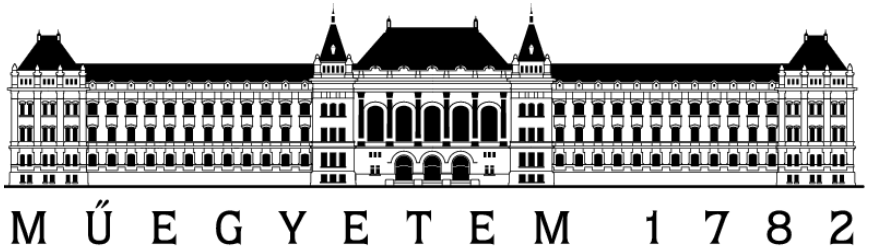


Towards Open Modular Critical Systems

Prof. András Pataricza
Budapest University of Technology and Economics
SC of the Hungarian ARTEMIS NTP

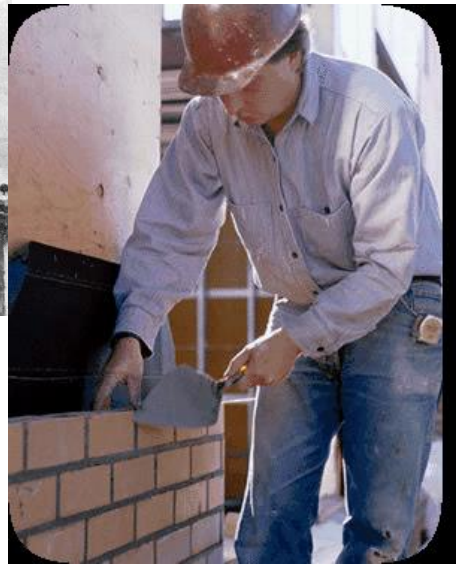
pataric@mit.bme.hu

Global technology challenges:

- Complexity: over a critical treshold.
  - Traditional, heuristic development?
  - Quality and safety certification: best effort processes.
    - Only indirect guarantees for product quality/safety
- Low level of automation and productivity.
  - Huge expert effort.
  - Qualified staff: bottleneck.
  - Long development times.
- Low level of reusability.
- Cultural divergence : branches of the ES industry:
  - Separation of application domains.
  - Production volumes below the optimal.
  - Education/training: global education is insufficient,
  - Domain standards:

ES paradigm shift

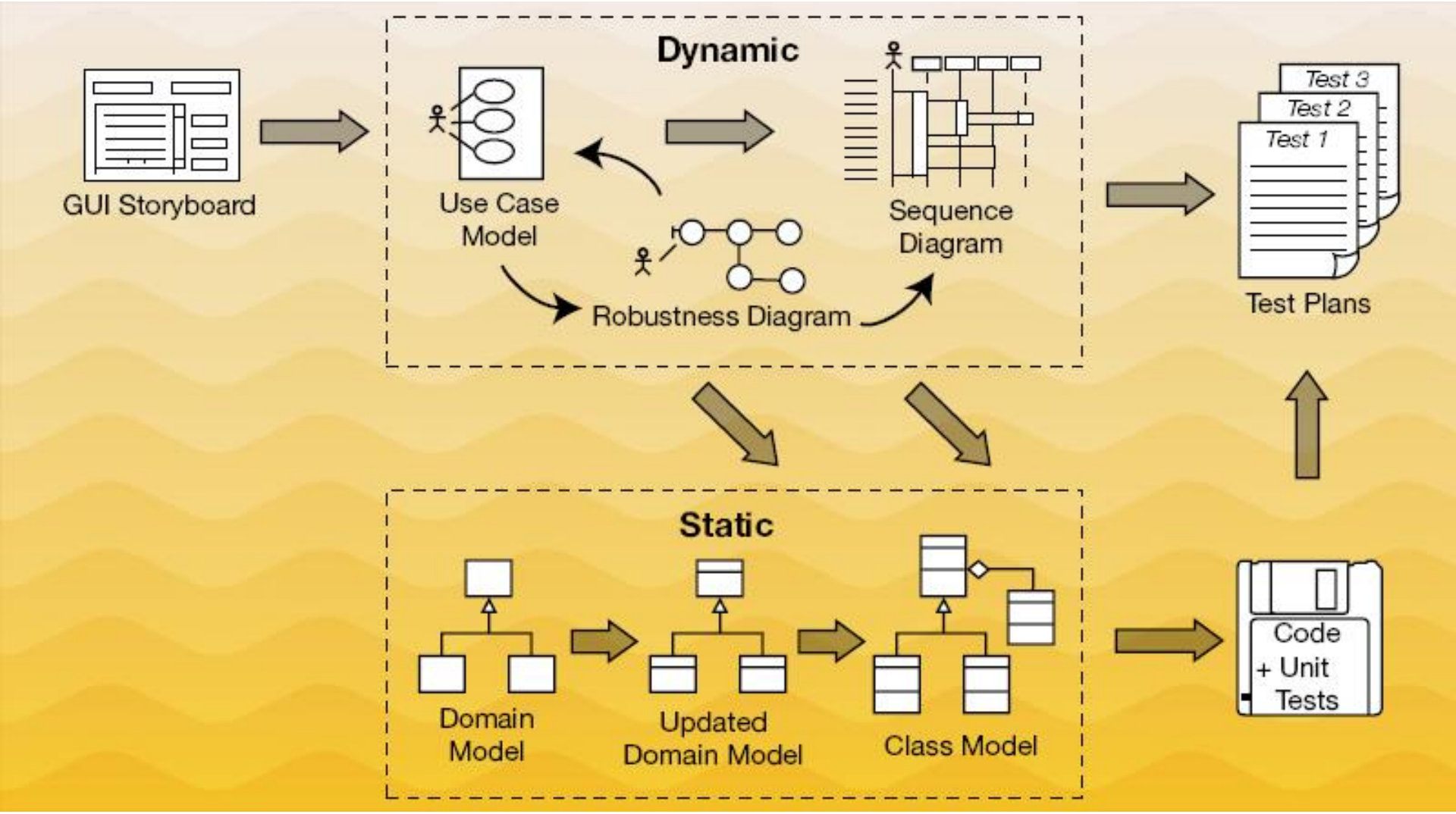
Traditional



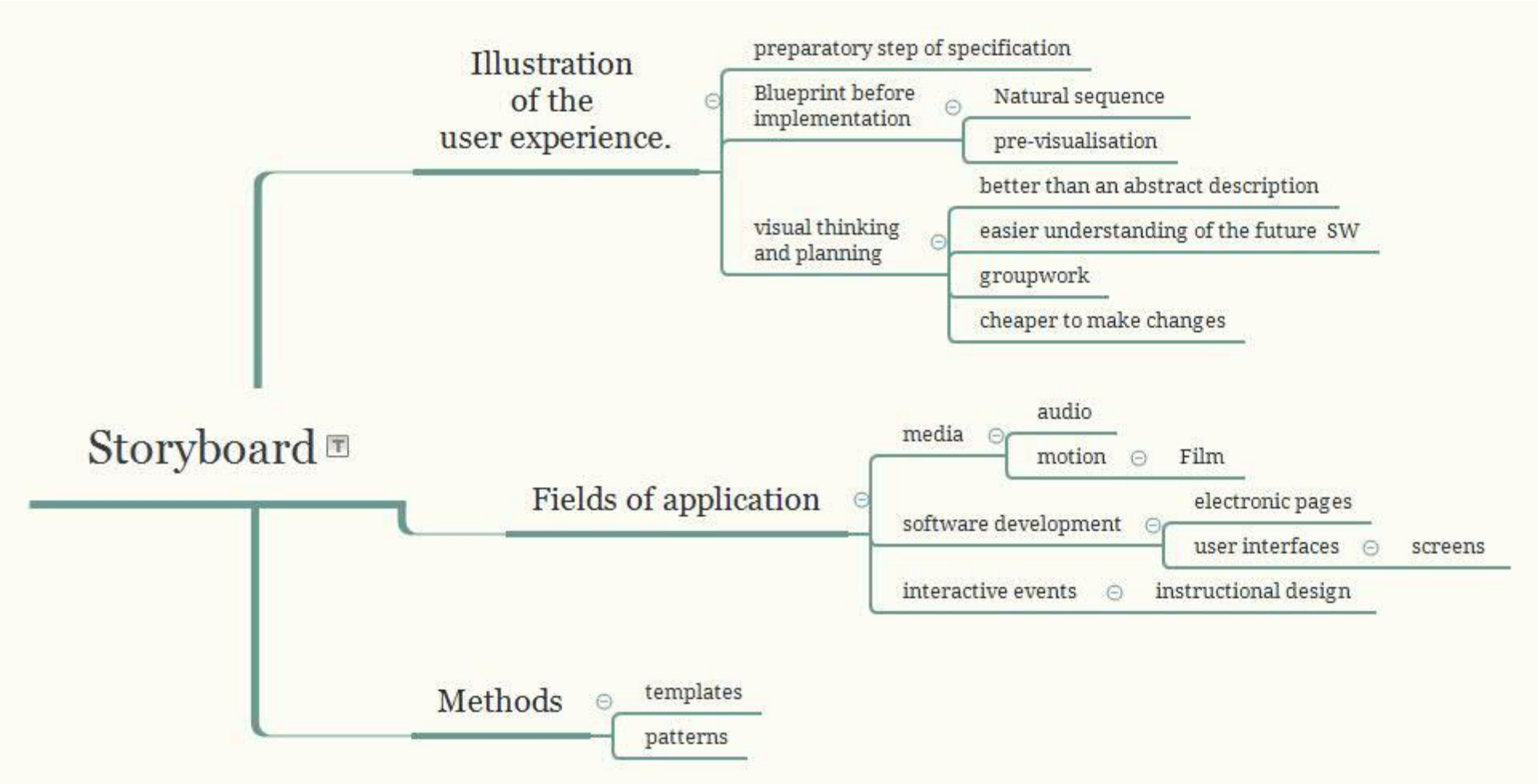
Industrialized

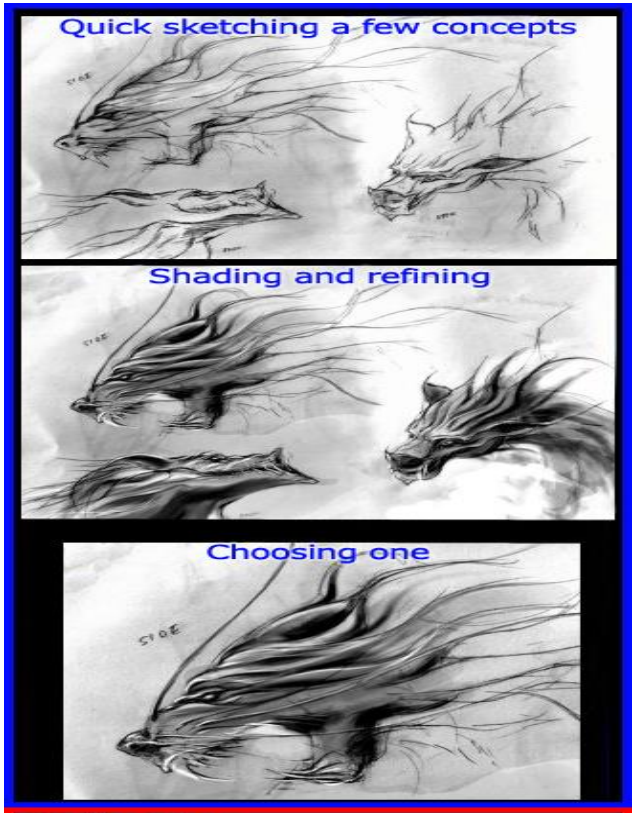
INDUSTRIAL PRODUCTION NEEDS A PROCESS

ICONIX

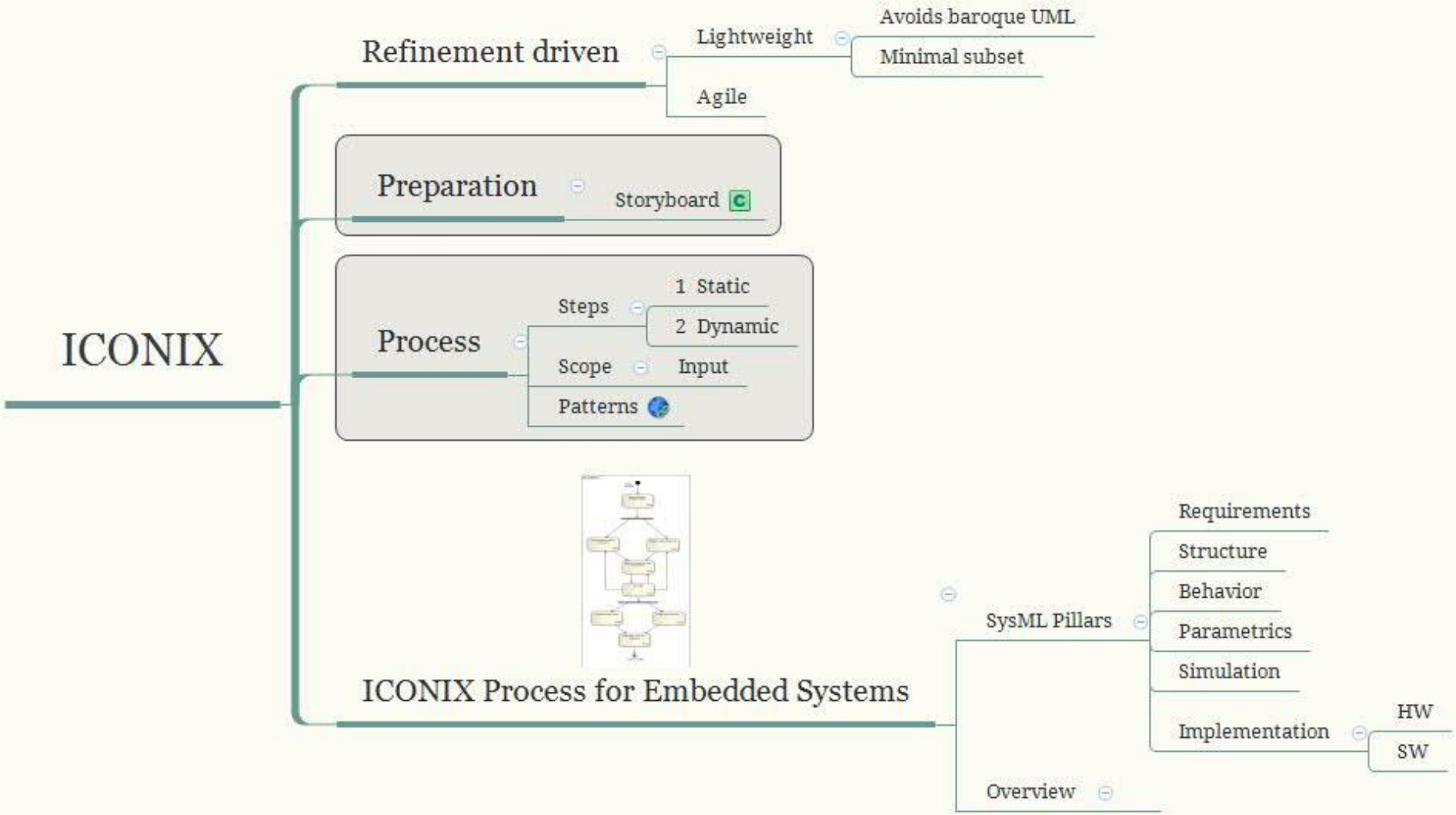

Dynamic
GUI Storyboard
Use Case Model
Sequence Diagram
Robustness Diagram
Test 3
Test 2
Test 1
Test Plans
Static
Domain Model
Updated Domain Model
Class Model
Code + Unit Tests

Storyboard


Storyboard
Illustration of the user experience.
preparatory step of specification
Blueprint before implementation
Natural sequence
pre-visualisation
visual thinking and planning
better than an abstract description
easier understanding of the future SW
groupwork
cheaper to make changes
Fields of application
media
audio
motion
Film
software development
electronic pages
user interfaces
screens
interactive events
instructional design
Methods
templates
patterns


Quick sketching a few concepts
Shading and refining
Choosing one
3D Modeling
Refining 3D model

A sample process model



Model based thinking is the fundamental approach in engineering

Good engineering needs multi-aspect thinking and modeling

Modeling is the basis for formal methods

OMG promised a silver bullet with UML,
today we know, that it is not

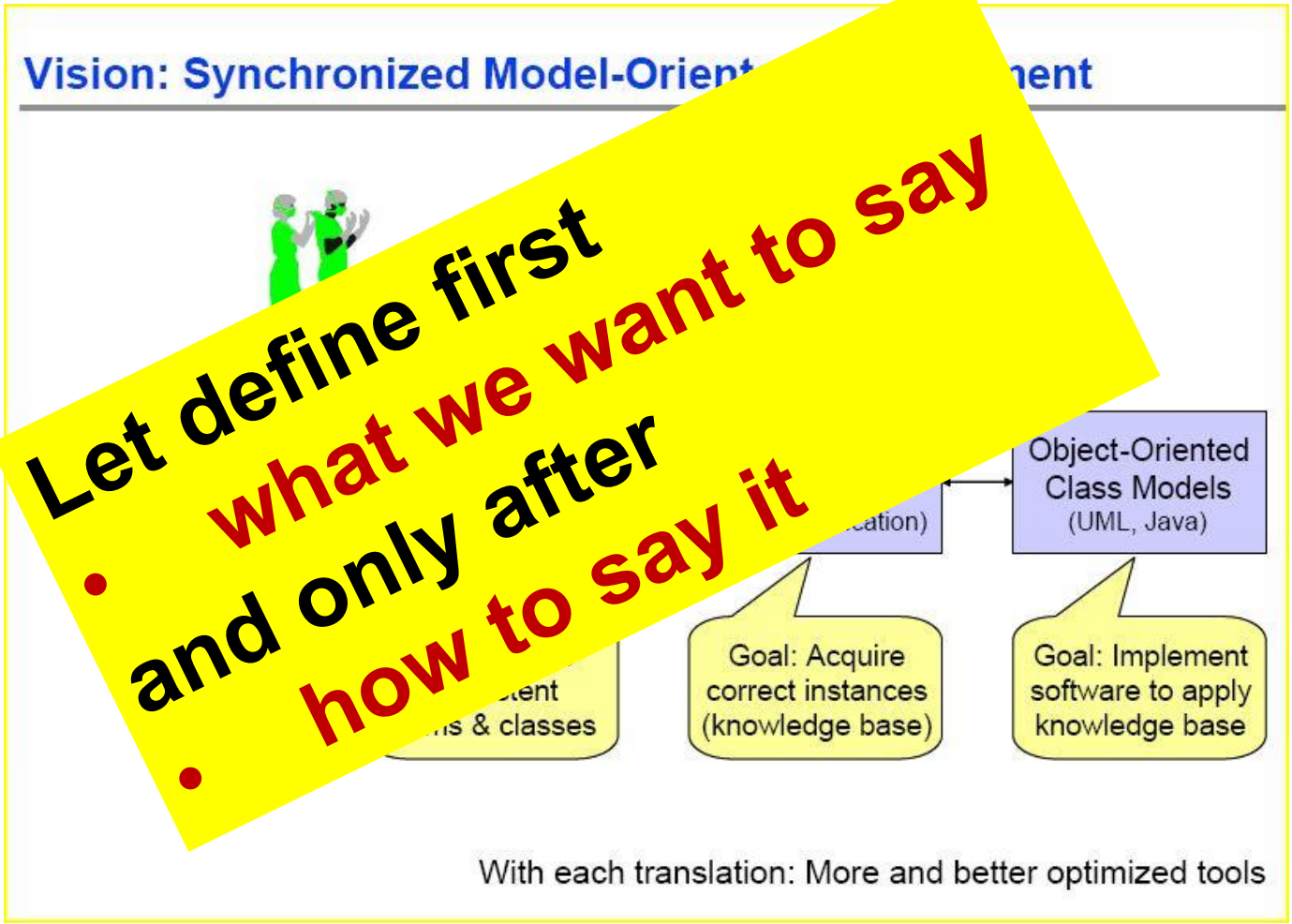
Design and analysis need clear concepts

Standards demand for formal methods
(IEC 61508, CENELEC, ISO 26262,DO-178C)

BUT: What does it mean „fail silent”?

STARTING POINT FOR MDA
FORMALIZED CONCEPTS

From ontologies to metamodels



H. Knublauch:
Ontology Design and Software Technology, Colloquium - Stanford Medical Informatics, 2003

Ontology

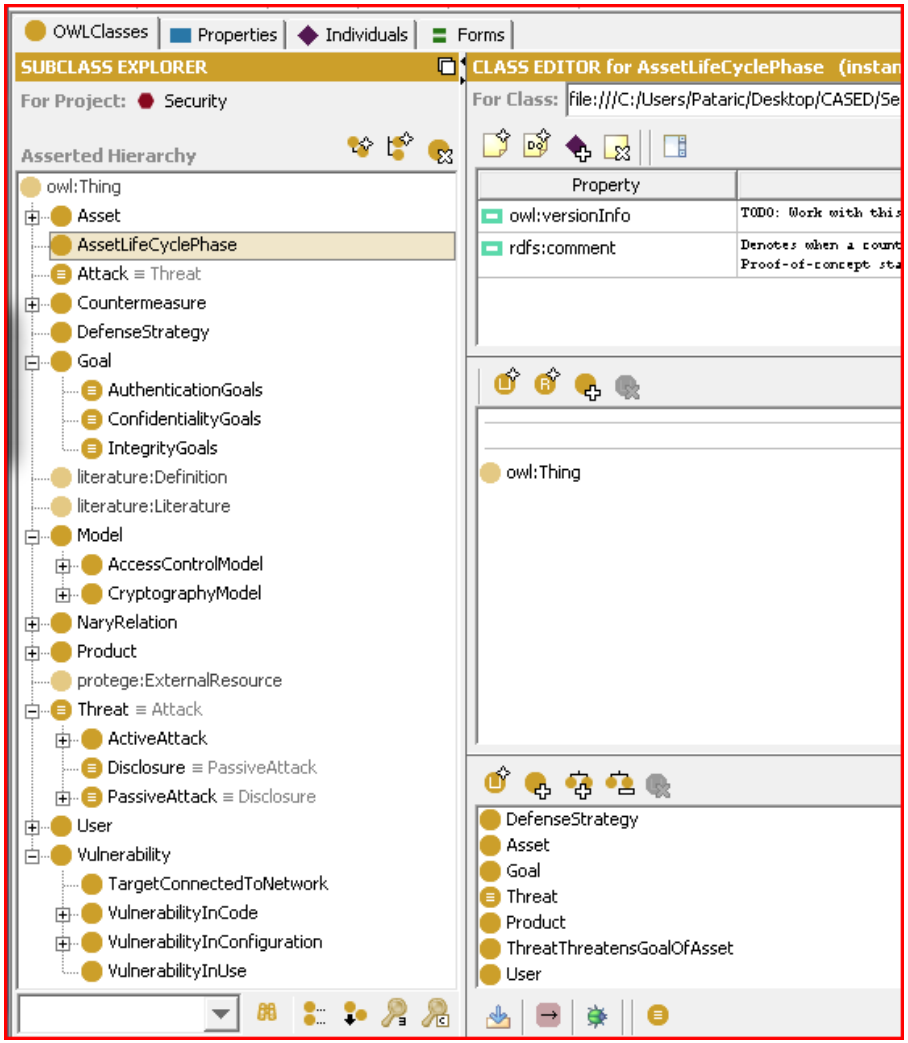
A data model that
- represents a domain and
- Has a logic in the background
- is used to reason about
  - the objects in that domain and
  - the relations between them.

Ontologies generally describe:
- **Individuals:** basic objects
- **Classes:** sets, collections, or types of objects
- **Attributes:** properties, features… that objects can have and share
- **Relations:** ways that objects can be related

Reasoning :

Concept space traversal
- **subsumption test** wrt. different profiles
- **consistency check**: satisfiability
- **circular containment** of classes

Example: part of the security ontology



A. Herzog et al:
An Ontology of Information Security
Int. J. of Inf. Security and Privacy (1), 4

ISO 24707:2007 Common logic

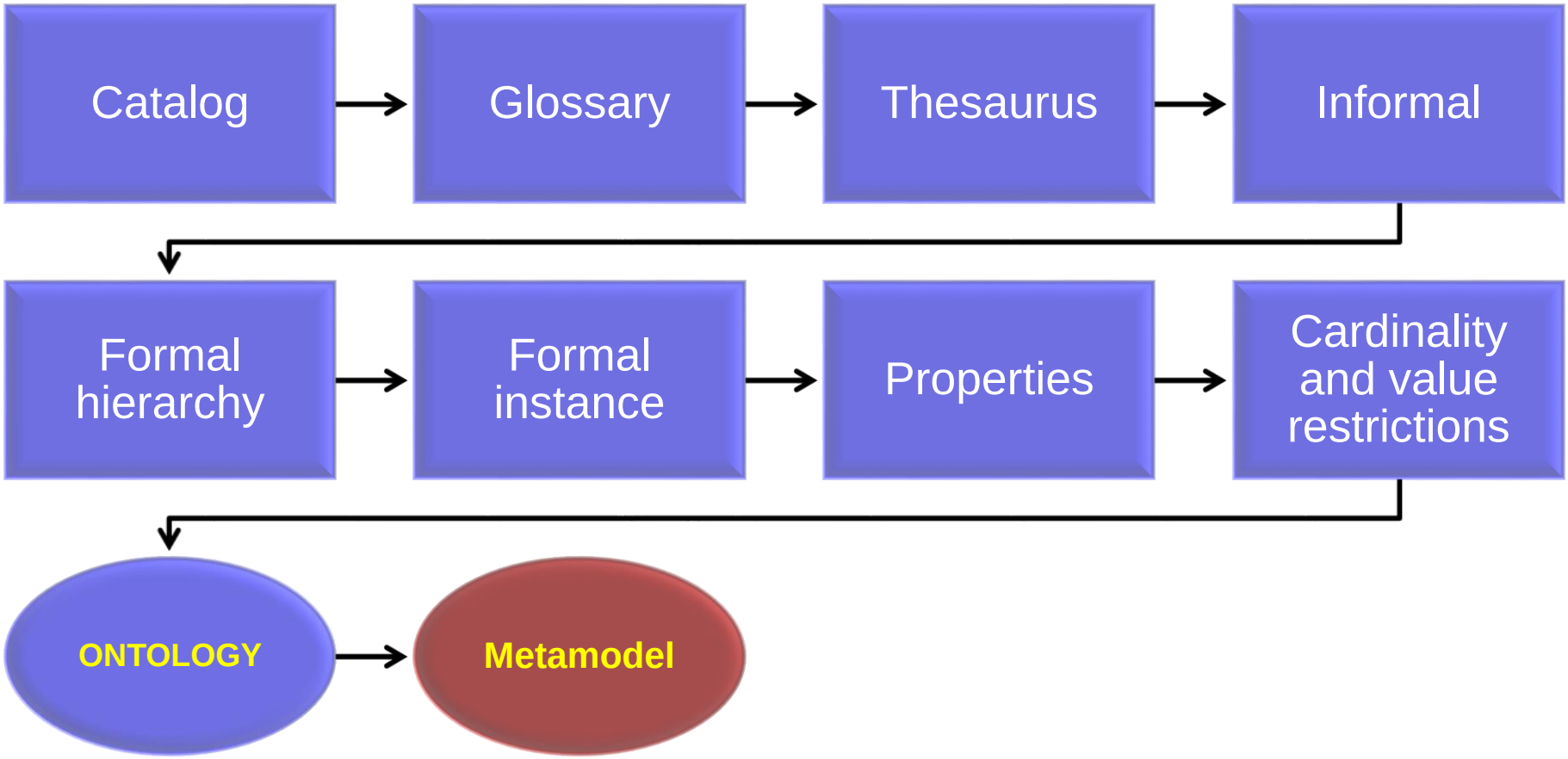
Information technology — Common Logic (CL): a framework for a family of logic-based languages

- Framework for a family of logic languages,
  - based on first-order logic,
  - Exchange of knowledge in IT systems.
- Supports different syntactic forms (dialects).
  - syntactic CL conformance of dialect -> CL semantics for free
  - all CL dialects are equivalent mechanical translation

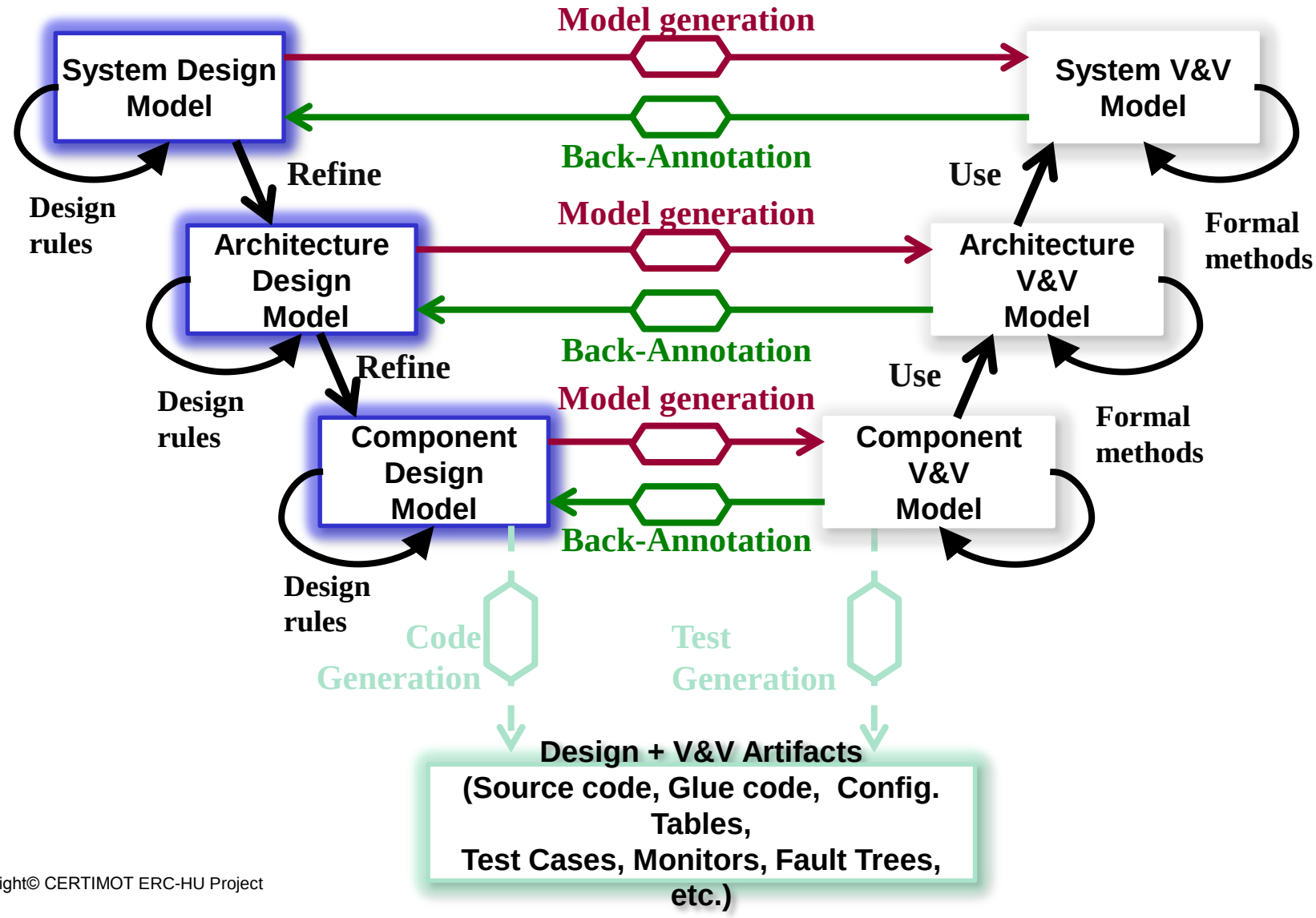
V&V: use of reasoners

- Metamodel level
  - consistency check: inconsistent class (satisfiability check)
    - no instances satisfying the class descriptions (e.g. multiplicity conflict)
  - subsumption test: e.g. redundant concepts
  - cycle detection: loop in the concept hierarchy
  - uniqueness: e.g. redundant concepts
- Model level
  - Consistency check
    - checks the conformance of the model to the metamodel
    - consistency of the instance model w.r.t. the metamodel ontology
  - Property check
    - Reduction of technical problems to satisfiability check
      E.g. is there an instance violating security requirements?

Ontology/metamodel design workflow



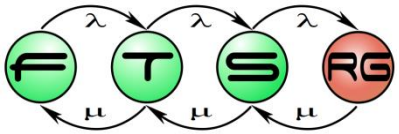
„Y-model” in MDE of Critical Systems



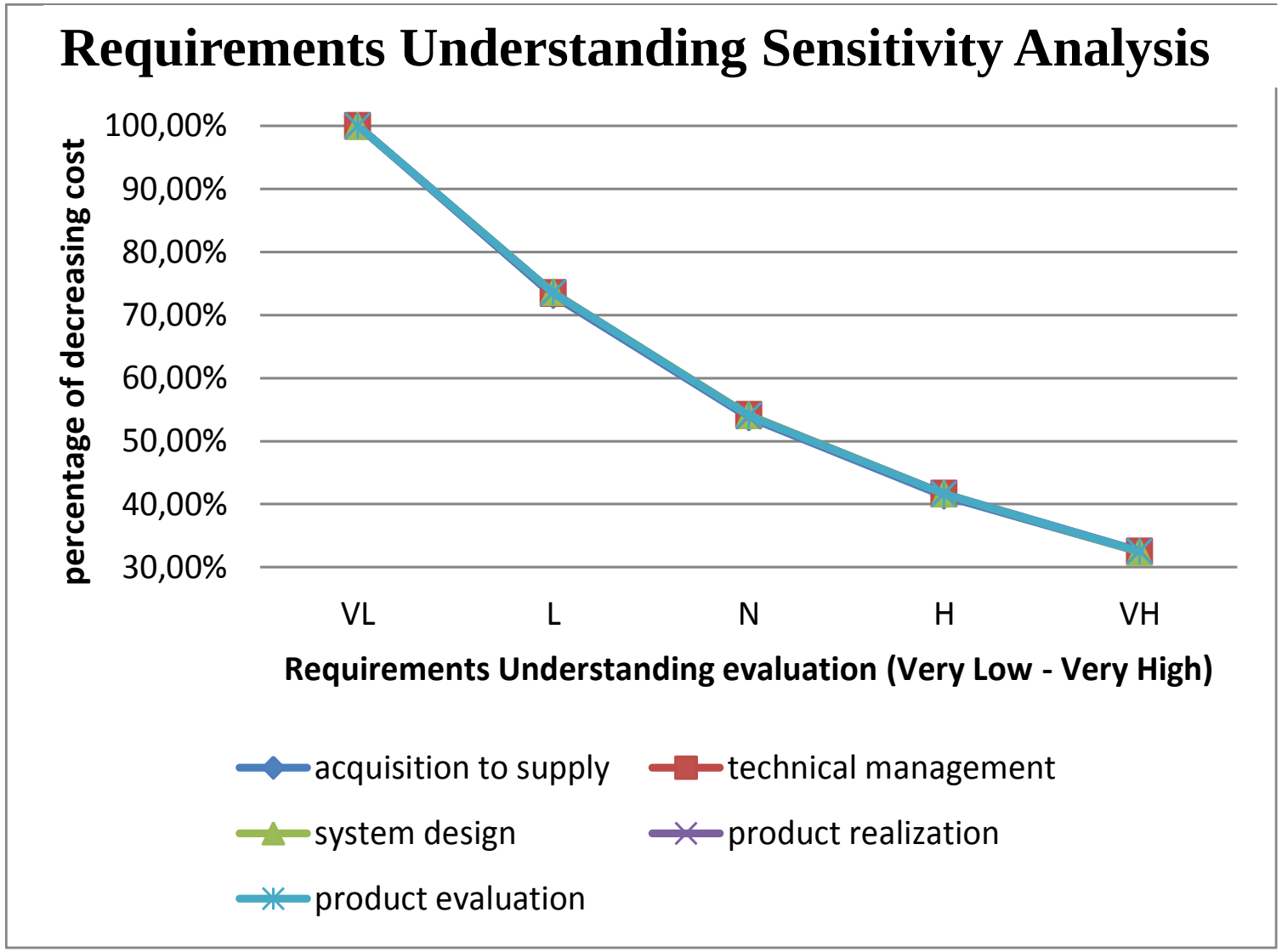
Requirements Interchange Format (ReqIF) V1.02

Based on http://www.omg.org/spec/ReqIF/1.2

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



Sensitivity analysis: Req understanding



Cost impact estimation

# of System Requirements	Easy	Nom.	Diff.
# New	0,5	1,0	5,0
# Design For Reuse	0,7	1,4	6,9
# Modified	0,3	0,7	3,3
# Deleted	0,3	0,5	2,6
# Adopted	0,2	0,4	2,2
# Managed	0,1	0,2	0,8

1. Quality and stability Modification: ~ 70% !
2. Requirement set complexity reduction

- 4 similar problems
- Separate solution:

$$4 \times New = \mathbf{400\%}$$

- Global solution:

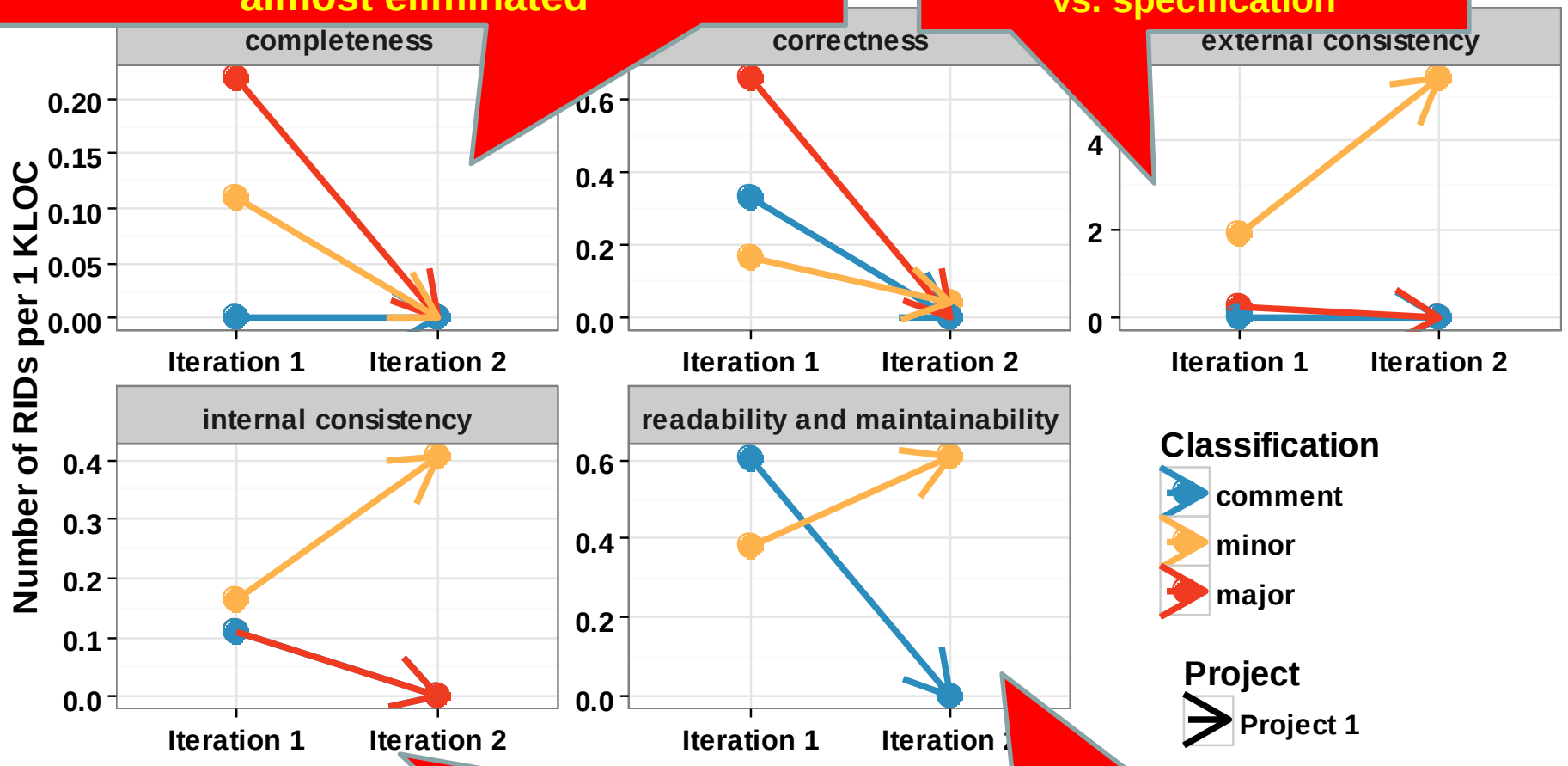
$$1 \times Reuse + 4 \times Adopt = \mathbf{310\%}$$

- **DECOMPOSITION**

Proportion of faulty artefacts: Project 1

Correctness and completeness faults: almost eliminated

Lot of (mostly major) faults vs. specification

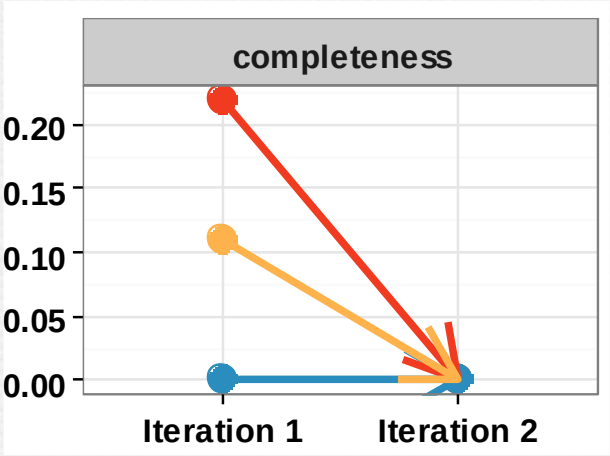


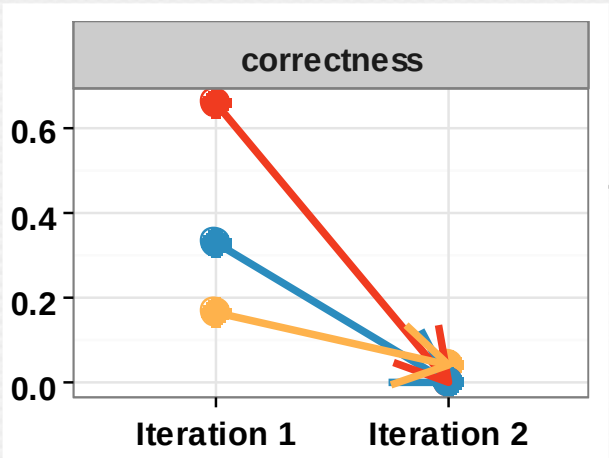
Consistency of the code got worse!

Readability did not improve….

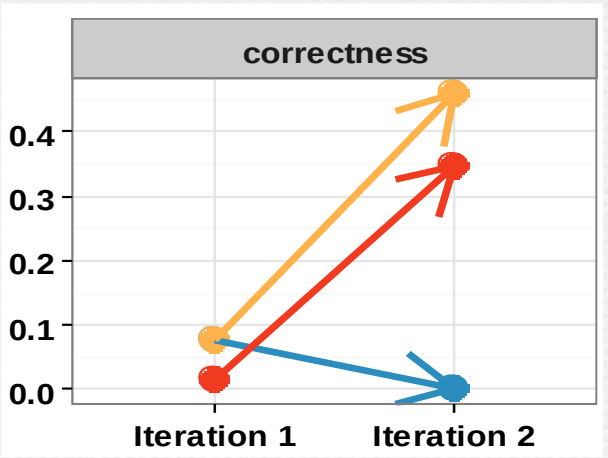
Fundamental differences in the trends of faulty artefacts

Project1

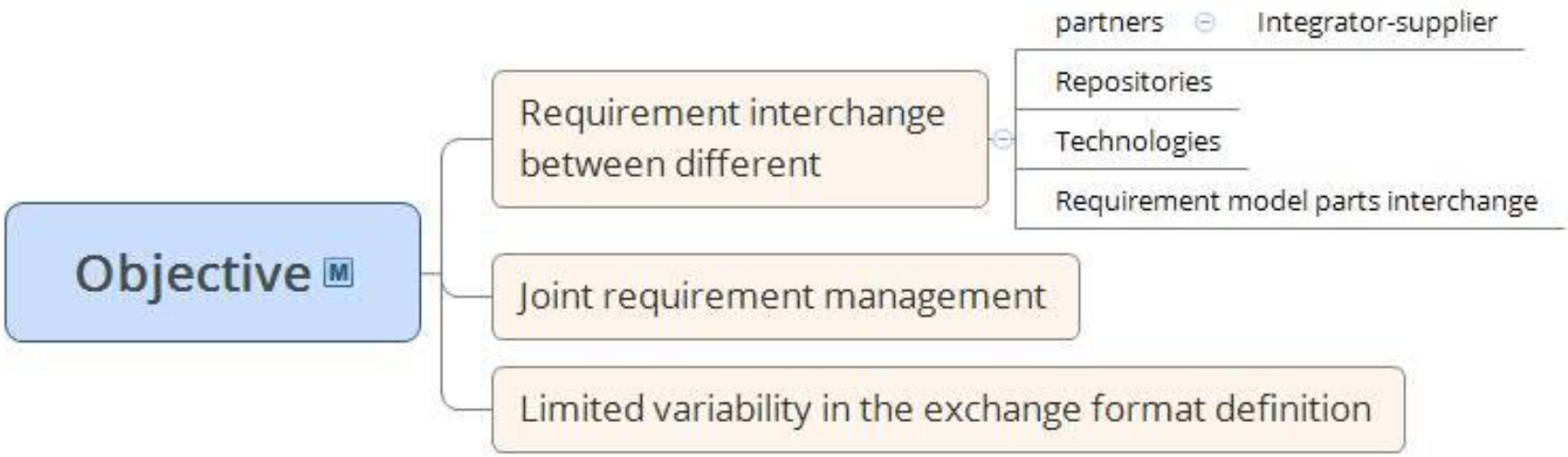




Project2



Requirements Interchange Format (ReqIF): Objective



Traceability

REQUIREMENTS TRACEABILITY MATRIX

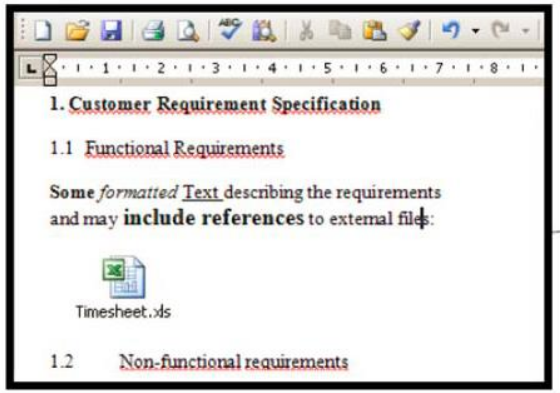
Project Name:	<optional>
National Center:	<required>
Project Manager N	<required>
Project Description	<required>

REQUIREMENTS TRACEABILITY MATRIX

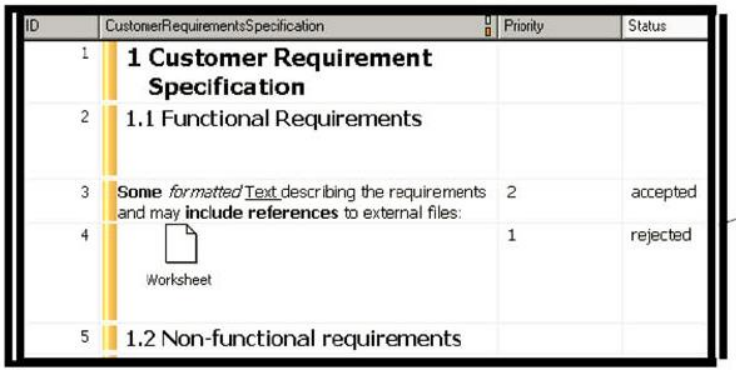
Project Name:	<optional>
National Center:	<required>
Project Manager Name:	<required>
Project Description:	<required>

ID	Assoc ID	Technical Assumption(s) and/or Customer Need(s)	Functional Requirement	Status	Architectural/Design Document	Technical Specification	System Component(s)	Software Module(s)	Test Case Number	Tested In	Implemented In	Verification	Additional Comments
001	1.1.1												
002	2.2.2												
003	3.3.3												
004	4.4.4												
005	5.5.5												
006													
007													
008													
009													
010													
011													
012													
013													
014													
015													
016													
017													
018													
019													
020													
021													
022													
023													
024													
025													
026													
027													
028													
029													
030													
031													
032													
033													
034													

Requirements authoring tools vs. word processing



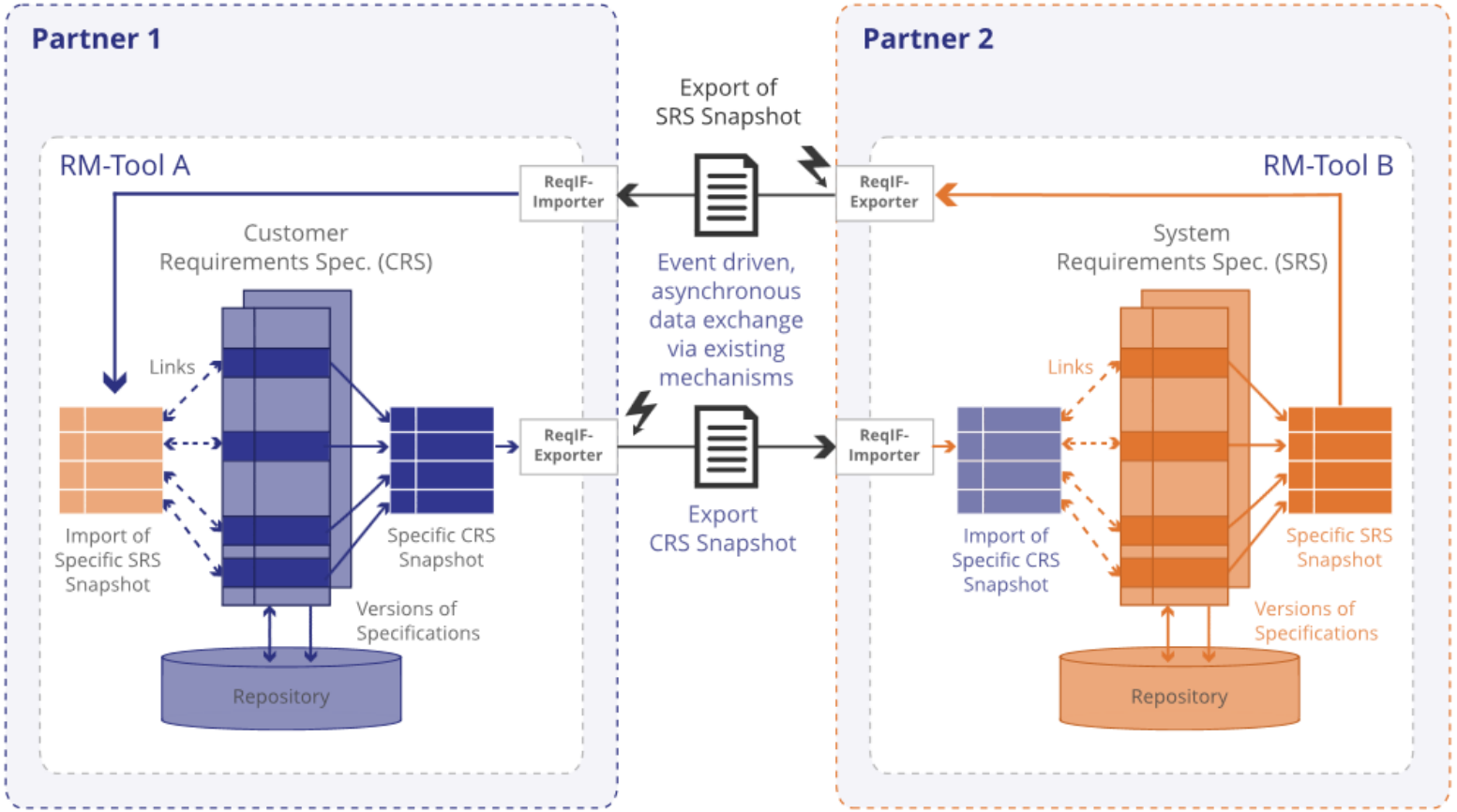
Sample specification authored by using word processor



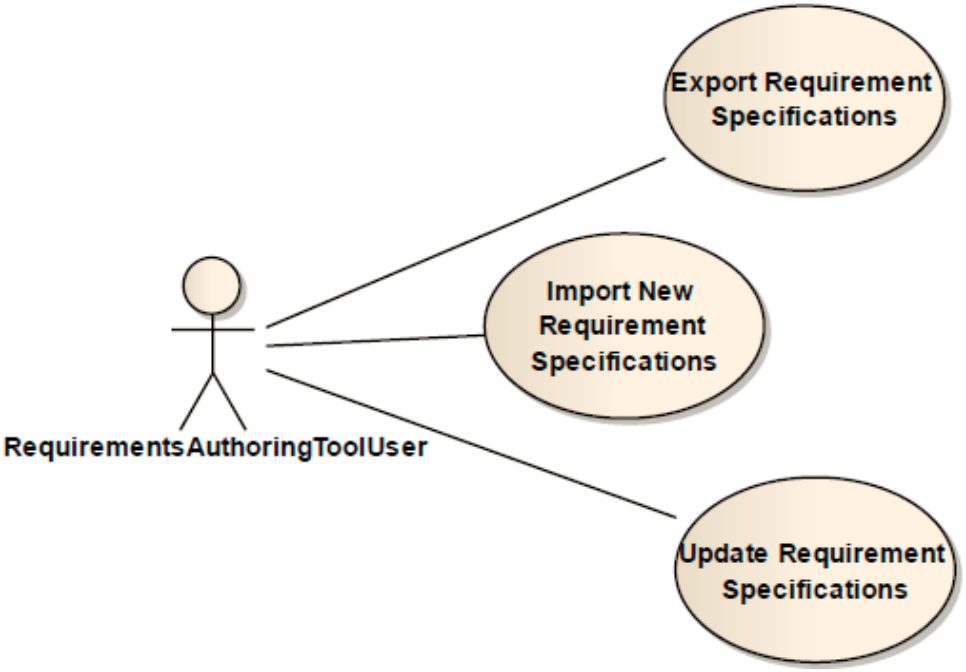
Sample specification authored by using requirements authoring tool

- Formatted text -> structured text
- Uniquely identified requirements
- Tree structure
- Association of attributes with requirements
- Relations between requirements

Concept


Partner 1
RM-Tool A
ReqIF-Importer
Customer Requirements Spec. (CRS)
Links
Import of Specific SRS Snapshot
Specific CRS Snapshot
Versions of Specifications
Repository
ReqIF-Exporter
Export of SRS Snapshot
Event driven, asynchronous data exchange via existing mechanisms
Export CRS Snapshot
Partner 2
RM-Tool B
ReqIF-Exporter
ReqIF-Importer
System Requirements Spec. (SRS)
Links
Import of Specific CRS Snapshot
Specific SRS Snapshot
Versions of Specifications
Repository

Use cases


Export Requirement Specifications
Import New Requirement Specifications
Update Requirement Specifications
RequirementsAuthoringToolUser

Exchange Scenarios

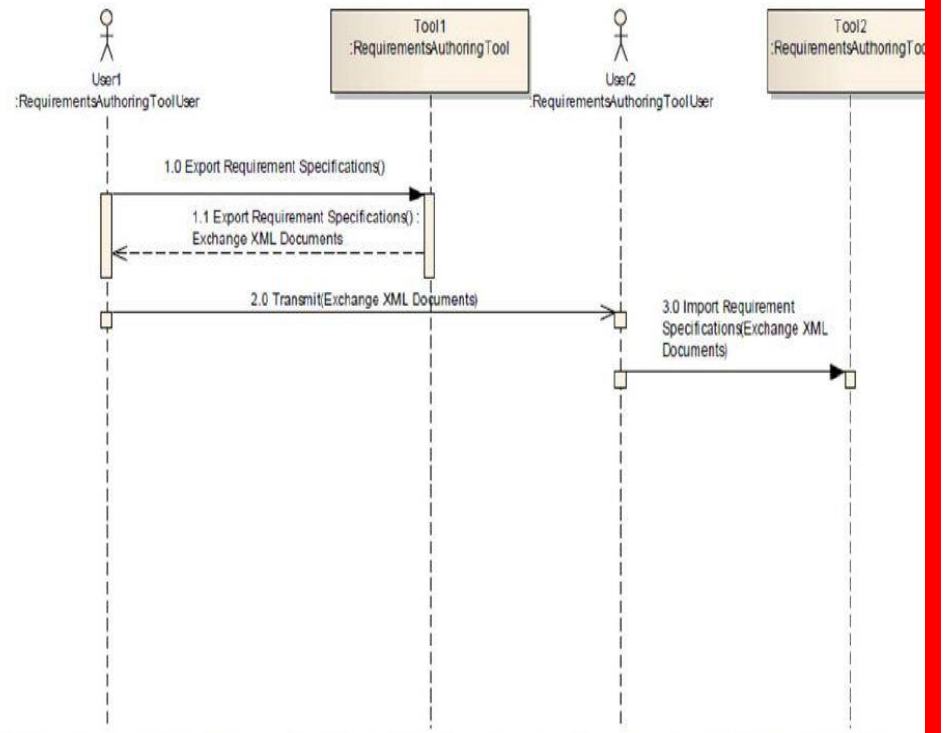


Figure 7.2 - One-Way exchange of requirements between two requirements authoring tools using ReqIF

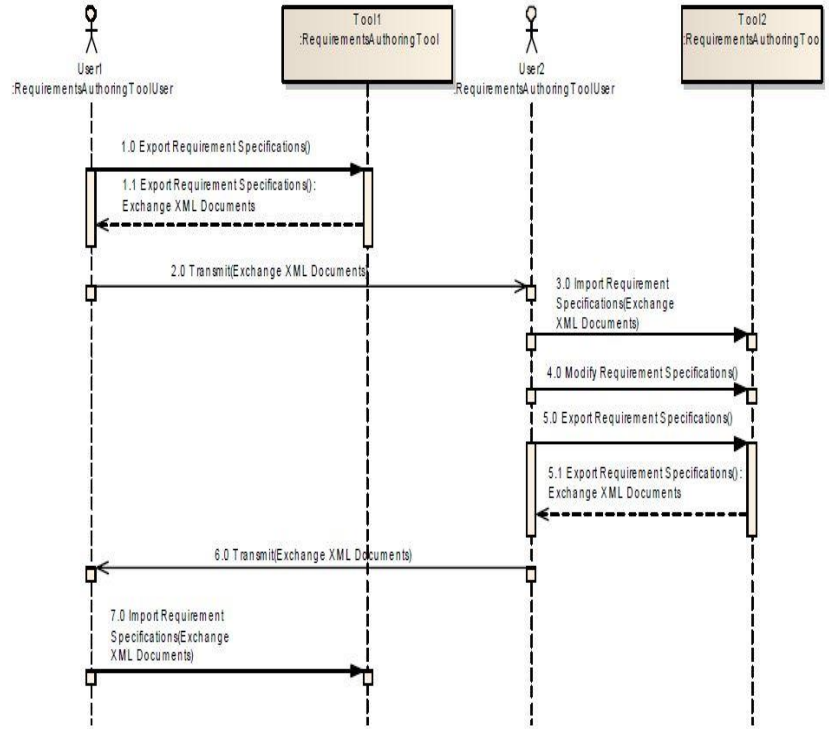
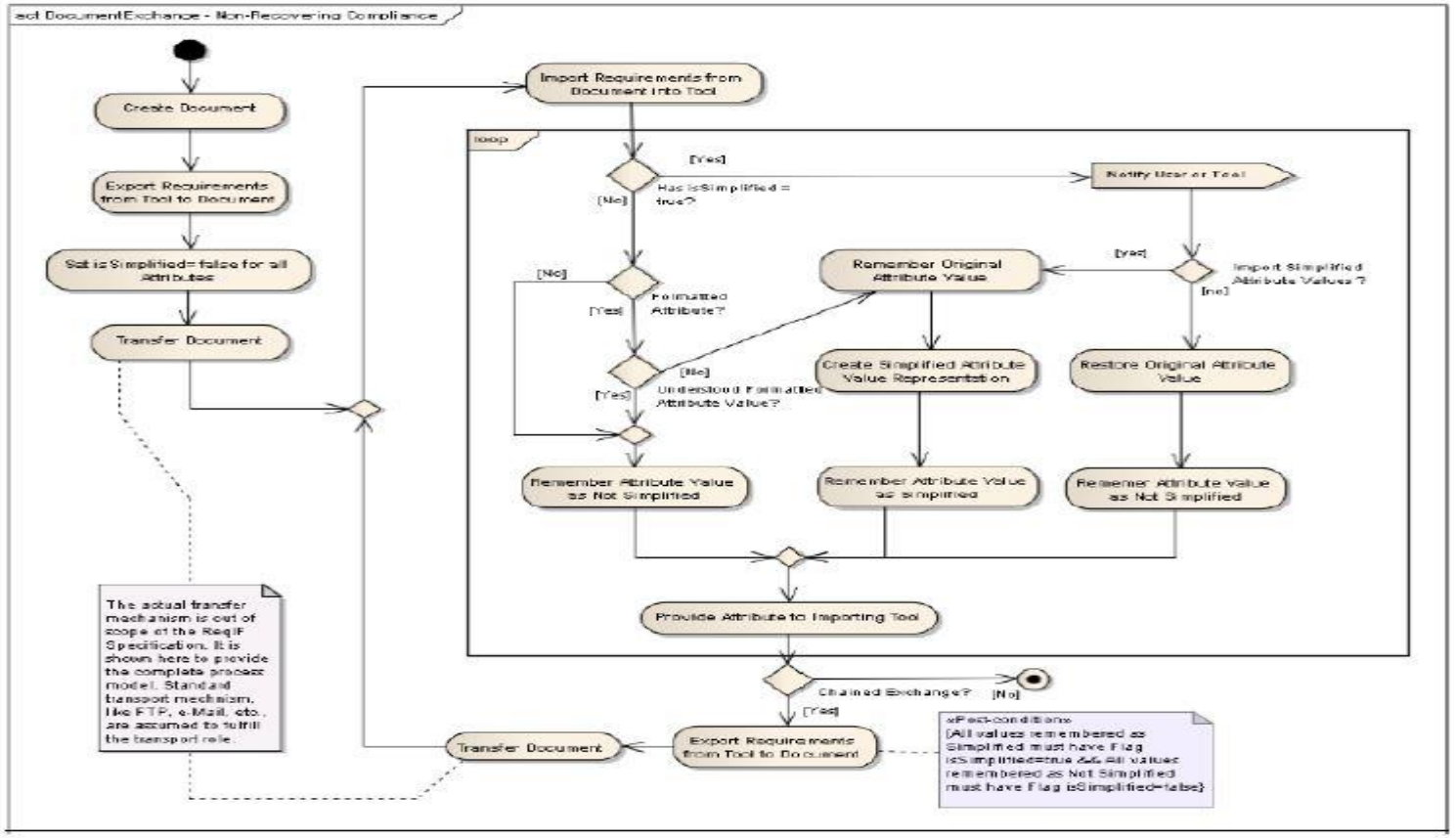


Figure 7.3 - Roundtrip exchange of requirements between two requirements authoring tools using ReqIF

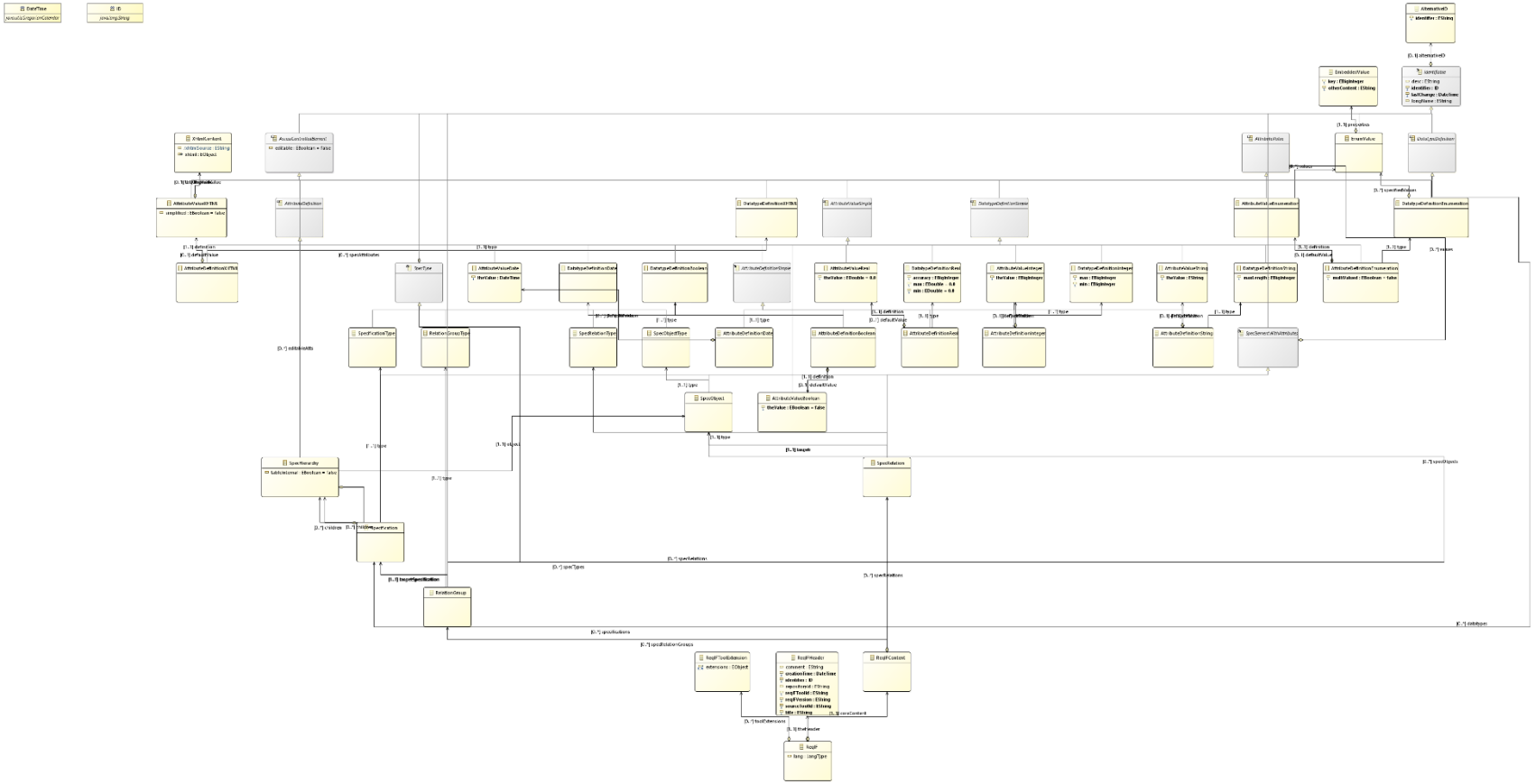
Detailed exchange workflow



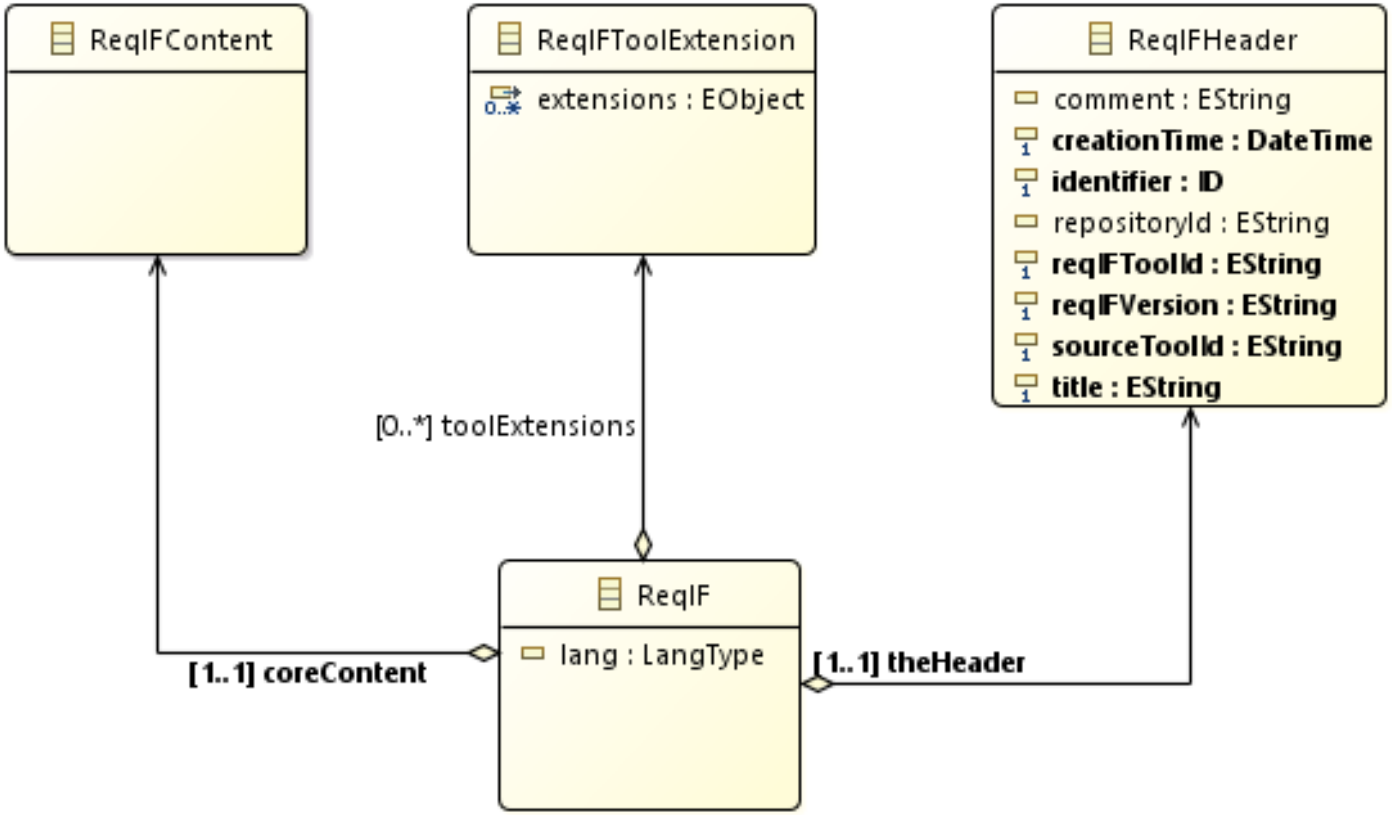
https://reqif.academy/

REQIF STUDIO DEMO

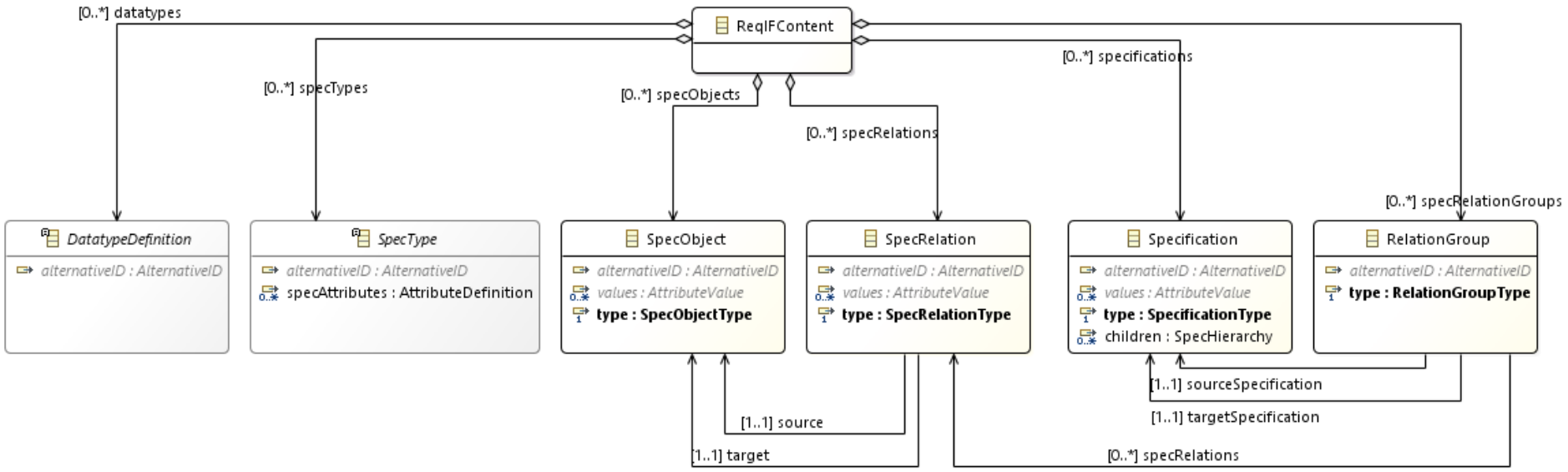
REQIF metamodel



Exchange Document Structure



Exchange Document Content



Unique identification of Elements

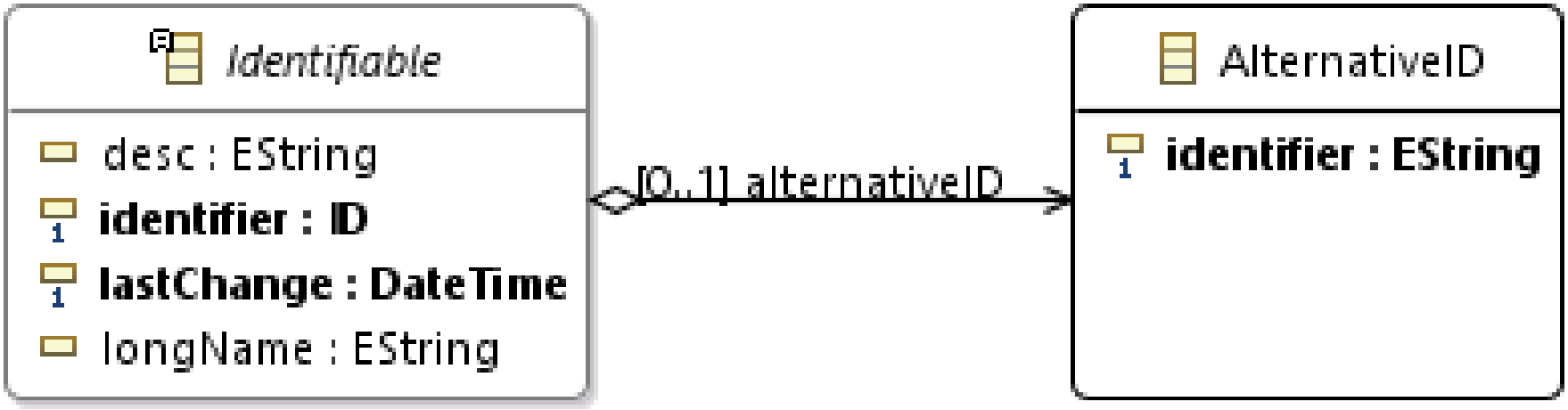
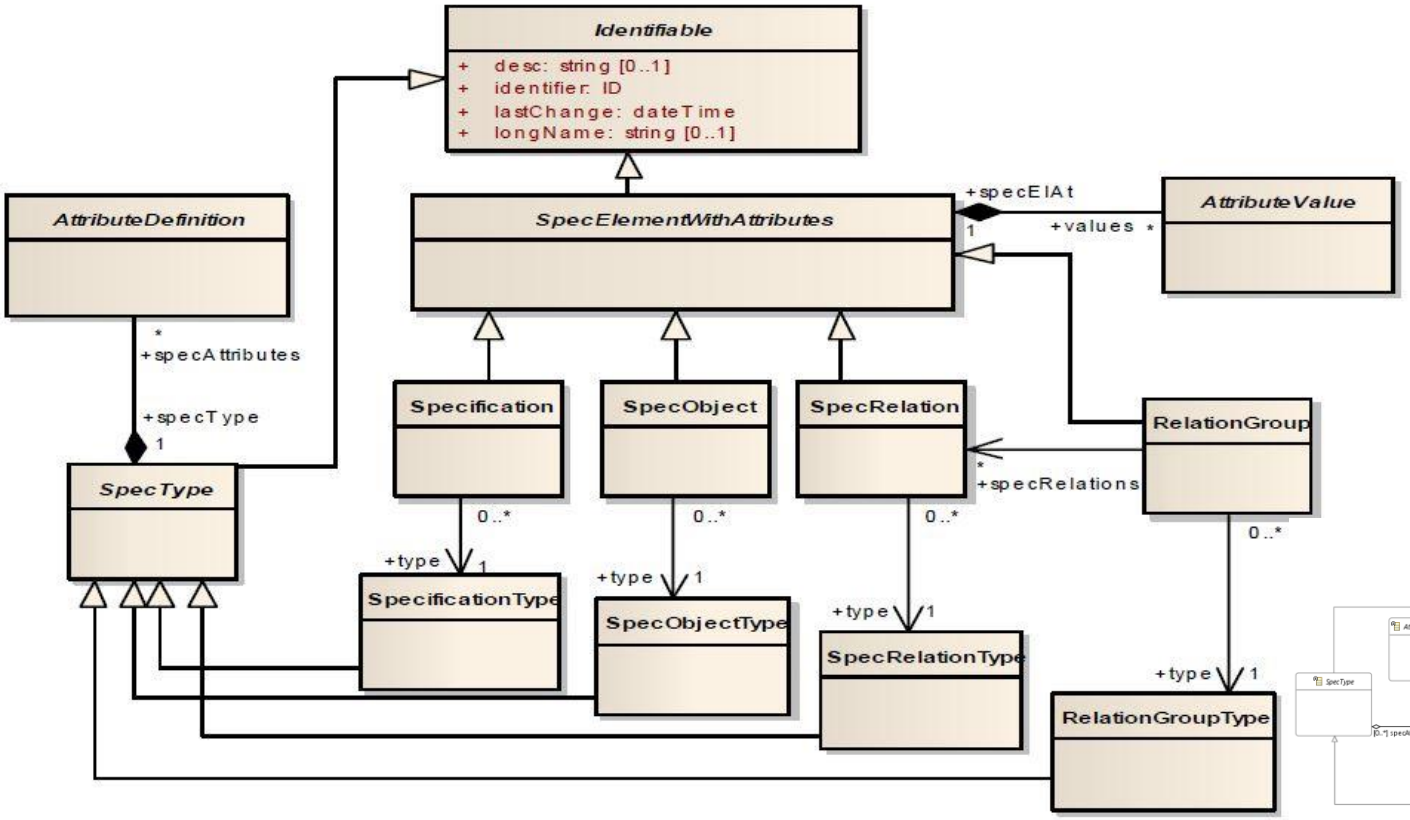
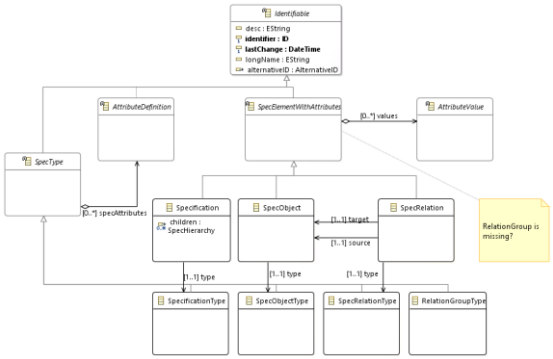


Figure 10.2 – Primary and alternative identifier

Specifications, Requirements, and Attributes





AttributeDefinition class hierarchy

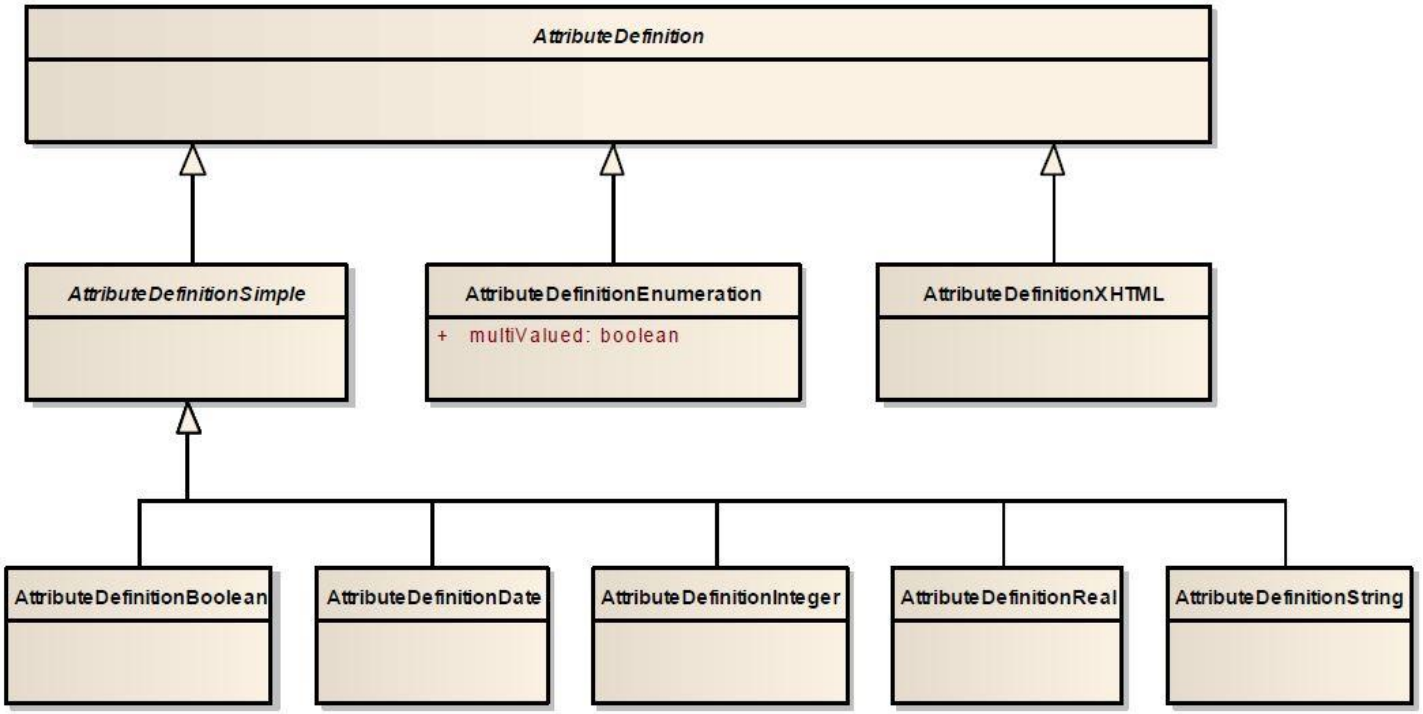
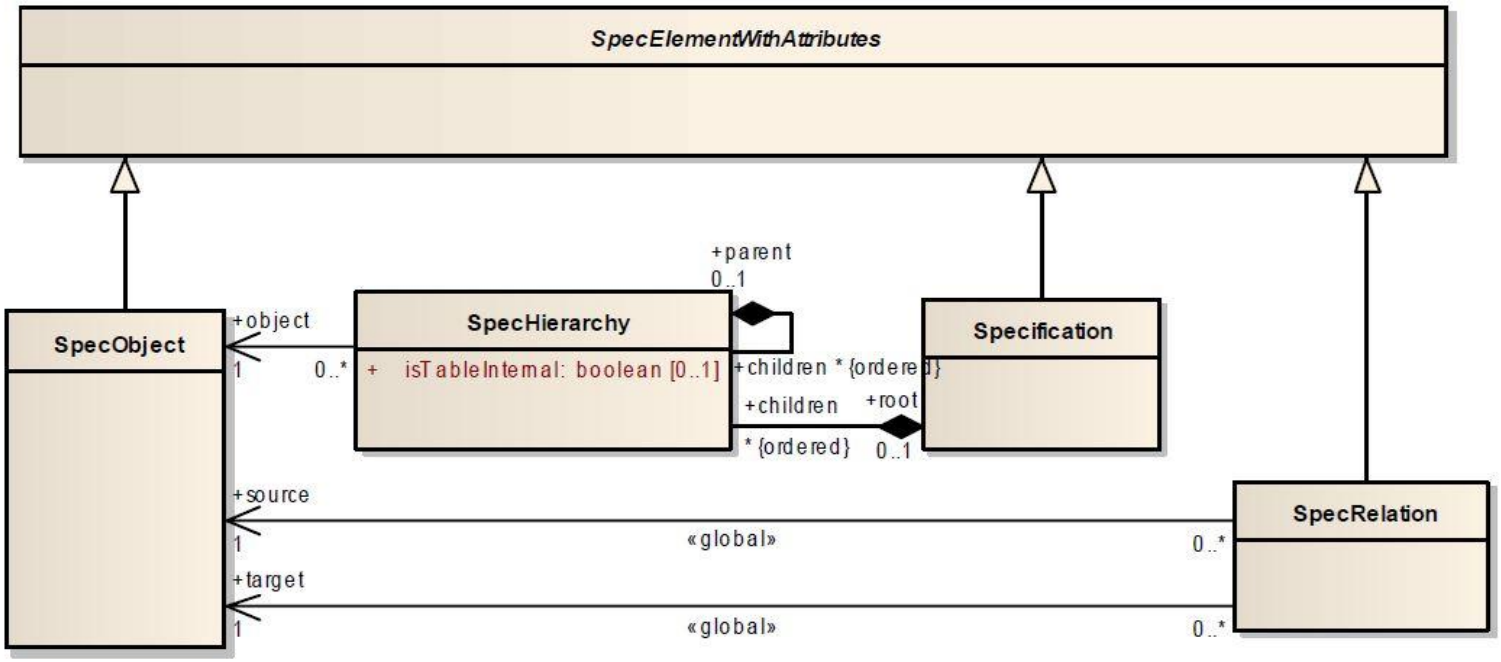


Figure 10.4 - AttributeDefinition class hierarchy

Hierarchy of Requirements and Req. Relations



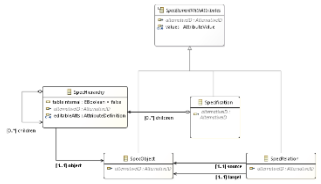
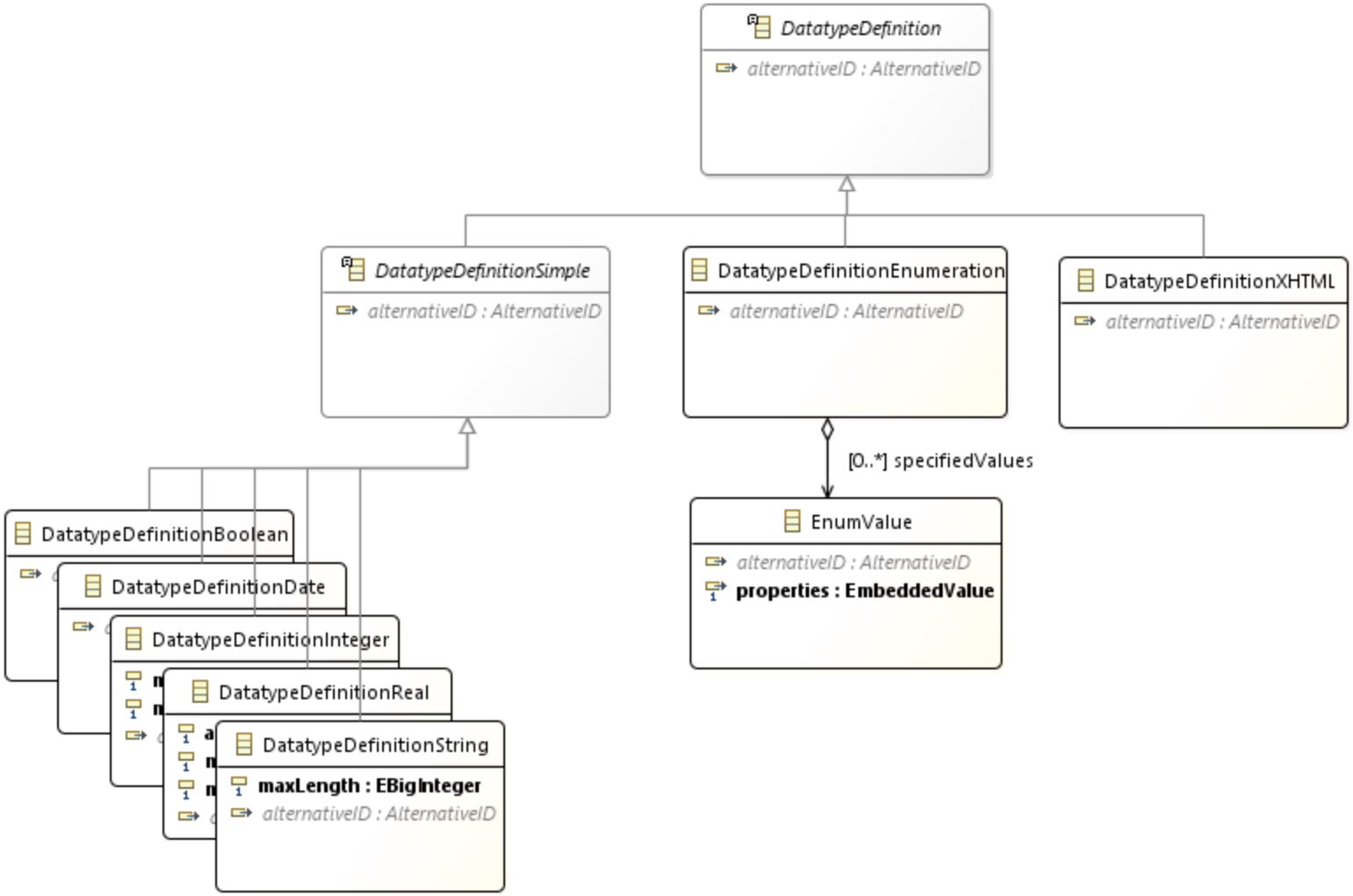
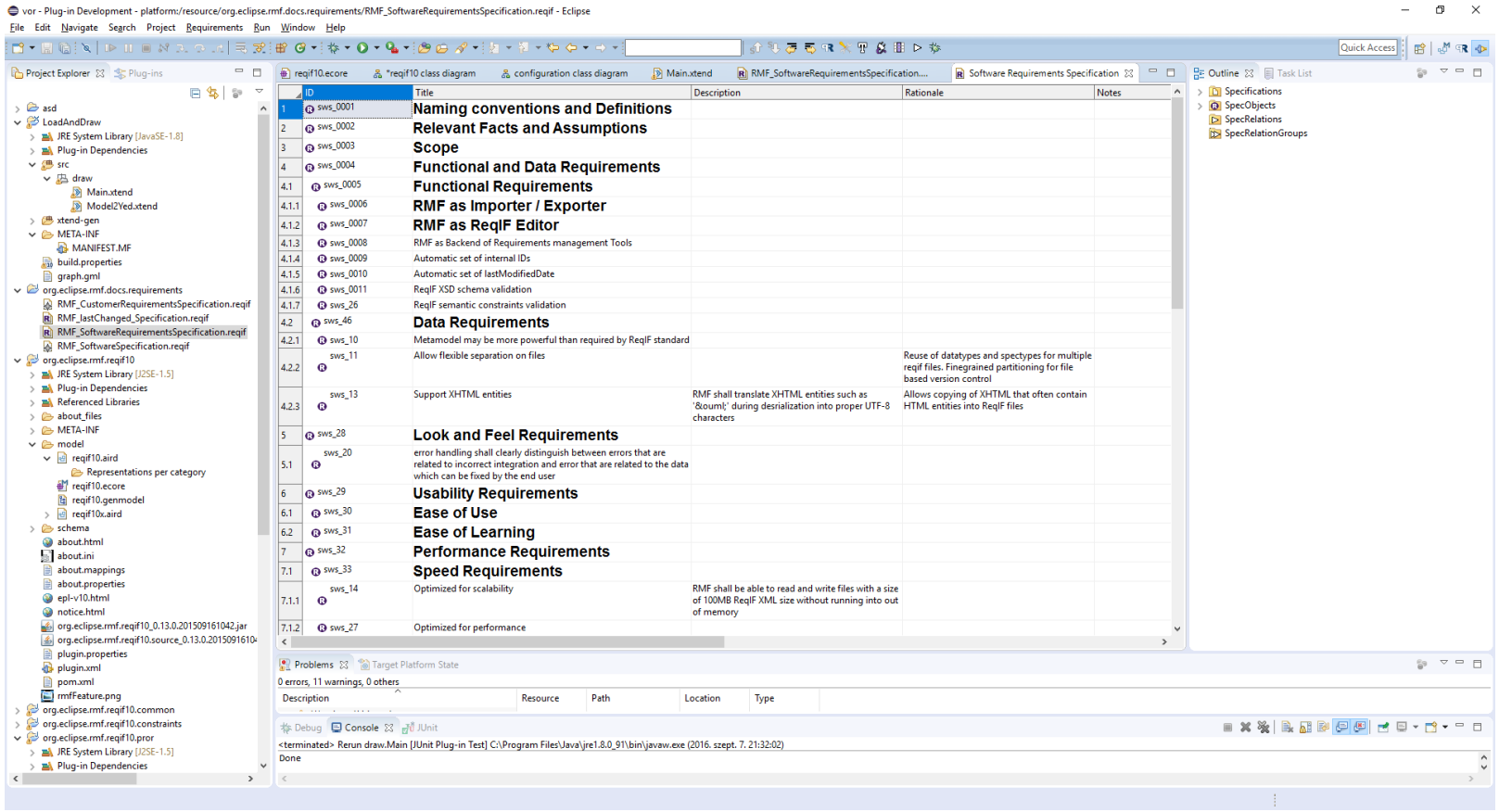


Figure 10.6 - Requirements, requirement relations and how requirements are structured hierarchically in a specification

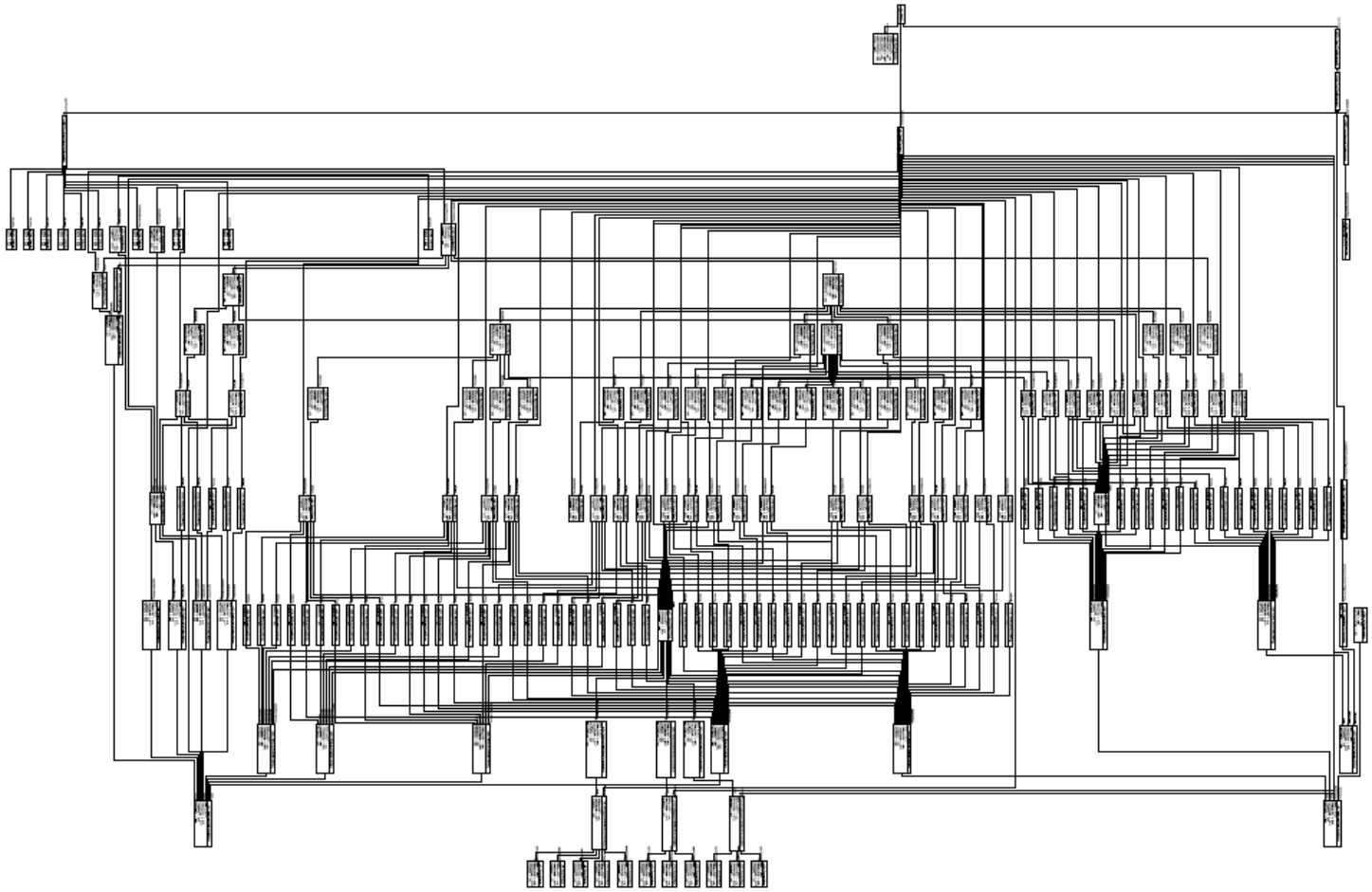
DatatypeDefinition class hierarchy

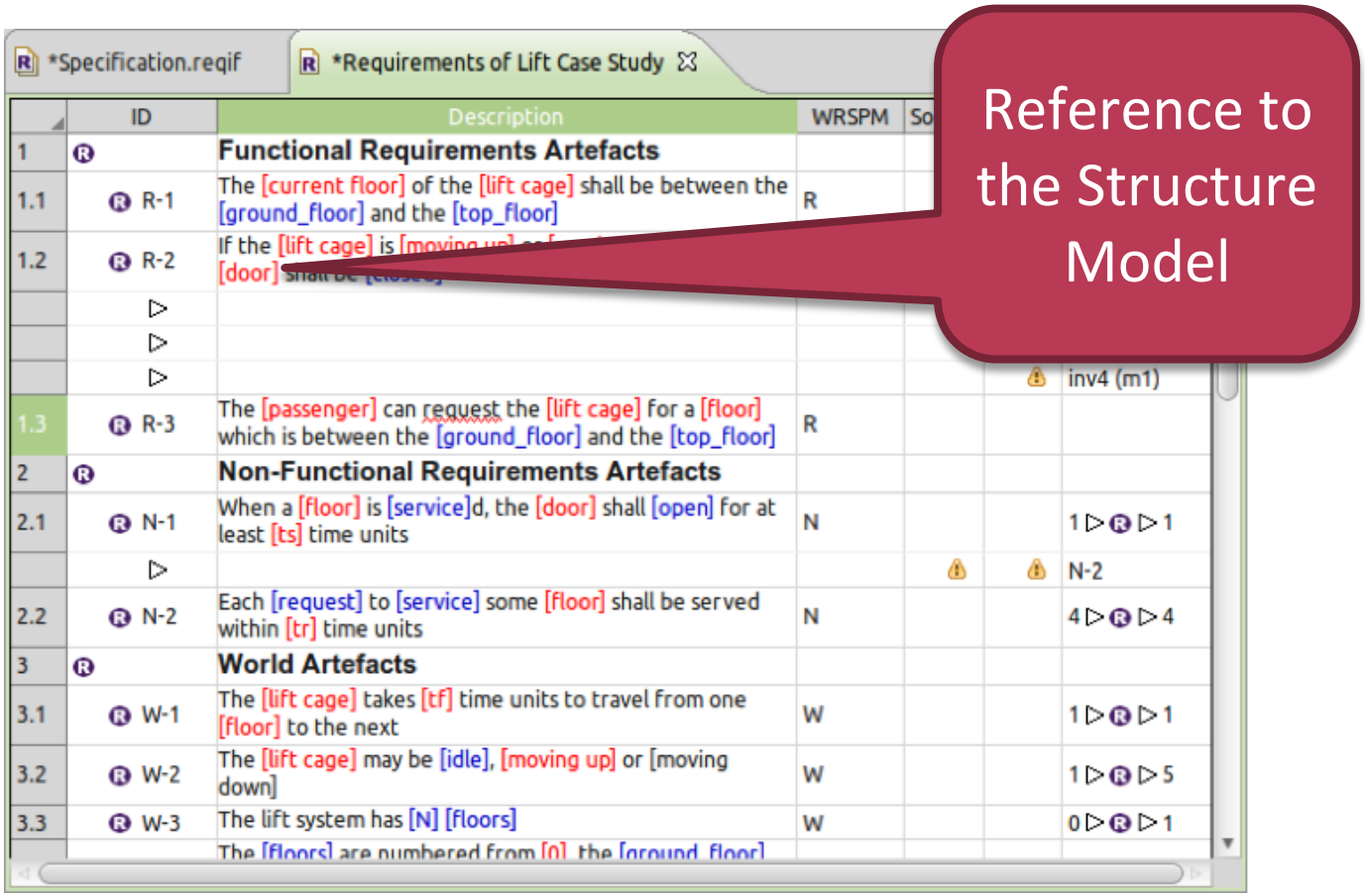


Editor


vor - Plug-in Development - platform:/resource/org.eclipse.rmf.docs.requirements/RMF_SoftwareRequirementsSpecification.reqif - Eclipse
File Edit Navigate Search Project Requirements Run Window Help
Project Explorer
Plug-ins
asd
LoadAndDraw
JRE System Library [JavaSE-1.8]
Plug-in Dependencies
src
draw
Main.xtend
Model2Yed.xtend
xtend-gen
META-INF
MANIFEST.MF
build.properties
graph.gml
org.eclipse.rmf.docs.requirements
RMF_CustomerRequirementsSpecification.reqif
RMF_lastChanged_Specification.reqif
RMF_SoftwareRequirementsSpecification.reqif
RMF_SoftwareSpecification.reqif
org.eclipse.rmf.reqif10
JRE System Library [J2SE-1.5]
Plug-in Dependencies
Referenced Libraries
about_files
META-INF
model
reqif10.aird
Representations per category
reqif10.ecore
reqif10.genmodel
reqif10x.aird
schema
about.html
about.ini
about.mappings
about.properties
epl-v10.html
notice.html
org.eclipse.rmf.reqif10_0.13.0.201509161042.jar
org.eclipse.rmf.reqif10.source_0.13.0.2015091610-
plugin.properties
plugin.xml
pom.xml
rmfFeature.png
org.eclipse.rmf.reqif10.common
org.eclipse.rmf.reqif10.constraints
org.eclipse.rmf.reqif10.pror
JRE System Library [J2SE-1.5]
Plug-in Dependencies
reqif10.ecore
*reqif10 class diagram
configuration class diagram
Main.xtend
RMF_SoftwareRequirementsSpecification....
Software Requirements Specification
ID
Title
Description
Rationale
Notes
1 sws_0001 Naming conventions and Definitions
2 sws_0002 Relevant Facts and Assumptions
3 sws_0003 Scope
4 sws_0004 Functional and Data Requirements
4.1 sws_0005 Functional Requirements
4.1.1 sws_0006 RMF as Importer / Exporter
4.1.2 sws_0007 RMF as ReqIF Editor
4.1.3 sws_0008 RMF as Backend of Requirements management Tools
4.1.4 sws_0009 Automatic set of internal IDs
4.1.5 sws_0010 Automatic set of lastModifiedDate
4.1.6 sws_0011 ReqIF XSD schema validation
4.1.7 sws_26 ReqIF semantic constraints validation
4.2 sws_46 Data Requirements
4.2.1 sws_10 Metamodel may be more powerful than required by ReqIF standard
4.2.2 sws_11 Allow flexible separation on files
Reuse of datatypes and spectypes for multiple reqif files. Finegrained partitioning for file based version control
4.2.3 sws_13 Support XHTML entities
RMF shall translate XHTML entities such as '&oum;' during desrialization into proper UTF-8 characters
Allows copying of XHTML that often contain HTML entities into ReqIF files
5 sws_28 Look and Feel Requirements
5.1 sws_20 error handling shall clearly distinguish between errors that are related to incorrect integration and error that are related to the data which can be fixed by the end user
6 sws_29 Usability Requirements
6.1 sws_30 Ease of Use
6.2 sws_31 Ease of Learning
7 sws_32 Performance Requirements
7.1 sws_33 Speed Requirements
7.1.1 sws_14 Optimized for scalability
RMF shall be able to read and write files with a size of 100MB ReqIF XML size without running into out of memory
7.1.2 sws_27 Optimized for performance
Outline
Task List
Specifications
SpecObjects
SpecRelations
SpecRelationGroups
Problems
Target Platform State
0 errors, 11 warnings, 0 others
Description
Resource
Path
Location
Type
Debug
Console
JUnit
<terminated> Rerun draw.Main [JUnit Plug-in Test] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (2016. szept. 7. 21:32:02)
Done

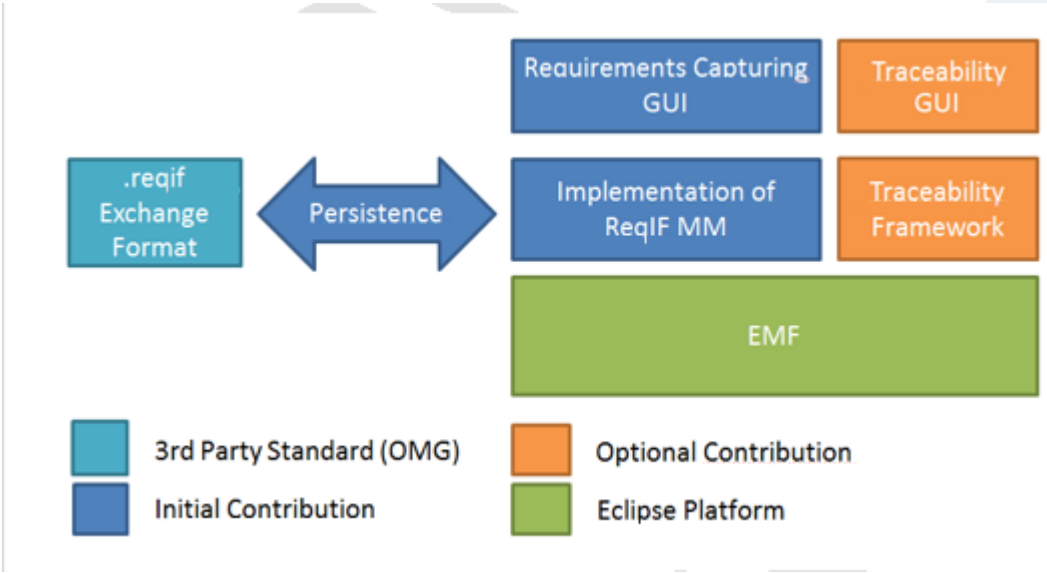
Graph representation ☺



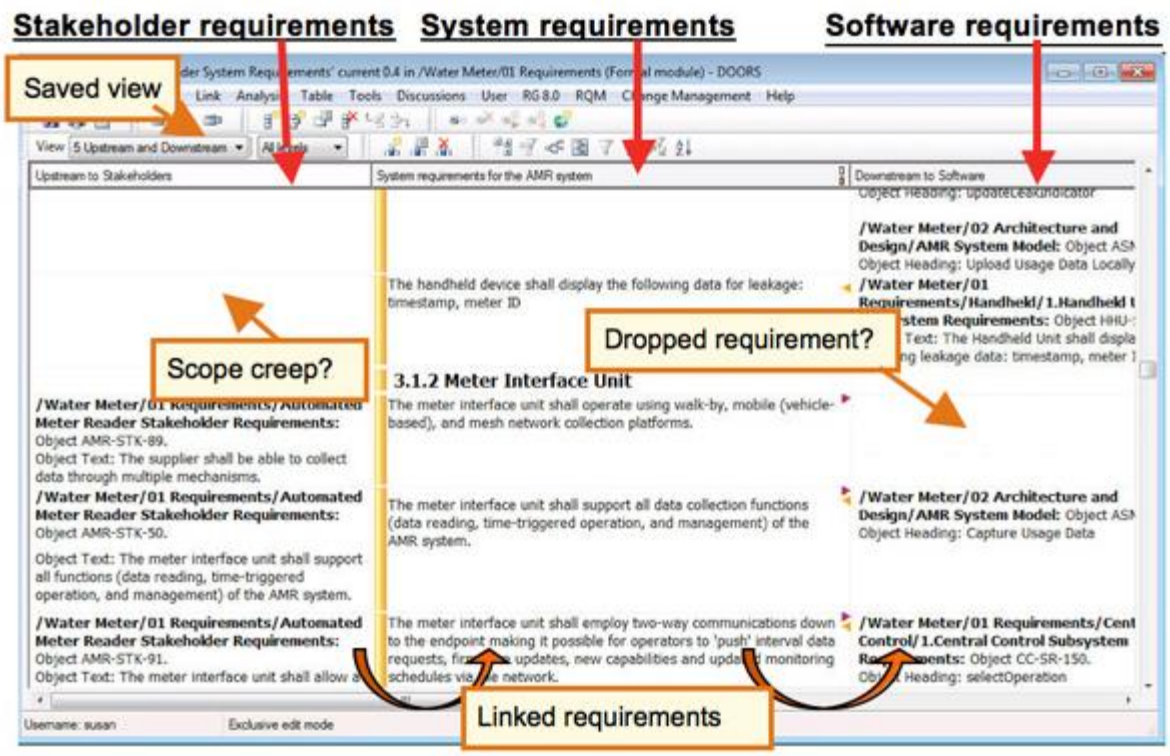

*Specification.reqif
*Requirements of Lift Case Study
ID
Description
WRSPM
1 Functional Requirements Artefacts
1.1 R-1 The [current floor] of the [lift cage] shall be between the [ground_floor] and the [top_floor] R
1.2 R-2 If the [lift cage] is [moving up] [door]
inv4 (m1)
1.3 R-3 The [passenger] can request the [lift cage] for a [floor] which is between the [ground_floor] and the [top_floor] R
2 Non-Functional Requirements Artefacts
2.1 N-1 When a [floor] is [service]d, the [door] shall [open] for at least [ts] time units N 1 ▷ R ▷ 1
N-2
2.2 N-2 Each [request] to [service] some [floor] shall be served within [tr] time units N 4 ▷ R ▷ 4
3 World Artefacts
3.1 W-1 The [lift cage] takes [tf] time units to travel from one [floor] to the next W 1 ▷ R ▷ 1
3.2 W-2 The [lift cage] may be [idle], [moving up] or [moving down] W 1 ▷ R ▷ 5
3.3 W-3 The lift system has [N] [floors] W 0 ▷ R ▷ 1
Reference to the Structure Model

Eclipse ProR

- http://www.eclipse.org/rmf/pror/



Traceability view



A professional and expensive tool…