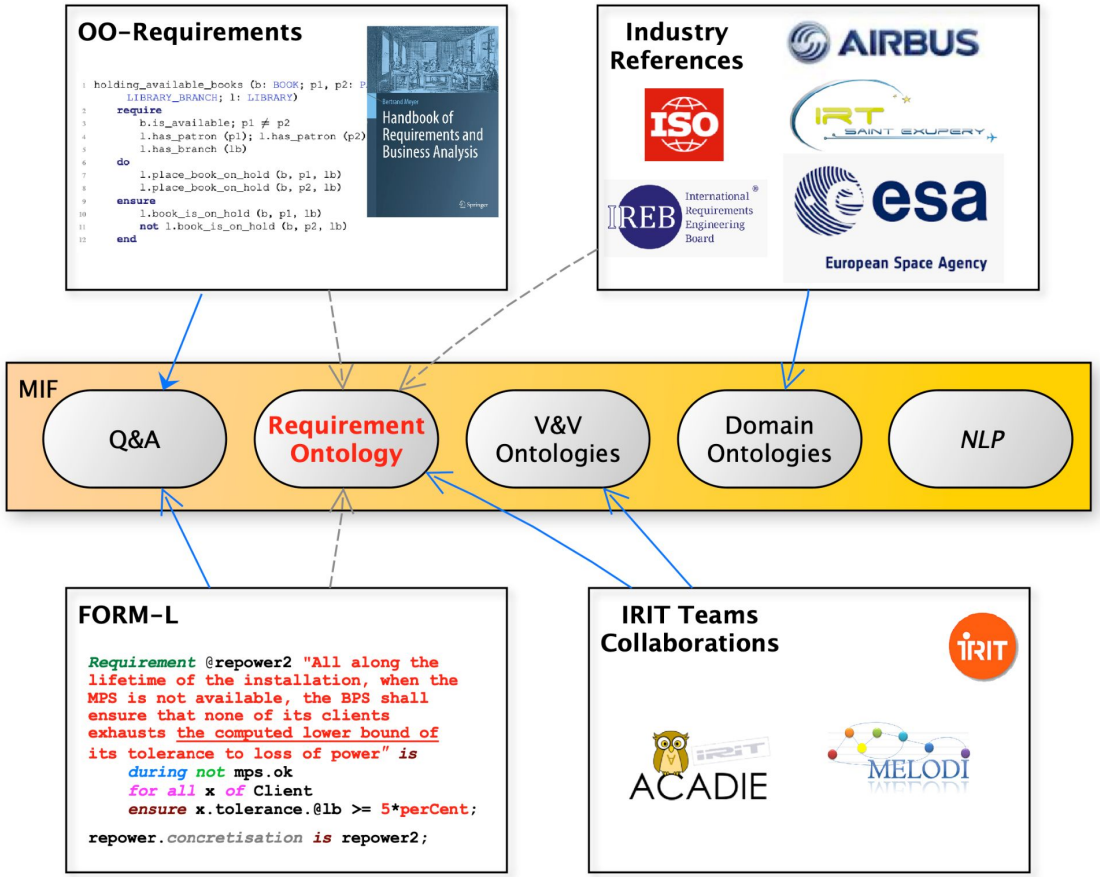
- Markdown-like format
- GitHub itself
- Quality metrics & rules **implemented**

Requirements documents can be tested!

```
#-----------------------------
# language: en
Feature: Book mutual references
    The books should follow the mutual references rules.

Scenario: The Environment book must not refer to the Goals and Project books
    Given The Environment book
    Then No reference should include the Goals book
    And No reference should include the Project book
    And Only E.5 section can refer to the System book

Scenario: The Goals book must not refer to the Project and System books
    Given The Goals book
    Then No reference should include the Project book
    And No reference should include the System book

Scenario: The System book must not refer to the Project book
    Given The System book
    Then No reference should include the Project book
```

One last thing...

We are hiring! (and looking for collaborations)



Concrete positions

- 2 Ph.D. positions on
  - “Requirements Analysis in the Aeronautic Industry: Enhancing Quality and Usability”
  - “V&V in an Industrial DevOps Context”
- Summer Internships (B.Sc/M1) on DLS/MDE

Discussion time!