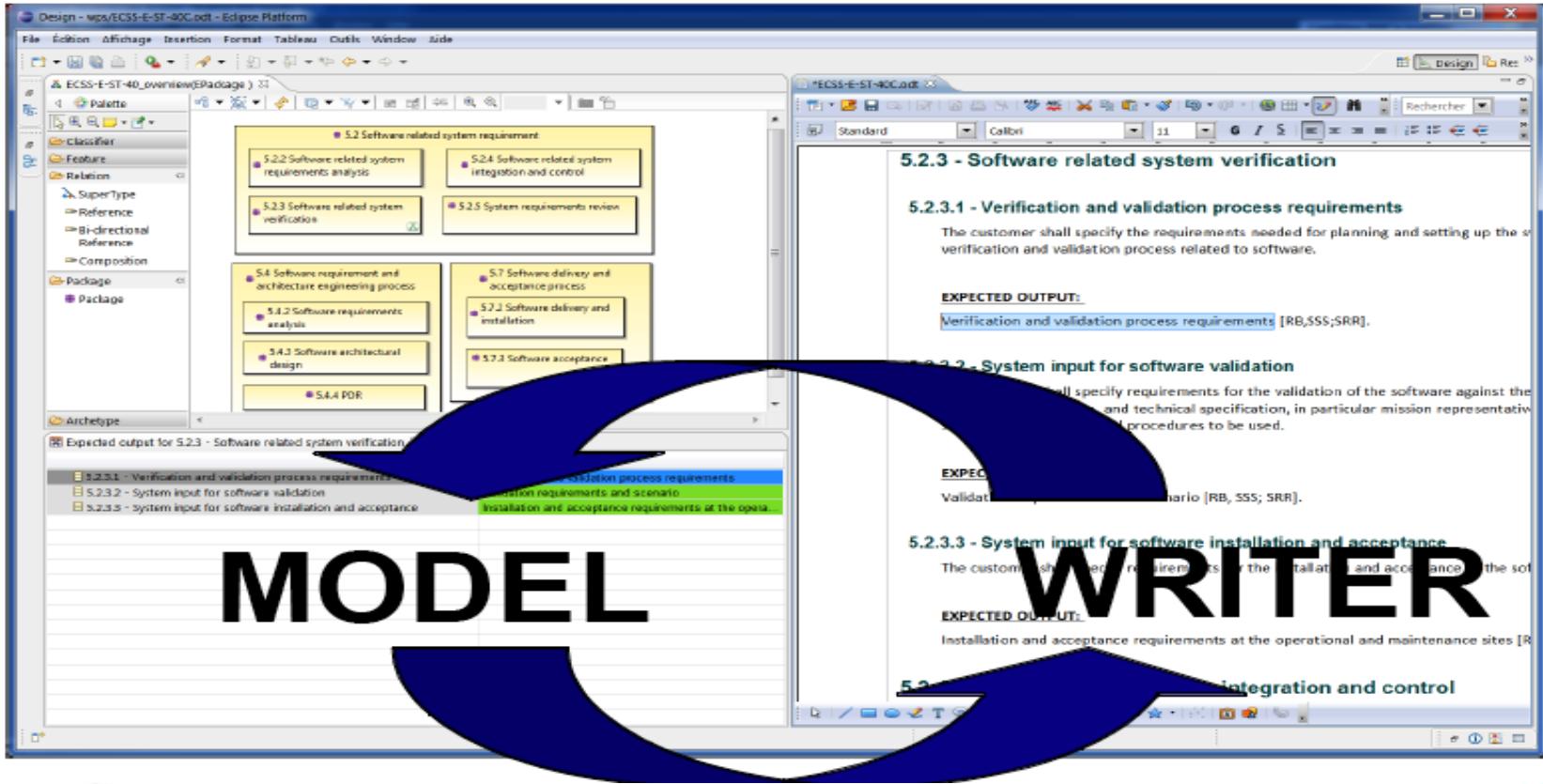


ModelWriter

Text & Model-Synchronized Document Engineering Platform



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

Project Email: project@modelwriter.eu

Revised Version of Eureka-ITEA 1st Review Meeting for Turkish Reviewers

Introduction and Agenda

Ferhat Erata

UNIT, ModelWriter Project Leader

Participants

ModelWriter 2014-2/2015-1 Review



UNIT

- Ferhat Erata
- Dr. Moharram Challenger



- Doç. Dr. Geylani Kardaş
- Mehmet Önat

Mantis

Software Company

- Dr. Güven Köse
- Yard. Dr. Erhan Mengüsoğlu



- Ersan Gürdoğan



- Dr. Eray Tuzun
- Yagup Çetin



- Dr. Emrah Kınav
- Yasir Tuncer

Agenda - Doç.Dr. TANSEL ÖZYER

1. Overview of the project (25min) [09:45 - 10:10]
2. Progress Status per Work Package (30 min) [10:10 - 10:40]
3. Use-cases and exploitation prospects (30 min) [10:40 - 11:10]
4. Demonstrations of the Software Deliverables (30 min) [11:10 - 11:40]
5. Summary (achievements and exploitations) (5 min) [11:40 - 11:45]
6. ITEA -EUREKA Review Meeting Conclusions & Actions
7. Financial Assessments of Turkish Consortium
8. Questions

1 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

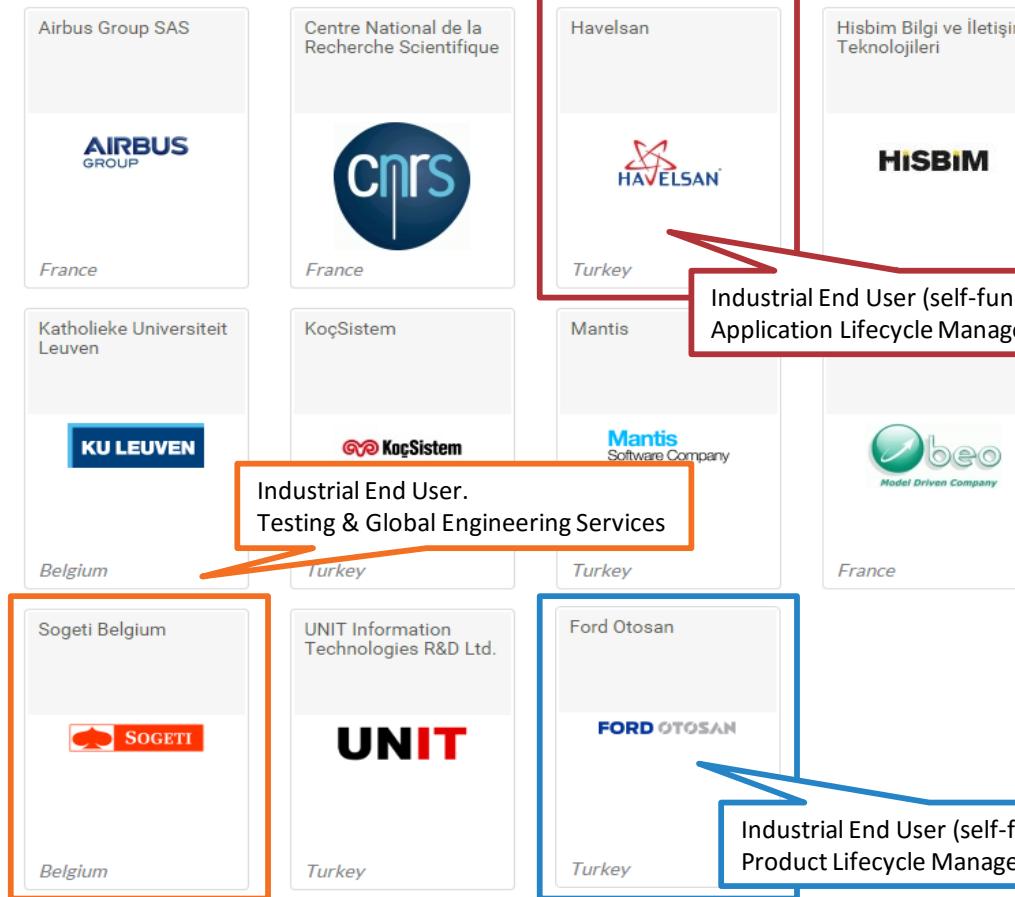
Project name	13028 ModelWriter
Status	Running
Period	Oct 2014 - Sep 2017
Call	ITEA 2 Call 8
Challenge	Knowledge-based society
Website	www.modelwriter.eu
Partners	10
Countries	Belgium France Turkey

Project leader



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

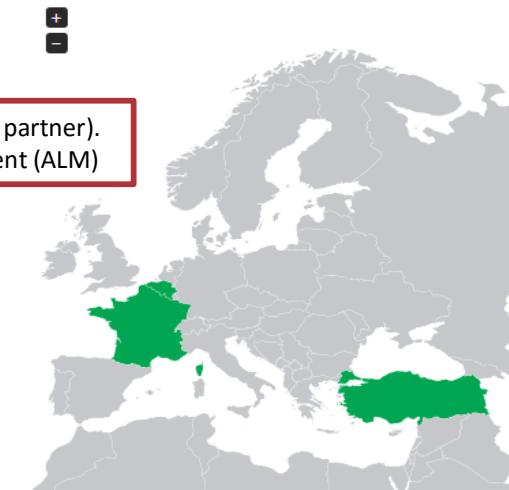
Project partners



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

Industrial Use Cases
Success Stories
Long Term Agreements

Large
Organization

Inject
Requirements

SME

Industrialization

ModelWriter

Technology
Transfer

Prototypes

Researchers

Innovation

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

AIRBUS
GROUP



KoçSistem

SOGETI

HISBIM

FORD OTOSAN



Resource Allocation: 68,71 person year

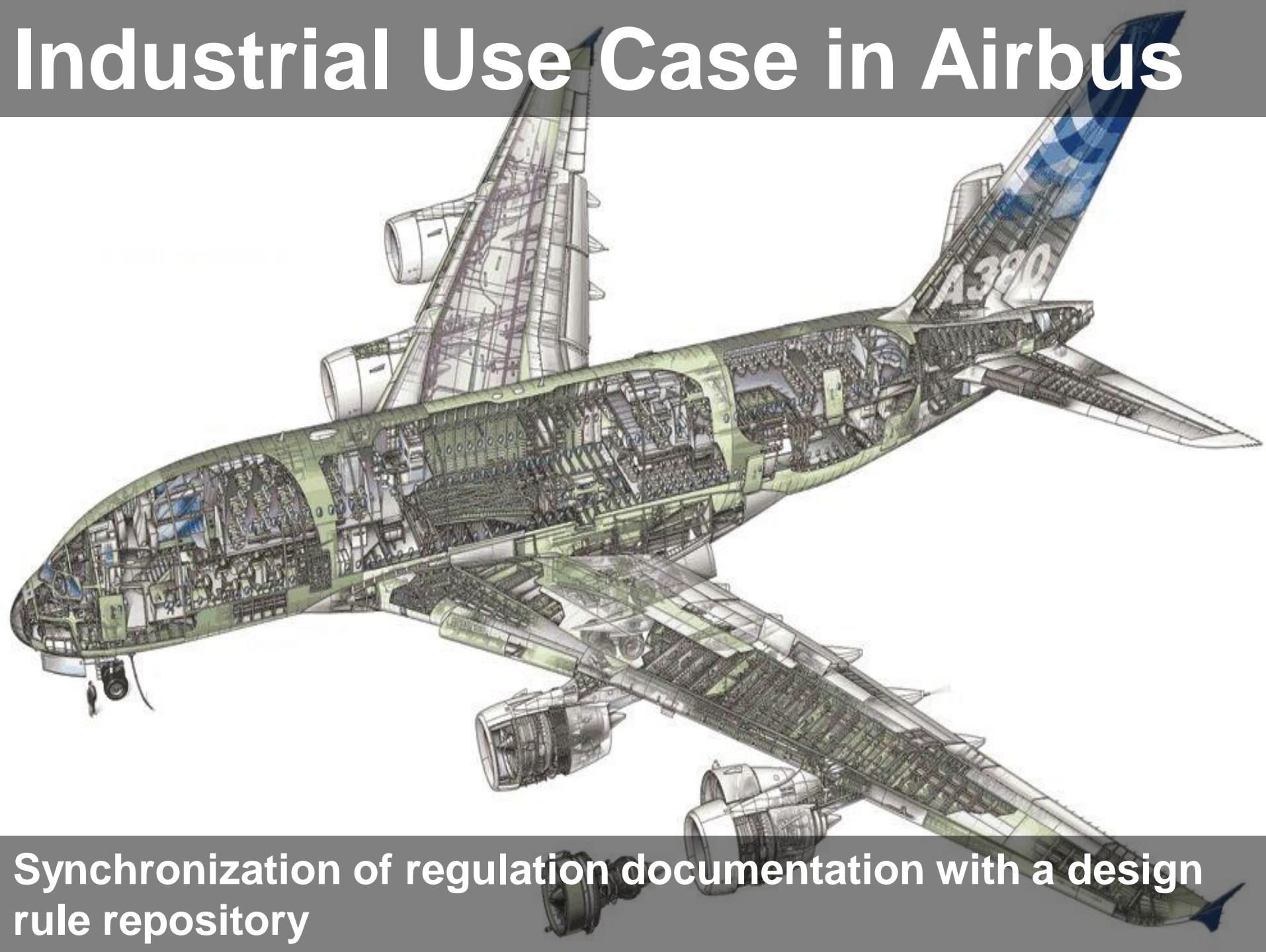
Project Duration: 36 months

Planned Budget: 5,543,000 Euro

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

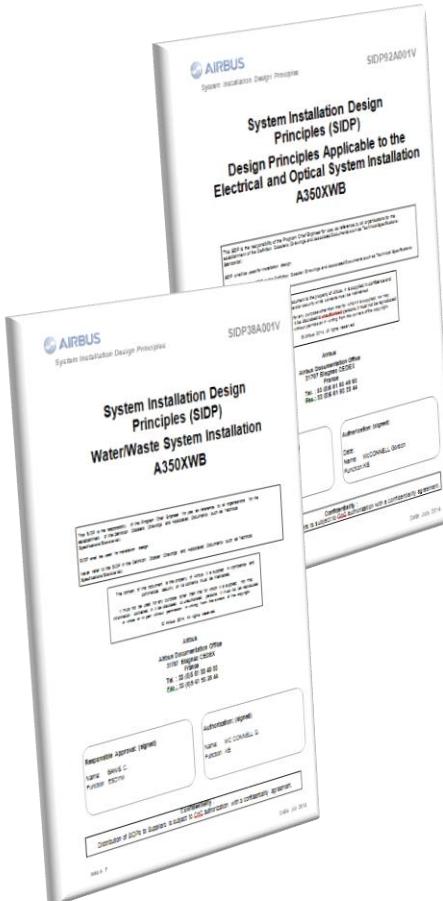
Open-Source Software Platform to be submitted to Eclipse Foundation

Industrial Use Case in Airbus



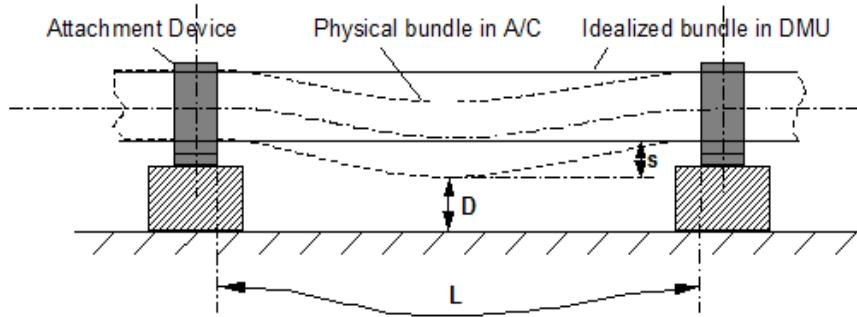
Synchronization of regulation documentation with a design rule repository

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.)

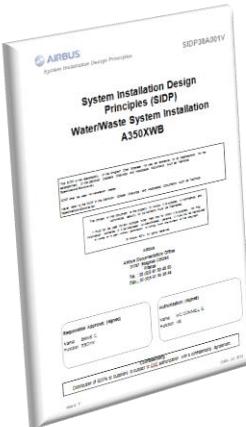
D... Required Distance

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



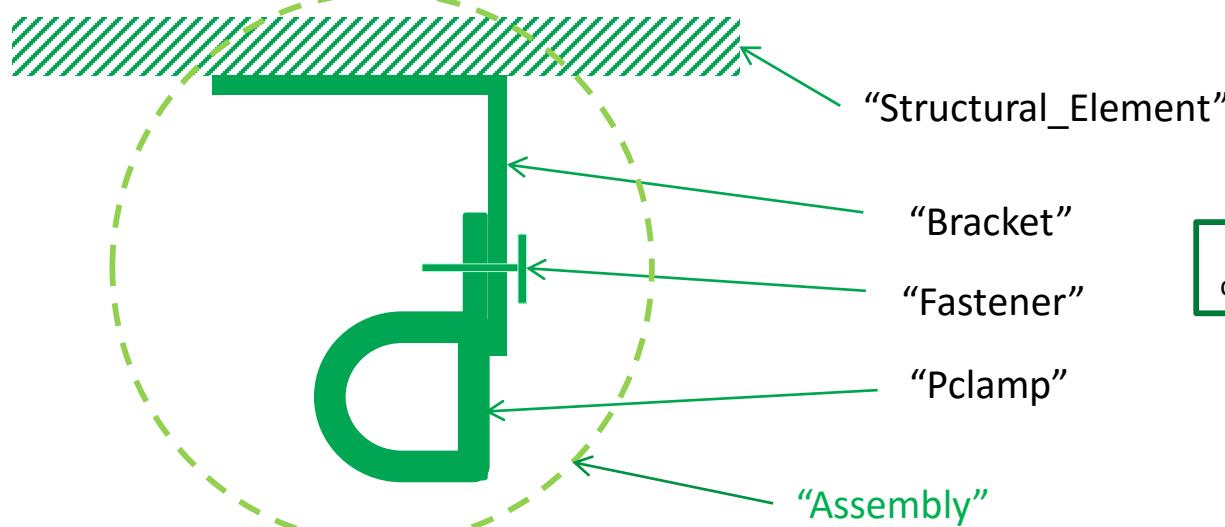
- SIDP documents explain how to install the aircraft systems and attach them to the structure. They capitalize the best practices & proven technical solutions.
- SIDP are defined per **ATA chapter** (~functional domain) to be applied for **each given A/C project**: for example "**A350 Electrical installation**"
- 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.
- **In industrial context of AIRBUS, SIDP are edited using MSWord**

Industrial high level needs

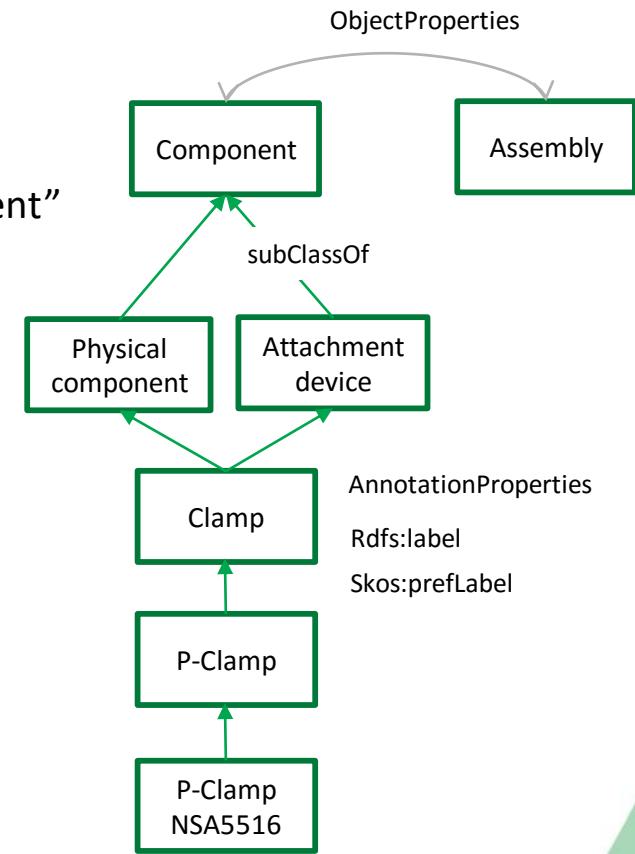
To improve SIDP creation, maintenance and consultation in order to:

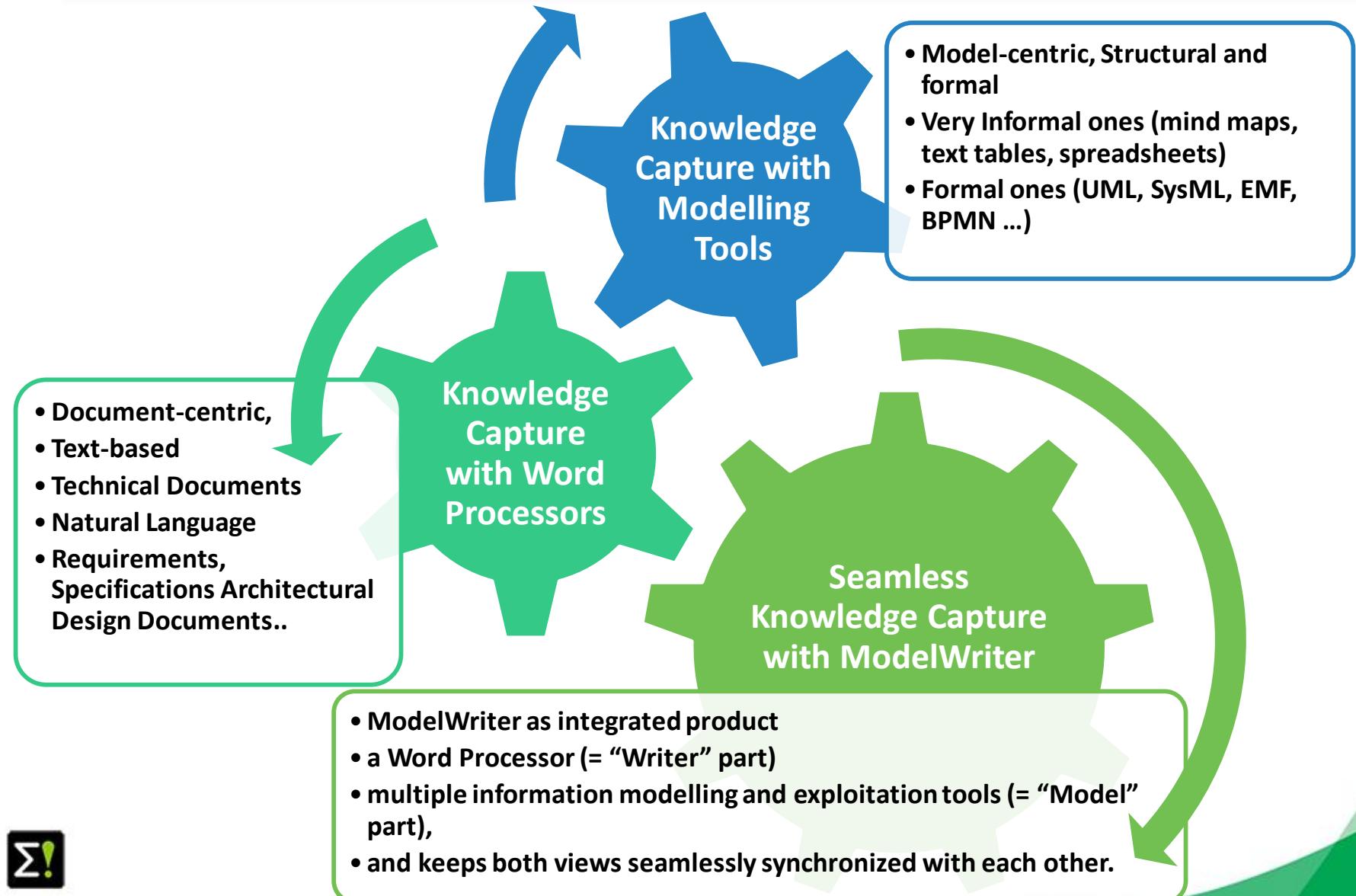
- save costs by supporting DP consultation / retrieval in accordance to the needs
- avoid non-compliance of design caused by DP updating lead times
- keep traceability with upstream regulations and requirements and downstream design models

Component classes taxonomy



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





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.odata - 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.odata" 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

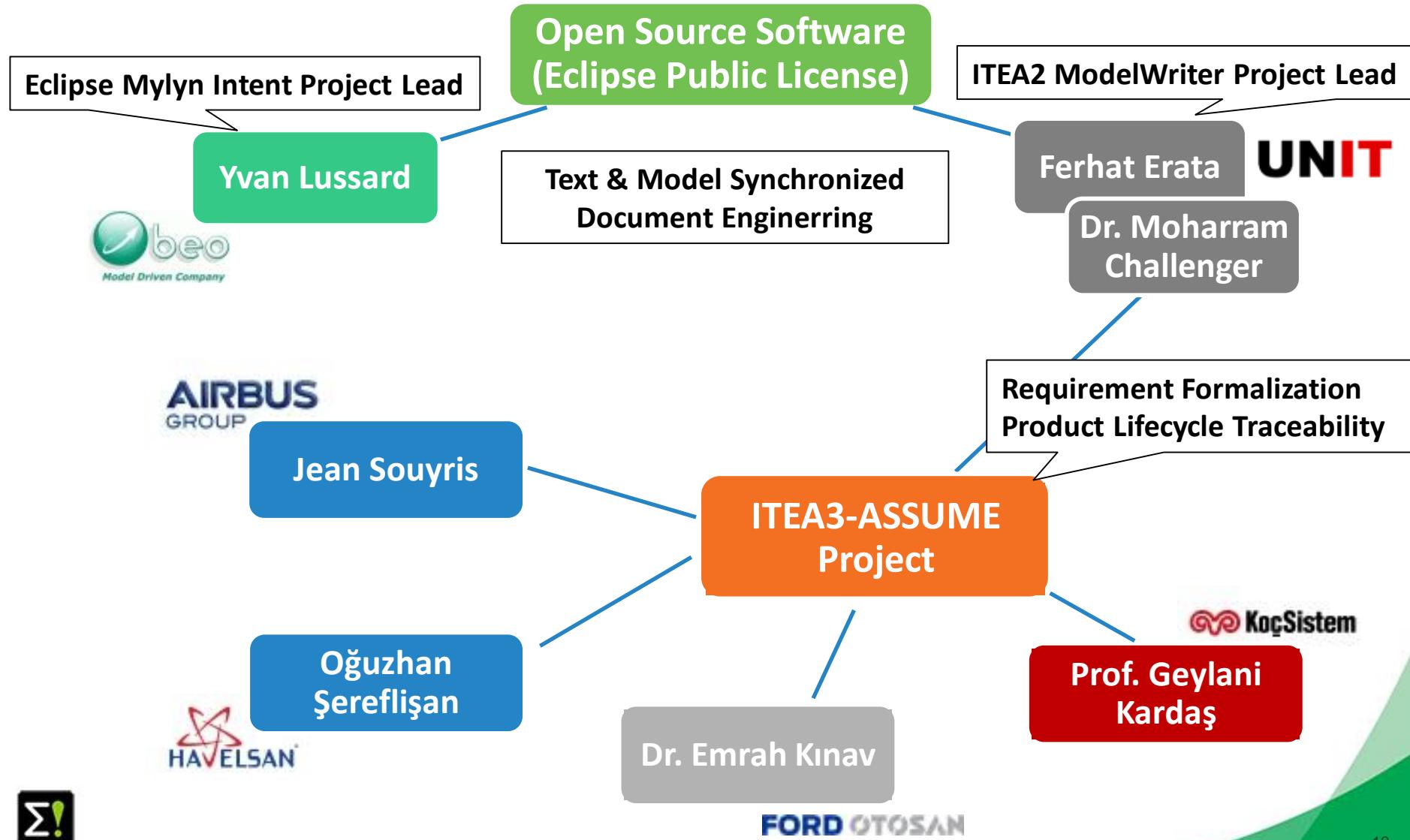
Consistency Checking: the objective to provide a Consistency Checker that automates Quality Reviews of Requirements Engineering

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)

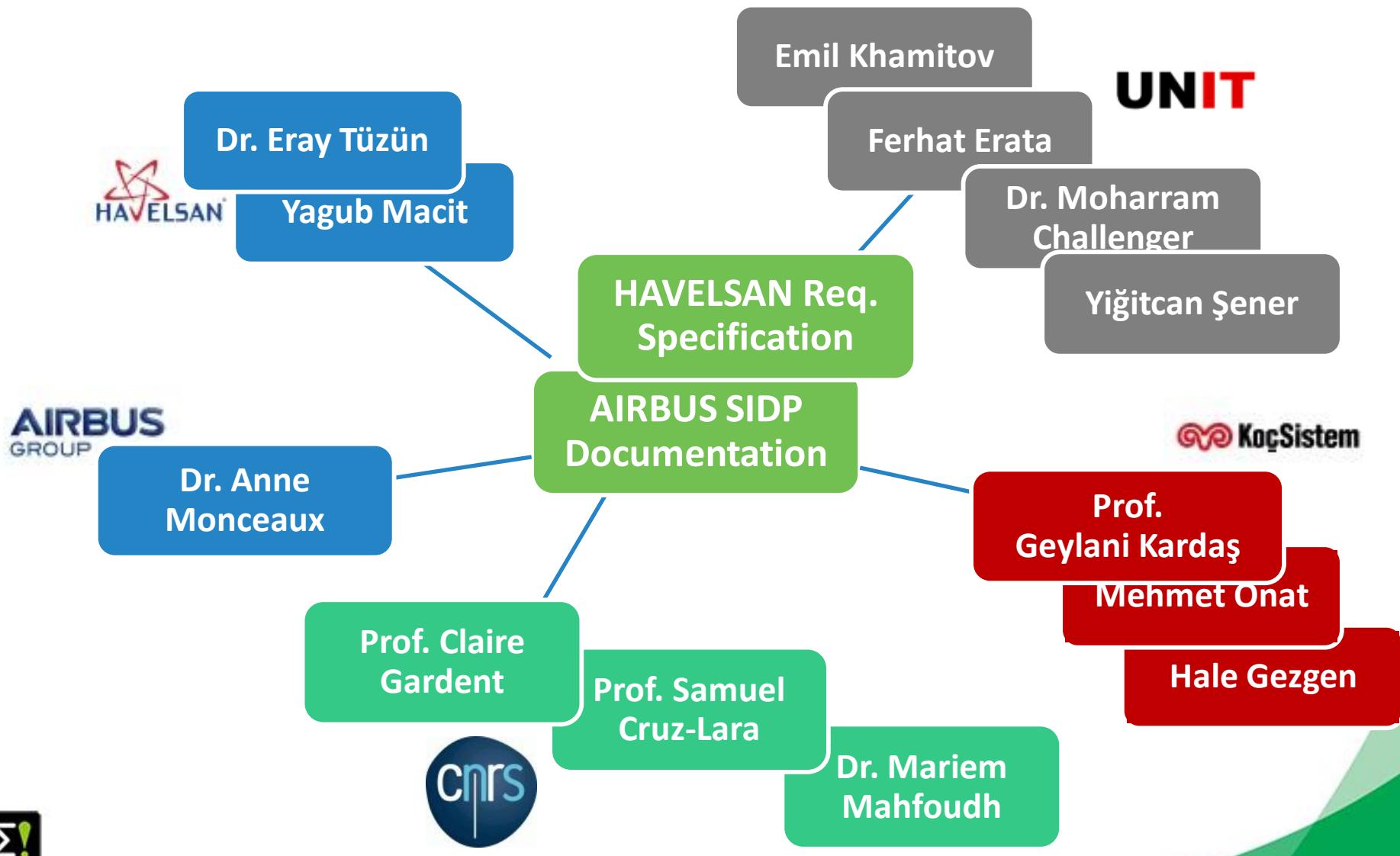
Level of Collaboration within ModelWriter

Exploitation of MW in Eclipse & ASSUME



Level of Collaboration within ModelWriter

International Collaboration (through UCs)

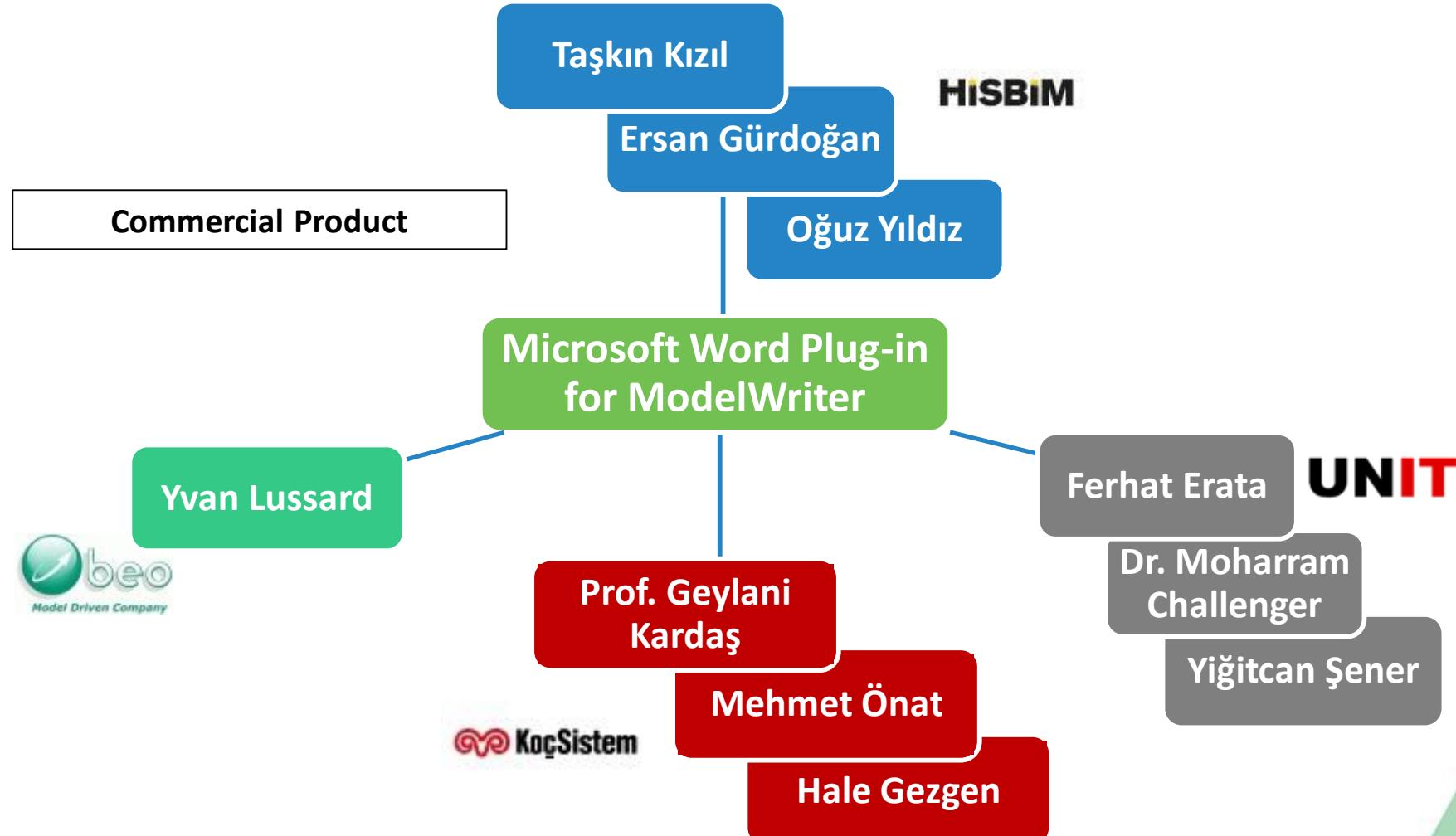


Level of Collaboration within ModelWriter

International Collaboration (through plugins)

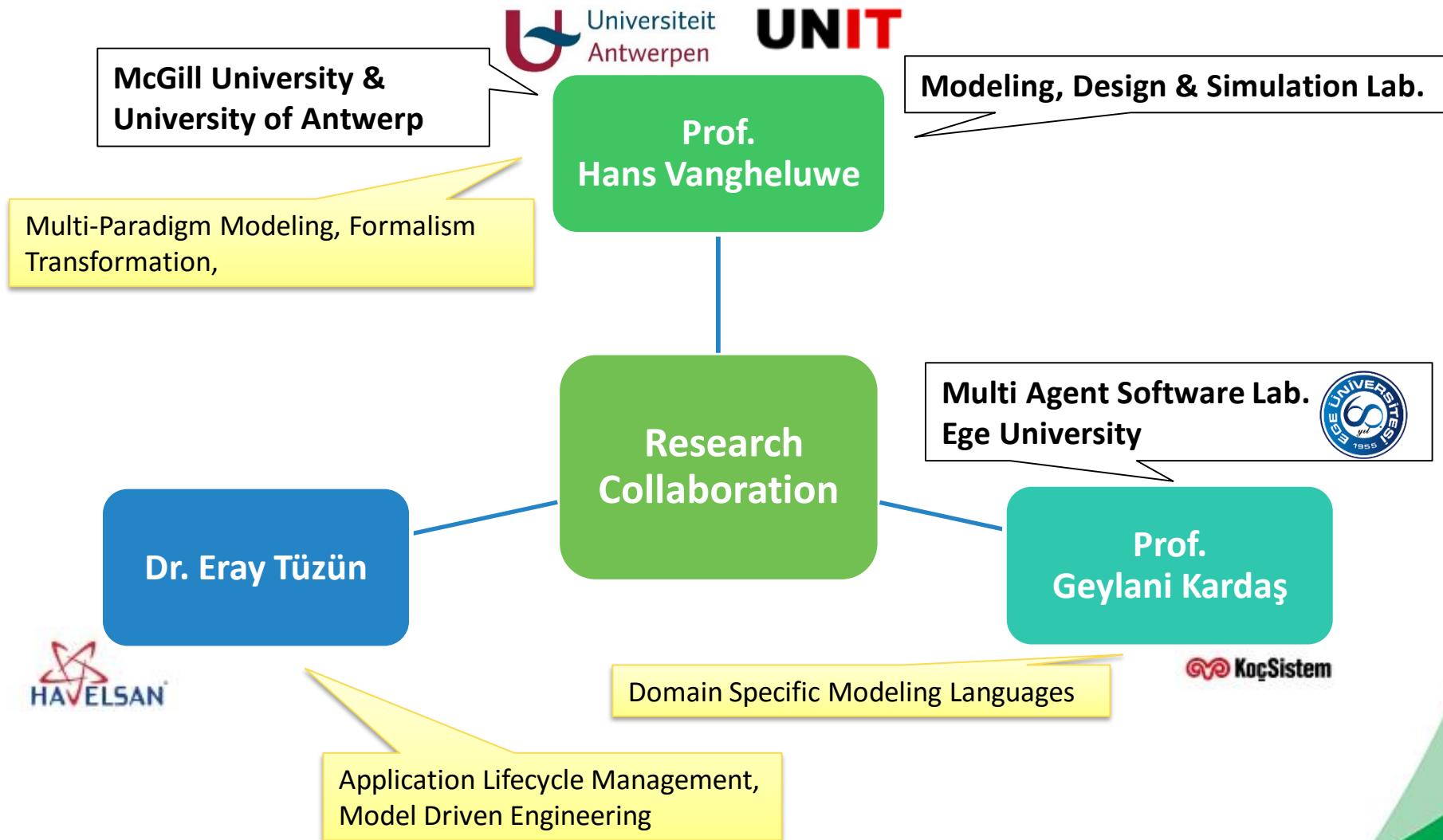


ITEAS



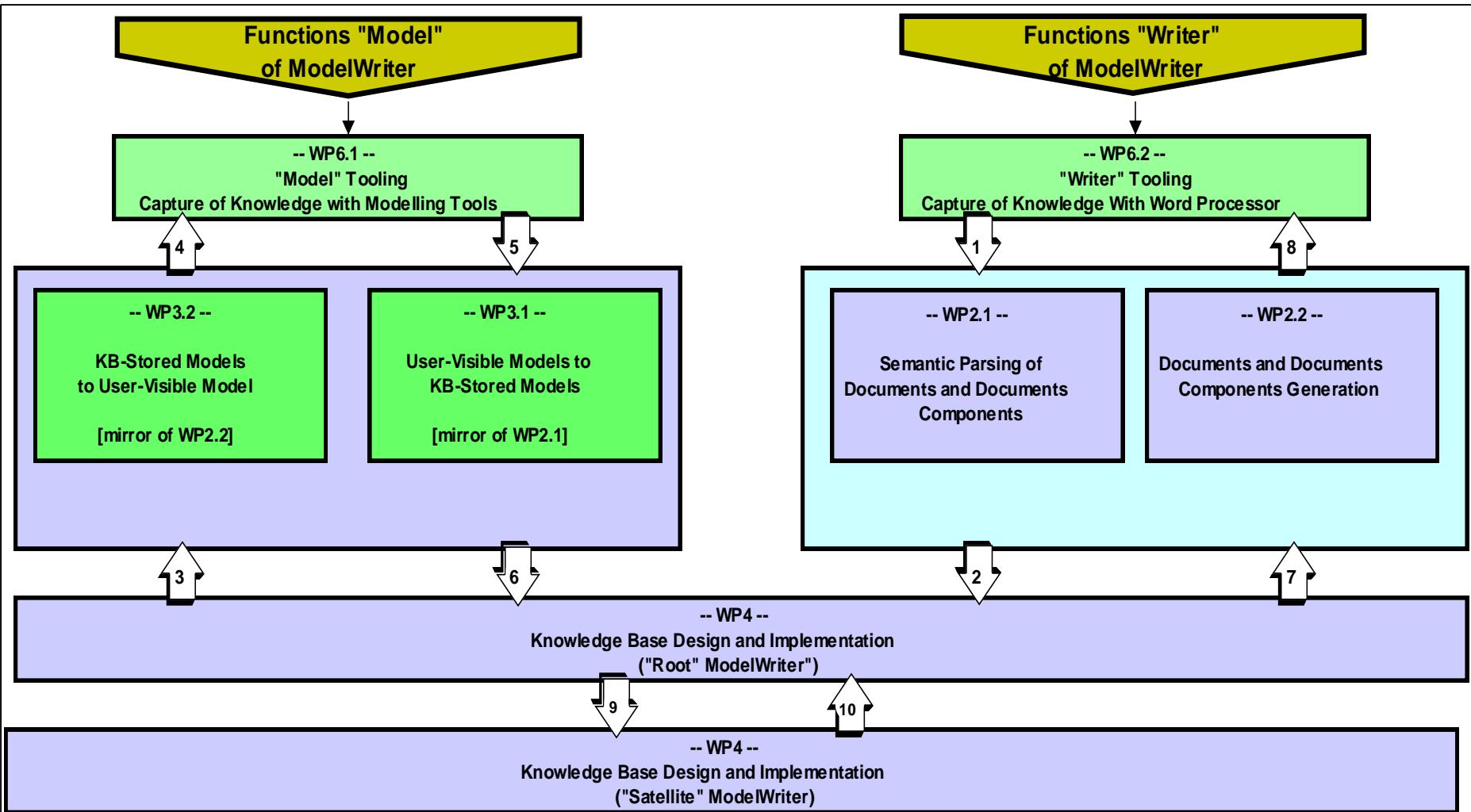
Level of Collaboration within ModelWriter

International Collaboration (research)



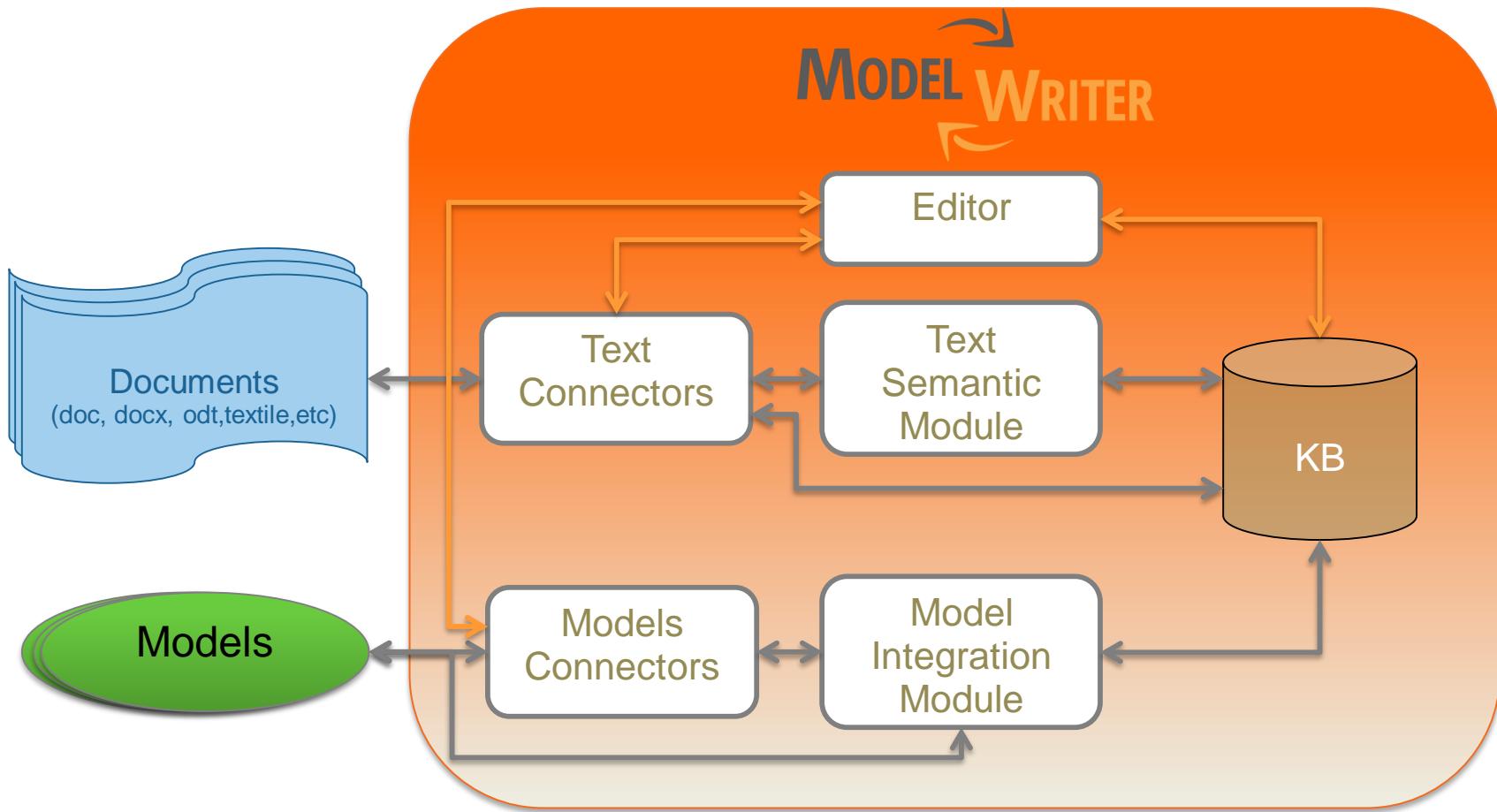
Technological components & interactions

Collaboration by WP interactions



ModelWriter

Conceptual Architecture

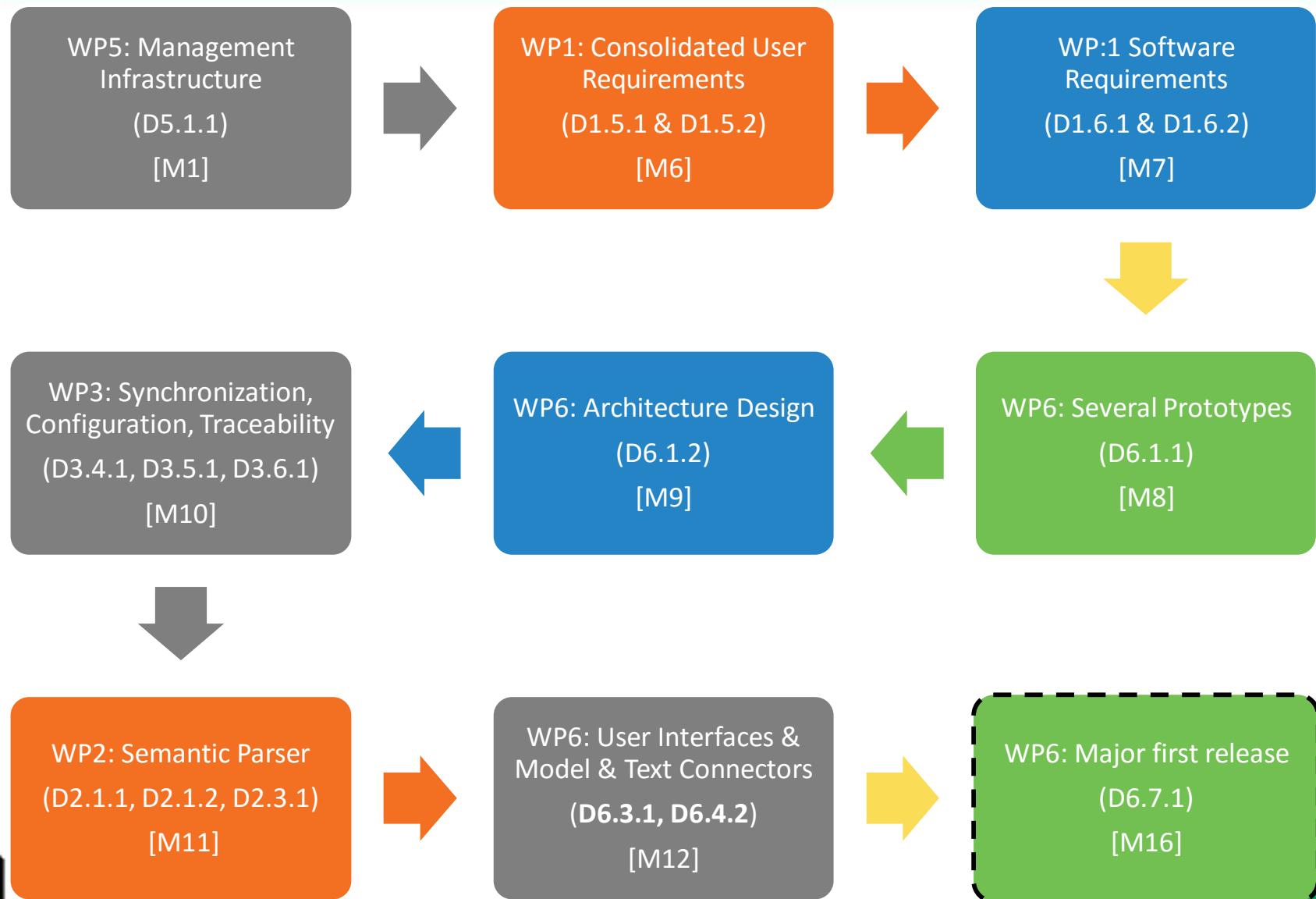


WP1 Industrial Use Cases and Requirements (AIRBUS)

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 1st Year



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

Organized by KoçSistem

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]

Organized by UNIT

Exploitation: Havelsan's participation

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

Consolidated User Requirements & Review

Organized by Eureka-ITEA Office

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

Organized by Sogeti

Software Requirements Review & Architecture

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

Organized by Eclipse Foundation

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

Exploitation: FordOtosan's participation

Organized by Eureka-ITEA Office

ModelWriter Workshops in the 2nd Year



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

Exploitation in ITEA-3 project

ITEA3 Assume-Project Kick-off in Berlin, Germany (Oct 1-2 2015) [M13]

Organized Daimler & Bosch

Hosted by Eclipse Foundation

The 5th International ModelWriter Workshop in Ludwigsburg, Germany (Nov 2-5, 2015) [M14]

Integration of Components

The 6th International ModelWriter Workshop in Paris, France (Feb 15-16, 2016) [M17]

Product Owner Review Meeting

Organized by Airbus SAS

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 | [https://itea3.org/project/...](https://itea3.org/project/) | project@modelwriter.eu

Repositories | **People 25** | **Teams 27** | **Settings**

Filters | | **+ New repository**

WP3
Work Package 3 - Model to/from Knowledge Base (UNIT)
Updated 15 minutes ago

WP5
Work Package 5 - Project Management (UNIT)
Updated 22 hours ago

Ferhat PRIVATE
Updated a day ago

Workshops
Repository dedicated to workshops
Updated 5 days ago

People 25 >

A grid of 25 user profile pictures and corresponding GitHub icons. The icons include various symbols like a brain, a castle, a person, and a star.

Invite someone



ModelWriter Requirements & User Stories

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



ModelWriter/Requirements

Add Issue

User Stories 73 100

Confirmed 29 8

In Progress 0

Done 2

Filter Board

116 The system must support the specification of the connector to the document

Functional WFE - ModelWriter Architecture Integration and Evaluation Software Requirements Document (SRD)

115 Generator

Functional Mandatory Reversible Semantic Engine Software Requirements Document (SRD) WP2 - Semantic Parsing and Generation of Documents

114 Parser

Functional Mandatory Reversible Semantic Engine Software Requirements Document (SRD) WP2 - Semantic Parsing and Generation of Documents

113 Grammar-Extractor

Desirable Functional Reversible Semantic Engine Software Requirements Document (SRD) WP2 - Semantic Parsing and Generation of Documents

112 Grammar

Functional Mandatory Reversible Semantic Engine Software Requirements Document (SRD) WP2 - Semantic Parsing and Generation of Documents

111 Lexicon-Extractor

Desirable Functional Semantic Annotator Software Requirements Document (SRD) WP2 - Semantic Parsing and Generation of Documents

110 Lexicon

Confirmed 29 8

20 The System shall propose an eclipse editor to handle documentation/models mappings

Mandatory UC-FR-01 Sync. between Models and Documentation User Requirements Document (URD)

52 Text-Based Knowledge Extraction with ModelWriter (Semantic Word Processor)

Mandatory ModelWriter's Feature from FPP User Requirements Document (URD)

29 The system shall be easy to integrate in an RCP application

Mandatory UC-FR-02 Enterprise Architecture UC-TR-04 Requirements Engineering with SysML Designer User Requirements Document (URD)

33 The System shall not enforce any dependency on non-open source artifacts such as tools, applications, and libraries.

Mandatory UC-FR-01 Sync. between Models and Documentation User Requirements Document (URD)

25 The system should allow to filter synchronization warnings, errors and information.

Mandatory UC-FR-01 Sync. between Models and Documentation User Requirements Document (URD)

31 The system shall allow the user to activate/deactivate a synchronization direction

Mandatory UC-FR-02 Enterprise Architecture User Requirements Document (URD)

47 The user does not want to be bothered with information such as mapping links.

Mandatory UC-FR-01 Sync. between Models and Documentation

In Progress

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

30 The system shall allow the users to work in a collaborative manner

Mandatory UC-FR-02 Enterprise Architecture User Requirements Document (URD)

53 ModelWriter as a Next Generation Requirements Engineering Tool: ModelWriter should be equipped with Requirements Engineering features.

Mandatory ModelWriter's Feature from FPP UC-BE-01 Requirements IT UC-TR-03 Generation and management of feature models UC-TR-04 Requirements Engineering with SysML Designer User Requirements Document (URD)



ModelWriter Requirements & User Stories

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



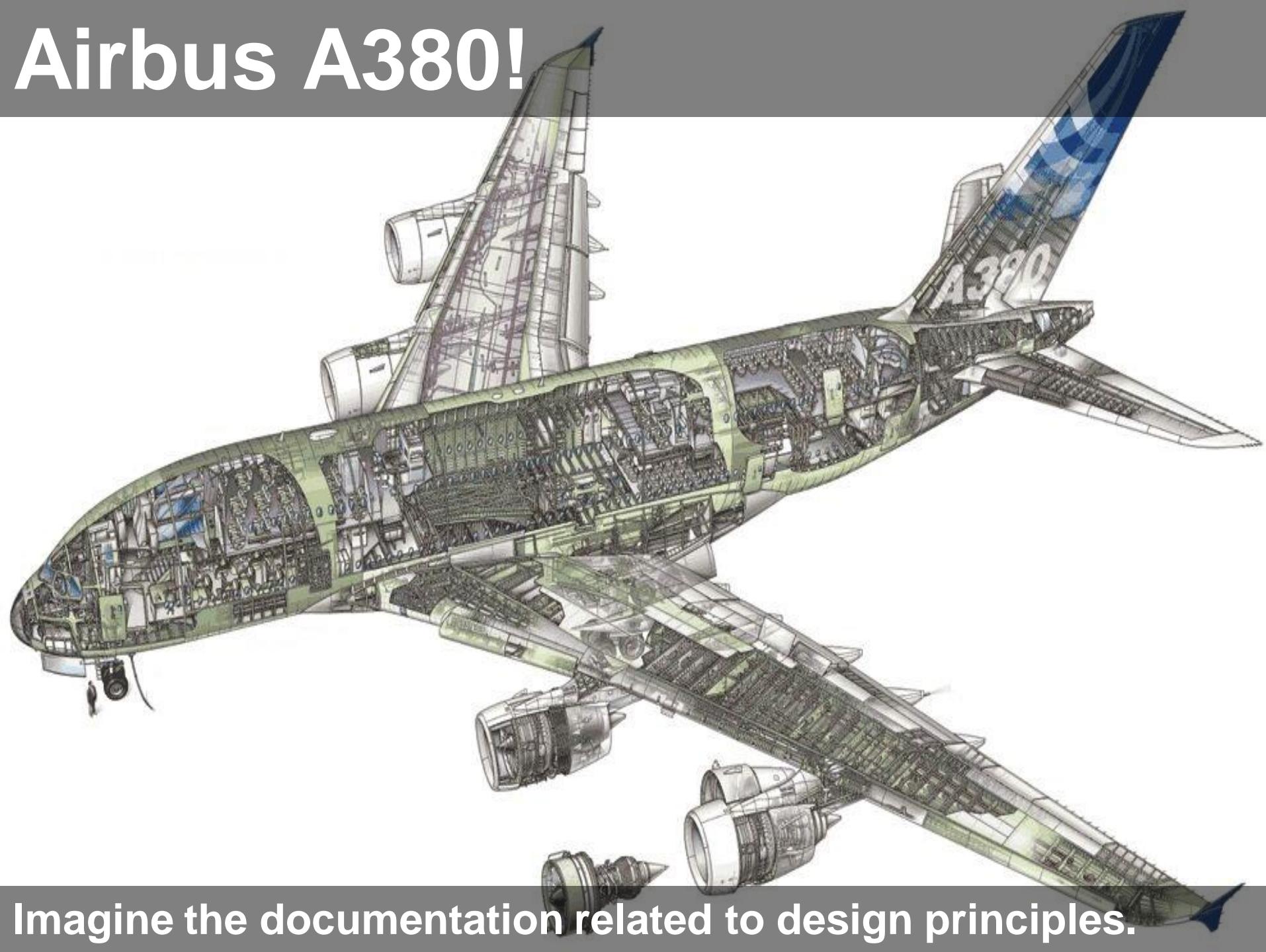
The screenshot shows a Waffle board for the project "modelwriter/wp5". The board is organized into five columns: Queue, Backlog, ToDo, In Progress, and Done.

- Queue:** Contains issues 93, 51, 50, 81, 99, 96, 95, 94, and 105. Most issues are labeled "ModelWriter" and "T5.2 - Project Coordination and Reporting".
- Backlog:** Contains issues 68, 58, 98, and 97. These are also associated with "T5.2 - Project Coordination and Reporting".
- ToDo:** Contains issues 98 and 97, both labeled "ASSUME" and "T5.2 - Project Coordination and Reporting".
- In Progress:** Contains issue 21, which is labeled "ASSUME", "T5.2 - Project Coordination and Reporting", and "UNIT".
- Done:** Contains issues 92, 91, 89, 90, 27, 52, and 84. These are associated with various labels including "help wanted", "T5.2 - Project Coordination and Reporting", "UNIT", "Management", "T5.3 - Project Controls", "T5.2 - Project Coordination and Reporting", "invalid", "Development", and "T5.2 - Project Coordination and Reporting".

What is the problem?



Airbus A380!

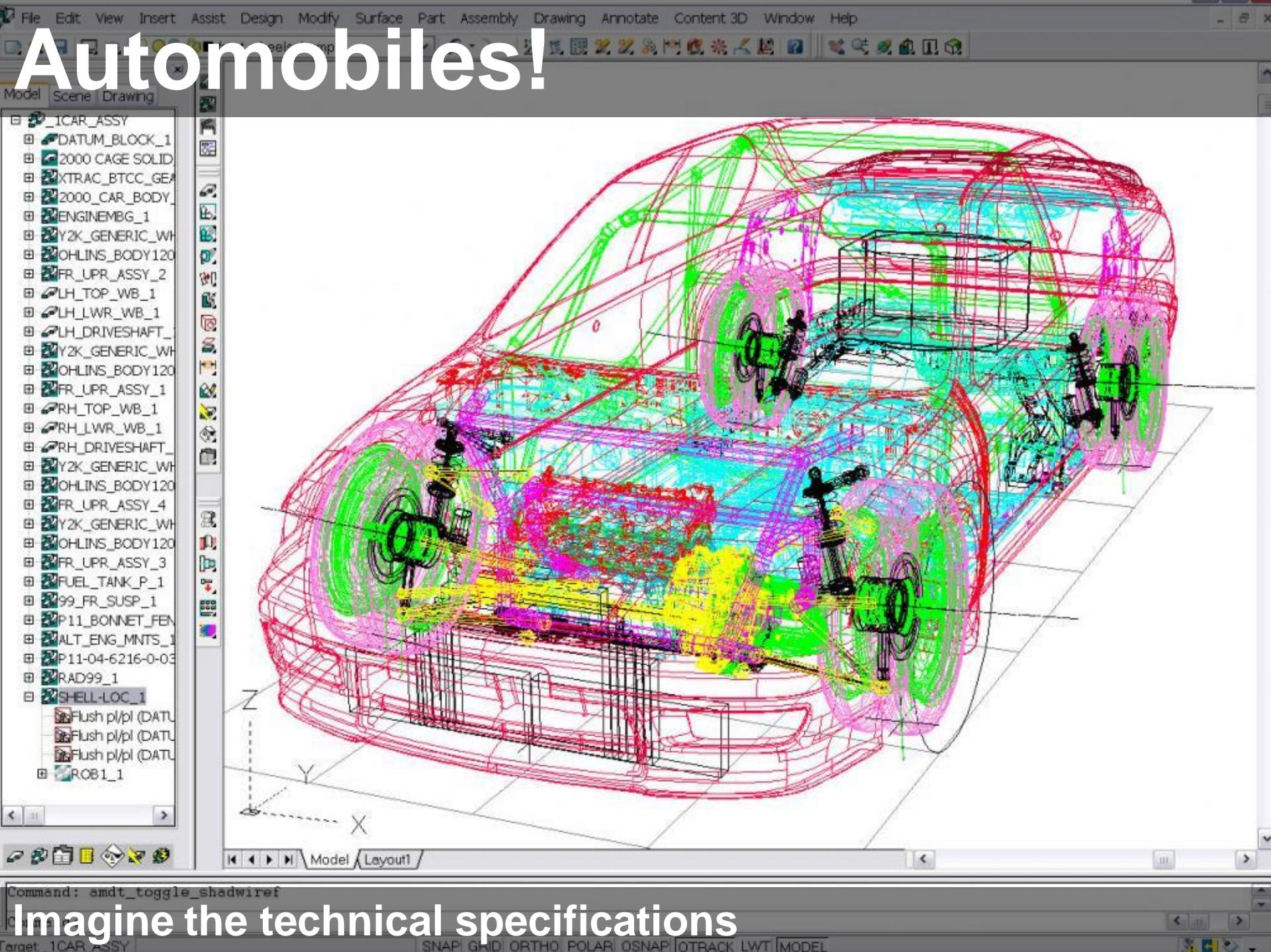


Imagine the documentation related to design principles.



Plants!

Imagine the documentation of a construction site.



Automobiles!

Imagine the technical specifications

Documentation!



Write once ... and never look at after

Documentation!



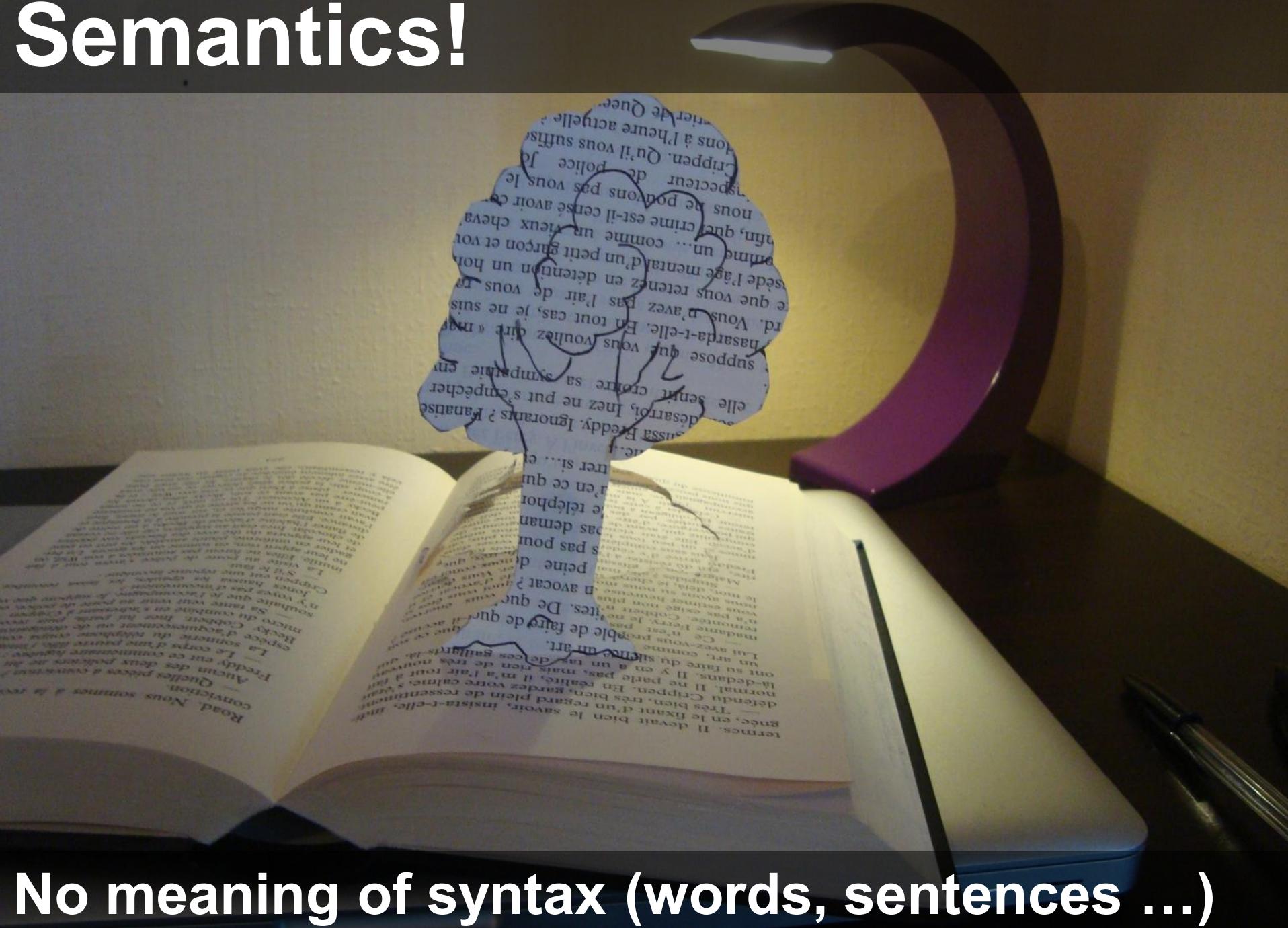
Hard to keep it up-to-date

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 a tree view of headings and pages, and a status bar at the bottom indicates "PAGE 1 OF 1" and "76 WORDS".

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 structure includes elements like <w:body>, <w:p>, <w:pPr>, <w:pStyle>, <w:bookmarkStart>, <w:bookmarkEnd>, <w:r>, <w:t>, <w:numPr>, and <w:ilvl>. The code shows a title section with a bookmark and a heading 1 section with a numbered list style. Below the XML editor, there are tabs for 'Design' and 'Source'. A toolbar with various icons for document operations is visible. At the bottom, there's a property grid showing a single entry for 'w:val' with the value 'Heading1'. The status bar at the bottom of the IDE shows 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 document structure:

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

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 ...

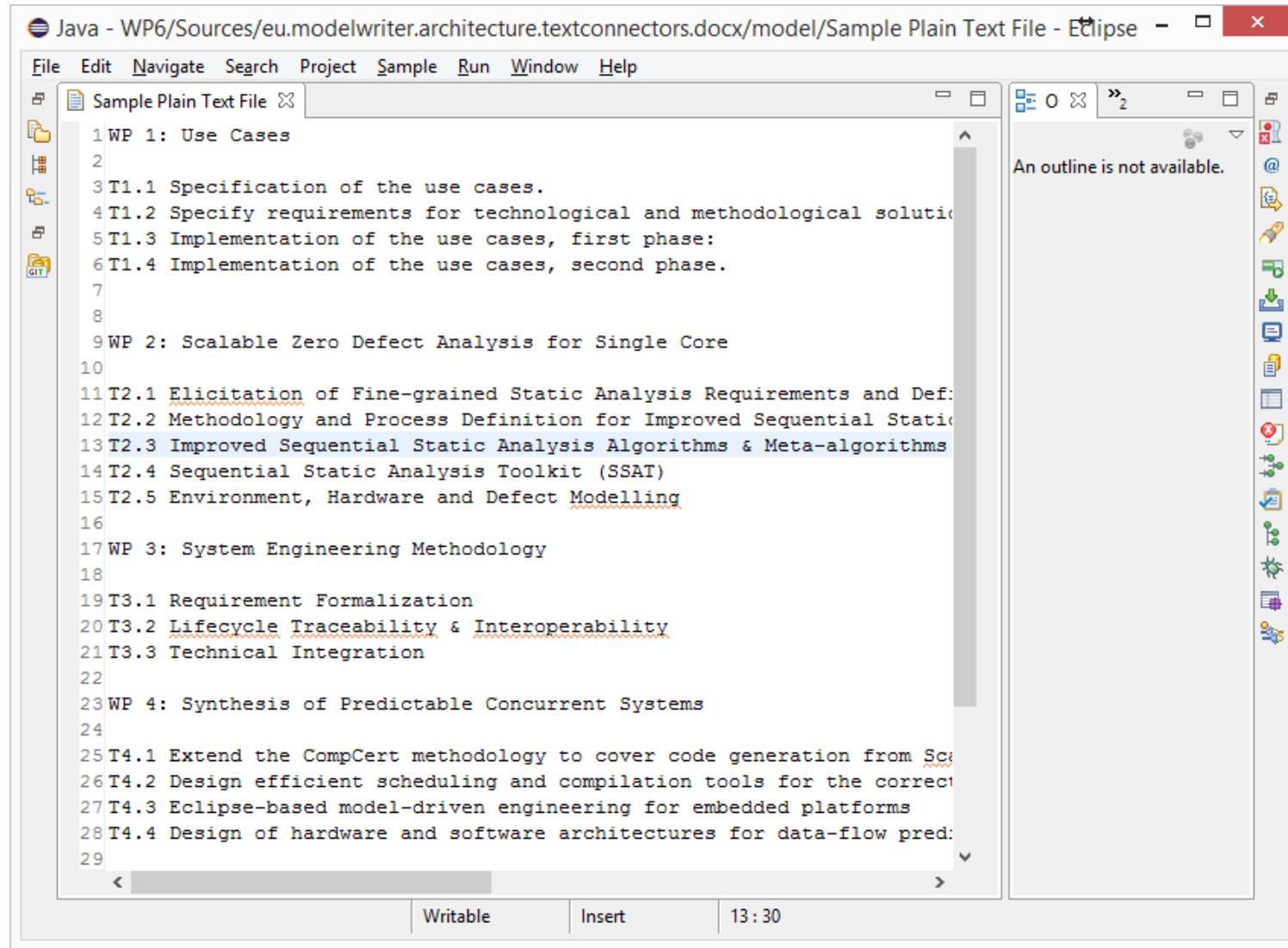
Markdown Source Preview

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)



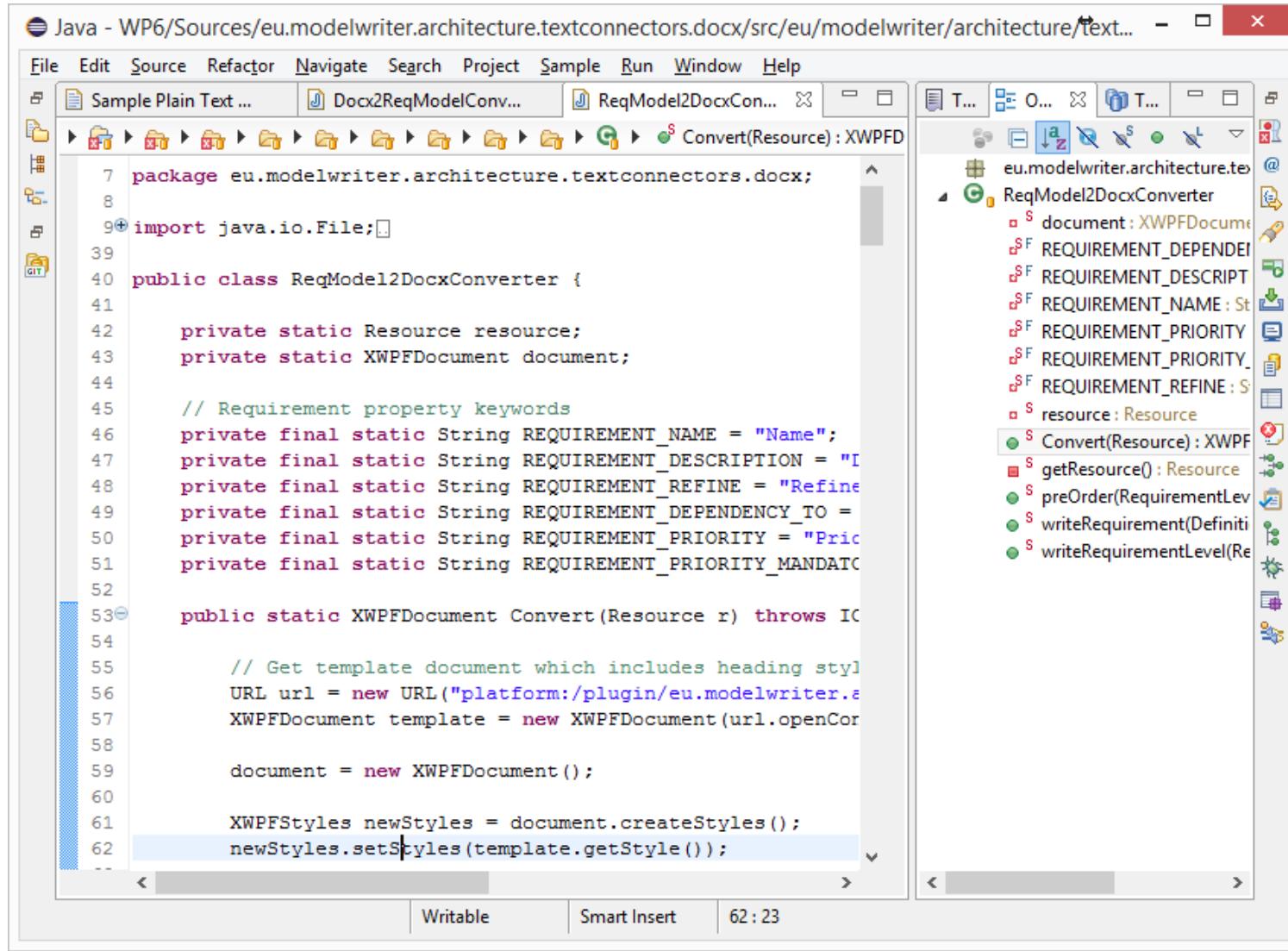
The screenshot shows the Eclipse IDE interface with a text editor open. The title bar reads "Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Sample Plain Text File - Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Sample, Run, Window, and Help. The left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

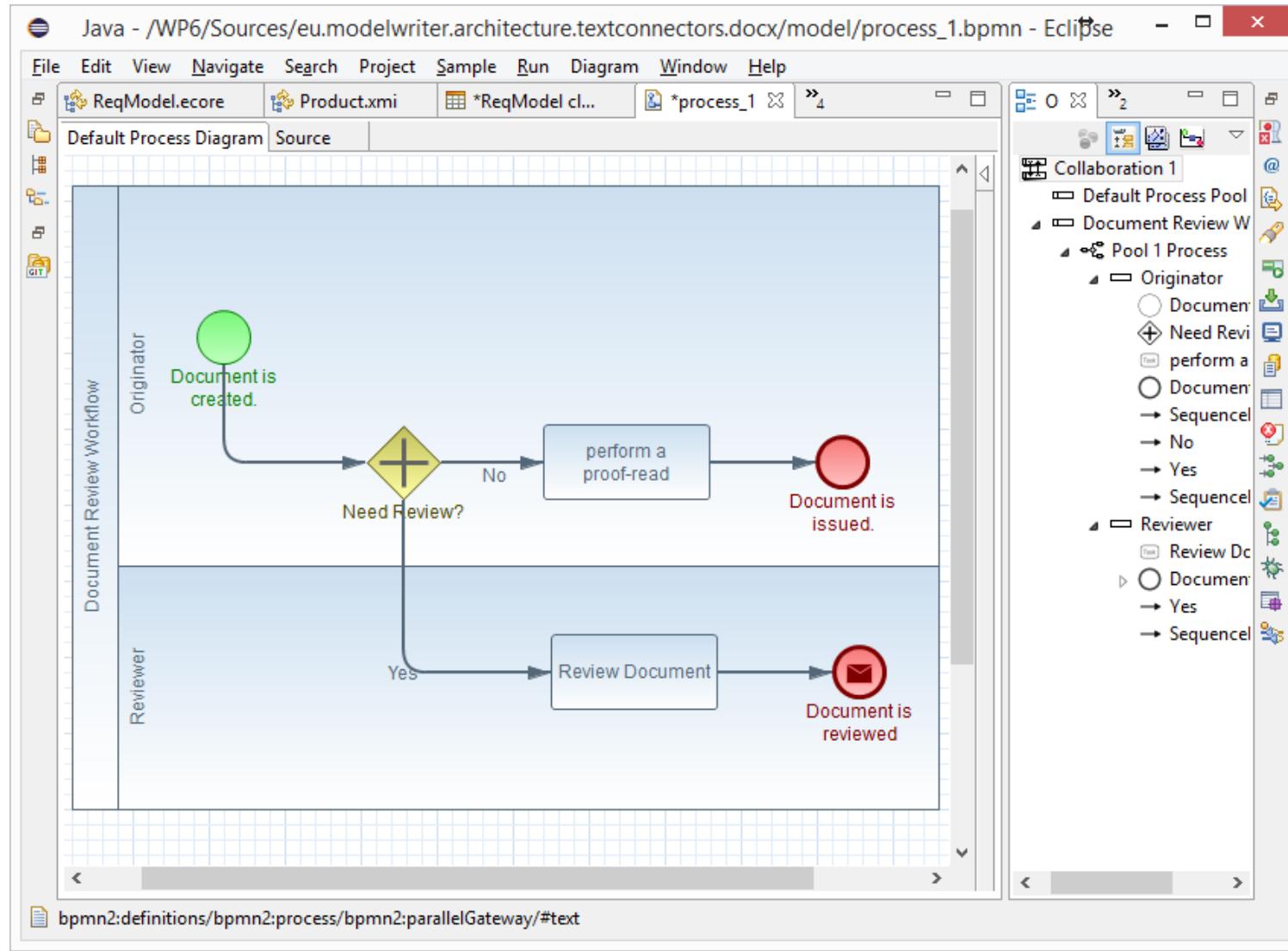
Properties

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

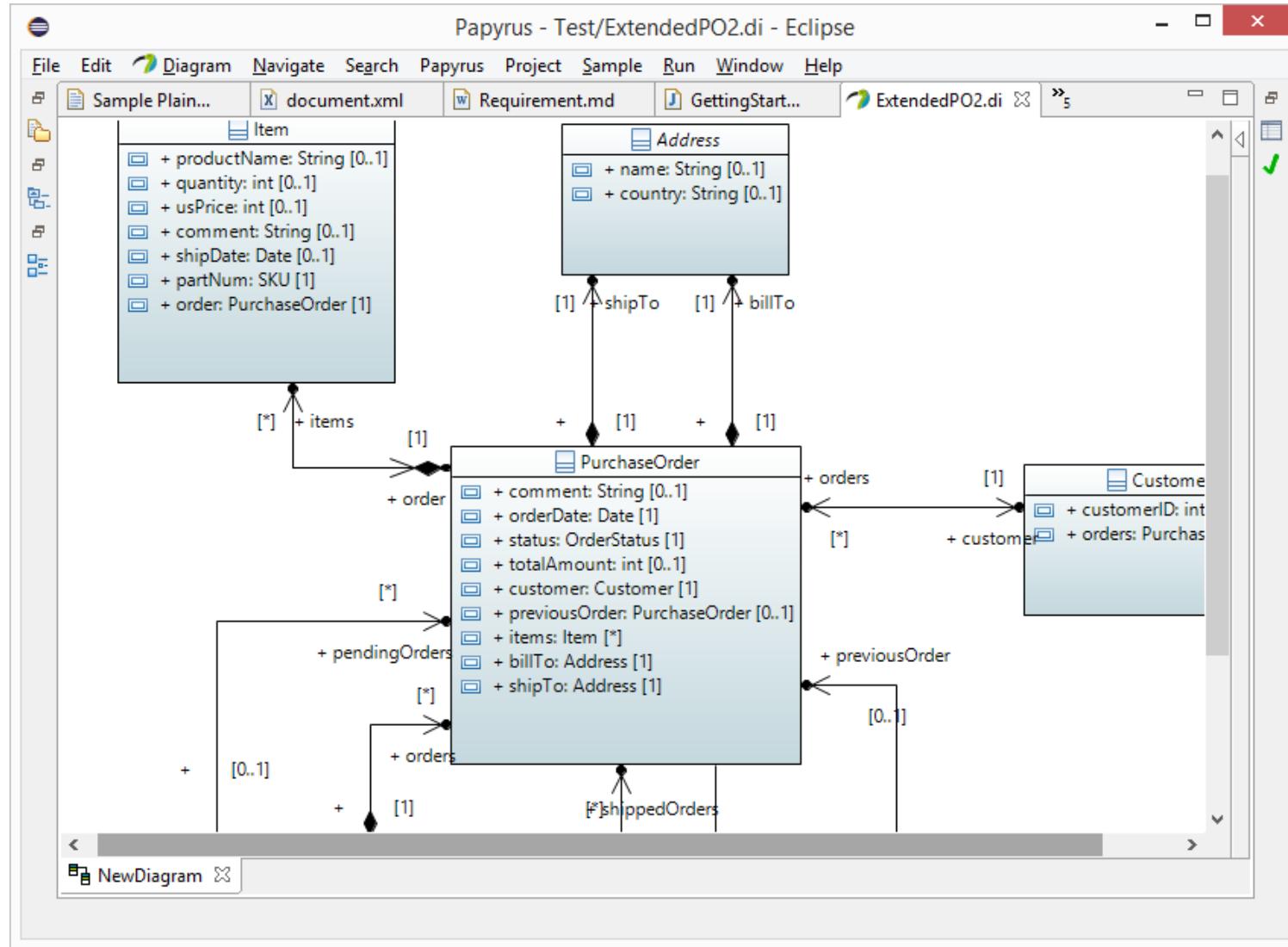
Everything is a model! (BPMN Standard)

Business Process Model & Notation



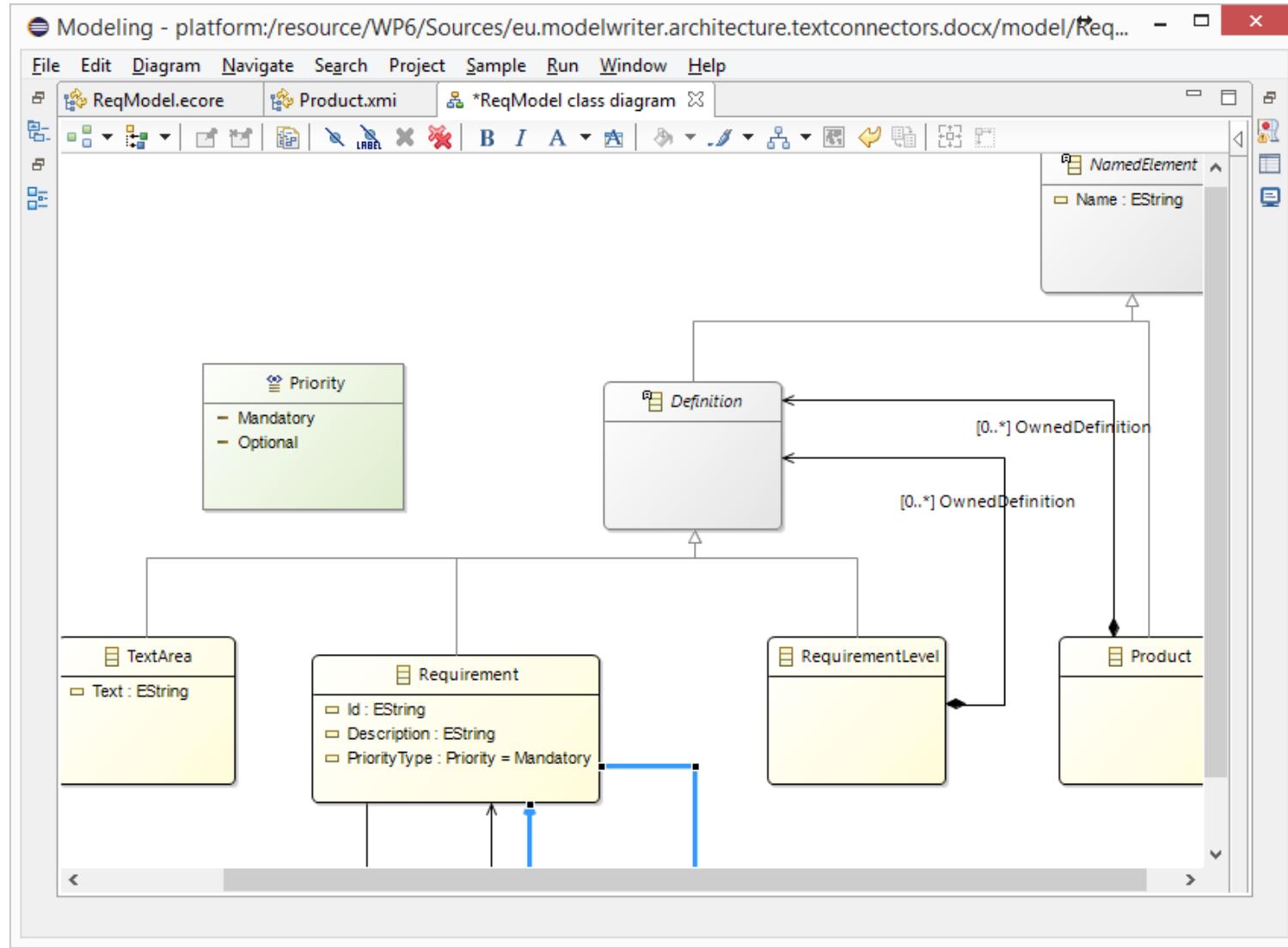
Everything is a model! (UML Standard)

UML Modeling Languages



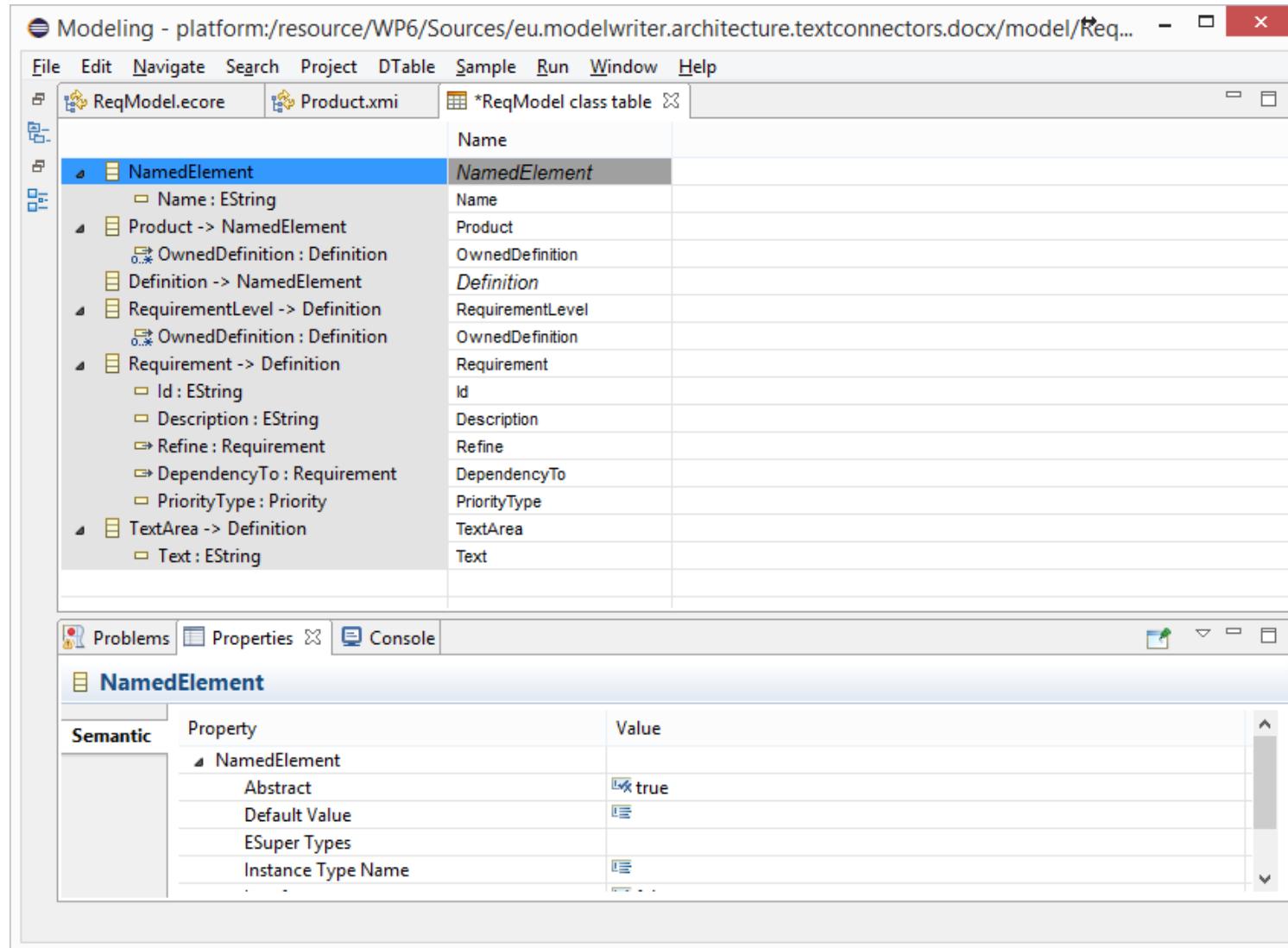
Everything is a model!

Eclipse Modeling Framework (EMF)



Everything is a model!

Tree-based or Tabular Representations



The screenshot shows a modeling environment with the following components:

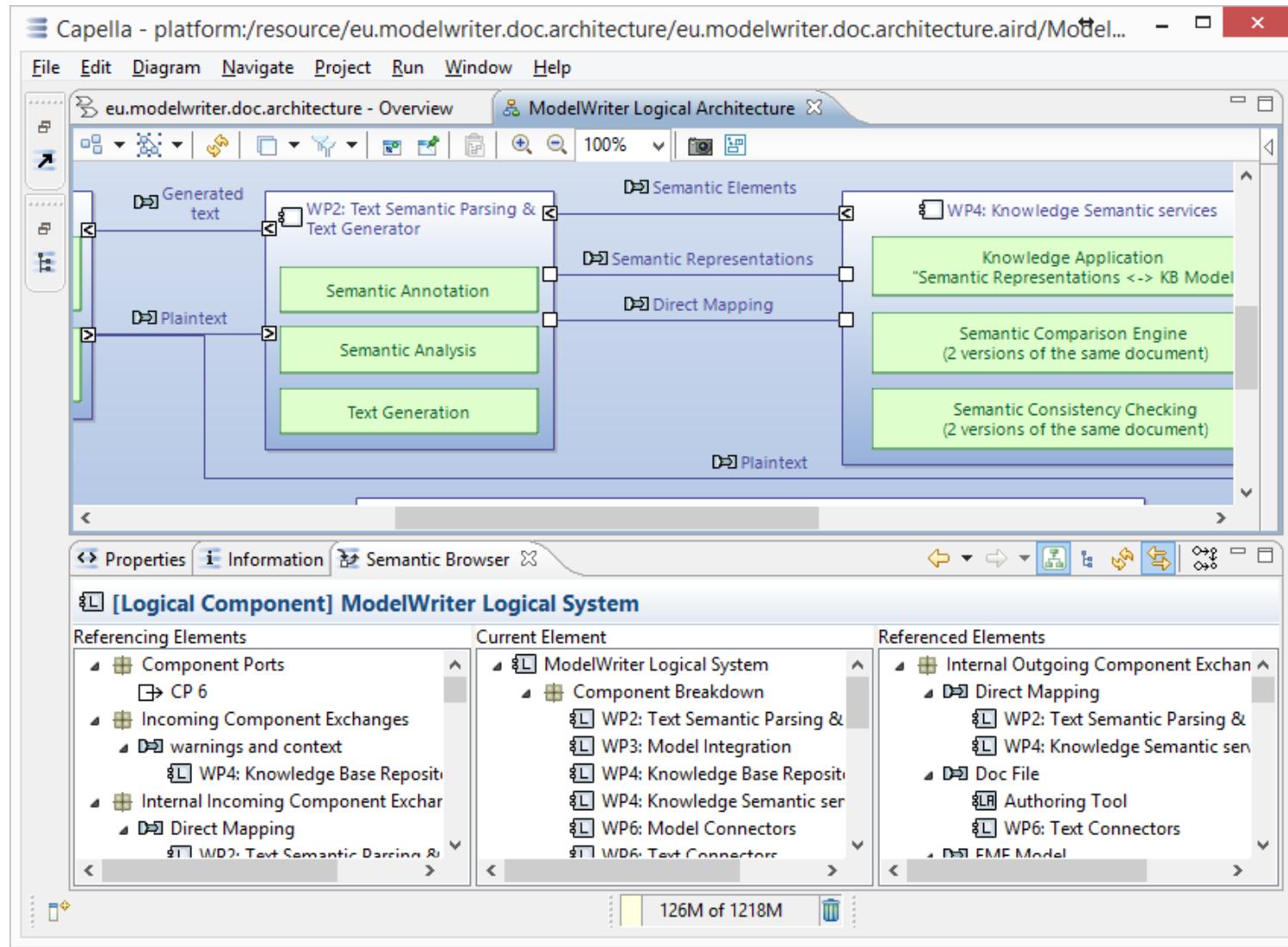
- Top Bar:** File, Edit, Navigate, Search, Project, DTable, Sample, Run, Window, Help.
- Left Sidebar:** Shows a tree view of model elements:
 - ReqModel.ecore
 - Product.xmi
 - *ReqModel class table
 - NamedElement
 - Name : EString
 - Product -> NamedElement
 - OwnedDefinition : Definition
 - Definition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

Name	Value
Name	NamedElement
Product	
OwnedDefinition	
Definition	
RequirementLevel	
OwnedDefinition	
Requirement	
Id	
Description	
Refine	
DependencyTo	
PriorityType	
TextArea	
Text	
- Bottom Area:** Problems, Properties, Console tabs. The Properties tab is active, showing the semantic properties of the "NamedElement" class.

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

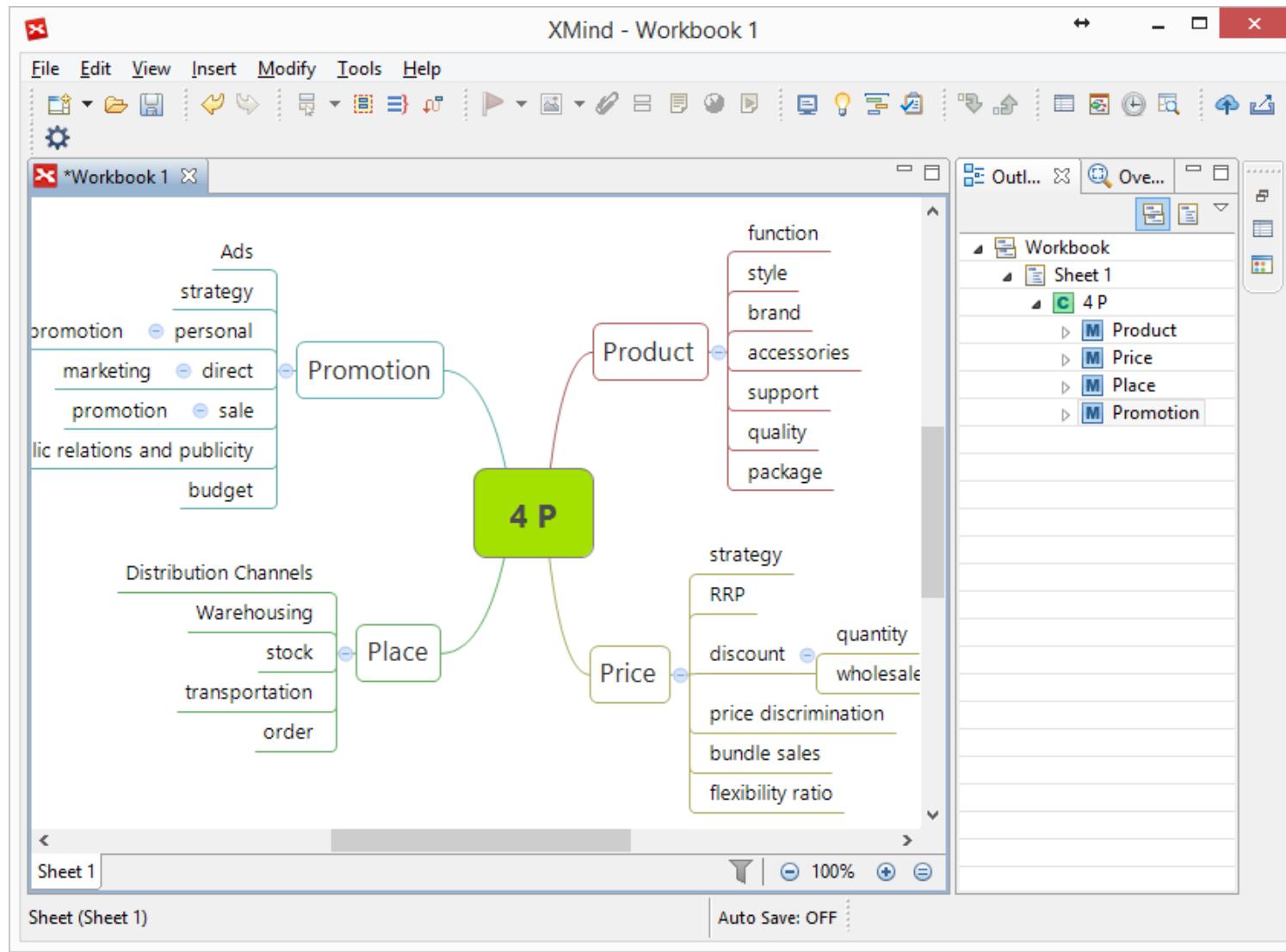
Everything is a model!

Software/System Architecture Design



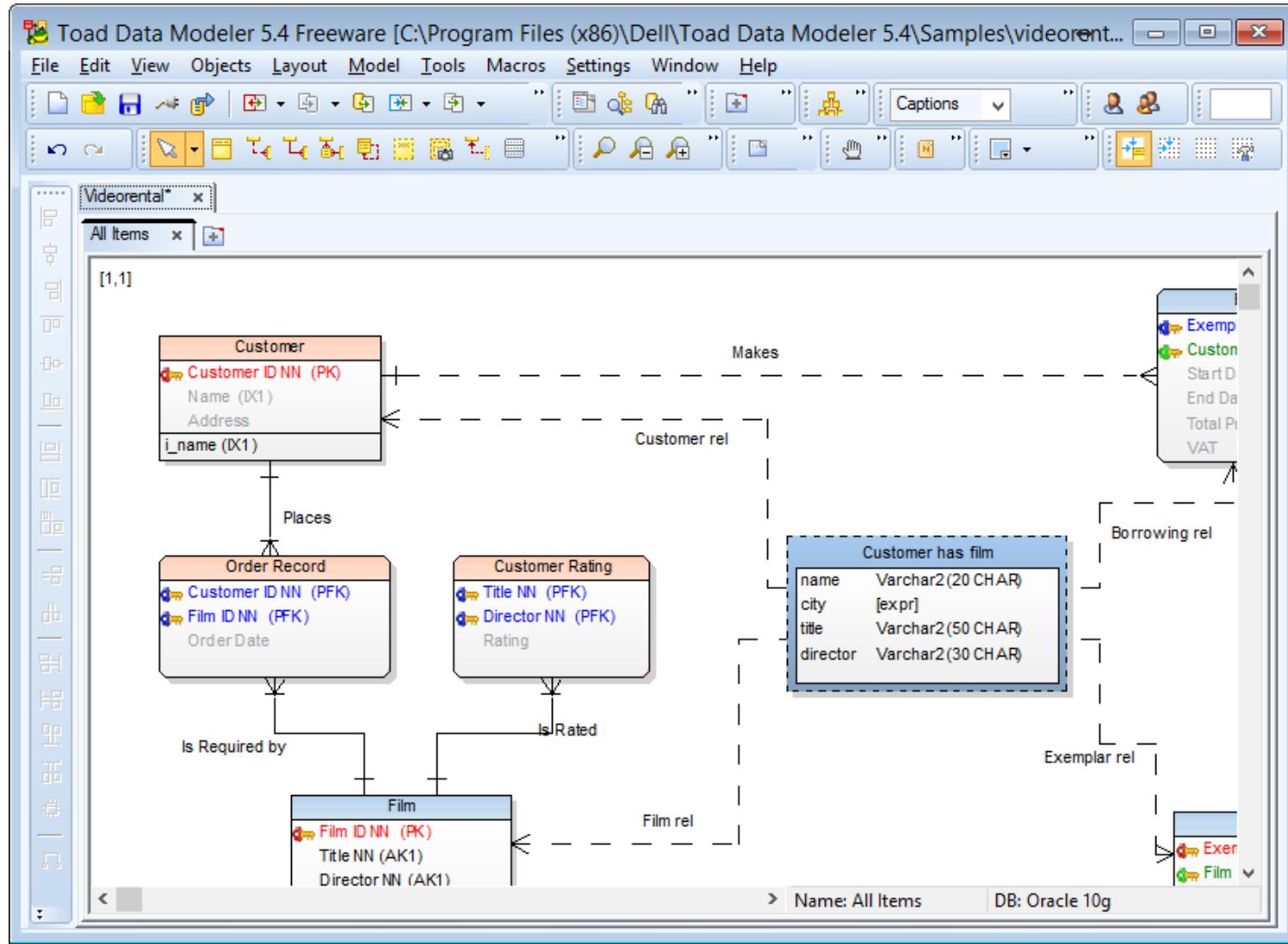
Everything is a model!

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)



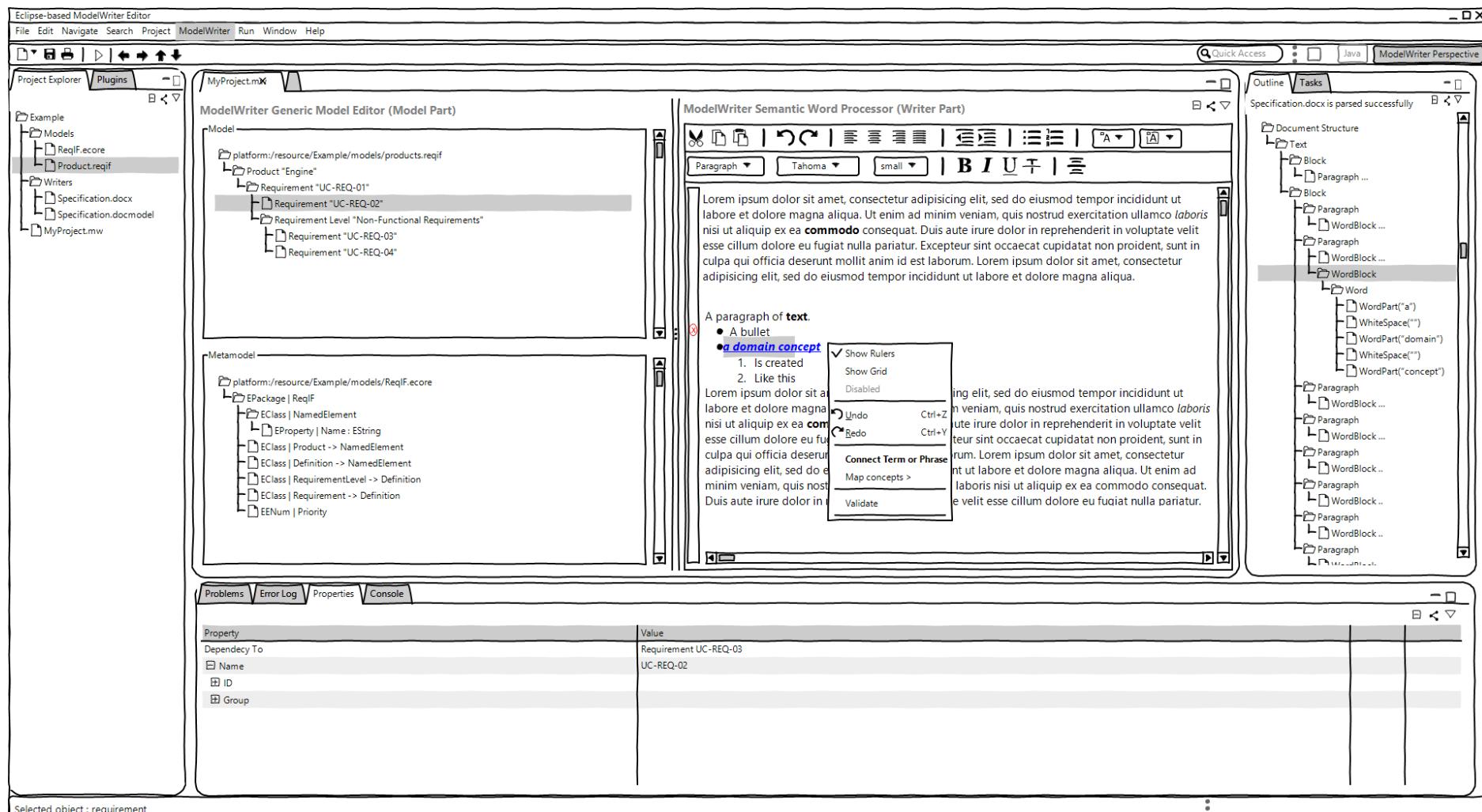
The screenshot shows a Java code editor with the following details:

- Title Bar:** Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/src/eu/modelwriter/architecture/text...
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Sample, Run, Window, Help.
- Toolbars:** Standard toolbar with icons for file operations like Open, Save, Print, and Find.
- Code Editor:** The main pane displays the `ReqModel2DocxConverter` class from the `eu.modelwriter.architecture.textconnectors.docx` package. The code includes imports for `java.io.File`, `XWPFDocument`, and `Resource`. It defines static fields for requirement keywords and a static method `Convert` that uses a template document to create a new `XWPFDocument`.
- Outline View:** A sidebar on the right shows the class hierarchy. `ReqModel2DocxConverter` is highlighted. Its methods are listed: `Convert(Resource)`, `getResource()`, `preOrder(RequirementLevel)`, `writeRequirement(Definition)`, and `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

Plug-in Development - eu.modelwriter.demonstration.requirements/HavelsanConfigurationModel.mw - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Specification.java

```
classDiagram
    class ContractRequirement
    class SystemRequirement
    class Implementation
    class Task
    class HardwareRequirement
    class SoftwareRequirement
    class TestCase

    ContractRequirement "1..2" --> "1..2" SystemRequirement : relate
    ContractRequirement "1..2" --> "1..2" SystemRequirement : system
    SystemRequirement "1..2" --> "1..2" Implementation : child
    Implementation "1..2" --> "1..2" HardwareRequirement : task
    Implementation "1..2" --> "1..2" SoftwareRequirement : extends
    SoftwareRequirement "1..2" --> "1..2" TestCase : test
```

Customer Requirements Specification.md

Customer Requirements Specification

UC-1 Create a new Spec Object

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

- We assume that a Specification exists and is open (not required for alternative scenario)
- Open a row's context menu (or in the empty editor space)
- Select the Child or Sibling submenu.
- Select the desired SpecObject Type (or none) from the submenu.
- This results in a new SpecHierarchy being created that is linked to a newly created SpecObject with the correct type.

59 changed lines

Alternative 1: Create in Outline

The same workflow works for elements that are shown underneath "Specifications" in the outline.

It is also possible to create children of the "SpecObjects" folder in the outline, but in this case, no SpecHierarchy will be created.

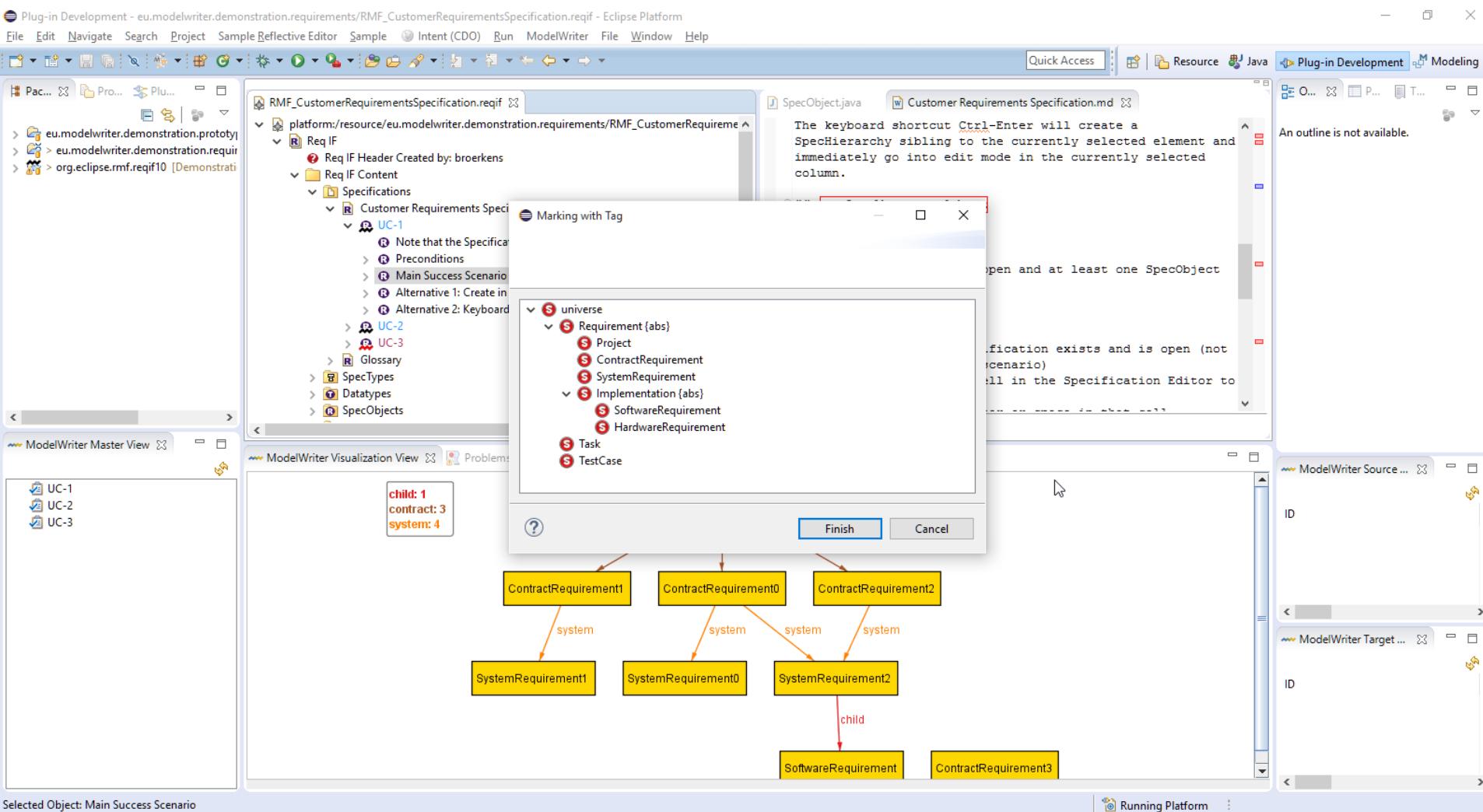
ModelWriter Visualization View

```
graph TD
    Project --- CR1[ContractRequirement1]
    Project --- CR2[ContractRequirement2]
    Project --- CR3[ContractRequirement3]
    CR1 --- SR1[SystemRequirement1]
    CR1 --- SR2[SystemRequirement2]
    CR1 --- SR3[SystemRequirement3]
    SR1 --- HW1[HardwareRequirement]
    SR2 --- SW1[SoftwareRequirement]
    SR3 --- CR4[ContractRequirement4]
```

Running Platform

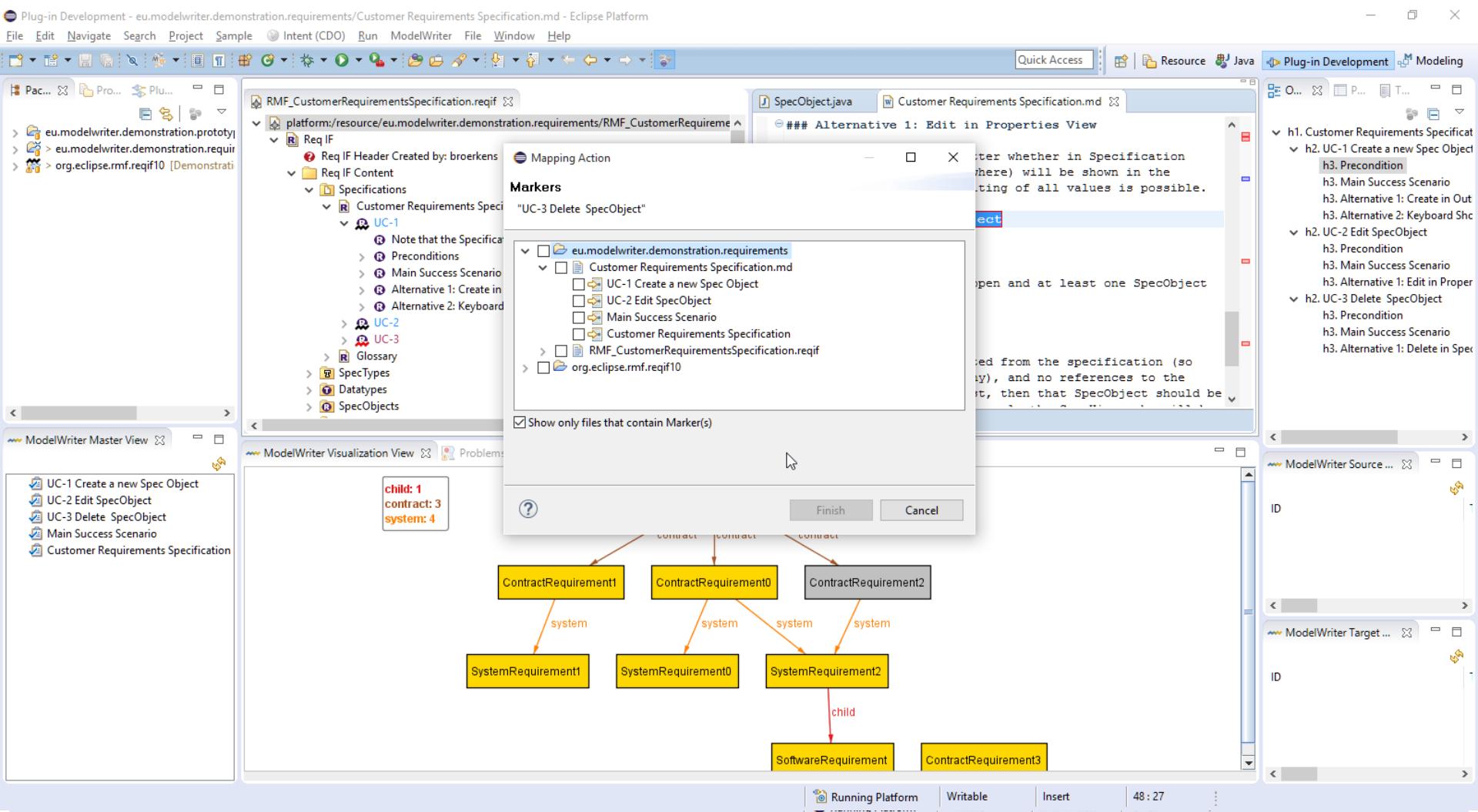
Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

Model and its Instances

Plug-in Development - eu.modelwriter.demonstration.requirements/RMF_CustomerRequirementsSpecification.reqif - Eclipse Platform

File Edit Navigate Project Sample Reflective Editor Sample Intent (CDO) Run ModelWriter Window Help

Customer Requirements Specification.md Specification.java RMF_CustomerRequirementsSpecification.reqif

Outline Properties

Property Value

Spec Hierarchy

- Desc
- Editable
- Editable Atts
- Identifier
- Last Change
- Long Name
- Object
- Table Internal

Spec Object

- Desc
- Identifier
- Last Change
- Long Name
- Type

Use Case Type

- Description
- ID

reqif10.ecore HavelsanConfigurationModel.mw

platform/resource/eu.modelwriter.demonstration.requirements/reqif10.ecore

reqif10

- AccessControlledElement -> Identifiable
- Identifiable
- AttributeValueXHTML -> AttributeValue
- AttributeValue
- SpecElementWithAttributes -> Identifiable
- AttributeDefinitionXHTML -> AttributeDefinition
- AttributeDefinition -> AccessControlledElement
- SpecType -> Identifiable
- ReqIFContent
- ReqIF
- ReqIFHeader
- ReqIFToolExtension
- SpecObject -> SpecElementWithAttributes
- SpecObjectType -> SpecType
- Specification -> SpecElementWithAttributes
- SpecificationType -> SpecType
- SpecHierarchy -> AccessControlledElement
- DatatypeDefinition -> Identifiable
- SpecRelation -> SpecElementWithAttributes
- SpecRelationType -> SpecType
- RelationGroup -> Identifiable
- RelationGroupType -> SpecType
- DatatypeDefinitionXHTML -> DatatypeDefinition
- AlternativeID
- AttributeDefinitionBoolean -> AttributeDefinitionSimple
- AttributeDefinitionSimple -> AttributeDefinition

ModelWriter Visualization View Problems Markers

Project

ContractRequirement2 ContractRequirement0 ContractRequirement1

SystemRequirement2 SystemRequirement0 SystemRequirement1

HardwareRequirement SoftwareRequirement0 ContractRequirement3 SoftwareRequirement1

Selected Object: UC-2

Running Platform

Text & Model-Synchronized Document Engineering Platform

OMG ReqIF ECore Metamodel

Plug-in Development - eu.modelwriter.demonstration.requirements/reqif10.ecore - Eclipse Platform

File Edit Navigate Project Sample Intent (CDO) Run ModelWriter Window Help

Package Project Ex Plug-ins

eu.modelwriter.demonstration.prototypes [Demo] eu.modelwriter.demonstration.requirements [Demo] Customer Requirements Specification.md HavelsanConfigurationModel.mw reqif10.ecore

reqif10.ecore HavelsanConfigurationModel.mw

Customer Requirem Specification.java RMF_CustomerRequir reqif10.ecore

Property Value
Abstract false
Default Value
ESuper Types
Instance Type Name
Interface false
Name ReqIF

Quick Access Java Plug-in Development Git

375 </eAnnotations>
376 <eAnnotations source="http://org/eclipse/sphinx/emf/seriali
377 <details key="wrapperName" value="SPEC-RELATION-GROUPS"/>
378 <details key="featureWrapperElement" value="true"/>
379 <details key="featureElement" value="false"/>
380 <details key="classifierWrapperElement" value="false"/>
381 <details key="classifierElement" value="true"/>
382 </eAnnotations>
383 </eStructuralFeatures>
384 </eClassifiers>
385 <eClassifiers xsi:type="ecore:EClass" name="ReqIF">
386 <eAnnotations source="http://org/eclipse/emf/ecore/util/Exter
387 <details key="name" value="REQ-IF"/>
388 <details key="kind" value="elementOnly"/>
389 </eAnnotations>
390 <eAnnotations source="http://org/eclipse/sphinx/emf/serializa
391 <details key="wrapperName" value="REQ-IF"/>
392 </eAnnotations>
393 <eStructuralFeatures xsi:type="ecore:EAttribute" name="lang" c
394 <details key="unsettable" value="true"/>
395 <eAnnotations source="http://org/eclipse/emf/ecore/util/Ext
396 <details key="name" value="lang"/>
397 <details key="kind" value="attribute"/>
398 <details key="namespace" value="http://www.w3.org/XML/1998
399 </eAnnotation>
400 </eStructuralFeatures>
401 <eStructuralFeatures xsi:type="ecore:EReference" name="theHea
402 <details key="name" value="theHeader"/>
403 <details key="kind" value="reference"/>
404 <details key="upperBound" value="1"/>
405 <details key="multiplicity" value="1..1"/>
406 <details key="type" value="RMF_CustomerRequir"/>
407 <details key="isOrdered" value="true"/>
408 <details key="isUnique" value="true"/>
409 <details key="isContainment" value="true"/>
410 <details key="isVolatile" value="false"/>

Design Source

ModelWriter Visualization View Problems Markers

child: 2 contract: 3 system: 5

```
graph TD; Project -- contract --> CR2[ContractRequirement2]; Project -- contract --> CR0[ContractRequirement0]; Project -- contract --> CR1[ContractRequirement1]; CR2 -- system --> SR2[SystemRequirement2]; CR0 -- system --> SR0[SystemRequirement0]; CR1 -- system --> SR1[SystemRequirement1]; SR2 -- child --> HW[HardwareRequirement]; SR0 -- child --> SW0[SoftwareRequirement0]; SR1 -- child --> SW1[SoftwareRequirement1]; SR1 -- child --> CR3[ContractRequirement3]
```

Selected Object: ReqIF

Running Platform

Text & Model-Synchronized Document Engineering Platform

Java Classes and Interfaces

Plug-in Development - org.eclipse.rmf.reqif10/src/org/eclipse/rmf/reqif10/Specification.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Quick Access Java Plug-in Development Git

Outline Properties

Specification.java

```
Customer Requirements Specification.md Specification.java RMF_CustomerRequirementsSpecification.reqif
```

Specification.java code:

```
1 module Haveksan/Requirement
2
3 abstract sig Requirement {}
4
5 sig Task {
6   precede: lone Task,
7 }{ all t: Task | lone t.~task}
8
9 one sig Project extends Requirement {
10   contract: some ContractRequirement
11 }
12 sig ContractRequirement extends Requirement {
13   system: set SystemRequirement,
14   relate: set ContractRequirement
15 }{all c: ContractRequirement | one c.~contract}
16
17 --@name: "System Requirement"
18 sig SystemRequirement extends Requirement {
19   child: some Implementation
20 }{ all s: SystemRequirement | one s.~system}
21
22 abstract sig Implementation extends Requirement {
23   task: set Task
24 }{ all i: Implementation | one i.~child}
25
26 --@context.editor: "ReqIFEEditor"
27 sig SoftwareRequirement extends Implementation {
```

Properties view:

- org.eclipse.rmf.reqif10
- Specification
 - getChildren(): EList<SpecHierarchy>
 - getType(): SpecificationType
 - isSetChildren(): boolean
 - isSetType(): boolean
 - setType(SpecificationType): void
 - unsetChildren(): void
 - unsetType(): void

ModelWriter Visualization View

Project

ContractRequirement2, ContractRequirement0, ContractRequirement1

SystemRequirement2, SystemRequirement0, SystemRequirement1

HardwareRequirement, SoftwareRequirement0, ContractRequirement3, SoftwareRequirement1

Relationships:

- Project has three ContractRequirements (contract)
- Each ContractRequirement has three SystemRequirements (system)
- SystemRequirements have children: HardwareRequirement, SoftwareRequirement0, ContractRequirement3, SoftwareRequirement1 (child)

Text & Model-Synchronized Document Engineering Platform

Arbitrary Text Files (markdown...)

Plug-in Development - eu.modelwriter.demonstration.requirements/HavelsanConfigurationModel.mw - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Specification.java

```
classDiagram
    class ContractRequirement
    class SystemRequirement
    class Implementation
    class Task
    class HardwareRequirement
    class SoftwareRequirement
    class TestCase

    ContractRequirement "1..2" --> "1..2" SystemRequirement : relate
    ContractRequirement "1..2" --> "1..2" SystemRequirement : system
    SystemRequirement "1..2" --> "1..2" Implementation : child
    Implementation "1..2" --> "1..2" HardwareRequirement : task
    Implementation "1..2" --> "1..2" SoftwareRequirement : extends
    SoftwareRequirement "1..2" --> "1..2" TestCase : test
    Task "1..2" --> "1..2" Task : precede
```

Customer Requirements Specification.md

Customer Requirements Specification

UC-1 Create a new Spec Object

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

- We assume that a Specification exists and is open (not required for alternative scenario)
- Open a row's context menu (or in the empty editor space)
- Select the Child or Sibling submenu.
- Select the desired SpecObject Type (or none) from the submenu.
- This results in a new SpecHierarchy being created that is linked to a newly created SpecObject with the correct type.

59 changed lines

Alternative 1: Create in Outline

The same workflow works for elements that are shown underneath "Specifications" in the outline.

It is also possible to create children of the "SpecObjects" folder in the outline, but in this case, no SpecHierarchy will be created.

Markdown Source Preview

ModelWriter Visualization View

Project

```
graph TD
    Project --- CR1[ContractRequirement1]
    Project --- CR2[ContractRequirement2]
    Project --- CR3[ContractRequirement3]
    CR1 --- SR1[SystemRequirement1]
    CR1 --- SR2[SystemRequirement2]
    CR1 --- SR3[SystemRequirement3]
    CR2 --- SR4[SystemRequirement4]
    CR2 --- SR5[SystemRequirement5]
    CR2 --- SR6[SystemRequirement6]
    CR3 --- SR7[SystemRequirement7]
    CR3 --- SR8[SystemRequirement8]
    CR3 --- SR9[SystemRequirement9]
    SR1 --- CR4[ContractRequirement4]
    SR2 --- CR5[ContractRequirement5]
    SR3 --- CR6[ContractRequirement6]
    SR4 --- CR7[ContractRequirement7]
    SR5 --- CR8[ContractRequirement8]
    SR6 --- CR9[ContractRequirement9]
    SR7 --- CR10[ContractRequirement10]
    SR8 --- CR11[ContractRequirement11]
    SR9 --- CR12[ContractRequirement12]
    CR4 --- HW1[HardwareRequirement1]
    CR5 --- SW1[SoftwareRequirement1]
    CR6 --- CR13[ContractRequirement13]
    CR7 --- SW2[SoftwareRequirement2]
    CR8 --- CR14[ContractRequirement14]
    CR9 --- CR15[ContractRequirement15]
```

Running Platform

Text & Model-Synchronized Document Engineering Platform

Is it possible to vizualize the trace links?



Traceability: Havelsan example

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter File Window Help

Pac... Pro... Plu... eu.modelwriter.demonstration.protobuf eu.modelwriter.demonstration.requirements org.eclipse.rmf.reqif10 [Demonstrati

RMF_CustomerRequirementsSpecification.reqif

Req IF
Req IF Header Created by: broerkens
Req IF Content
Specifications
Customer Requirements Specification
UC-1
UC-2
UC-3
Glossary
SpecTypes
Datatypes
SpecObjects

SpecObject.java Customer Requirements Specification.md

Customer Requirements Specification
UC-1 Create a new Spec Object
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)

ModelWriter Master View

ModelWriter Visualization View

Project

ContractRequirement1, ContractRequirement0, ContractRequirement2

SystemRequirement1, SystemRequirement0, SystemRequirement2

SoftwareRequirement, ContractRequirement3

ModelWriter context menu: Add/Remove Type, Delete Marker, MapMarker

ModelWriter Source ...

ID

ID

The screenshot shows the ModelWriter application interface. On the left, there's a tree view of requirements (Req IF) and a list of recent files. The main area has two tabs: 'SpecObject.java' and 'Customer Requirements Specification.md'. The 'md' tab contains a requirement 'UC-1 Create a new Spec Object' with its details. Below it is a 'ModelWriter Visualization View' showing a hierarchy of requirements: 'Project' at the top, followed by three 'ContractRequirement' nodes, which then point to three 'SystemRequirement' nodes. A 'ModelWriter' context menu is open over one of the 'SystemRequirement' nodes, showing options like 'Add/Remove Type', 'Delete Marker', and 'MapMarker'. At the bottom, there are toolbars for 'Running Platform', 'Writable', 'Insert', and a timestamp '10:23'.

A Formal Specification Model to configure the ModelWriter

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

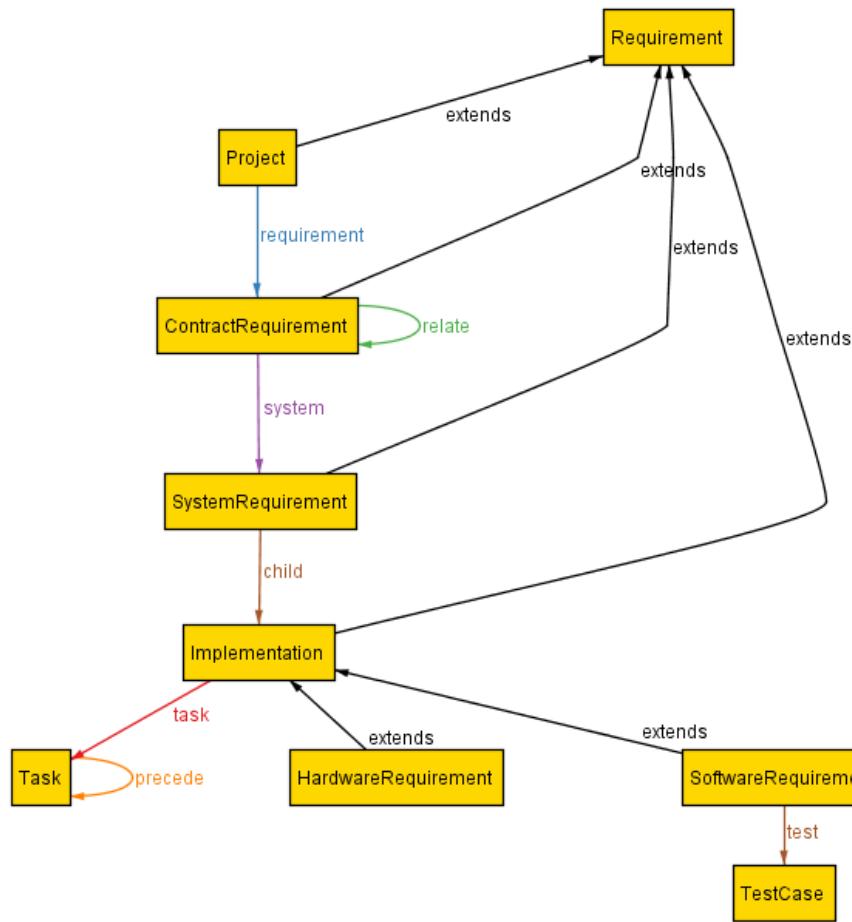
```
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/id270">
  <j:0:hasOffset>270</j:0:hasOffset>
  <j:0:isSameAs>http://www.linkedmodel.org/schema/vaem#id</j:0:isSameAs>
  <j:0:hasValue>id</j:0:hasValue>
</rdf:Description>
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/attach818">
  <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

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
}
```

Specification: model view

Plug-in Development - eu.modelwriter.demonstration.requirements/HavelsanConfigurationModel.mw - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Quick Access Java Plug-in Development Git

Customer Requirements Specification.md RMF_CustomerRequirementsSpecification.reqif

Customer Requirements Specification

UC-1 Create a new Spec Object

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

- We assume that a Specification exists and is open (not required for alternative scenario)
- Open a row's context menu (or in the empty editor space)
- Select the Child or Sibling submenu.
- Select the desired SpecObject Type (or none) from the submenu.
- 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

The same workflow works for elements that are shown underneath "Specifications" in the outline.

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

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.

Specification Source

ModelWriter Visualization View Problems Markers

Project

Requirement

ContractRequirement

SystemRequirement

Implementation

Task

HardwareRequirement

SoftwareRequirement

ContractRequirement2

ContractRequirement0

ContractRequirement1

SystemRequirement2

SystemRequirement0

SystemRequirement1

child: 6
contract: 1
precede: 1
relate: 1
system: 1
task: 1
test: 1

child: 2
contract: 3
system: 5

```
graph TD; Project -- contract --> CR[ContractRequirement]; Project -- extends --> R[Requirement]; CR -- relate --> CR; CR -- system --> SR[SystemRequirement]; SR -- child --> I[Implementation]; I -- task --> Task; I -- extends --> HW[HardwareRequirement]; I -- extends --> SW[SoftwareRequirement]; CR -- contract --> CR2[ContractRequirement2]; CR -- contract --> CR0[ContractRequirement0]; CR -- contract --> CR1[ContractRequirement1]; CR2 -- system --> SR2[SystemRequirement2]; CR0 -- system --> SR0[SystemRequirement0]; CR1 -- system --> SR1[SystemRequirement1]; SR0 -- child --> SR1
```

A Formal Specification Model to configure the ModelWriter

Specification: source view

Plug-in Development - eu.modelwriter.demonstration.requirements/HavelsanConfigurationModel.mw - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Customer Requirements Specification.md RMF_CustomerRequirementsSpecification.reqif

reqif10.ecore Specification.java HavelsanConfigurationModel.mw

```
1 module Havelsan/Requirement
2
3 abstract sig Requirement {}
4
5 sig Task {
6   precede: lone Task,
7 }{ all t: Task | lone t.~task}
8
9 one sig Project extends Requirement {
10   contract: some ContractRequirement
11 }
12 sig ContractRequirement extends Requirement {
13   system: set SystemRequirement,
14   relate: set ContractRequirement
15 }{all c: ContractRequirement | one c.~contract}
16
17 --@name: "System Requirement"
18 sig SystemRequirement extends Requirement {
19   child: some Implementation
20 }{ all s: SystemRequirement | one s.~system}
21
22 abstract sig Implementation extends Requirement {
23   task: set Task
24 }{ all i: Implementation | one i.~child}
25
26 --@context.editor: "ReqIFEditor"
27 sig SoftwareRequirement extends Implementation {
28   test: some TestCase
29 }
30
31 sig HardwareRequirement extends Implementation {}
```

Quick Access Java Plug-in Development Git

Customer Requirements Specification

UC-1 Create a new Spec Object

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

- We assume that a Specification exists and is open (not required for alternative scenario)
- Open a row's context menu (or in the empty editor space)
- Select the Child or Sibling submenu.
- Select the desired SpecObject Type (or none) from the submenu.
- 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

The same workflow works for elements that are shown underneath "Specifications" in the outline. 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

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.

Markdown Source Preview

ModelWriter Visualization View Problems Markers

Project

ContractRequirement2

ContractRequirement0

ContractRequirement1

SystemRequirement2

SystemRequirement0

SystemRequirement1

child

child

```
graph TD; Project -- contract --> CR2[ContractRequirement2]; Project -- contract --> CR0[ContractRequirement0]; Project -- contract --> CR1[ContractRequirement1]; CR2 -- system --> SR2[SystemRequirement2]; CR0 -- system --> SR0[SystemRequirement0]; CR1 -- system --> SR1[SystemRequirement1]; SR0 -- child --> C1[ ]; SR1 -- child --> C2[ ];
```

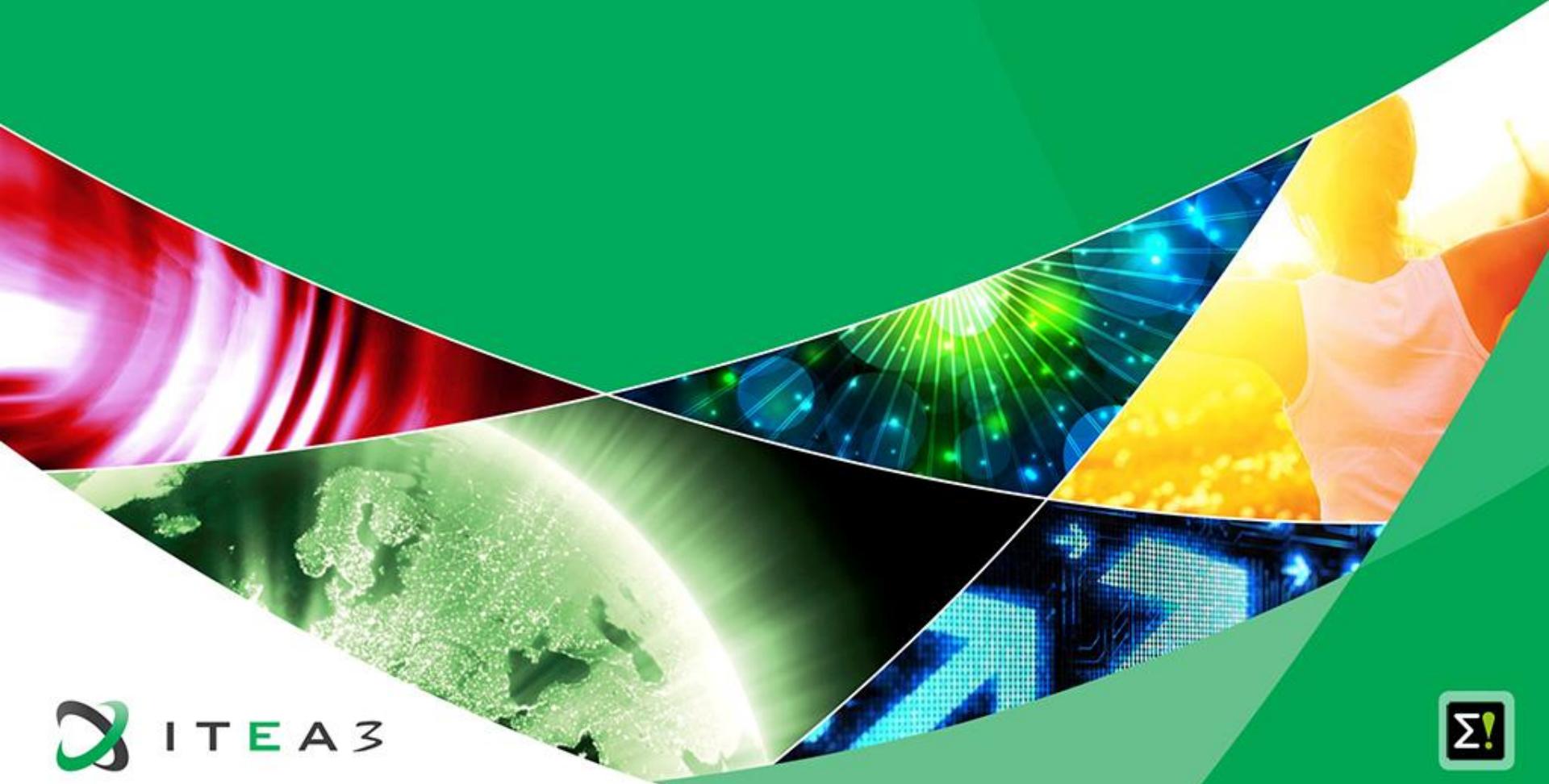
A Formal Specification Model to configure the ModelWriter



Synchronization is maintained!

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

2 Progress on Workpackages



WP1 - Industrial use case and requirements

Leader: Anne MONCEAUX
AIRBUS GROUP

WP1-Industrial Use Cases & Requirements

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

- HISBIM, UNIT, KOCSISTEM, AIRBUS, OBEO, 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

T.1.1 Evaluation Methods & Tools

Evaluation should show:

- What actually occurred?
- Whether it had an impact, expected or unexpected, and
- What links exist between a program and its observed impacts?

This will result in D1.1.1 Evaluation Methods & Tools

D1.1.1 Evaluation Methods & Tools

Methods & Tools should answer the questions below?

- What are we trying to achieve?
- Is the role of the group to advise or steer the project?
- Who will draft the plan (or protocol)?
- Who and how will the data be collected?
- Who and how will the data be analysed?
- Who will write up the final report?
- What is the time frame?

Usability evaluation with the Use Case Evaluation (UCE) method consists of three overall activities, see Fig. 1:

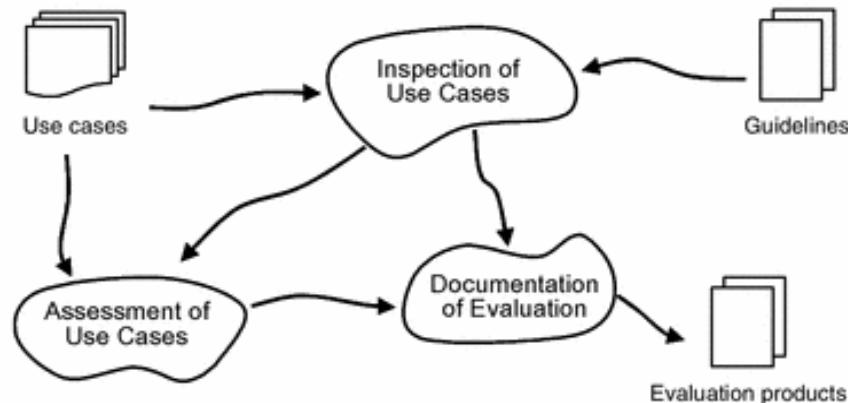


Fig. 1. Overview of Activities and Materials in Use Case Evaluation (UCE)

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



- 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

T1.6 Consolidated Software Requirements and Review



- 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

Requirement Management – GitHub

<https://github.com/ModelWriter/Requirements/issues>



The screenshot shows the GitHub Issues page for the repository "ModelWriter / Requirements". The page displays 103 open issues. The first few issues listed are:

- The system must support the specification of the connector to the document**
Labels: Functional, Software Requirements Document (SRD), WP6 - ModelWriter Architecture Integration and Evaluation
#116 opened 19 hours ago by annemonceaux
- Generator**
Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#115 opened on Sep 16 by scruzlara
- Parser**
Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#114 opened on Sep 16 by scruzlara
- Grammar-Extractor**
Labels: Desirable, Functional, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#113 opened on Sep 16 by scruzlara
- Grammar**
Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#112 opened on Sep 16 by scruzlara
- Lexicon-Extractor**
Labels: Desirable, Functional, Semantic Annotator, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#111 opened on Sep 16 by scruzlara
- Lexicon**
Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#110 opened on Sep 16 by scruzlara
- Reversible-Semantic-Engine**
Labels: Functional, Mandatory, Semantic Analysis, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#109 opened on Sep 16 by scruzlara
- Domain-Ontology**
Labels: Mandatory, Non-Functional, Semantic Analysis, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents
#108 opened on Sep 16 by scruzlara

Requirement Management – Waffle.io

<https://waffle.io/ModelWriter/Requirements>



The screenshot shows the ModelWriter Requirements application interface. At the top, there's a navigation bar with icons for Home, Model, Requirements, Test Cases, and Issues. The main area displays a grid of requirement cards, each containing a title, description, status, and various buttons for managing the issue.

User Stories (73 total):

- 116**: The system must support the specification of the connector to the document. Status: Confirmed (29). Subtasks: 8. Labels: Functional, WPs - ModelWriter Architecture Integration and Evaluation, Software Requirements Document (SRD).
- 115**: Generator. Status: Confirmed (29). Subtasks: 8. Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents.
- 114**: Parser. Status: Confirmed (29). Subtasks: 8. Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents.
- 113**: Grammar-Extractor. Status: Confirmed (29). Subtasks: 8. Labels: Desirable, Functional, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents.
- 112**: Grammar. Status: Confirmed (29). Subtasks: 8. Labels: Functional, Mandatory, Reversible Semantic Engine, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents.
- 111**: Lexicon-Extractor. Status: Confirmed (29). Subtasks: 8. Labels: Desirable, Functional, Semantic Annotator, Software Requirements Document (SRD), WP2 - Semantic Parsing and Generation of Documents.
- 110**: Lexicon. Status: Confirmed (29). Subtasks: 8. Labels: Functional, WPs - ModelWriter Architecture Integration and Evaluation, Software Requirements Document (SRD).

Confirmed (29 total):

- 20**: The System shall propose an eclipse editor to handle documentation/models mappings. Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, UC-FR-01 Sync. between Models and Documentation, User Requirements Document (URD).
- 52**: Text-Based Knowledge Extraction with ModelWriter (Semantic Word Processor). Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, ModelWriter's Feature from FPP, User Requirements Document (URD).
- 29**: The system shall be easy to integrate in an RCP application. Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, UC-FR-02 Enterprise Architecture, UC-TR-04 Requirements Engineering with SysML Designer, User Requirements Document (URD).
- 23**: The System shall not enforce any dependency on non-open source artifacts such as tools, applications, and libraries. Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, UC-FR-01 Sync. between Models and Documentation, User Requirements Document (URD).
- 25**: The system should allow to filter synchronization warnings, errors and information. Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, UC-FR-01 Sync. between Models and Documentation, User Requirements Document (URD).
- 31**: The system shall allow the user to activate/deactivate a synchronization direction. Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, UC-FR-02 Enterprise Architecture, User Requirements Document (URD).
- 47**: The user does not want to be bothered with information such as mapping links. Status: Confirmed (29). Subtasks: 8. Labels: Mandatory, UC-FR-01 Sync. between Models and Documentation.

In Progress (0 total):

Done (2 total):

- 30**: The system shall allow the users to work in a collaborative manner. Status: Done (2). Labels: Mandatory, UC-FR-02 Enterprise Architecture, User Requirements Document (URD).
- 53**: ModelWriter as a Next Generation Requirements Engineering Tool: ModelWriter should be equipped with Requirements Engineering features. Status: Done (2). Labels: Mandatory, ModelWriter's Feature from FPP, UC-BE-01 Requirements IT, UC-TR-03 Generation and management of feature models, UC-TR-04 Requirements Engineering with SysML Designer, User Requirements Document (URD).

User Requirement Document (URD) & Software Requirement Document (SRD) I



Plug-in Development - org.eclipse.rmf.docs.requirements/Customer Requirements Specification.md - Eclipse Platform

File Edit Navigate Search Project Run ModelWriter File Window Help

Parse Alloy Set Marker Visibility Preferences Project Management >

Create User Requirements Document (URD)
Create Software Requirements Document (SRD)
Create Software Requirements Review Meeting Document (SRR)
Create User Requirements Review Meeting Document (URR)

Customer Requirements Specification

Customer Requirements Specification

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

Precondition

Main Success Scenario

We assume that a Specification exists and is open (not required for alter

Open a row's context menu (or in the empty editor space)

Select the Child or Sibling submenu

Select the desired Spec Object Type (or none) from the submenu

This results in a new SpecHierarchy being created that is linked to a new

Alternative 1: Create in Outline

Alternative 2: Keyboard Shortcut

UC-2

Precondition

Main Success Scenario

Alternative 1: Edit in Properties View

UC-3

Glossary

SpecTypes

Datatypes

SpecObjects

Specification

UC-2 Edit SpecObject

UC-1 Create a new SpecObject

Customer Requirements Specification

Precondition

Precondition

Specification

Specification Editor

ReqIF model exists and is open.

Main Success Scenario

We assume that a Specification exists and is open (not required for alternative scenario)

Open a row's context menu (or in the empty editor space)

Select the Child or Sibling submenu

Select the desired Spec Object Type (or none) from the submenu

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

The same workflow works for elements that are shown underneath

Markdown Source Preview

ModelWriter Master View

ID Text

Specification

UC-2 Edit SpecObject

UC-1 Create a new SpecObject

Customer Requirements Specification

Precondition

Precondition

Specification

ModelWriter Target Mapping View

ID Text

Running Platform Writable Insert 10:29

User Requirement Document (URD) & Software Requirement Document (SRD) II



Plug-in Development - org.eclipse.rmf.docs.requirements/Customer Requirements Specification.md - Eclipse Platform

File Edit Navigate Search Project Run ModelWriter File Window Help

RMF_CustomerRequirementsSpecification.reaif SpecObject.java reaif10.ecore Customer Requirements Specification.md

Select Requirements :

Please select from user requirements to add in the document

- Issue 53 - ModelWriter as a Next Generation Requirements Engineering Tool: ModelWriter should be equipped with Requirements Engineering f
- Issue 52 - Text-Based Knowledge Extraction with ModelWriter (Semantic Word Processor)
- Issue 51 - "MW" Knowledge Dissemination Standard
- Issue 50 - ModelWriter shall support Rich-Blended Modeling Environments.
- Issue 49 - The system shall provide a unified Graphical User Interface (for both Model and Writer parts).
- Issue 48 - The system should be able to perform semantic parsing.
- Issue 47 - The user does not want to be bothered with information such as mapping links.
- Issue 42 - ModelWriter should support at least one Document Markup Language and one Lightweight Markup Language
- Issue 41 - The system shall help to synchronize the SDFP natural language document with the modeled rules without forced modification.
- Issue 40 - The system shall show on demand (coloured mark or other mean) the text elements that are linked to the "visual model" concepts, ar
- Issue 39 - The system shall allow the user to configure the document generation content
- Issue 38 - The system shall provide a user friendly way to manage any additional concepts needed.
- Issue 37 - A specification for an improve and controlled formulation of the rules in semi-structured natural language should be proposed
- Issue 35 - The system shall allow semantic retrieving or reasoning using the model elements.
- Issue 33 - The system shall allow the end user to edit text and "visual" model (such as tables, diagrams or 2D drawings) synchronously
- Issue 32 - The system shall allow the end user to keep his/her usual working environment.
- Issue 31 - The system shall allow the user to activate/deactivate a synchronization direction
- Issue 30 - The system shall allow the users to work in a collaborative manner
- Issue 29 - The system shall be easy to integrate in an RCP application
- Issue 28 - The system shall offer a notification system
- Issue 25 - The system should allow to filter synchronization warnings, errors and information.
- Issue 23 - The System shall not enforce any dependency on non-open source artifacts such as tools, applications, and libraries.
- Issue 20 - The System shall propose an eclipse editor to handle documentation/models mappings

Select All Deselect All OK Cancel

ModelWriter Target Mapping View Text

Specification

Running Platform Writable Insert 10:29



User Requirement Document (URD) & Software Requirement Document (SRD) III



D1.6.2 Software Requirements Document (SRD).docx - Word

FILE HOME INSERT DESIGN PAGE LAYOUT REFERENCES MAILINGS REVIEW VIEW ADD-INS Acrobat

Cut Copy Format Painter

Font Paragraph Styles

Navigation

Search document

HEADINGS PAGES RESULTS

Document History

- 1. Introduction
 - 1.1. Role of the deliverable
 - 1.2. The List of Technical Work Packages
 - 1.3. Conventions
 - 1.4. Structure of the document
 - 1.5. Terms, abbreviations and definitions
- 2. Software Requirements
 - 2.1. REQ-UR-67
 - 2.2. REQ-UR-66
 - 2.3. REQ-UR-65
 - 2.4. REQ-UR-64
 - 2.5. REQ-UR-63
 - 2.6. REQ-UR-62
 - 2.7. REQ-UR-61
 - 2.8. REQ-UR-60
 - 2.9. REQ-UR-59
 - 2.10. REQ-UR-57
 - 2.11. REQ-UR-56
 - 2.12. REQ-UR-55
 - 2.13. REQ-UR-54
 - 2.14. REQ-UR-WP6-44
 - 2.15. REQ-UR-WP6-43
 - 2.16. REQ-UR-24
 - 2.17. REQ-UR-WP3-14
 - 2.18. REQ-UR-WP3-13

ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands
T +31 88 003 6136
E info@itea3.org
W www.itea3.org
ITEA 3 is a EUREKA strategic ICT cluster programme

D1.6.2 Software Requirements Document (SRD)

ModelWriter

Text & Model-Synchronized Document Engineering Platform

PAGE 1 OF 13 1196 WORDS ENGLISH (UNITED KINGDOM) SAVING AUTORECOVERY FILE NORMAL.DOTM %120

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 - Semantic Parsing and Generation of Documents and Documents Components



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.

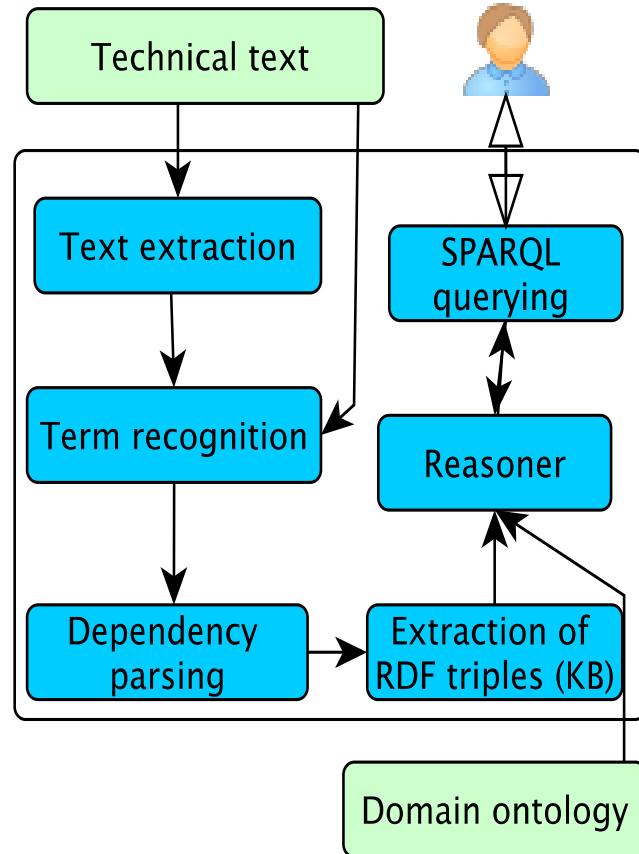
T2.1 Data Collection

- 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)
 - Lexicalization: automatic acquisition of a lexicon mapping RDF properties to natural language expressions
 - Document planning: automatic detection of typical document structures using DBpedia and Wikipedia

Automatically Creation of Synchronization Links



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

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>

Semantic Annotation

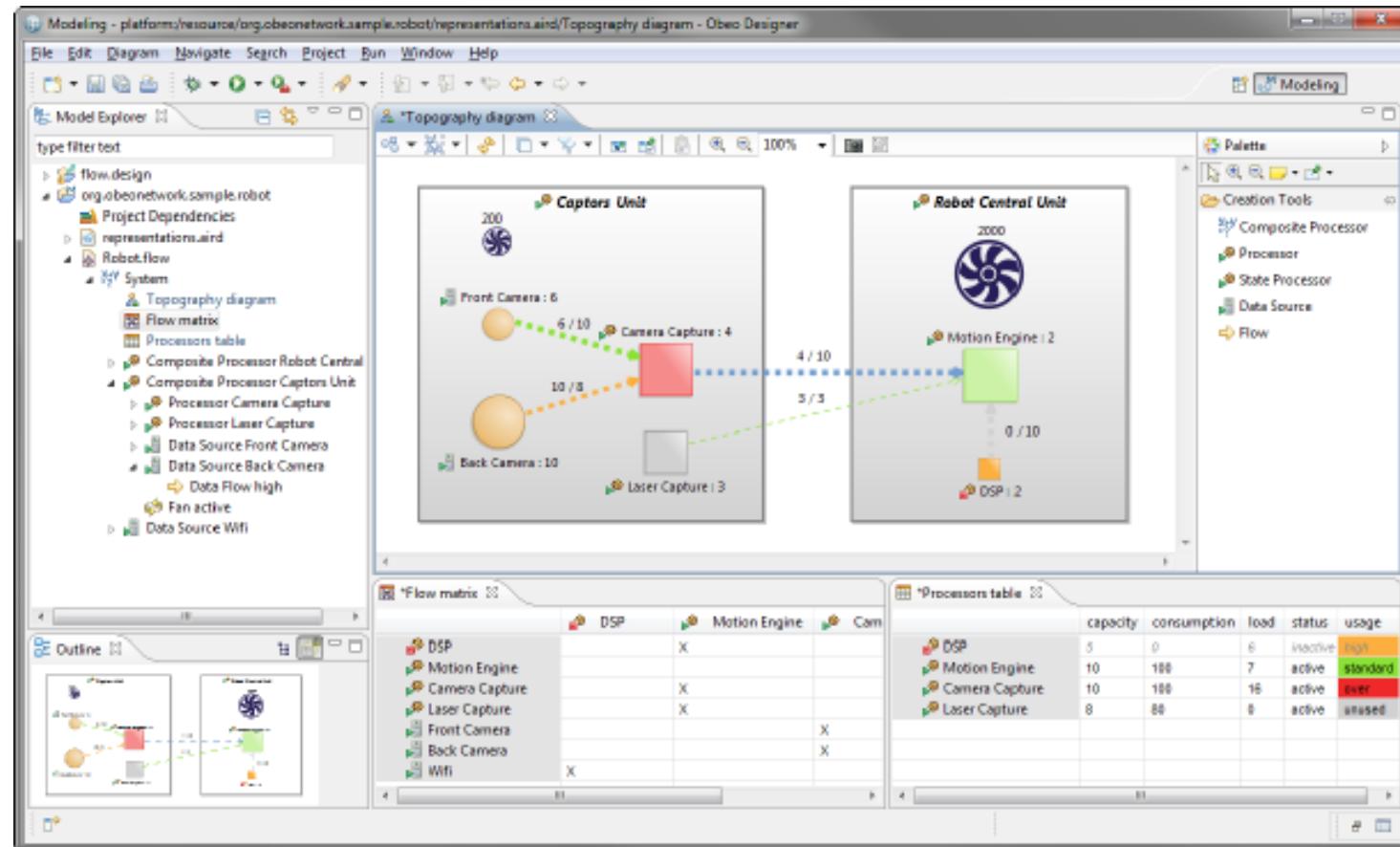
- EMF (Eclipse Modeling Framework)
 - The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model
 - From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes 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

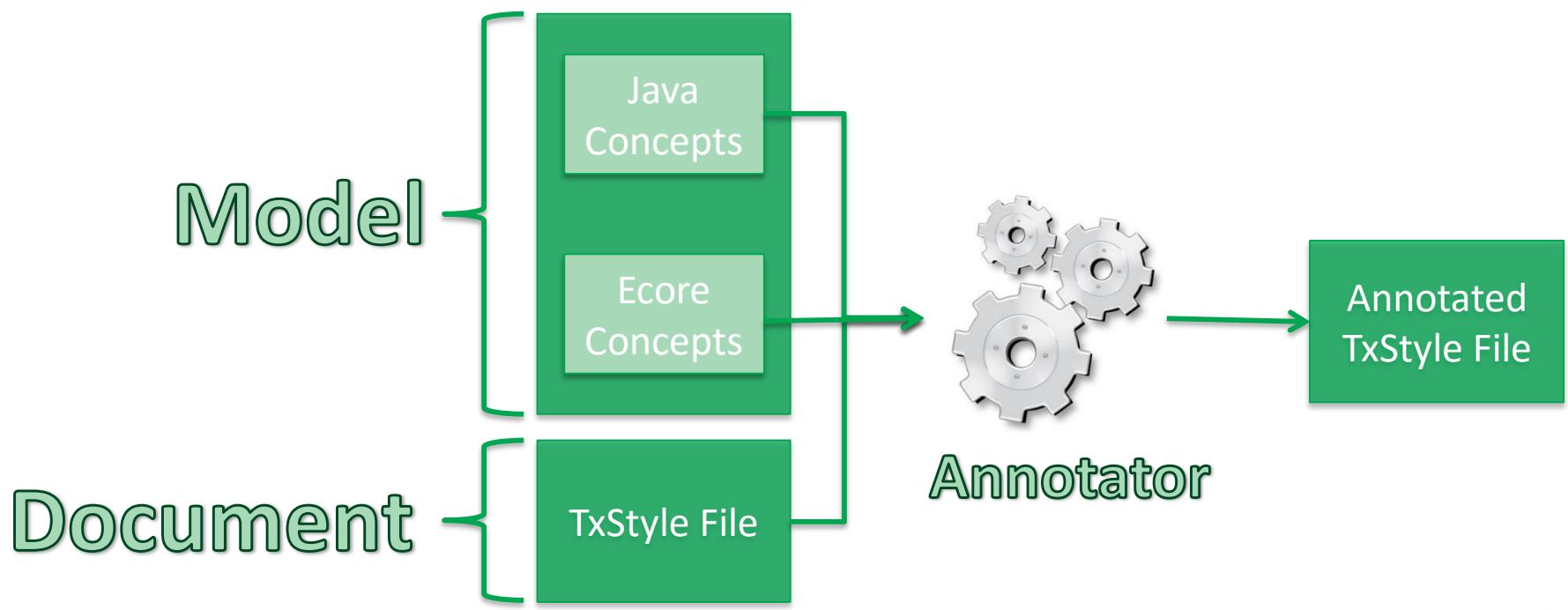


Semantic Annotation

- ***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



WP3 - Model to/from Knowledge Base Synchronization Mechanism

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

WP3 - Model to/from Knowledge Base Synchronization Mechanism



- 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

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

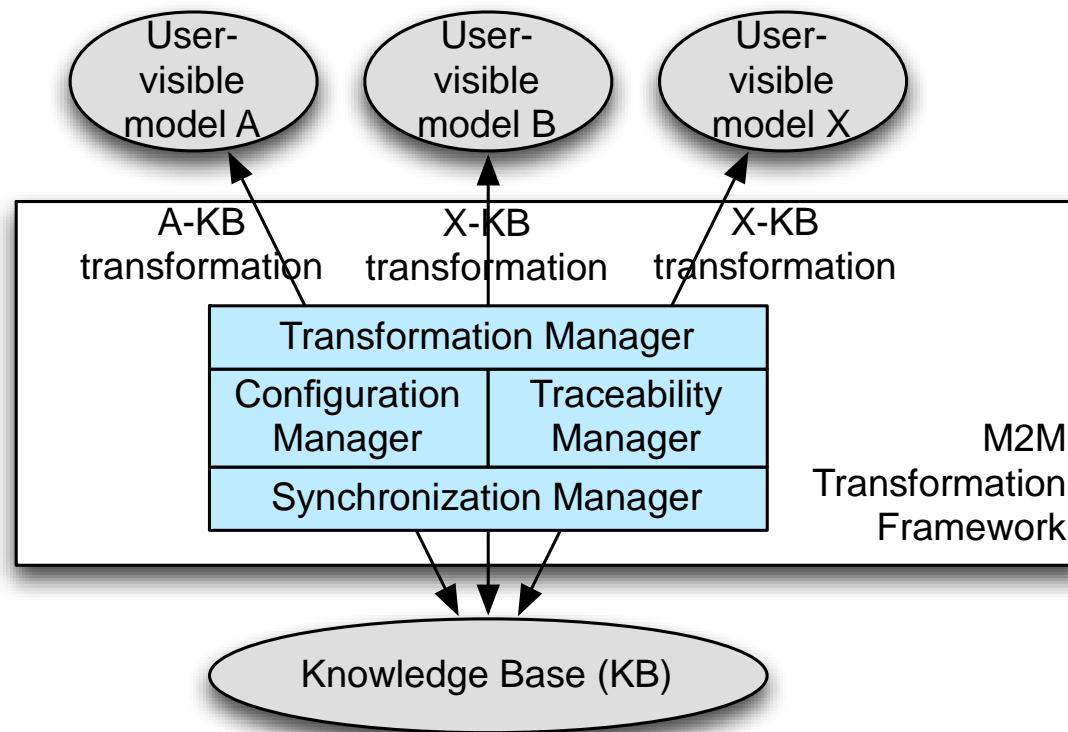
T3.2 - Specification and design of the M2M Transformation Framework



- 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:

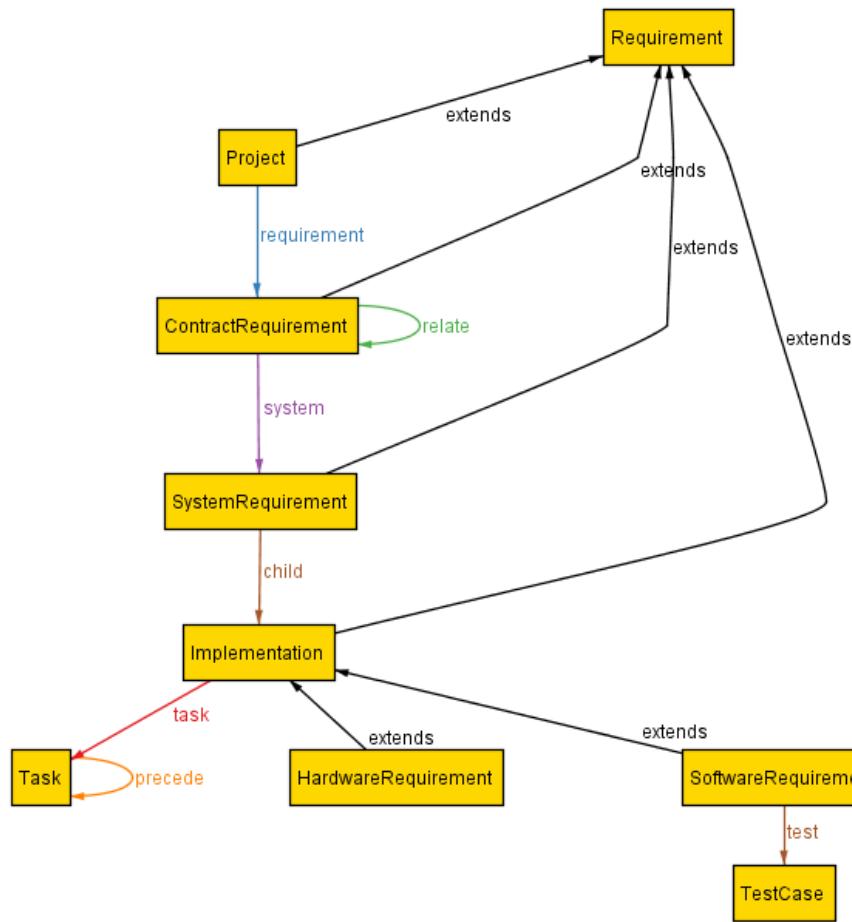


T3.3, T3.4, T3.5, T3.6 (UNIT + KOCSISTEM)

- 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.

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
}
```

Specification: model view

Plug-in Development - eu.modelwriter.demonstration.requirements/HavelsanConfigurationModel.mw - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Quick Access Java Plug-in Development Git

Customer Requirements Specification.md RMF_CustomerRequirementsSpecification.reqif

Customer Requirements Specification

UC-1 Create a new Spec Object

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

- We assume that a Specification exists and is open (not required for alternative scenario)
- Open a row's context menu (or in the empty editor space)
- Select the Child or Sibling submenu.
- Select the desired SpecObject Type (or none) from the submenu.
- 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

The same workflow works for elements that are shown underneath "Specifications" in the outline.

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

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.

Specification Source

ModelWriter Visualization View Problems Markers

Project

Requirement

ContractRequirement

SystemRequirement

Implementation

Task

HardwareRequirement

SoftwareRequirement

ContractRequirement2

ContractRequirement0

ContractRequirement1

SystemRequirement2

SystemRequirement0

SystemRequirement1

child: 6
contract: 1
precede: 1
relate: 1
system: 1
task: 1
test: 1

child: 2
contract: 3
system: 5

```
graph TD; Project -- contract --> ContractRequirement; Project -- extends --> Requirement; ContractRequirement -- relate --> SystemRequirement; ContractRequirement -- system --> SystemRequirement; SystemRequirement -- child --> Implementation; Implementation -- task --> Task; Implementation -- extends --> HardwareRequirement; Implementation -- extends --> SoftwareRequirement; Requirement -- extends --> ContractRequirement2; Requirement -- extends --> ContractRequirement0; Requirement -- extends --> ContractRequirement1; ContractRequirement2 -- system --> SystemRequirement2; ContractRequirement0 -- system --> SystemRequirement0; ContractRequirement1 -- system --> SystemRequirement1; SystemRequirement0 -- child --> SystemRequirement1
```

A Formal Specification Model to configure the ModelWriter

Specification: source view

Plug-in Development - eu.modelwriter.demonstration.requirements/HavelsanConfigurationModel.mw - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter Window Help

Customer Requirements Specification.md RMF_CustomerRequirementsSpecification.reqif

reqif10.ecore Specification.java HavelsanConfigurationModel.mw

```
1 module Havelsan/Requirement
2
3 abstract sig Requirement {}
4
5 sig Task {
6   precede: lone Task,
7 }{ all t: Task | lone t.~task}
8
9 one sig Project extends Requirement {
10   contract: some ContractRequirement
11 }
12 sig ContractRequirement extends Requirement {
13   system: set SystemRequirement,
14   relate: set ContractRequirement
15 }{all c: ContractRequirement | one c.~contract}
16
17 --@name: "System Requirement"
18 sig SystemRequirement extends Requirement {
19   child: some Implementation
20 }{ all s: SystemRequirement | one s.~system}
21
22 abstract sig Implementation extends Requirement {
23   task: set Task
24 }{ all i: Implementation | one i.~child}
25
26 --@context.editor: "ReqIFEditor"
27 sig SoftwareRequirement extends Implementation {
28   test: some TestCase
29 }
30
31 sig HardwareRequirement extends Implementation {}
```

Quick Access Java Plug-in Development Git

Customer Requirements Specification

UC-1 Create a new Spec Object

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

- We assume that a Specification exists and is open (not required for alternative scenario)
- Open a row's context menu (or in the empty editor space)
- Select the Child or Sibling submenu.
- Select the desired SpecObject Type (or none) from the submenu.
- 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

The same workflow works for elements that are shown underneath "Specifications" in the outline. 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

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.

Markdown Source Preview

ModelWriter Visualization View Problems Markers

Project

ContractRequirement2

ContractRequirement0

ContractRequirement1

SystemRequirement2

SystemRequirement0

SystemRequirement1

child

child

```
graph TD; Project -- contract --> CR2[ContractRequirement2]; Project -- contract --> CR0[ContractRequirement0]; Project -- contract --> CR1[ContractRequirement1]; CR2 -- system --> SR2[SystemRequirement2]; CR0 -- system --> SR0[SystemRequirement0]; CR1 -- system --> SR1[SystemRequirement1]; SR0 -- child --> C1[ ]; SR1 -- child --> C2[ ];
```

A Formal Specification Model to configure the ModelWriter

Traceability: Havelsan example

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter File Window Help

Pac... Pro... Plu... eu.modelwriter.demonstration.protobuf eu.modelwriter.demonstration.requirements org.eclipse.rmf.reqif10 [Demonstrati

RMF_CustomerRequirementsSpecification.reqif

Req IF
Req IF Header Created by: broerkens
Req IF Content
Specifications
Customer Requirements Specification
UC-1
UC-2
UC-3
Glossary
SpecTypes
Datatypes
SpecObjects

SpecObject.java Customer Requirements Specification.md

Customer Requirements Specification
UC-1 Create a new Spec Object
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)

ModelWriter Master View

ModelWriter Visualization View

Project

ContractRequirement1, ContractRequirement0, ContractRequirement2

SystemRequirement1, SystemRequirement0, SystemRequirement2

SoftwareRequirement, ContractRequirement3

ModelWriter context menu: Add/Remove Type, Delete Marker, MapMarker

ModelWriter Source ...

ID

ID

The screenshot shows the ModelWriter application interface. On the left, there's a tree view of requirements (Req IF) and a list of recent files. The main area has two tabs: 'SpecObject.java' and 'Customer Requirements Specification.md'. The 'md' tab contains a requirement 'UC-1 Create a new Spec Object' with its details. Below it is a 'ModelWriter Visualization View' showing a hierarchy from 'Project' down to 'SystemRequirement' and 'SoftwareRequirement' levels. A context menu is open over 'ContractRequirement2', showing options like 'Add/Remove Type', 'Delete Marker', and 'MapMarker'. The bottom status bar shows 'Running Platform' and 'Writable'.

A Formal Specification Model to configure the ModelWriter

Synchronization: Havelsan example

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform

File Edit Navigate Search Project Sample Intent (CDO) Run ModelWriter File Window Help

Pac... Pro... Plu... eu.modelwriter.demonstration.protobuf eu.modelwriter.demonstration.requirements org.eclipse.rmf.reqif10 [Demonstrati

RMF_CustomerRequirementsSpecification.reqif

Req IF
Req IF Header Created by: broerkens
Req IF Content
Specifications
Customer Requirements Specification
UC-1
UC-2
UC-3
Glossary
SpecTypes
Datatypes
SpecObjects

SpecObject.java Customer Requirements Specification.md

Customer Requirements Specification
UC-1 Create a new Spec Object
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)

ModelWriter Master View

ModelWriter Visualization View

Project

ContractRequirement1, ContractRequirement0, ContractRequirement2

SystemRequirement1, SystemRequirement0, SystemRequirement2

SoftwareRequirement, ContractRequirement3

ModelWriter Context Menu

Add/Remove Type
Delete Marker
MapMarker

ID

ID

The screenshot shows the ModelWriter application interface. On the left, the 'RMF_CustomerRequirementsSpecification.reqif' view displays a tree structure of requirements and specifications. In the center, the 'SpecObject.java' and 'Customer Requirements Specification.md' views show the 'UC-1 Create a new Spec Object' use case. Below these, the 'ModelWriter Visualization View' shows a hierarchical diagram where a 'Project' node branches into three 'ContractRequirement' nodes, which further branch into 'SystemRequirement' nodes. A context menu is open over the 'SystemRequirement2' node, showing options like 'Add/Remove Type', 'Delete Marker', and 'MapMarker'. The bottom status bar indicates the platform is 'Running Platform' and the time is '10:23'.

A Formal Specification Model to configure the ModelWriter

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

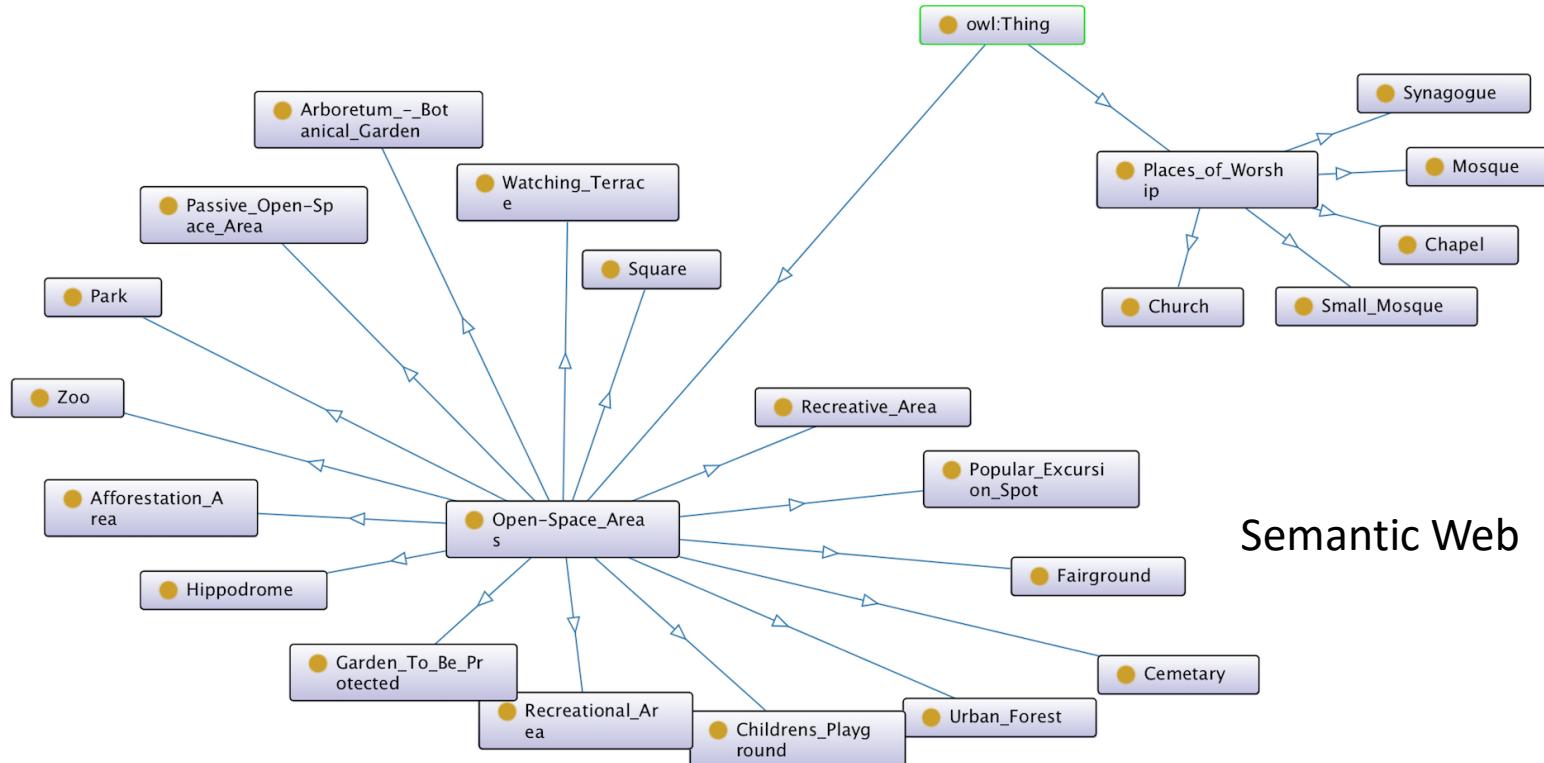
- Design and implement the ModelWriter's federated Knowledge Base itself, hosting multiple formalisms.
- Design and implement its bi-directional text-model synchronization mechanism.
- Design and implement its API.
- Design and implement a set of specialised modules (plug-ins) that exploit the Knowledge 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).

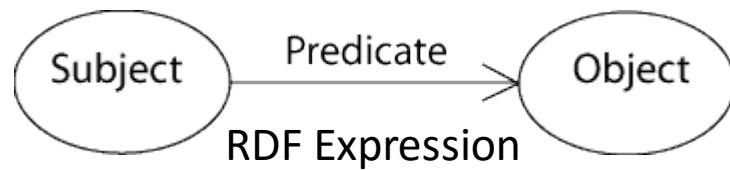
WP4 Knowledge Base Design and Implementation

- 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)

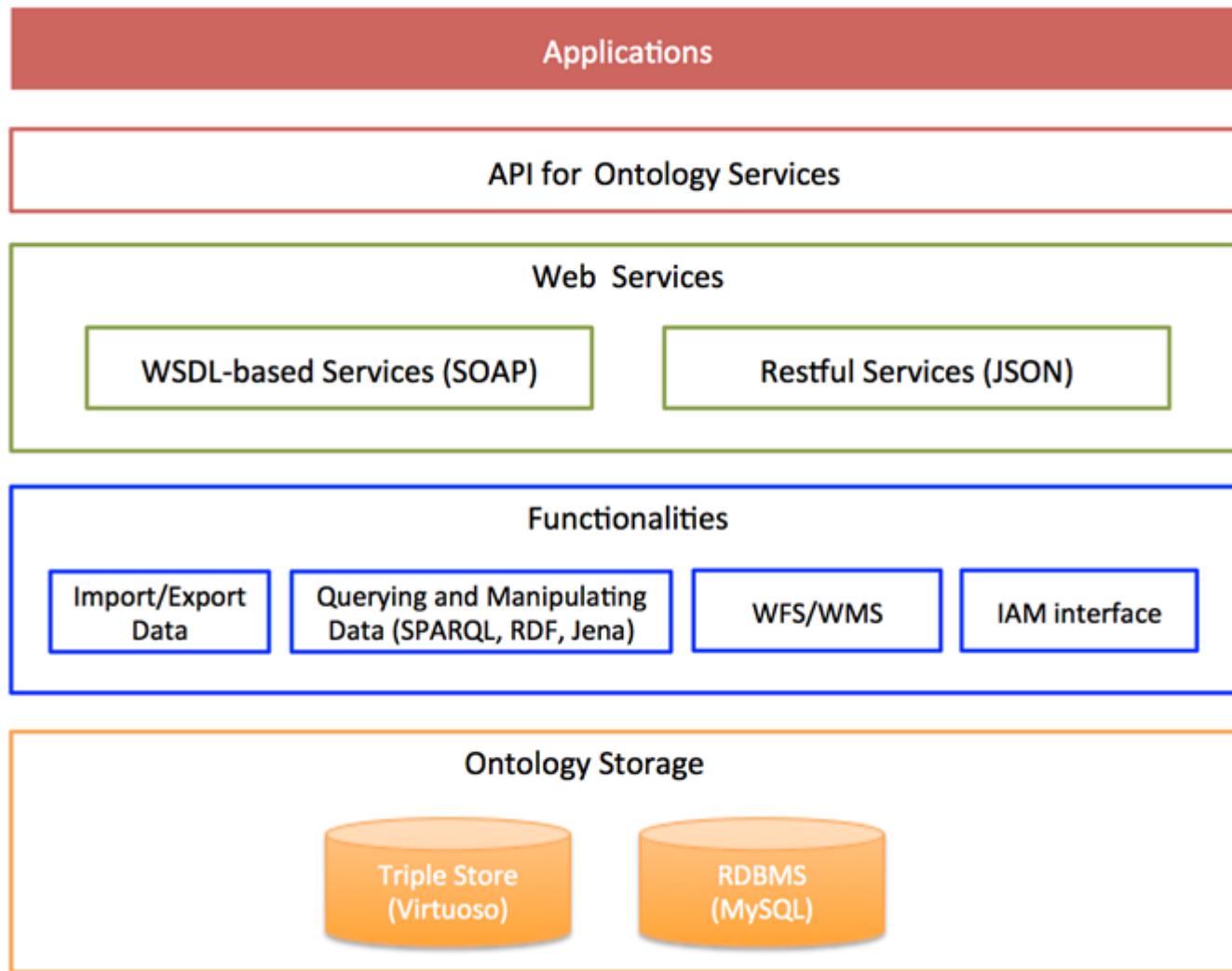
WP4 - T4.1 – Design of the Knowledge Base



Semantic Web



Ontology as a service



Implemented Services

- **executeQuery** (*ontology*, *queryString*): Executes *queryString* (a SPARQL query) against *ontology* and returns triples.
- **exportFromOntology** (*ontology*, *outputFile*, *type*): Exports the content of *ontology* into *outFile* of type *type*. *Type* is either CSV or RDF.
- **importIntoOntology** (*ontology*, *inputFile*, *type*): Imports the content of *inputFile* of type *type* into *ontology*. *Type* is either CSV or RDF.
- **insertTriple** (*ontology*, *triple*): Inserts *triple* into *ontology*.
- **removeTriple** (*ontology*, *triple*): Deletes *triple* from *ontology*.
- **updateTriple** (*ontology*, *tripleOld*, *tripleNew*): Changes the value of *tripleOld* as *tripleNew* in *ontology*.

WP5 – Project Management (UNIT)

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

WP5 - Overview

- Objectives:
 - To perform overall project governance, and to establish and maintain a communication and controlling infrastructure to run the project smoothly.
- Status:
 - Project Coordination Committee (PCC) and Technical Coordination Committee (TCC) are constituted.
 - Collaboration mechanism is established
 - Development management environment is created
- Next:
 - The 1st release will be issued

WP5 - Tasks

- T5.1 – Communication Management and Collaboration Infrastructure (UNIT + WP7 leader)
- T5.2 – Project Coordination and Reporting (UNIT + Country Coordinators + WP Leaders)
- T5.3 – Project Controls (UNIT + WP Leaders)
- T5.4 – Closing Project (UNIT + WP Leaders)

T5.1 Communication Management and Infrastructure



Deliverable Management – GitHub

<https://github.com/ModelWriter/Deliverables/issues>



Screenshot of the GitHub repository "ModelWriter / Deliverables" showing the Issues page.

The repository has 74 open issues. The search bar shows the filter "is:issue is:open".

Issues listed:

- D5.3.2 Updated State-of-the-art (Public Deliverable) - Document, Public, T5.3 - Project Controls, UNIT, WP5. #132 opened on Apr 9 by ferhaterata. M37
- D5.3.1 Final Project Report - Document, Private, T5.3 - Project Controls, UNIT, WP5. #131 opened on Apr 9 by ferhaterata. M37
- D5.2.5 Project Progress Report (fifth half year) - Document, Private, T5.2 - Project Coordination and Reporting, UNIT, WP5. #130 opened on Apr 9 by ferhaterata. M31
- D5.2.4 Project Progress Report (fourth half year) - Document, Private, T5.2 - Project Coordination and Reporting, UNIT, WP5. #129 opened on Apr 9 by ferhaterata. M25
- D5.2.3 Project Progress Report (third half year) - Document, Private, T5.2 - Project Coordination and Reporting, UNIT, WP5. #128 opened on Apr 9 by ferhaterata. M19
- D5.2.2 Project Progress Report (second half year) - Document, Private, T5.2 - Project Coordination and Reporting, UNIT, WP5. #127 opened on Apr 9 by ferhaterata. M13
- D7.7.1 ModelWriter and standardization activities - LORIA, Public, Software, T7.7 - Standardization, WP7. #123 opened on Dec 7, 2014 by ferhaterata. M34
- D7.3.1 Newsletter - International Conference Announcement - Document, Public, T7.3 - Workshops & Events, UNIT, WP7. #115 opened on Dec 7, 2014 by ferhaterata. M31
- D7.2.2 Roadmap for future exploitation and pre-competition survey - Document, HISBIM, Public, T7.2 - Business Model & Exploitation Plan, WP7. #114 opened on Dec 7, 2014 by ferhaterata. M34
- D7.2.1-3 Exploitation and Marketing Plan (release 3) - AIRBUS, Document, Public, T7.2 - Business Model & Exploitation Plan, WP7. #112 opened on Dec 7, 2014 by ferhaterata. M34
- D7.2.1-2 Exploitation and Marketing Plan (release 2) - AIRBUS, Document, Public, T7.2 - Business Model & Exploitation Plan, WP7. #111 opened on Dec 7, 2014 by ferhaterata. M22



Deliverable Management – Waffle.io

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



modelwriter/deliverables

+ Add Issue

Filter Board

Backlog 58

Ready 9

In Progress 7

Done 0

Done

Issues closed in the last week are shown in this column. Drag issues here to close them.

The Waffle.io board displays a grid of tasks organized into four columns: Backlog, Ready, In Progress, and Done. Each task card includes a unique ID, a brief description, and a list of associated tags. The 'Ready' and 'In Progress' columns contain several tasks related to ModelWriter, while the 'Backlog' and 'Done' columns contain tasks related to project reporting and compliance.

Column	Task ID	Description	Tags
Ready	132	D5.3.2 Updated State-of-the-art (Public Deliverable)	M37 Document Public T5.3 - Project Controls UNIT WP5
	131	D5.3.1 Final Project Report	M37 Document Private T5.3 - Project Controls UNIT WP5
	130	D5.2.5 Project Progress Report (fifth half year)	M31 Document Private T5.2 - Project Coordination and Reporting UNIT WP5
	129	D5.2.4 Project Progress Report (fourth half year)	M25 Document Private T5.2 - Project Coordination and Reporting UNIT WP5
	128	D5.2.3 Project Progress Report (third half year)	M19 Document Private T5.2 - Project Coordination and Reporting UNIT WP5
	123	D7.7.1 ModelWriter and standardization activities	M34 LORIA Public Software T7.7 - Standardization WP7
	115	D7.3.1 Newsletter - International Conference Announcement	M31 Document Public T7.3 - Workshops & Events UNIT WP5
	114	D7.2.2 Roadmap for future exploitation and pre-competition survey	M34 Document HISBIM Public T7.2 - Business Model & Exploitation Plan
	66	D4.6.2-1 Proof-of-concept semantic comparison engine (release 1)	M13 OBOE Public Software
	27	D2.5.2-1 Natural Language Generator and Documentation (release 1)	M16 LORIA Public Software
103	D6.7.1-1 ModelWriter major release (release 1)	M16 OBOE Public Software T6.7 - ModelWriter Integration WP6	
100	D6.6.1-1 Acceptance Test Procedures (release 1)	M14 Document KOCSYSTEM Public T6.6 - Acceptance Test Procedures WP5	
69	D4.7.1 Future ModelWriter-Enabled Use Cases	M14 Document MANTIS Public	
106	D6.8.1-1 Evaluation report (release 1)	M17 Document KOCSYSTEM Public	
2	D1.2.1 Industrial Use Cases for Belgian Consortium	M13 Document Public SOGETI	
59	D4.3.1-1 Knowledge Base (release 1)	M13 MANTIS Public Software T4.3 - Implementation of the Knowledge Base WP4	
23	D2.4.1 Specification of the Knowledge Representation Language	M14 Document LORIA Public	
127	D5.2.2 Project Progress Report (second half year)	M13 Document Private T5.2 - Project Coordination and Reporting UNIT WP5	
15	D1.7.1-1 Annual Product Owner Review - 1	M17 AIRBUS Document Public T1.7 - Annual Product Review WP1	
52	D3.8.1-1 Source code of each m2m transformation (release 1)	M13 Public Software	
64	D4.5.1 Technical Note for KB serialization and reuse	M13 Document MANTIS Public	
14	T4.5 - Knowledge Base serialization and reuse plug-in	WP4	



Source Code Management – GitHub, EGit

<https://github.com/ModelWriter>



Screenshot of the Eclipse IDE interface showing the GitHub integration for the ModelWriter repository.

The left side shows the Eclipse Java EE Perspective with the Package Explorer containing the ModelWriter project structure. The Problems view shows several merge conflicts and code review comments from contributors like Emre Kirmizi and Ferhat Erata.

The center of the screen displays the GitHub repository page for "ModelWriter / WP3". It shows a timeline graph of commits from Nov 2, 2014, to Dec 8, 2015. Below the timeline are four detailed commit history charts for contributors:

- emrekirmizi**: 168 commits / 38,894 ++ / 32,796 --
- serhat93**: 167 commits / 165,683 ++ / 31,735 --
- aniloztрук**: 146 commits / 86,187 ++ / 8,333 --
- ferhaterata**: 117 commits / 92,617 ++ / 106,433 --

The right side of the interface shows the Outline view, which lists the source code files and their traceability validation status. The "Interpreter.java" file is currently selected.



T5.2 Project Coordination and Reporting

Yearly ITEA review meetings



National review meetings (e.g. in every 6 month in Turkey)



International workshops: face to face meetings (in each 3 months)



Monthly International Telco meetings



Weekly collaboration meeting: e.g. UNIT-KoçSistem



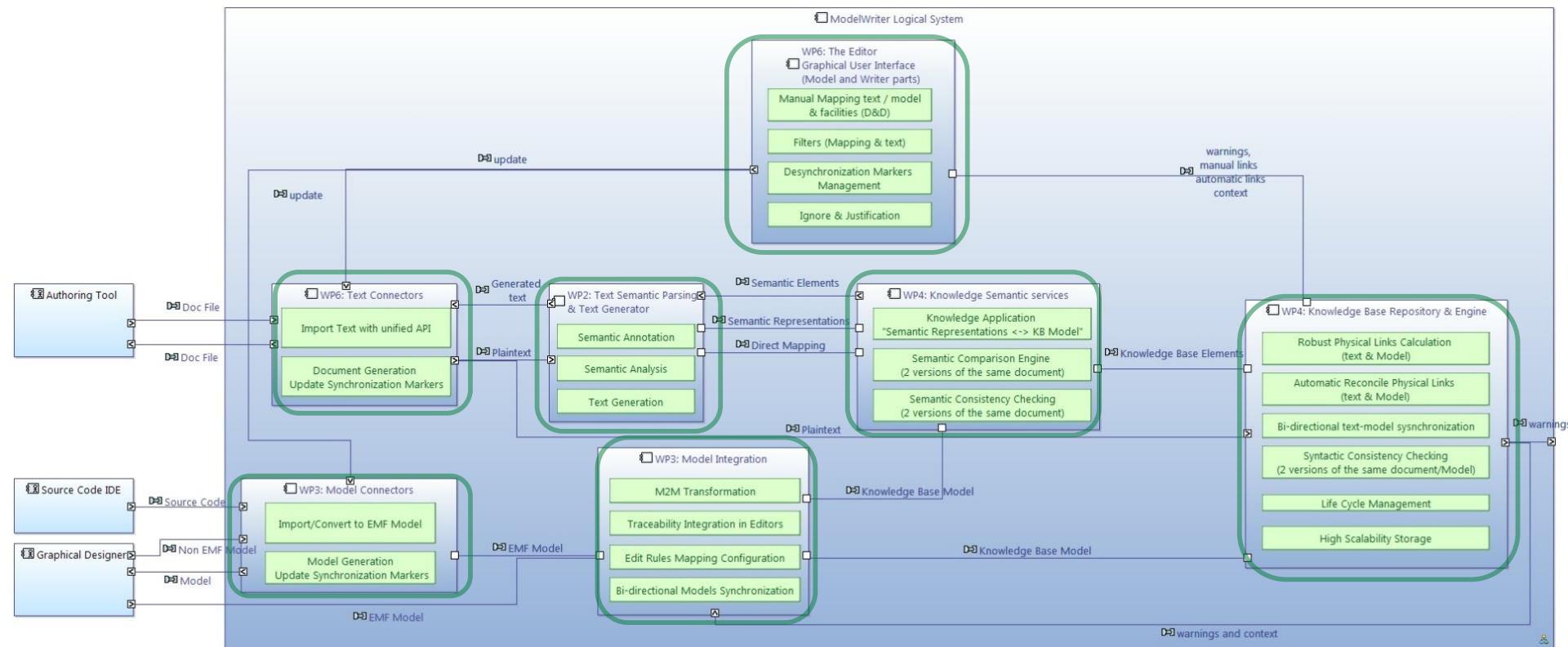
Action specific meetings: e.g. Airbus-UNIT action, UNIT-Havelsan ...

T5.3 – Project Controls

- Monitoring the deliverables
- Monitoring the requirements
- Decision mechanism for addressing issues, changes, risks
- Progress controls by means of regular meetings and workshops

WP6 – Architecture, Integration and Evaluation (OBEO)

Yvan lussaud
OBEO



- Github repository
- Checkstyle and code templates
- Target platforms

Source code



- A suitable tool will be selected (Jenkins)
- Ease the release process

Continuous Integration



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

Next steps



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 (OBEO)

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

- 6 workshops – 2 open workshops



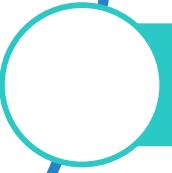
Parsing text into RDF

- 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

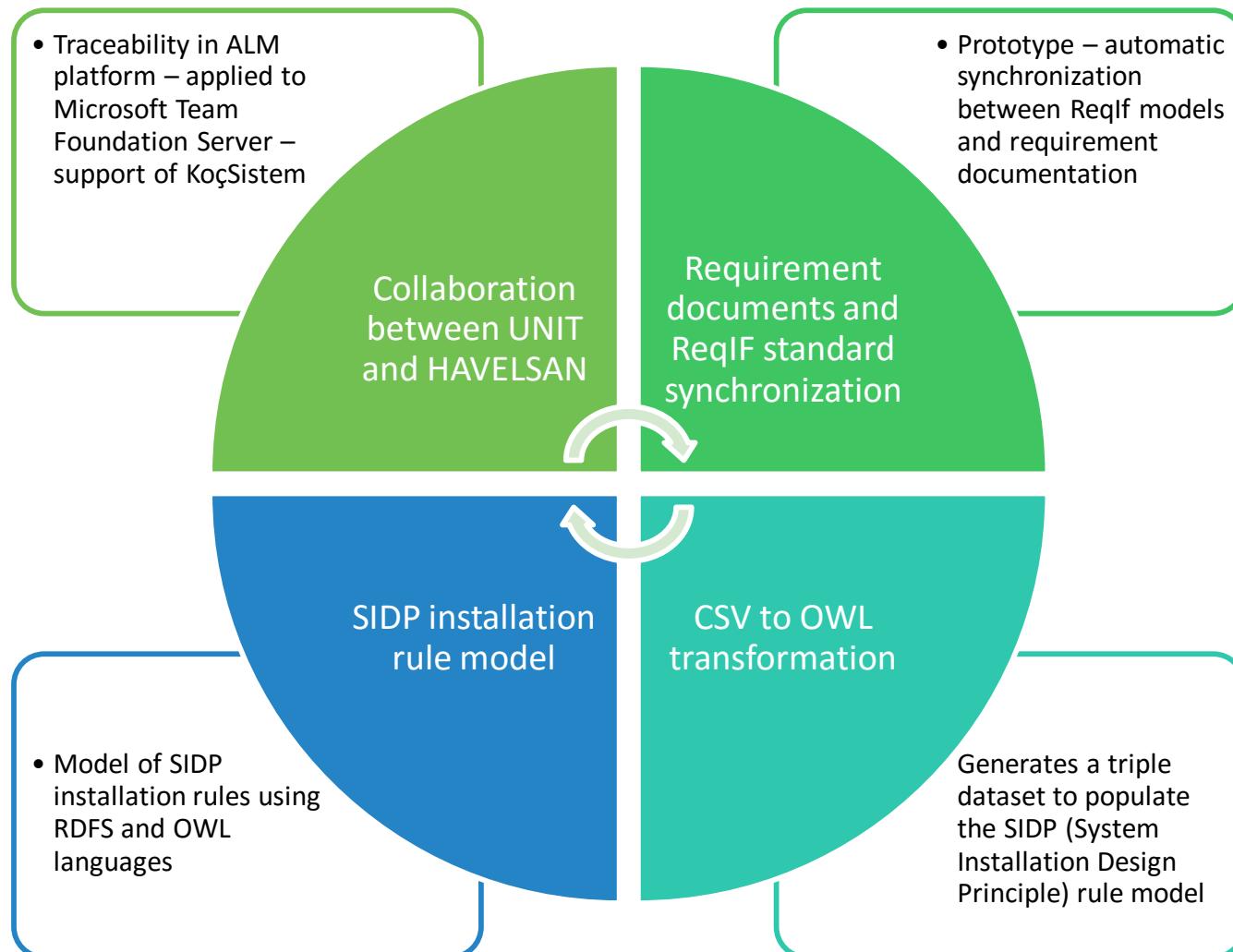
- Spanish Natural Language Processing Conference – academics and industrials – a project session



Keynote speech at International Workshop on Advanced Topic in Software engineering

- Present Eclipse ecosystem and Modeling approach to software engineering

WP7 Exploitation



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.**

4 Progress on the Industrial Use Cases

*Prof. Geylani Kardaş, Moderator
KoçSistem*

UC-TR-01 - Documents of Quality Assurance Department

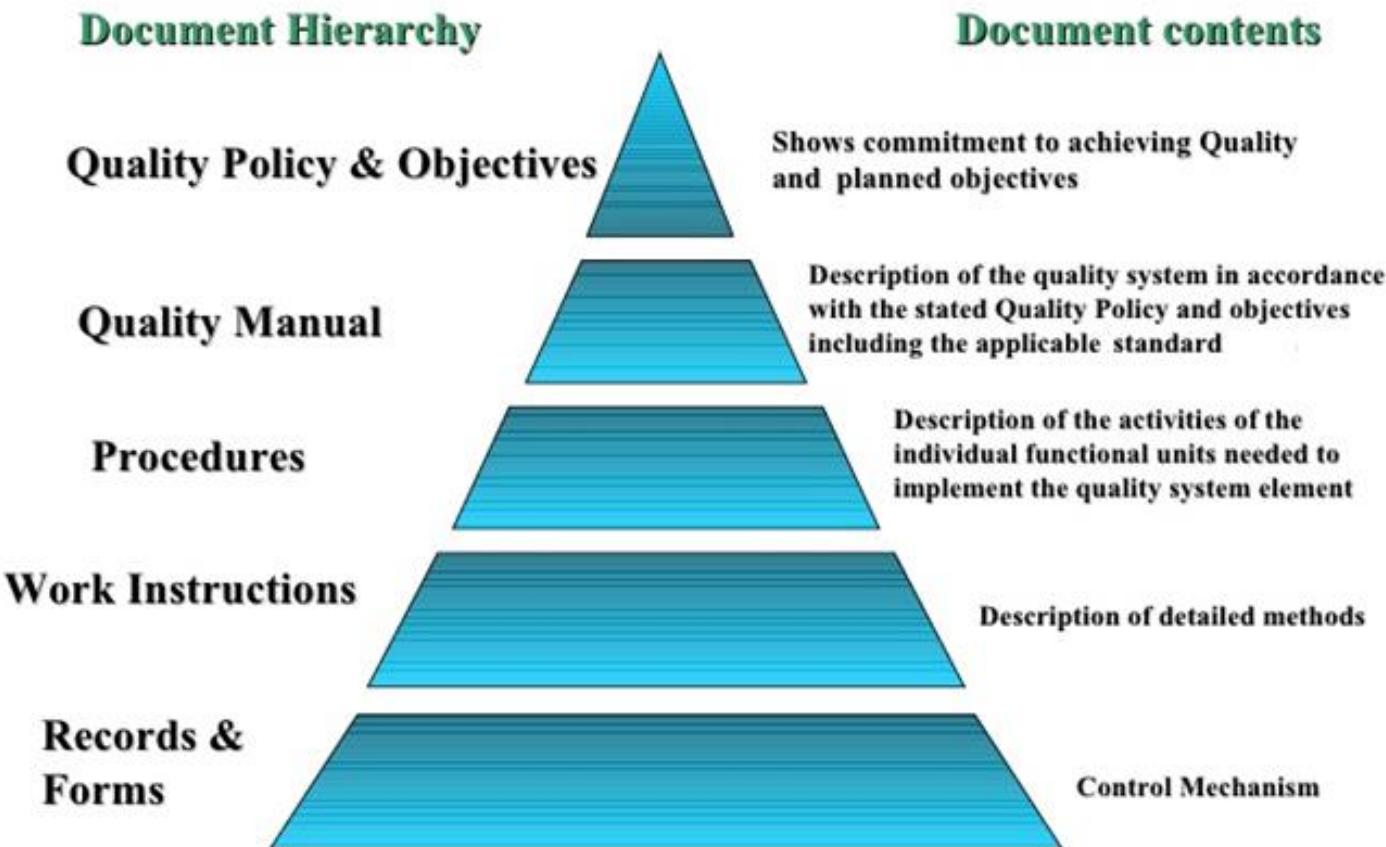
Ersan GÜRDOĞAN
HISBİM

T1.4 – Industrial Use Cases for Turkish Consortium

UC-TR-01	Documents of Quality Assurance Department
Version	V1.0.0 dated 15-Nov-2014
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 standard Rejected and Approved products forms are standard Quality Certification standards are always applied
Steps	Products information forms are created according to product information comes from ModelWriter system Products are measured by quality control department authorised personals according to standards To decide product is standard or not according to measurement report Product would be sent rejected products section if its measures are out of standards ModelWriter system is created Rejected Product Quality Form Else Approved Product Quality Form is created by ModelWriter System Approved products are sent to warehouse as accepted products
Alternatives	If rejected products' measurement out of range in rework standards, it means cannot be applied rework on this product, it returns to scrap. Then it should be sent wasteland.
Issues	Standardising form designs is hard because of all customers have their own unique reporting tools/formats

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. NDAs can be "mutual", meaning both parties are restricted in their use of the materials provided, or they can restrict the use of material by a single party. It is also possible for an employee to sign an NDA or NDA-like agreement with an employer. In fact, some employment agreements will include a clause restricting employees' use and dissemination of company-owned confidential information.</p>
Actors	Responsible/Authorised personnel in both parties.
Assumptions	Agreements are prepared and written according to European Business Law.
Steps	<ul style="list-style-type: none"> To define both parties who would sign the NDA To write items of agreement according to scope of NDA Reviewing agreement by decision maker then getting approval Sharing NDA each other Send feedback to ModelWriter system if any change applies on NDA Sharing final version of NDA then signing by both parties
Alternatives	<ul style="list-style-type: none"> After sharing NDA, both parties sign without any change and no feedback imzalanmasi. Cancellation of NDA
Issues	Different laws could be applied in and out of European Union

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

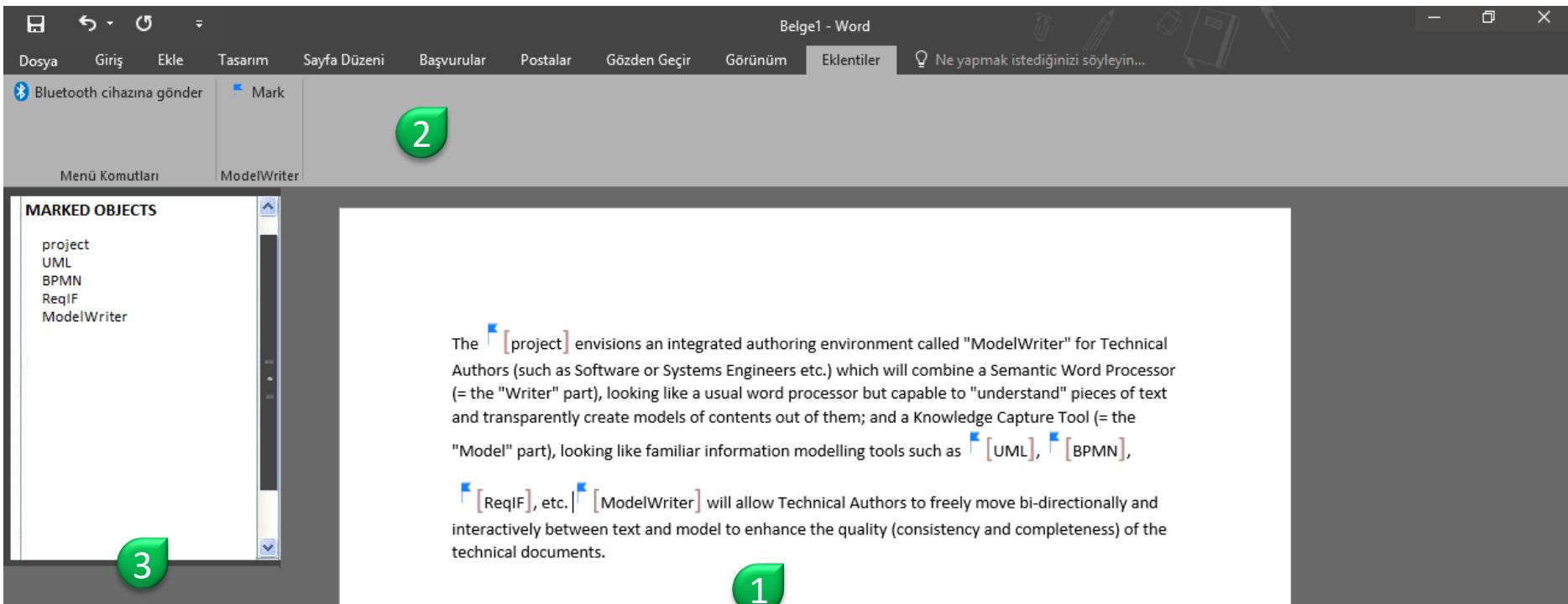
What we expect ModelWriter about NDAs:

- 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

ModelWriter Plug-in for Microsoft Word



- 1) Written text in main window of Ms-Word
- 2) Ribbon in upper menu tabs
- 3) Action Pane on left-hand side

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

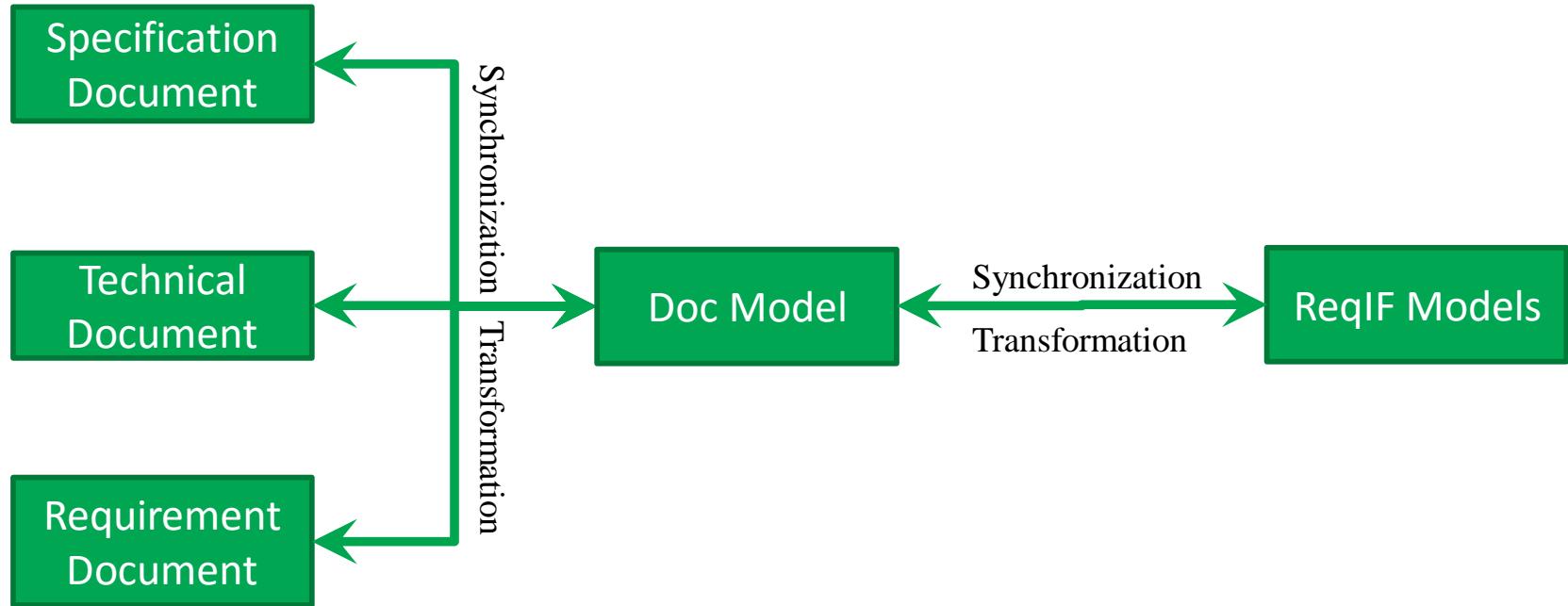
*Moharram Challanger
UNIT- KoçSistem*

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 eğitmeni 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" yetkilendirmeleri yapılacaktır.
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 sağlanacaktır.
R 1.1.2.9	Nesneleri görüntüleme ve raporlama yetkisine sahip "Görüntüleyici" yetki profili sağlanacaktır.
R 1.1.2.10	Yetki profillerinin kullanıcılara/gruplara atanmasına olanak sağlanması
R 1.1.2.11	Kullanıcıların/grupların dosya/dizinler üzerinde seçilebilecek yetkiler ile yetkilendirme yapılabilir.
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 sağlanacaktır.

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.

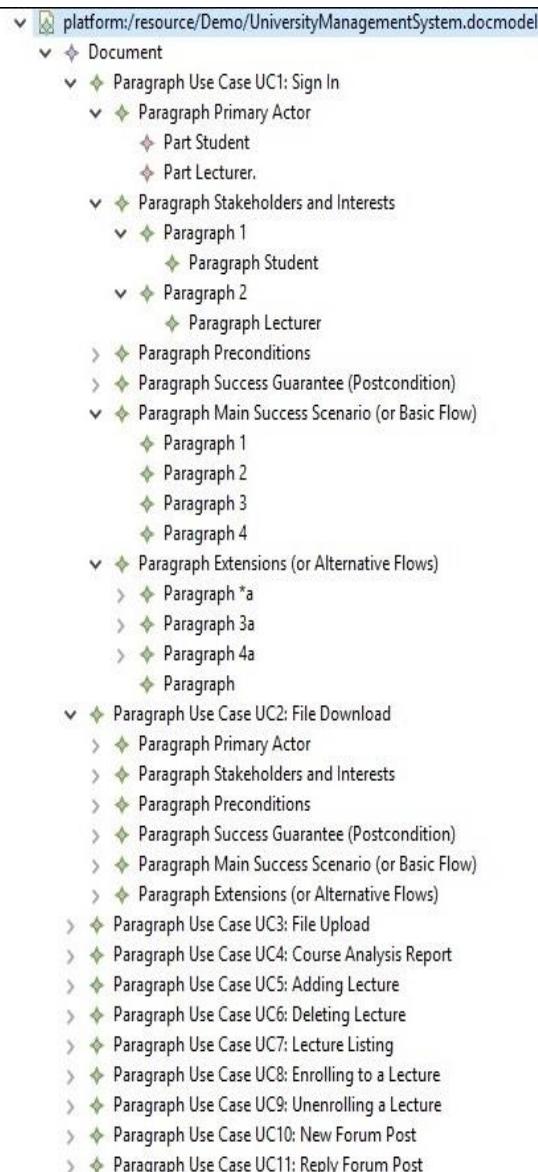
Main Success Scenario (or Basic Flow):

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.

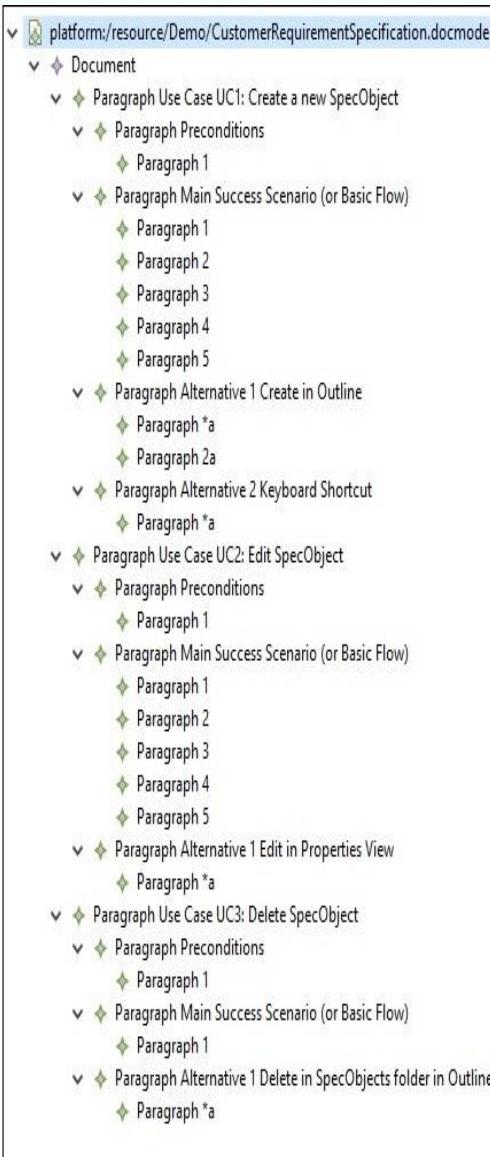
2 USE CASE UC2: EDIT SPECOBJECT

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)

UNIT - KoçSistem

Requirement Work Item

Customized Form

Attributes

WorkItem Number Patterns

History

Discussion

Requirement 4878*: Üniversitenin tüm akademik birimlerini hiyerarşik olarak görebileceklerdir.

Code Title

YGB_BLG_001

Üniversitenin tüm akademik birimlerini hiyerarşik olarak görebileceklerdir.

DESCRIPTION ANALYSIS STORYBOARDS ALLOCATIONS LINKS ATTACHMENTS HISTORY PLAN OTHER

CLASSIFICATION

Area YGO Faitim

New

Proposed

Yagup MACİT

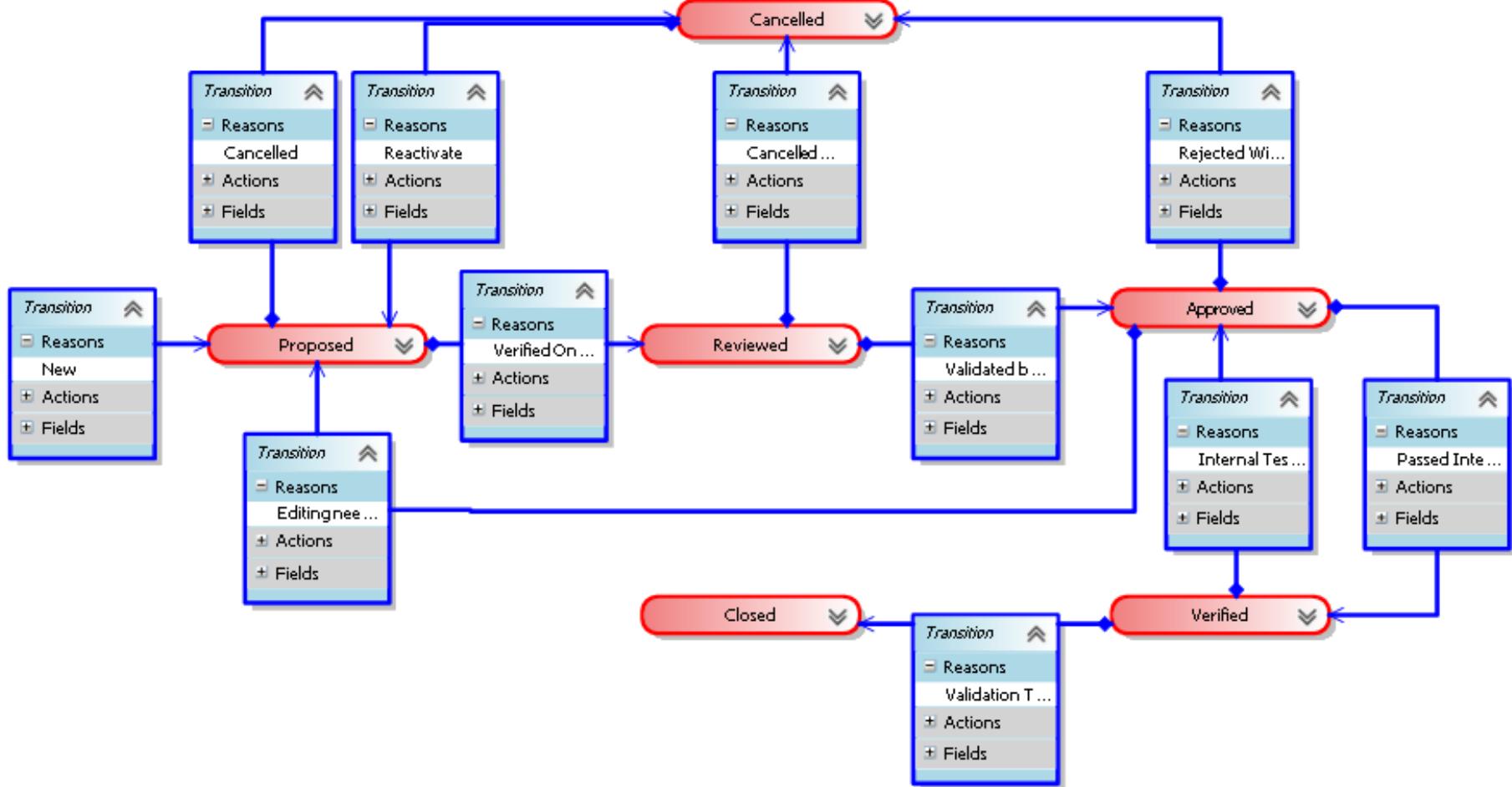
7/26/2012

DISCUSSION ONLY ALL CHANGES

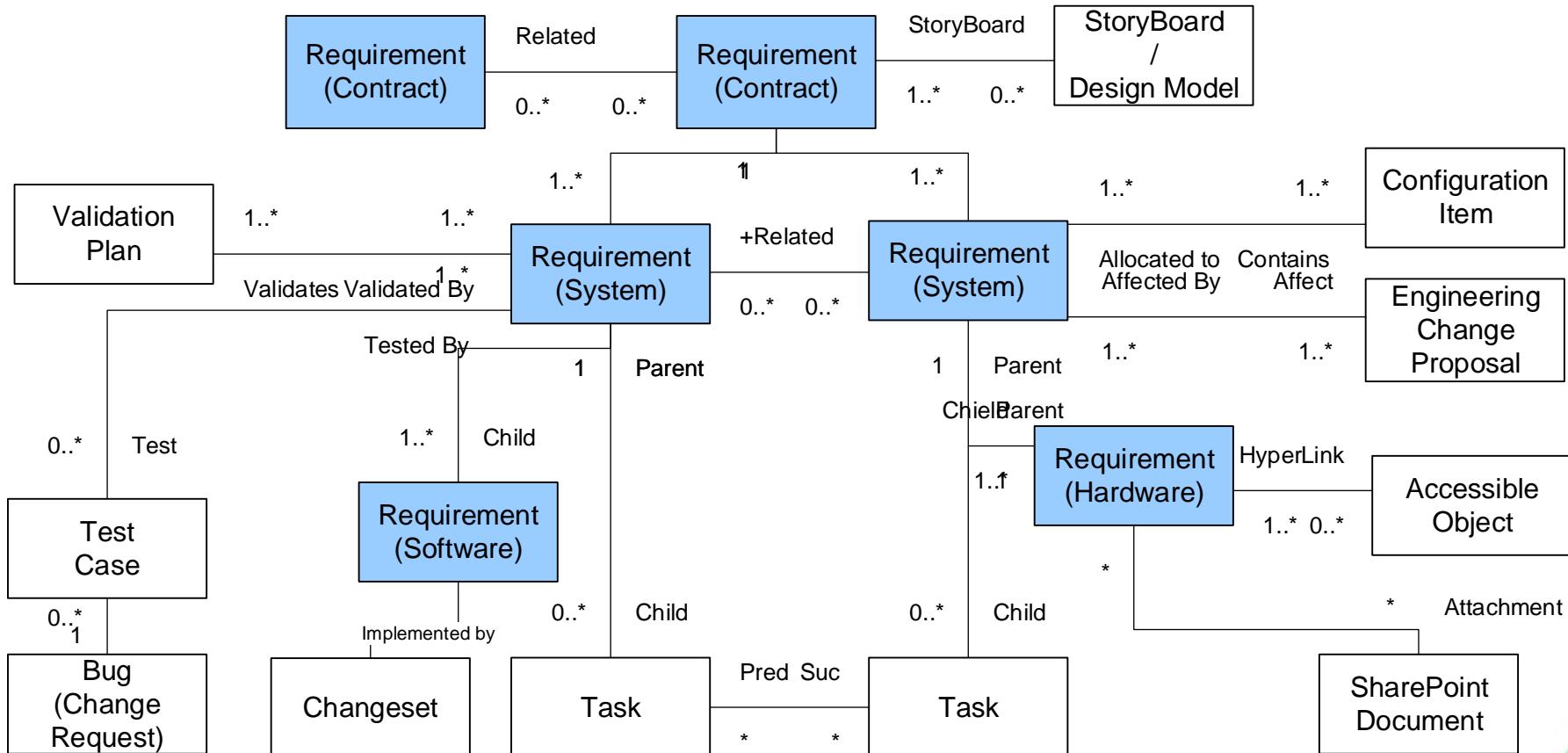
- ▶  Kürsat İNCE made field changes (2 weeks ago)
- ▶  Kürsat İNCE made field changes (2 weeks ago)
- ▶  Nahit Fırat DOĞAN added link (3 months ago)
- ◀  Yagup MACİT created the Requirement (8 months ago)
 - ▶ Fields

Field	New Value
Iteration Path	UYY\Sürüm 2\Tur 4
Iteration ID	20
Team Project	UYY
Node Name	II. Faz - Gereksinim Yönetimi (A-H91201.02.10)
Area Path	UYY\YGO (A-H91201.02)\II. Faz - Gereksinim Yönetimi (A-H91201.02.10)
Area ID	84

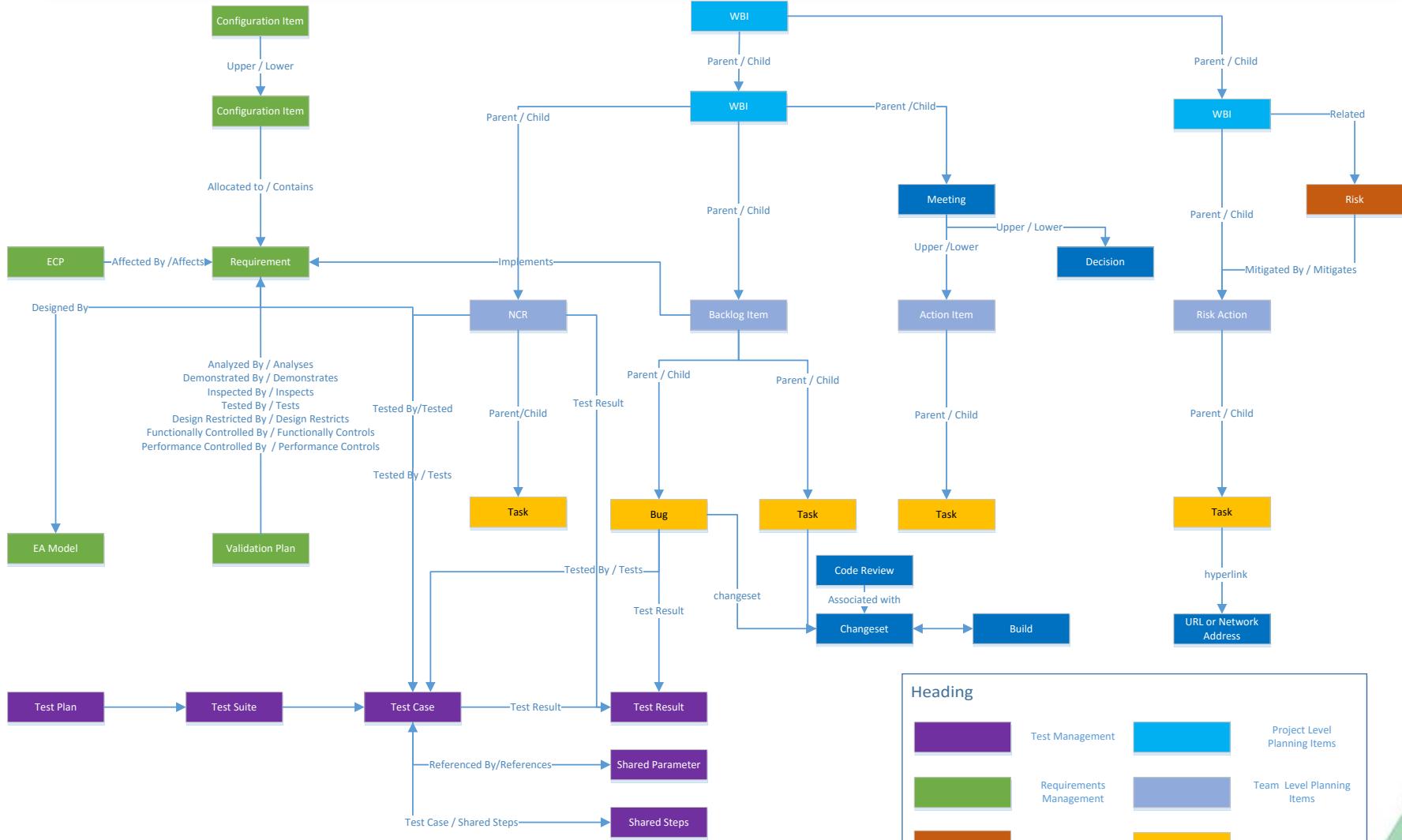
Requirement LifeCycle



Requirements Traceability



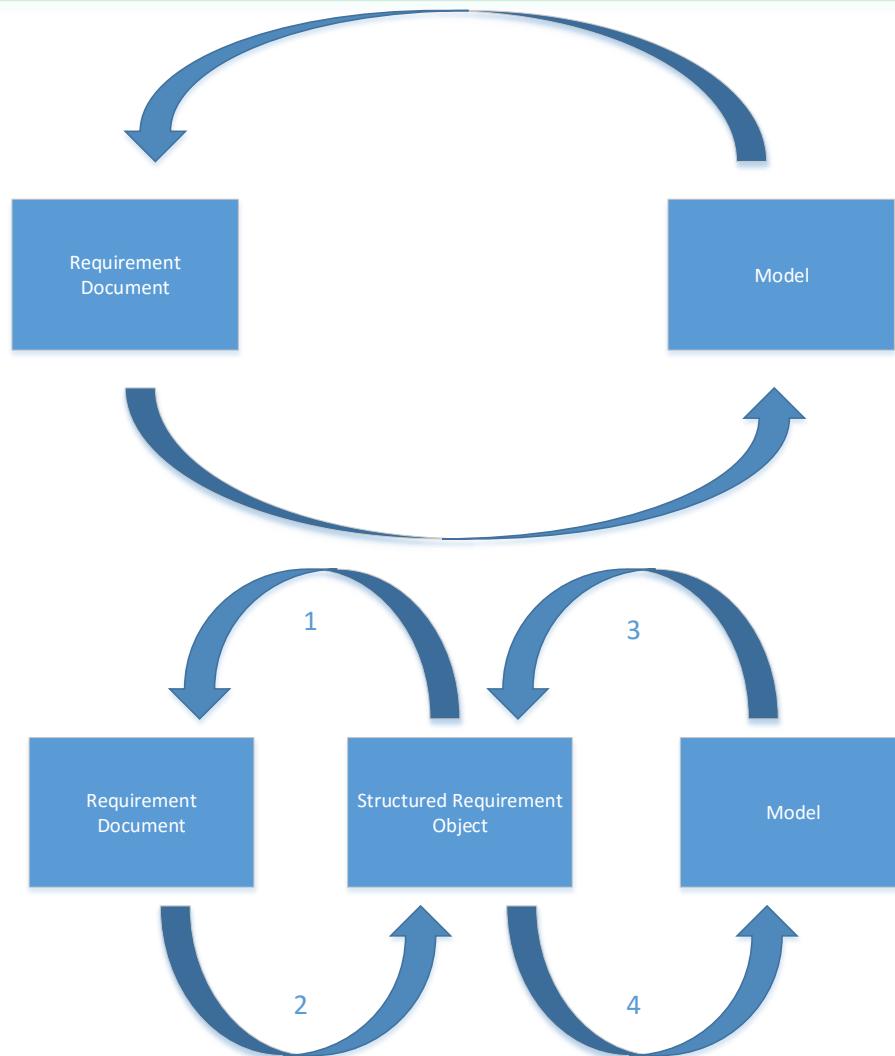
Requirements in ALM



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



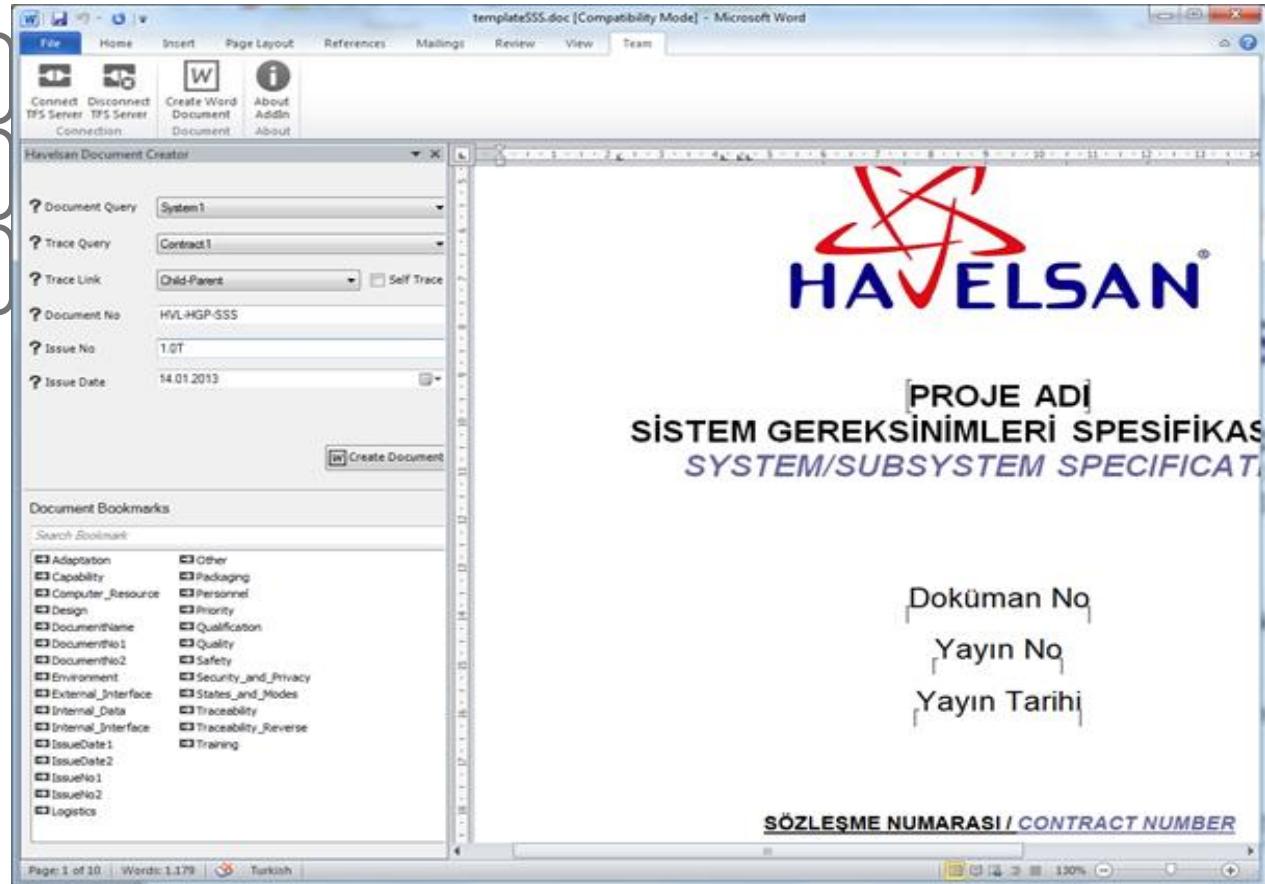
Currently we support scenario #1 and #4, and interested in Scenario #3, #2

Havelsan Ext - Document Generation

Preview

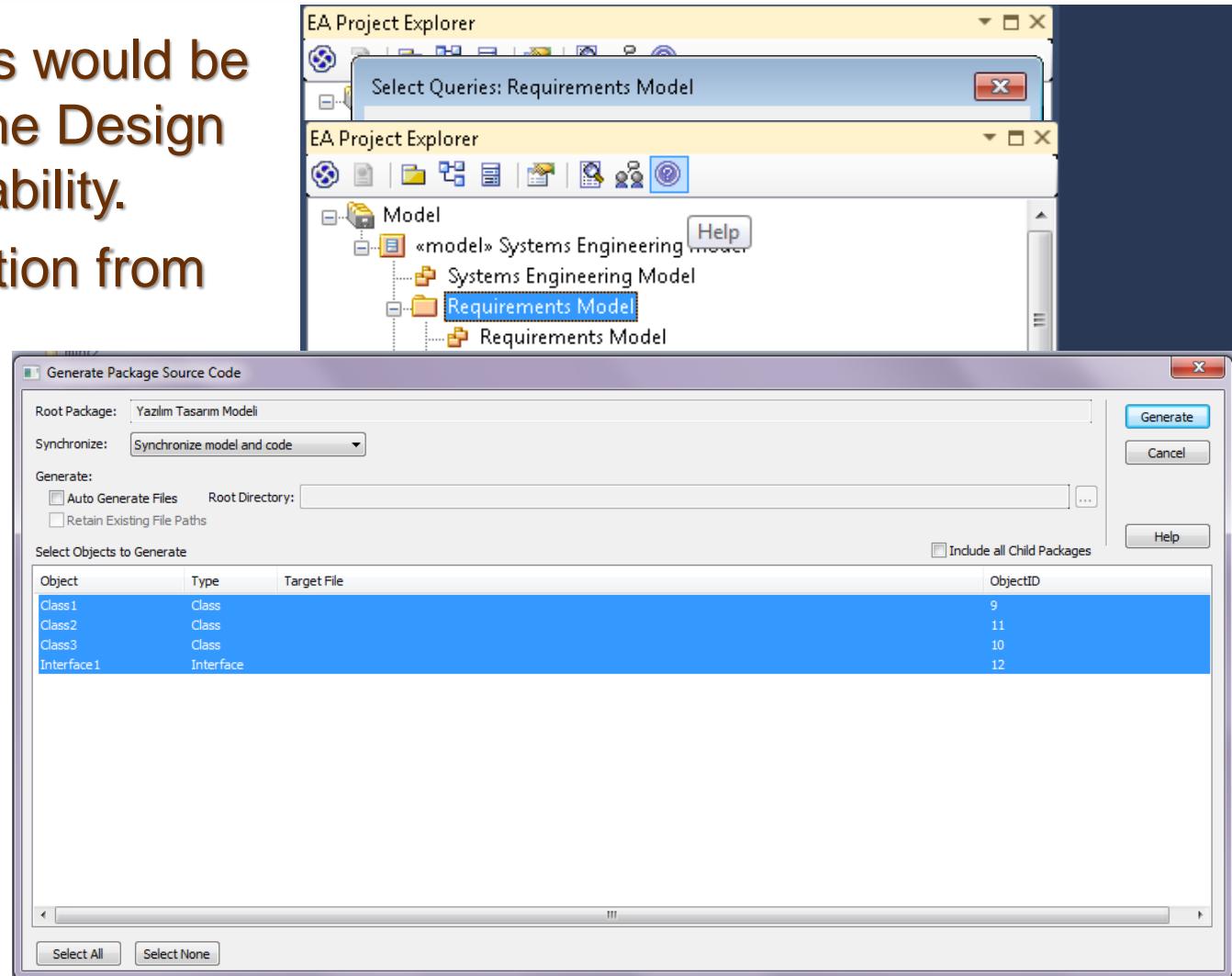
Word Extension

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, UNIT

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:

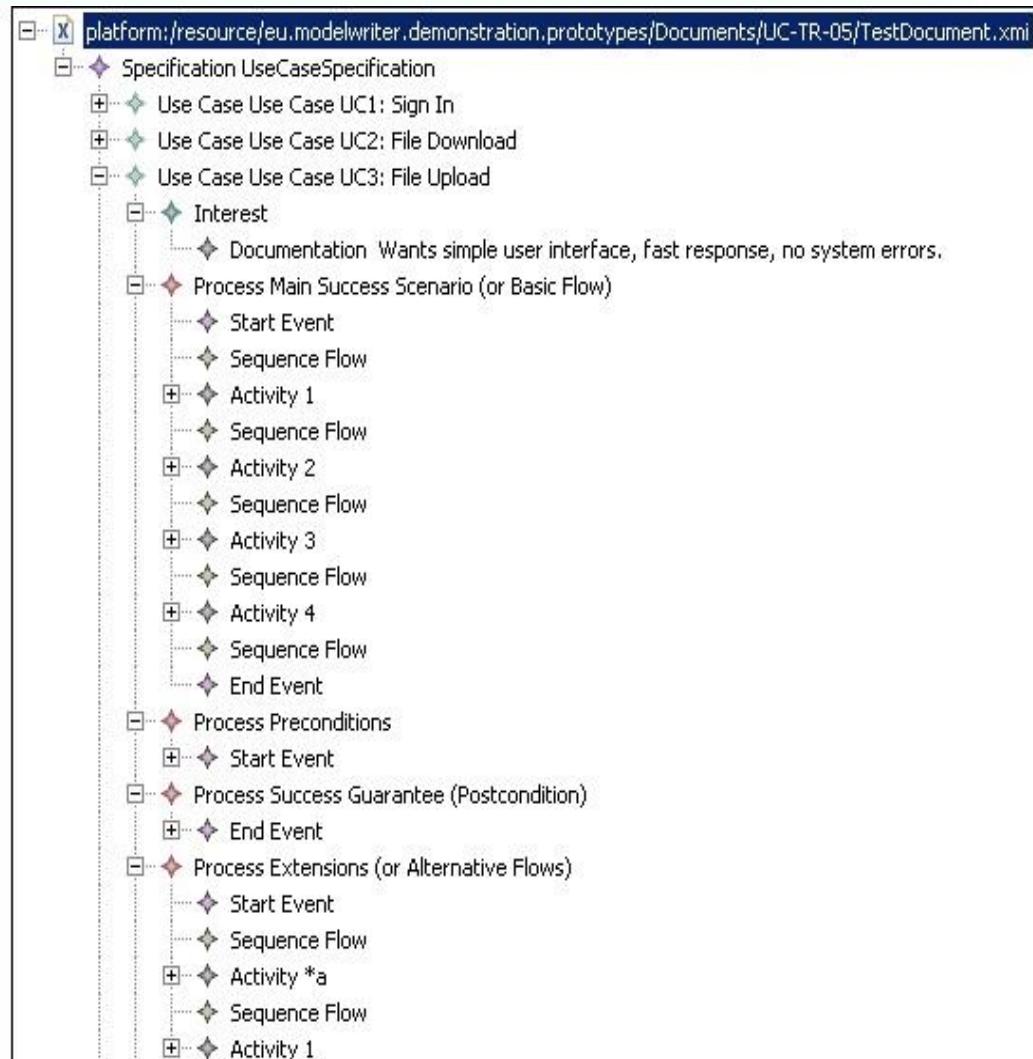
1. System signals error to the Student, records the error, and enters a clean state.

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.

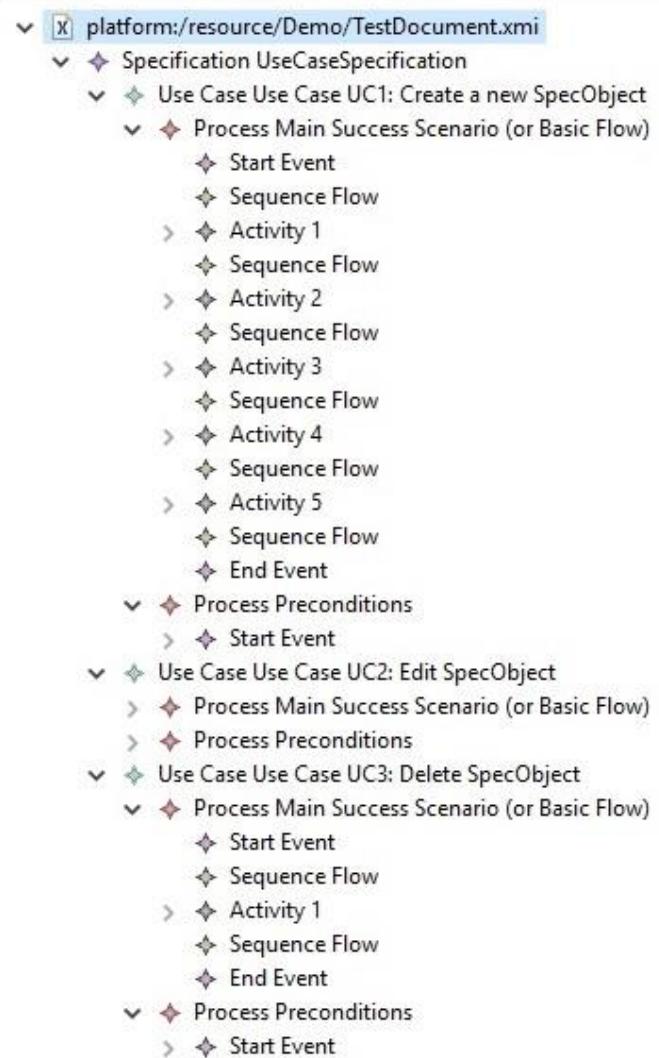
Alternative 2 Keyboard Shortcut:

- *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.**

5 Demonstrations

Ferhat Erata (UNIT, ModelWriter Project Leader)

Dr. Mariem Mahfoudh (CNRS/LORIA)

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.**