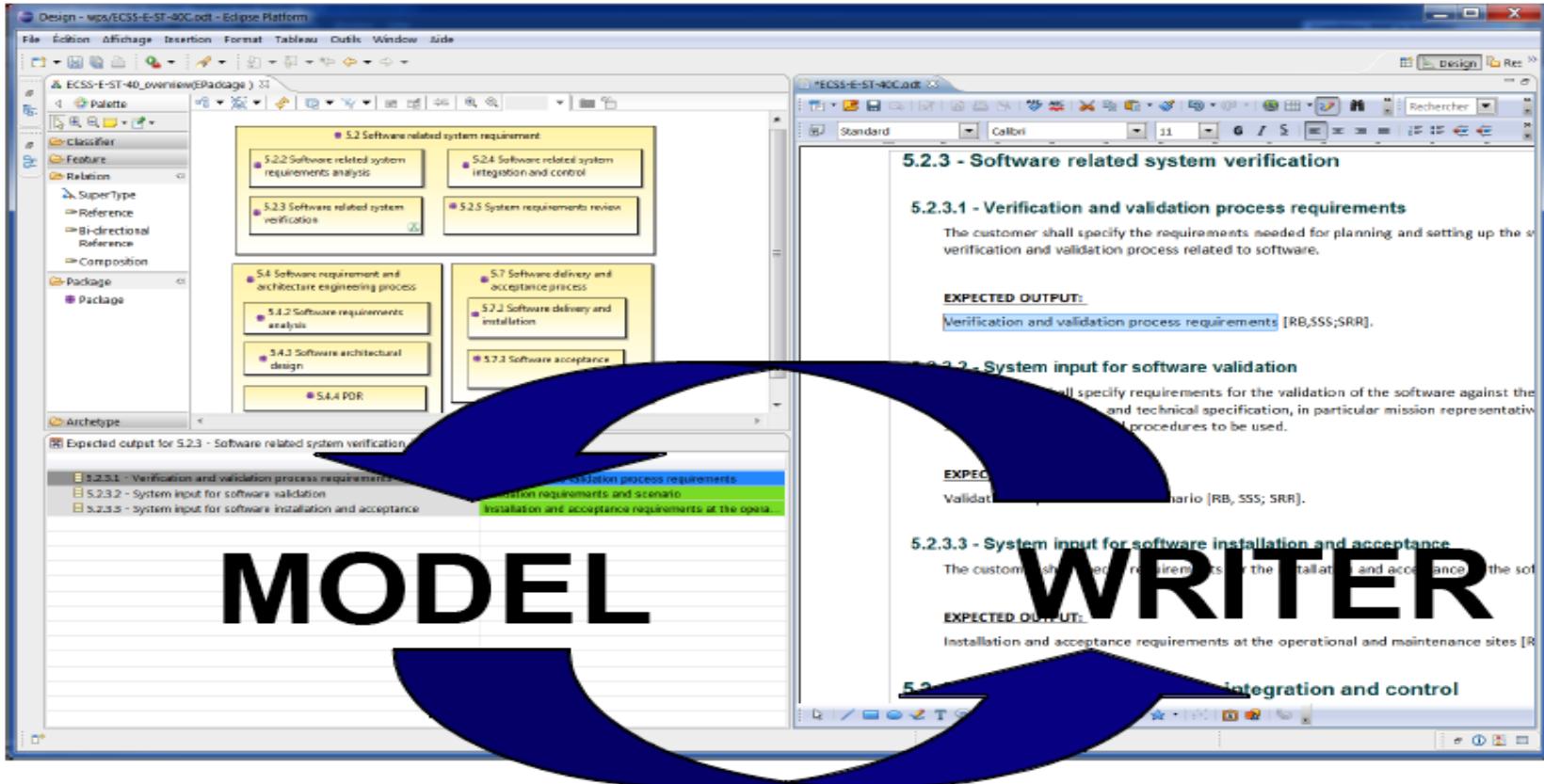


ModelWriter

Text & Model-Synchronized Document Engineering Platform



Project Leader: Ferhat Erata (ferhat@computer.org)

Project Email: project@modelwriter.eu

1 Introduction

Ferhat Erata

UNIT, ModelWriter Project Leader

Participants

ModelWriter #1 Review



- Ferhat Erata
- Dr. Moharram Challenger



- Prof. Claire Gardent,
- Prof. Samuel Cruz-Lara
- Dr. Mariem Mahfoudh



- Dr. Anne Monceaux
- Dr. Eray Tuzun



- Yvan Lussaud



- Prof. Geylani Kardas
- Hale Gezgen



- Prof. Erhan Mengüsoğlu



- Ersan Gürdoğan
- Taskin Kızıl

Agenda [9:00 - 12:30]

1. Introduction (10 min) [09:00 - 09:10]
2. Overview of the project (20 min) [09:10 - 09:30]
3. Use-cases and exploitation prospects (45 min) [09:30 - 10:15]
4. Progress Status per WP (45 min) [10:15 - 11:00]
 - Break (10 mins)
5. Demonstrations (30 min): [11:10 - 11:40]
6. Summary (achievements and exploitations) (5 min) [11:40 - 11:45]
7. Reviewers Private Section (30 min) [11:45 - 12:15]
8. Feedback Session (15min) [12:15 - 12:30]
9. Lunch (30 min) [12:30 - 13:00]

2 Overview of the Project

Ferhat Erata

UNIT, ModelWriter Project Leader

ModelWriter

Text & Model-Synchronized Document Engineering Platform

The project envisions an integrated authoring environment called "ModelWriter" for Technical Authors (such as Software or Systems Engineers etc.) which will combine a Semantic Word Processor (= the "Writer" part) and a Knowledge Capture Tool (= the "Model" part).

Project information

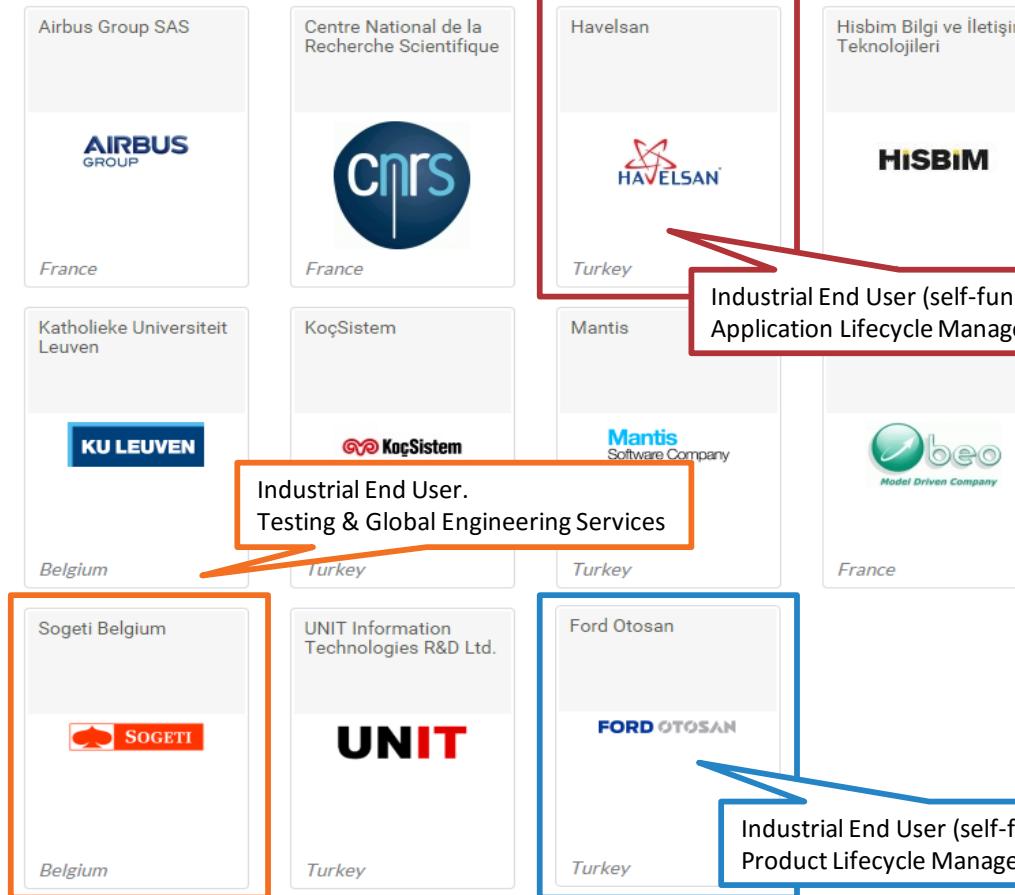
Project name	13028 ModelWriter
Status	Running
Period	Oct 2014 - Sep 2017
Call	ITEA 2 Call 8
Challenge	Knowledge-based society
Website	www.modelwriter.eu
Partners	10
Countries	Belgium France Turkey

Project leader



Name
Ferhat Erata
Organisation
UNIT Information Technologies R&D Ltd.
Country
Turkey
Project involvement
13028 ModelWriter

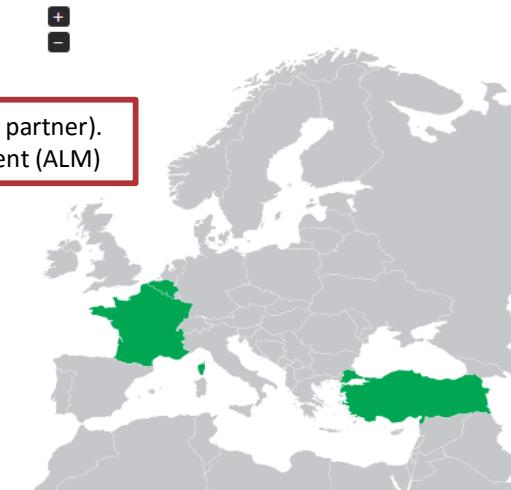
Project partners



Project documents

Project publications

- ITEA Annual report 2013 published online
- ModelWriter Posters Co-summit 2015
- ModelWriter Project Leaflet



Industrialization Triangle in ModelWriter

Open Source Software



UNIT



Mantis
Software Company

Tool Providers:
Commercialization
- New Product & Services
Standardization
- Open Source Software



Products
&
Expertise

SME

Industrialization

Technology
Transfer

Industrial Use Cases
Success Stories
Long Term Agreements

Large
Organization

Inject
Requirements

ModelWriter

Prototypes

Researchers

Innovation

Technology Providers:
New Standard or Standard Extension
Publications, Open Source Software

AIRBUS
GROUP



KoçSistem

SOGETI

HISBIM

FORD OTOSAN



ModelWriter

Project Overview



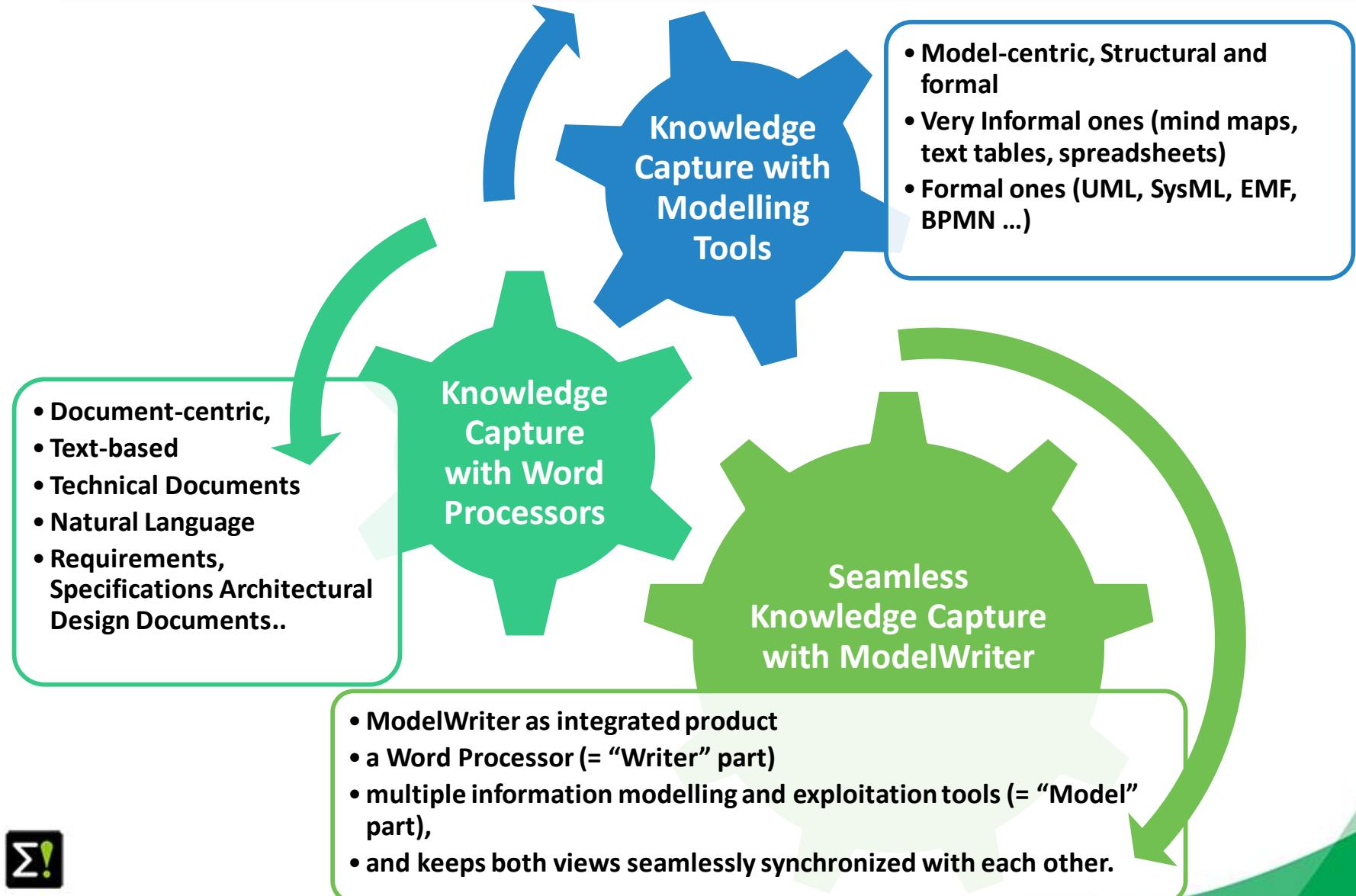
Resource Allocation: 68,71 person year

Project Duration: 36 months

Planned Budget: 5,543,000 Euro

Start and Finish Date: 01 Oct 2014 – 30 Sep 2017

Open-Source Software Platform to be submitted to Eclipse Foundation



ModelWriter

bi-directional Knowledge Capture tool



The screenshot displays the ModelWriter application interface. On the left, a model editor window titled "Design - mps/ECSS-E-ST-40C.otp - Eclipse Platform" shows a hierarchical structure of requirements. The main tree node is "ECSS-E-ST-40C_overview(EPackage)". Under it, there are several packages and their contents:

- 5.2 Software related system requirement
 - 5.2.2 Software related system requirements analysis
 - 5.2.4 Software related system integration and control
 - 5.2.3 Software related system verification
 - 5.2.5 System requirements review
- 5.4 Software requirement and architecture engineering process
 - 5.4.2 Software requirements analysis
 - 5.4.3 Software architectural design
 - 5.4.4 PDR
- 5.7 Software delivery and acceptance process
 - 5.7.2 Software delivery and installation
 - 5.7.3 Software acceptance

Below the model editor, a large blue circle contains the word "MODEL".

On the right, a document viewer window titled "ECSS-E-ST-40C.otp" is open, displaying the "5.2.3 - Software related system verification" section. The section includes:

- 5.2.3.1 - Verification and validation process requirements**:
The customer shall specify the requirements needed for planning and setting up the verification and validation process related to software.
- EXPECTED OUTPUT:**
Verification and validation process requirements [RB;SSS;SRR].
- 5.2.3.2 - System input for software validation**:
The customer shall specify requirements for the validation of the software against the functional and technical specification, in particular mission representative scenarios and procedures to be used.
- EXPECTED OUTPUT:**
Validation requirements for the validation scenario [RB; SSS; SRR].
- 5.2.3.3 - System input for software installation and acceptance**:
The customer shall specify requirements for the installation and acceptance of the software.
- EXPECTED OUTPUT:**
Installation and acceptance requirements at the operational and maintenance sites [RB; SSS; SRR].

Below the document viewer, another large blue circle contains the word "WRITER".

Semantic Word Processor (Text-Based Knowledge Extractor)

Understands the various textual parts of a document expressed in Natural Language

Reveals concepts and relationships between them (“Model”-part)

Consistency & Completeness Checking

“Everywhere” Document Regeneration: “tell once, show everywhere”: recycling knowledge from (1) the same document, or of (2) another related document

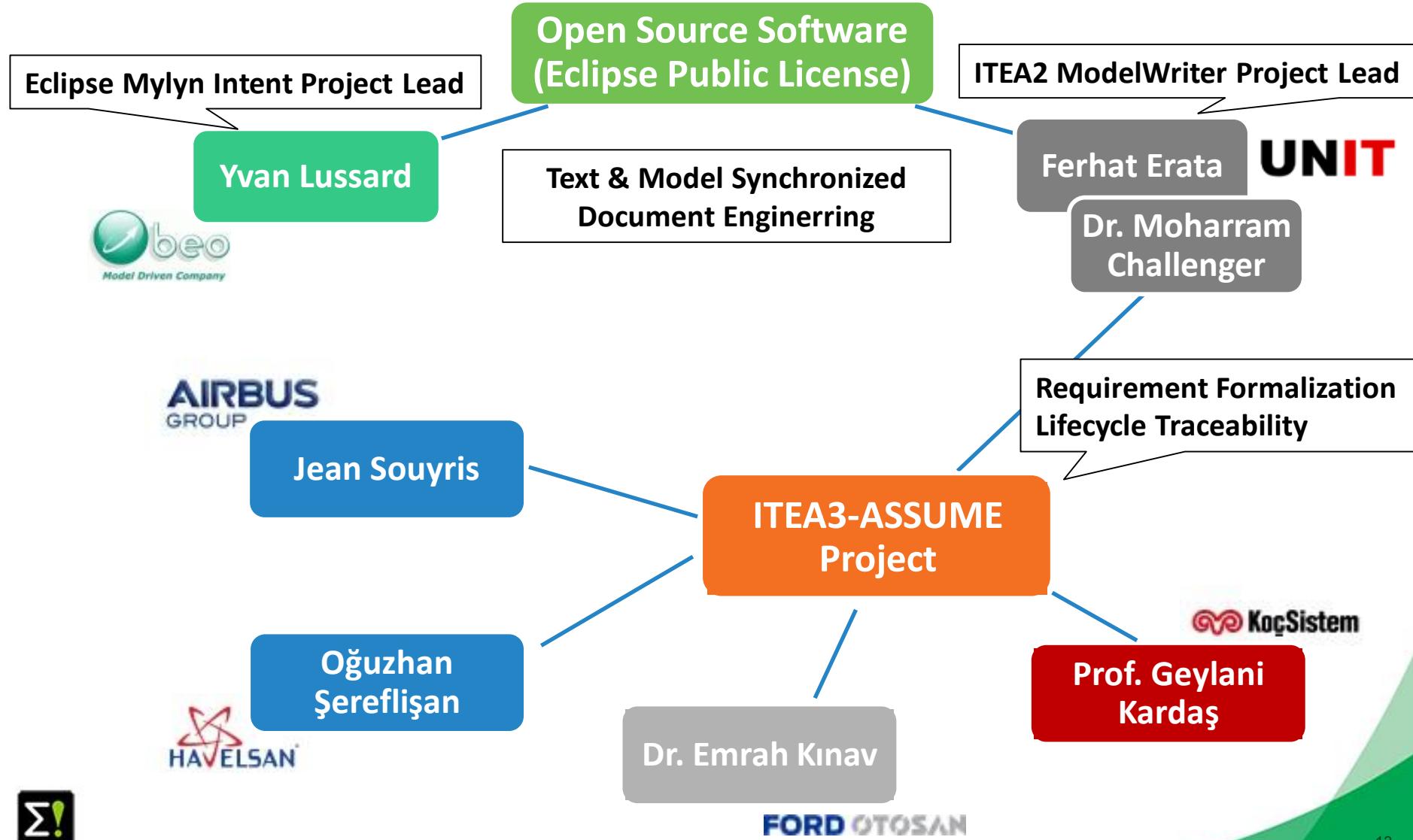
Consistency Checking: the objective to provide a Consistency Checker that automates Quality Reviews of Requirements Engineering

Open Source Software under Eclipse Foundation for Future Dissemination and Exploitation to further extend the Business Value Chain

“MW” Knowledge Dissemination Standard (.mw ModelWriter exchange format)

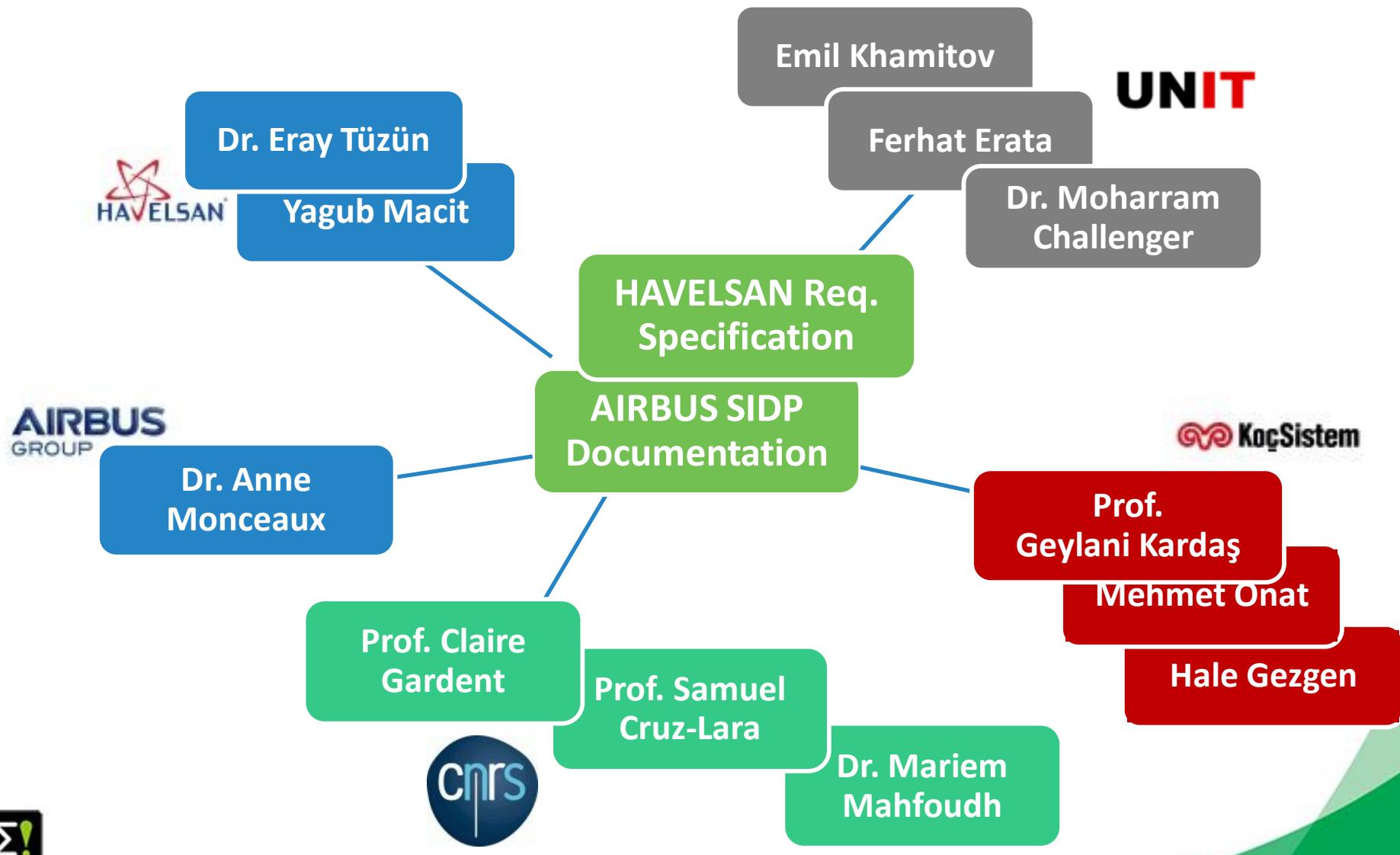
Level of Collaboration within ModelWriter

Exploitation of MW in Eclipse & ASSUME



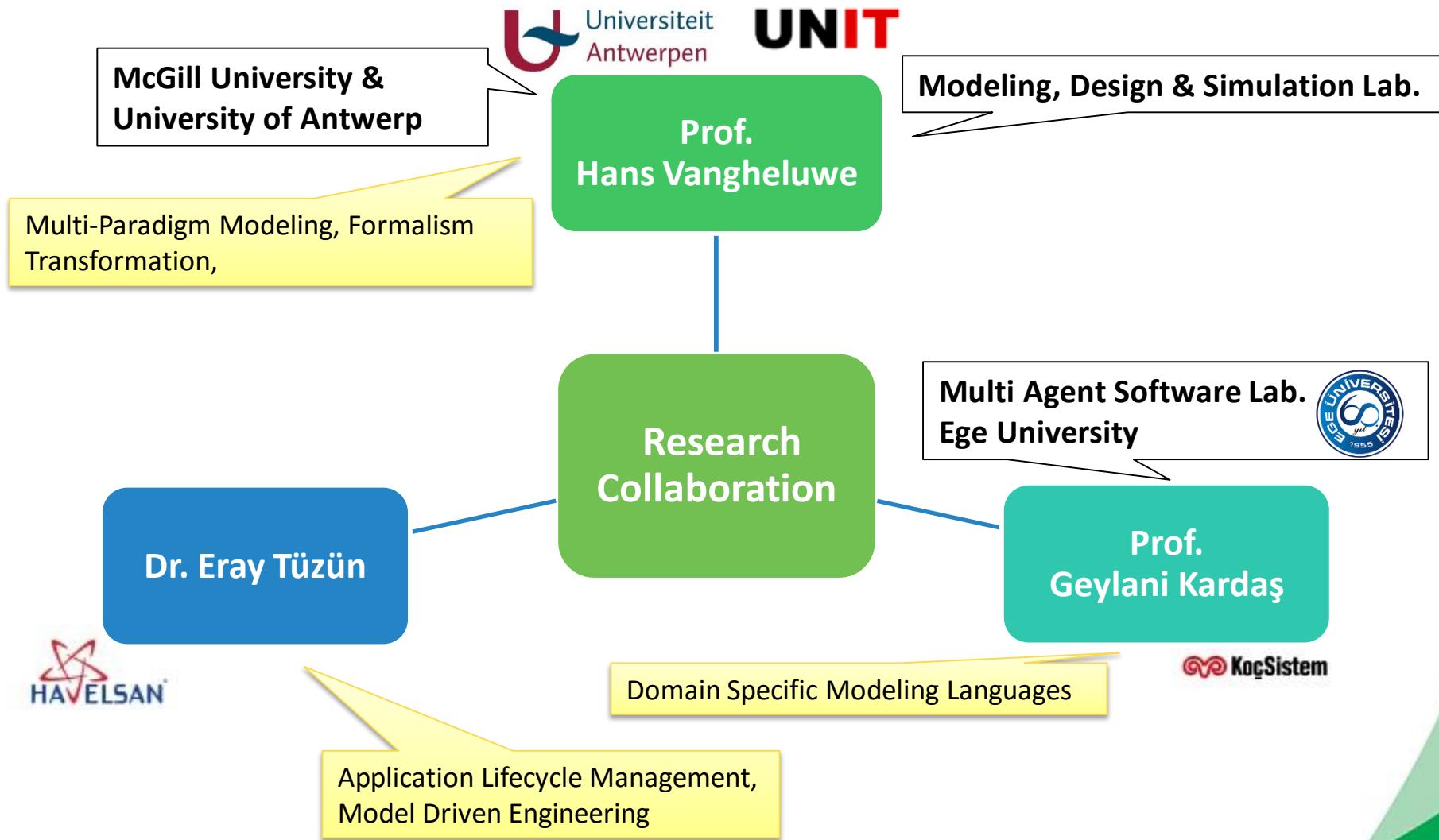
Level of Collaboration within ModelWriter

International Collaboration (through UCs)



Level of Collaboration within ModelWriter

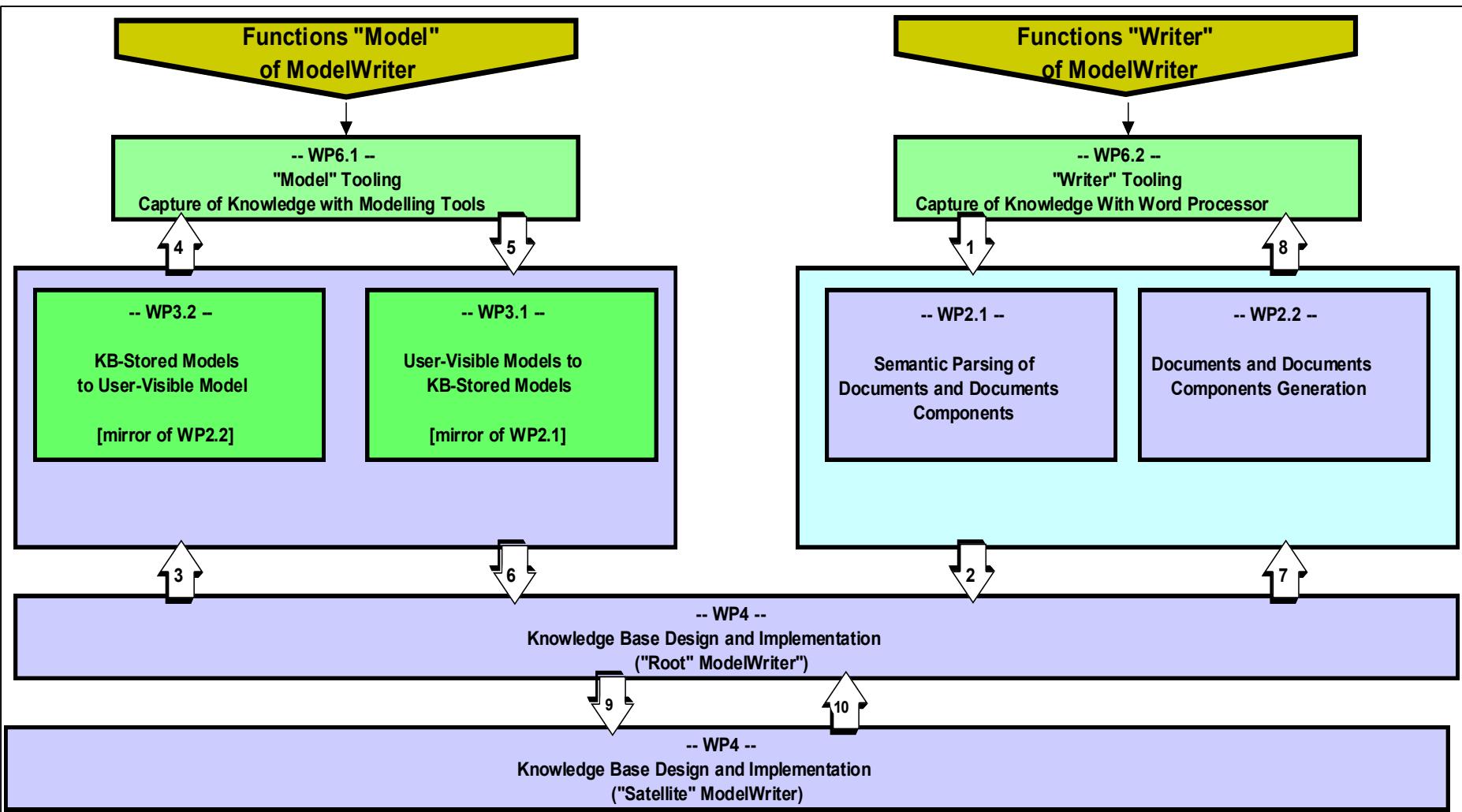
International Collaboration (research)



Technoloaical components & interactions

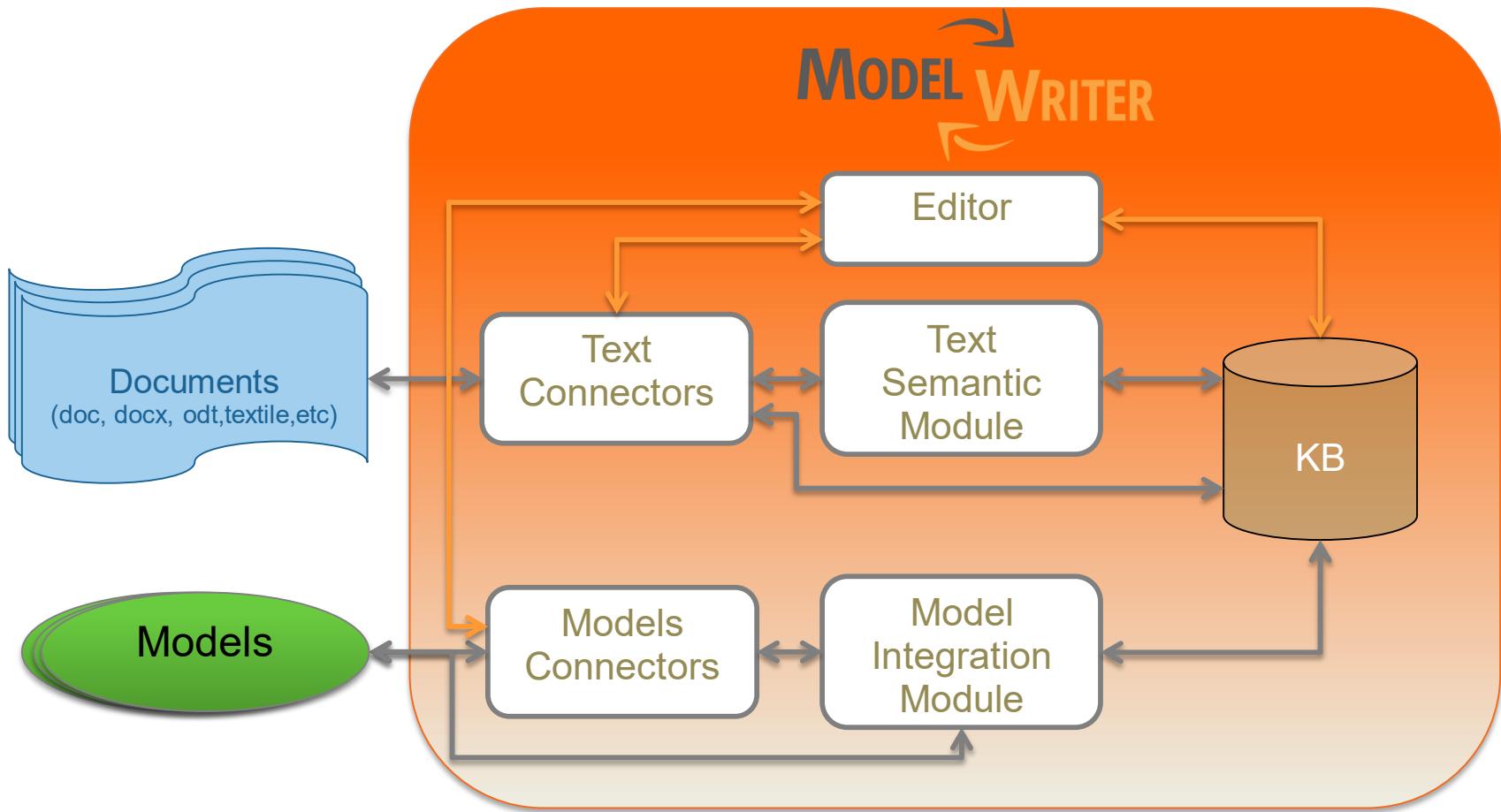


Text & Model-Synchronized Document
Engineering Platform (ModelWriter-ITEA-2013)
-- VALUE CHAIN --
v8.0.0 dated 13-May-2013



ModelWriter

Conceptual Architecture

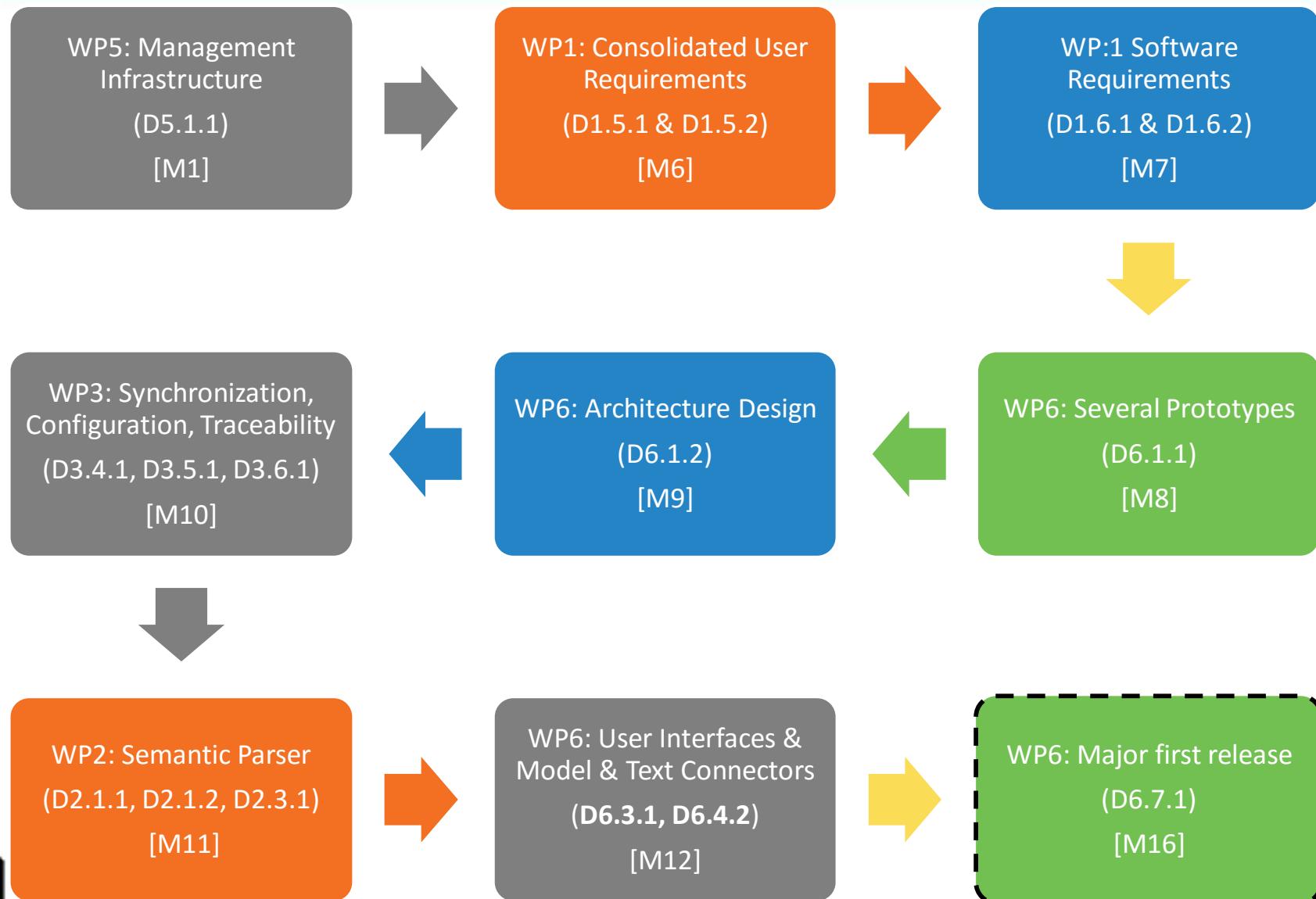


WP1 Industrial Use Cases and Requirements (AIRBUS)

WP2 (LORIA)	WP3 (UNIT)	WP4 (MANTIS)	WP6 (OBEO)
<ul style="list-style-type: none">• Semantic Parser• Document Generation• bi-directional transformation between text and formal knowledge representation	<ul style="list-style-type: none">• Bi-directional synchronization mechanism between texts and models• Configuration & Traceability Components• Consistency checker plug-in for consistency	<ul style="list-style-type: none">• A federated Knowledge Base and its API• Synchronization mechanism between texts/models & knowledge base	<ul style="list-style-type: none">• A complete “ModelWriter” tool integrating of all these in a consistent way• User Interfaces

WP5 Project Management (UNIT)

WP7 Standardization, Dissemination and Exploitation (OBEO)



ModelWriter Workshops in the First Year



<https://github.com/modelwriter/workshops>

Project Kick-off in Istanbul, Turkey (Nov 08, 2014) [M1]

Initial Architectural Design, Industrial Use Cases, Technical WP discussions

Collaboration Infrastructure

The 1st International ModelWriter Workshop in Izmir, Turkey (Jan 15-17, 2015) [M4]

Exploitation: Havelsan's participation

The 1st International Eureka Project Exhibition in Berlin, Germany (Mar 10-11, 2015) [M6]

Consolidated User Requirements & Review

The 2nd International ModelWriter Workshop in Brussels, Belgium (Apr 30, 2015) [M7]

Software Requirements Review & Architecture

The 3rd International ModelWriter Workshop in Toulouse, France (Jun 22-23, 2015) [M10]

Integration of software components

The 4th International ModelWriter Workshop in Brussels, Belgium (Sep 23-24, 2015) [M12]

The 5th International ModelWriter Workshop in Ludwigsburg, Germany (Nov 2-5, 2015) [M16]

Open Source Campaign

Open Call for Industrial User Stories

- *Shape the future ModelWriter*
- *Early adaptation of the technology*
- *Long Term Support*

ModelWriter Open Source Campaign

<https://github.com/modelwriter>



Screenshot of the GitHub repository page for the ITEA2-ModelWriter Project.

ITEA2-ModelWriter Project
Text & Model-Synchronized Document Engineering Platform
Europa | <http://www.modelwriter.eu> | project@modelwriter.eu

Repositories (highlighted) | **People** 25 | **Teams** 27 | **Settings**

Filters | | **+ New repository**

WP5
Work Package 5 - Project Management (UNIT)
Updated 2 hours ago

WP3
Work Package 3 - Model to/from Knowledge Base (UNIT)
Updated 3 days ago

Workshops
Repository dedicated to workshops
Updated 3 days ago

Deliverables
Project Monitoring, Tracking, Communication Management and Infrastructure
Updated 5 days ago

People 25 >

A grid of 25 user profile pictures and four large, stylized icons representing teams or groups: a person's head, a green 'T', a purple 'H', and a pink 'X'.

Invite someone

ModelWriter Requirements & User Stories

<https://waffle.io/modelwriter/requirements>



modelwriter/requirements

Add Issue

Filter Board

User Stories 13

Confirmed 3

In Progress 0

Done 2

14 The TRAM will be able to compose transformations.
Software Requirements Document (SRD)
WP3 - Model to/from Knowledge Base

13 The TRAM validates such parameters and also the input models before a transformation takes place.
Software Requirements Document (SRD)
WP3 - Model to/from Knowledge Base

12 A mechanism is needed to register the available transformations in ModelWriter
Software Requirements Document (SRD)
WP3 - Model to/from Knowledge Base

11 M2M Transformation Framework must synchronize the output models after its input models or configurations have been modified.
Software Requirements Document (SRD)
WP3 - Model to/from Knowledge Base

10 M2M Transformation Framework must keep traces between transformed models and its source models.

17 The system must support an open requirement authoring tools (such as RMF)
UC-TR-03 Generation and management of feature models
User Requirements Document (URD)

16 The system should support ReqIF standard.
UC-TR-03 Generation and management of feature models
User Requirements Document (URD)

15 BPMN 2.X shall be supported as the Business Process Modeling Language in the tool.
UC-TR-05 Synchronous Business Process Design with Use Cases
User Requirements Document (URD)

In Progress

Let others know you're working on an issue by dragging it to In Progress.

19 Fe

18 readme folder added

A screenshot of a Waffle board interface titled "modelwriter/requirements". The board is divided into four columns: "User Stories" (13 items), "Confirmed" (3 items), "In Progress" (0 items), and "Done" (2 items). Each item card contains a number, a brief description, and links to "Software Requirements Document (SRD)" and "WP3 - Model to/from Knowledge Base". The "Confirmed" column includes cards for requirement authoring tools (RMF) and ReqIF standard support. The "Done" column includes cards for "Fe" and "readme folder added". A sidebar on the left shows navigation icons for filters, search, and user profile. A green circular badge in the bottom right corner indicates 11 unread notifications.

ModelWriter Requirements & User Stories



<https://waffle.io/modelwriter/wp5>

A screenshot of a Waffle board titled "modelwriter/wp5". The board is organized into five columns: Queue (1 issue), Backlog (3 issues), ToDo (5 issues), In Progress (3 issues), and Done (13 issues). Each issue card contains a unique ID, a brief description, and a detailed breakdown of tasks and sprint assignments.

Queue	Backlog	ToDo	In Progress	Done
6 Share ModelWriter Documents - Share Google Drive Link	44 Develop a web based application which produces burndown chart of waffle.io <small>Sprint #2 5d Development</small> T5.0.1 - European and National Coordination and Reporting	23 Prepare a GitHub and SourceTree Video <small>Sprint #2 4h</small> T5.0.1 - European and National Coordination and Reporting	26 UYMK 2014 Presentation Preparation for ModelWriter <small>Sprint #1 4h</small> T5.0.1 - European and National Coordination and Reporting T7.3 - Workshops & Events	45 Create a Type and Access column for each deliverable of every Work Package <small>Sprint #2 2h</small> T5.0.1 - European and National Coordination and Reporting
37 Develop an application to archive issues of modelwriter repositories <small>Sprint #3 2d Development</small> T5.0.1 - European and National Coordination and Reporting	7 ITEA & ARTEMIS Co-summit 2015 - ModelWriter Exhibition requirements Call for participation to <small>Sprint #2 8h</small> T5.0.1 - European and National Coordination and Reporting T7.3 - Workshops & Events	21 Create an initial version for PCA and identify items to be discussed <small>Sprint #2 2d enhancement</small> T5.0.1 - European and National Coordination and Reporting	31 1st National Workshop in Ankara, Turkey - Meeting Agenda <small>Sprint #2 2h</small> T7.3 - Workshops & Events	40 International Workshop Program <small>Sprint #1 2h</small>
27 Send an email to Dr. Michael jastram to ask his advice about our project management approach <small>Sprint #3 1h</small> T5.0.1 - European and National Coordination and Reporting	19 Planning Product-Backlog/Releases/Sprint (Milestones/Deliverables) <small>Sprint #2 8h</small> T5.0.1 - European and National Coordination	32 1st International Workshop in Izmir, Turkey - Meeting Agenda <small>Sprint #3 4h</small> T7.3 - Workshops & Events	18 Create a User Manual for Github-SourceTree <small>Sprint #1 2h</small> T5.0.1 - European and National Coordination and Reporting	17 Create a User Manual for Waffle-GitHub Integration <small>Sprint #1 2h</small> T5.0.1 - European and National Coordination and Reporting
				20



What is a text?

What is a text? (document file formats)

Office Open XML (.docx) (ISO/IEC 29500)



The screenshot shows a Microsoft Word document titled "Library Management System.docx". The ribbon menu is visible at the top, showing tabs like FILE, HOME, INSERT, DESIGN, etc. The main content area contains the following text:

Library Management System

GLOSSARY

1.1 BOOK
[Book is a kind of collection item. It has an author or editor and \(...\)](#)

1.2 ...

REQUIREMENTS

1.1 REQUIREMENT - RESPONSE TIME FOR BOOK SEARCHES
The [system](#) shall perform all book search operations in less than 3 seconds.

1.2 REQUIREMENT - VALIDATION OF THE BOOK
The system allows the [user](#) to add new [book](#) data through a special [book form](#). The system [validates](#) [book](#) before storing it.

1.3 ...

At the bottom left, there is a small watermark-like logo with a large 'Σ' and an exclamation mark inside a circle. The status bar at the bottom shows "PAGE 1 OF 1", "76 WORDS", "ENGLISH (UNITED STATES)", and a zoom level of "%80".

What is a text? (document file formats)

Office Open XML (.docx) (ISO/IEC 29500)



Java - PropertyPage/test/document.xml - Eclipse

File Edit Source Navigate Search Project Sample Run Window Help

Sample Plain Text File document.xml

```
19    xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
20    <w:body>
21        <w:p w:rsidR="001D662C" w:rsidRDefault="004D2229" >
22            <w:pPr>
23                <w:pStyle w:val="Title" />
24            </w:pPr>
25            <w:bookmarkStart w:id="0" w:name="_GoBack" />
26            <w:bookmarkEnd w:id="0" />
27            <w:r>
28                <w:t xml:space="preserve">Library Management System </w:t>
29            </w:r>
30        </w:p>
31        <w:p w:rsidR="004D2229" w:rsidRDefault="004D2229" w:rsidP="004D2229">
32            <w:pPr>
33                <w:pStyle w:val="Heading1" />
34            <w:numPr>
35                <w:ilvl w:val="0" />
36                <w:numId w:val="0-1" />
37            </w:numPr>
```

Design Source

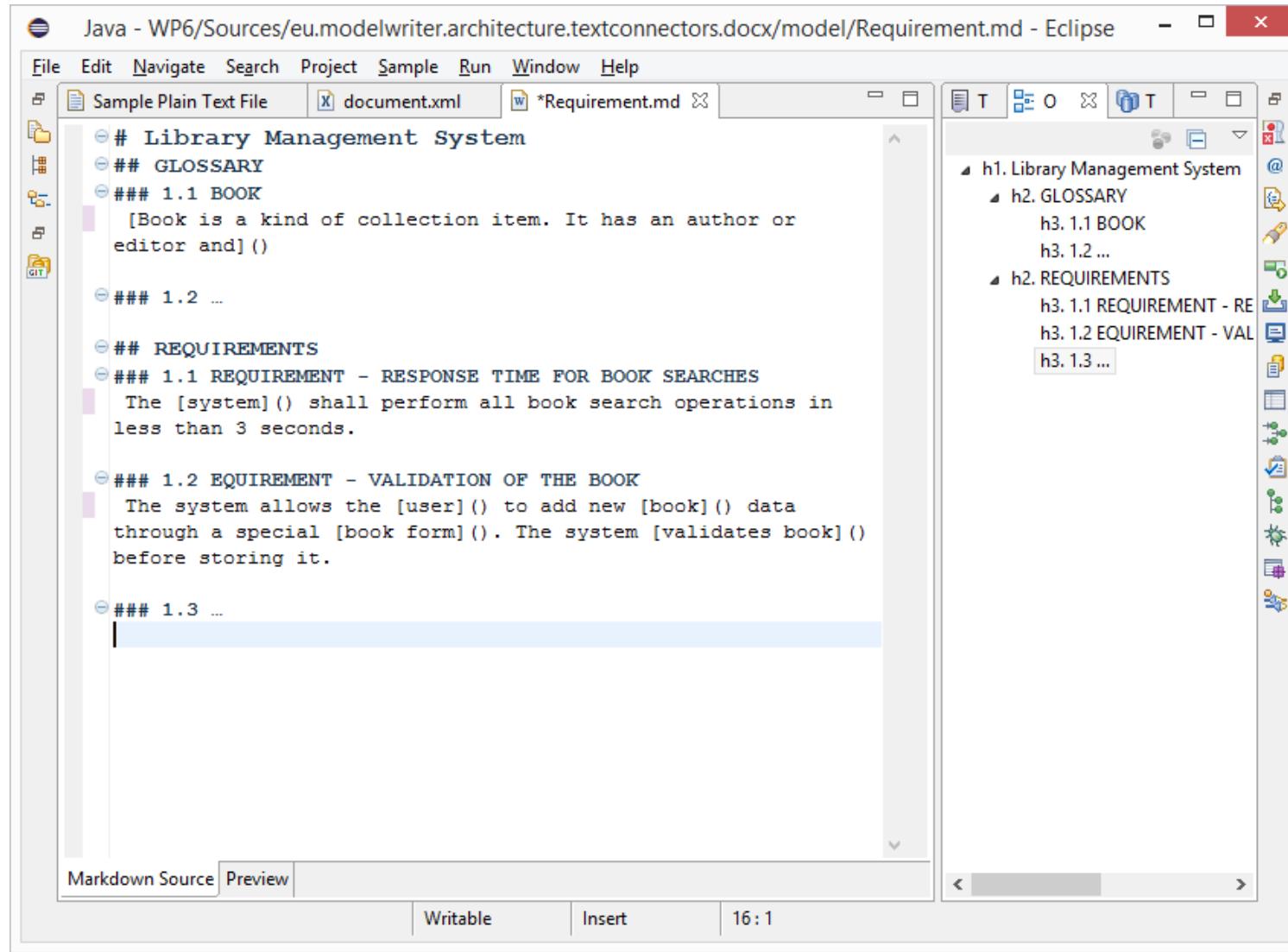
P... @ J... D... S... P... G... C... H... P... E... C... T... E... D... E... P...

Property	Value
w:val	Heading1

w:document/w:body/w:...:pPr/w:pStyle/w:val Writable Smart Insert 33 : 38

A screenshot of the Eclipse IDE interface. The main window displays the XML code for a Microsoft Word document (.docx). The code is color-coded to highlight different XML elements and attributes. The Eclipse toolbar is visible at the bottom, and the status bar at the bottom right shows the current file path, file type, and some status indicators. The interface is designed for editing and viewing XML files.

What is a text? (.md source file) text/markdown (ICANN Standard)



The screenshot shows the Eclipse IDE interface with a Markdown file open. The title bar reads "Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Requirement.md - Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Sample, Run, Window, and Help. The toolbar has icons for Sample Plain Text File, document.xml, and *Requirement.md. The left pane shows a tree view of the document structure:

- # Library Management System
 - ## GLOSSARY
 - ### 1.1 BOOK
 - [Book is a kind of collection item. It has an author or editor and] ()
 - ### 1.2 ...
- ## REQUIREMENTS
 - ### 1.1 REQUIREMENT - RESPONSE TIME FOR BOOK SEARCHES
 - The [system] () shall perform all book search operations in less than 3 seconds.
 - ### 1.2 REQUIREMENT - VALIDATION OF THE BOOK
 - The system allows the [user] () to add new [book] () data through a special [book form] (). The system [validates book] () before storing it.
 - ### 1.3 ...

The right pane shows a detailed tree view of the same structure, with icons for each node. At the bottom, there are tabs for "Markdown Source" and "Preview", and status bars for "Writable", "Insert", and "16:1".

What is a text? (HTML Preview) text/markdown (ICANN Standard)



The screenshot shows the Eclipse IDE interface with a Markdown editor open. The title bar indicates the file is "Requirement.md". The left pane displays the content of the Markdown file:

```
Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Requirement.md - Eclipse
File Edit Navigate Search Project Sample Run Window Help
*ReqModel pa... ReqModel.emf Requirement.md >4
```

Library Management System

GLOSSARY

1.1 BOOK

Book is a kind of collection item. It has an author or editor and

1.2 ...

REQUIREMENTS

1.1 REQUIREMENT - RESPONSE TIME FOR BOOK SEARCHES

The system shall perform all book search operations in less than 3 seconds.

1.2 REQUIREMENT - VALIDATION OF THE BOOK

The system allows the user to add new book data through a special book form.
The system validates book before storing it.

1.3 ...

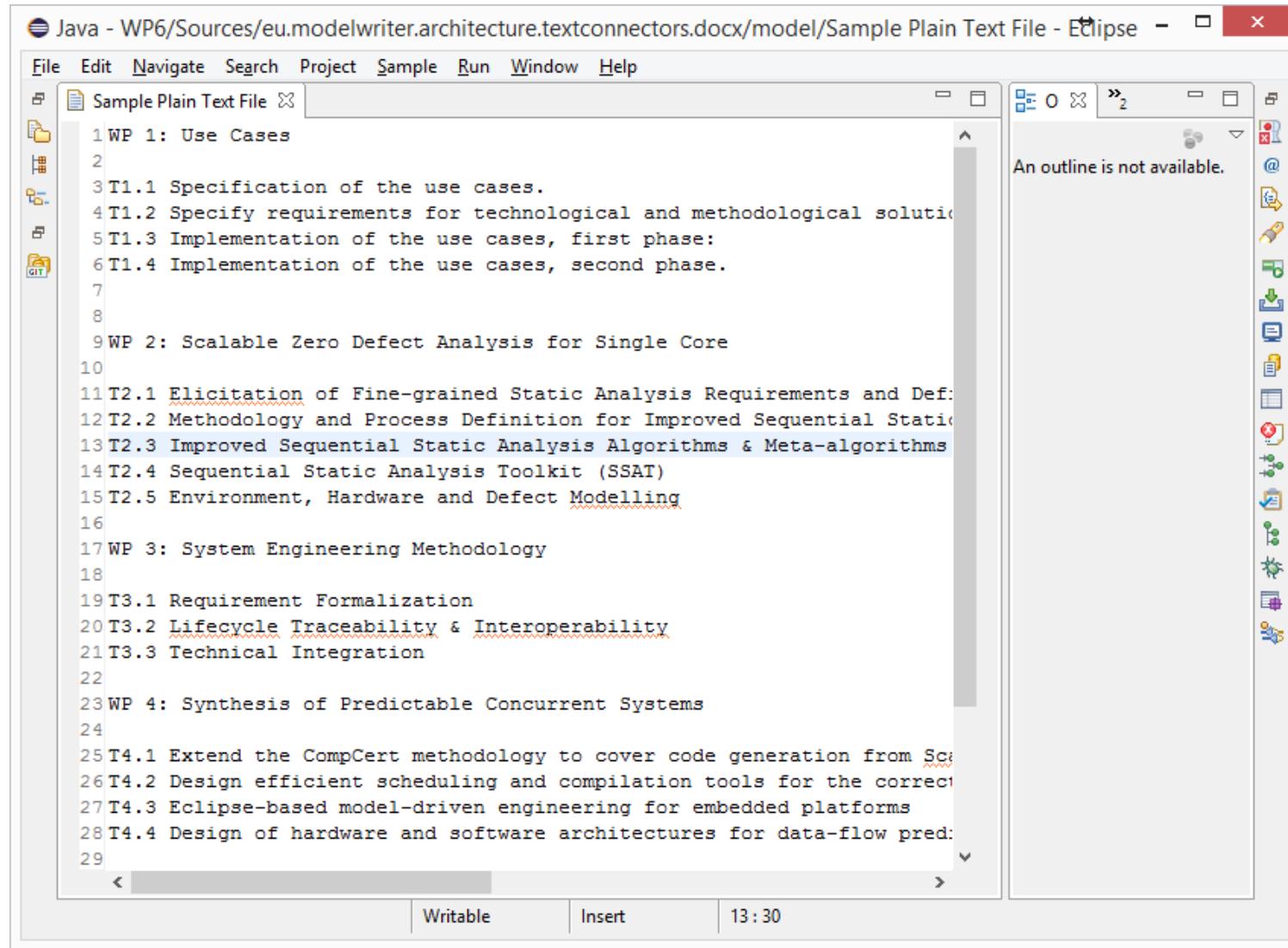
Markdown Source Preview

Writable Insert 16 : 1

The right pane shows a tree view of the document's structure:

- h1. Library Management System
 - h2. GLOSSARY
 - h3. 1.1 BOOK
 - h3. 1.2 ...
 - h2. REQUIREMENTS
 - h3. 1.1 REQUIREMENT - RESPON
 - h3. 1.2 EQUIREMENT - VALIDATI
 - h3. 1.3 ...

What is a text? (unformatted text) text/plain (ICANN Standard)



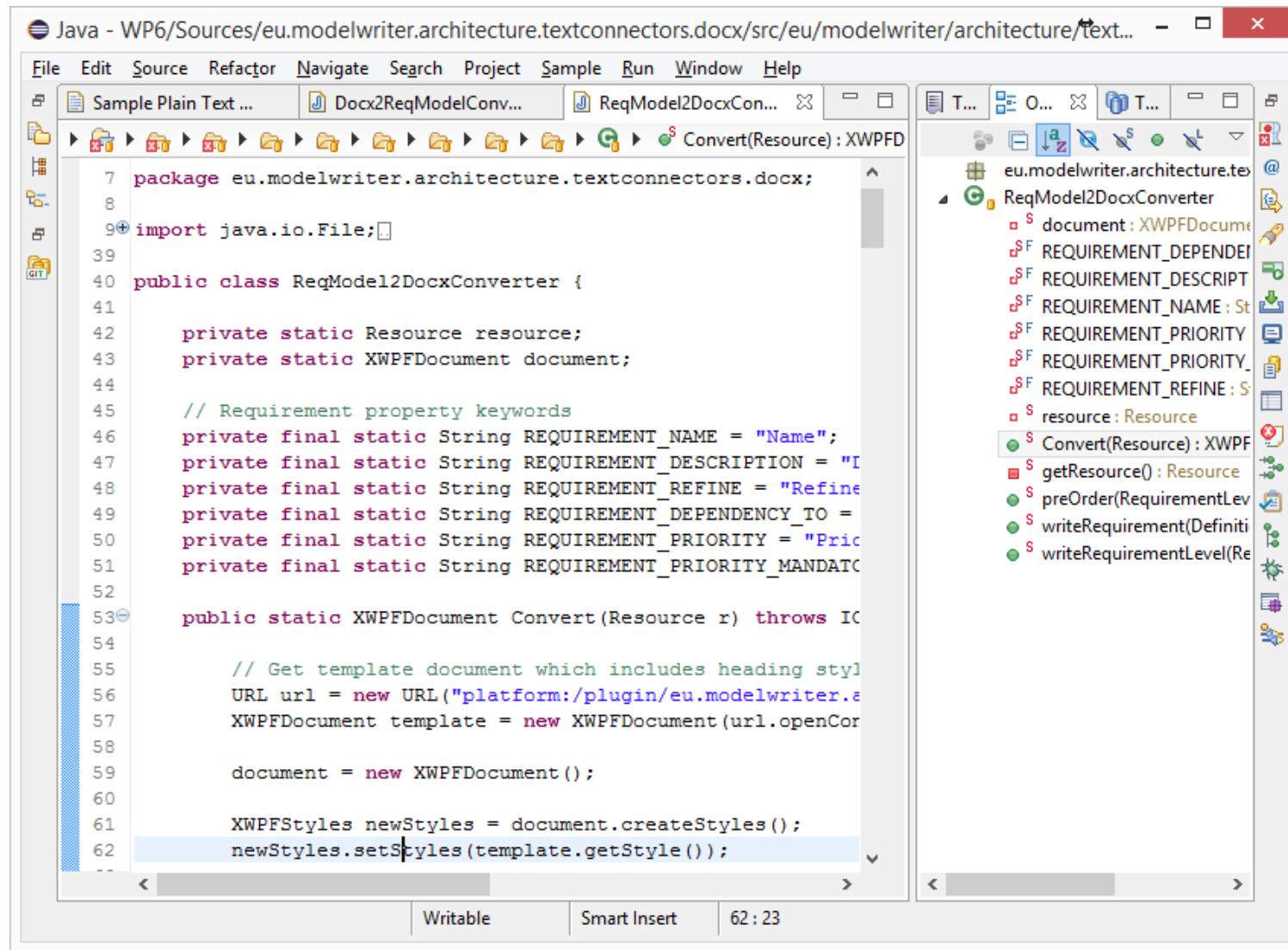
The screenshot shows the Eclipse IDE interface with a text editor open. The title bar reads "Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Sample Plain Text File - Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Sample, Run, Window, and Help. The left sidebar shows a file tree with "Sample Plain Text File" selected. The main editor area contains the following text:

```
1 WP 1: Use Cases
2
3 T1.1 Specification of the use cases.
4 T1.2 Specify requirements for technological and methodological solution
5 T1.3 Implementation of the use cases, first phase:
6 T1.4 Implementation of the use cases, second phase.
7
8
9 WP 2: Scalable Zero Defect Analysis for Single Core
10
11 T2.1 Elicitation of Fine-grained Static Analysis Requirements and Defe
12 T2.2 Methodology and Process Definition for Improved Sequential Static
13 T2.3 Improved Sequential Static Analysis Algorithms & Meta-algorithms
14 T2.4 Sequential Static Analysis Toolkit (SSAT)
15 T2.5 Environment, Hardware and Defect Modelling
16
17 WP 3: System Engineering Methodology
18
19 T3.1 Requirement Formalization
20 T3.2 Lifecycle Traceability & Interoperability
21 T3.3 Technical Integration
22
23 WP 4: Synthesis of Predictable Concurrent Systems
24
25 T4.1 Extend the CompCert methodology to cover code generation from Sc
26 T4.2 Design efficient scheduling and compilation tools for the correct
27 T4.3 Eclipse-based model-driven engineering for embedded platforms
28 T4.4 Design of hardware and software architectures for data-flow pred:
29
```

The status bar at the bottom indicates "Writable" and the time "13:30". To the right of the editor, there is an "Outline" view which displays the message "An outline is not available." Below the editor are several toolbars with various icons.

What is a text? (code files)

Java, C++ ... Programming Languages



The screenshot shows a Java IDE interface with two main panes. The left pane displays the source code for a Java class named `ReqModel2DocxConverter`. The right pane shows the class hierarchy and methods for this class.

```
Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/src/eu/modelwriter/architecture/text... - □ X
```

File Edit Source Refactor Navigate Search Project Sample Run Window Help

Sample Plain Text ... Docx2ReqModelConv... ReqModel2DocxCon... Convert(Resource) : XWPFD

```
7 package eu.modelwriter.architecture.textconnectors.docx;
8
9+ import java.io.File;
10
11 public class ReqModel2DocxConverter {
12
13     private static Resource resource;
14     private static XWPFDocument document;
15
16     // Requirement property keywords
17     private final static String REQUIREMENT_NAME = "Name";
18     private final static String REQUIREMENT_DESCRIPTION = "I";
19     private final static String REQUIREMENT_REFINE = "Refine";
20     private final static String REQUIREMENT_DEPENDENCY_TO =
21     private final static String REQUIREMENT_PRIORITY = "Priorit";
22     private final static String REQUIREMENT_PRIORITY_MANDATORY =
23
24
25     public static XWPFDocument Convert(Resource r) throws IOException {
26
27         // Get template document which includes heading styles
28         URL url = new URL("platform:/plugin/eu.modelwriter.a";
29         XWPFDocument template = new XWPFDocument(url.openConnection());
30
31         document = new XWPFDocument();
32
33         XWPFFormats newStyles = document.createStyles();
34         newStyles.setStyles(template.getStyle());
35 }
```

Writable Smart Insert 62 : 23

T... O... T... T... L... R... @... E... G... ReqModel2DocxConverter

- eu.modelwriter.architecture.textconnectors.docx
- ReqModel2DocxConverter
 - document : XWPFDocument
 - REQUIREMENT_DEPENDENCY_TO : String
 - REQUIREMENT_DESCRIPTION : String
 - REQUIREMENT_NAME : String
 - REQUIREMENT_PRIORITY : String
 - REQUIREMENT_PRIORITY_MANDATORY : String
 - REQUIREMENT_REFINE : String
 - resource : Resource
 - Convert(Resource) : XWPFDocument
 - getResource() : Resource
 - preOrder(RequirementLevel)
 - writeRequirement(Definition)
 - writeRequirementLevel(RequirementLevel)

What is a model?

Everything is a model! (ReqIF Standard)

Requirements Interchange Format



ProR - platform:/resource/LibraryManagementSystem/My.reqif - formalmind Studio

File Edit Search Requirements fmStudio Window Help

Quick Access

My.reqif Requirements Document

Outline

ID Name Description

ID	Name	Description
1	Librarian	Librarian
1.1	R123	Response Time for Book Searches
1.2	R123	The system shall perform all book search operations in less than 3 seconds.
1.3	UC071	Add new Book
1.4	R124	Validation of the Book

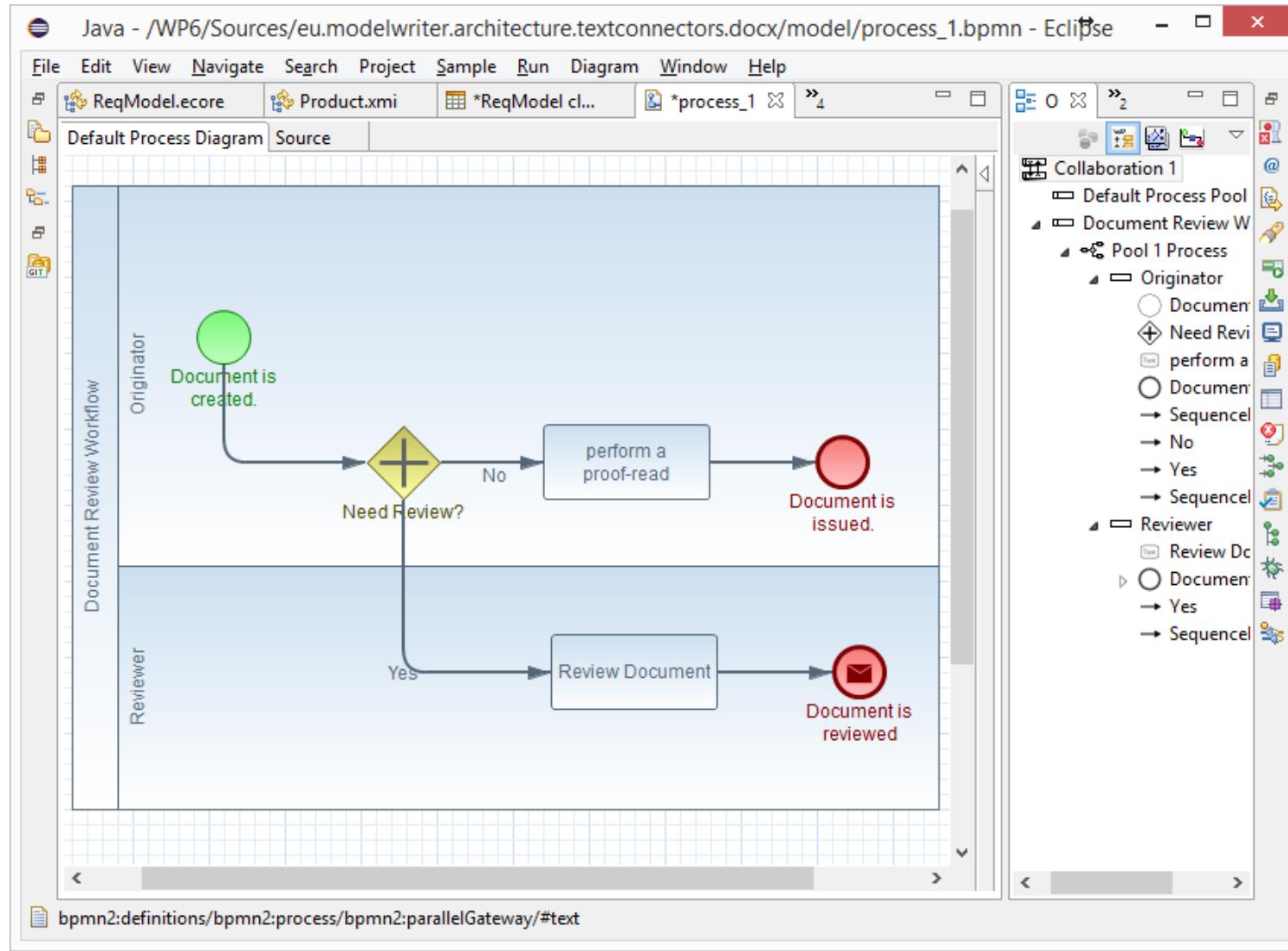
Properties

Property	Value
Requirement Type	
Description	The system shall perform all book search operations in less than 3 second
ID	R123
Name	Response Time for Book Searches
Responsible	Ferhat
Version	1
Spec Object	
Type	Requirement Type (Spec Object)

Standard Attributes All Attributes

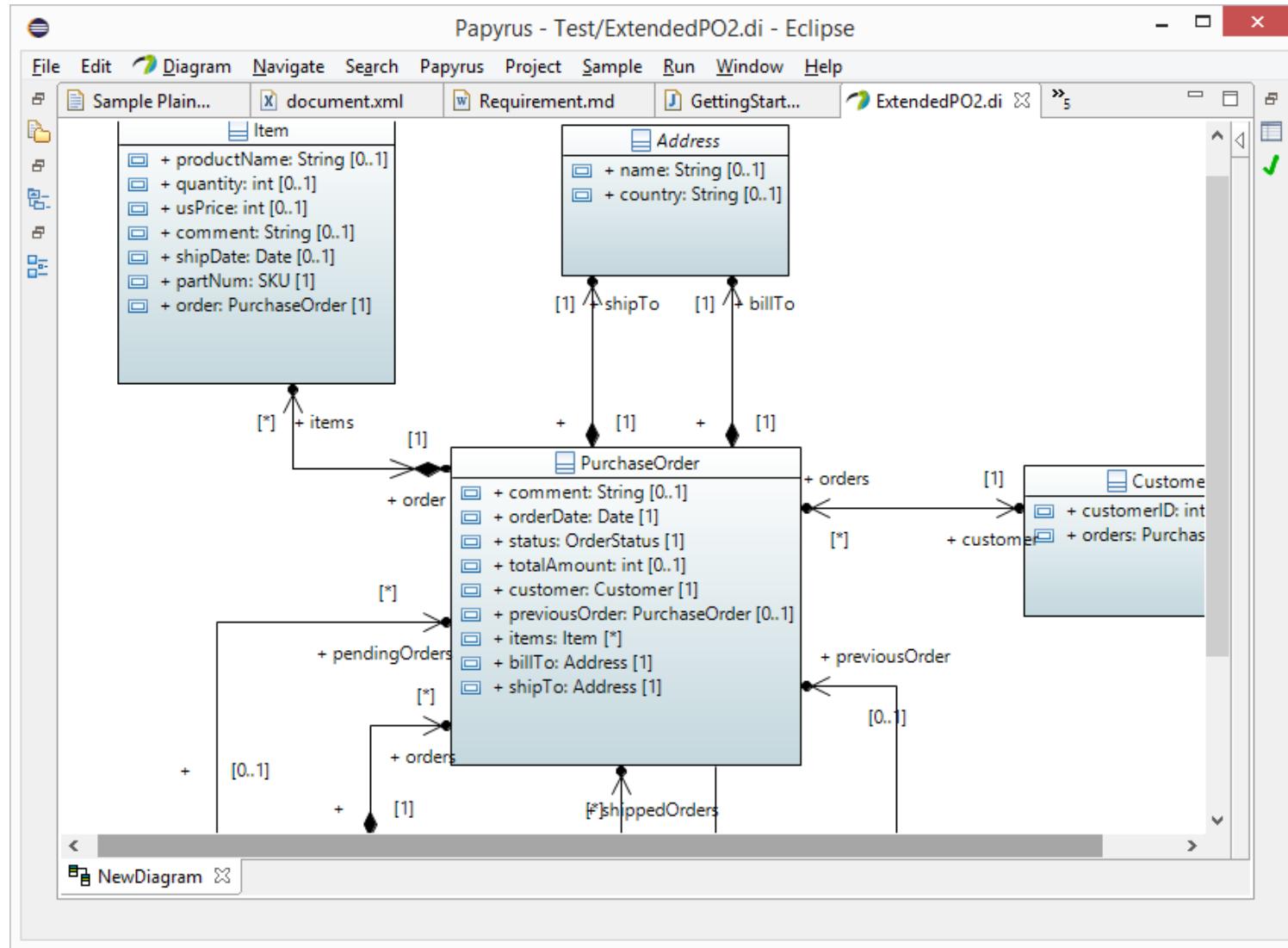
Everything is a model! (BPMN Standard)

Business Process Model & Notation



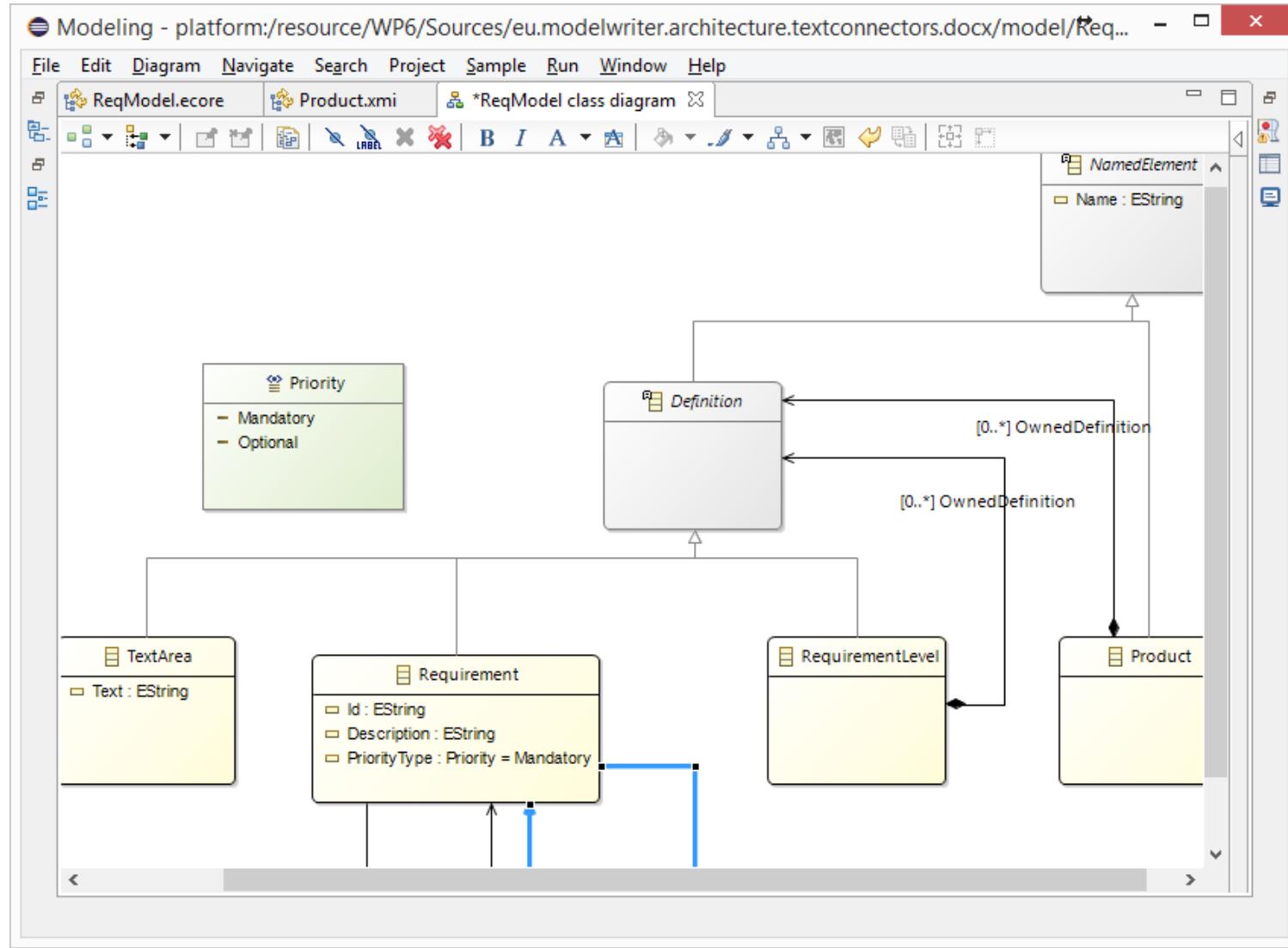
Everything is a model! (UML Standard)

UML Modeling Languages



Everything is a model!

Eclipse Modeling Framework (EMF)



Everything is a model!

Tree-based or Tabular Representations



The screenshot shows a modeling environment with the following components:

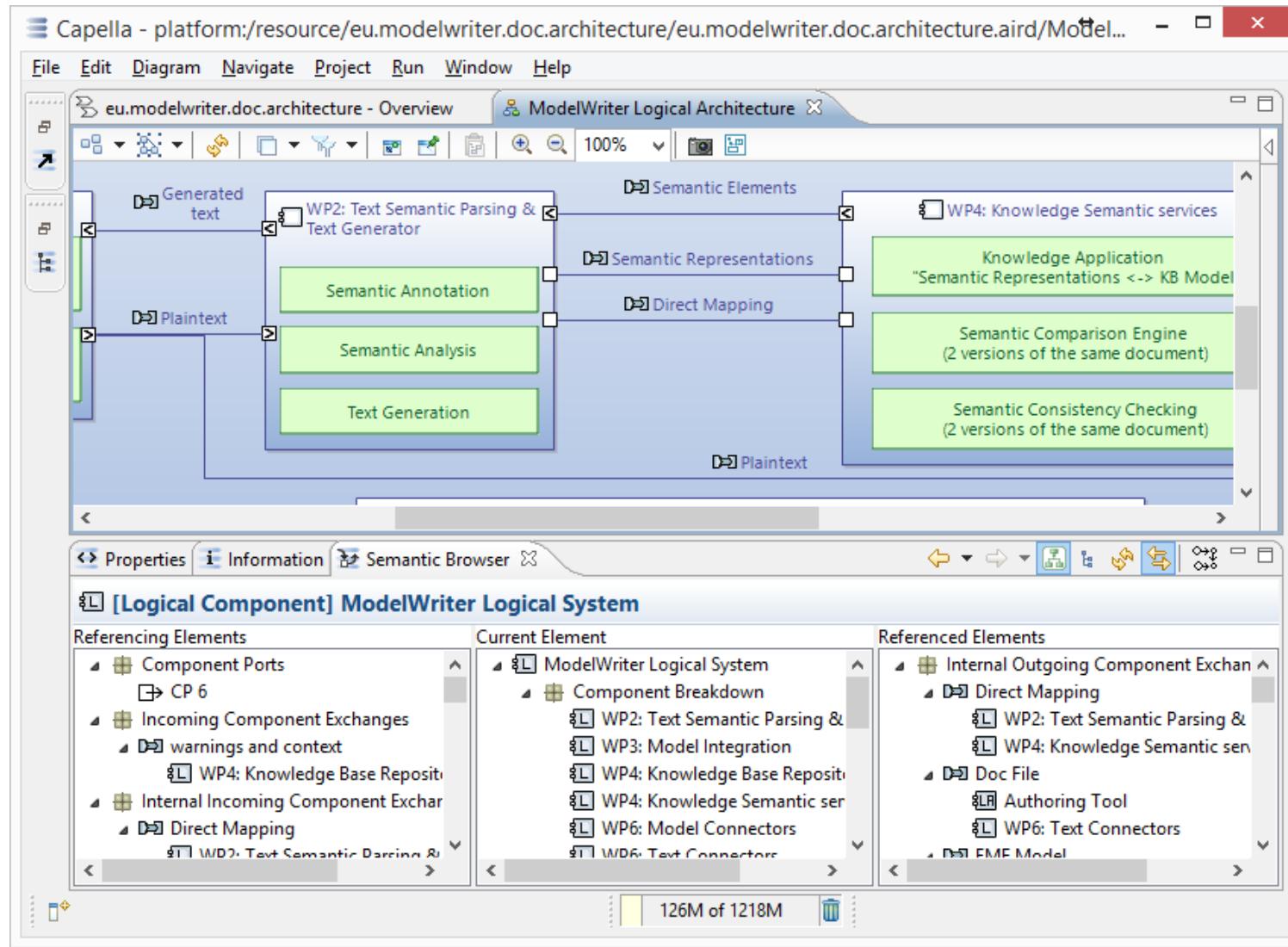
- Top Bar:** File, Edit, Navigate, Search, Project, DTable, Sample, Run, Window, Help.
- Left Sidebar:** Shows a tree view of model elements:
 - ReqModel.ecore
 - Product.xmi
 - *ReqModel class table
 - NamedElement
 - Name : EString
 - Product -> NamedElement
 - OwnedDefinition : Definition
 - Definition -> NamedElement
 - RequirementLevel -> Definition
 - OwnedDefinition : Definition
 - Requirement -> Definition
 - Id : EString
 - Description : EString
 - Refine : Requirement
 - DependencyTo : Requirement
 - PriorityType : Priority
 - TextArea -> Definition
 - Text : EString
- Central View:** A table titled "NamedElement" with columns "Name" and "Value".

Name	Value
Name	NamedElement
Product	
OwnedDefinition	
Definition	
RequirementLevel	
OwnedDefinition	
Requirement	
Id	
Description	
Refine	
DependencyTo	
PriorityType	
TextArea	
Text	
- Bottom View:** A table titled "NamedElement" with columns "Semantic" and "Property".

Semantic	Property	Value
NamedElement	Abstract	true
NamedElement	Default Value	
NamedElement	ESuper Types	
NamedElement	Instance Type Name	

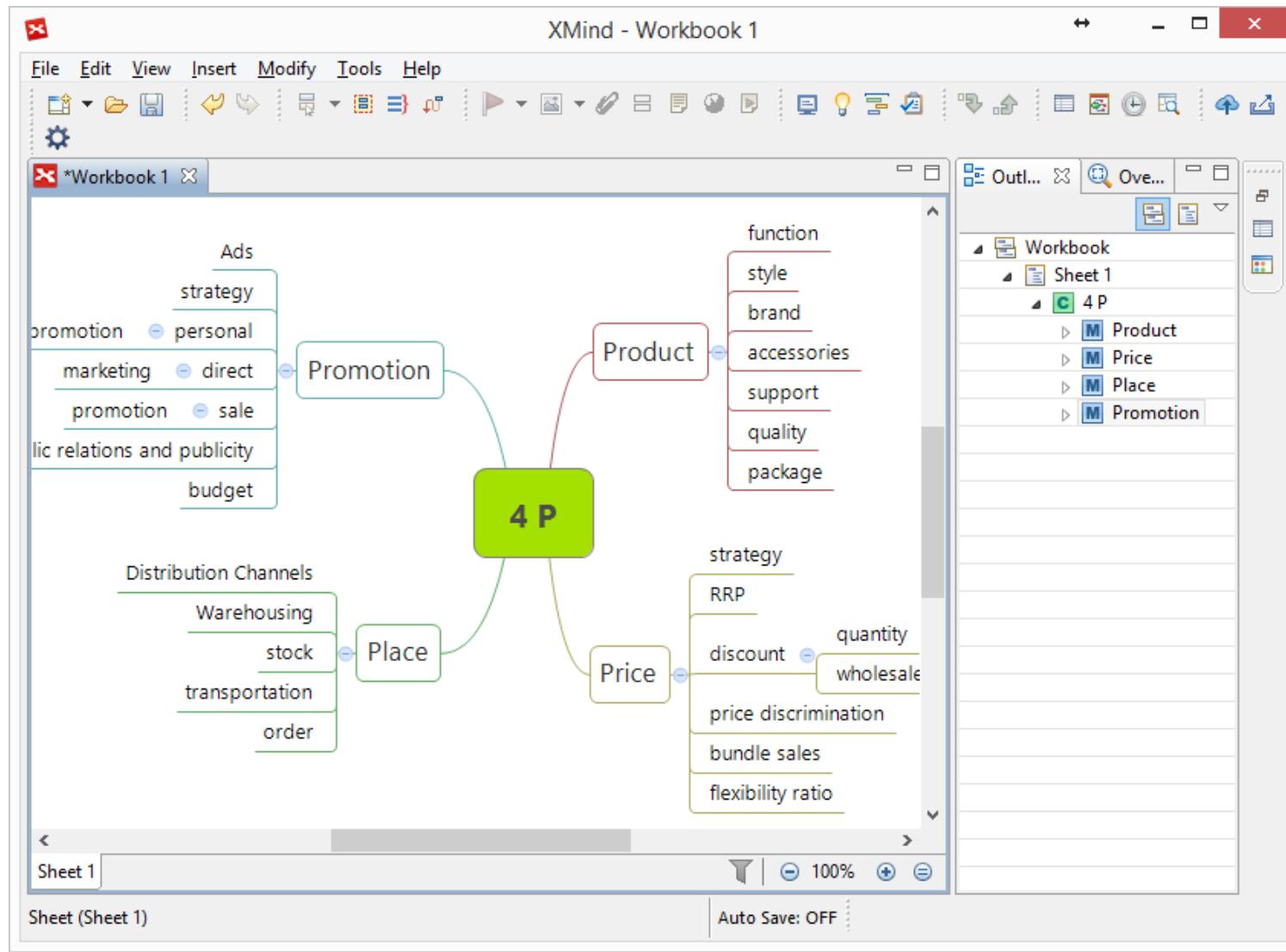
Everything is a model!

Software/System Architecture Design



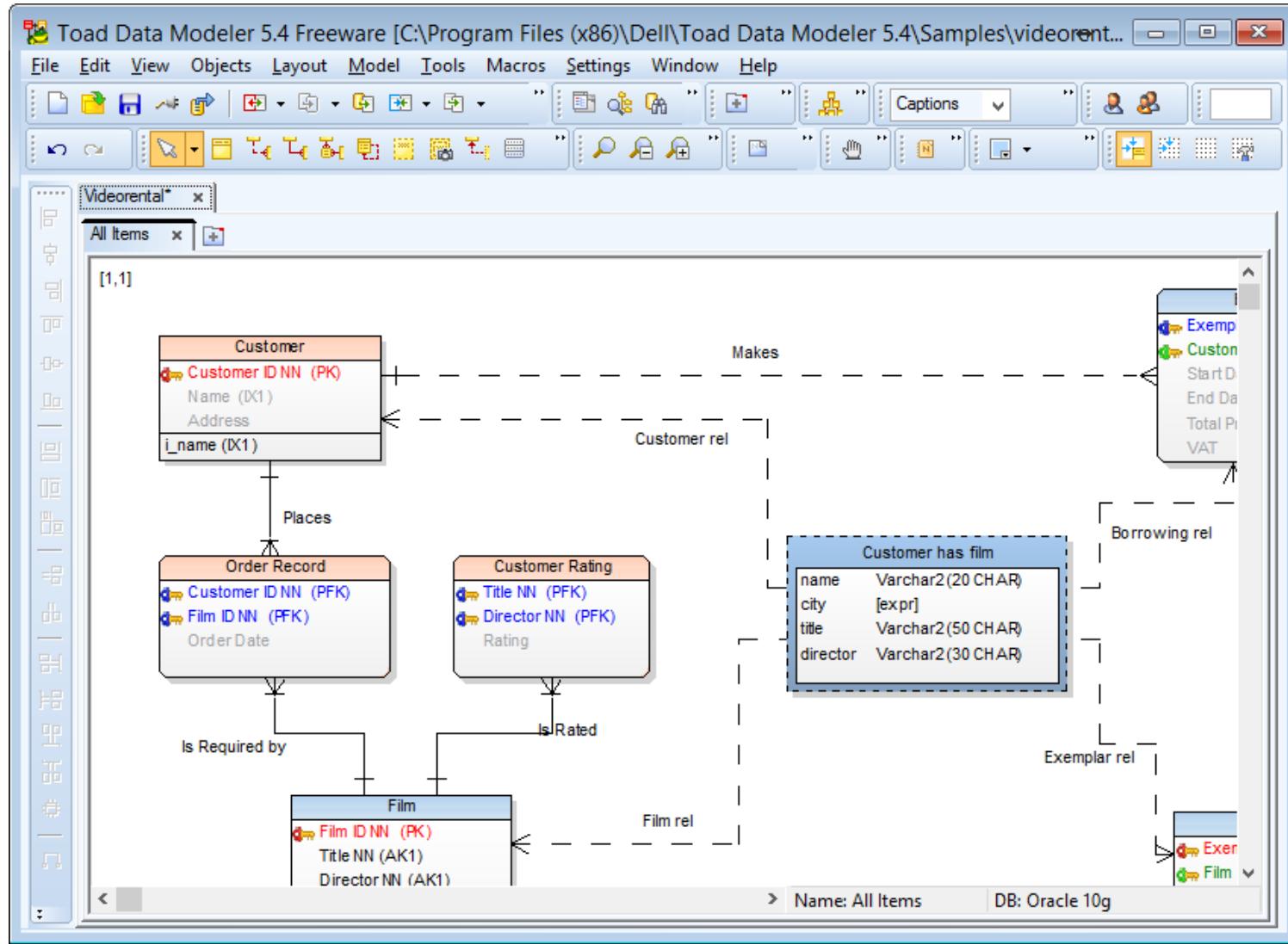
Everything is a model!

Topic Maps, Mind Maps, Vocabularies ...



Everything is a model!

Databases (ER, IDEF1.x)



Everything is a model! (Textual Lang.)

Domain Specific Languages



Modeling - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/ReqModel.emf - Eclipse

File Edit Navigate Search Project Sample Run Window Help

ReqModel.ecore Product.xmi *ReqModel cl... *ReqModel do... ReqModel.emf »

```
1 @namespace(uri="eu.modelwriter.architecture.textconnectors.docx.reqmodel", prefix="ReqMod")
2 package ReqModel;
3
4 @gmf.node(label="name")
5 abstract class NamedElement {
6     attr String Name;
7 }
8
9 @gmf.diagram
10 @gmf.node(label="Name")
11 class Product extends NamedElement {
12
13     @gmf.compartment(collapsible="true")
14     val Definition[*] OwnedDefinition;
15 }
16
17 abstract class Definition extends NamedElement {
18 }
19
20 @gmf.node(figure="rectangle", label.icon="true", label="Name", label.pattern="{0}")
21 class RequirementLevel extends Definition {
22
23     @gmf.compartment(collapsible="true", layout="list")
24     val Definition[*] OwnedDefinition;
25 }
26
27 @gmf.node(figure="rounded", label.icon="true", label="Name", label.pattern="{0}")
28 class Requirement extends Definition {
29     attr String Id = "";
30 }
```

Writable Insert 11:9

Everything is a model! (Java, C++, etc.)

Even Programming Languages (ASTs)



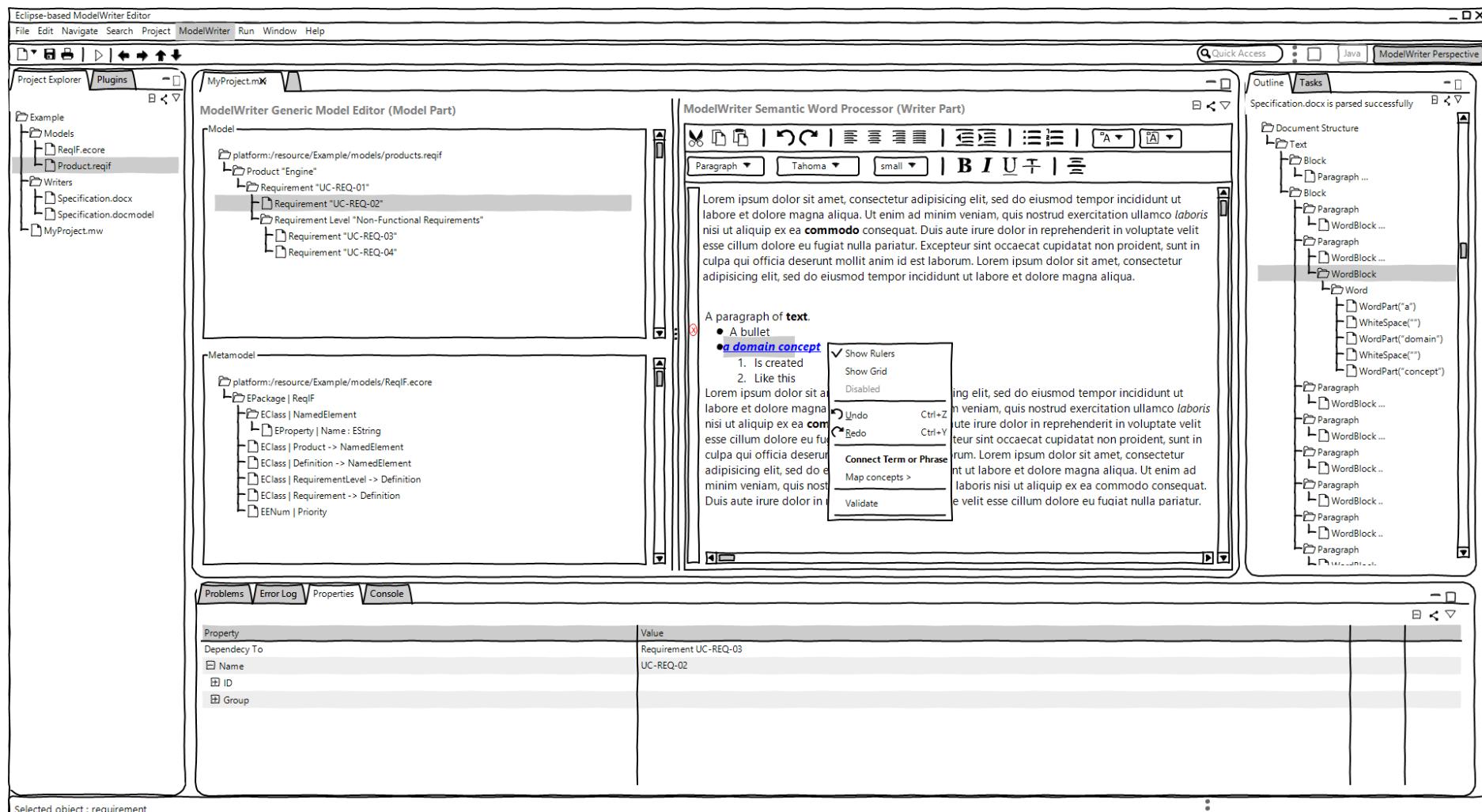
A screenshot of an IDE interface. On the left, a code editor displays Java code for a class named `ReqModel2DocxConverter`. The code includes imports for `java.io.File` and `XWPFD`, and defines static fields for `Resource` and `XWPFDocument`. It also contains static final strings for requirement properties like `NAME`, `DESCRIPTION`, `REFINE`, `DEPENDENCY_TO`, `PRIORITY`, and `MANDATORY`. The code implements a `Convert(Resource r)` method that creates a new `XWPFDocument`, sets its styles from a template, and then performs some operations on it. On the right, a tool palette provides various icons for file operations, requirements management, and document conversion. A tree view shows the class structure of `ReqModel2DocxConverter`, listing methods such as `Convert(Resource : XWPFDocument)`, `getResource() : Resource`, `preOrder(RequirementLevel)`, `writeRequirement(Definition)`, and `writeRequirementLevel(RequirementLevel)`.

```
7 package eu.modelwriter.architecture.textconnectors.docx;
8
9 import java.io.File;
10
11 public class ReqModel2DocxConverter {
12
13     private static Resource resource;
14     private static XWPFDocument document;
15
16     // Requirement property keywords
17     private final static String REQUIREMENT_NAME = "Name";
18     private final static String REQUIREMENT_DESCRIPTION = "I";
19     private final static String REQUIREMENT_REFINE = "Refine";
20     private final static String REQUIREMENT_DEPENDENCY_TO =
21     private final static String REQUIREMENT_PRIORITY = "Prior";
22     private final static String REQUIREMENT_PRIORITY_MANDATOR
23
24
25     public static XWPFDocument Convert(Resource r) throws I
26
27         // Get template document which includes heading styl
28         URL url = new URL("platform:/plugin/eu.modelwriter.a
29         XWPFDocument template = new XWPFDocument(url.openCor
30
31         document = new XWPFDocument();
32
33         XWPFStyles newStyles = document.createStyles();
34         newStyles.setStyles(template.getStyle());
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62 }
```

Is it possible to connect and keep arbitrary software/system engineering artifacts synchronized ?

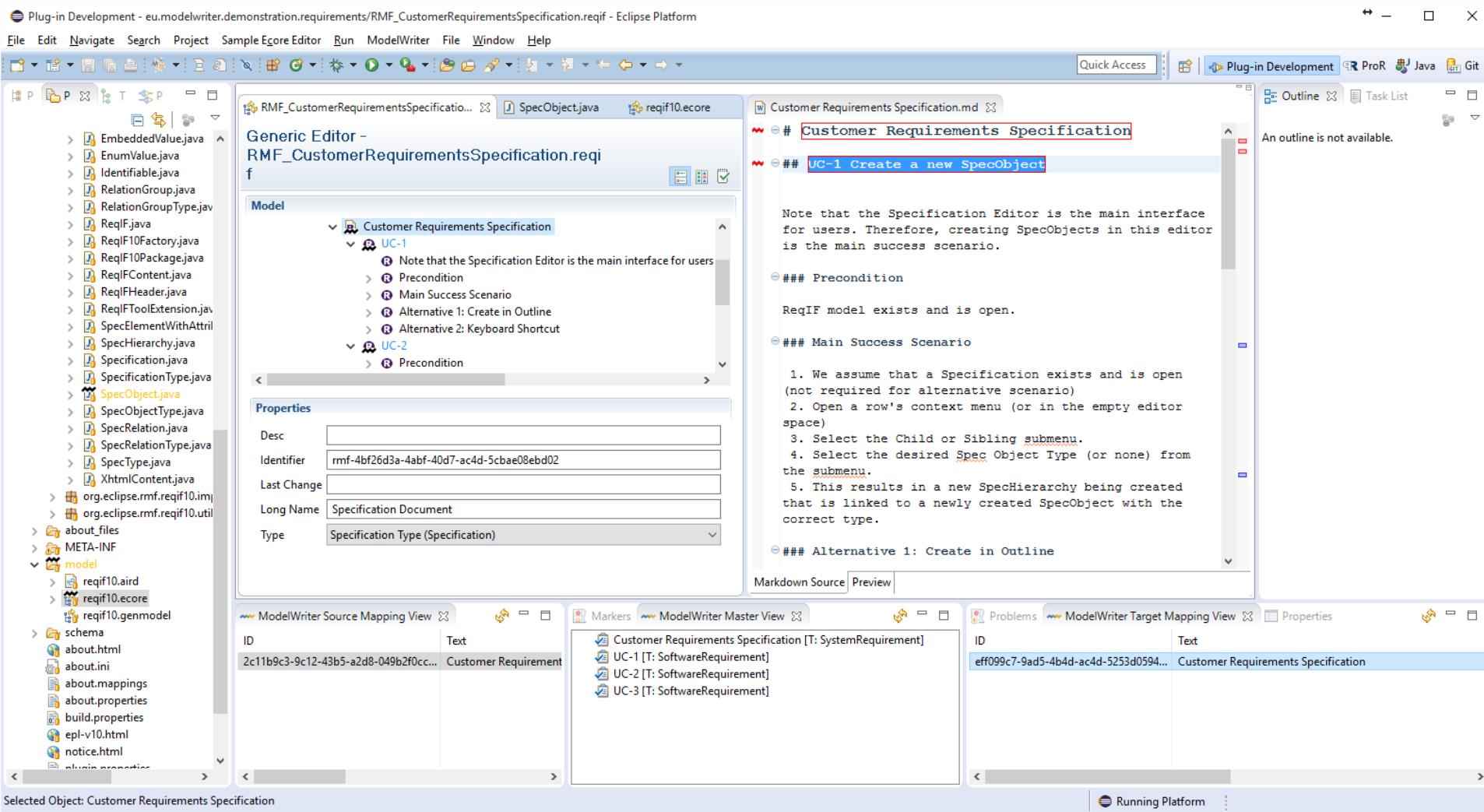


ModelWriter – The Solution



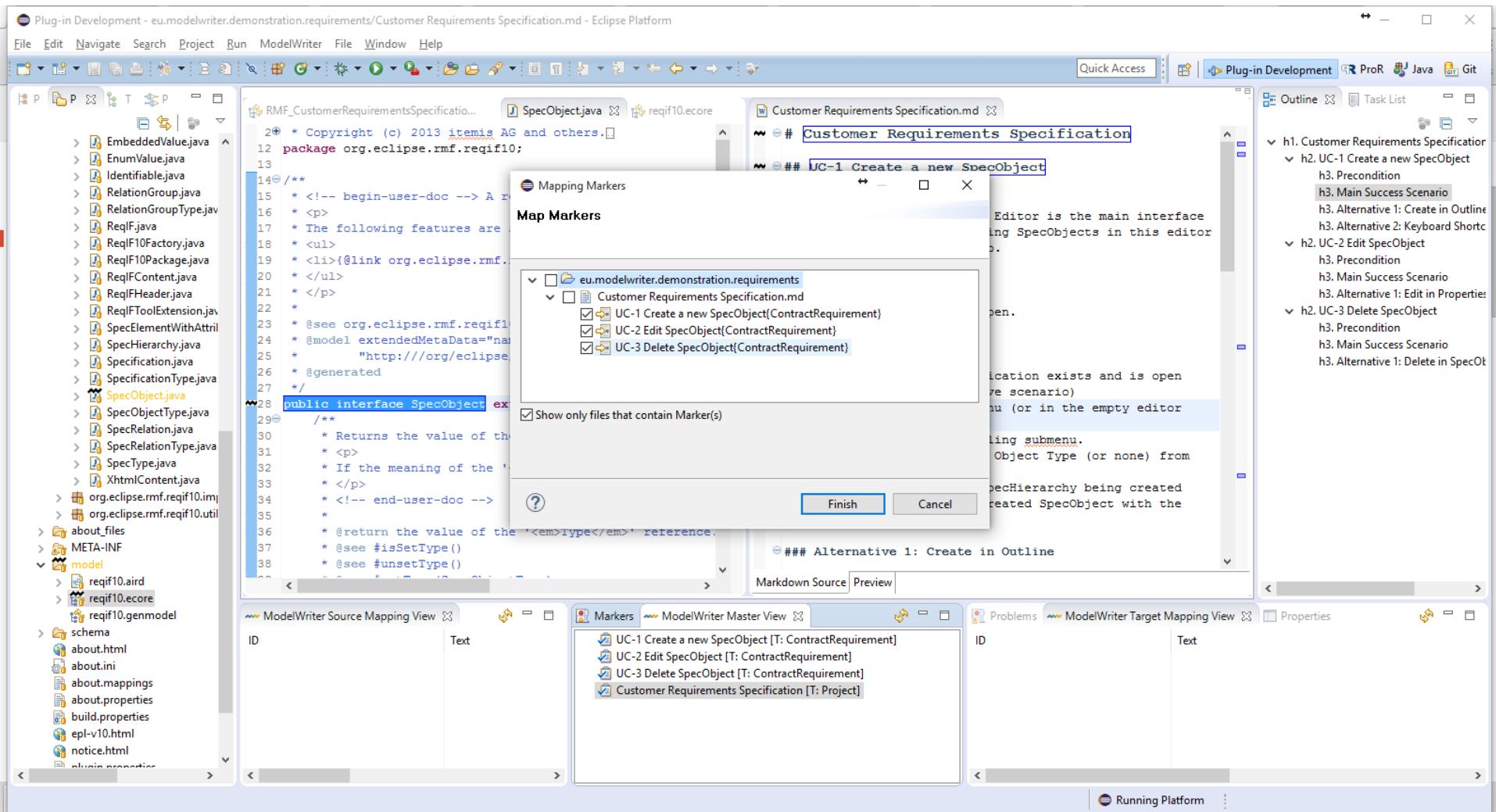
Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



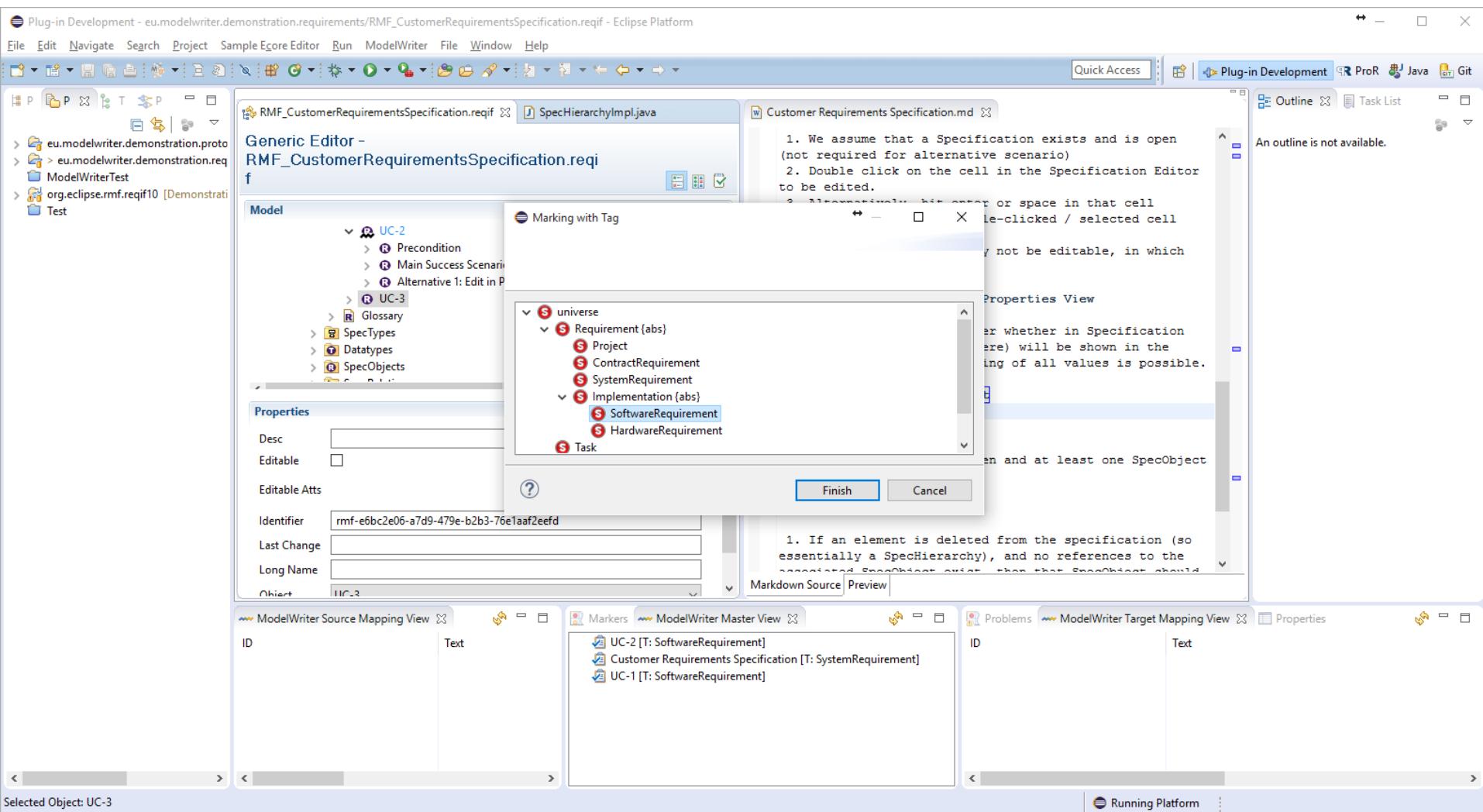
Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

Is it possible to extract
knowledge from texts
fragments based on a given
ontology (model) ?



Solution – Knowledge Extraction

ModelWriter Project

File Link Change Statistic

The

- Generate Links
- Search Link
- 2 P
- Add Link
- ABS: Remove Link

unction zones
shall be used

Flexible Hoses Shall be defined with a maximum length of 500 mm regardless of
ABS2195 -LRB- preferred for weight saving -RRB- or NSA5516J P-Clamp Shall be U
Rigid Pipes Shall be segregated to fixed Structure by not less than 10 mm as sh
Flexible Hoses Shall be segregated to rigid Component/Item/Object by not less t
Rigid Pipes Shall be segregated to movable Component/Item/Object by not less
Flexible Hoses Shall be segregated to movable Component/Item/Object by not le
Pipes Shall be fixed
Unions Shall be fixed on Pipes at alternating positions as shown in the attach
Unions Shall be positioned close to one fixation point .

The Model

Plain Tree

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:acs="http://airbus-group/aircraft-system#"
  xmlns:evt="http://airbus-group.installsys/event#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:spin="http://spinrdf.org/spin#"
  xmlns:qudt="http://qudt.org/schema/qudt#"
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:arg="http://spinrdf.org/arg#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:vaem="http://www.linkedmodel.org/schema/vaem#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:voag="http://voag.linkedmodel.org/voag#"
  xmlns:comp="http://airbus-group.installsys/component#"
  xmlns:qudt-dimension="http://qudt.org/vocab/dimension#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:iems="http://airbus-group/installationMeasure#"
  xmlns:dtype="http://www.linkedmodel.org/schema/dtype#"
  xmlns:mat="http://airbus-group/material#"
```

The links between text and model

T2M M2T Link

```
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/id270">
  <j:0:hasOffset>270</j:0:hasOffset>
  <j:0:isSameAs>http://www.linkedmodel.org/schema/vaem#id</j:0:isSameAs>
  <j:0:hasValue>id</j:0:hasValue>
</rdf:Description>
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/attach818">
  <j:0:hasOffset>818</j:0:hasOffset>
  <j:0:isSameAs>http://airbus-group/opp-function#Attach</j:0:isSameAs>
  <j:0:hasValue>attach</j:0:hasValue>
</rdf:Description>
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/attached709">
  <j:0:hasOffset>709</j:0:hasOffset>
  <j:0:isMorphologySimilarTo>http://airbus-group/opp-function#Attach</j:0:isMorphologySim
  <j:0:hasValue>attached</j:0:hasValue>
</rdf:Description>
</rdf:RDF>
```

Text & Model-Synchronized Document Engineering Platform



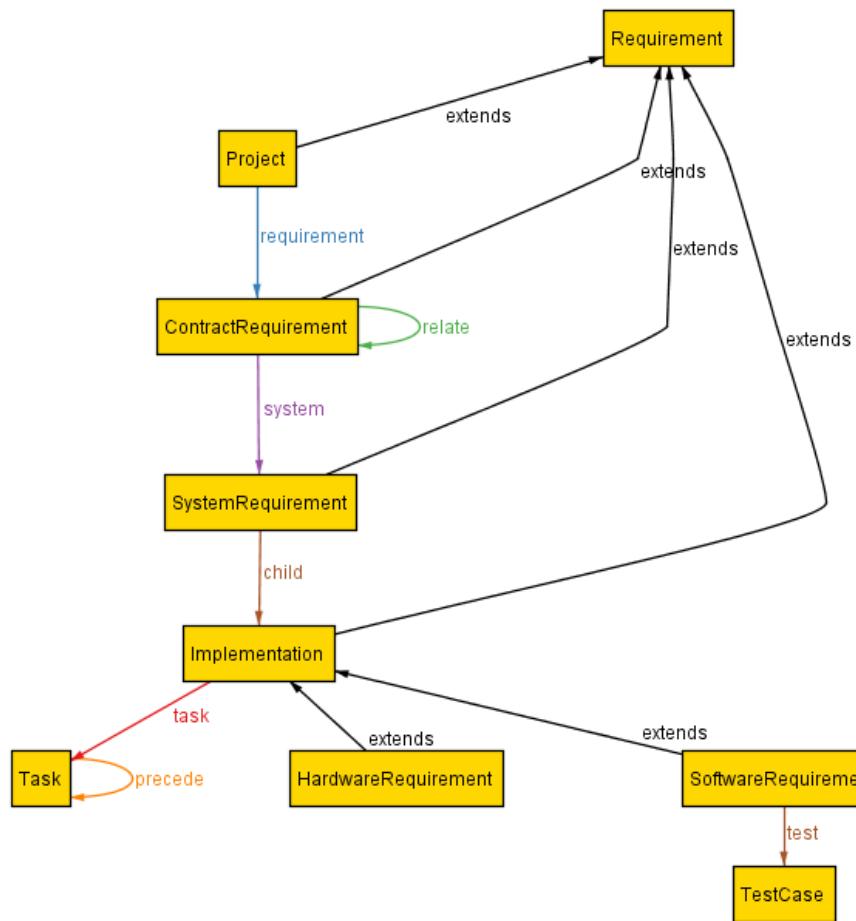
Synchronization is maintained!

What about configuration/formalization of the platform?

Configuration: Havelsan example

```

extends: 6
child: 1
precede: 1
relate: 1
requirement: 1
system: 1
task: 1
test: 1
  
```



```

module Havelsan/Requirement

abstract sig Requirement {}

sig Task {
    precede: lone Task,
    { all t: Task | one t.^precede }
}

one sig Project extends Requirement {
    requirement: some ContractRequirement
}

sig ContractRequirement extends Requirement {
    system: set SystemRequirement,
    relate: set ContractRequirement
}{all c: ContractRequirement | one c.^~requirement}

--@name: "System Requirement"
sig SystemRequirement extends Requirement {
    child: some Implementation
}{all s: SystemRequirement | one s.^~system}

abstract sig Implementation extends Requirement {
    task: set Task
}{all i: Implementation | one i.^~child}

--@context.editor: "ReqIFEditor"
sig SoftwareRequirement extends Implementation {
    test: some TestCase
}

sig HardwareRequirement extends Implementation {}

sig TestCase {}{ all t: TestCase | one t.^~test}

fact noSelfRelation{
    no c: ContractRequirement | c in c.relate
    no t: Task | t in t.^precede
}

fact noCycles{no t: Task | t in t.^precede}

fact realismConstraint {
    some ContractRequirement
    some HardwareRequirement
    some SoftwareRequirement
    some precede
}
  
```

A Formal Specification Model to configure the ModelWriter

Is it possible to vizualize the trace links?



Traceability: Havelsan example

The screenshot shows a software interface for traceability and specification management. On the left, a 'Traceability Virtualization' window displays a hierarchical requirement structure. At the top is a 'Project' node, which branches into three 'ContractRequirement' nodes: 'ContractRequirement2', 'ContractRequirement0', and 'ContractRequirement1'. 'ContractRequirement1' has a green arrow labeled 'system' pointing to a 'SystemRequirement' node. This 'SystemRequirement' node has three child nodes: 'SoftwareRequirement2', 'SoftwareRequirement0', and 'SoftwareRequirement1'. 'SoftwareRequirement1' has a blue arrow labeled 'task' pointing to a 'Task1' node, which in turn has a red arrow labeled 'precede' pointing to a 'Task0' node. A status bar at the bottom left indicates: 'child: 3', 'precede: 1', 'requirement: 3', 'system: 1', and 'task: 1'.

Customer Requirements Specification.md

- # Customer Requirements Specification
- ## UC-1 Create a new SpecObject

Note that the Specification Editor is the main interface for users. Therefore, creating SpecObjects in this editor is the main success scenario.

- ### Precondition
- ReqIF model exists and is open.
- ### Main Success Scenario
- 1. We assume that a Specification exists and is open (not required for alternative scenario)
- 2. Open a row's context menu (or in the empty editor space)
- 3. Select the Child or Sibling submenu.
- 4. Select the desired Spec Object Type (or none) from the submenu.
- 5. This results in a new SpecHierarchy being created that is linked to a newly created SpecObject with the correct type.

Alternative 1: Create in Outline

ModelWriter Source Mapping View

ID	Text
eff099c7-9ad5-4b4d-ac4d-5253d0594...	Customer Requirement

Markers ModelWriter Master View

ID	Text
	SpecObject [T: Task]

Problems ModelWriter Target Mapping View

ID	Text

Properties

A Formal Specification Model to configure the ModelWriter

**Thank you for your attention
We value your opinion and
questions.**

3 Progress on the Industrial Use Cases

Prof. Geylani Kardaş, Moderator
KoçSistem

UC-FR-01 - Synchronization between models and documents

Yvan lussaud
OBEO

Artifacts

- Graphical Mapping DSL
- Source code
- Documentation (specifier, user)

Life cycle

- Release train Eclipse
- Dependencies

Team

- Core team
- External contributors

Methodology

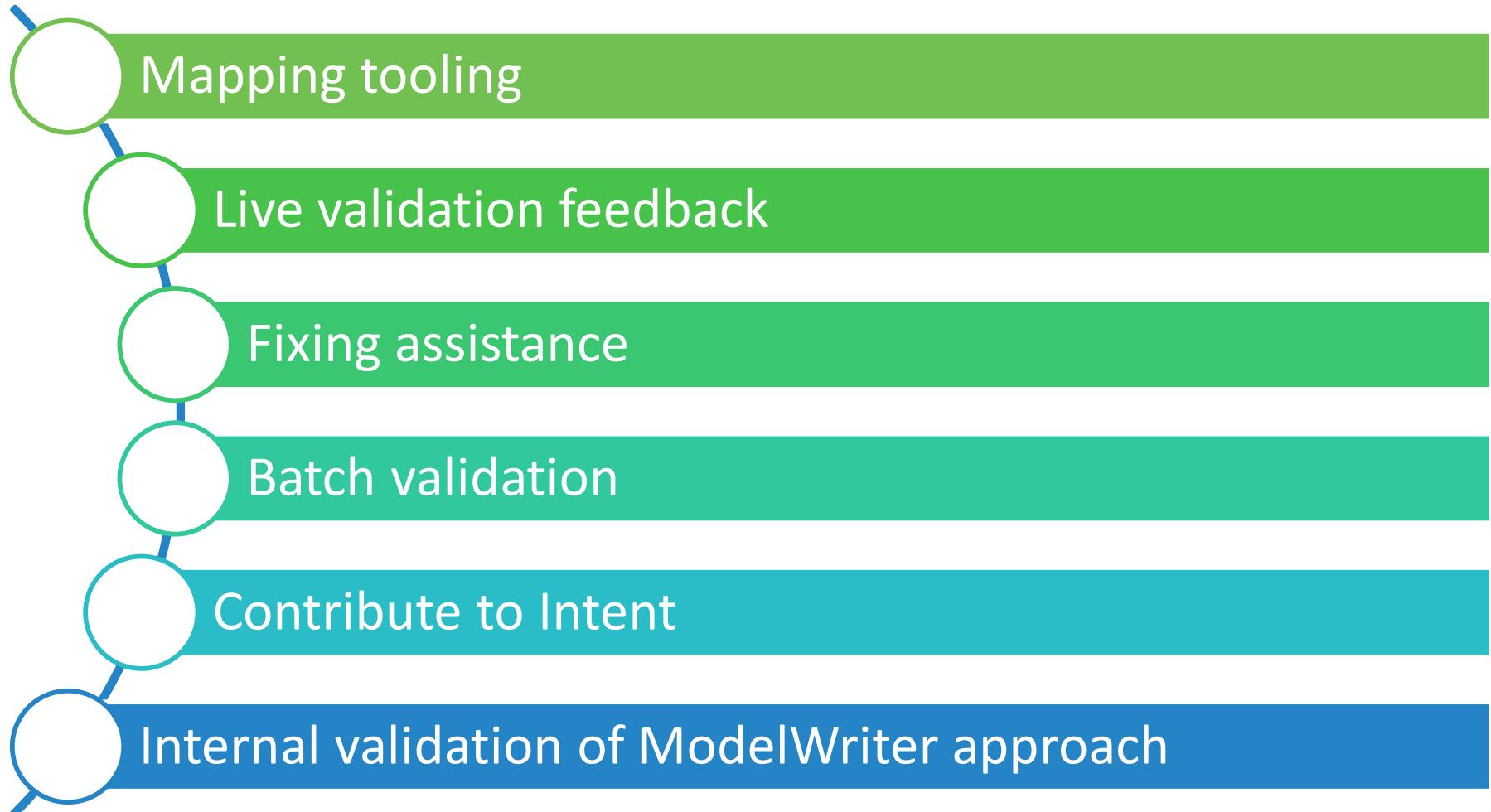
- Code review
- Plan manual synchronization of artifacts

Synchronization gap

- Development
- Artifacts knowledge is volatile

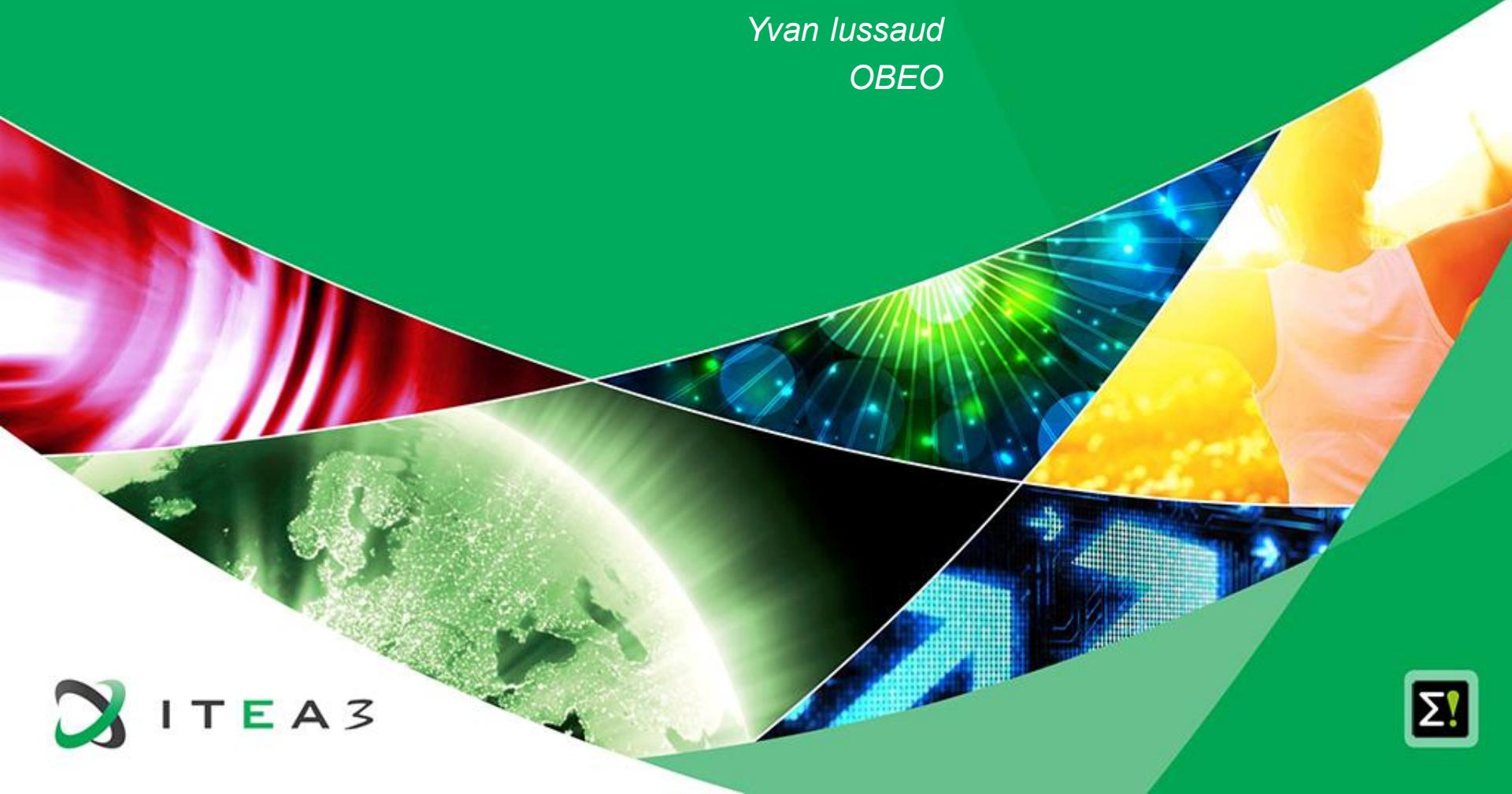
Validation

- Release
- Quality



UC-FR-02 - Enterprise Architecture

Yvan lussaud
OBEO



Define As-Is state

- Import documents
- Consolidate the company model

Define To-Be state

- Identify possible scenario
- Evaluate possible scenario
- Modify the company model

Define trajectories

- Gap analysis
- Impact analysis
- Define milestones

Documentation

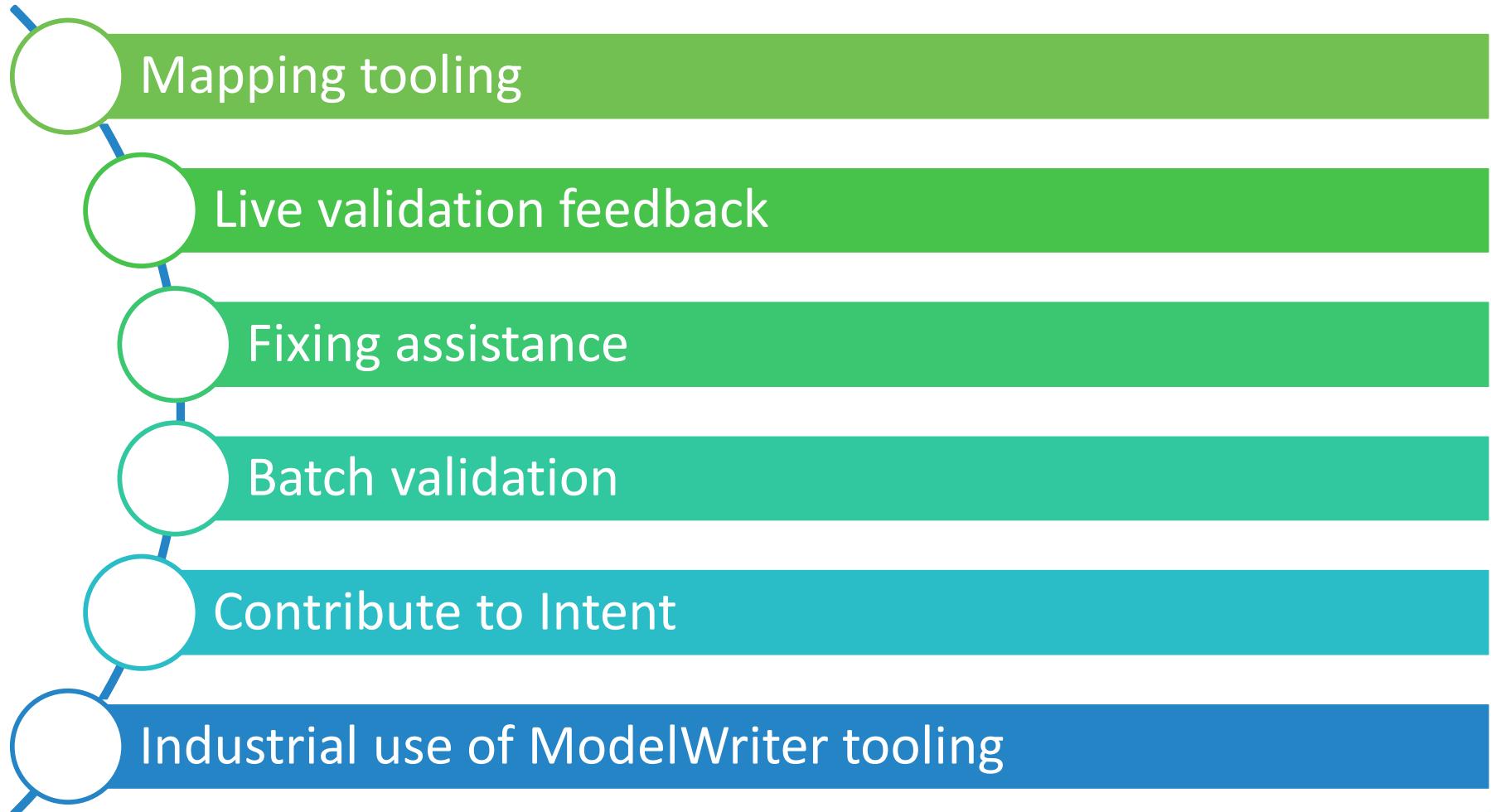
- Keep track of sourced documents
- Keep representation and portal up-to-date

Synchronization gap

- Organizational changes
- Artifacts knowledge is volatile

Validation

- Produce documentation of organizational changes
- Quality



UC-FR-03 - Synchronization of Regulation Documentation with a Design Rule Repository

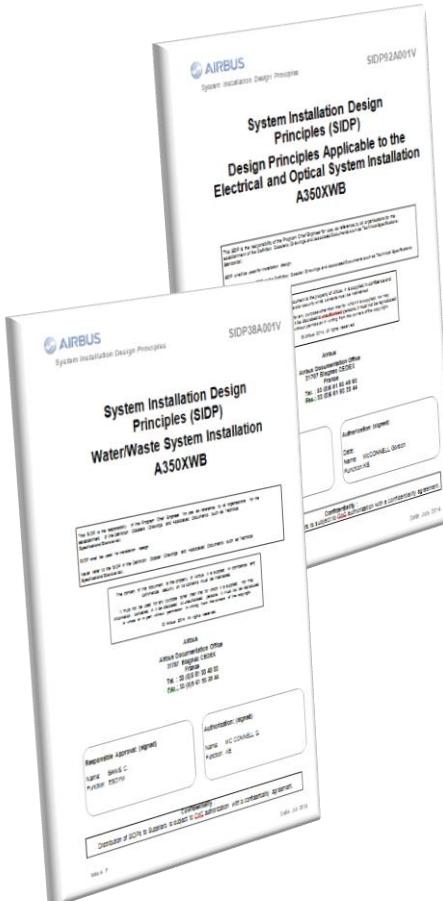
Anne MONCEAUX

Airbus Group Innovation

Louis ROUCH, Ayhan MOHOVIC

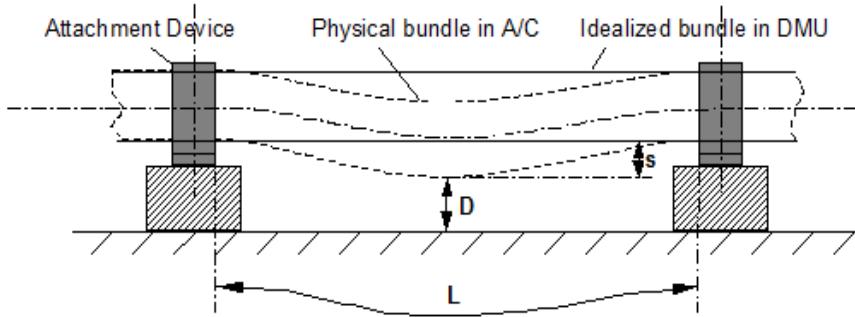
Airbus SAS

SIDP: “System Installation Design Principles”



SIDP92A001V-A-784

For installation of optical and electrical harnesses additional clearance for sagging (s) shall be provided as detailed below:



s ... Sagging of bundle (real behavior of physical bundle in A/C due to gravity, ageing, etc.)

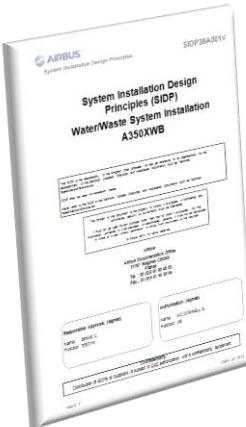
D... Required Distance

L... Actual length of a bundle segment between two Attachment Points (as designed in DMU)

Figure 6: Sagging of bundles between attachment points

Note: Unless the bundle has a straight routing, L is bigger than the pitch between the Attachment Points.

Context and problem



- SIDP documents explain how to install the aircraft systems and attach them to the structure. They capitalize the best practices & proven technical solutions.
- SIDP are defined per **ATA chapter** (~functional domain) to be applied for **each given A/C project**: for example “A350 Electrical installation”
- SIDP are **open to Extended Enterprise**: installation tasks are performed by risk sharing partners.
- SIDP are **living documents**: during the aircraft development any new DP allowing to satisfy all targets/constraints can be added, assuming it is validated by Airbus dedicated committee.

Industrial high level needs

To improve SIDP creation, maintenance and consultation in order to:

- save costs by supporting DP consultation / retrieval in accordance to the needs
- avoid non-compliance of design caused by DP updating lead times
- keep traceability with upstream regulations and requirements and downstream design models

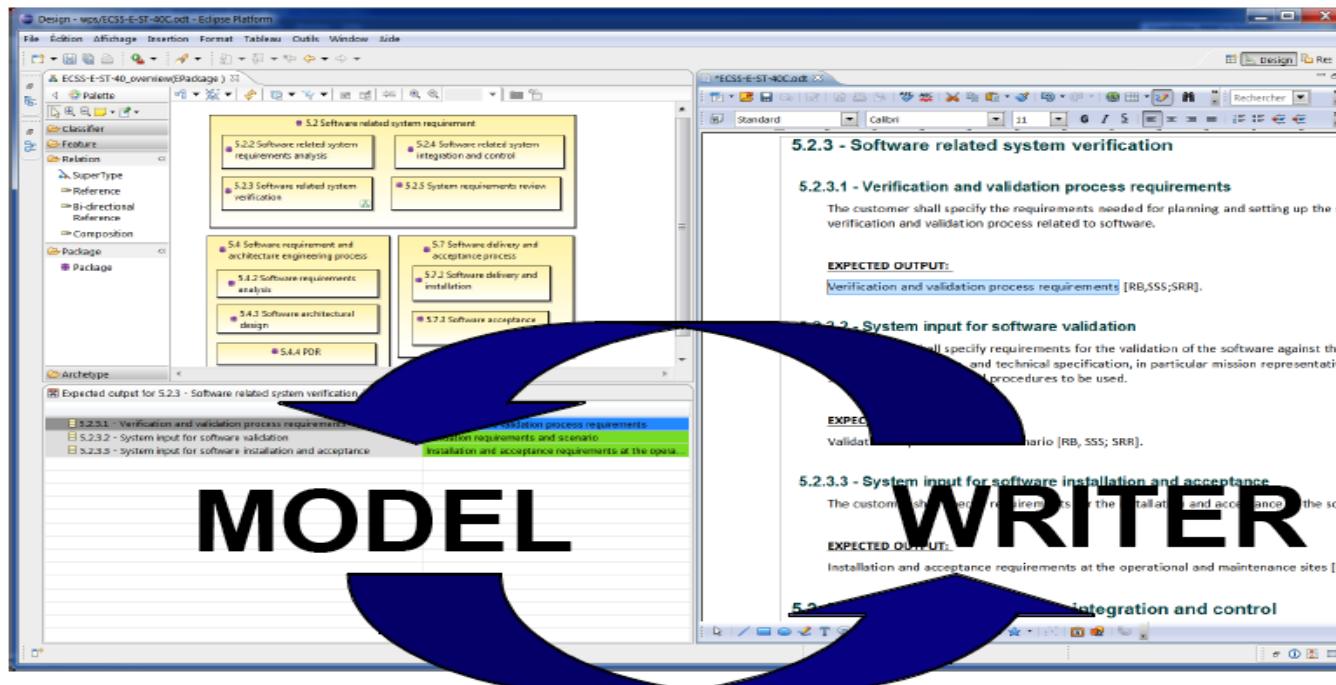
Synchronization of regulation documentation with a design rule repository

UC-FR-03 Synchronization of regulation documentation with a design rule repository



Goal: synchronizing the SIDP database content with documentation

- Create links between text fragments & model elements (manual annotation?, semi-automatic?...)
- Manage consistent synchronization (manage changes)



UC-FR-03 Synchronization of regulation documentation with a design rule repository



- Approach: limited case on Electrical Installation functional domain
- (Confidential Data - Non Disclosure Agreement finalized in June 2015).
 - Text
 - 1 document: SIDP ATA 92
 - In our industrial context SIDP are edited using MSWord
 - Models
 - An OWL model is built that reflects the DB schema: “Rule ontology” (30 classes, 35 object properties and 54 data properties)
 - Automatic population mechanism of the model from a csv BD export produces the KB (45781 triples)

UC-FR-03 Synchronization of regulation documentation with a design rule repository



Status 1st iteration: synchronizing the SIDP KB content and text part

Model part = rule KB

Textual part = rule sentences

Thermal variation shall be taken into account
Structural deformation shall be taken into account
Pressurization taken into account
Component/Item qualified to withstand temperature reached by the routes they are in contact with

UC-FR-04 - Production of a Context Specific Design Document

Anne MONCEAUX

Airbus Group Innovation

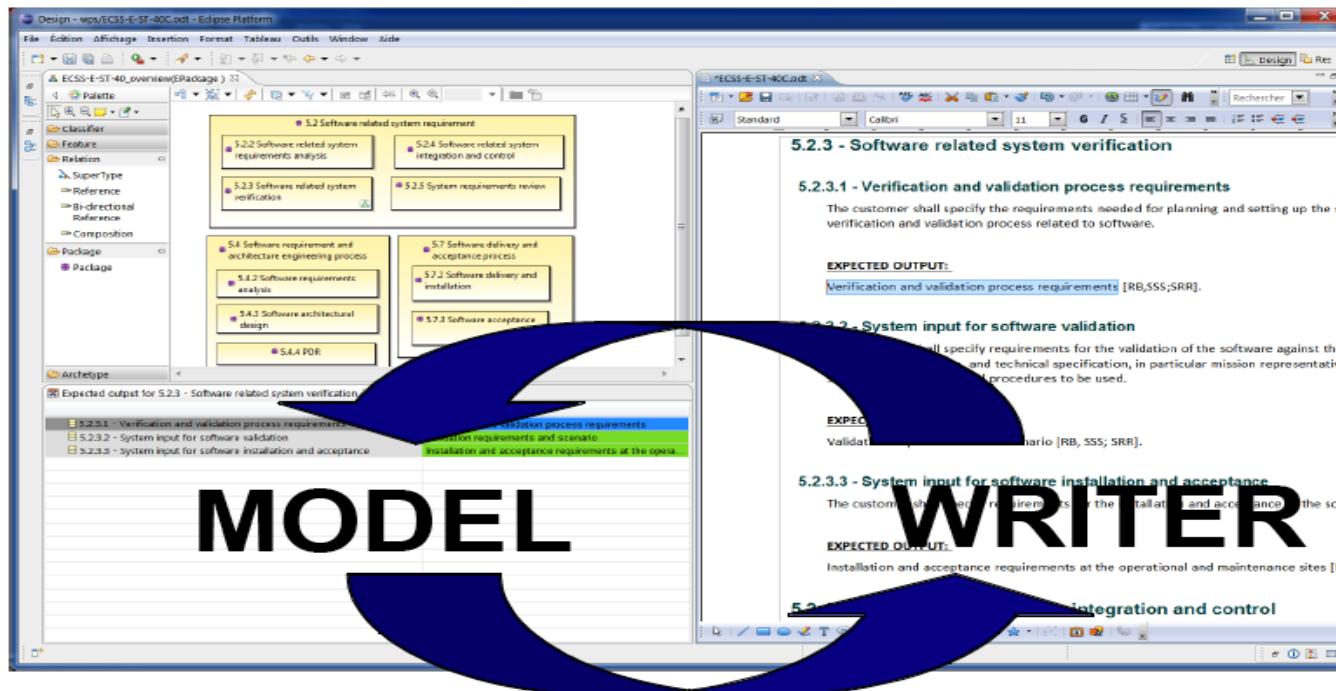
Louis ROUCH, Ayhan MOHOVIC

Airbus SAS

UC-FR-04 Production of a context specific design document

Goal: producing document according to usage “needs”

- Produce “filtered” document with subset of Design Principle textual elements according to usage “needs”

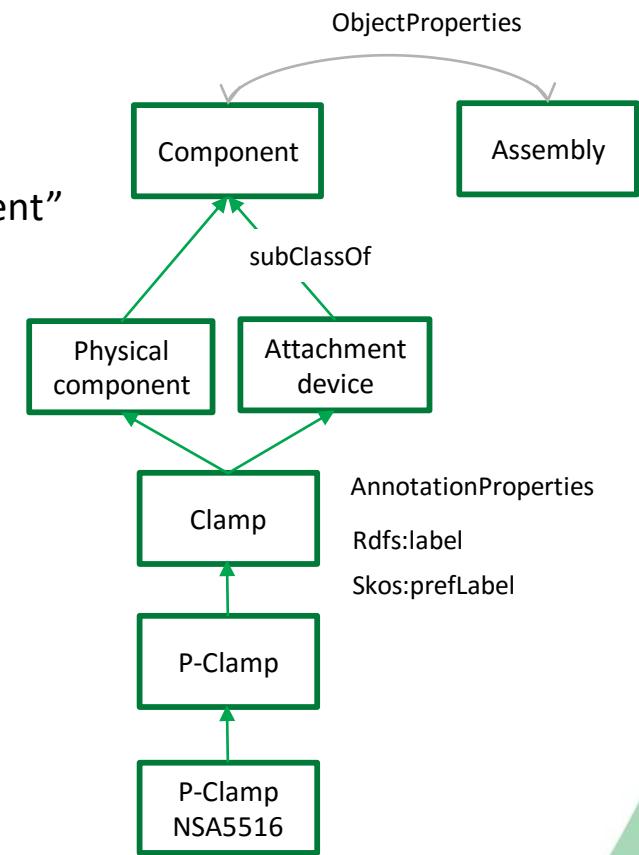
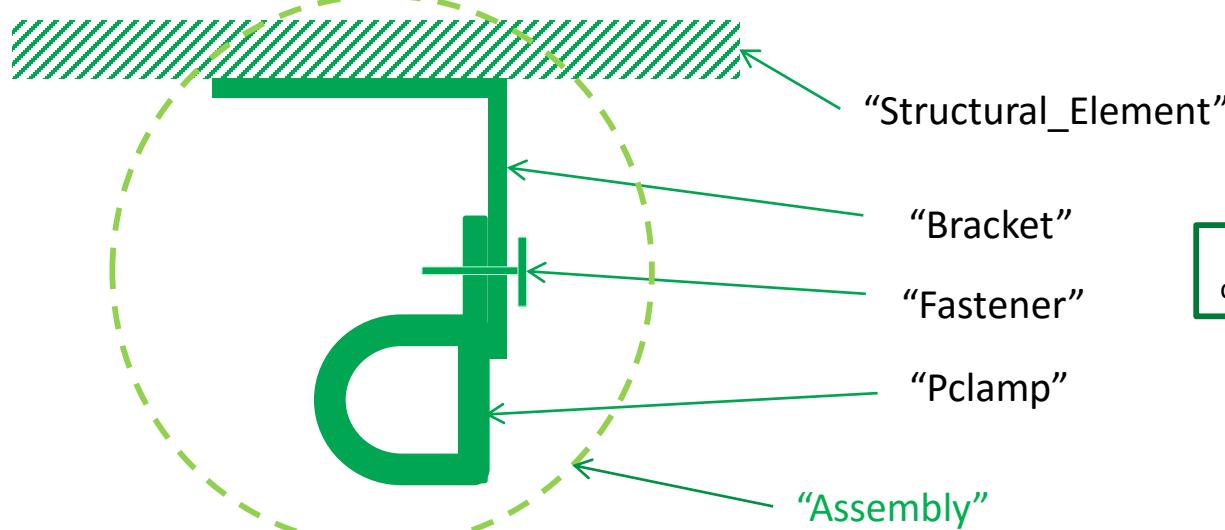


UC-FR-04 Production of a context specific design document



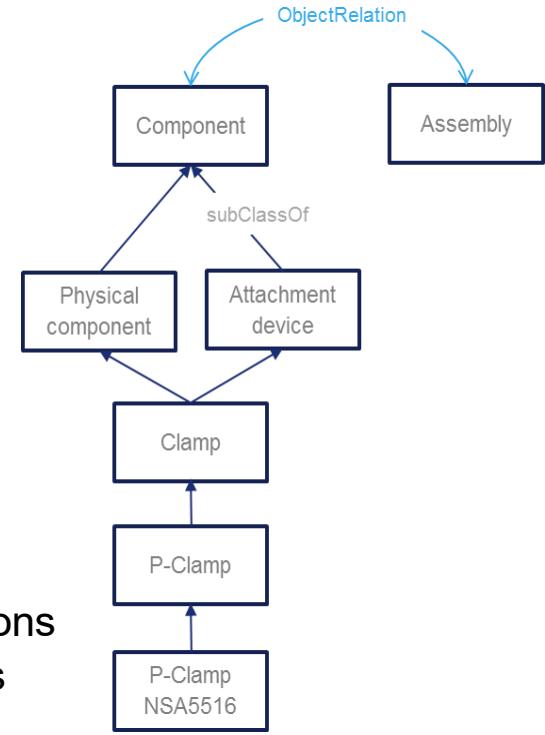
- Approach:
 - use case Electrical Installation functional domain
 - Confidential Data - Non Disclosure Agreement finalized in June 2015
 - Use model elements to retrieve relevant Design Principles
- Text
 - 1 document: SIDP ATA 92
 - In our industrial context SIDP are edited using MSWord
- Models
 - The previous Rule KB (populated Rule ontology)
 - **Component ontology** (476 classes, 21 ObjectProperties and 35 DataProperties)

Component classes taxonomy



Component classes taxonomy

“P-clamp” NSA5516 can be fixed on X with Y
 “Physical component” “Standard reference”

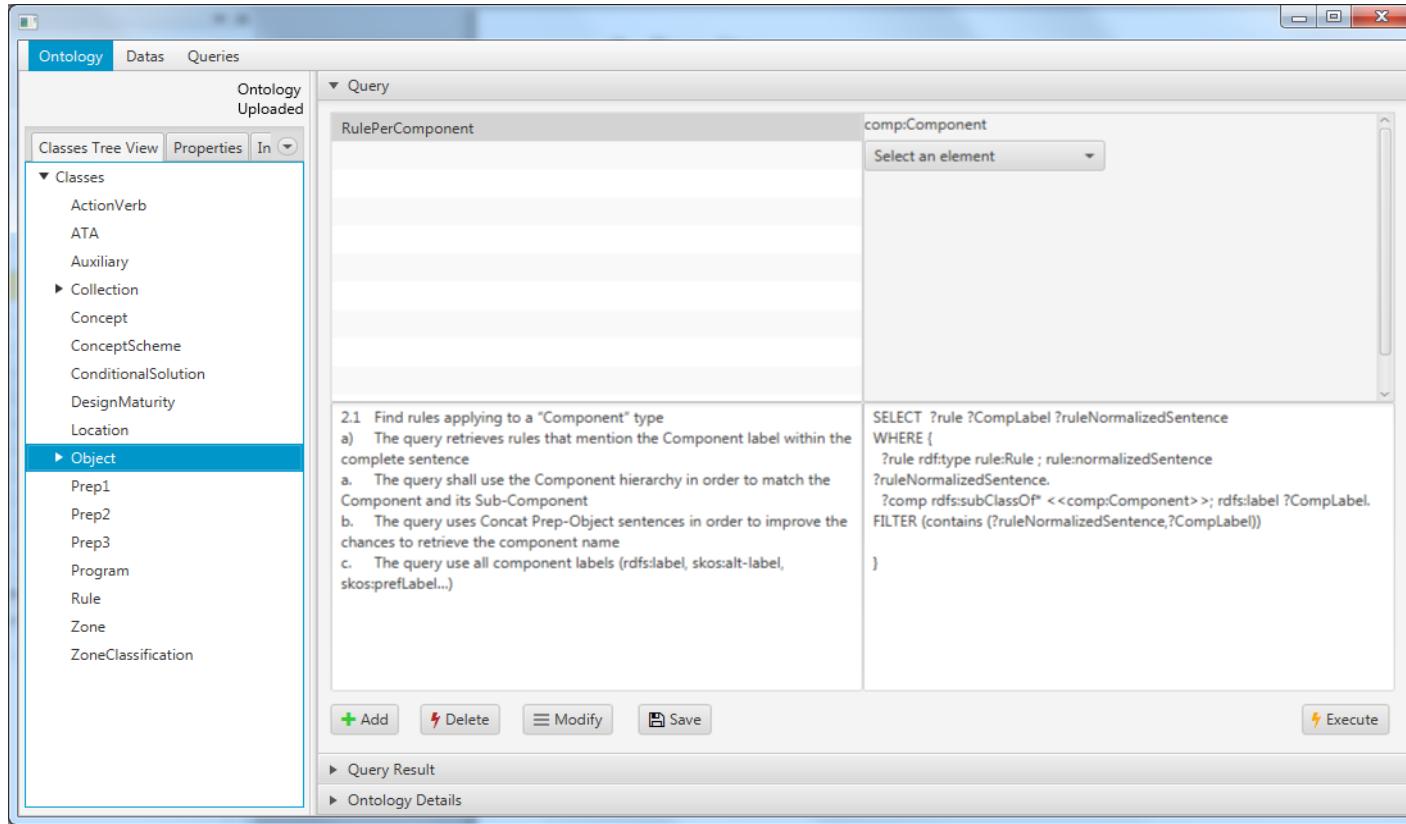


- NLP **Parsing** uses this taxonomy. Labels + assumptions such as a physical component may be referred to using its name or its reference or both concatenated
- **Inference** rule: a rule applying to a component type (Attachment device) applies to its subtypes (P-clamp)
- **Document display** : when searching rules applying to a component type (P-Clamp) → retrieve and display rules applying to super-types

SPARQL queries

Status : model based queries specification

- Preliminary study with Loria (Text to RDF)
- 1 internship on RDFizer and Query management



The screenshot shows the RDFizer application interface. On the left, there's an 'Ontology' tab with a tree view of classes like ActionVerb, ATA, Auxiliary, Collection, Concept, ConceptScheme, ConditionalSolution, DesignMaturity, Location, and Object. The 'Object' node is selected. Below the tree are buttons for Add, Delete, Modify, Save, and Execute. The main area has tabs for 'Query' and 'Data'. In the 'Query' tab, there's a section for 'RulePerComponent' with a dropdown menu 'Select an element'. Below it is a detailed description of a query and its SPARQL code.

2.1 Find rules applying to a "Component" type

a) The query retrieves rules that mention the Component label within the complete sentence

a. The query shall use the Component hierarchy in order to match the Component and its Sub-Component

b. The query uses Concat Prep-Object sentences in order to improve the chances to retrieve the component name

c. The query use all component labels (rdfs:label, skos:alt-label, skos:preflabel...)

```

SELECT ?rule ?CompLabel ?ruleNormalizedSentence
WHERE {
  ?rule rdf:type rule:Rule ; rule:normalizedSentence
  ?ruleNormalizedSentence.
  ?comp rdfs:subClassOf* <> comp:Component>; rdfs:label ?CompLabel.
  FILTER (contains (?ruleNormalizedSentence,?CompLabel))
}
  
```

UC-TR-01 - Documents of Quality Assurance Department

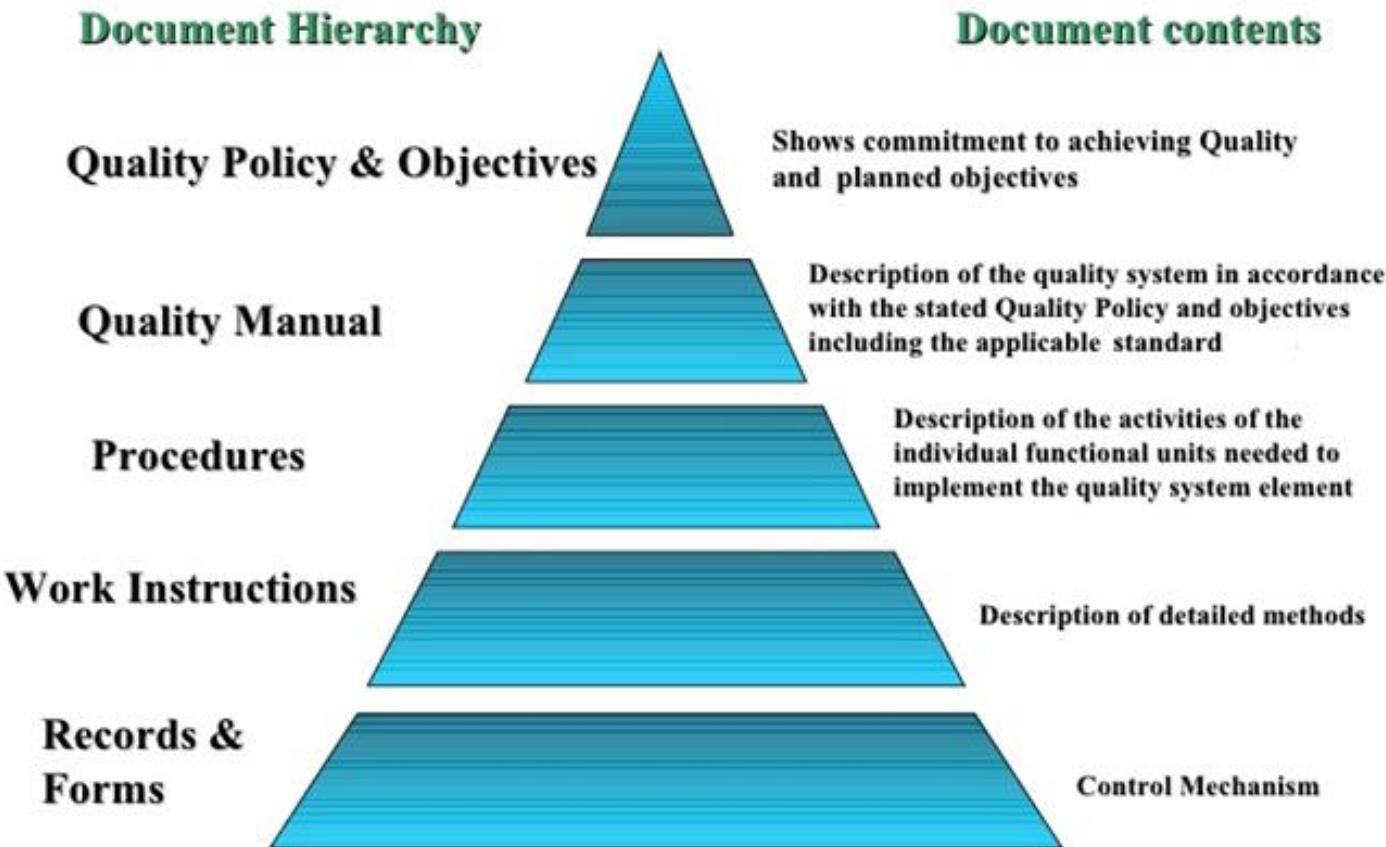
*Ersan GÜRDOĞAN
HISBİM*

T1.4 – Industrial Use Cases for Turkish Consortium

UC-TR-01	Documents of Quality Assurance Department
Description	To create faster and more accurate the forms that are used in quality control progress and trigger related forms (re-work form triggering, revision needed, approved, rejected etc.)
Actors	Quality Managers, Quality Measurement Specialists, Quality Control Personnel, Quality Auditors, Production Crews
Assumptions	Quality control measurement units are standart Rejected and Approved products forms are standart Quality Certification standarts are always applied

TR-UC-01 Documents of Quality Assurance Department

Quality Management System Documentation



TR-UC-01 Documents of Quality Assurance Department

What we expect ModelWriter about QDMS:

- Quality documents and forms are mostly too generic and similar. However some customers and users would like to customized documents. We would like to develop ModelWriter system could automatically generated QDMS related docs and forms with respect to Quality Test results.
- With sharing utility QDMS documents will be send to company managers, quality department authorised personnel, related customers' responsibles and suppliers2 responsibles if manufactured part comes from subcontractor.

What we have done until today about QDMS use case:

- Customer segmentation done, customer database and data relations are ready for ModelWriter
- Suppliers information are acqisied from database, ready to integrate ModelWriter system
- Models of Quality Documents and forms have been completed

UC-TR-02 - Non-Disclosure Agreements

Ersan GÜRDOĞAN
HISBİM

T1.4 – Industrial Use Cases for Turkish Consortium

UC-TR-02	Non-Disclosure Agreements
Version	V1.0.0 dated 15-Nov-2014
Description	<p>Non-Disclosure Agreement (NDA), also known as a confidentiality agreement (CA), confidential disclosure agreement (CDA), proprietary information agreement (PIA), or secrecy agreement (SA), is a legal contract between at least two parties that outlines confidential material, knowledge, or information that the parties wish to share with one another for certain purposes, but wish to restrict access to or by third parties. It is a contract through which the parties agree not to disclose information covered by the agreement. An NDA creates a confidential relationship between the parties to protect any type of confidential and proprietary information or trade secrets. As such, an NDA protects nonpublic business information.</p> <p>NDAs are commonly signed when two companies, individuals, or other entities (such as partnerships, societies, etc.) are considering doing business and need to understand the processes used in each other's business for the purpose of evaluating the potential business relationship</p>
Actors	Responsible/Authorised personnel in both parties.

TR-UC-02 Non-Disclosure Agreements

MUTUAL NON-DISCLOSURE AGREEMENT

In connection with discussions between _____ ("Company") and the
_____ ("Company 2"), with respect to a

[REDACTED]

This Agreement shall be governed by and interpreted in accordance with the laws of the European Commission.

This Agreement shall commence on the date last signed below

Company

Company 2

Signature: _____

Print or type name: _____

Title: _____

Date: _____

Signature: _____

Print or type name: _____

Title: _____

Date: _____

Company Information of
Parties who sign NDA

Items of NDA with respect
to contracted project(s)
between parties, project
related information such
date of document, content
of project etc.

Responsible names and their
signatures from each party

TR-UC-02 Non-Disclosure Agreements

What we expect ModelWriter about NDAs:

- Automatically captured parties information, main company and customer information
- Automatically filled names and signatures (if digital/electronical signatures)
- Automaticaly generate items according to given project ID
- With sharing utility ModelWriter will be able to send finalised and revised NDA to related parties and their contracted lawyers to review.

What we have done until today about NDA use case:

- Company information acquisition from database is done
- Customer segmentations completed ready to integrate and acquisite their data to ModelWriter
- Projects database relations have been built, ready to integrate ModelWriter engine

UC-TR-03 - Synchronization of ReqIF models from requirement specifications

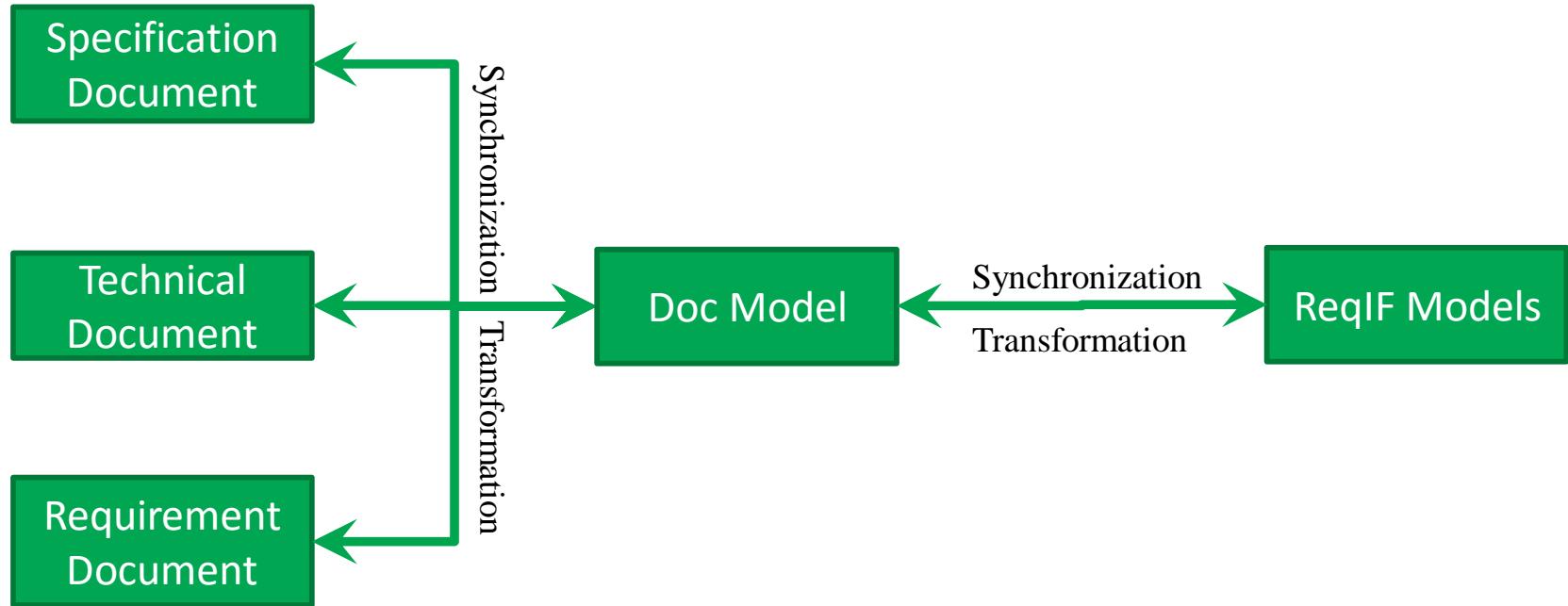
Ferhat Erata
UNIT

UC-TR-03 - Synchronization of ReqIF models from requirement specifications



- Technical documents are usually long and have complex structure
 - For example requirement or specification documents
- These documents keep changing in the time frame and need to be consistent with the other artifacts
 - For example, with a ReqIF model
- In this use case we aim to keep these documents and models synchronized
 - This will include bidirectional transformation of documents and models

UC-TR-03 - Synchronization of ReqIF models from requirement specifications



- At the current progress status of this UC:
 - The transformation is done in one way (left to right)
 - The synchronization is done only between DocModel and ReqIF models

Sample Applications:

- Airbus SIDP templates \leftrightarrow ReqIF
 - Havelsan Requirement doc templates \leftrightarrow ReqIF
 - University Management System docs \leftrightarrow ReqIF
 - Eclipse RMF specifications \leftrightarrow ReqIF
-
- The initial implemented version of this use case will be presented in the demonstration session.



Design Principles Applicable to the xx System
Installation - Program

SIDPREF1

7 Design Principles

7.1 Design Principles Applicable to the Entire Design

General

SIDP92A001V-A-269

The effects of thermal variations, structural deformation, pressurization variation, etc., shall be taken into account.

SIDP92A001V-A-280

Each item in direct contact with ATA92 bundles shall be qualified to withstand conditions detailed in Table 3 below.

Type of Route	All areas except in high temperature zones		High temperature zones	
	Peak condition *	Continuous operating condition **	Peak condition *	Continuous operating condition **
G routes	170°C	170°C	260°C	260°C
P, X routes	150°C	95°C	260°C	200°C
S, R, T, U, V routes	85°C	85°C	130°C	130°C
Others types (M, S, Q,...)	135°C	95°C	260°C	200°C

* duration of 100 hours
** duration A/C life

Table 3. Operating conditions for items in direct contact with ATA92 bundles

Attachment devices placed inside boxes, which contain power cables, shall withstand:

- a minimum of 150°C for peak condition and
- a minimum of 110°C for continuous operating condition.

ID	Description
R 1	Introduction
R 1.1	Purpose
R 1.2	Definitions
R 1.3	Nomenclature/Abbreviation
R 1.3.1	A/C Aircraft
R 1.3.2	ATA Air Transport Association
R 1.4	Document Precedence
R 2	Objectives
R 3	Reference Regulations/Documents
R 3.1	Airworthiness Regulations
R 3.1.1	JAR 25.607 Fasteners
R 3.1.2	REF Title
R 3.2	Others
R 3.2.1	A350XWB_SIDP V&V Policy PL0901917
R 4	Responsibilities
R 5	Structures/Systems Configuration
R 6	Application Domain
R 7	Design Principles
R 7.1	Design Principles Applicable to the Entire Design
R 7.1.1	General
R 7.1.3	SIDP92A001V-A-280
R 7.1.4	SIDP92A001V-A-356
R 7.1.4.1	Locking Of Bolted Fastenings
R 7.1.2	SIDP92A001V-A-269
R 7.1.5	SIDP92A001V-A-413
R 7.1.6	SIDP92A001V-A-424
R 7.1.6.1	Installation In Ceiling Area
R 7.1.7	SIDP92A001V-A-3763
R 7.1.7.1	Installation In Fuel Tanks
R 7.1.8	SIDP92A001V-A-472
R 7.1.9	Positioning of Bundles in the Aircraft Considering Environmental Constraints
R 7.1.9.1	General
R 7.1.9.2	SIDP92A001V-A-557
R 7.1.9.3	SIDP92A001V-A-579
R 7.1.10	Segregation or Clearances of Bundles to A/C Structure, other Systems or Between
R 7.1.10.1	General Applications
R 7.1.10.1.1	Sagging (s)
R 7.1.10.1.2	SIDP92A001V-A-784

HAVELSAN OZEL

	HAVELSAN YGO PROJESİ YAZILIM KONFIGÜRASYON YÖNETİMİ TEKNİK ŞARTNAMESİ	Doküman No : HVL-YGO-TS-003 Yayın No : 1.0 Yayın Tarihi : Ağustos 2011
--	---	--

1. İSTEK VE ÖZELLİKLER**1.1 GENEL ÖZELLİKLER****1.1.1 Yönetim ve Yapılandırma**

1.1.1.1 Bütün HAVELSAN Birimlerinin ve Projelerinin merkezi ve tek bir kurulum üzerinde çalışmasına olanak sağlanacaktır.

1.1.1.2 İşletim sistemlerinden bağımsız olarak grafik arayüz ile erişime olanak veren istemci sağlanacaktır.

1.1.1.3 Konfigürasyon yönetim sistemi sunucularına bağlı olmadan çalışmasına olanak sağlanacaktır.

1.1.1.4 Çoklu kullanıcı desteği sağlanacaktır.

1.1.1.5 İşletim sistemlerinden bağımsız olarak ve en az görüntüleme amaçlı ürün (Web) tabanlı çalışmasına olanak sağlanacaktır.

1.1.1.6 Yeni kullanıcı tanımlamasına, var olan kullanıcıların güncellenmesine ve silinmesine olanak sağlanacaktır.

1.1.1.7 Kullanıcıların gruplara atanmasına ve gruplardan çıkartılmasına olanak sağlanacaktır.

1.1.1.8 Kullanıcı profiline ve proje yapısına göre var olan deponun genişletilmesine, yeni depo tanımlamasına olanak sağlanacaktır.

1.1.2 Yetkilendirme ve Güvenlik

1.1.2.1 Kendi veritabanındaki kullanıcı bilgilerini kullanarak kullanıcı kimlik denetimi yapabilecektir.

1.1.2.2 Aktif Dizin'de (Active Directory) tanımlı kullanıcı bilgilerini kullanarak kullanıcı kimlik denetimi yapabilecektir.

platform:/resource/Demo/Havelsan.docmodel	
▼	Document
▼	Paragraph İSTEK VE ÖZELLİKLER
▼	Paragraph GENEL ÖZELLİKLER
▼	Paragraph Yönetim ve Yapılandırma
◆	Paragraph Bütün HAVELSAN Birimlerinin ve Projelerinin merkezi ve tek bir kuru
◆	Paragraph İşletim sistemlerinden bağımsız olarak grafik arayüz ile erişime olanak
◆	Paragraph Konfigürasyon yönetim sistemi sunucularına bağlı olmadan çalışılm
◆	Paragraph Çoklu kullanıcı desteği sağlanacaktır.
◆	Paragraph İşletim sistemlerinden bağımsız olarak ve en az görüntüleme amaçlı
◆	Paragraph Yeni kullanıcı tanımlamasına, var olan kullanıcıların güncellenmesi
◆	Paragraph Kullanıcıların gruplara atanmasına ve gruplardan çıkartılmasına olan
◆	Paragraph Kullanıcı profiline ve proje yapısına göre var olan deponun genişletil
◆	Paragraph Yetkilendirme ve Güvenlik
◆	Paragraph İşlevsellik
◆	Paragraph İzleme ve Rapor
◆	Paragraph Entegrasyon
◆	Paragraph Dış sistemlerle tümleştirme için Uygumala Geliştirme Arayüzü sağlar
◆	Paragraph Depodaki bir konfigürasyon elemanından başlanarak bu ögenin altı
◆	Paragraph Dosya sistemindeki bir dizin ve altının özyine olarak içeri alınmasını
◆	Paragraph LDAP veya Aktif Dizin bağlantısı için gerekli olan parametrelerin giril
◆	Paragraph Ürün Kılavuzları
◆	Paragraph Satıcı/Yüklenici ürünle ilgili Tablo 3'de listelenen kılavuzları geçici ka
◆	Paragraph Kullanıcı kılavuzları Türkçe veya İngilizce olarak sağlanacaktır.
◆	Paragraph Ürün için çevirmişi yardım ekranları sağlanacaktır.
◆	Paragraph Eğitimler
◆	Paragraph Sistemlerin kullanımı alınması ve yaygınlaştırılması için Tablo 4'de li
◆	Paragraph Eğitim süreleri yüklenici önerisi ve HAVELSAN onayı ile değiştirilebil
◆	Paragraph Satıcı/Yüklenici tarafından katılımcı sayısı kadar basılı eğitim matery
◆	Paragraph Eğitim yeri HAVELSAN/ANKARA tesisleridir.
◆	Paragraph Satıcı/Yüklenici eğitimden bir hafta önce eğitici özgeçmişini, eğiti

ID	Description
R 1	İSTEK VE ÖZELLİKLER
R 1.1	İSTEK VE ÖZELLİKLER
R 1.1	GENEL ÖZELLİKLER
R 1.1.1	Yönetim ve Yapılandırma
R 1.1.1.1	Bütün HAVELSAN Birimlerinin ve Projelerinin merkezi ve tek bir kurulum üzerinde çalışmasına olanak sağlanması
R 1.1.1.2	İşletim sistemlerinden bağımsız olarak grafik arayüz ile erişime olanak veren istemci sağlanacaktır.
R 1.1.1.3	Konfigürasyon yönetim sistemi sunucularına bağlı olmadan çalışmasına olanak sağlanması
R 1.1.1.4	Çoklu kullanıcı desteği sağlanacaktır.
R 1.1.1.5	İşletim sistemlerinden bağımsız olarak ve en az görüntüleme amaçlı ürün (Web) tabanlı çalışmasına olanak sağlanması
R 1.1.1.6	Yeni kullanıcı tanımlamasına, var olan kullanıcıların güncellenmesine ve silinmesine olanak sağlanması
R 1.1.1.7	Kullanıcıların gruplara atanmasına ve gruplardan çıkartılmasına olanak sağlanması
R 1.1.1.8	Kullanıcı profiline ve proje yapısına göre var olan deponun genişletilmesine, yeni depo tanımlamasına olanak sağlanması
R 1.1.2	Yetkilendirme ve Güvenlik
R 1.1.2.1	Kendi veritabanındaki kullanıcı bilgilerini kullanarak kullanıcı kimlik denetimi yapabilecektir.
R 1.1.2.2	Aktif Dizin'de (Active Directory) tanımlı kullanıcı bilgilerini kullanarak kullanıcı kimlik denetimi yapabilecektir.
R 1.1.2.3	Aktif Dizin'den elde edilmiş kullanıcı kimliğini, Tek Giriş (Single Sign On - SSO) ilkesine göre doğrulanmış kullanıcılar için yetkilendirme yapılacaktır.
R 1.1.2.4	Nesne erişimlerinin yetkilendirilmesi için, en az "Ekleme", "Silme", "Düzenleme", "Görüntüleme" yetkileri sağlanacaktır.
R 1.1.2.5	Proje yönetim ve bakım işlevlerinin yetkilendirilmesi için "Yönetim" yetkisi sağlanacaktır.
R 1.1.2.6	Proje yönetim ve bakım işlevleri ile nesneler üzerinde yapılabilen tüm iş ve işlemleri ya
R 1.1.2.7	Nesneler üzerinde yapılabilen tüm iş ve işlemleri yapma yetkisine sahip "Düzenleyici" yetkisi sağlanacaktır.
R 1.1.2.8	Nesneler üzerinde yapılabilen tüm iş ve işlemleri yapma yetkisine sahip "Görüntüleyici" yetkisi profil sa
R 1.1.2.9	Yetkili profillerinin kullanıcılara/gruplara atanmasına olanak sağlanması
R 1.1.2.10	Kullanıcıların/grupların dosya/dizinler üzerinde seçilebilecek yetkiler ile yetkilendirme yetkisi sağlanacaktır.
R 1.1.2.11	Kullanıcıların/grupların dosya/dizinler üzerinde seçilebilecek yetkiler ile yetkilendirme yetkisi sağlanacaktır.
R 1.1.3	İşlevsellik
R 1.1.3.1	Konfigürasyon elemanlarının çalışma kopyasının yaratılmasına olanak sağlanması
R 1.1.3.2	Çalışma kopyasında değişiklik yapılan konfigürasyon elemanlarının depoya gönderilmesi
R 1.1.3.3	Konfigürasyon elemanı üzerinde yapılan değişikliğin depoya gönderilmesi sırasında y
R 1.1.3.4	Komut satırından çalışma olanağı sağlanacaktır.
R 1.1.3.5	Depoya gönderme işleminin parçalanamaz şekilde (atomic commit) gerçekleşmesi sağlanacaktır.
R 1.1.3.6	Yan dal (branch) açmaya olanak sağlanması

1 USE CASE UC1: SIGN IN

Primary Actor: Student, Lecturer.

Stakeholders and Interests:

- Student: Wants simple user interface, fast response, no system errors.
- Lecturer: ?

Preconditions:

- Student is registered.

Success Guarantee (Postcondition):

- Student is logged in.

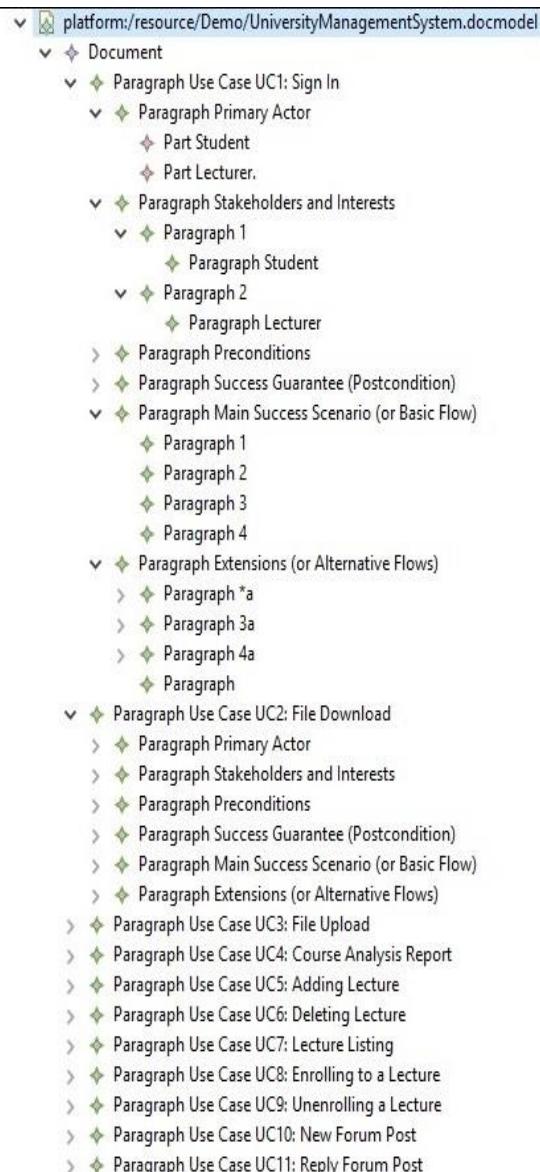
Main Success Scenario (or Basic Flow):

1. Student visits system home page.
2. System shows home page with login form and sign up button.
3. Student enters his/her username and password then click login button.
4. System shows Student's home page.

Extensions (or Alternative Flows):

*a. At any time, system fails, to support recovery, ensure all transaction sensitive state be recovered from any step of the scenario.

1. Student restarts System and requests recovery of prior state.
2. System reconstructs prior state.
 - 2a. System detects anomalies preventing recovery.
 1. System signals error to the Student, records the error, and exits state.
- 3a. Student enters invalid username or password.
 1. System shows errors and request to Student to retry.
 2. Student enters his/her username and password.
- 4a. System detects failure to communicate with server.
 1. System signals error and rejects the request.



ID	Description
R 1	Use Case UC1: Sign In
R 1.1	Primary Actor
R 1.1.1	Student
R 1.1.2	Lecturer.
R 1.2	Stakeholders and Interests
R 1.2.1	Student
R 1.2.1.1	Wants simple user interface, fast response, no system errors.
R 1.2.1.2	Lecturer
R 1.2.1.3	?
R 1.3	Preconditions
R 1.3.1	Student is registered.
R 1.4	Success Guarantee (Postcondition)
R 1.4.1	Student is logged in.
R 1.5	Main Success Scenario (or Basic Flow)
R 1.5.1	Student visits system home page
R 1.5.2	System shows home page with login form and sign up button
R 1.5.3	Student enters his/her username and password then click login b
R 1.5.4	System shows Student's home page
R 1.6	Extensions (or Alternative Flows)
R 1.6.1	At any time, system fails, to support recovery, ensure all transaction sensitive state and events can be recovered from any step of the scenario
R 1.6.1.1	Student restarts System and requests recovery of prior state
R 1.6.1.2	System reconstructs prior state
R 1.6.1.2.1	System detects anomalies preventing recovery
R 1.6.1.2.1.1	System signals error to the Student, records the error, and enters a
R 1.6.2	Student enters invalid username or password
R 1.6.2.1	System shows errors and request to Student to retry
R 1.6.2.2	Student enters his/her username and password
R 1.6.3	System detects failure to communicate with server
R 1.6.3.1	System signals error and rejects the request
R 2	Use Case UC2: File Download
R 2.1	Primary Actor
R 2.1.1	Student
R 2.2	Stakeholders and Interests
R 2.2.1	Student

1 USE CASE UC1: CREATE A NEW SPECOBJECT

Preconditions:

- ReqIF model exists and is open.

Main Success Scenario (or Basic Flow):

1. We assume that a Specification exists and is open (not required for alternative scenario)
2. Double click on the cell in the Specification Editor to be edited.
3. Select the Child or Sibling submenu.
4. Select the desired Spec Object Type (or none) from the submenu.
5. Note that some cells may not be editable, in which case nothing will happen.

Alternative 1 Create in Outline:

*a. The same workflow works for elements that are shown underneath "Specifications" in the outline.
 2a. It is also possible to create children of the "SpecObjects" folder in the outline, but in this SpecHierarchy will be created.

Alternative 2 Keyboard Shortcut:

*a. The keyboard shortcut Ctrl-Enter will create a SpecHierarchy sibling to the currently selected element and immediately go into edit mode in the currently selected column.

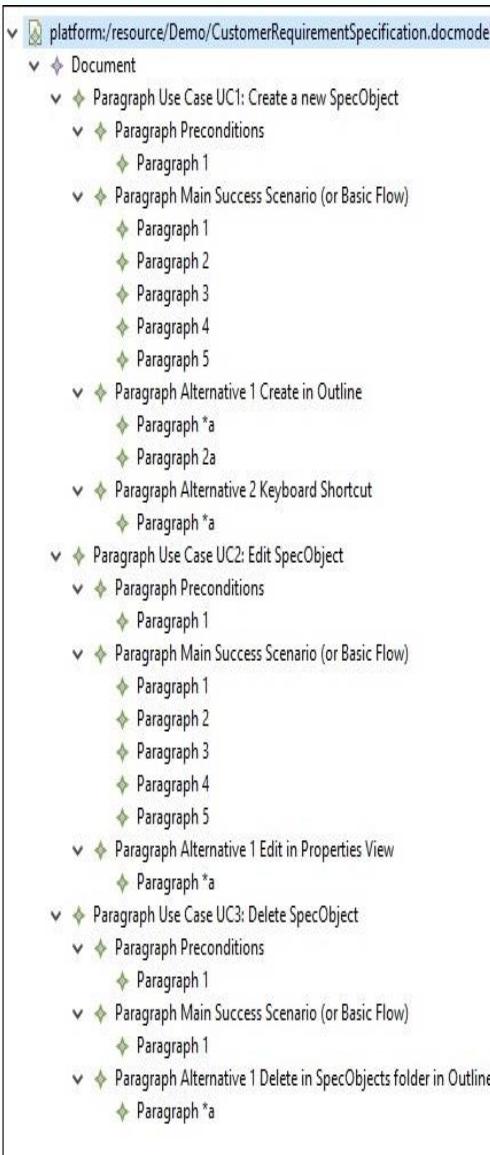
2 USE CASE UC2: EDIT SPECOBJECT

Preconditions:

- A ReqIF model exists, is open and at least one SpecObject exists.

Main Success Scenario (or Basic Flow):

1. We assume that a Specification exists and is open (not required for alternative scenario)
2. Open a row's context menu (or in the empty editor space)



ID	Description
R 1	Use Case UC1: Create a new SpecObject
R 1.1	Preconditions
R 1.1.1	ReqIF model exists and is open.
R 1.2	Main Success Scenario (or Basic Flow)
R 1.2.1	We assume that a Specification exists and is open (not required for alternative scenario)
R 1.2.2	Double click on the cell in the Specification Editor to be edited
R 1.2.3	Select the Child or Sibling submenu
R 1.2.4	Select the desired Spec Object Type (or none) from the submenu
R 1.2.5	Note that some cells may not be editable, in which case nothing will happen
R 1.3	Alternative 1 Create in Outline
R 1.3.1	The same workflow works for elements that are shown underneath "Specifications" in the outline
R 1.3.2	It is also possible to create children of the "SpecObjects" folder in the outline, but in this case, no SpecObject will be created.
R 1.4	Alternative 2 Keyboard Shortcut
R 1.4.1	The keyboard shortcut Ctrl-Enter will create a SpecHierarchy sibling to the currently selected element and immediately go into edit mode in the currently selected column.
R 2	Use Case UC2: Edit SpecObject
R 2.1	Preconditions
R 2.1.1	A ReqIF model exists, is open and at least one SpecObject exists.
R 2.2	Main Success Scenario (or Basic Flow)
R 2.2.1	We assume that a Specification exists and is open (not required for alternative scenario)
R 2.2.2	Open a row's context menu (or in the empty editor space)
R 2.2.3	Alternatively, hit enter or space in that cell
R 2.2.4	In both cases, the double-clicked / selected cell will switch to edit mode
R 2.2.5	This results in a new SpecHierarchy being created that is linked to a newly created SpecObject with the same name as the original cell.
R 2.3	Alternative 1 Edit in Properties View
R 2.3.1	A selected element (no matter whether in Specification Editor or Outline or elsewhere) will be shown in the Properties View.
R 3	Use Case UC3: Delete SpecObject
R 3.1	Preconditions
R 3.1.1	A ReqIF model exists, is open and at least one SpecObject exists.
R 3.2	Main Success Scenario (or Basic Flow)
R 3.2.1	If an element is deleted from the specification (so essentially a SpecHierarchy), and no references to the element exist, the element will be removed.
R 3.3	Alternative 1 Delete in SpecObjects folder in Outline
R 3.3.1	If the SpecObject is deleted from the SpecObjects folder in the outline, it will be removed, no matter what its parent is.

Next Steps

- Next steps:
 - ReqIF model transformation to documents (bi-directional transformation)
 - Synchronization between a document and Doc-Model (fully synchronization)



Model Writer

Work Package 1 - Industrial Use Cases and Requirements

Integration with Application Lifecycle Management tools

Eray TÜZÜN (HAVELSAN)

Yagup MACİT (HAVELSAN)

Requirement Work Item

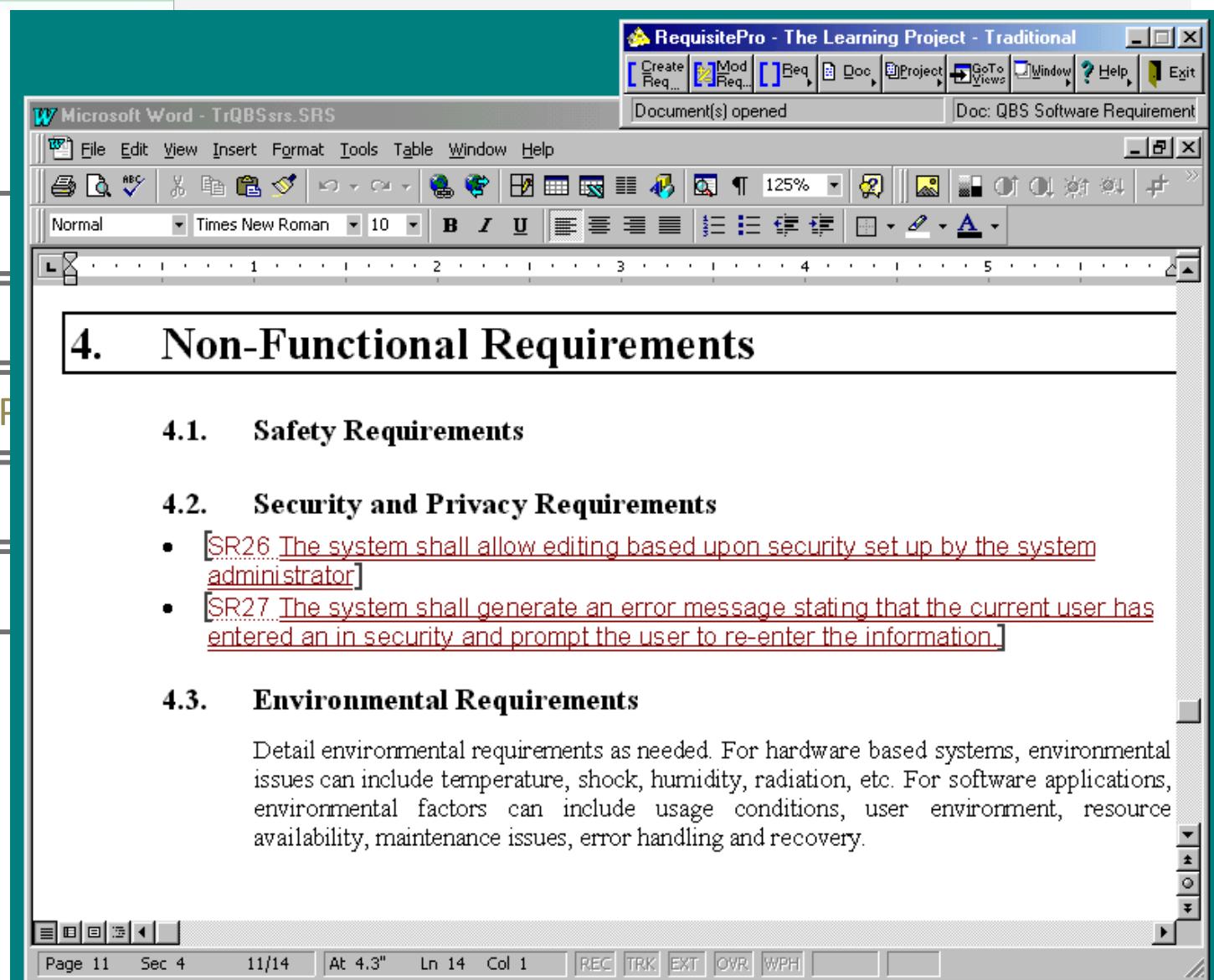
Customized Form

Attributes

WorkItem Number P

History

Discussion



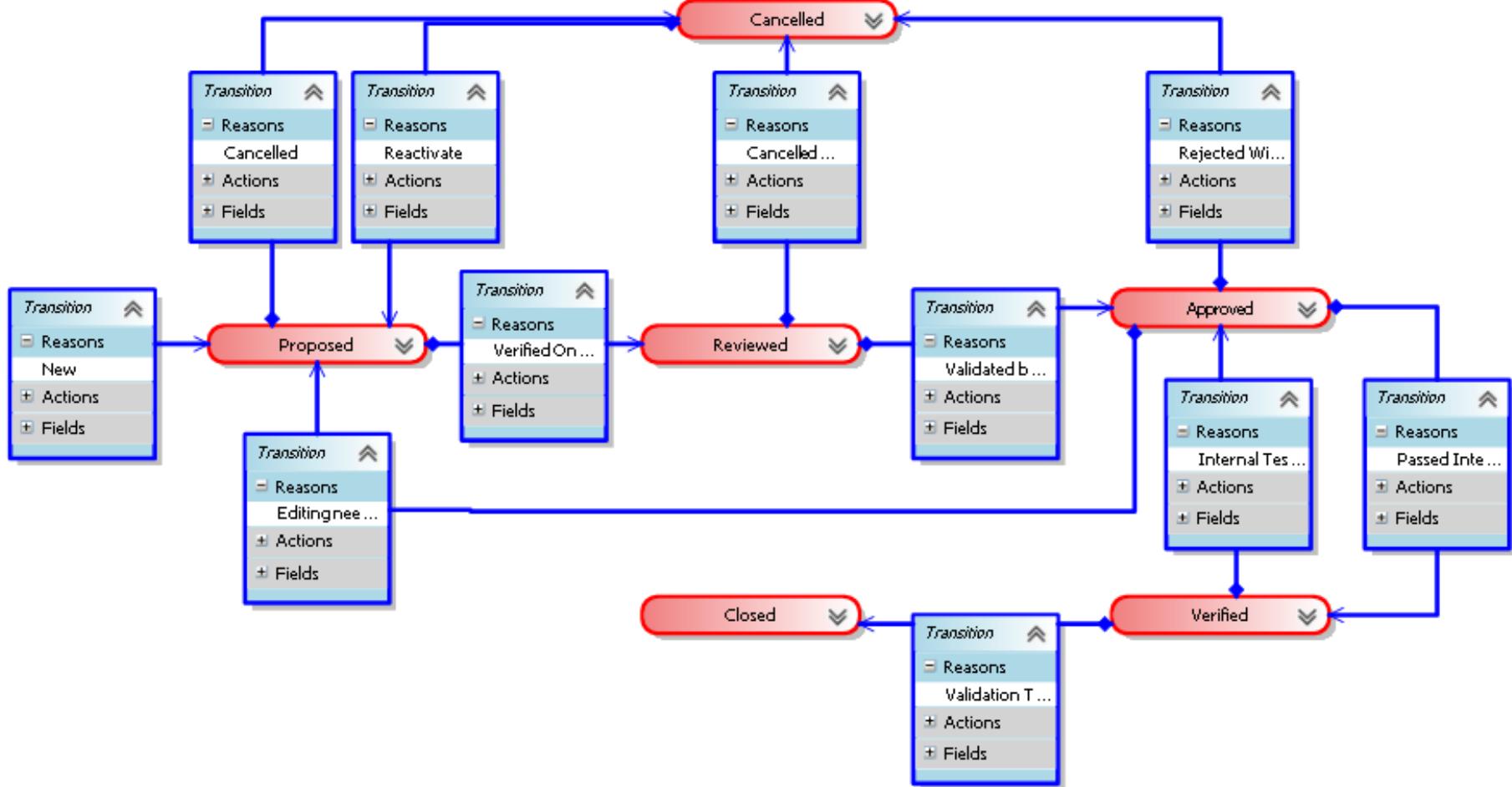
The screenshot shows a Microsoft Word document titled "TrQBSsrs.SRS" open in RequisitePro. The document contains the following structure:

- 4. Non-Functional Requirements**
 - 4.1. Safety Requirements**
 - 4.2. Security and Privacy Requirements**
 - [SR26 The system shall allow editing based upon security set up by the system administrator]
 - [SR27 The system shall generate an error message stating that the current user has entered an in security and prompt the user to re-enter the information.]
 - 4.3. Environmental Requirements**

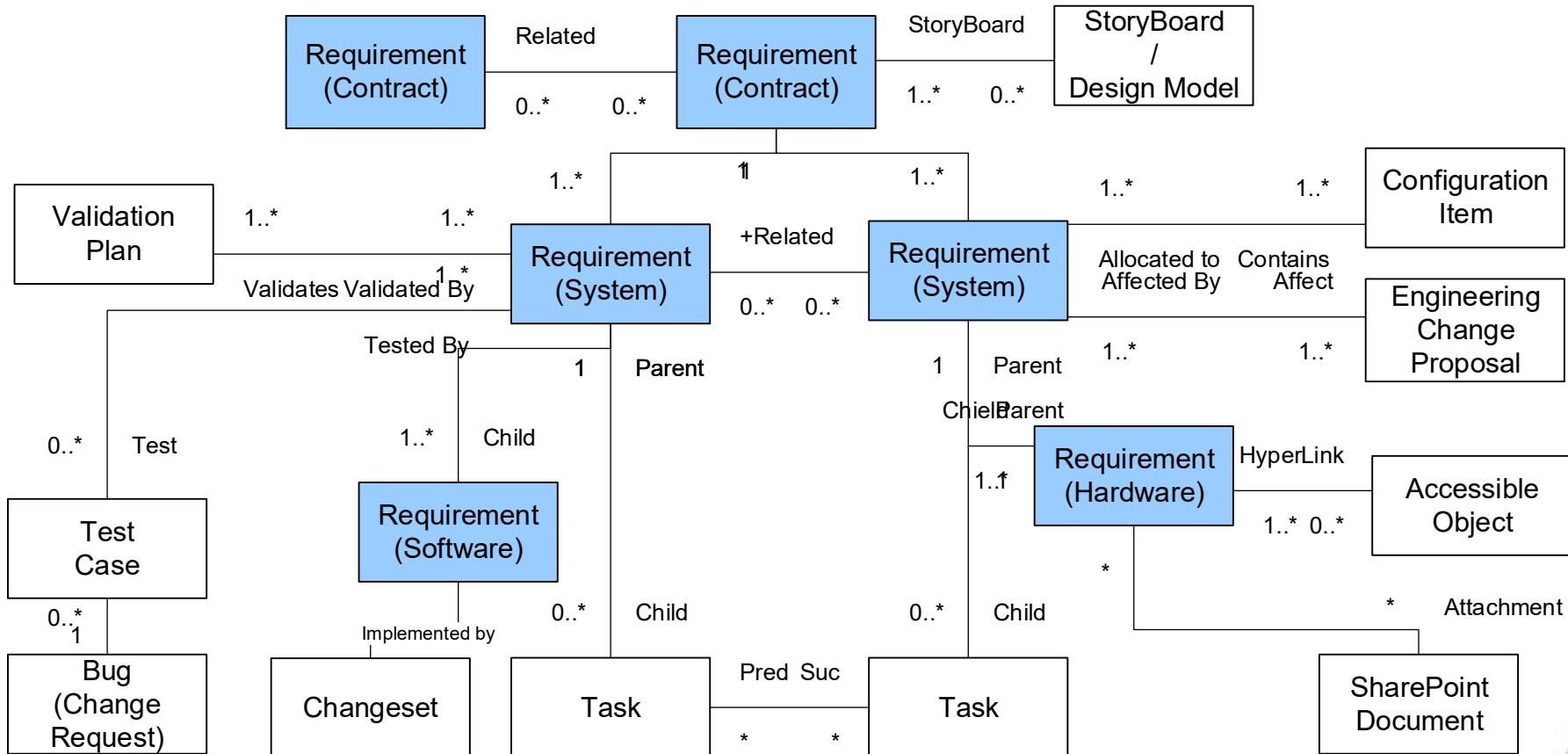
Detail environmental requirements as needed. For hardware based systems, environmental issues can include temperature, shock, humidity, radiation, etc. For software applications, environmental factors can include usage conditions, user environment, resource availability, maintenance issues, error handling and recovery.



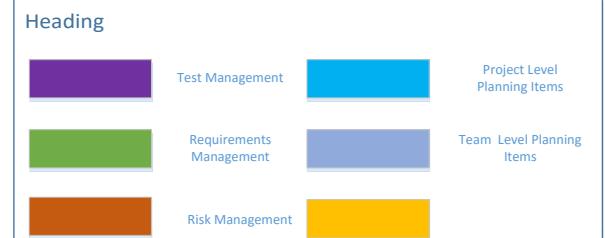
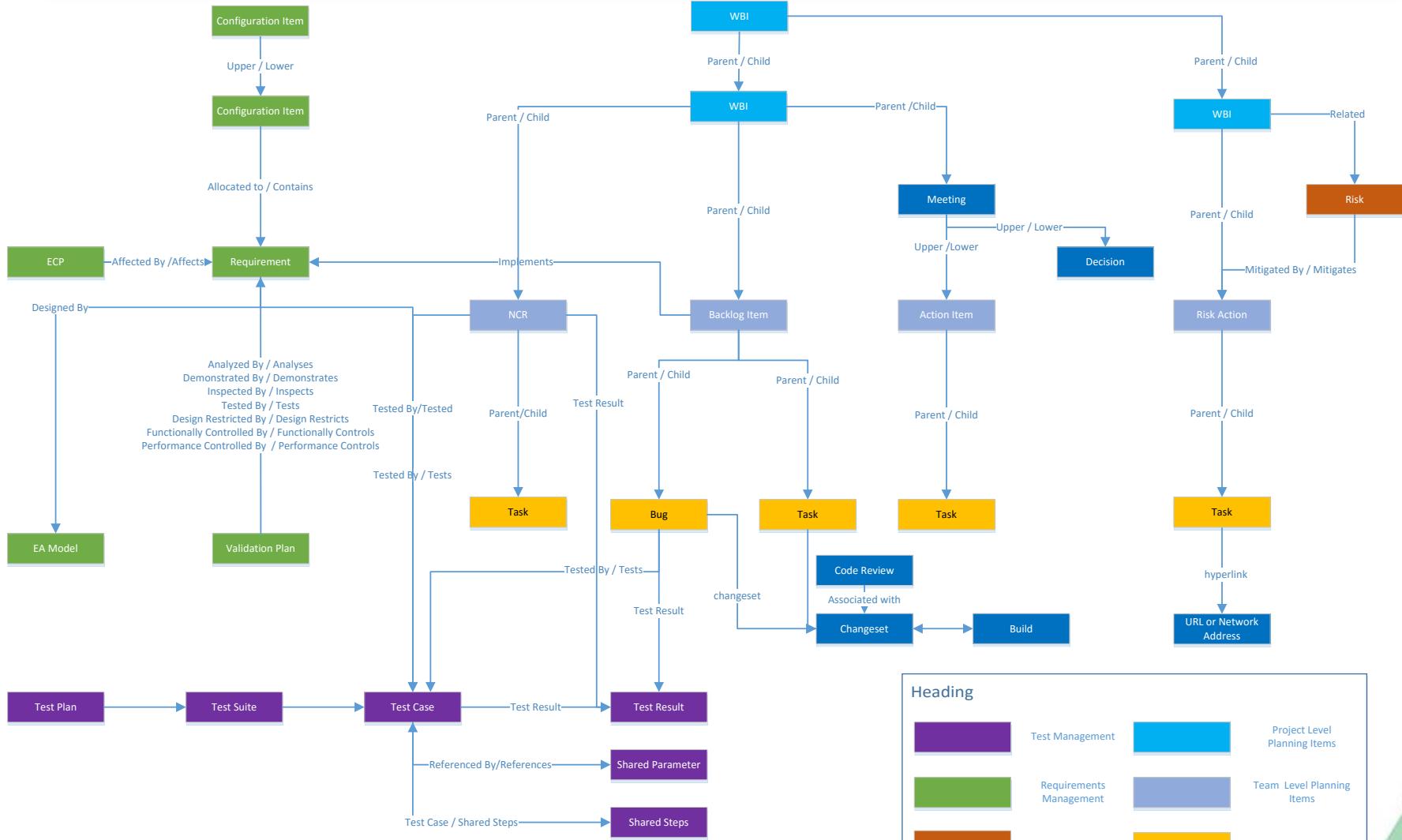
Requirement LifeCycle



Requirements Traceability



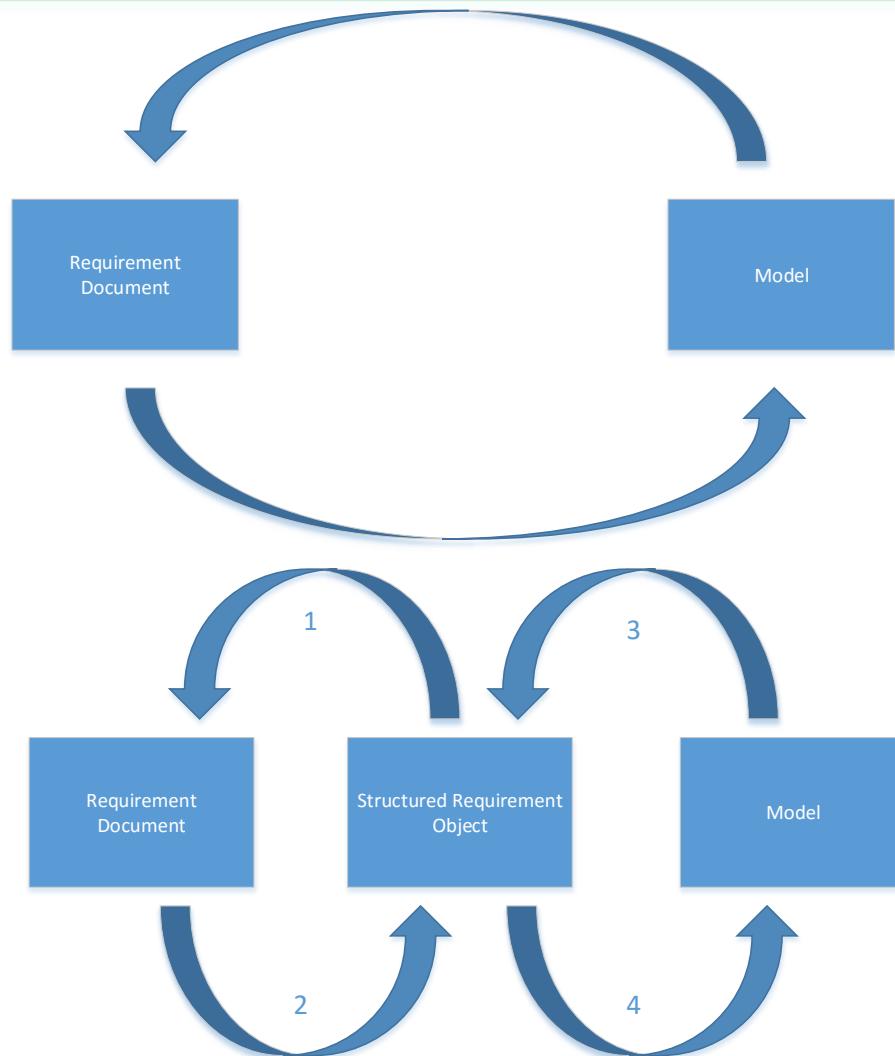
Requirements in ALM



Requirements in ALM

- Traceability with other artifacts is key
 - Requirements to other requirements
 - Customer/System/Software/Hardware..
 - Dependency relation between requirements
 - Requirements to tasks (Project management)
 - Requirements to Test Cases
 - Requirements to Design elements
 - Requirements to generated documents
 - Requirements to source code
 - Requirements to Build
 - Requirements to bugs
 - Requirements to risks
 - ...

HAVELSAN Use case for ModelWriter



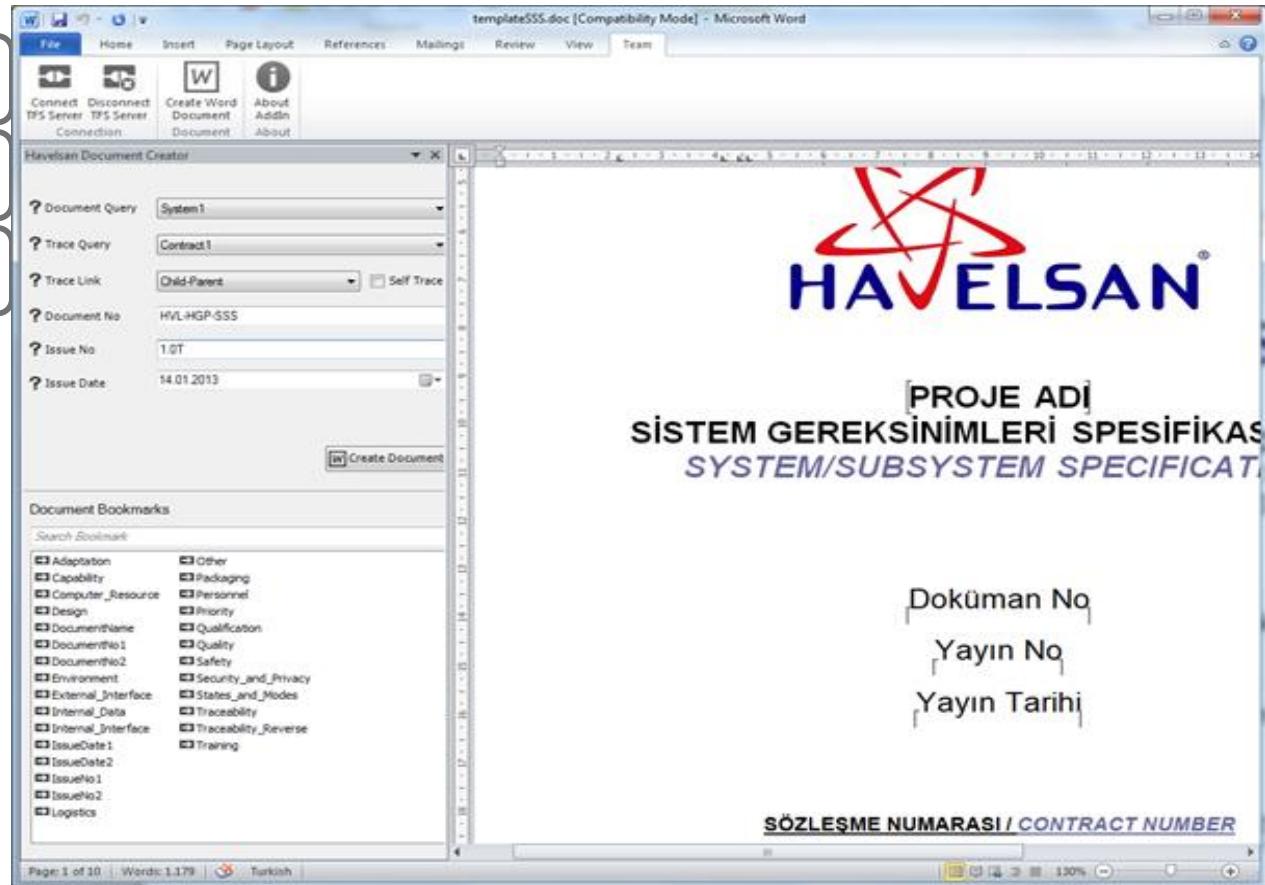
Currently we support scenario #1 and #4, and interested in Scenario #3, #2

Havelsan Ext - Document Generation

Preview

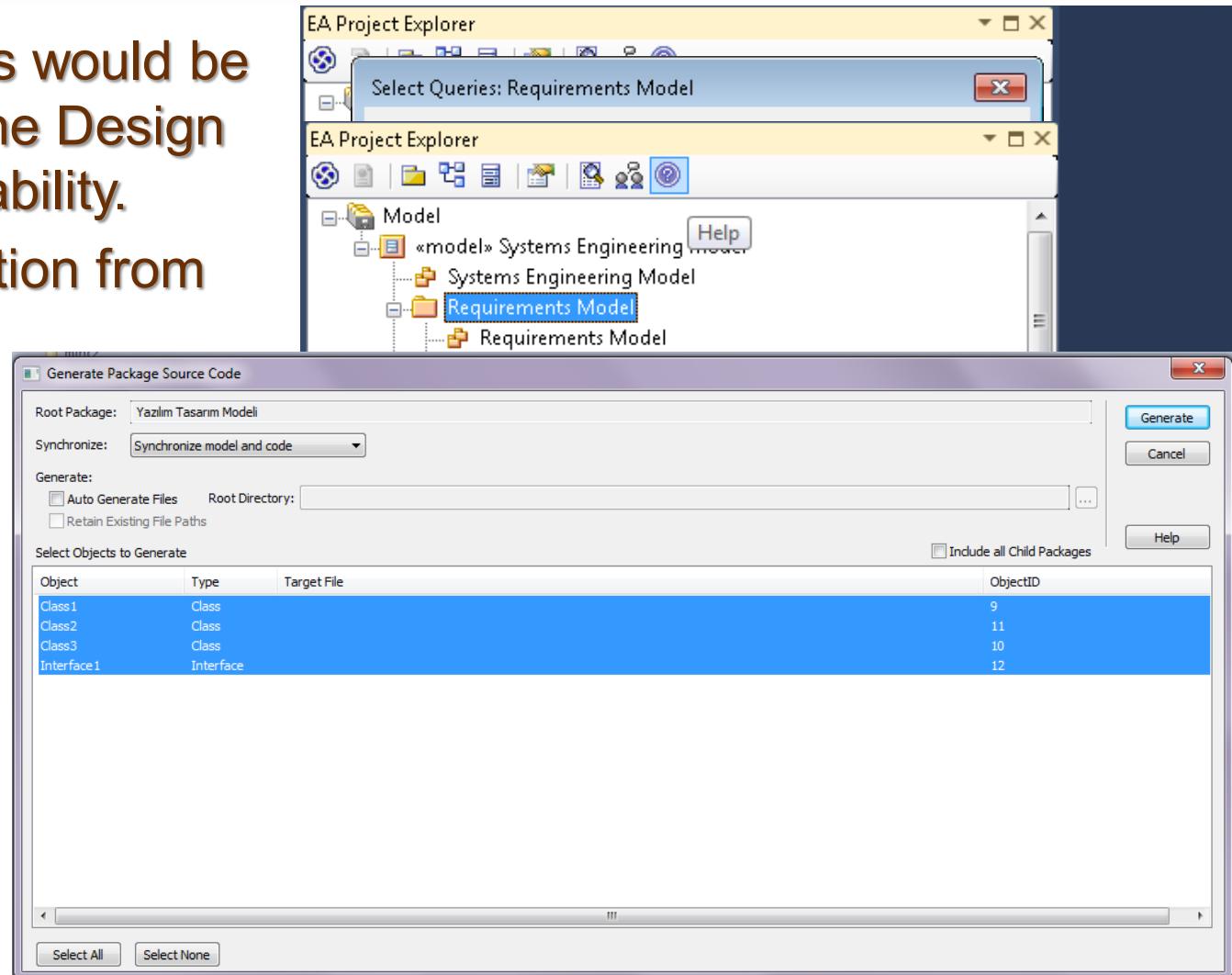
Word Extension

Support for Templates



Requirements - Design Model Traceability

- Requirements would be imported to the Design tool for traceability.
- Code generation from design



UC-TR-05 - Synchronous Business Process Design with Use Cases

Geylani Kardaş

KoçSistem

UC-TR-05 - Synchronous Business Process Design with Use Cases



- Use cases are one of the main approaches to represent the requirements.
 - A use case is a list of actions or event steps, typically defining the interactions between a role (a.k.a actor in the UML).
- BPMN provides a graphical notation for specifying the processes in a diagram based on a flowcharting technique (similar to activity diagrams in UML).
 - The aim is to support business process management, for both technical and business users, by providing a notation which enables to represent complex process semantics.
- However, the transformation and synchronization of use cases and BPMN models are challenges addressed in this UC.

UC-TR-05 - Synchronous Business Process Design with Use Cases



- Applications:
 - University Management System Use cases docs \Leftrightarrow BPMN
 - Eclipse RMF use case specifications \Leftrightarrow BPMN
- At the current state of progress this UC:
 - The transformations are done in one way (left to right)

University Management System Use cases

3 USE CASE UC3: FILE UPLOAD

Primary Actor: Student

Stakeholders and Interests:

- Student: Wants simple user interface, fast response, no system errors.

Preconditions: Student is identified and authenticated.

Success Guarantee (Postcondition): File is uploaded.

Main Success Scenario (or Basic Flow):

1. Student visits file upload page.
2. System opens file browser dialog.
3. Student chooses the file that she/he is wanted to upload.
4. System starts the upload process.

Extensions (or Alternative Flows):

*a. At any time, System fails: To support recovery, ensure all transaction sensitive state and events can be recovered from any step of the scenario.

1. Student restarts System and requests recovery of prior state.
2. System reconstructs prior state.

2a. System detects anomalies preventing recovery:

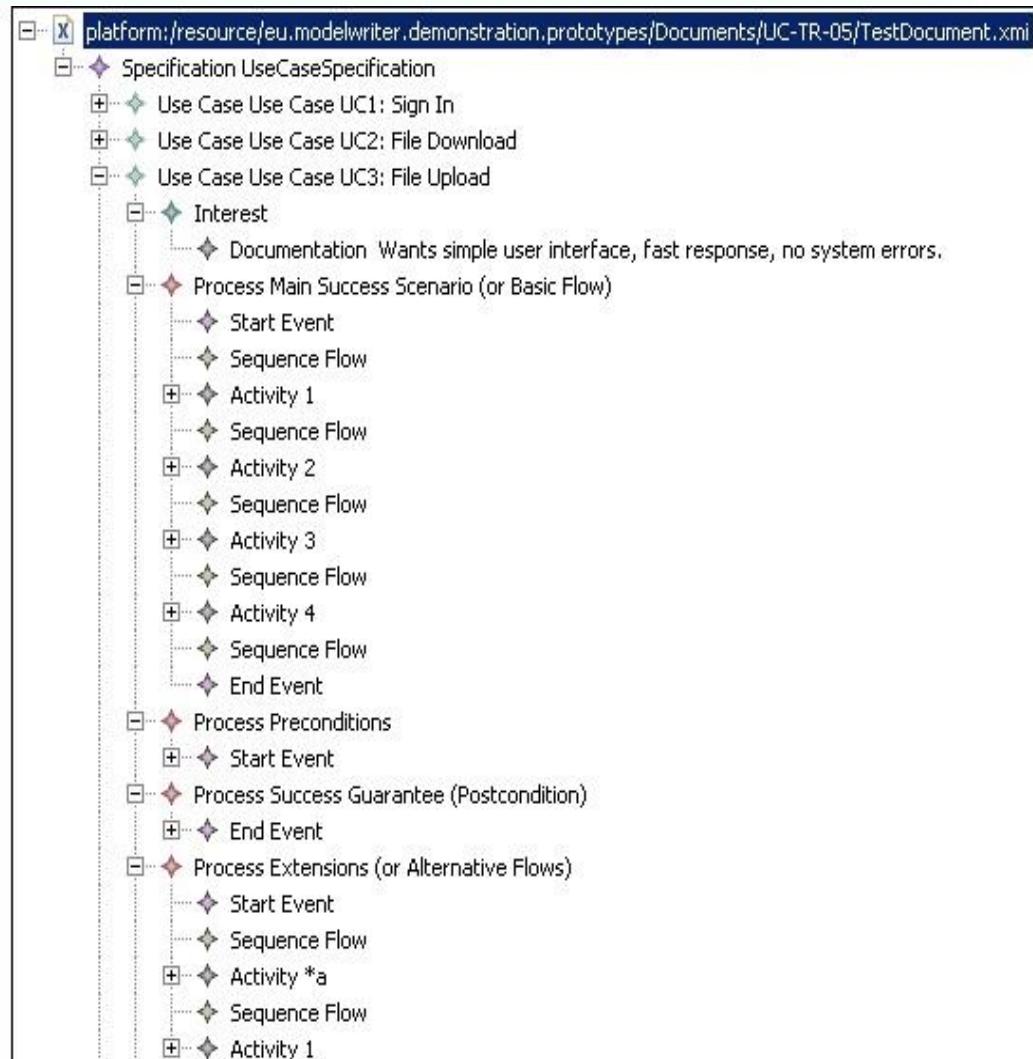
1. System signals error to the Student, records the error, and enters a clean state.

3a. Invalid file:

1. System shows the error and returns the file upload page.

4a. System detects failure to communicate with server:

1. System signals error and rejects the request.



Eclipse RMF use case specifications

1 USE CASE UC1: CREATE A NEW SPECOBJECT

Preconditions:

- ReqIF model exists and is open.

Main Success Scenario (or Basic Flow):

1. We assume that a Specification exists and is open (not required for alternative scenario)
2. Double click on the cell in the Specification Editor to be edited.
3. Select the Child or Sibling submenu.
4. Select the desired Spec Object Type (or none) from the submenu.
5. Note that some cells may not be editable, in which case nothing will happen.

Alternative 1 Create in Outline:

- *a. The same workflow works for elements that are shown underneath "Specifications" in the outline.
 2a. It is also possible to create children of the "SpecObjects" folder in the outline, but in this case, no SpecHierarchy will be created.

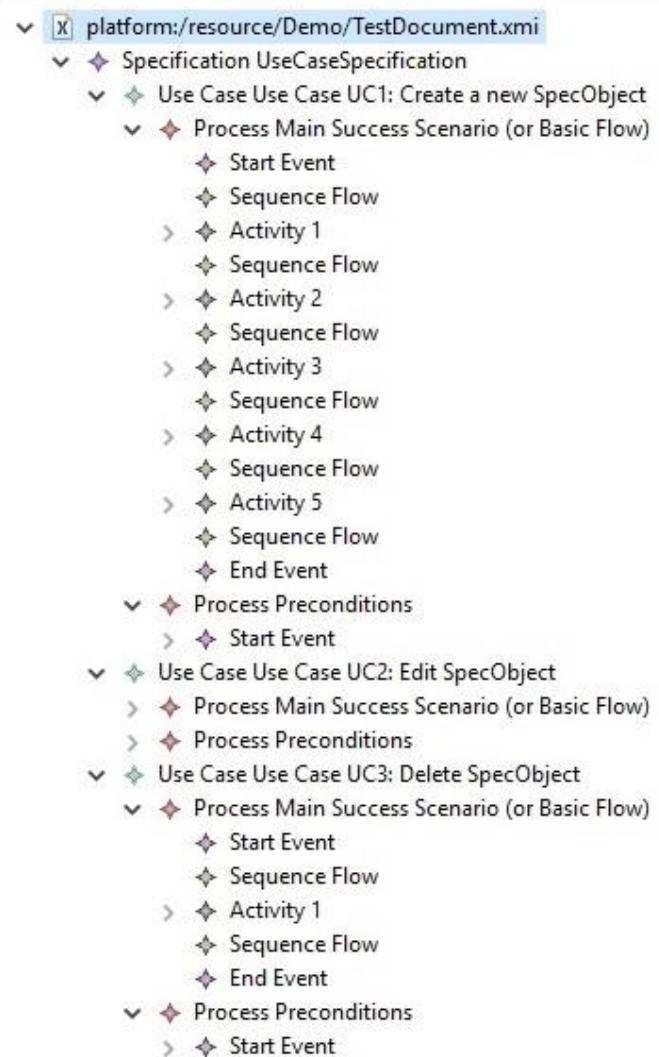
Alternative 2 Keyboard Shortcut:

- *a. The keyboard shortcut Ctrl-Enter will create a SpecHierarchy sibling to the currently selected element and immediately go into edit mode in the currently selected column.

2 USE CASE UC2: EDIT SPECOBJECT

Preconditions:

- A ReqIF model exists, is open and at least one SpecObject exists.



UC-TR-05 - Synchronous Business Process Design with Use Cases



- Demonstration:
 - The initial implemented version of this use case will be presented in the demonstration session.
- Next steps:
 - BPMN model transformation to Use case documents (bi-directional transformation)
 - Synchronization between documents and BPMN Models
 - Using technique documents of Ford-Otosan;

**Thank you for your attention
We value your opinion and
questions.**

4 Progress on Workpackages

Prof. Geylani Kardaş (Moderator)
KoçSistem

WP1 - Industrial use case and requirements

*Anne MONCEAUX
AIRBUS GROUP*

WP1

To describe and define the industrial, real life Use Cases, their associated requirements and evaluation method.

Tasks:

- T1.1 Evaluation Methods & Tools
- T1.2 Industrial Use Cases for Belgium Consortium
- T1.3 Industrial Use Cases for French Consortium
- T1.4 Industrial Use Cases for Turkish Consortium
- T1.5 Consolidated User Requirements and Review
- T1.6 Consolidated Software Requirements and Review
- T1.7 Annual Product Review
- T1.8 Technical Risk Assessment

T.1.1 Evaluation Methods & Tools

- UNIT, KOCSISTEM, AIRBUS, OBEO, HISBIM, MANTIS
 - To define evaluation methods, including the identification of metrics to quantify performance with and without ModelWriter
- Status
 - Survey of available evaluation method and tools
 - D1.1.1 Evaluation Methods & Tools
- Next:
 - Specification of use Cases KPI ; common KPI and selection of evaluation method and tools

T1.3 Industrial Use Cases for French Consortium



- OBEO, AIRBUS
- Status
 - Use case description
 - Discussed at 1st International ModelWriter Workshop in Izmir, Turkey
 - D1.3.1-Industrial Use cases for French consortium
 - Data collection
 - Part of the corpus data is provided by the partners.
 - Detailed description in *D2.1.2 Documentation of the corpora*
 - Public/private status
 - AIRBUS-OBEO-LORIA Non Disclosure Agreement finalized in June 2015
 - as some of the partners have not also decided for the privacy or publicity of the data, all of the corpus data is kept in a private repository in the GitHub
- *Remain to be done*
 - *Make public corpora available for all UC*

T1.4 Industrial Use Cases for Turkish Consortium



- MANTIS + UNIT + KOCSISTEM + HISBIM
- Status
 - Use cases description
 - D1.4.1 Industrial Use Cases for Turkish Consortium
 - Data collection
 - Part of the corpus data is provided by the partners.
 - However, some of the other partners have not decided on their corpus cases.
 - Public / private status
 - as some of the partners have not also decided for the privacy or publicity of the data, all of the corpus data is kept in a private repository in the GitHub

T1.5 Consolidated User Requirements and Review



- AIRBUS, OBEO, MANTIS, UNIT, KOCSISTEM, ALL
 - To share a common vision of User needs and expectations
- Status
 - UC driven User requirements capture
 - Consolidation through collective review (2nd International ModelWriter Workshop in Brussels, Belgium)
 - User requirements are stored & managed in GitHub
 - D1.5.1 Minutes of the User Requirements Review meeting
 - D1.5.2 User Requirements Document (URD) was automatically generate from Github
 - Technical Risks based on the defined requirements are identified in D1.8.1 Technical Risk Assessment v1.0

- AIRBUS, LORIA, UNIT, MANTIS, OBEO, KOCSISTEM, ALL
 - The [minimum] objective of the first year (Y1) is to integrate the key pieces of software together (modelling tools with a word processor within an IDE) to prove that we can have a unified prototype ModelWriter platform.
- Status
 - Technical partners refined software requirements based on URD
 - Consolidation through collective review (2nd International ModelWriter Workshop in Brussels, Belgium)
 - Software requirements are stored & managed in GitHub
 - D1.6.1 Minutes of the Software Requirements Review meeting
 - D1.6.2 Software Requirements Document (SRD) was automatically generate from Github
 - See also D1.8.1 Technical Risk Assessment v1.0

T1.8 Technical Risk Assessment

- OBEO, UNIT, KOCSISTEM + ALL
 - To identify, monitor and mitigate risks on the achievement of the project
- Status
 - 1st evaluation using Actuarial Approach of Technical Risk Assessment (TRA) of risks linked to requirements (URD, SDR) and technologies.
 - D1.8.1 Technical Risk Assessment v1.0
- Next
 - *The document may be up-dated throughout the project with special review at the same time as for the software requirements and the architectural design review, depending on the further details and requirements we get from the industrial use case providers.*

WP2 - Semantic Parsing and Generation of Documents and Documents Components

Claire GARDENT, Mariem MAHFOUDH

CNRS / LORIA

Samuel CRUZ-LARA

University of Lorraine / LORIA

WP2



Goal: Provide tools and methods for:

- Annotating text fragments with model elements
- Converting texts to models and models to text

Tasks:

- T2.1 Data Collection
- T2.2 Semantic Parsing
- T2.3 Natural Language Generation
- T2.4 Definition of a common target semantic language
- T2.5 Development of a Semantic Parser and of a Natural Language Generator

T2.1 Data Collection

- AIRBUS (Confidential Data)
 - Text
 - System Installation Design Principles (SIDP) Documents
 - 986 semi-structured SIDP rules
 - Models
 - The Rule ontology represents the SIDP rules concepts. An OWL ontology composed of 30 classes, 35 object properties and 54 data properties.
 - The Component ontology represents the concepts and the vocabulary used in system installation rules. It is an OWL-DL ontology and it is composed in its current version of 476 classes, 21 ObjectProperties and 35 DataProperties.



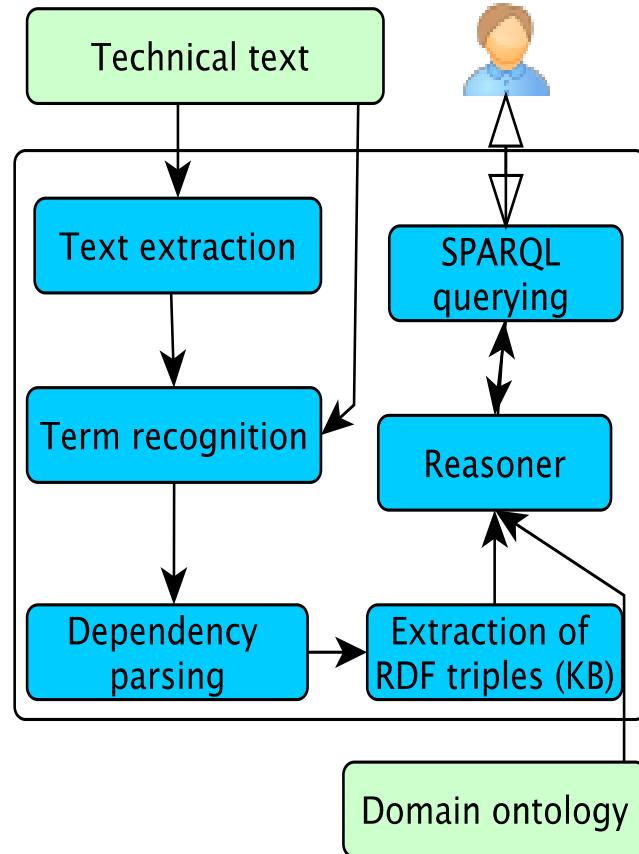
Non Disclosure Agreement finalised in June 2015.

T2.1 Data Collection

- OBEO
 - Text
 - "TxStyle" Files: a set of files in natural language (i.e., English) related to the documentation of the application being modelled by Sirius
 - Models
 - Java Concepts: a list of Java identifiers (i.e., classes, interfaces, methods, etc.) related to Sirius
 - Ecore Concepts: a list of concepts related to Ecore (the Eclipse Modeling Framework meta model) and to Sirius



T2.2 Semantic Parsing



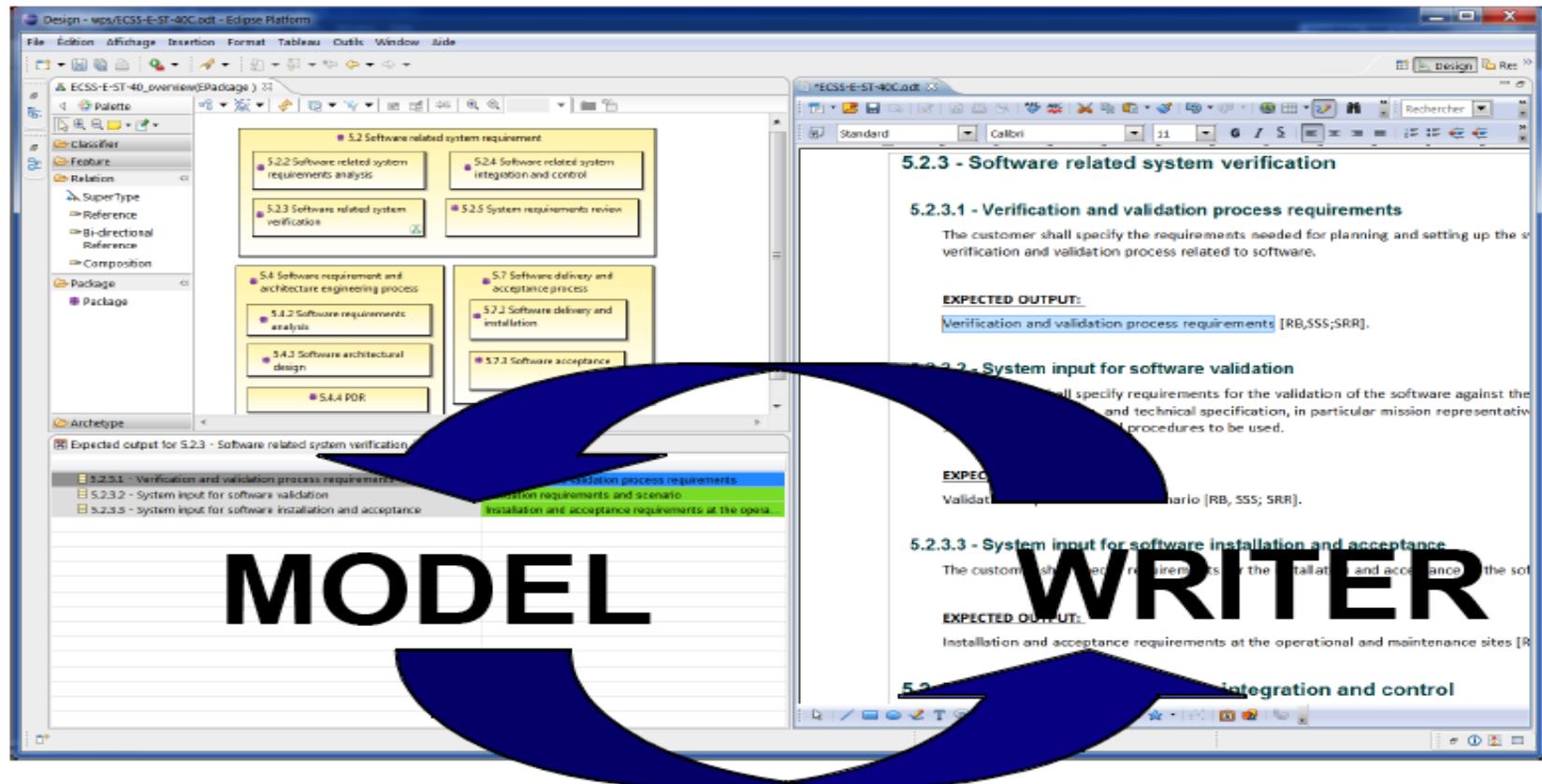
- Developed a prototype illustrating the automatic construction of an RDFS KB from text (CNRS/LORIA)
- « *Parsing Text into RDF* »
B. Batouche, C. Gardent and A. Monceaux. SEPLN 2015, Alicante, Spain.
- Full scale Implementation applied to AIRBUS SIDP rules (Airbus)

T2.3 Natural Language Generation

- D2.2.1 Report: Overview and Comparison of Existing Generators
- Keynote at SEPLN 2015, Alicante, Spain
- 2 Internships on **generation from RDF data** (ongoing)
 - Lexicalisation: automatic acquisition of a lexicon mapping RDF properties to natural language expressions
 - Document planning: automatic detection of typical document structures using DBpedia and Wikipedia

Semantic Annotation: Creating and Maintaining Synchronization Links, Checking their Consistency





Creating synchronization links (relating text and model)
Checking the consistency of created synchronization links
Maintaining synchronization links (create, search, delete, modify)

- Automatically

- Exact matching: identified using String matching
 - Ex: Attach (text element) IsSameAs http://airbus-group/opp-function#Attach (ontology concept)
- Morphological matching: identified using lemmatization and Stanford CoreNLP tools
 - Ex: Attached IsMorphologicallySimilarTo http://airbus-group/opp-function#Attach
- Semantic matching: identified based on ontology and SKOS labels
 - Ex: Fixation isSynonymTo http://airbus-group.installsys/component#AttachmentPoint

- Manually

- UserLink: Created by the user

Checking the Consistency of Synchronization Links



- Consistency check based on ontology's axioms and properties
 - Rule:
 - If a text element and an ontology concept are semantically disjoint, then they cannot be synchronized
 - Ex: rigid Component **cannot be synchronized with** <http://airbus-group.installsys/component#FlexibleComponent>

- Link maintenance operations:
 - Add New Link (user given)
 - Search Link (create or retrieve)
 - Remove Link
- Synchronisation between text and links
 - RenameTextElement
 - Add TextElement
 - RemoveTextElement

Semantic Annotation, Links Synchronization and Consistency Check

ModelWriter Project

File Link Change Statistic

The Text The Model Plain Tree



Application to Airbus Industrial Case

Semantic Annotation, Links Synchronization and Consistency Check



- Semantic parsing and consistency check:
 - The prototype can be accessed on the GitHub Model Writer repository:
 - <https://github.com/ModelWriter/WP2/tree/master/Tool>

Semantic Annotation



Semantic Annotation

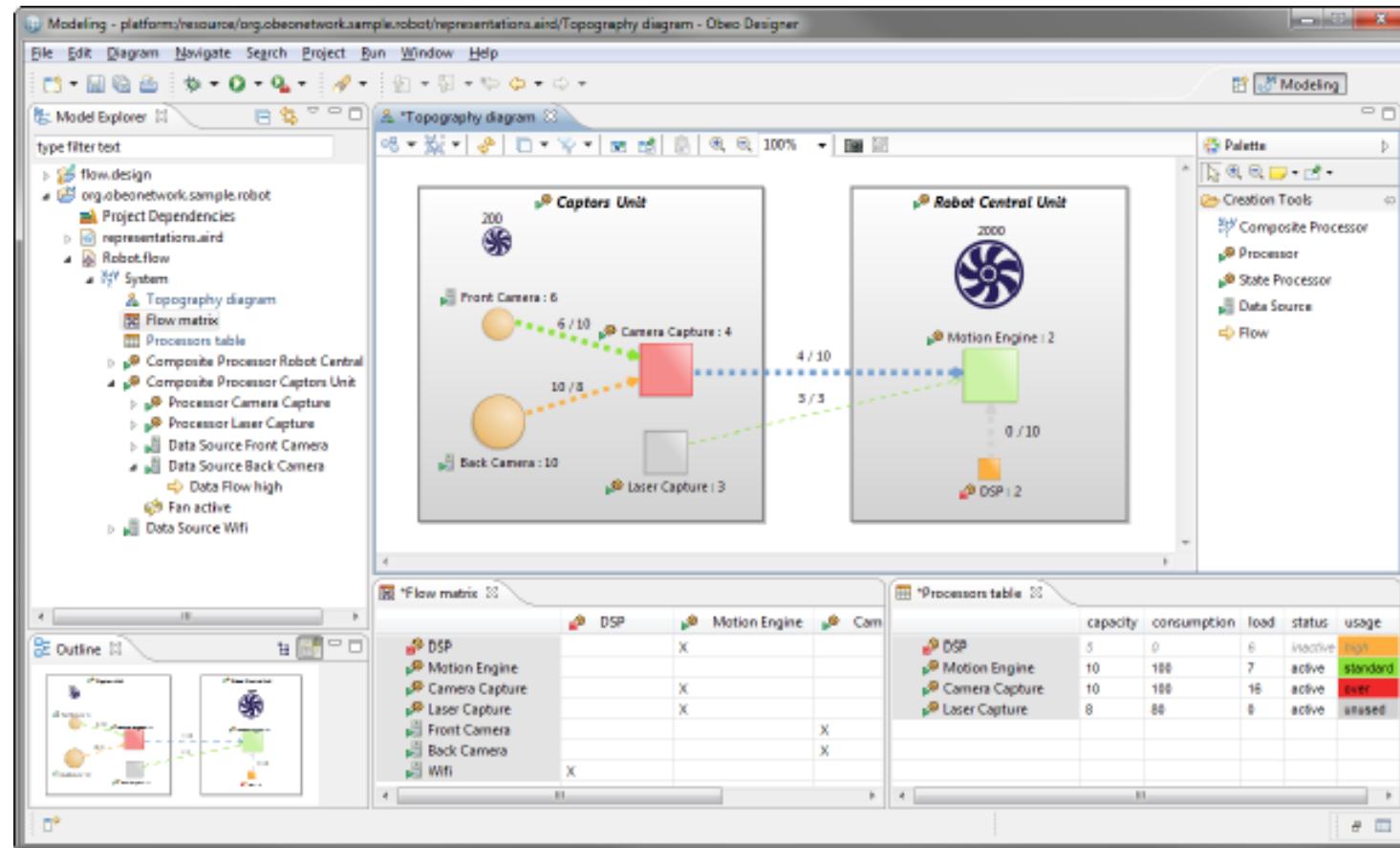
- OBEO Corpora
 - EMF (Eclipse Modeling Framework)
 - The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model
 - From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor
 - SIRIUS
 - Is an Eclipse project based on EMF

Semantic Annotation

- SIRIUS
 - A modeling workbench created with Sirius is composed of a set of Eclipse editors (diagrams, tables and trees) that allow the users to create, edit and visualize EMF models
 - The editors are defined by a model that defines the complete structure of the modeling workbench, its behavior and all the edition and navigation tools
 - For supporting specific need for customization, Sirius is extensible in many ways, notably by providing new kinds of representations, new query languages and by being able to call Java code to interact with Eclipse or any other system

Semantic Annotation

SIRIUS

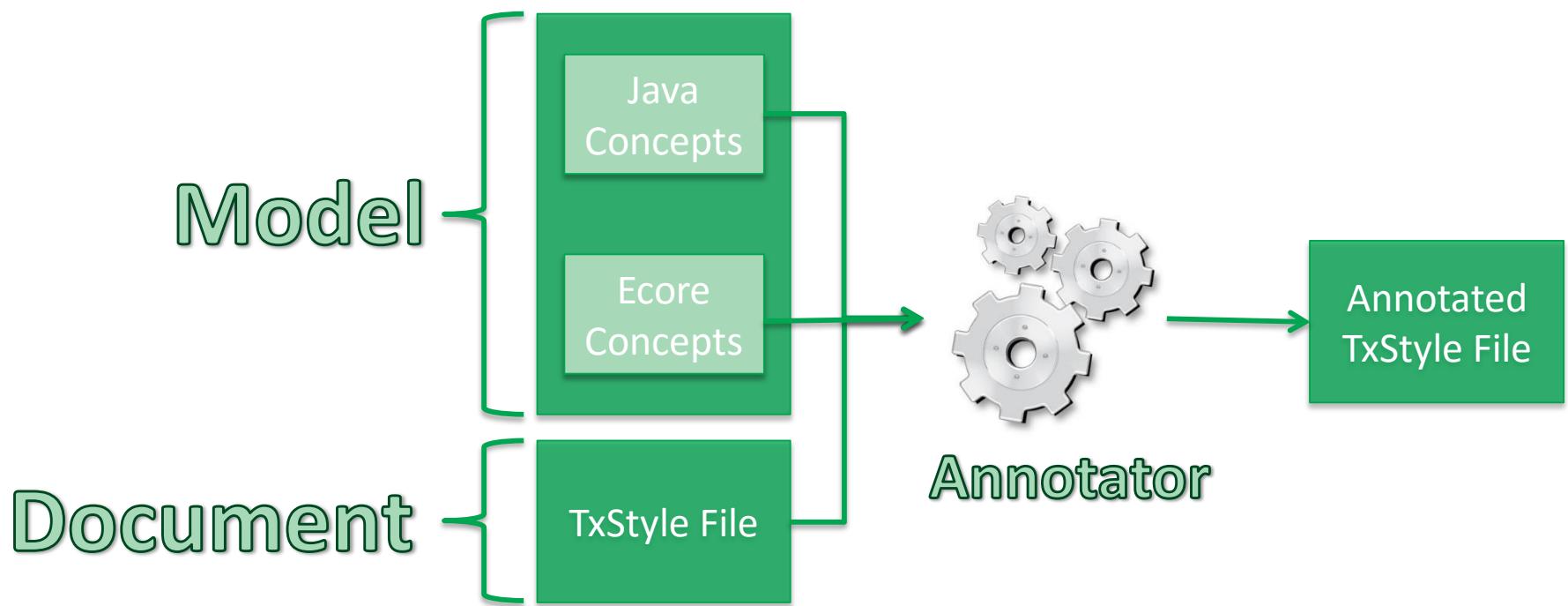


Semantic Annotation

- OBEO Corpora
 - **Java Concepts**: a list of Java identifiers (i.e., classes, interfaces, methods, etc.) related to Sirius
 - **Ecore Concepts**: a list of concepts related to Ecore (the EMF meta model) and to Sirius
 - **“TxStyle” Files**: a set of files in natural language (i.e., English) related to the documentation of the application being modeled by Sirius

Semantic Annotation

- A semantic annotator
 - We have developed is a basic prototype allowing to annotate the “TxStyle” files by establishing links to Java Concepts and to Ecore Concepts



Semantic Annotation

- A semantic annotator
 - The prototype can be accessed on the GitHub Model Writer repository:
 - <https://github.com/ModelWriter/WP6/tree/master/EcoreConcepts-JavaConcepts-Annotator>

WP3 - Model to/from Knowledge Base Synchronization Mechanism

*Moharram Challenger, R&D Director
UNIT Information Technologies R&D*

WP3

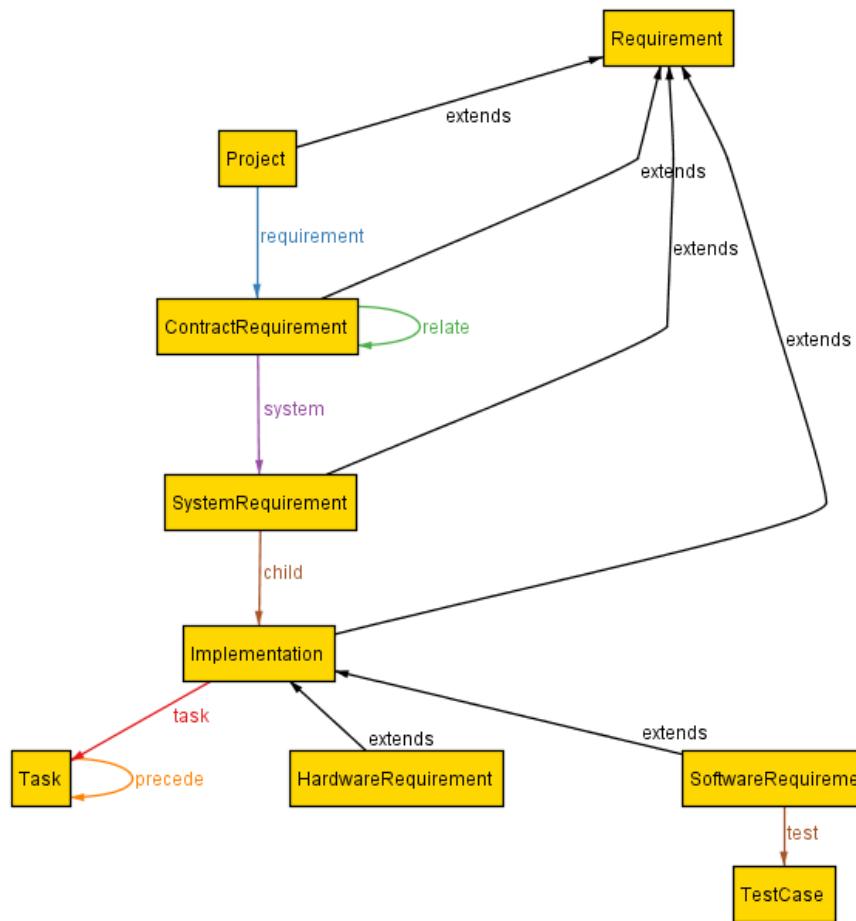
- Objective: provide the **synchronization mechanism** to keep the “**user-visible models**” consistent with the “**KB-stored models**”
- This will consist of the following main **plug-in** components:
 - **Transformation Manager**: provides the infrastructure to register and launch transformations.
 - **Configuration Manager**: for personalizing the behaviour of the framework to meet the needs of a specific standard / organization / project / individual.
 - **Traceability Manager**: keeps links between elements of user-visible models and elements of the KB.
 - **Synchronization Manager**: triggering transformations when synchronization is needed.

Tasks

- T3.1 - Review of M2M transformation approaches
- T3.2 - Specification and design of the M2M Transformation Framework
- T3.3 – Development of the **Transformation Manager** component
- T3.4 – Development of the **Configuration Manager** component
- T3.5 – Development of the **Traceability Manager** component
- T3.6 – Development of the **Synchronization Manager** component
- T3.7 – Design of the model-to-model transformations
- T3.8 – Implementation of the model-to-model transformations
- T3.9 – Validation of the M2M Transformation Framework

Configuration: Havelsan example

extends: 6
 child: 1
 precede: 1
 relate: 1
 requirement: 1
 system: 1
 task: 1
 test: 1



```

module Havelsan/Requirement

abstract sig Requirement {}

sig Task {
  precede: lone Task,
  { all t: Task | one t.^precede }
}

one sig Project extends Requirement {
  requirement: some ContractRequirement
}

sig ContractRequirement extends Requirement {
  system: set SystemRequirement,
  relate: set ContractRequirement
}{all c: ContractRequirement | one c.^~requirement}

--@name: "System Requirement"
sig SystemRequirement extends Requirement {
  child: some Implementation
}{all s: SystemRequirement | one s.^~system}

abstract sig Implementation extends Requirement {
  task: set Task
}{all i: Implementation | one i.^~child}

--@context.editor: "ReqIFEditor"
sig SoftwareRequirement extends Implementation {
  test: some TestCase
}

sig HardwareRequirement extends Implementation {}

sig TestCase {}{ all t: TestCase | one t.^~test}

fact noSelfRelation{
  no c: ContractRequirement | c in c.relate
  no t: Task | t in t.^precede
}

fact noCycles{no t: Task | t in t.^precede}

fact realismConstraint {
  some ContractRequirement
  some HardwareRequirement
  some SoftwareRequirement
  some precede
}
  
```

A Formal Specification Model to configure the ModelWriter

Traceability: Havelsan example

The screenshot shows a software interface for traceability and specification management. On the left, a 'Traceability Virtualization' window displays a hierarchical requirement structure. At the top is a 'Project' node, which branches into three 'ContractRequirement' nodes: 'ContractRequirement2', 'ContractRequirement0', and 'ContractRequirement1'. 'ContractRequirement1' has a green arrow labeled 'system' pointing to a 'SystemRequirement' node. This 'SystemRequirement' node has three child nodes: 'SoftwareRequirement2', 'SoftwareRequirement0', and 'SoftwareRequirement1'. 'SoftwareRequirement1' has a blue arrow labeled 'task' pointing to a 'Task1' node, which in turn has a red arrow labeled 'precede' pointing to a 'Task0' node. A status bar at the bottom left indicates: 'child: 3', 'precede: 1', 'requirement: 3', 'system: 1', and 'task: 1'.

Customer Requirements Specification.md

- # Customer Requirements Specification
- ## UC-1 Create a new SpecObject

Note that the Specification Editor is the main interface for users. Therefore, creating SpecObjects in this editor is the main success scenario.

- ### Precondition
- ReqIF model exists and is open.
- ### Main Success Scenario
- 1. We assume that a Specification exists and is open (not required for alternative scenario)
- 2. Open a row's context menu (or in the empty editor space)
- 3. Select the Child or Sibling submenu.
- 4. Select the desired Spec Object Type (or none) from the submenu.
- 5. This results in a new SpecHierarchy being created that is linked to a newly created SpecObject with the correct type.

Alternative 1: Create in Outline

ModelWriter Source Mapping View

ID	Text
eff099c7-9ad5-4b4d-ac4d-5253d0594...	Customer Requirement

Markers ModelWriter Master View

ID	Text
	SpecObject [T: Task]

Problems ModelWriter Target Mapping View

ID	Text

Properties

A Formal Specification Model to configure the ModelWriter

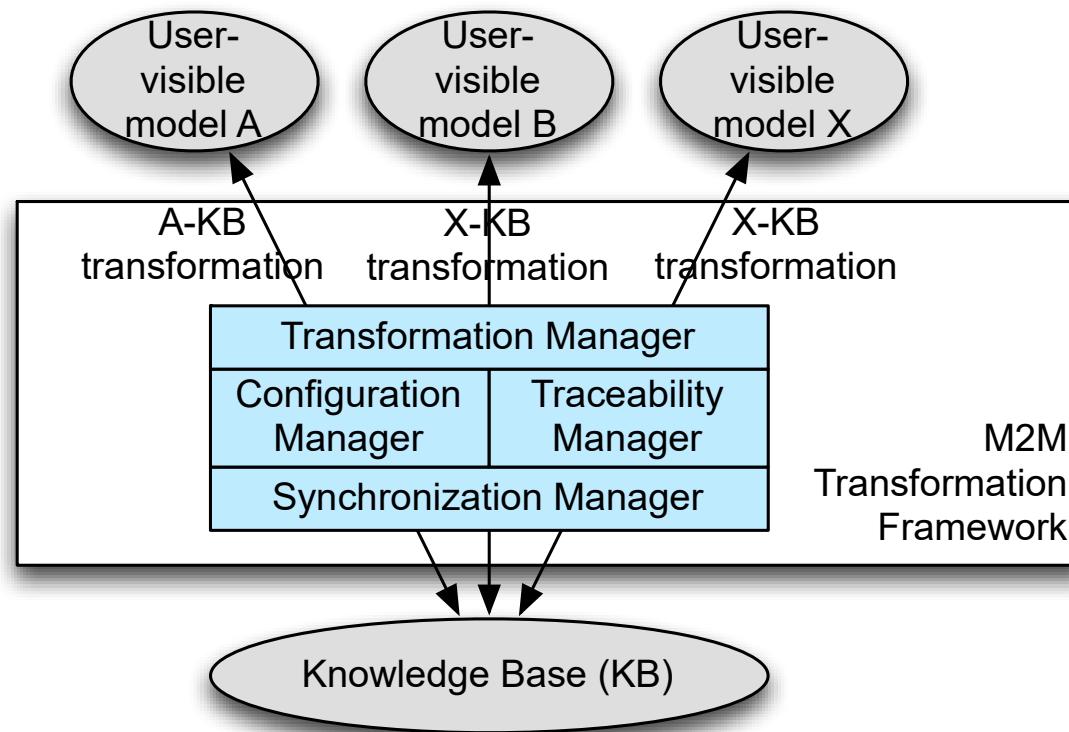
T3.1 - Review of M2M transformation approaches

- UNIT, KOCSISTEM
 - A systematic review of model-to-model transformation approaches, and a selection of the most convenient and widely used in the industry for inclusion into the ModelWriter tool
- Status
 - Survey of available approaches and tools are available at:
 - D3.1.1Review of model-to-model transformation approaches and technologies
- Next:
 - The document may be updated based on the new approaches and tools in SotA

- UNIT + KOCSISTEM
 - Objective: Designing the M2M Transformation Framework whose main goal is to make the ModelWriter tool able to launch M2M transformations
- Status:
 - D3.2.1 - M2M Transformation Framework architectural design document (incl. Transformation, Configuration, Traceability, and Synchronization architectural design)
- Next:
 - The architecture may be updated based on the new needs during the project progress.

T3.2 - Specification and design of the M2M Transformation Framework

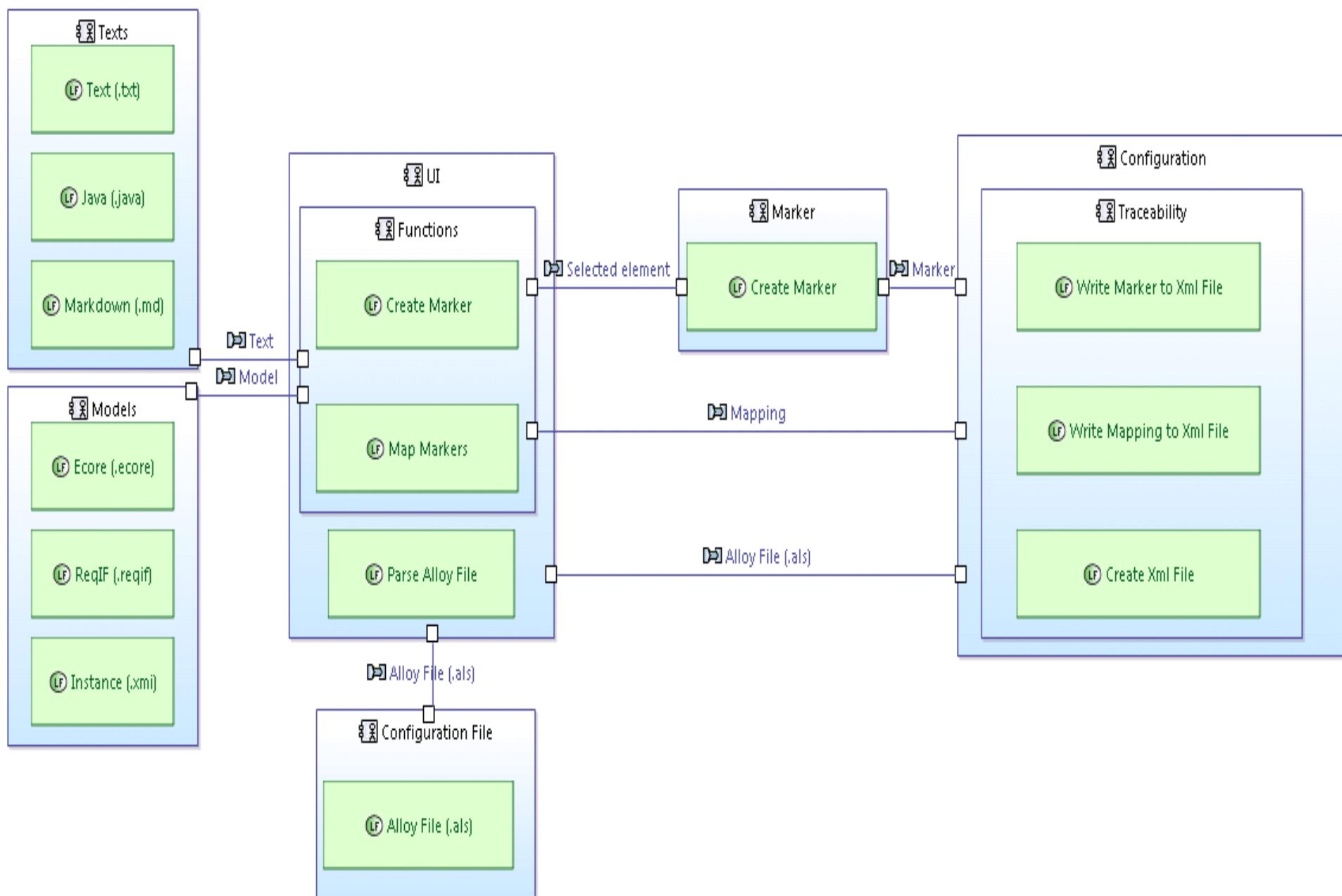
- Overview of the components of the M2M Transformation Framework:



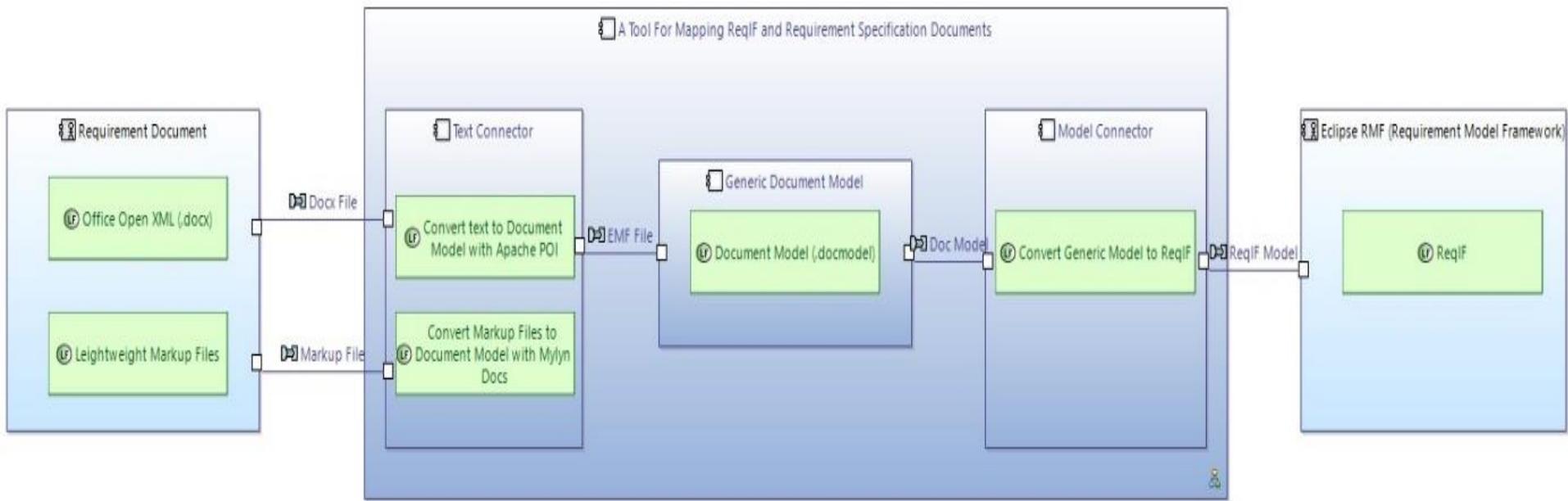
T3.3, T3.4, T3.5, T3.6 (UNIT + KOCSISTEM)

- Development of the:
 - T3.3 Transformation Manager component
 - T3.4 Configuration Manager component
 - T3.5 Traceability Manager component
 - T3.6 Synchronization Manager component
- Status:
 - These tasks has software deliverables which are developed and are available at GitHub
- Next:
 - These components will be updated.

T3.3, T3.4, T3.5, T3.6 (UNIT + KOCSISTEM)



T3.3, T3.4, T3.5, T3.6 (UNIT + KOCSISTEM)



T3.3, T3.4, T3.5, T3.6 (UNIT + KOCSISTEM)

- The fully functional demonstration of the main components of WP3 (T3.3, T3.4, T3.5, T3.6) will be presented at demonstration session.

WP4 – Knowledge base Design and Implementation

Prof. Dr. Erhan Mengusoglu
MANTIS

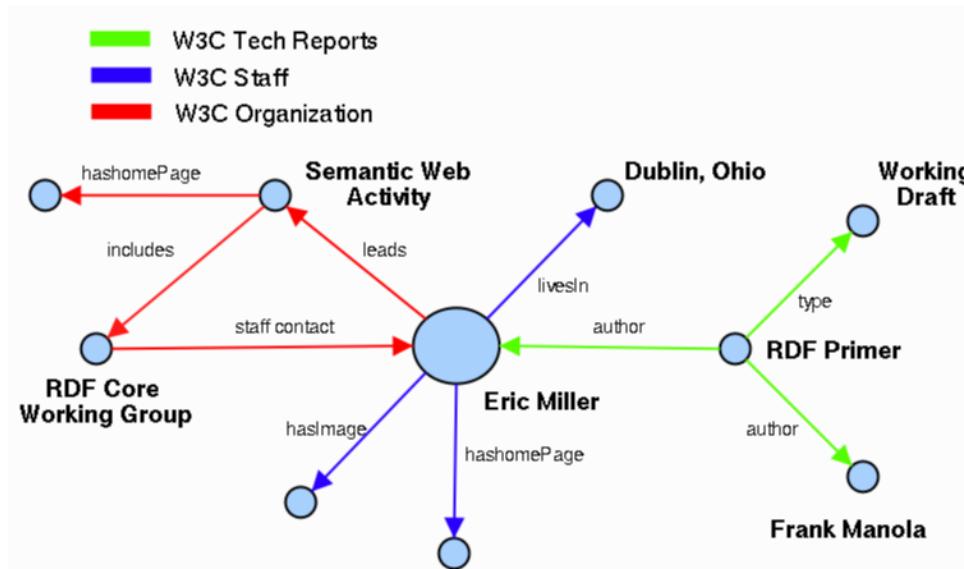
- Design and implement the ModelWriter's federated Knowledge Base itself, hosting multiple formalisms.
- Design and implement its bi-directional text-model synchronization mechanism.
- Design and implement its API.
- Design and implement a set of specialised modules (plug-ins) that exploit the Knowledge Base in ways that make the tasks of Technical Authors much more productive, e.g. consistency checks.
- Design and implement the collaborative functions linking and hierarchically organizing multiple ModelWriter KBs used by different Technical Authors on different sites.

- Plug-in #1 – This provides consistency and completeness checks within the same software lifecycle document, allowing automatic quality review of the content (meaning).
- Plug-in #2 – This provides consistency and completeness checks between related set of documents.
- Plug-in #3 – This provides semantic comparison between two versions of the same software lifecycle document (i.e. what conceptual changes have happened).

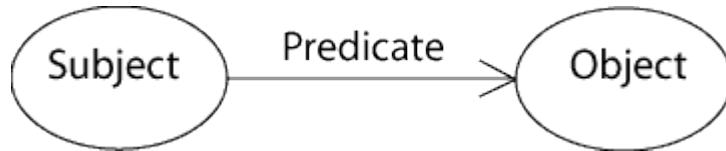
WP4 Knowledge Base Design and Implementation

- T4.1 - Design of the Knowledge Base (MANTIS + OBEO + KUL2 + UNIT + KOCSISTEM)
- T4.2 - API of the Knowledge Base (KOCSISTEM + KUL2 + + OBEO + UNIT + HISBIM)
- T4.3 - Implementation of the Knowledge Base (KUL2 + MANTIS + HISBIM)
- T4.4 – Plug-in #1: ModelWriter-assisted requirements review (KUL2 + MANTIS)
- T4.5 – Knowledge Base serialization and reuse plug-in (MANTIS)
- T4.6 – Plug-in #3: ModelWriter-assisted semantic comparison of 2 documents (OBEO + MANTIS + HISBIM)
- T4.7 – Plug-in #2: ModelWriter-assisted compliance review (MANTIS + UNIT + AIRBUS + SOGETI)
- T4.8 – Internal bi-directional synchronization mechanism (OBEO + UNIT)
- T4.9 – External synchronization mechanism for collaborating ModelWriters (HISBIM)

WP4 - T4.1 – Design of the Knowledge Base



Semantic Web



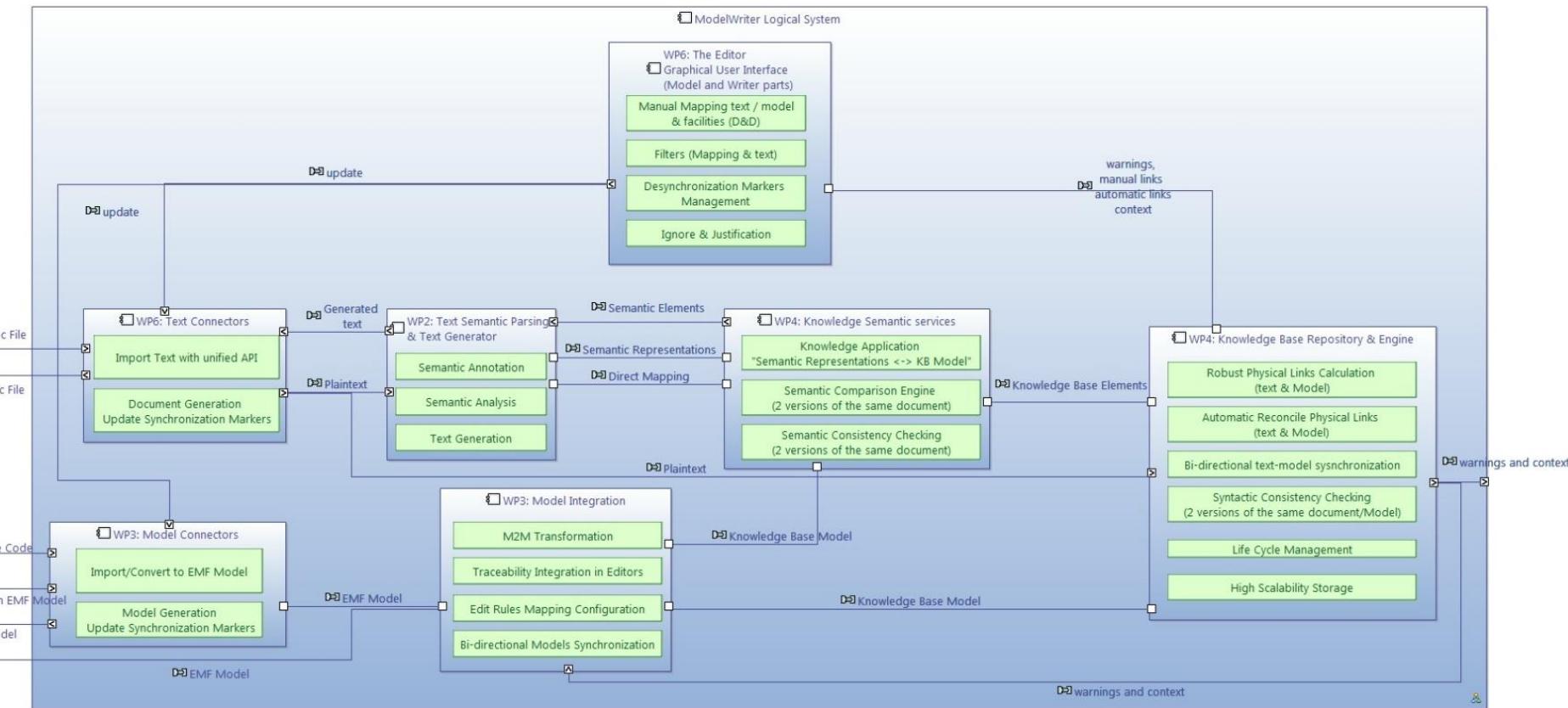
RDF Expression

WP5 – Project Management

*Moharram Challenger, R&D Director
UNIT Information Technologies Ltd.*

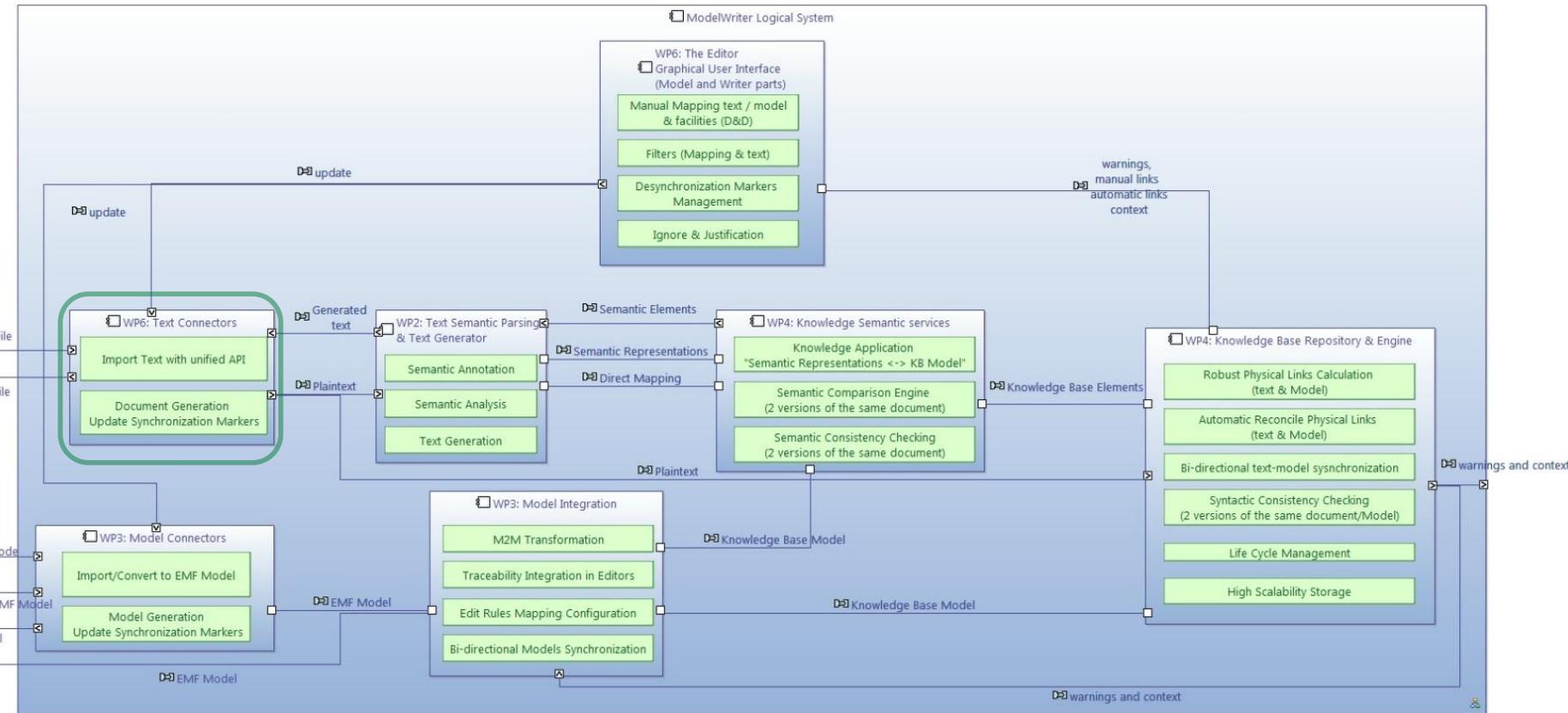
WP6 – Architecture, Integration and Evaluation

Yvan lussaud
OBEO



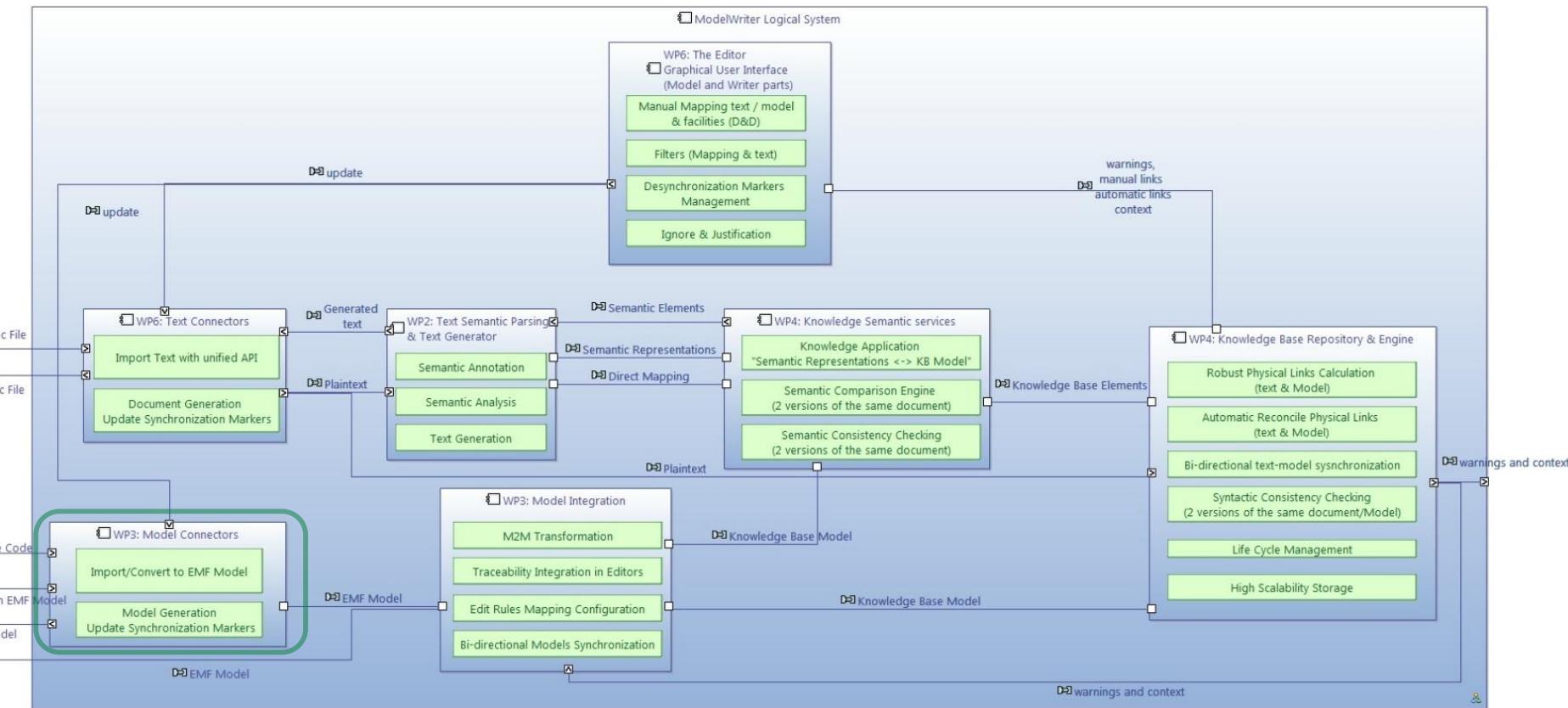
WP6

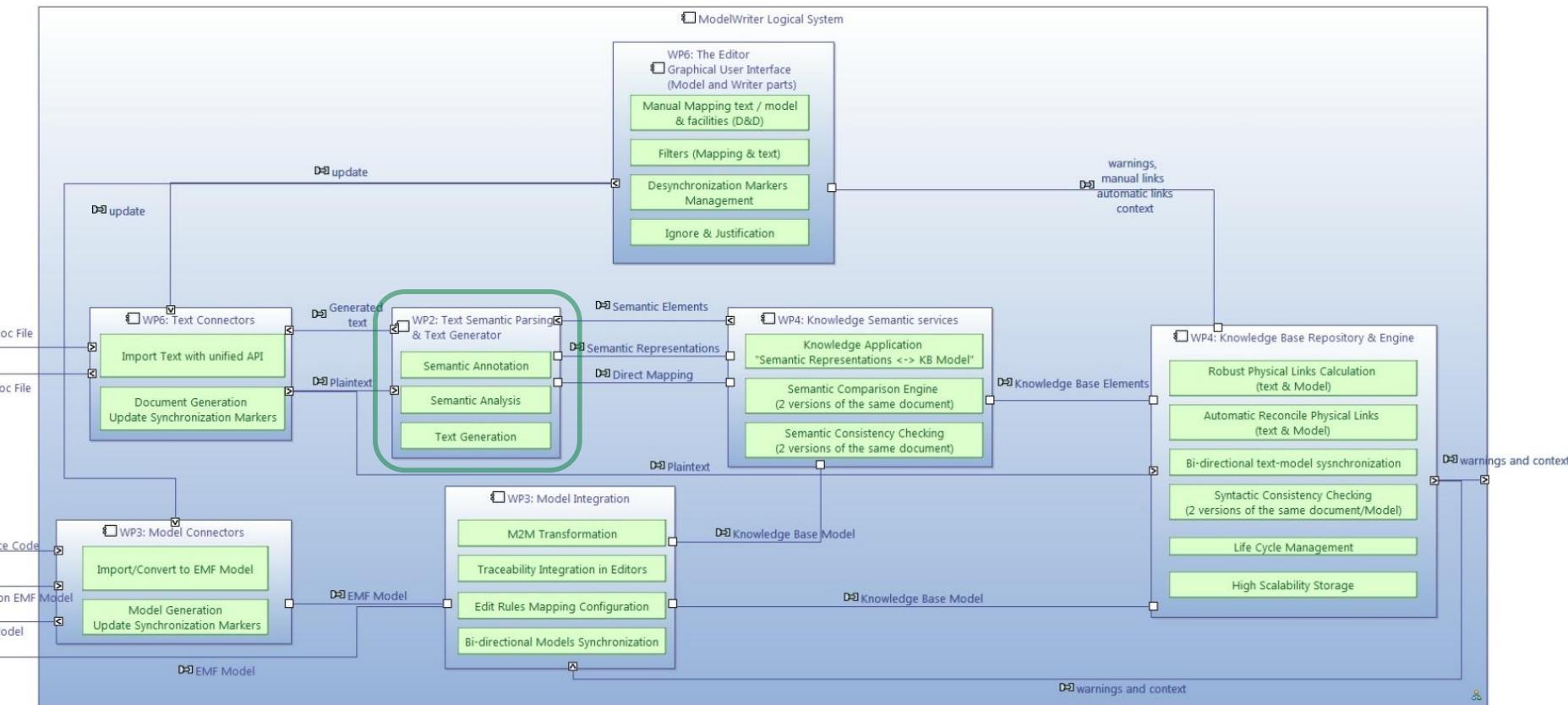
Text connectors



WP6

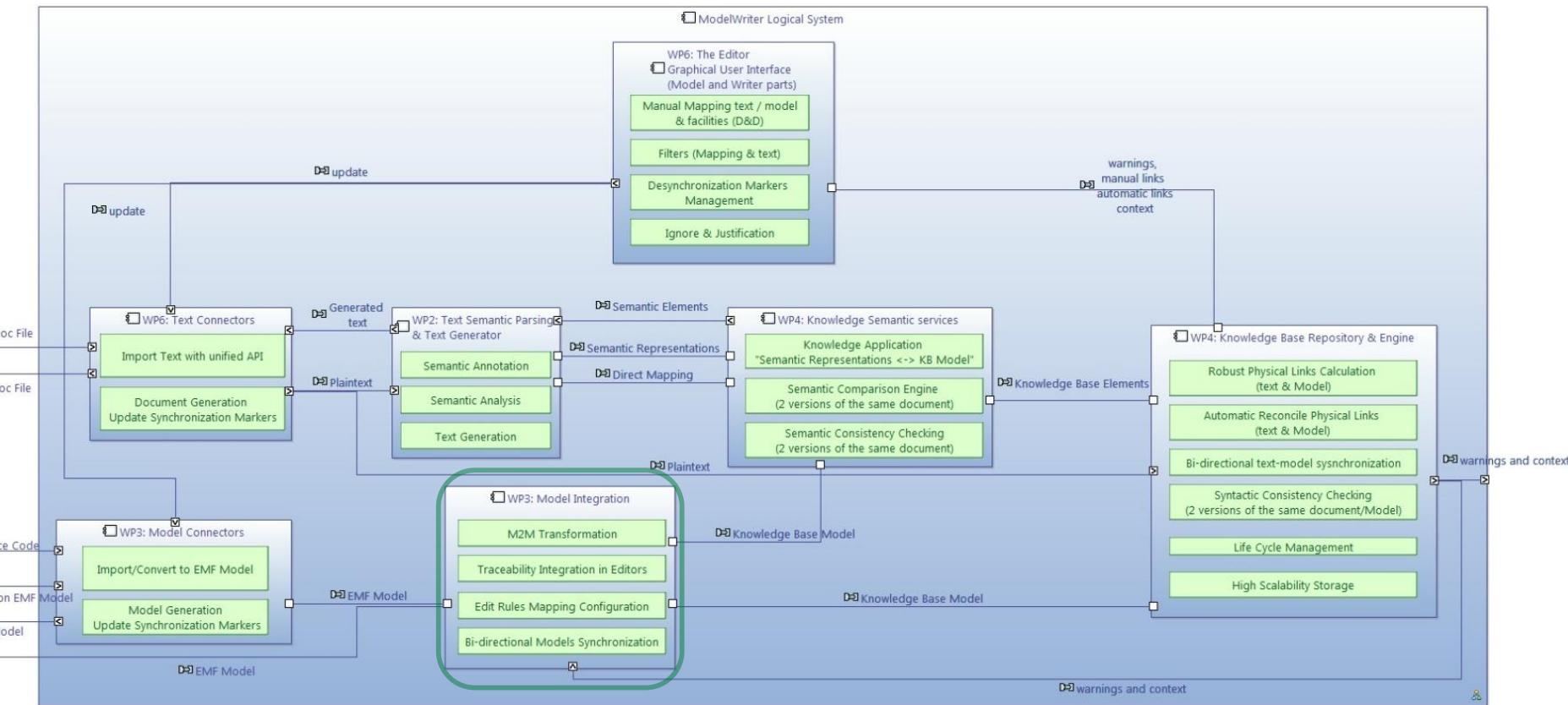
Model connectors

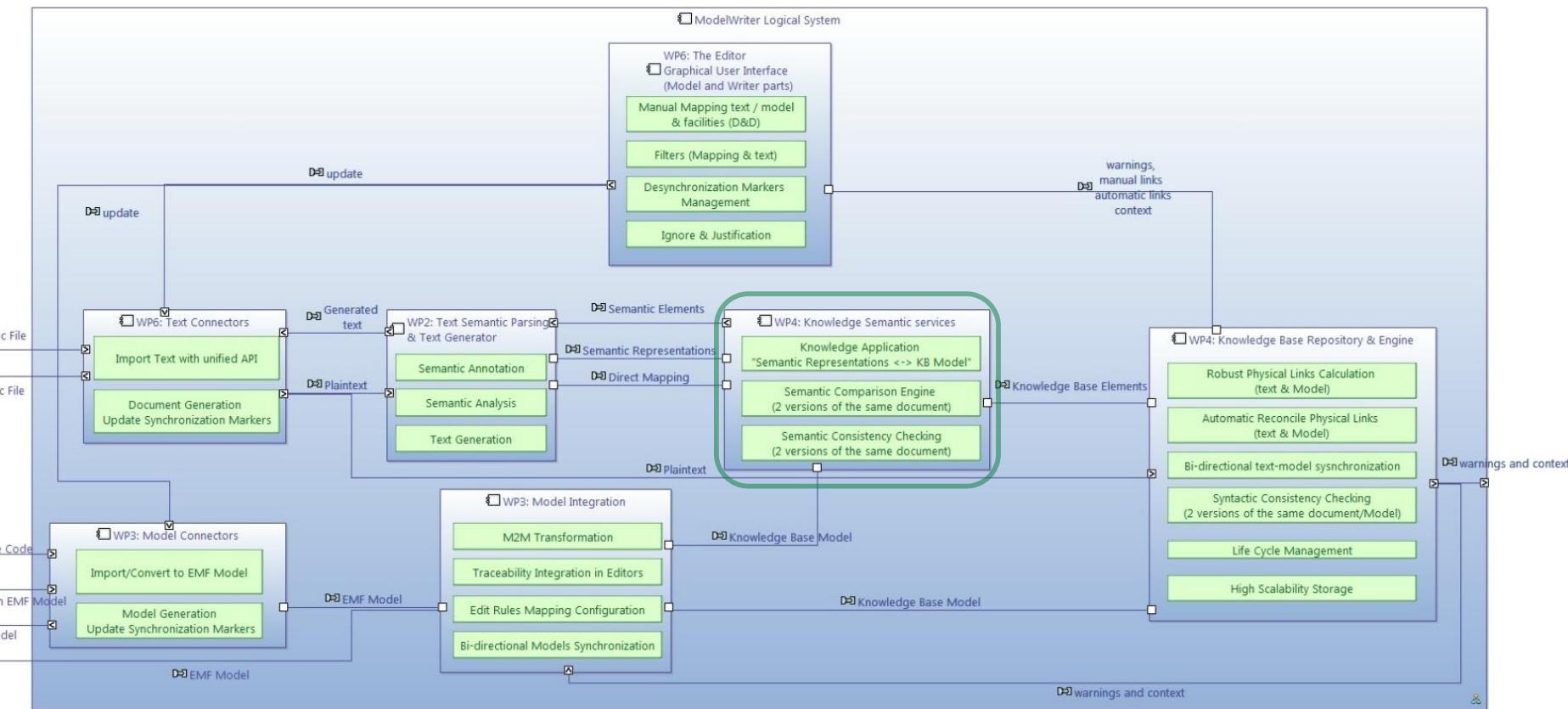


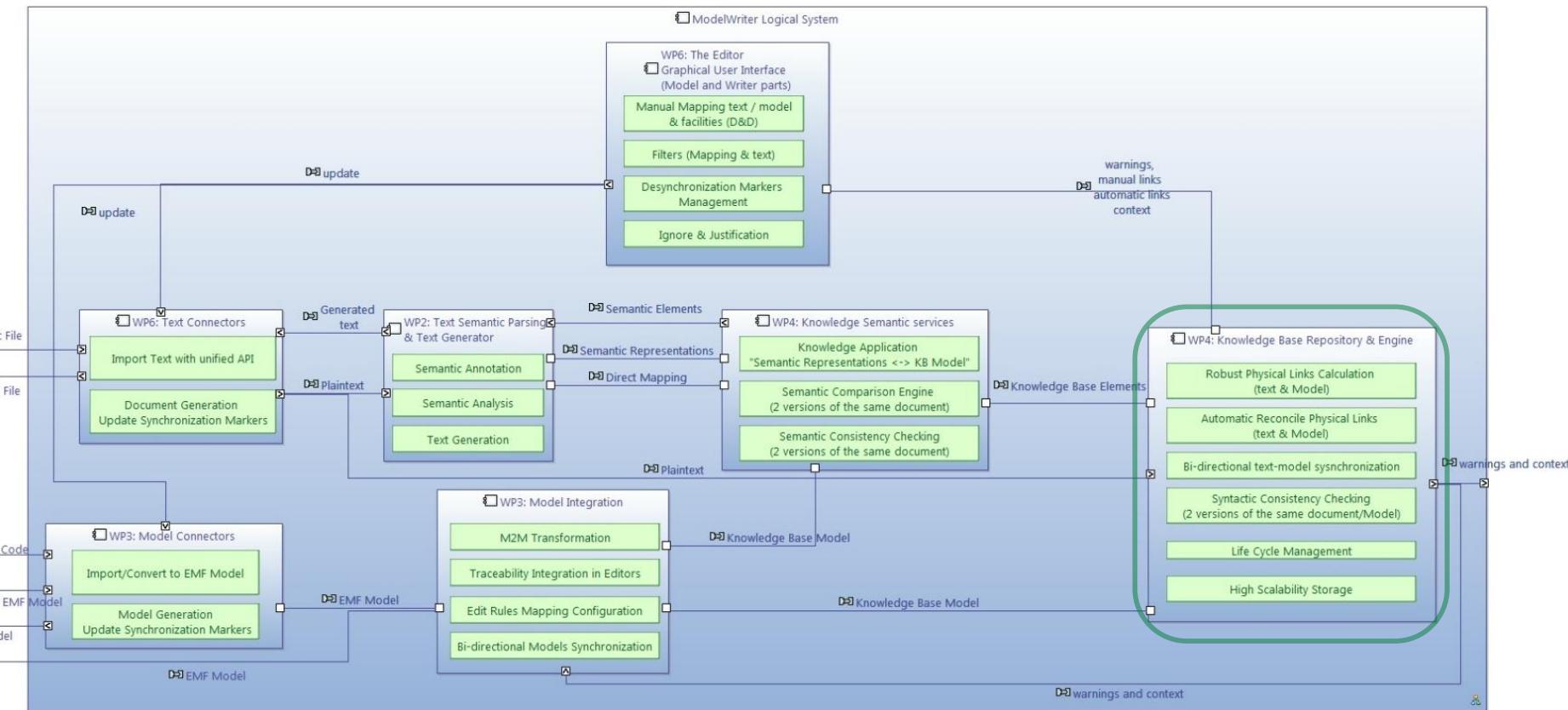


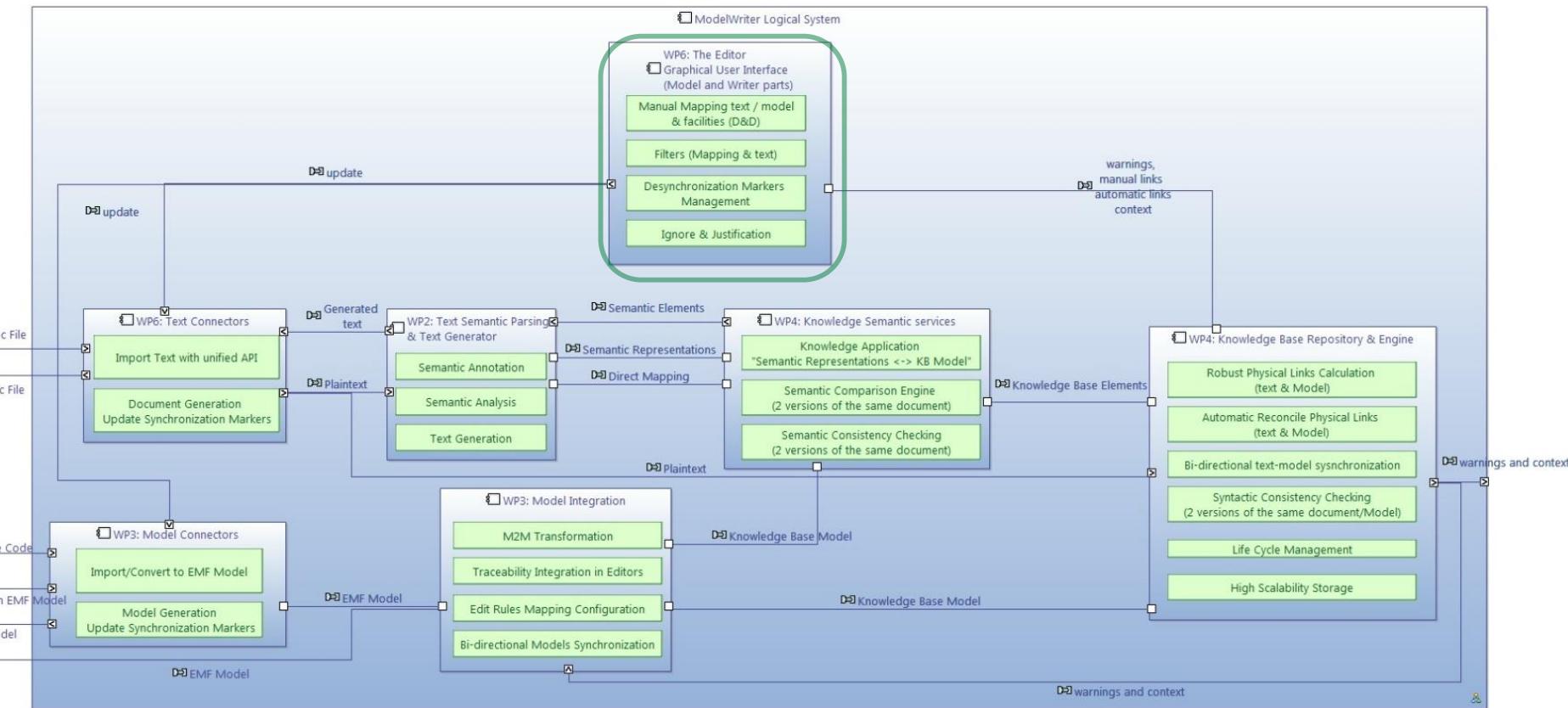
WP6

Model Integration









Source code

- Github repository
- Checkstyle and code templates
- Target platforms

Continuous integration

- Jenkins
- Ease the release process

Next steps

- integrate existing components
- Provide an update site and an Eclipse product

Unit testing

- JUnit
- Code coverage (Eclemma)

Integration testing

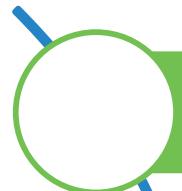
- Functional testing via GUI (RCPTT)
- Jenkins will run all tests on a daily basis

Use cases

- Drives features and enhancements
- Milestone functional testing

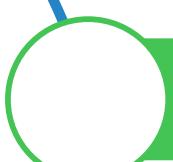
WP7 – Standardization, Dissemination and Exploitation

Yvan Iussaud
OBEO



Specification and verification of ALM platform

- Open source software - traceability



Change impact analysis and visualization

- Open source software – relational calculus



System installation component ontology

- De facto standard – Airbus vocabulary



Semantic annotator

- Open source software – API for text annotation



Synchronization engine prototype

- Open source software – Eclipse Intent contribution



International ModelWriter workshops

- 5 workshops – 2 open workshops



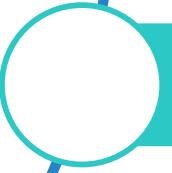
Parsing text into RDF

- Publication poster – propose a RDF-based method for querying the content of a text



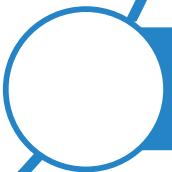
5th Turkish software Architecture conference

- Develop an open source community for model and text synchronization



Keynote on text generation at SEPLN 2015

- Spanish Natural Language Processing Conference – academics and industrials – a project session



Keynote speech at International Workshop on Advanced Topic in Software engineering

- Present Eclipse ecosystem and Modeling approach to software engineering

Collaboration between UNIT and HAVELSAN

- Traceability in ALM platform – applied to Microsoft Team Foundation Server – support of KoçSistem

Requirement documents and ReqIF standard synchronization

- Prototype – automatic synchronization between ReqIf models and requirement documentation

CSV to OWL transformation

- Generates a triple dataset to populate the SIDP (System Installation Design Principle) rule model

SIDP installation rule model

- Model of SIDP installation rules using RDFS and OWL languages

Enhancement in text connector for Airbus

- Syntactical parsing of SIDP rules based on templates

Collaboration/Participation of FORD-Otosan

- Long term support – semantic parsing and traceability for Product Life Cycle documents

Collaboration between Obeo and Airbus

- Discussions on topics related to the ModelWriter scope

Expertise on document extraction

- Improve expertise on information extraction for reverse engineering purpose

**Thank you for your attention
We value your opinion and
questions.**

5 The Project Idea with Demonstrations

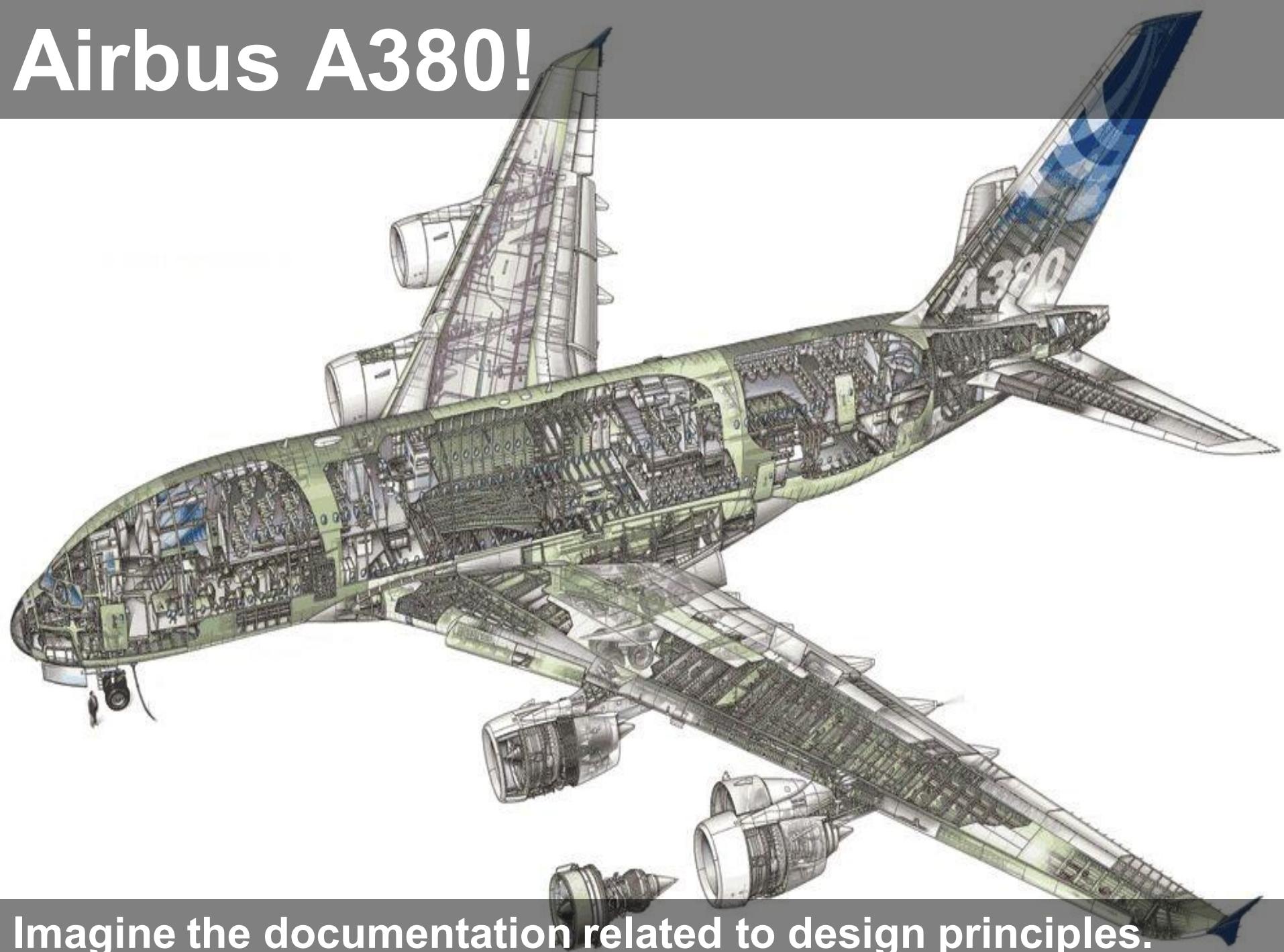
Ferhat Erata (UNIT, ModelWriter Project Leader)

Dr. Mariem Mahfoudh (CNRS/LORIA)

What is the problem?



Airbus A380!

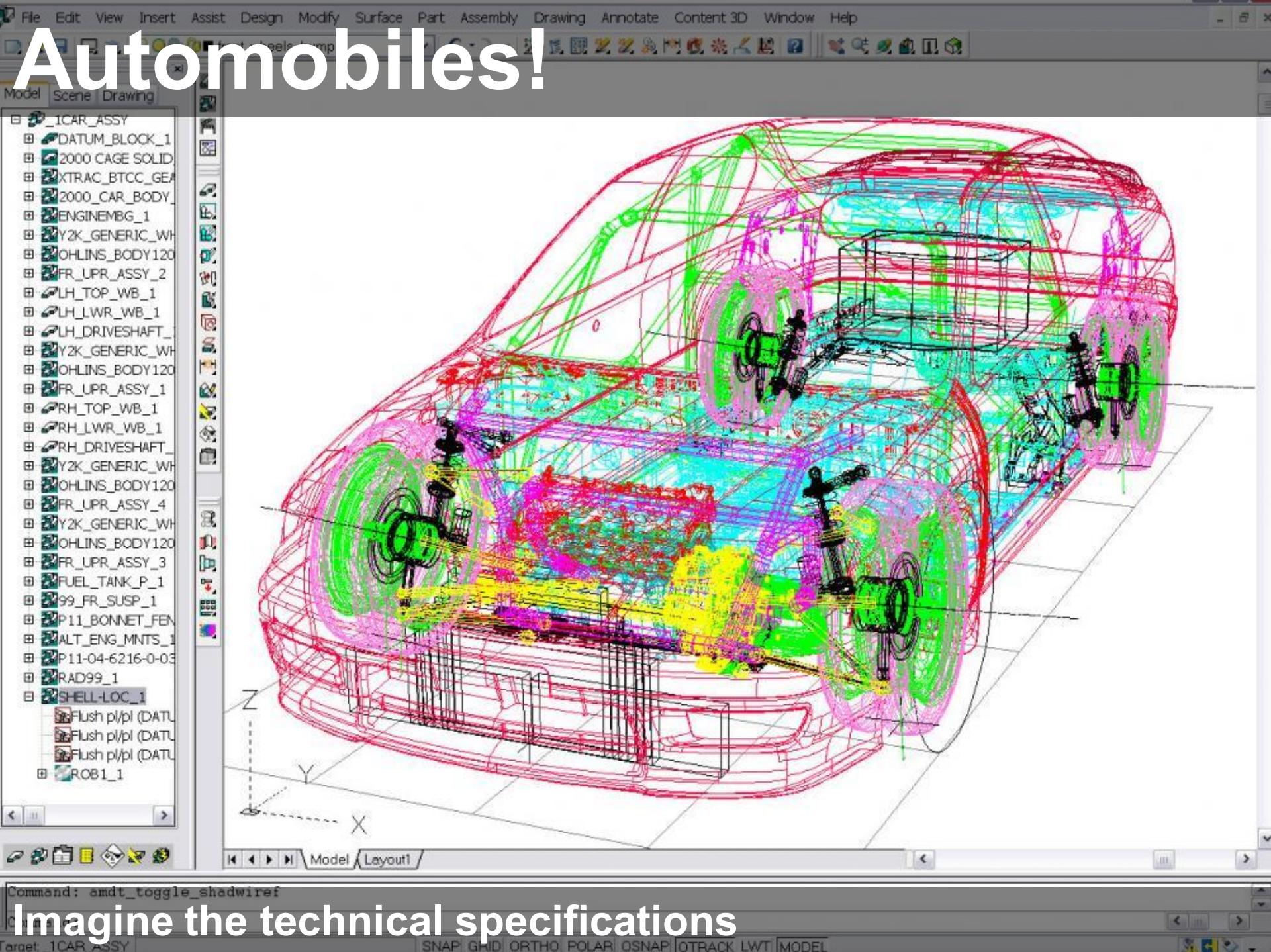


Imagine the documentation related to design principles.



Plants!

Imagine the documentation of a construction site.



Automobiles!

Command: amdt_toggle_shadwire

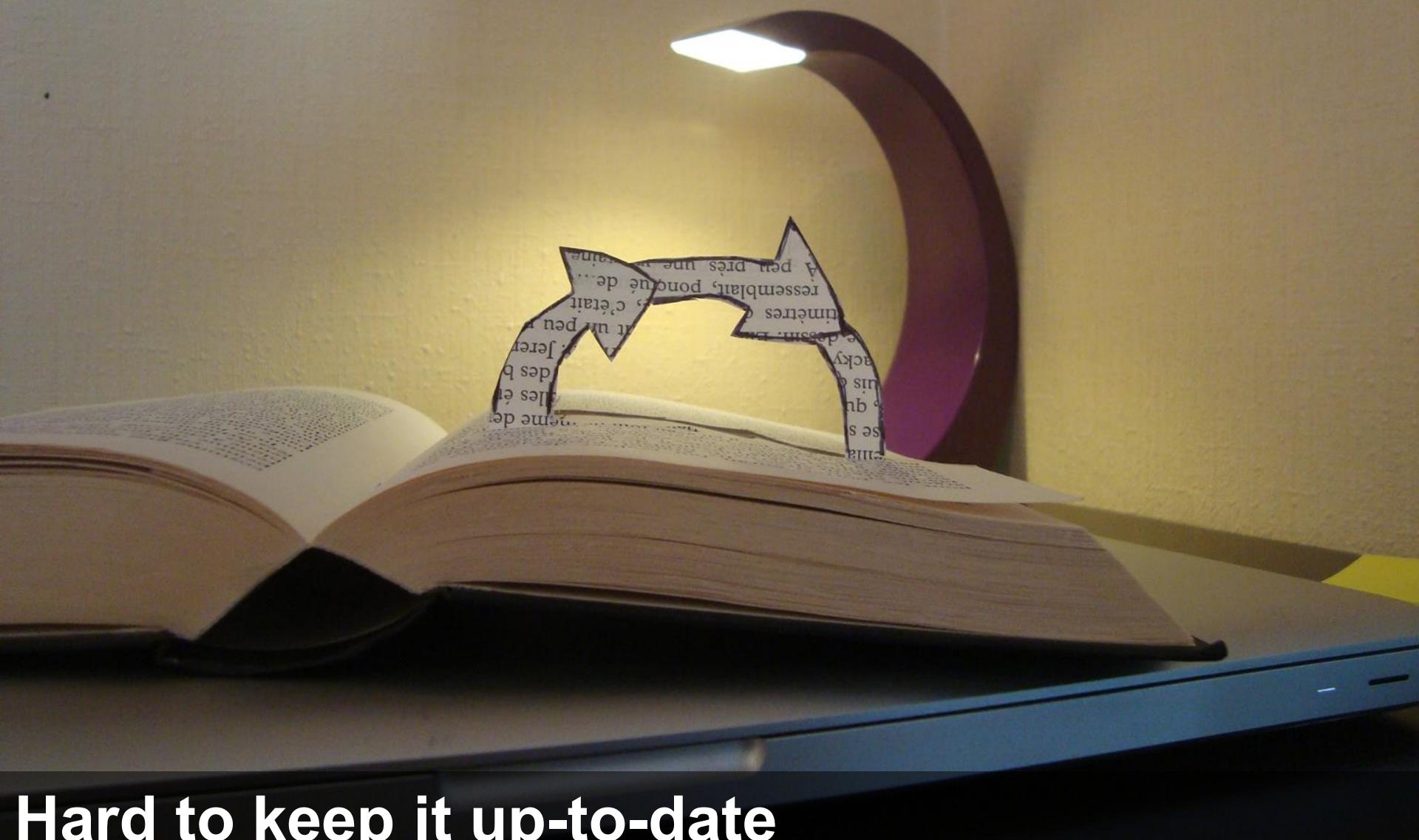
Imagine the technical specifications

Documentation!



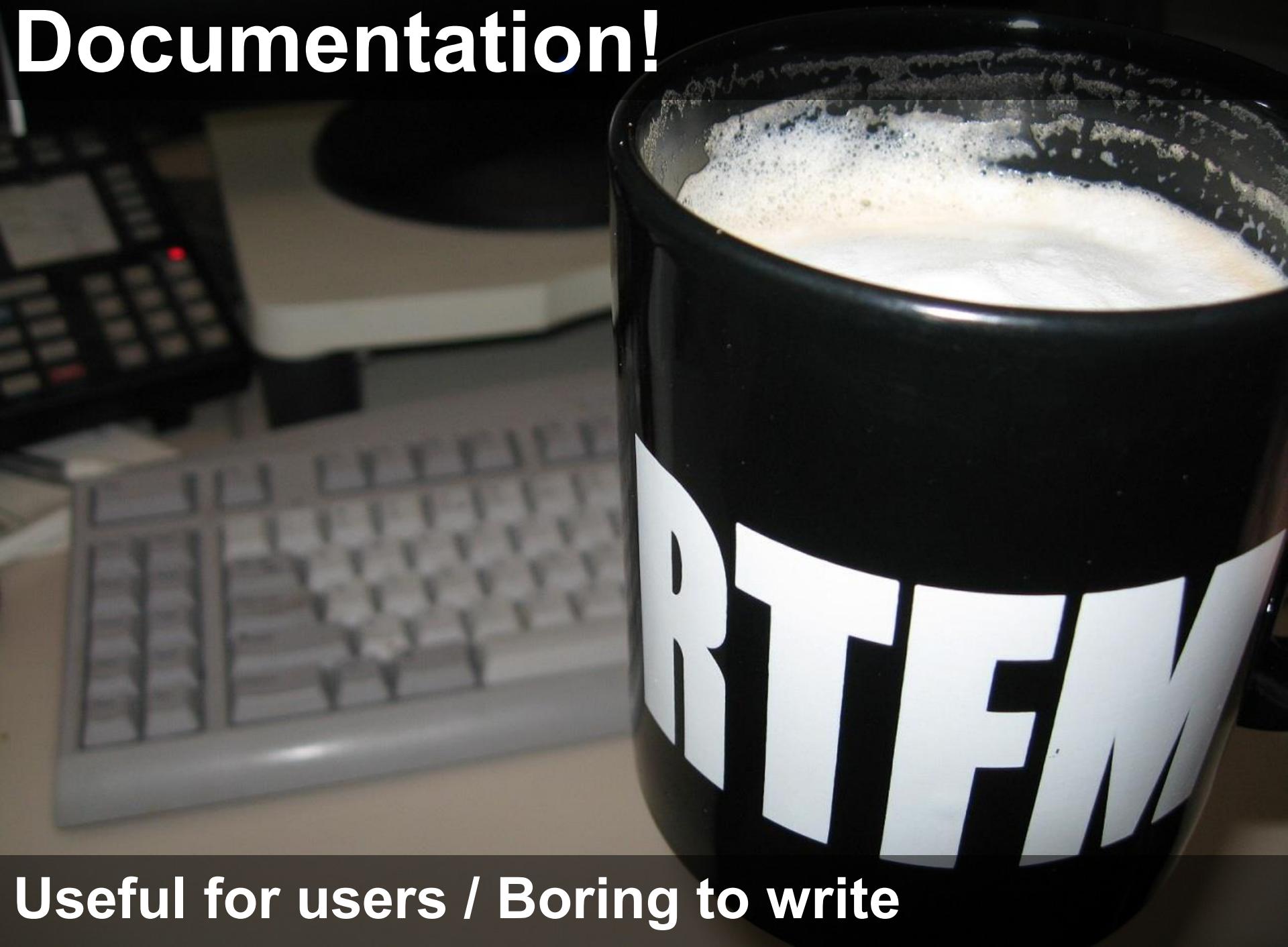
Write once ... and never look at after

Documentation!



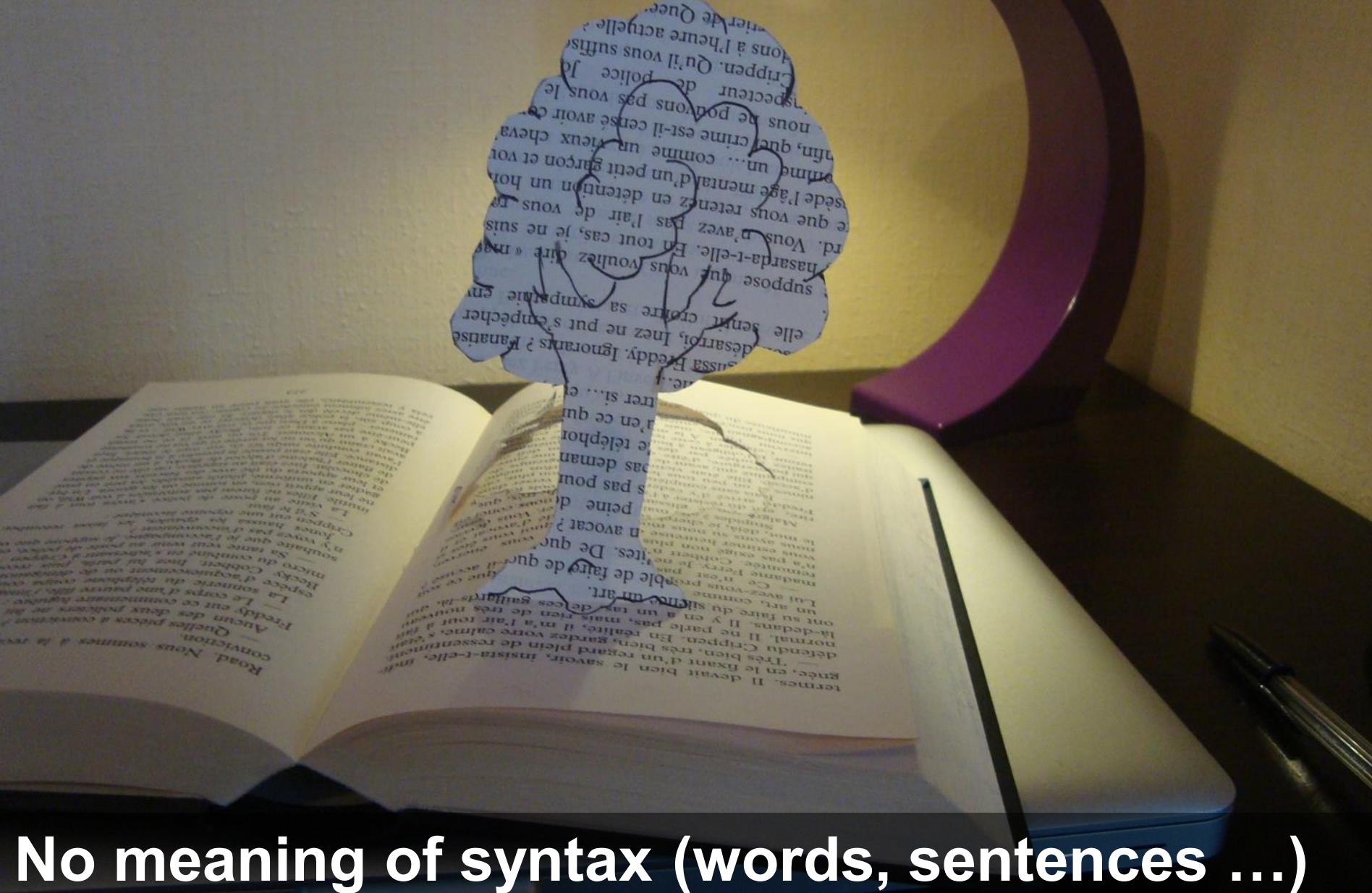
Hard to keep it up-to-date

Documentation!



Useful for users / Boring to write

Semantics!



No meaning of syntax (words, sentences ...)

What is a text?

What is a text? (document file formats)

Office Open XML (.docx) (ISO/IEC 29500)



The screenshot shows a Microsoft Word document titled "Library Management System.docx". The document structure is as follows:

- GLOSSARY**
 - 1.1 BOOK
Book is a kind of collection item. It has an author or editor and (...)
 - 1.2 ...
- REQUIREMENTS**
 - 1.1 REQUIREMENT - RESPONSE TIME FOR BOOK SEARCHES
The system shall perform all book search operations in less than 3 seconds.
 - 1.2 REQUIREMENT - VALIDATION OF THE BOOK
The system allows the user to add new book data through a special book form. The system validates book before storing it.
 - 1.3 ...

The Word ribbon is visible at the top, showing tabs like FILE, HOME, and INSERT. The left sidebar includes a navigation pane with headings and pages, and a search bar. The bottom status bar shows page 1 of 1, 76 words, and the language is set to English (United States).

What is a text? (document file formats)

Office Open XML (.docx) (ISO/IEC 29500)



Java - PropertyPage/test/document.xml - Eclipse

File Edit Source Navigate Search Project Sample Run Window Help

Sample Plain Text File document.xml

```
19    xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
20    <w:body>
21        <w:p w:rsidR="001D662C" w:rsidRDefault="004D2229" >
22            <w:pPr>
23                <w:pStyle w:val="Title" />
24            </w:pPr>
25            <w:bookmarkStart w:id="0" w:name="_GoBack" />
26            <w:bookmarkEnd w:id="0" />
27            <w:r>
28                <w:t xml:space="preserve">Library Management System </w:t>
29            </w:r>
30        </w:p>
31        <w:p w:rsidR="004D2229" w:rsidRDefault="004D2229" w:rsidP="004D2229">
32            <w:pPr>
33                <w:pStyle w:val="Heading1" />
34            <w:numPr>
35                <w:ilvl w:val="0" />
36                <w:numId w:val="0-1" />
37            </w:numPr>
```

Design Source

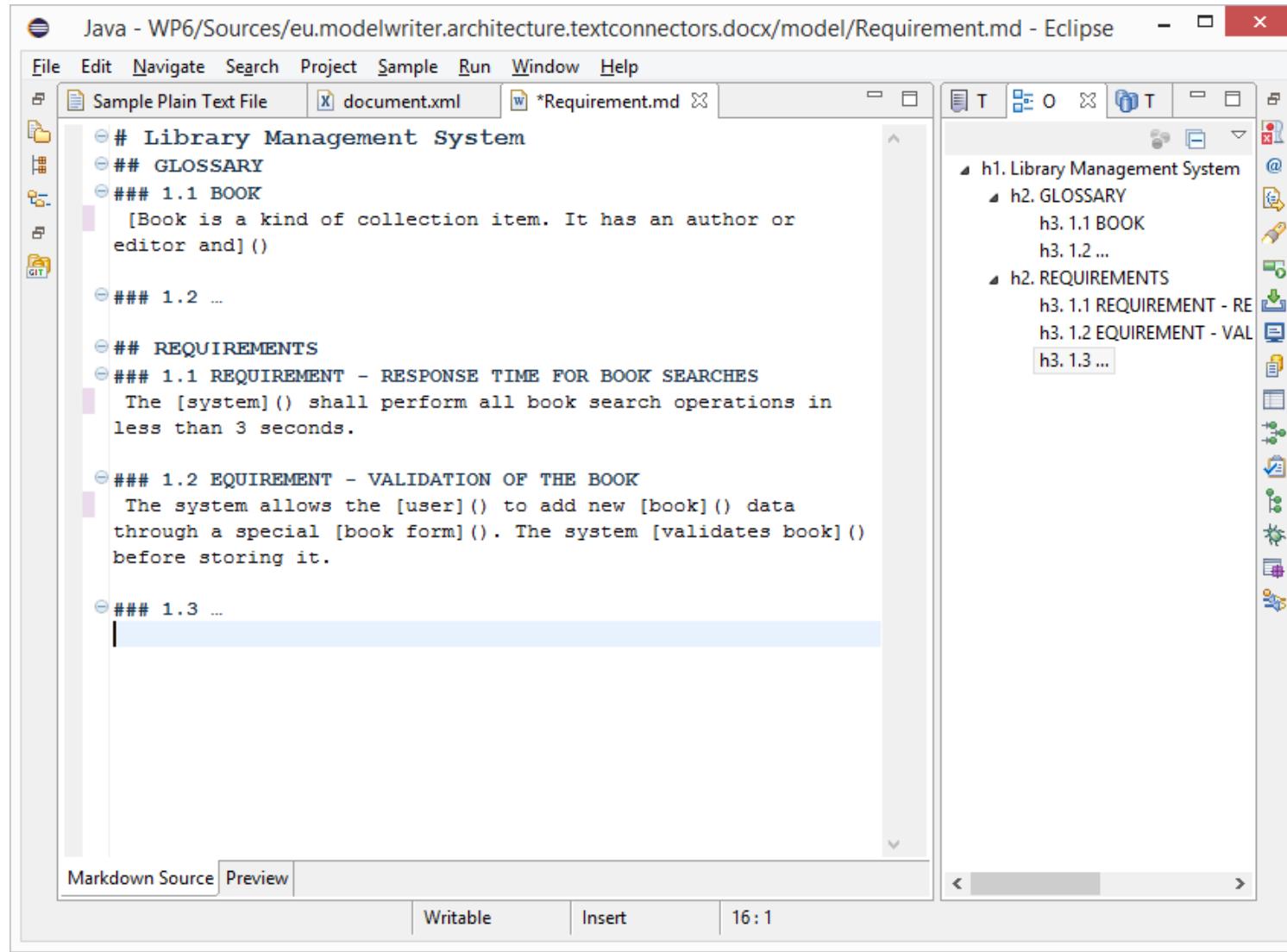
P... @ J... D... S... P... G... C... H... P... E... C... T... E... D... E... P...

Property	Value
w:val	Heading1

w:document/w:body/w:...:pPr/w:pStyle/w:val Writable Smart Insert 33 : 38

A screenshot of the Eclipse IDE interface. The main window displays the XML code for a Microsoft Word document (.docx). The XML code includes elements like <w:body>, <w:p>, <w:pPr>, <w:pStyle>, <w:bookmarkStart>, <w:bookmarkEnd>, <w:r>, <w:t>, <w:pPr>, <w:pStyle>, <w:numPr>, <w:ilvl>, and <w:numId>. The code shows a title 'Library Management System' and a heading 1 'Heading1'. Below the XML editor, there are tabs for 'Design' and 'Source', and a toolbar with various icons. At the bottom, there is a property grid showing the 'w:val' property of a style element set to 'Heading1'. The status bar at the bottom indicates the path 'w:document/w:body/w:...:pPr/w:pStyle/w:val', the status 'Writable', and the time '33 : 38'.

What is a text? (.md source file) text/markdown (ICANN Standard)



The screenshot shows the Eclipse IDE interface with a Markdown file open. The title bar reads "Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Requirement.md - Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Sample, Run, Window, and Help. The toolbar has icons for Sample Plain Text File, document.xml, and *Requirement.md. The left pane shows a tree view of the document structure:

- # Library Management System
 - ## GLOSSARY
 - ### 1.1 BOOK
 - [Book is a kind of collection item. It has an author or editor and] ()
 - ### 1.2 ...
- ## REQUIREMENTS
 - ### 1.1 REQUIREMENT - RESPONSE TIME FOR BOOK SEARCHES
 - The [system] () shall perform all book search operations in less than 3 seconds.
 - ### 1.2 REQUIREMENT - VALIDATION OF THE BOOK
 - The system allows the [user] () to add new [book] () data through a special [book form] (). The system [validates book] () before storing it.
 - ### 1.3 ...

The right pane shows a detailed tree view of the same structure, with icons for each node. At the bottom, there are tabs for "Markdown Source" and "Preview", and status bars for "Writable", "Insert", and "16:1".

What is a text? (HTML Preview) text/markdown (ICANN Standard)



The screenshot shows the Eclipse IDE interface with a Markdown editor open. The title bar indicates the file is "Requirement.md". The left pane displays the content of the Markdown file:

```
Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Requirement.md - Eclipse
File Edit Navigate Search Project Sample Run Window Help
*ReqModel pa... ReqModel.emf Requirement.md >4
```

Library Management System

GLOSSARY

1.1 BOOK

Book is a kind of collection item. It has an author or editor and

1.2 ...

REQUIREMENTS

1.1 REQUIREMENT - RESPONSE TIME FOR BOOK SEARCHES

The system shall perform all book search operations in less than 3 seconds.

1.2 REQUIREMENT - VALIDATION OF THE BOOK

The system allows the user to add new book data through a special book form. The system validates book before storing it.

1.3 ...

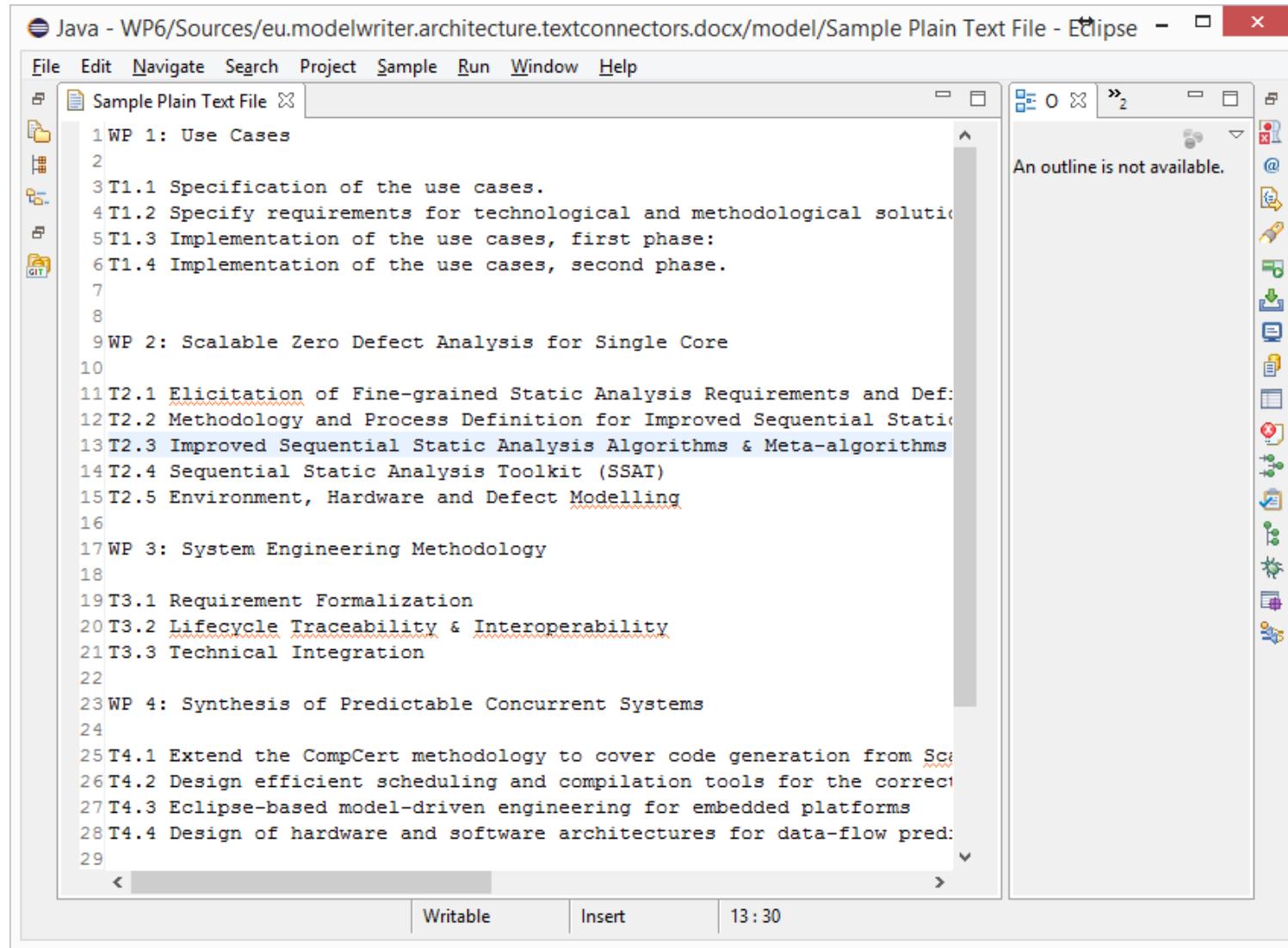
Markdown Source Preview

Writable Insert 16 : 1

The right pane shows a tree view of the document structure:

- h1. Library Management System
 - h2. GLOSSARY
 - h3. 1.1 BOOK
 - h3. 1.2 ...
 - h2. REQUIREMENTS
 - h3. 1.1 REQUIREMENT - RESPON
 - h3. 1.2 EQUIREMENT - VALIDATI
 - h3. 1.3 ...

What is a text? (unformatted text) text/plain (ICANN Standard)



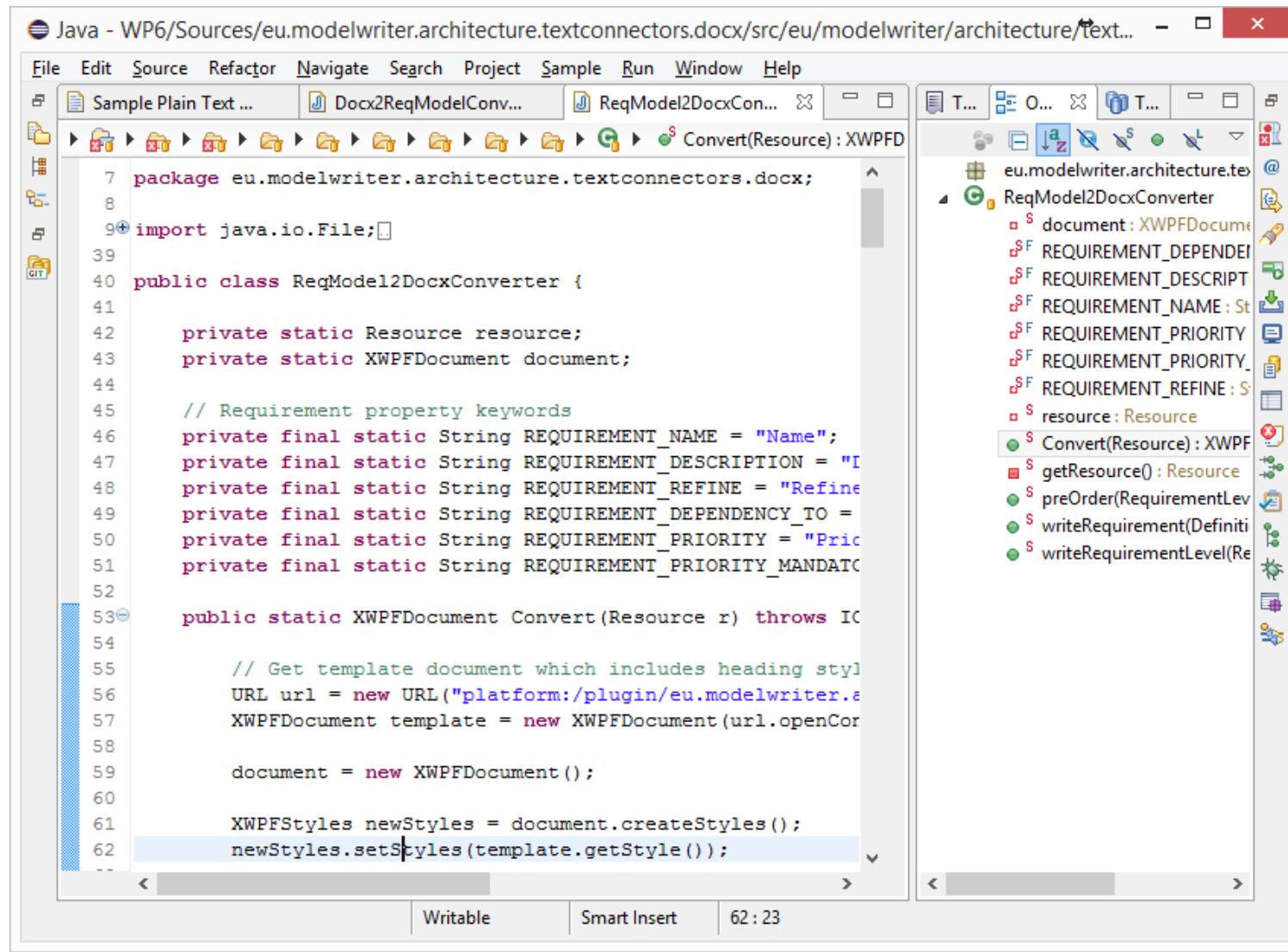
The screenshot shows the Eclipse IDE interface with a text editor open. The title bar reads "Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Sample Plain Text File - Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Sample, Run, Window, and Help. The left sidebar shows a file tree with "Sample Plain Text File" selected. The main editor area contains the following text:

```
1 WP 1: Use Cases
2
3 T1.1 Specification of the use cases.
4 T1.2 Specify requirements for technological and methodological solution
5 T1.3 Implementation of the use cases, first phase:
6 T1.4 Implementation of the use cases, second phase.
7
8
9 WP 2: Scalable Zero Defect Analysis for Single Core
10
11 T2.1 Elicitation of Fine-grained Static Analysis Requirements and Defe
12 T2.2 Methodology and Process Definition for Improved Sequential Static
13 T2.3 Improved Sequential Static Analysis Algorithms & Meta-algorithms
14 T2.4 Sequential Static Analysis Toolkit (SSAT)
15 T2.5 Environment, Hardware and Defect Modelling
16
17 WP 3: System Engineering Methodology
18
19 T3.1 Requirement Formalization
20 T3.2 Lifecycle Traceability & Interoperability
21 T3.3 Technical Integration
22
23 WP 4: Synthesis of Predictable Concurrent Systems
24
25 T4.1 Extend the CompCert methodology to cover code generation from Sc
26 T4.2 Design efficient scheduling and compilation tools for the correct
27 T4.3 Eclipse-based model-driven engineering for embedded platforms
28 T4.4 Design of hardware and software architectures for data-flow pred:
29
```

The status bar at the bottom indicates "Writable" and the time "13:30". To the right of the editor, there is an "Outline" view which displays the message "An outline is not available." Below the editor are several toolbars with various icons.

What is a text? (code files)

Java, C++ ... Programming Languages



The screenshot shows a Java IDE interface with two main panes. The left pane displays the source code for a Java class named `ReqModel2DocxConverter`. The right pane shows the class hierarchy and methods for this class.

```
Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/src/eu/modelwriter/architecture/text... - □ X
```

File Edit Source Refactor Navigate Search Project Sample Run Window Help

Sample Plain Text ... Docx2ReqModelConv... ReqModel2DocxCon... Convert(Resource) : XWPFD

```
7 package eu.modelwriter.architecture.textconnectors.docx;
8
9+ import java.io.File;
10
11 public class ReqModel2DocxConverter {
12
13     private static Resource resource;
14     private static XWPFDocument document;
15
16     // Requirement property keywords
17     private final static String REQUIREMENT_NAME = "Name";
18     private final static String REQUIREMENT_DESCRIPTION = "I";
19     private final static String REQUIREMENT_REFINE = "Refine";
20     private final static String REQUIREMENT_DEPENDENCY_TO =
21     private final static String REQUIREMENT_PRIORITY = "Priorit";
22     private final static String REQUIREMENT_PRIORITY_MANDATORY =
23
24
25     public static XWPFDocument Convert(Resource r) throws IOException {
26
27         // Get template document which includes heading styles
28         URL url = new URL("platform:/plugin/eu.modelwriter.a";
29         XWPFDocument template = new XWPFDocument(url.openConnection());
30
31         document = new XWPFDocument();
32
33         XWPFFormats newStyles = document.createStyles();
34         newStyles.setStyles(template.getStyle());
35 }
```

Writable Smart Insert 62 : 23

T... O... T... T... L... R... @... E... G... ReqModel2DocxConverter

- eu.modelwriter.architecture.textconnectors.docx
- ReqModel2DocxConverter
 - document : XWPFDocument
 - REQUIREMENT_DEPENDENCY_TO : String
 - REQUIREMENT_DESCRIPTION : String
 - REQUIREMENT_NAME : String
 - REQUIREMENT_PRIORITY : String
 - REQUIREMENT_PRIORITY_MANDATORY : String
 - REQUIREMENT_REFINE : String
 - resource : Resource
 - Convert(Resource) : XWPFDocument
 - getResource() : Resource
 - preOrder(RequirementLevel)
 - writeRequirement(Definition)
 - writeRequirementLevel(RequirementLevel)

What is a model?

Everything is a model! (ReqIF Standard)

Requirements Interchange Format



ProR - platform:/resource/LibraryManagementSystem/My.reqif - formalmind Studio

File Edit Search Requirements fmStudio Window Help

Quick Access

My.reqif Requirements Document

Outline

ID Name Description

ID	Name	Description
1	Librarian	Librarian
1.1	R123	Response Time for Book Searches
1.2	R123	The system shall perform all book search operations in less than 3 seconds.
1.3	UC071	Add new Book
1.4	R124	Validation of the Book

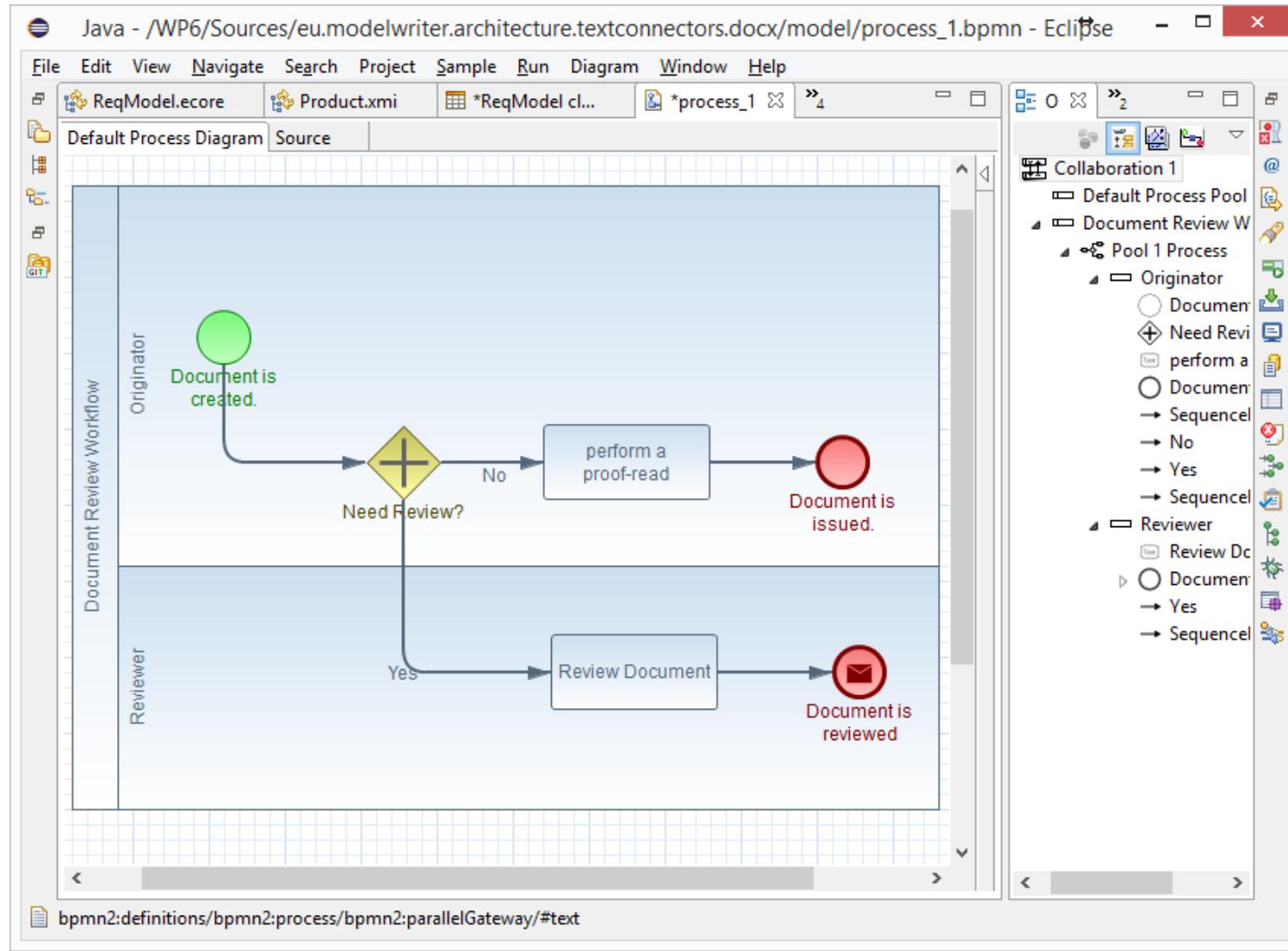
Properties

Property	Value
Requirement Type	
Description	The system shall perform all book search operations in less than 3 second
ID	R123
Name	Response Time for Book Searches
Responsible	Ferhat
Version	1
Spec Object	
Type	Requirement Type (Spec Object)

Standard Attributes All Attributes

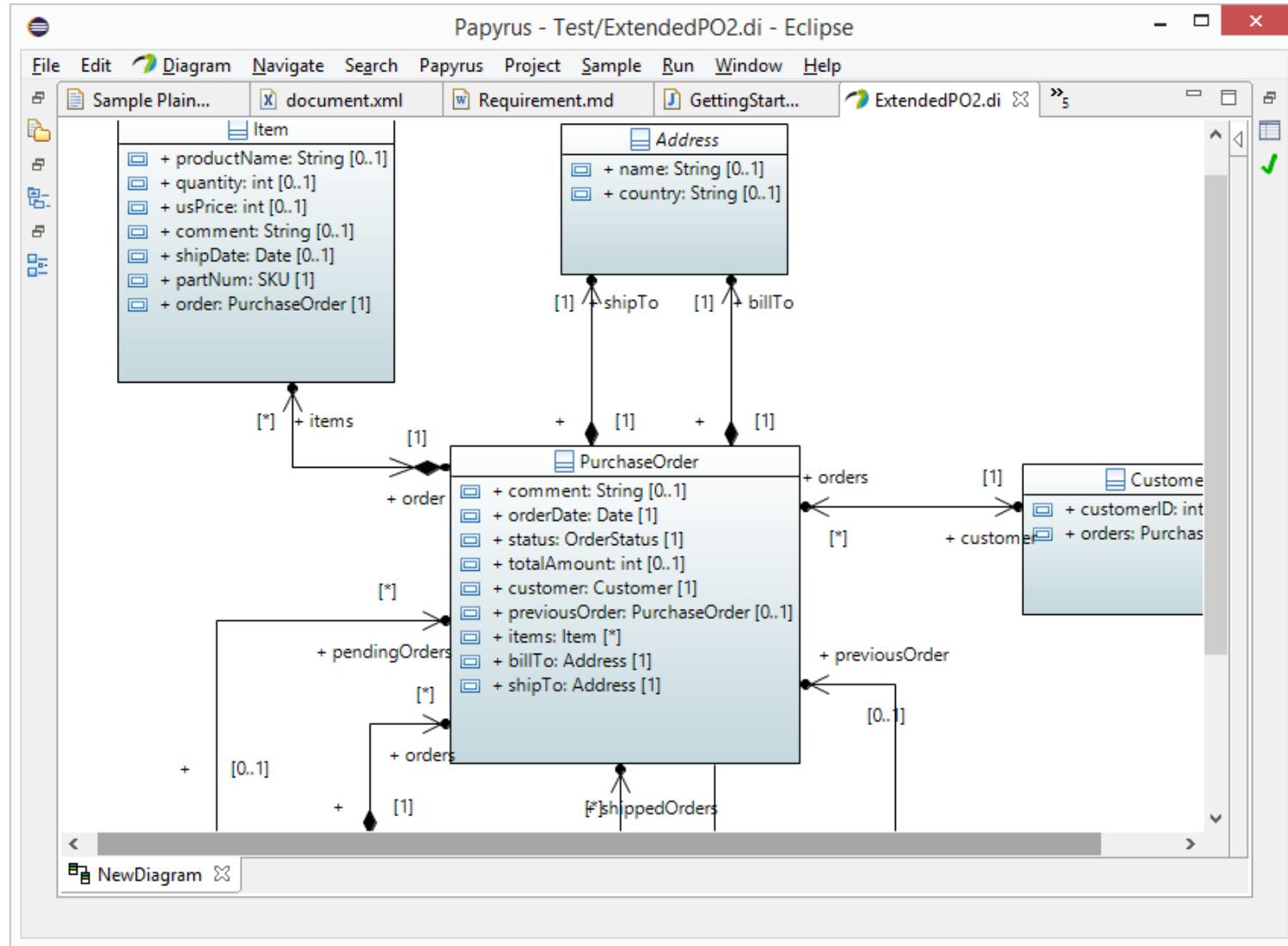
Everything is a model! (BPMN Standard)

Business Process Model & Notation



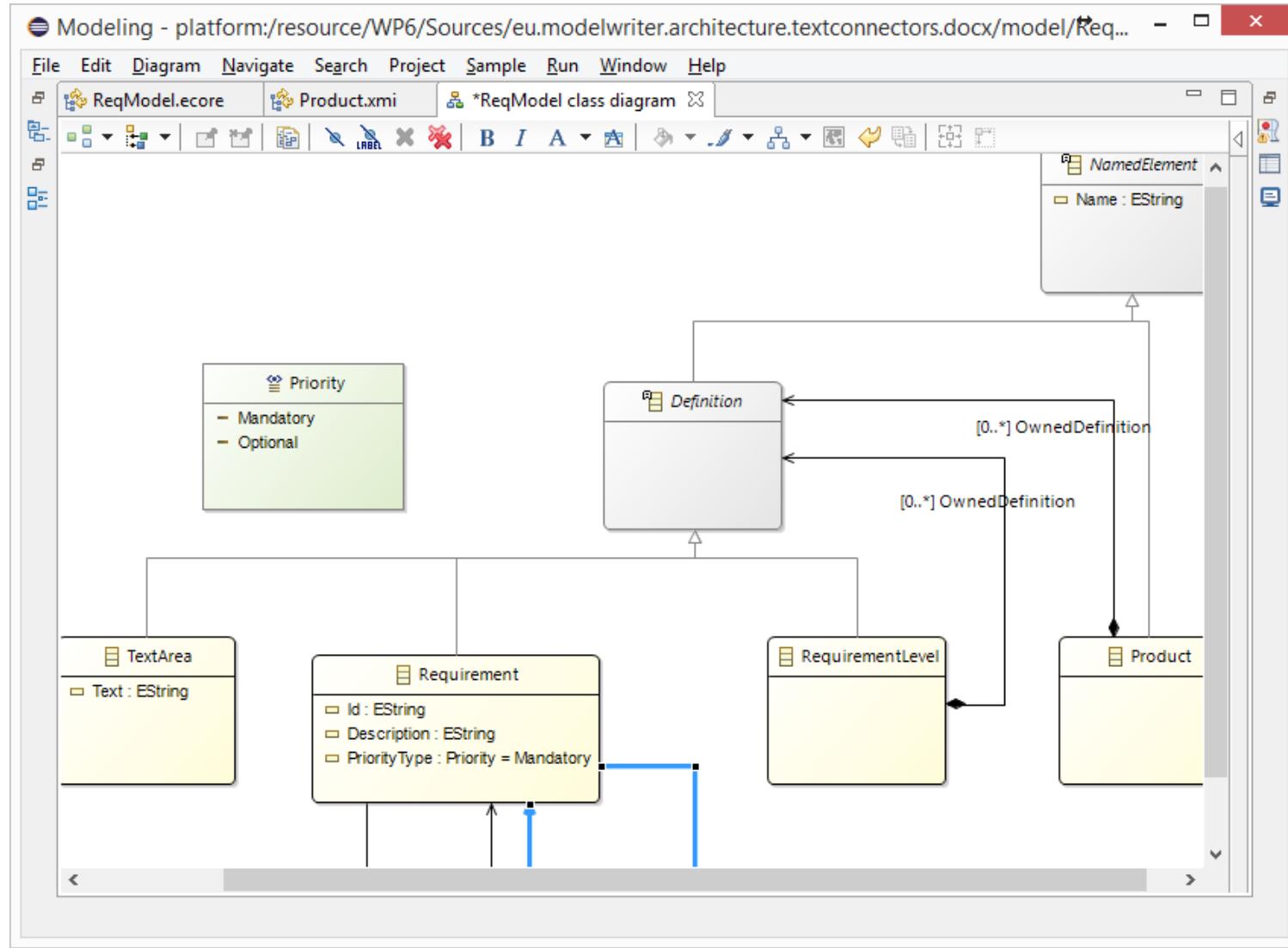
Everything is a model! (UML Standard)

UML Modeling Languages



Everything is a model!

Eclipse Modeling Framework (EMF)



Everything is a model!

Tree-based or Tabular Representations



The screenshot shows a modeling environment with the following components:

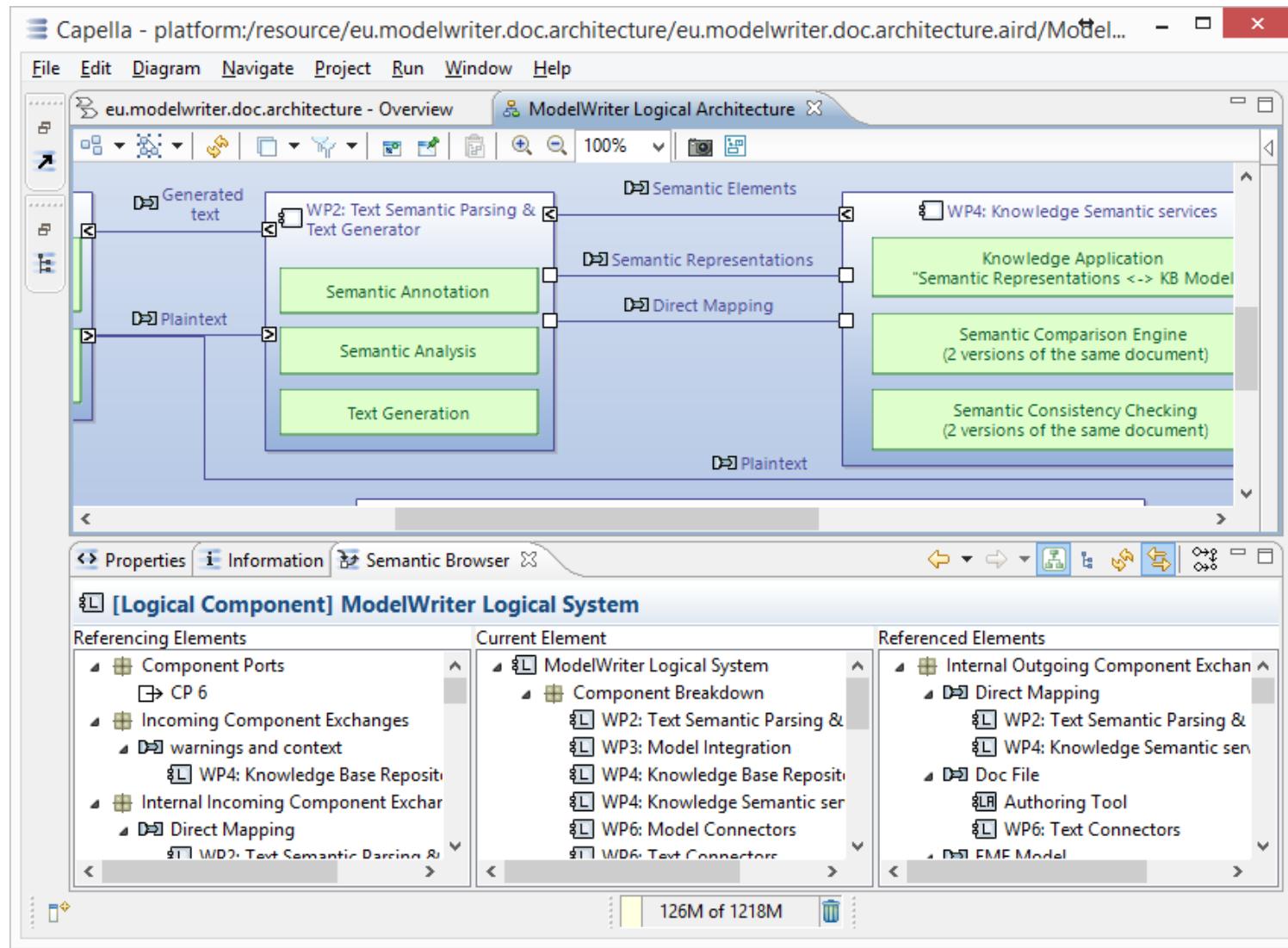
- Top Bar:** File, Edit, Navigate, Search, Project, DTable, Sample, Run, Window, Help.
- Left Sidebar:** Shows a tree view of model elements:
 - ReqModel.ecore
 - Product.xmi
 - NamedElement
 - Name : EString
 - Product -> NamedElement
 - OwnedDefinition : Definition
 - Definition -> NamedElement
 - RequirementLevel -> Definition
 - OwnedDefinition : Definition
 - Requirement -> Definition
 - Id : EString
 - Description : EString
 - Refine : Requirement
 - DependencyTo : Requirement
 - PriorityType : Priority
 - TextArea -> Definition
 - Text : EString
- Middle Panel:** A table titled "*ReqModel class table" showing properties for "NamedElement".

Name	Value
Name	NamedElement
Product	
OwnedDefinition	
Definition	
RequirementLevel	
OwnedDefinition	
Requirement	
Id	
Description	
Refine	
DependencyTo	
PriorityType	
TextArea	
Text	
- Bottom Panel:** Problems, Properties, Console.
- Properties View:** A table for "NamedElement" properties.

Semantic	Property	Value
NamedElement	Abstract	true
	Default Value	
	ESuper Types	
	Instance Type Name	

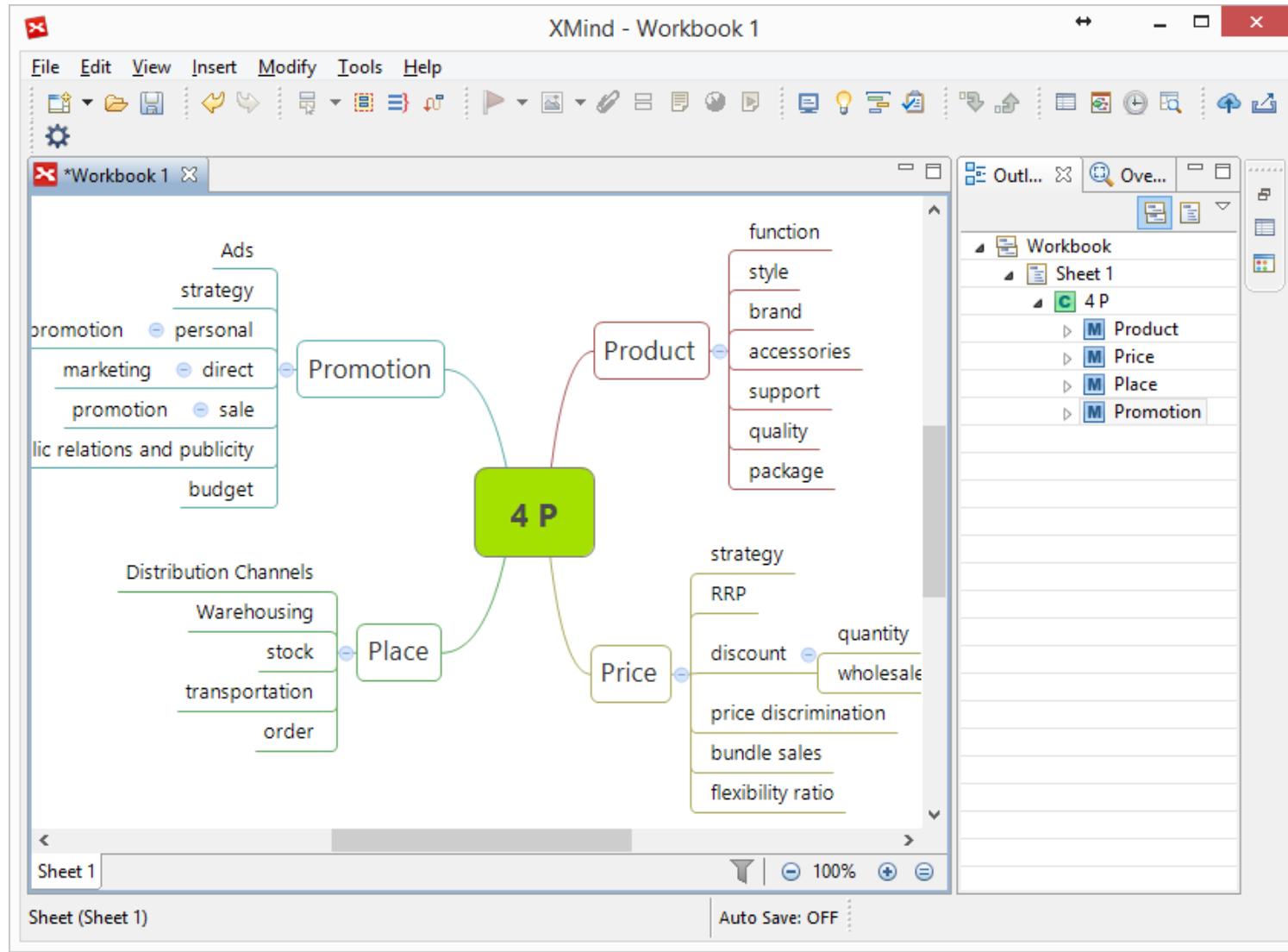
Everything is a model!

Software/System Architecture Design



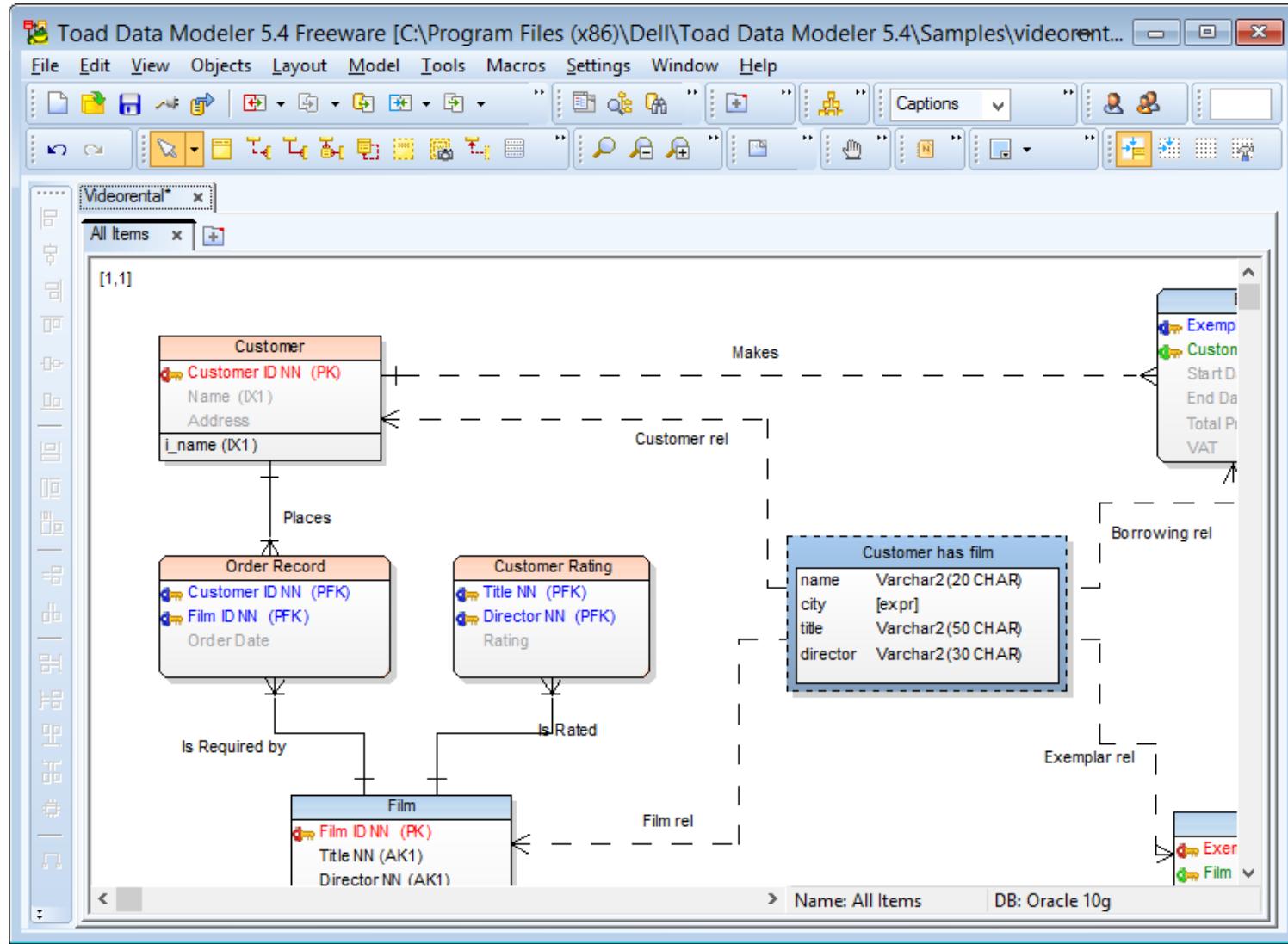
Everything is a model!

Topic Maps, Mind Maps, Vocabularies ...



Everything is a model!

Databases (ER, IDEF1.x)



Everything is a model! (Textual Lang.)

Domain Specific Languages



Modeling - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/ReqModel.emf - Eclipse

File Edit Navigate Search Project Sample Run Window Help

ReqModel.ecore Product.xmi *ReqModel cl... *ReqModel do... ReqModel.emf »

```
1 @namespace(uri="eu.modelwriter.architecture.textconnectors.docx.reqmodel", prefix="ReqMod")
2 package ReqModel;
3
4 @gmf.node(label="name")
5 abstract class NamedElement {
6     attr String Name;
7 }
8
9 @gmf.diagram
10 @gmf.node(label="Name")
11 class Product extends NamedElement {
12
13     @gmf.compartment(collapsible="true")
14     val Definition[*] OwnedDefinition;
15 }
16
17 abstract class Definition extends NamedElement {
18 }
19
20 @gmf.node(figure="rectangle", label.icon="true", label="Name", label.pattern="{0}")
21 class RequirementLevel extends Definition {
22
23     @gmf.compartment(collapsible="true", layout="list")
24     val Definition[*] OwnedDefinition;
25 }
26
27 @gmf.node(figure="rounded", label.icon="true", label="Name", label.pattern="{0}")
28 class Requirement extends Definition {
29     attr String Id = "";
30 }
```

Writable Insert 11:9

Everything is a model! (Java, C++, etc.)

Even Programming Languages (ASTs)



The screenshot shows a Java IDE interface with two main panes. The left pane displays the source code for `ReqModel2DocxConverter`. The right pane shows the Abstract Syntax Tree (AST) representation of the same code.

```
Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/src/eu/modelwriter/architecture/text... - □ X
```

File Edit Source Refactor Navigate Search Project Sample Run Window Help

Sample Plain Text ... Docx2ReqModelConv... ReqModel2DocxCon... Convert(Resource) : XWPFD

```
7 package eu.modelwriter.architecture.textconnectors.docx;
8
9+ import java.io.File;
10
11 public class ReqModel2DocxConverter {
12
13     private static Resource resource;
14     private static XWPFDocument document;
15
16     // Requirement property keywords
17     private final static String REQUIREMENT_NAME = "Name";
18     private final static String REQUIREMENT_DESCRIPTION = "I";
19     private final static String REQUIREMENT_REFINE = "Refine";
20     private final static String REQUIREMENT_DEPENDENCY_TO =
21     private final static String REQUIREMENT_PRIORITY = "Priorit";
22     private final static String REQUIREMENT_PRIORITY_MANDATORY =
23
24
25     public static XWPFDocument Convert(Resource r) throws IOException {
26
27         // Get template document which includes heading styles
28         URL url = new URL("platform:/plugin/eu.modelwriter.a";
29         XWPFDocument template = new XWPFDocument(url.openConnection());
30
31         document = new XWPFDocument();
32
33         XWPFFormats newStyles = document.createStyles();
34         newStyles.setStyles(template.getStyle());
35 }
```

Writable Smart Insert 62 : 23

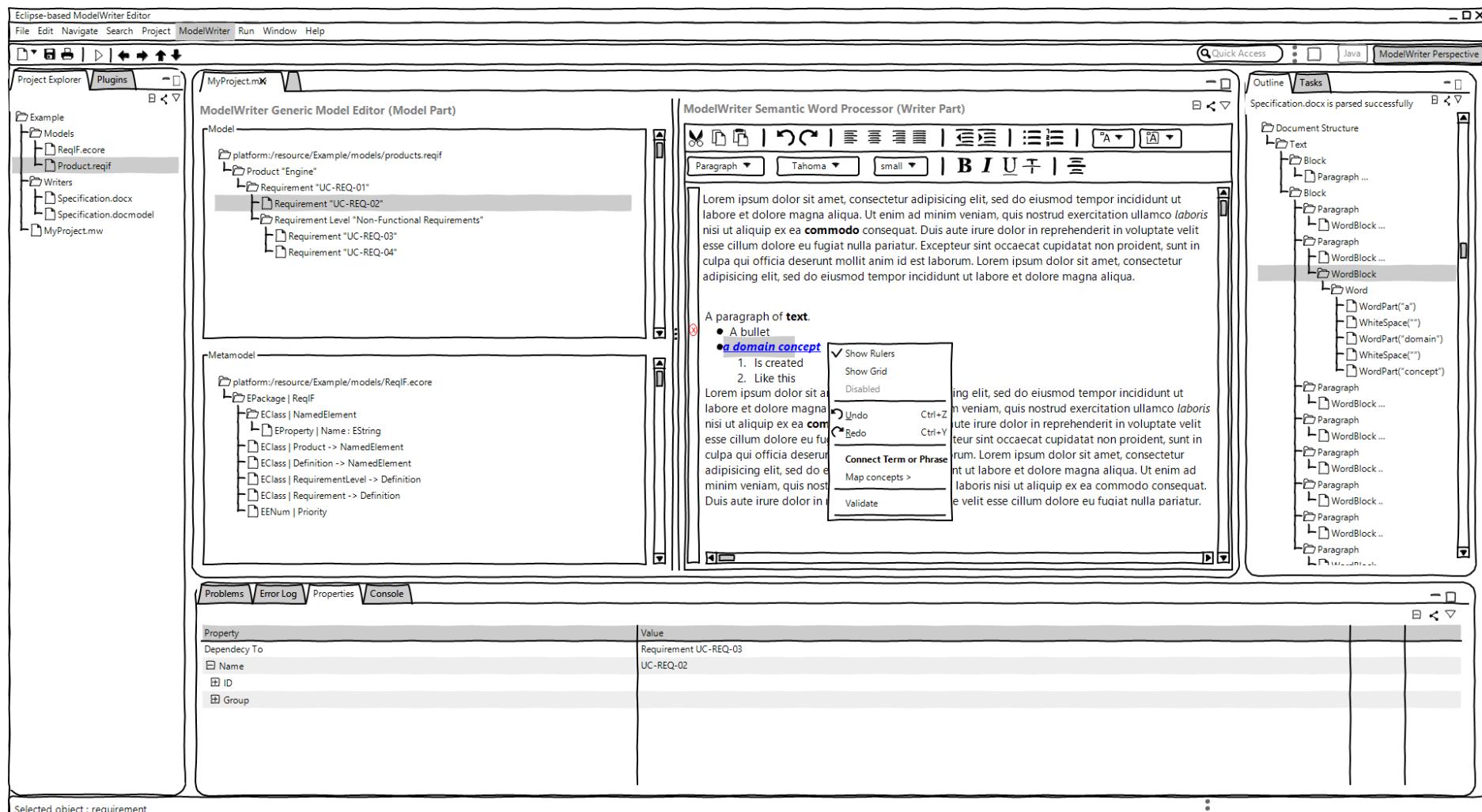
T... O... X T... L... R... @... E... S... L... ReqModel2DocxConverter

- eu.modelwriter.architecture.textconnectors.docx
- ReqModel2DocxConverter
 - S document : XWPFDocument
 - S REQUIREMENT_DEPENDENCY_TO : String
 - S REQUIREMENT_DESCRIPTION : String
 - S REQUIREMENT_NAME : String
 - S REQUIREMENT_PRIORITY : String
 - S REQUIREMENT_PRIORITY_MANDATORY : String
 - S REQUIREMENT_REFINE : String
 - S resource : Resource
 - S Convert(Resource) : XWPFDocument
 - S getResource() : Resource
 - S preOrder(RequirementLevel)
 - S writeRequirement(Definition)
 - S writeRequirementLevel(RequirementLevel)

Is it possible to connect and keep arbitrary software/system engineering artifacts synchronized ?

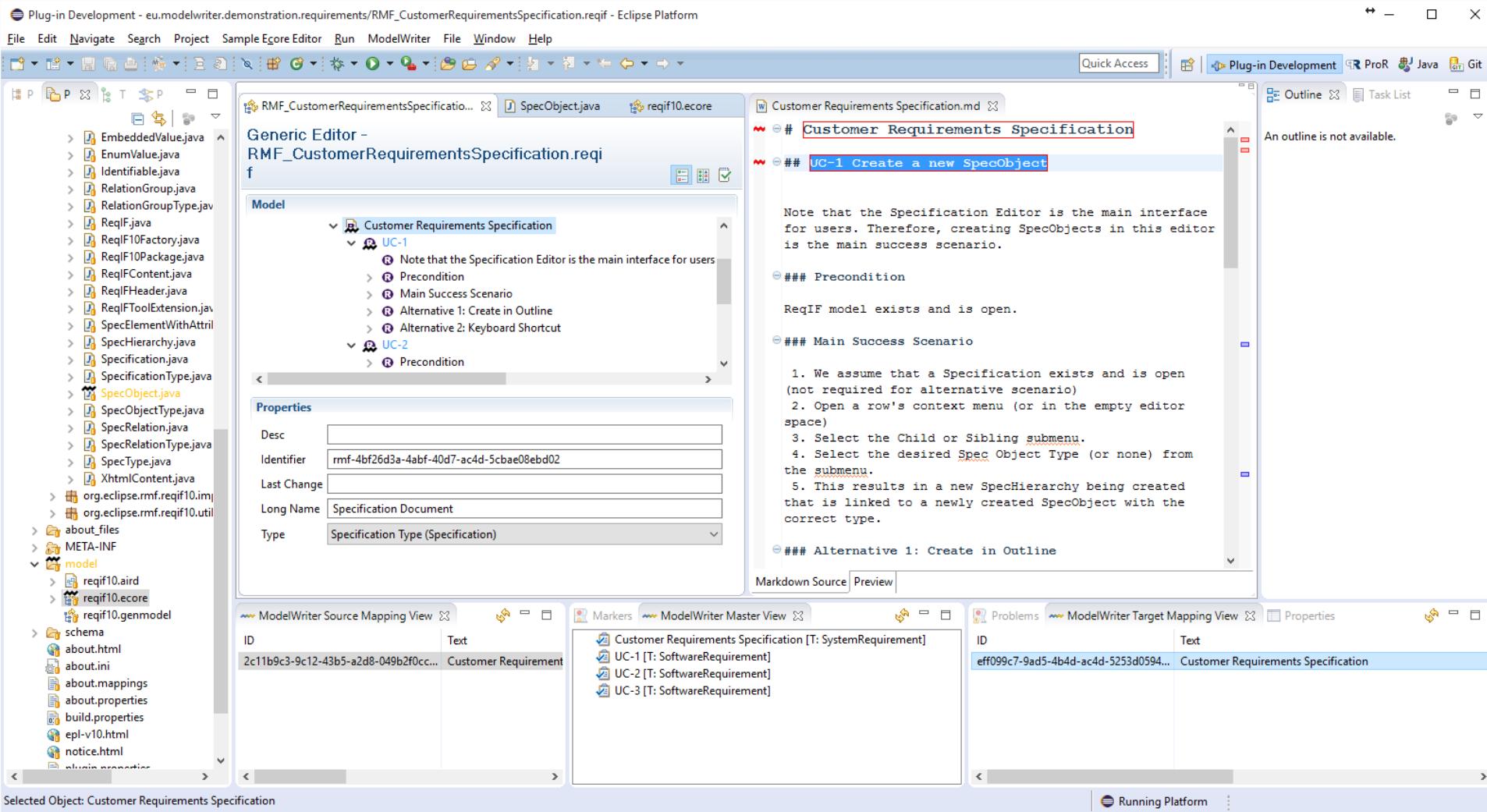


ModelWriter – The Solution



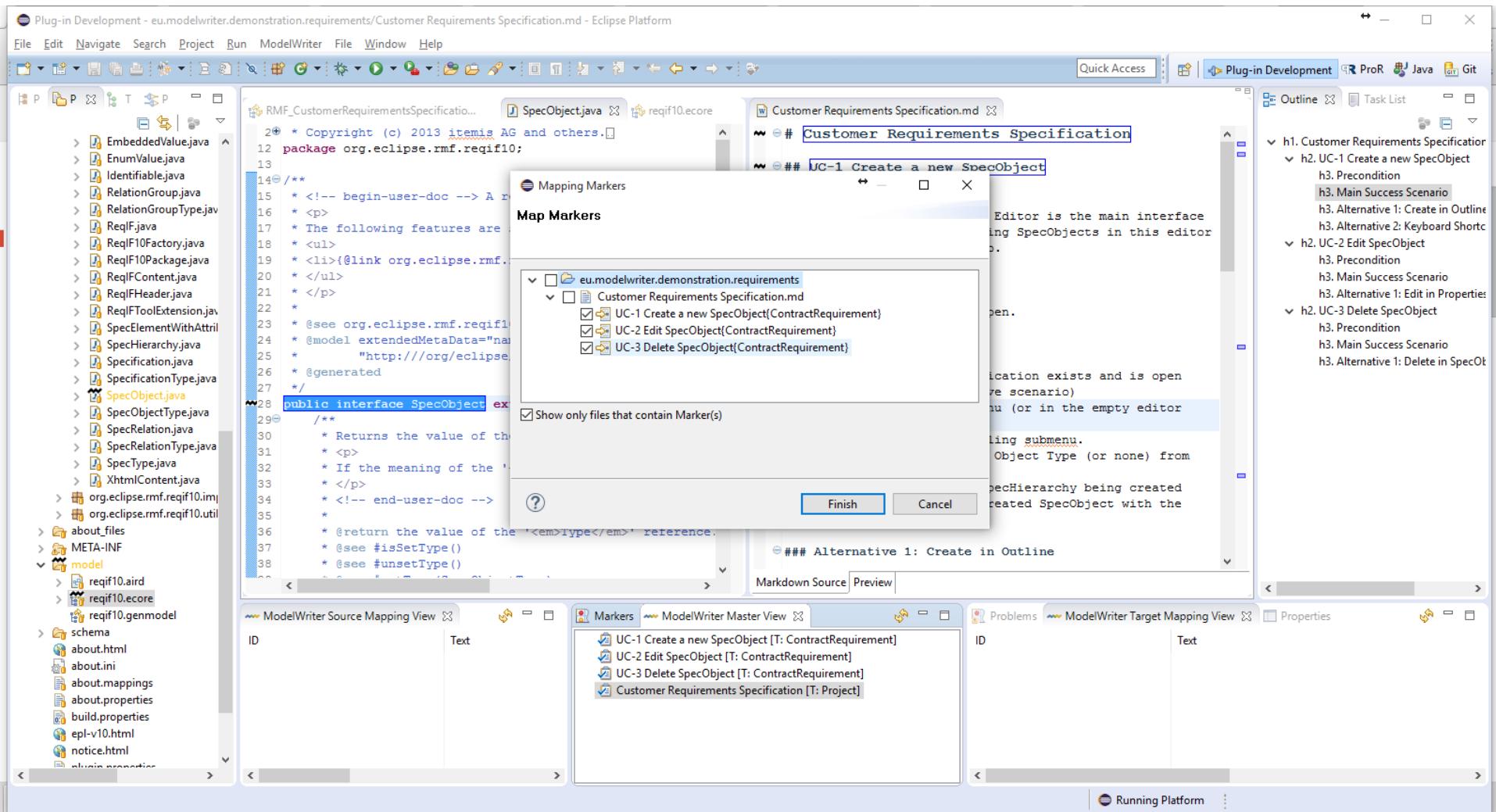
Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



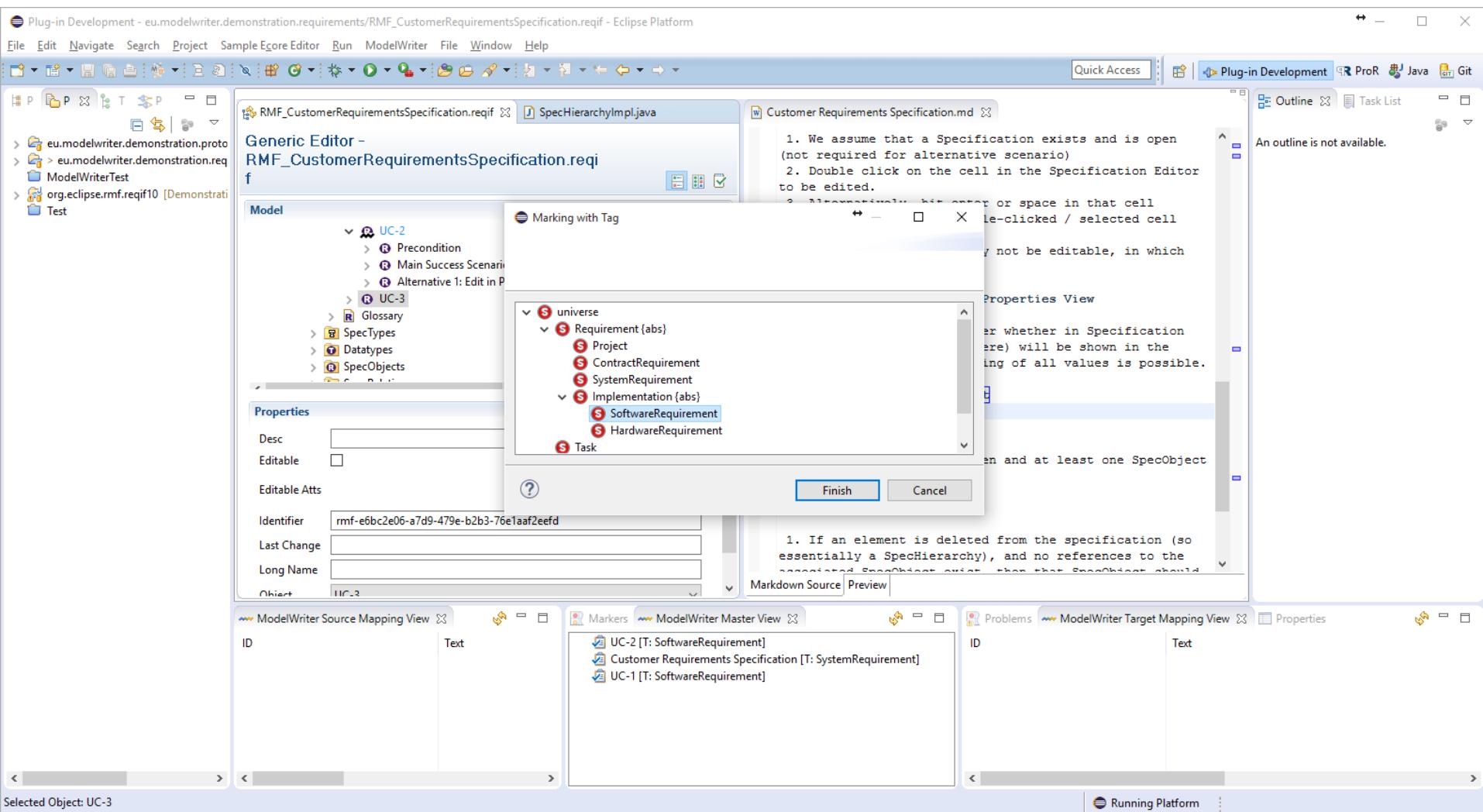
Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

Is it possible to extract
knowledge from texts
fragments based on a given
ontology (model) ?



Solution – Knowledge Extraction

ModelWriter Project

File Link Change Statistic

The

- Generate Links
- Search Link
- 2 P
- Add Link
- ABS: Remove Link

unction zones
shall be used

Flexible Hoses Shall be defined with a maximum length of 500 mm regardless of
ABS2195 -LRB- preferred for weight saving -RRB- or NSA5516J P-Clamp Shall be U
Rigid Pipes Shall be segregated to fixed Structure by not less than 10 mm as sh
Flexible Hoses Shall be segregated to rigid Component/Item/Object by not less t
Rigid Pipes Shall be segregated to movable Component/Item/Object by not less
Flexible Hoses Shall be segregated to movable Component/Item/Object by not le
Pipes Shall be fixed
Unions Shall be fixed on Pipes at alternating positions as shown in the attach
Unions Shall be positioned close to one fixation point .

The Model

Plain Tree

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:acs="http://airbus-group/aircraft-system#"
  xmlns:evt="http://airbus-group.installsys/event#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:spin="http://spinrdf.org/spin#"
  xmlns:qudt="http://qudt.org/schema/qudt#"
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:arg="http://spinrdf.org/arg#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:vaem="http://www.linkedmodel.org/schema/vaem#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:voag="http://voag.linkedmodel.org/voag/"
  xmlns:comp="http://airbus-group.installsys/component#"
  xmlns:qudt-dimension="http://qudt.org/vocab/dimension#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:iems="http://airbus-group/installationMeasure#"
  xmlns:dtype="http://www.linkedmodel.org/schema/dtype#"
  xmlns:mat="http://airbus-group/material#"
```

The links between text and model

T2M M2T Link

```
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/id270">
  <j:0:hasOffset>270</j:0:hasOffset>
  <j:0:isSameAs>http://www.linkedmodel.org/schema/vaem#id</j:0:isSameAs>
  <j:0:hasValue>id</j:0:hasValue>
</rdf:Description>
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/attach818">
  <j:0:hasOffset>818</j:0:hasOffset>
  <j:0:isSameAs>http://airbus-group/opp-function#Attach</j:0:isSameAs>
  <j:0:hasValue>attach</j:0:hasValue>
</rdf:Description>
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/attached709">
  <j:0:hasOffset>709</j:0:hasOffset>
  <j:0:isMorphologySimilarTo>http://airbus-group/opp-function#Attach</j:0:isMorphologySim
  <j:0:hasValue>attached</j:0:hasValue>
</rdf:Description>
</rdf:RDF>
```

Text & Model-Synchronized Document Engineering Platform



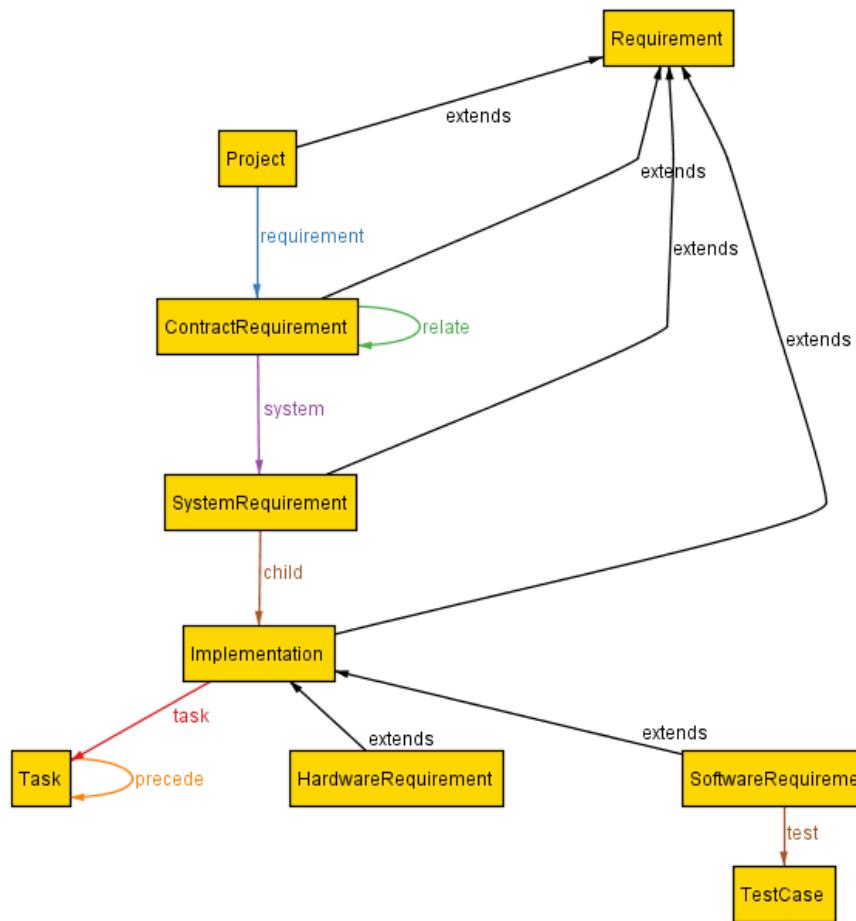
Synchronization is maintained!

What about configuration/formalization of the platform?

Configuration: Havelsan example

```

extends: 6
child: 1
precede: 1
relate: 1
requirement: 1
system: 1
task: 1
test: 1
  
```



```

module Havelsan/Requirement

abstract sig Requirement {}

sig Task {
    precede: lone Task,
    { all t: Task | one t.^precede }
}

one sig Project extends Requirement {
    requirement: some ContractRequirement
}

sig ContractRequirement extends Requirement {
    system: set SystemRequirement,
    relate: set ContractRequirement
}{all c: ContractRequirement | one c.^~requirement}

--@name: "System Requirement"
sig SystemRequirement extends Requirement {
    child: some Implementation
}{all s: SystemRequirement | one s.^~system}

abstract sig Implementation extends Requirement {
    task: set Task
}{all i: Implementation | one i.^~child}

--@context.editor: "ReqIFEditor"
sig SoftwareRequirement extends Implementation {
    test: some TestCase
}

sig HardwareRequirement extends Implementation {}

sig TestCase { }{ all t: TestCase | one t.^~test}

fact noSelfRelation{
    no c: ContractRequirement | c in c.relate
    no t: Task | t in t.^precede
}

fact noCycles{no t: Task | t in t.^precede}

fact realismConstraint {
    some ContractRequirement
    some HardwareRequirement
    some SoftwareRequirement
    some precede
}
  
```

A Formal Specification Model to configure the ModelWriter

Is it possible to vizualize the trace links?



Traceability: Havelsan example

The screenshot shows a software interface for traceability and specification management. On the left, a 'Traceability Virtualization' window displays a hierarchical requirement structure. At the top is a 'Project' node, which branches into three 'ContractRequirement' nodes: 'ContractRequirement2', 'ContractRequirement0', and 'ContractRequirement1'. 'ContractRequirement1' has a green arrow labeled 'system' pointing to a 'SystemRequirement' node. This 'SystemRequirement' node has three child nodes: 'SoftwareRequirement2', 'SoftwareRequirement0', and 'SoftwareRequirement1'. 'SoftwareRequirement1' has a blue arrow labeled 'task' pointing to a 'Task1' node, which in turn has a red arrow labeled 'precede' pointing to a 'Task0' node. A status bar at the bottom left indicates: 'child: 3', 'precede: 1', 'requirement: 3', 'system: 1', and 'task: 1'.

Customer Requirements Specification.md

- # Customer Requirements Specification
- ## UC-1 Create a new SpecObject

Note that the Specification Editor is the main interface for users. Therefore, creating SpecObjects in this editor is the main success scenario.

- ### Precondition
- ReqIF model exists and is open.
- ### Main Success Scenario
- 1. We assume that a Specification exists and is open (not required for alternative scenario)
- 2. Open a row's context menu (or in the empty editor space)
- 3. Select the Child or Sibling submenu.
- 4. Select the desired Spec Object Type (or none) from the submenu.
- 5. This results in a new SpecHierarchy being created that is linked to a newly created SpecObject with the correct type.

Alternative 1: Create in Outline

ModelWriter Source Mapping View

ID	Text
eff099c7-9ad5-4b4d-ac4d-5253d0594...	Customer Requirement

Markers ModelWriter Master View

ID	Text
	SpecObject [T: Task]

Problems ModelWriter Target Mapping View

ID	Text

Properties

A Formal Specification Model to configure the ModelWriter

6 Summary of the Current Status

Ferhat Erata

UNIT, ModelWriter Project Leader

Several Achievements

- Exploitation of ModelWriter in ITEA3-ASSUME
 - New system (Exploitation)
- Specification & Verification of ALM Platform
 - Open Source Software (Standardisation)
- Change Impact Analysis & Visualization
 - Open Source Software (Standardisation)
- System Installation Component Ontology
 - De facto standard (Standardisation)
- Semantic Annotator
 - Open Source Software (Standardisation)
- CSV to OWL transformation program
 - New product (Exploitation)
- SIDP Installation Rule Model
 - New product (Exploitation)

Summary of the first year

- We have a clear project structure and objectives.
- We positioned new industrial partners.
- We managed to restore the consortium with early changes in the leaderships
- We have still the same ambition.
- We have end users, clear needs; have enough tool & technology providers
- We have already significant Exploitation Related Achievements
- We have developed core ModelWriter
 - Knowledge Capture & Knowledge extraction
- All software components are platform independent and open source

**Thank you for your attention
We value your opinion and
questions.**