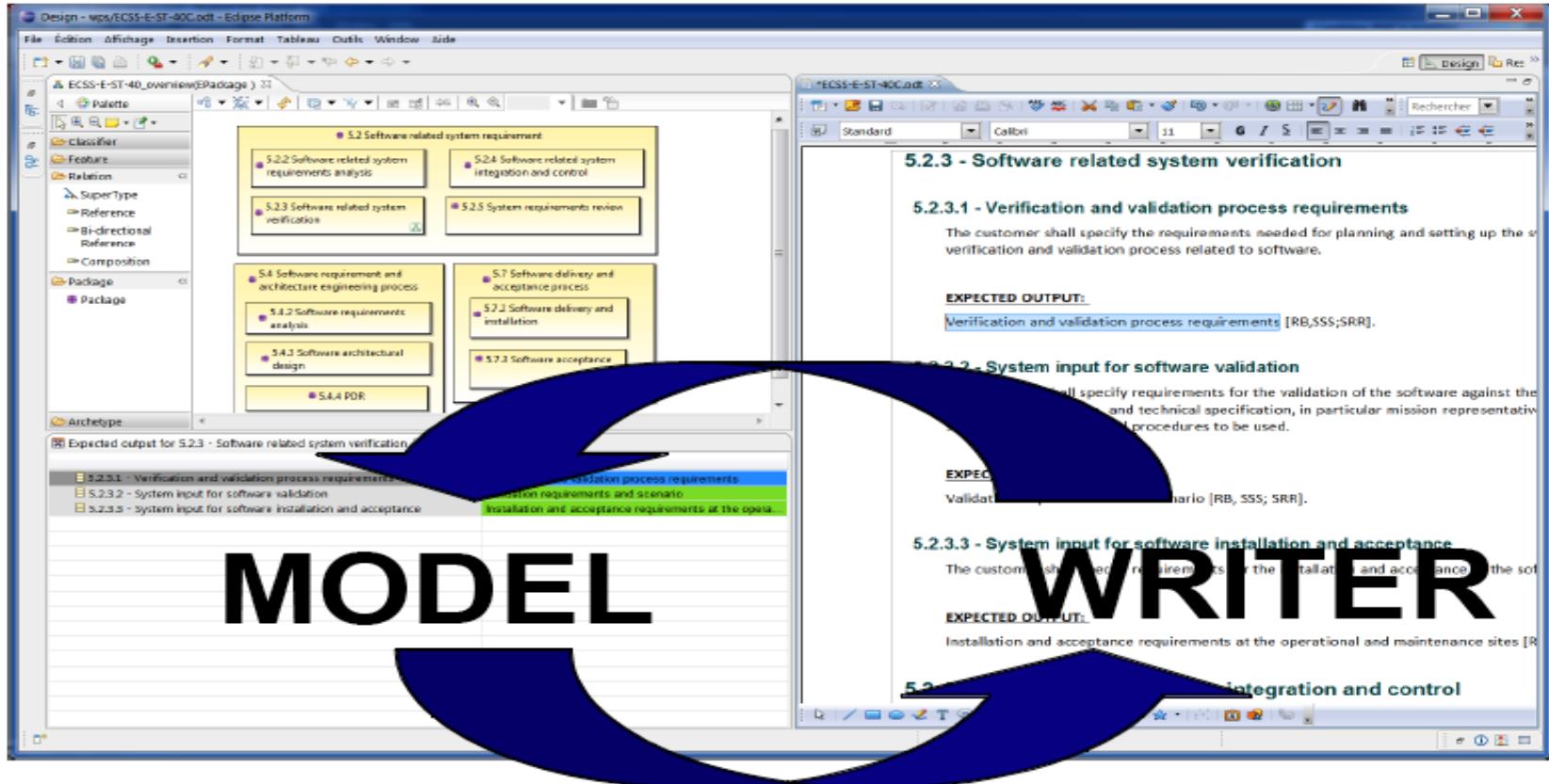


ModelWriter

Text & Model-Synchronized Document Engineering Platform

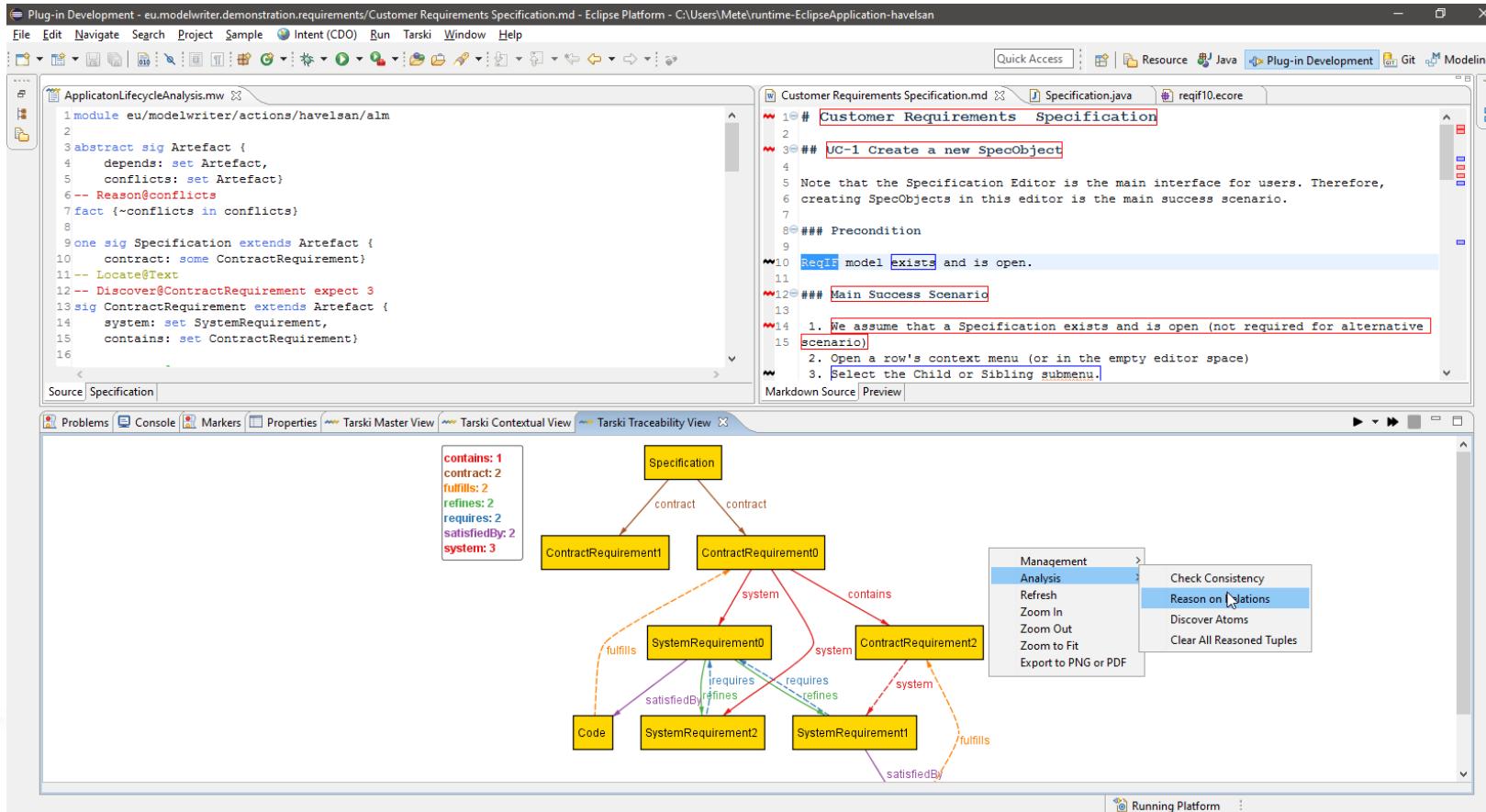


Project Leader: Ferhat Erata (ferhat@computer.org)

Project Email: project@modelwriter.eu

ModelWriter

Text & Model-Synchronized Document Engineering Platform



Project Leader: Ferhat Erata (ferhat@computer.org)
Project Email: project@modelwriter.eu

1 Introduction

*Ferhat Erata
UNIT, ModelWriter Project Leader*

Participants

ModelWriter #1 Review



UNIT

- Ferhat Erata
- Dr. Moharram Challenger
- Hasan Emre Kirmizi



- Dr. Claire Gardent
- Dr. Samuel Cruz-Lara
- Dr. Bikash Gyawali
- Anastasia Shimorina



- Dr. Anne Monceaux



- Yvan Lussaud
- Etienne Juliet



- Tuğçe Yiğiter
- Alan Endersoy
- Dr. Emrah Kinav



- Dr. Geylani Kardas
- Mehmet Onat
- Hale Gezgen



- Dr. Eray Tuzun
- Yagup Macit



Software Company

- Dr. Guven Köse
- Omurhan Soysal
- Aydin Can Polatkan



- Ersan Gürdoğan
- Taskin Kızıl
- Oguz Yavuz

Agenda [13:00 - 17:00]

1. Introduction (5 mins) [13:00 - 13:05]
2. Overview of the project (WP5) (15 min) [13:05 - 13:20]
3. Industrial Use Cases (WP1) (30 min) [13:20 - 13:50]
4. Exploitation Related Activities (WP7) (10 mins) [13:50 - 14:00]
5. Technical Innovations (WP2, 3 & 4) (30 mins) [14:00 - 14:30]
6. User Interfaces & Integration (WP6) (10 mins) [14:30 - 14:40]
 - Break (20 mins) [14:40 - 15:00]
6. Demonstrations (60 mins) [15:00 - 16:00]
7. Conclusions (10 mins) [16:00 - 16:10]
 - Reviewers' Private Section (30 min) [16:10 - 16:40]
 - Feedback Session (15min) [16:40 - 16:55]
 - Final Words (5 min) [16:55 - 17:00]

2 Overview of the Project

Ferhat Erata

UNIT, ModelWriter Project Leader

Industrialization Triangle in ModelWriter

Open Source Software (year #1)



UNIT

OBEO

Mantis
Software Company

New Product & Services
Commercialization
Open Source Software
Standardization



Products
&
Expertise

SME

Industrialization

Technology
Transfer

Industrial Use Cases
Success Stories
Long Term Agreements

**Large
Organization**

Inject
Requirements

ModelWriter

Prototypes

Researchers

New Standard or Standard Extension
Publications, Open Source Software

Innovation

AIRBUS
GROUP

HAVELSAN

KoçSistem

SOGETI

HISBIM

CNRS
 Universiteit
Antwerpen

KULEUVEN

UCLOUVAIN
UNIVERSITÉ
1817

Industrialization Triangle in ModelWriter



Open Source Software (year #2)

UNIT

OBEO

Mantis
Software Company

KocSistem

HSBIM

New Product & Services
Commercialization
Open Source Software
Standardization



Products
&
Expertise

SME

Industrialization

Technology
Transfer

Industrial Use Cases
Success Stories
Long Term Agreements

Large
Organization

Inject
Requirements

ModelWriter

Prototypes

Researchers

New Standard or Standard Extension
Publications, Open Source Software

Innovation

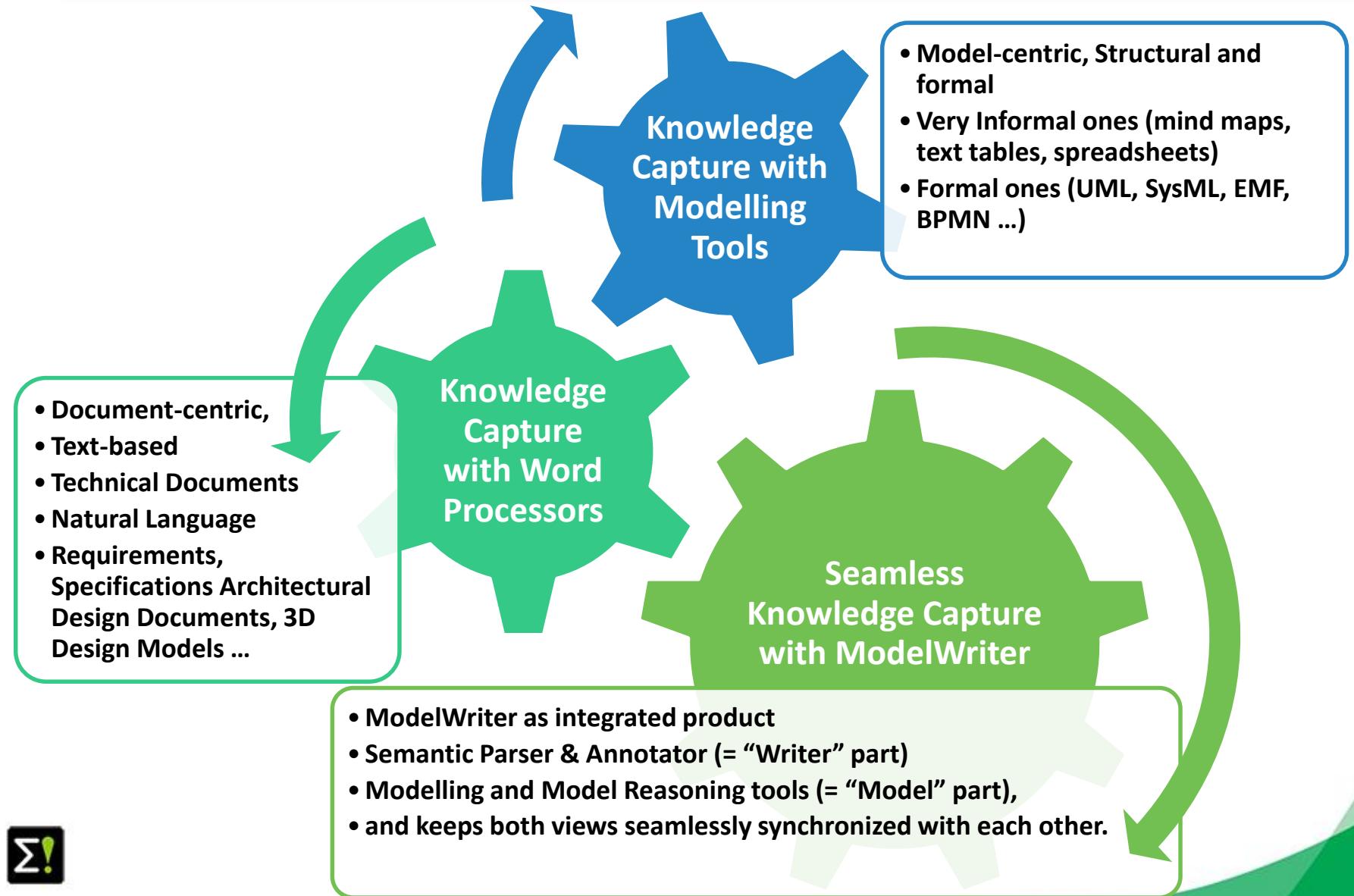
AIRBUS
 HAVELSAN

FORD OTOSAN

Cnrs
 Universiteit
Antwerpen

UNIT

**Ecole
Universitaire
de
Bruxelles
1855**



ModelWriter

bi-directional Knowledge Capture tool



The screenshot displays the ModelWriter application interface. On the left, a model editor window titled "Design - wps/ECSS-E-ST-40C.odt - Eclipse Platform" shows a hierarchical structure of requirements. The main pane contains several requirement groups:

- 5.2 Software related system requirement
 - 5.2.2 Software related system requirements analysis
 - 5.2.3 Software related system verification
 - 5.2.4 Software related system integration and control
 - 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 arrow points from the word "MODEL" to the "5.2.3 Software related system verification" section of a document viewer window.

The document viewer window is titled "ECSS-E-ST-40C.odt" and shows the following content:

5.2.3 - Software related system verification

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].

Integration and control

What is the problem?



Industrial Use Case in Airbus

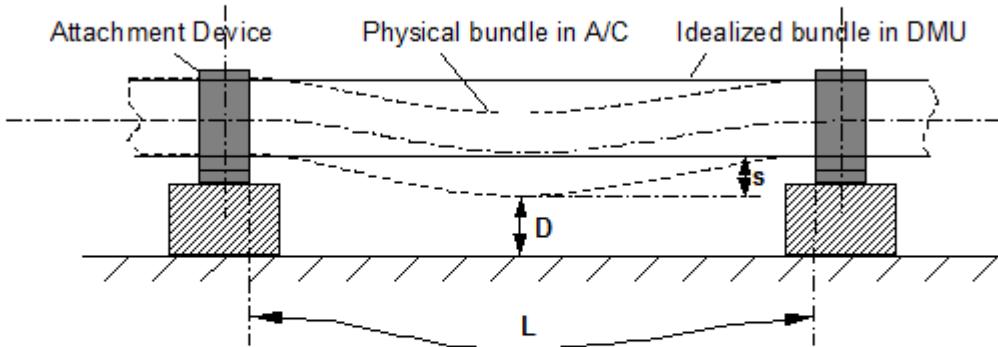


Synchronization of regulation documentation with a design rule repository

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)

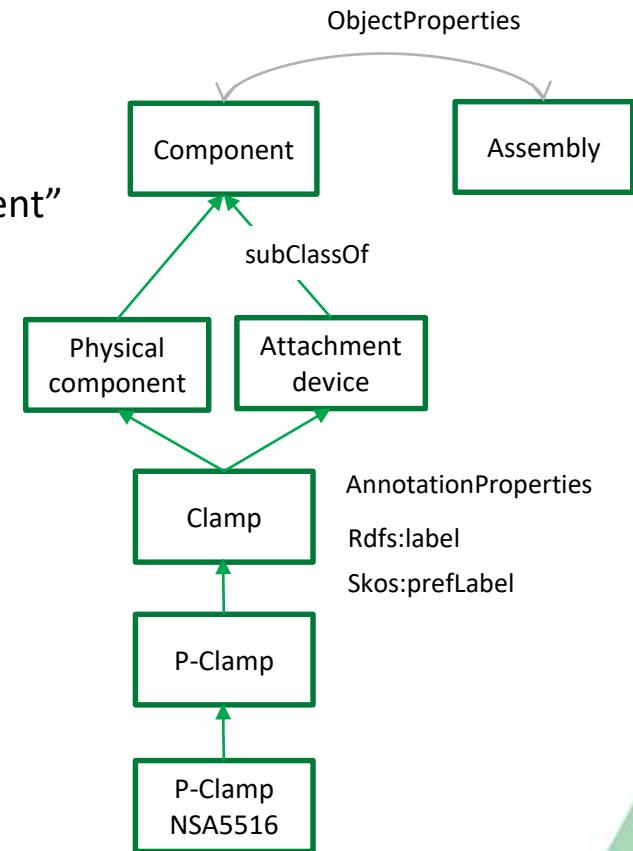
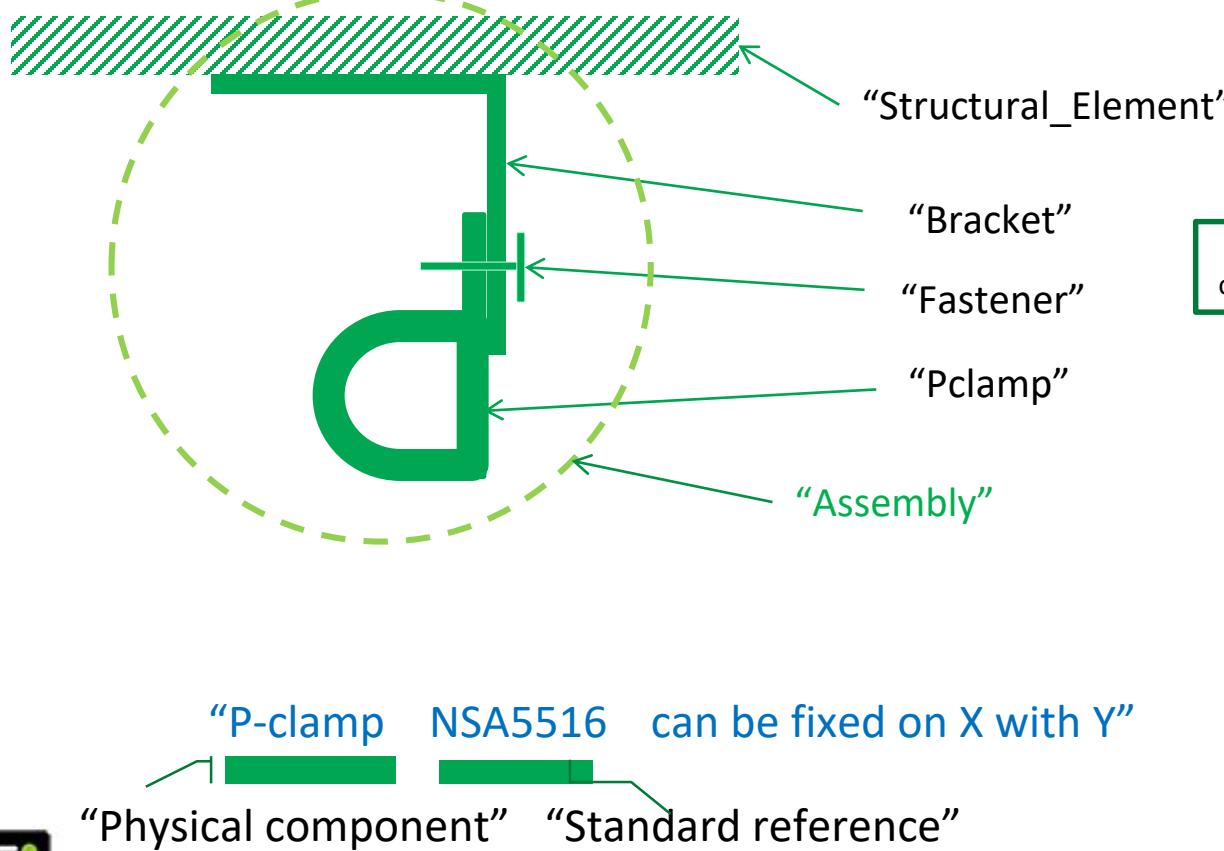
Figure 6: Sagging of bundles between attachment points

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

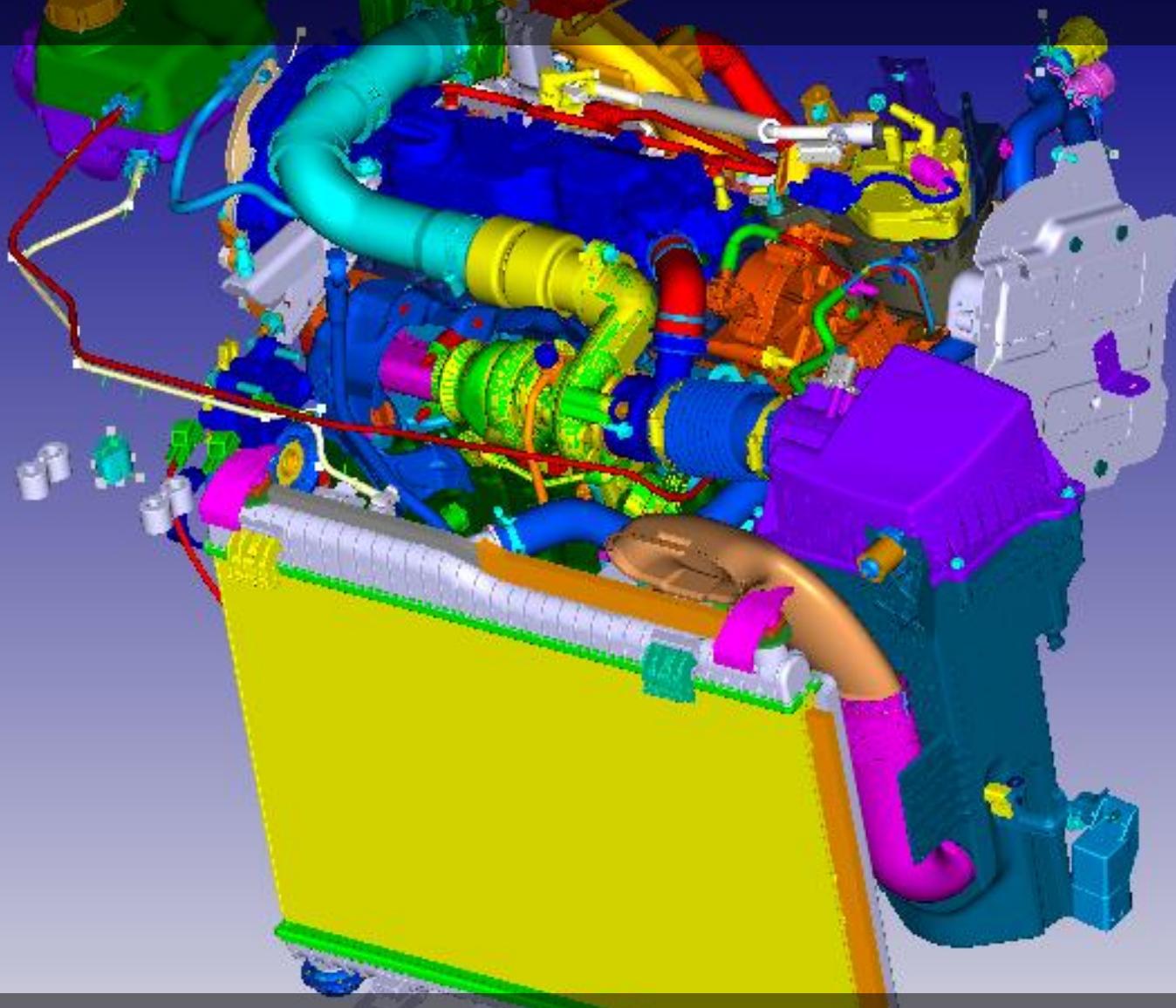
Component Ontology and Rules

Objectives:

- Manage rules/design principles and improve traceability
- Automate identification of design conflicts against rules



Industrial Use Case in Ford Otosan



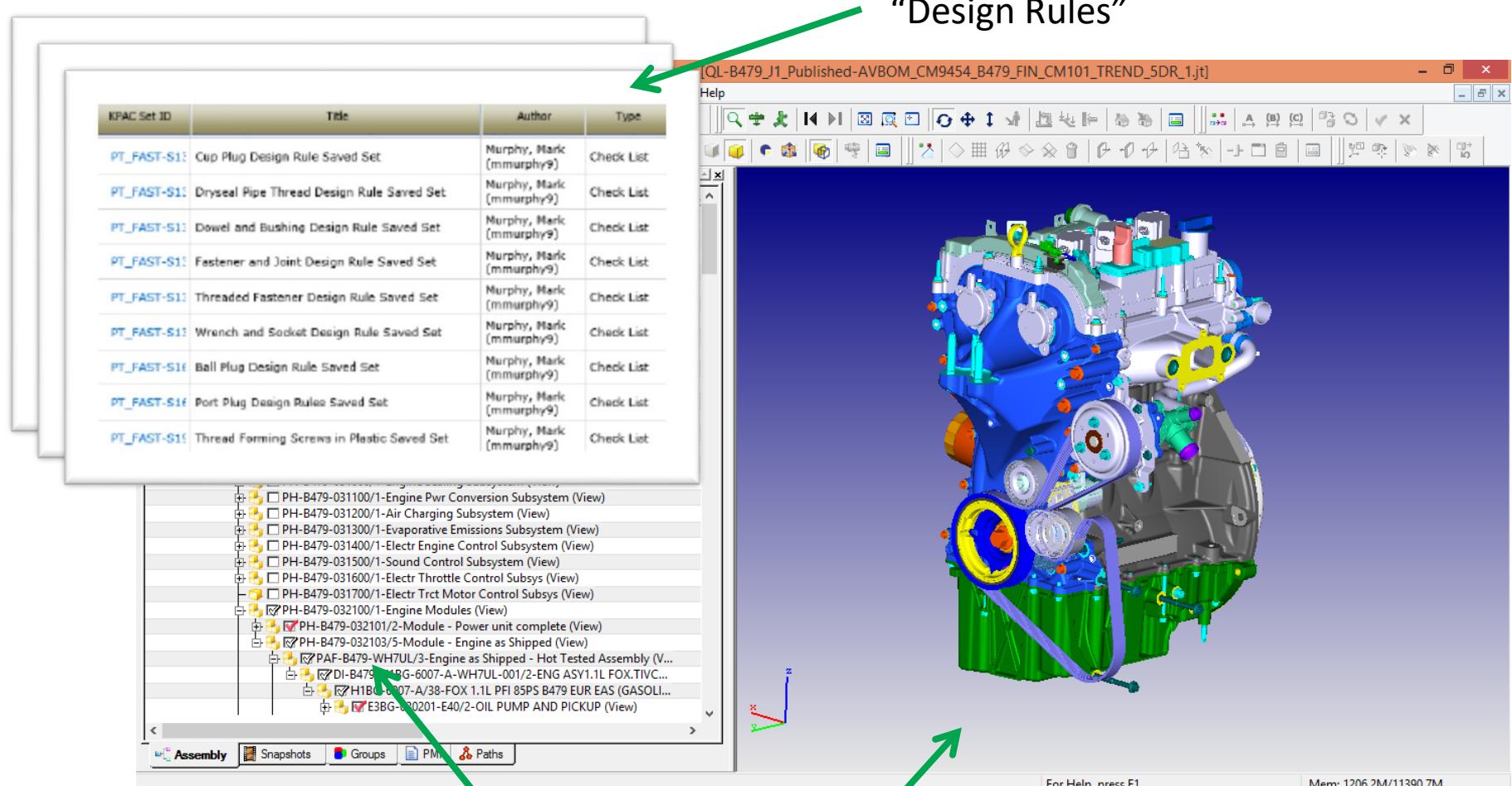
Synchronization of Design Specifications with Computer
Aided Design Data in Product Lifecycle Management

BOM and Design Specifications

“Design Rules”

“Bill of Material Data”

“Computer-aided Design Data”



The screenshot shows a CAD application window titled 'IQL-B479_J1_Published-AVBOM_CM9454_B479_FIN_CM101_TREND_5DR_1.jt'. The interface includes a toolbar at the top, a 3D view of a blue and grey engine assembly in the center, and a detailed Bill of Material (BOM) tree on the left. A large table at the top lists various design rules (KPAC Set ID, Title, Author, Type). A green arrow points from the 'Design Rules' label to the table. Another green arrow points from the 'Bill of Material Data' label to the BOM tree. A third green arrow points from the 'Computer-aided Design Data' label to the 3D engine model.

KPAC Set ID	Title	Author	Type
PT_FAST-S1:	Cup Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Dryseal Pipe Thread Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Dowel and Bushing Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Fastener and Joint Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Threaded Fastener Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Wrench and Socket Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Ball Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Port Plug Design Rules Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S1:	Thread Forming Screws in Plastic Saved Set	Murphy, Mark (mmurphy9)	Check List

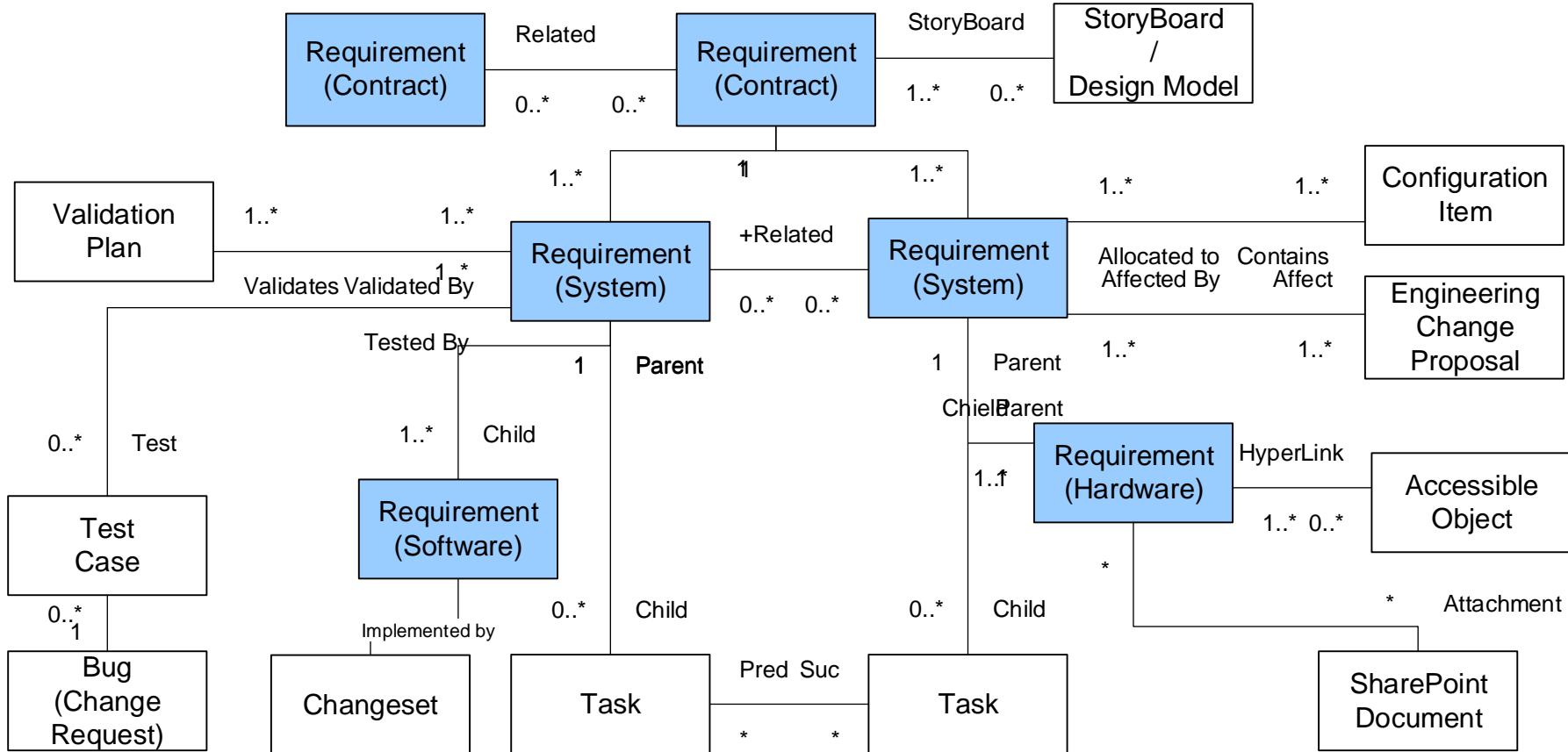
IQL-B479_J1_Published-AVBOM_CM9454_B479_FIN_CM101_TREND_5DR_1.jt

Help

Assembly Snapshots Groups PM Paths

For Help, press F1 Mem: 1206.2M/11390.7M

Industrial Use Case in Havelsan

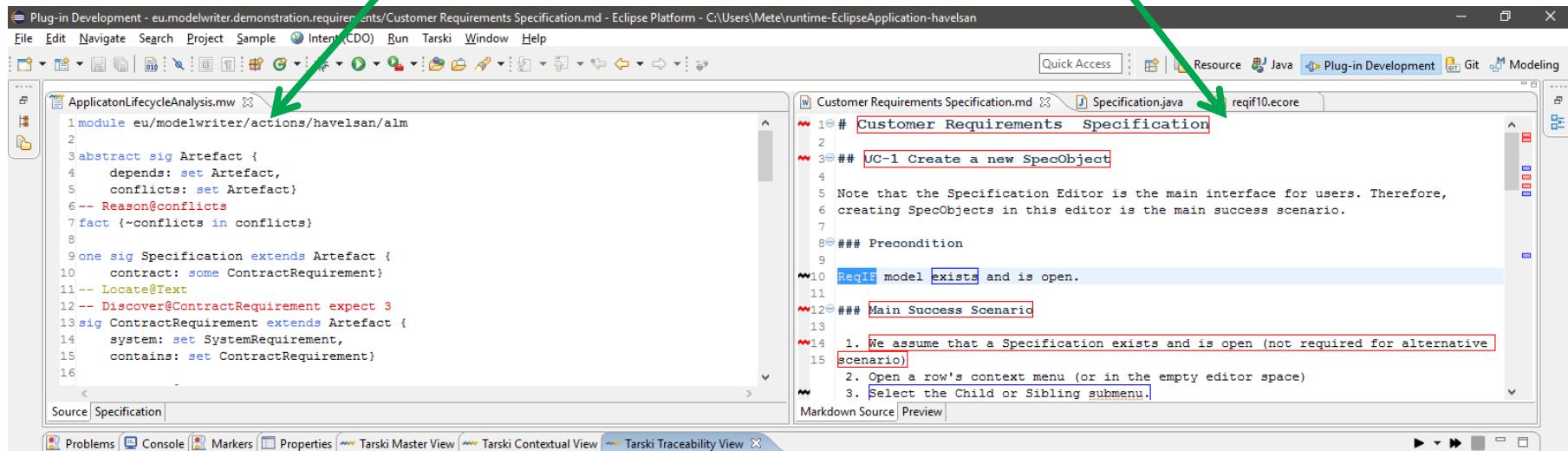


Integration with Application Lifecycle Management to ensure reliability and consistency in the system under development.

Automated Analysis of Dynamically Configured Traceability Semantics

“Traceability Rules to define traceability semantics”

Artefacts or part of artefacts



Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metz\runTime-EclipseApplication-havelsan

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

Customer Requirements Specification.md Specification.java reqif10.ecore

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact)
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10   contract: some ContractRequirement)
11 -- Locate@Text
12 -- Discover@ContractRequirement expect 3
13 sig ContractRequirement extends Artefact {
14   system: set SystemRequirement,
15   contains: set ContractRequirement)
16
```

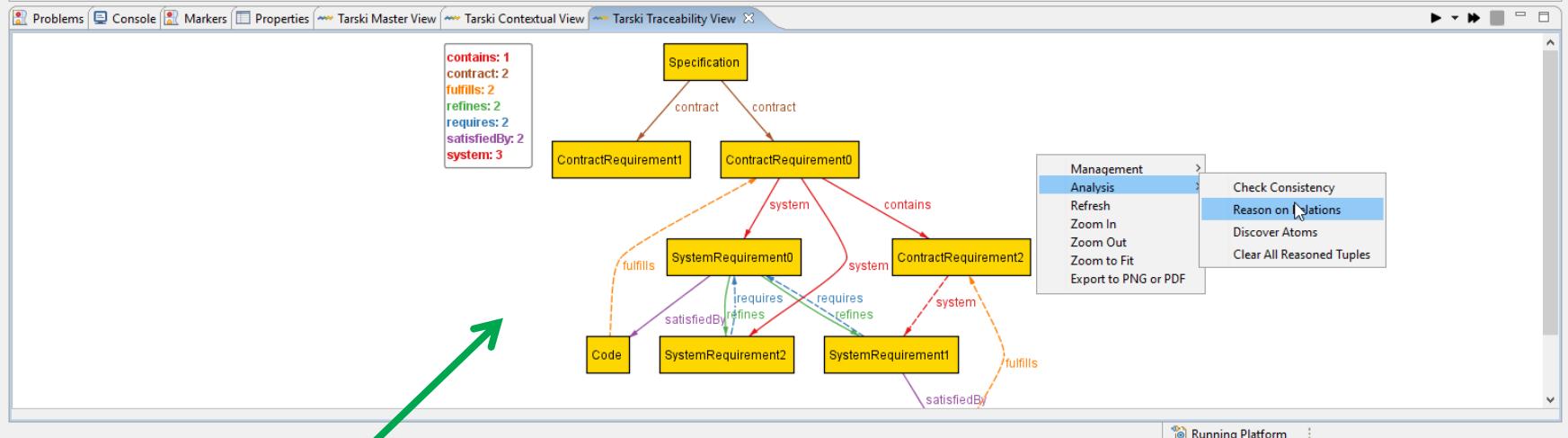
Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

Customer Requirements Specification.md Specification.java reqif10.ecore

1# Customer Requirements Specification
2
3## UC-1 Create a new SpecObject
4
5 Note that the Specification Editor is the main interface for users. Therefore,
6 creating SpecObjects in this editor is the main success scenario.
7
8### Precondition
9
10 ReqIF model exists and is open.
11
12### Main Success Scenario
13
14 1. We assume that a Specification exists and is open (not required for alternative
scenario)
15 2. Open a row's context menu (or in the empty editor space)
3. Select the Child or Sibling submenu.

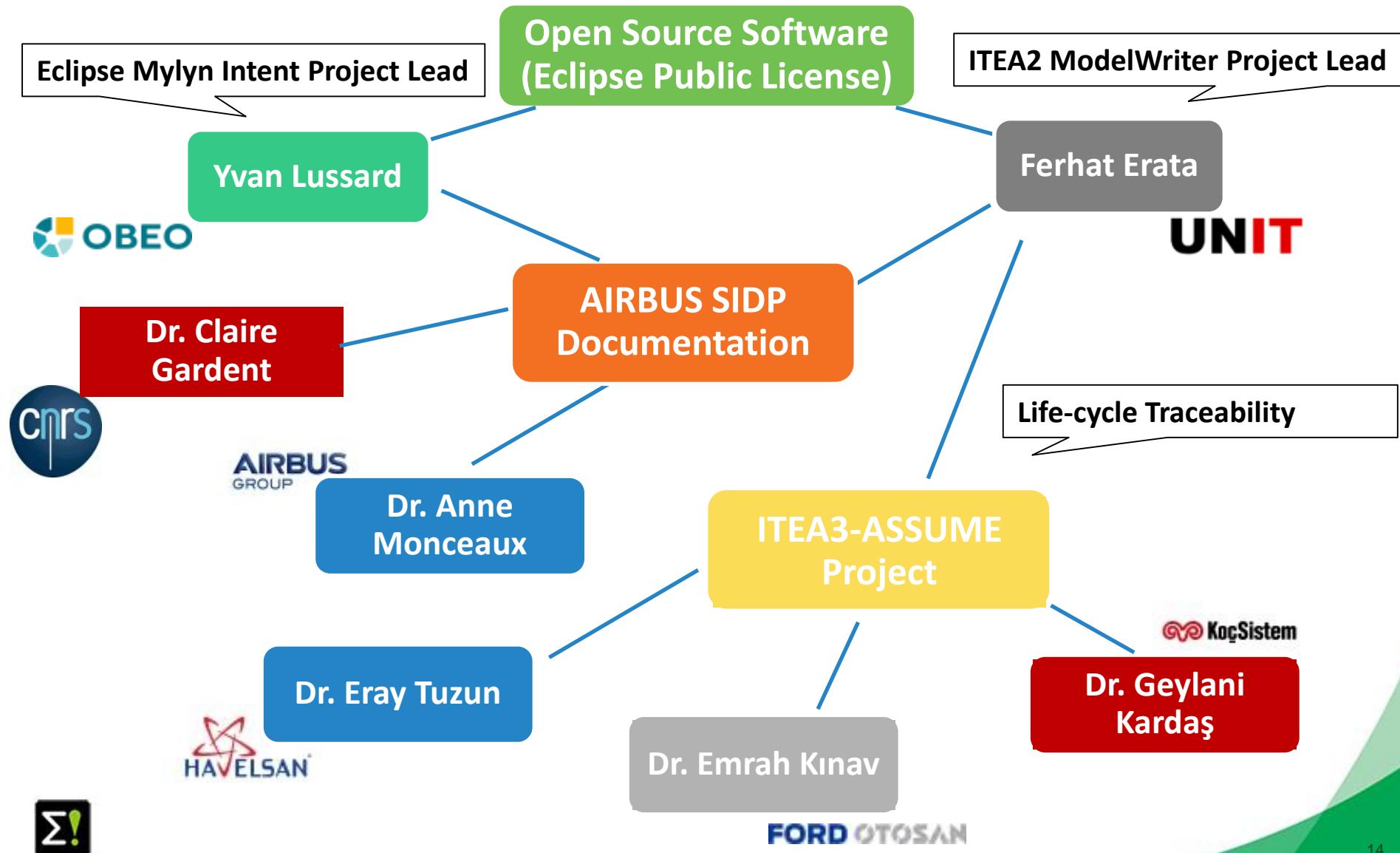
Markdown Source Preview



“Various Traceability Analysis might be performed”

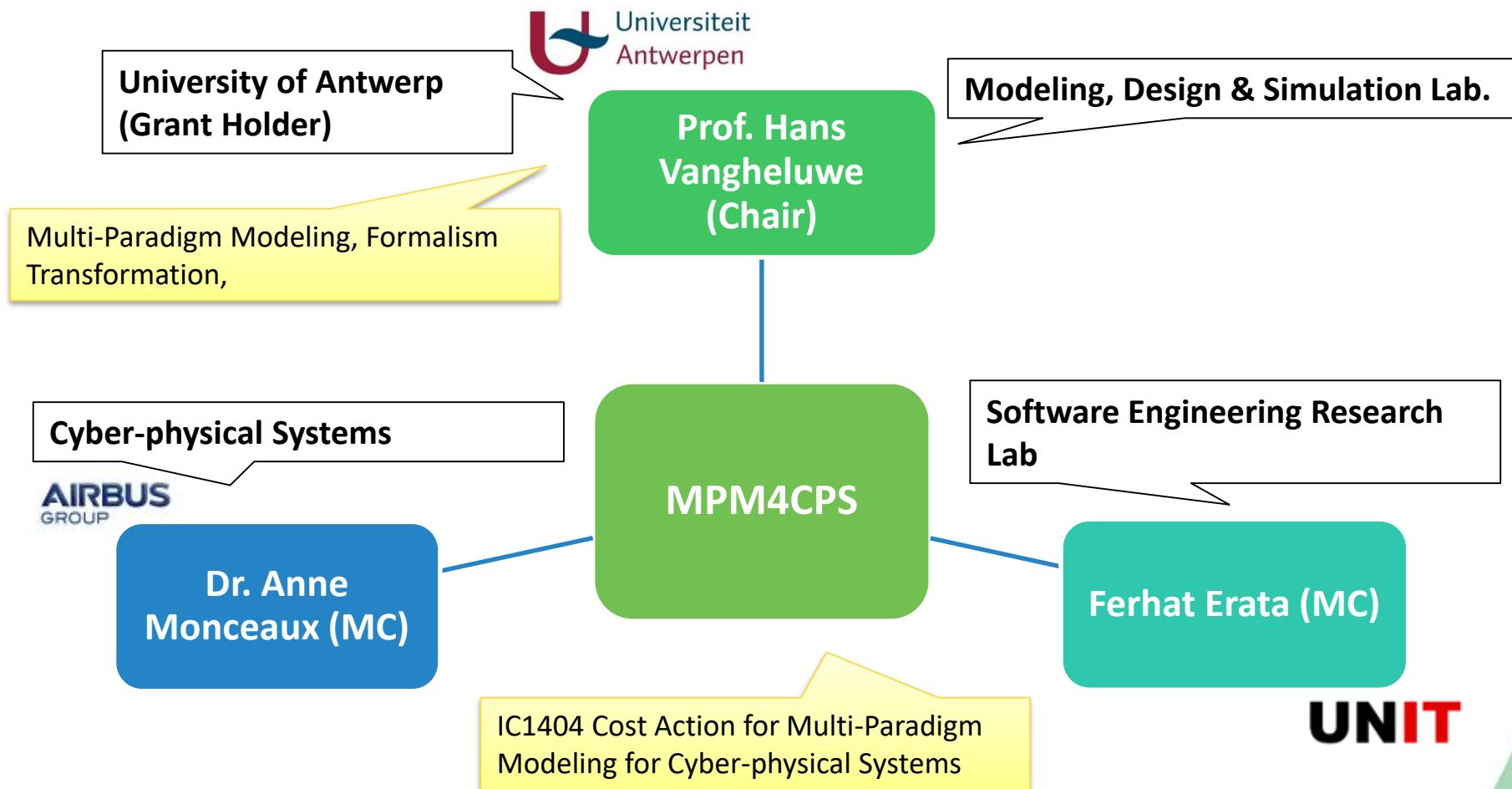
Level of Collaboration within ModelWriter

International Collaboration



Level of Collaboration within ModelWriter

International Collaboration



ModelWriter Activities in the First Year



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

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

Initial Architectural Design, Industrial Use Cases, Technical WP discussions

Collaboration Infrastructure

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

Exploitation: Havelsan's participation

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

Consolidated User Requirements & Review

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

Software Requirements Review & Architecture

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

Rehearsal & Review

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

Integration of software components

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

ModelWriter Activities in the Second Year



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

Product Owner Review Meeting

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

ModelWriter is positioned on the Working Groups of this Cost Action

ICT Cost Action - MPM4CPS WG meeting at Vienna, Austria, on the 15-16 April, 2016

The 7th International ModelWriter Technical Workshop in Toulouse, France (6 June 2016)

Future of ModelWriter is discussed

The 7th Int'l ModelWriter Brainstorming Session in Toulouse, France at Airbus (9 June 2016)

The 7th Int'l ModelWriter Coordination Meeting in Toulouse, France at Airbus (10 June 2016)

Poster Presentation

ModelWriter Poster Presentation SAT/SMT/AR Summer School 2016

Participation in International Joint Conference on Automated Reasoning (IJCAR) 2016

ModelWriter Activities in the Second Year



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

Verification Technology, Systems & Applications (VTSA) Summer School (Aug 29- Sept 02, 2016)

ICT Cost Action - MPM4CPS WG meeting at Gdansk, Poland (Sep 13-16, 2016)

A paper is submitted to ACM Applied Computing Symposium and under review.

Participation in Workshop on Software Correctness and Reliability (Oct 7-8 2016)

The 8th International ModelWriter Technical Workshop in Toulouse, France (13 June 2016)

ICT Cost Action - MPM4CPS WG meeting at Malaga, Spain (Nov 24-25, 2016)

WP1 Industrial Use Cases and Requirements (AIRBUS)

WP2 (LORIA)

- Semantic Parser
- Document Generation
- bi-directional transformation between text and formal knowledge representation

WP3 (UNIT)

- Bi-directional synchronization mechanism between texts and models
- Configuration & Traceability Components
- Consistency checker plug-in for consistency

WP4 (MANTIS)

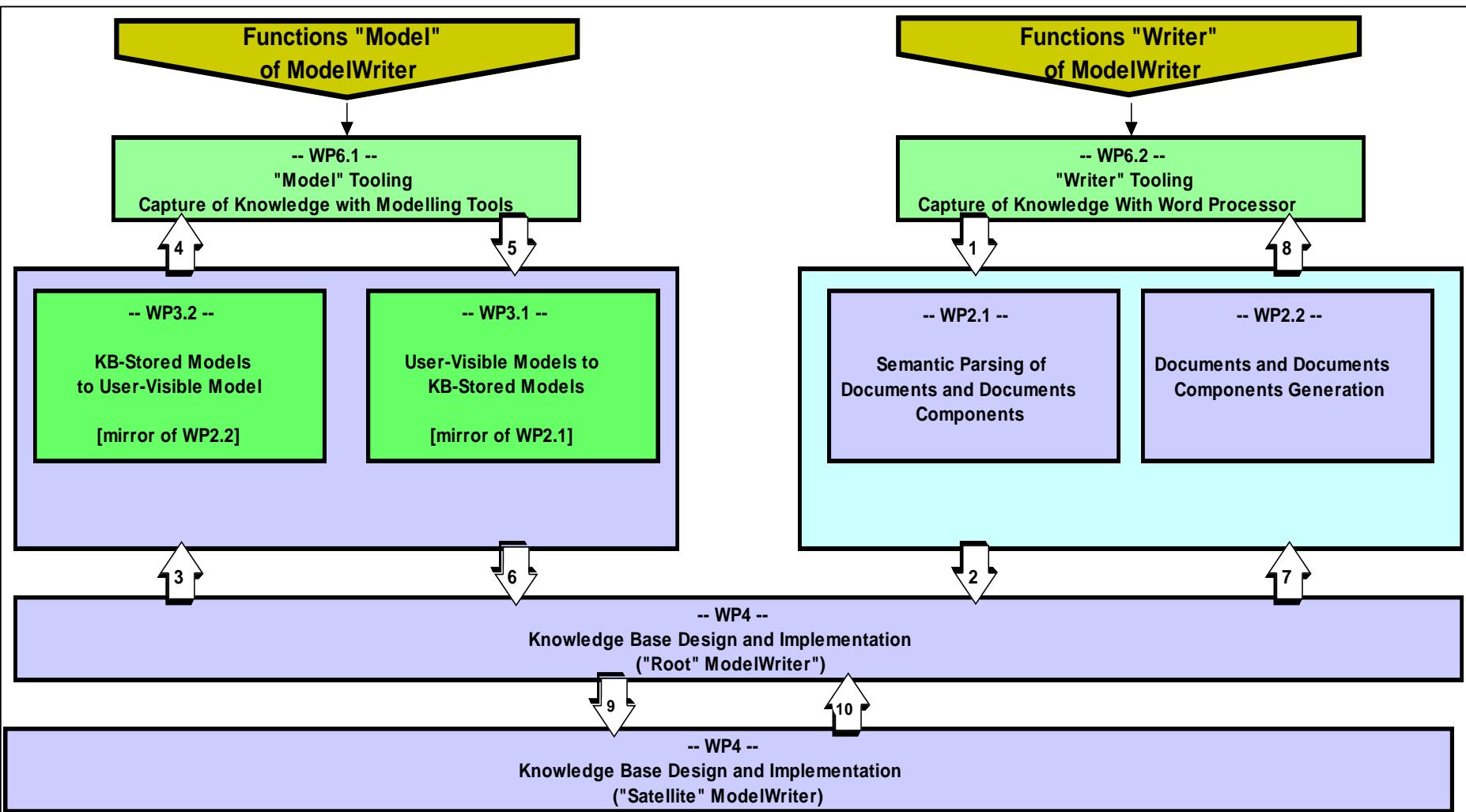
- A federated Knowledge Base and its API
- Synchronization mechanism between texts/models & knowledge base

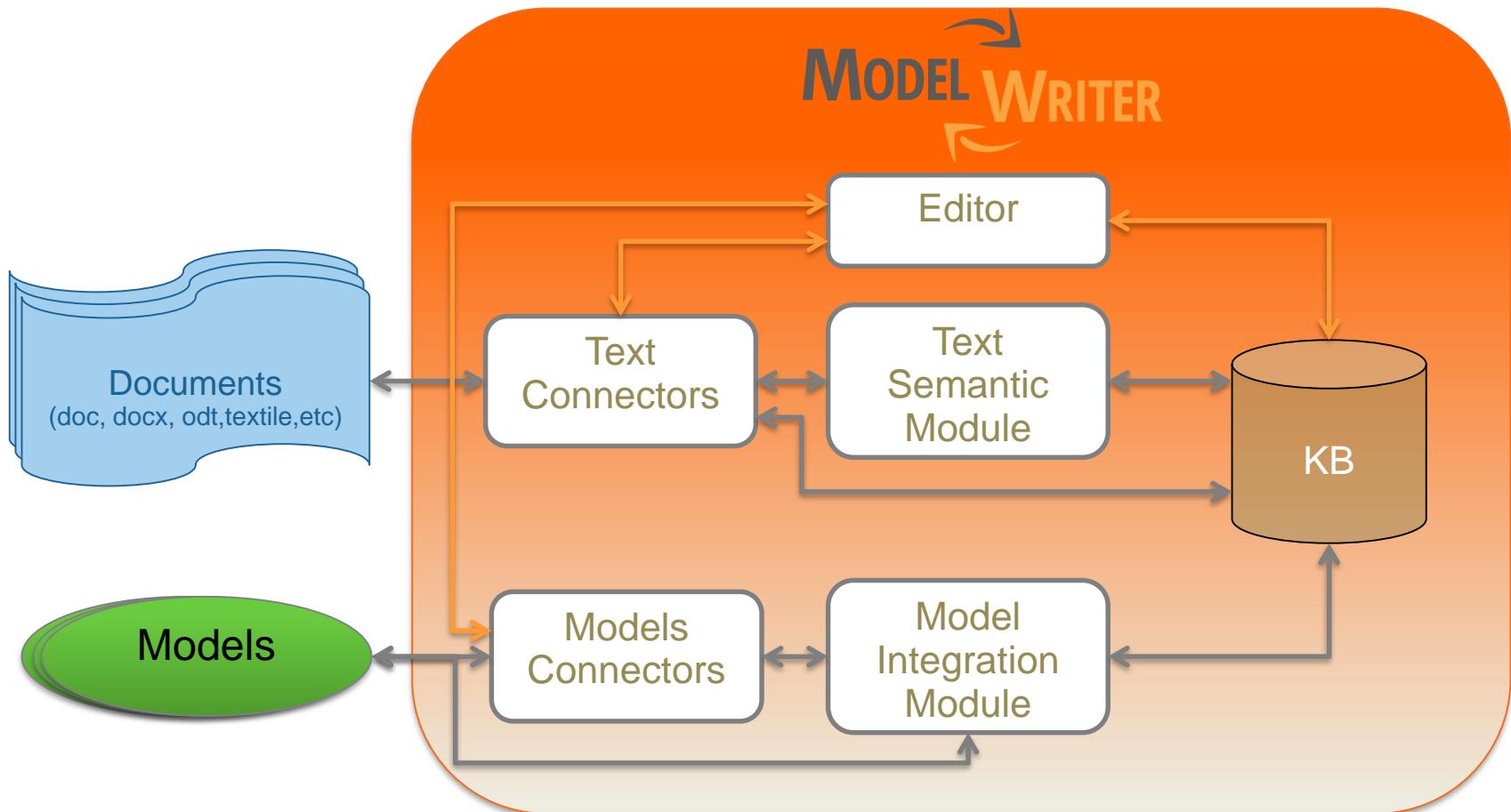
WP6 (OBEO)

- A complete “ModelWriter” tool integrating of all these in a consistent way
- User Interfaces

WP5 Project Management (UNIT)

WP7 Standardization, Dissemination and Exploitation (OBEO)





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

DRAFT

Model Writer

Progress on the Industrial Use Cases

DRAFT

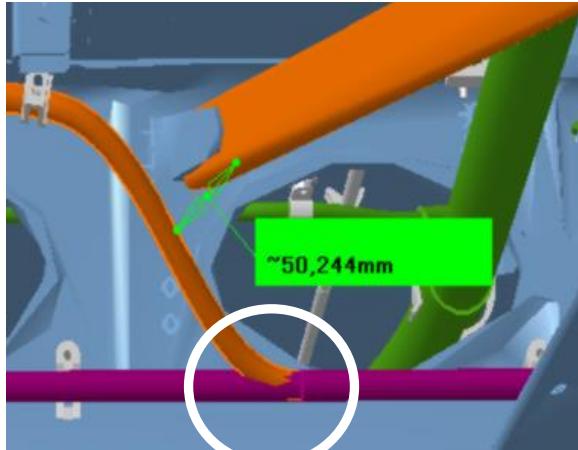
AIRBUS industrial use cases

Anne MONCEAUX

AIRBUS GROUP INNOVATIONS

Airbus use cases in ModelWriter

- We decided to concentrate on some problematics related to System Installation
- We primarily focus on Electrical, Water Waste, and Fuel Systems
- System installation:
 - obeys to some Regulations 
 - is specified by some rules/requirements/principles 
 - the design and the manufactured product are checked against these rules 



Certification Specification – CS 25 (EASA)

DRAFT 3

The screenshot shows the software interface for the EASA CS 25 Certification Specification. On the left, there is a tree-view table of contents. The following sections are listed under 'CS-25 BOOK 1 - Certification Specifications for Large Aeroplanes':

- TITLE PAGE - CS-25 Amdt 5
- CONTENTS
- PREAMBLE
- CS-25 BOOK 1 - Certification Specifications for Large Aeroplanes
 - SUBPART A – GENERAL
 - SUBPART B – FLIGHT
 - SUBPART C – STRUCTURE
 - SUBPART D – DESIGN AND CONSTRUCTION
 - SUBPART E – POWERPLANT
 - SUBPART F – EQUIPMENT
 - SUBPART G – OPERATING LIMITATIONS AND INFORMATION
 - SUBPART H – ELECTRICAL WIRING INTERCONNECTION SYSTEM** (highlighted with a red dashed border)
 - SUBPART J – AUXILIARY POWER UNIT INSTALLATIONS
- CS-25 BOOK 1 - APPENDICES
- CS-25 BOOK 2 - Acceptable Means of Compliance
- AMC - INTRODUCTION
- AMC – SUBPART B
- AMC – SUBPART C
- AMC – SUBPART D
- AMC – SUBPART E
- AMC – SUBPART F
- APPENDIX 1. ASSESSMENT METHODS.

CS-25 BOOK 1

SUBPART H – ELECTRICAL WIRING INTERCONNECTION SYSTEM

CS 25.1701 **Definition**
(See AMC 25.1701)

(a) Electrical wiring interconnection system (EWIS) means any wire, wiring device, or combination of these, including termination devices, installed in any area of the aeroplane for the purpose of transmitting electrical energy, including data and signals between two or more intended termination points. Except as provided for in subparagraph (c) of this paragraph, this includes:

- (1) Wires and cables.
- (2) Bus bars.
- (3) The termination point on electrical devices, including those on relays, interrupters, switches, contactors, terminal blocks, and circuit breakers and other circuit protection devices.
- (4) Connectors, including feed-through connectors.
- (5) Connector accessories.
- (6) Electrical grounding and bonding.

(i) Appropriate for the intended function and operating environment, and

(ii) Acceptable to the Agency.

(2) Portable electrical devices that are not part of the type design of the aeroplane. This includes personal entertainment devices and laptop computers.

(3) Fibre optics.

[Amdt. No.:25/5]

CS 25.1703 **Function and Installation; EWIS**
(See AMC 25.1703)

(a) Each EWIS component installed in any area of the aeroplane must:

- (1) Be of a kind and design appropriate to its intended function.
- (2) Be installed according to instructions specified for the EWIS

CS 25.1701 Definition

(a) Electrical wiring interconnection system (EWIS) means any wire, wiring device, or combination of these, including termination devices, installed in any area of the aeroplane for the purpose of transmitting electrical energy, including data and signals between two or more intended termination points (...)

Subparagraph (b) of this paragraph, EWIS components inside the following equipment, and the external connectors that are part of that equipment, are excluded from the definition in subparagraph (a) of this paragraph:

(1) Electrical equipment or avionics that is qualified to environmental conditions and testing procedures when those conditions and procedures are -

[Amdt. No.:25/5]

CS 25.1705 **Systems and Functions; EWIS**

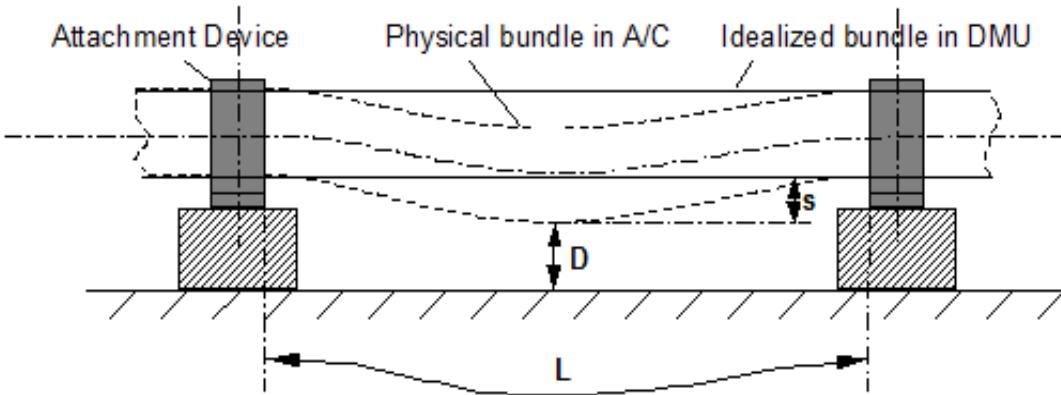
(a) EWIS associated with systems required for type certification or by operating rules must be



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.)

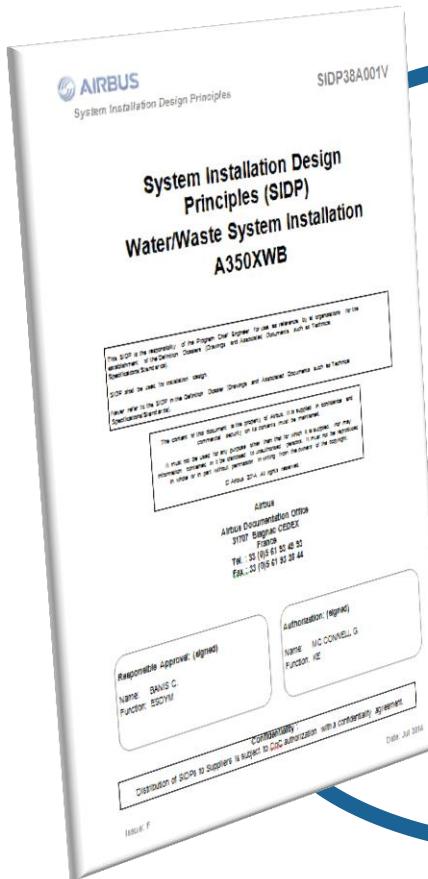
D ... Required Distance

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

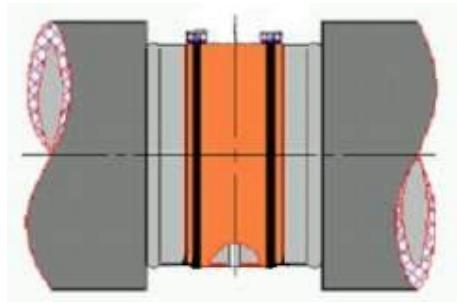
Figure 6: Sagging of bundles between attachment points

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

Rule traceability & design check



Traceability



Check

Problem statement & objectives

1. Model and manage the design rules: since SIDP (design principles) are todays kinds of natural language requirements and explanations:
 - not all the rules can be directly formalized in a way to be used to verify the design
 - rules are spread over various products/programs
 - rules evolve
 - rules are complex artefacts made of text and picture and tables....

⇒ This problem is the one we have been considering primarily in the scope of ModelWriter, trying to define a rule model and knowledge base
2. Enable mapping between rule model and design model, in order to automate identification of conflicts between rules and design, and then automate analysis of the impact of changes in rules or in design.

⇒ This 2d objective we think cannot be reached in the frame of Modelwriter because of the challenge of mapping 3D design models to the BOM ... and because lack of skills and data in the context of the project

BUT to be consider to make the system amenable for formal analysis

Problem statement & objectives

(1) Manage rules/design principles and improve traceability

Easier search/retrieval of rules	<ul style="list-style-type: none">- save costs/time by supporting DP consultation / retrieval in accordance to the needs
Easier change impact analysis at rules level	<ul style="list-style-type: none">- save DP updating costs/time- avoid non-compliance of design caused by DP updating lead time
Easier traceability between artefacts (BOM DB, design models, CAD models...)	<ul style="list-style-type: none">- keep traceability from upstream regulations to requirements and to downstream design models

(2) Automate identification of design conflicts against rules

Easier consistency checking of the design (CAD) data	<ul style="list-style-type: none">- Save time to retrieve applicable rules- avoid non-compliance of design wrt rules
--	---

Solutions explored in MW

		1	2	3	4
Use a Component taxonomy (BOM) as a reference to annotate rules (rules as Text part / Taxonomy as Model part)	MW synchronization mechanism to help annotation management	+	+	-	-
Extend first approach to other artefacts/models	MW synch between several artefacts	-	-	+	+
Add semantics to BOM to enable reasoning	MW synch	+	+	+	
Model rules written in natural language by formalizing the rules in description logic / ability to dynamically update the BOM/component ontology and apply some checks on design rules	MW writer part features that LORIA developed so far in the formalization process + MW synch			+	+
Model rules written in natural language by formalizing the rules in relational logic	MW writer part + Tarsky part + Synch mech			+	+

1=Rule retrieval – 2=Rule change – 3= traceability – 4=design check

DRAFT

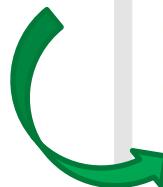
UC-FR-03 - Synchronization of Regulation Documentation with a Design Rule Repository

*Anne MONCEAUX
Airbus Group Innovation*

Ex. BOM, Component taxonomy, Component ontology

DRAFT 3

Item description	
33	jigou: Implicit physical Item or group of Items for consistency checks , when a function is the the Item with 0 in weight and next column (e.g. FCS PRIM : on IM
1082	tank drain valves
1083	Vent system
1084	Pressure relief
1085	Carbon OPP disc
1086	flame arrestor
1087	float valve
	NACA duct



```
<http://airbus-group.installsys/component>
  rdf:type owl:Ontology ;
  rdfs:comment "Component definitions are given and validated by Airbus ESIR dpt"^^xsd:string ;
  owl:imports <http://qudt.org/schema/qudt> ;
  owl:imports <http://qudt.org/vocab/unit> ;
  owl:versionInfo "Ontology for System Installation Components"^^xsd:string ;

  comp:ABS1759
    rdf:type owl:Class ;
    rdfs:label "ABS 1759 cable tie mount"^^xsd:string ;
    rdfs:label "ABS1759"^^xsd:string ;
    rdfs:label "cable tie mount ABS1759"^^xsd:string ;
    rdfs:subClassOf comp:CableTieMount ;
    skos:prefLabel "ABS 1759 cable tie mount"@en ;
  .

  comp:AFDXcable
    rdf:type owl:Class ;
    rdfs:label "AFDXcable"^^xsd:string ;
    rdfs:subClassOf comp:BusCable ;
    skos:prefLabel "AFDXcable"^^xsd:string ;
  .

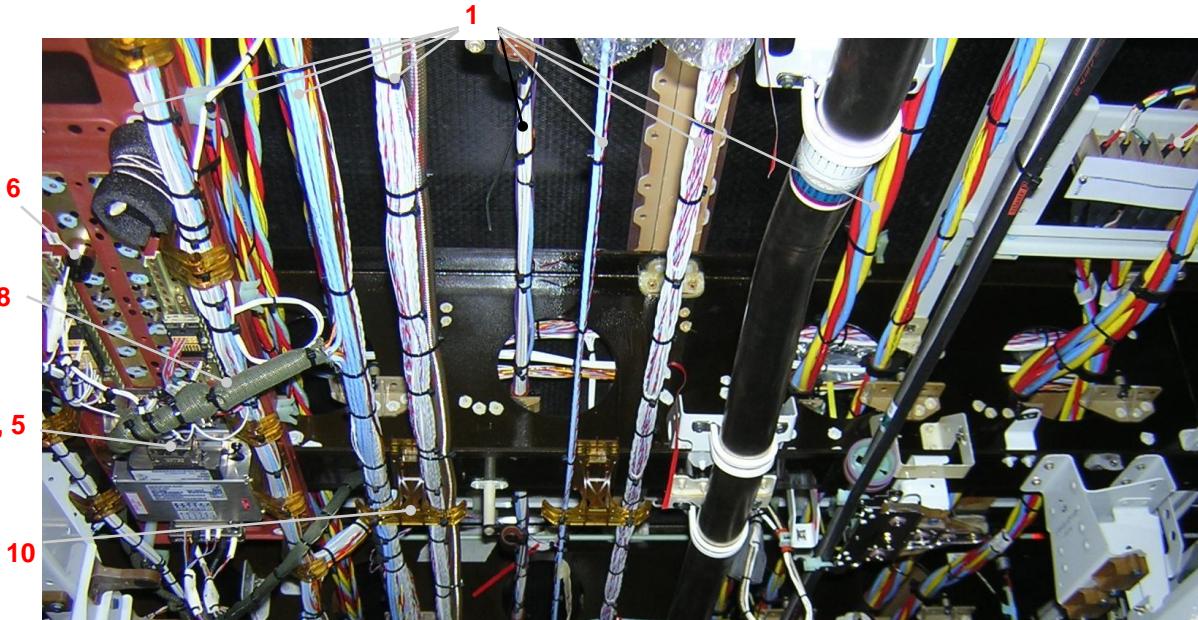
  comp:Active-Fastener
    rdf:type owl:Class ;
    rdfs:label "active fastener"^^xsd:string ;
    rdfs:subClassOf comp:Fastener ;
    rdfs:subClassOf [
      .
      rdf:type owl:Restriction ;
      owl:hasValue "true"^^xsd:boolean ;
      owl:onProperty comp:isActive ;
    ] ;
    skos:prefLabel "active fastener"@en ;
  .

  comp:Active_component
    rdf:type owl:Class ;
    rdfs:label "active component"@en ;
    rdfs:label "active element"@en ;
    rdfs:label "active item"@en ;
```

Electrical wiring interconnection system (EWIS)

DRAFT

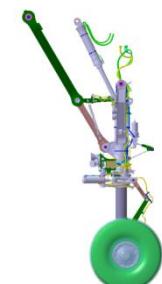
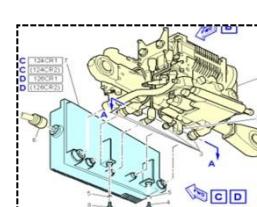
⇒ Mainly: Aircraft electrical common installation (ATA92)



⇒ But also: part of Systems equipment wiring:

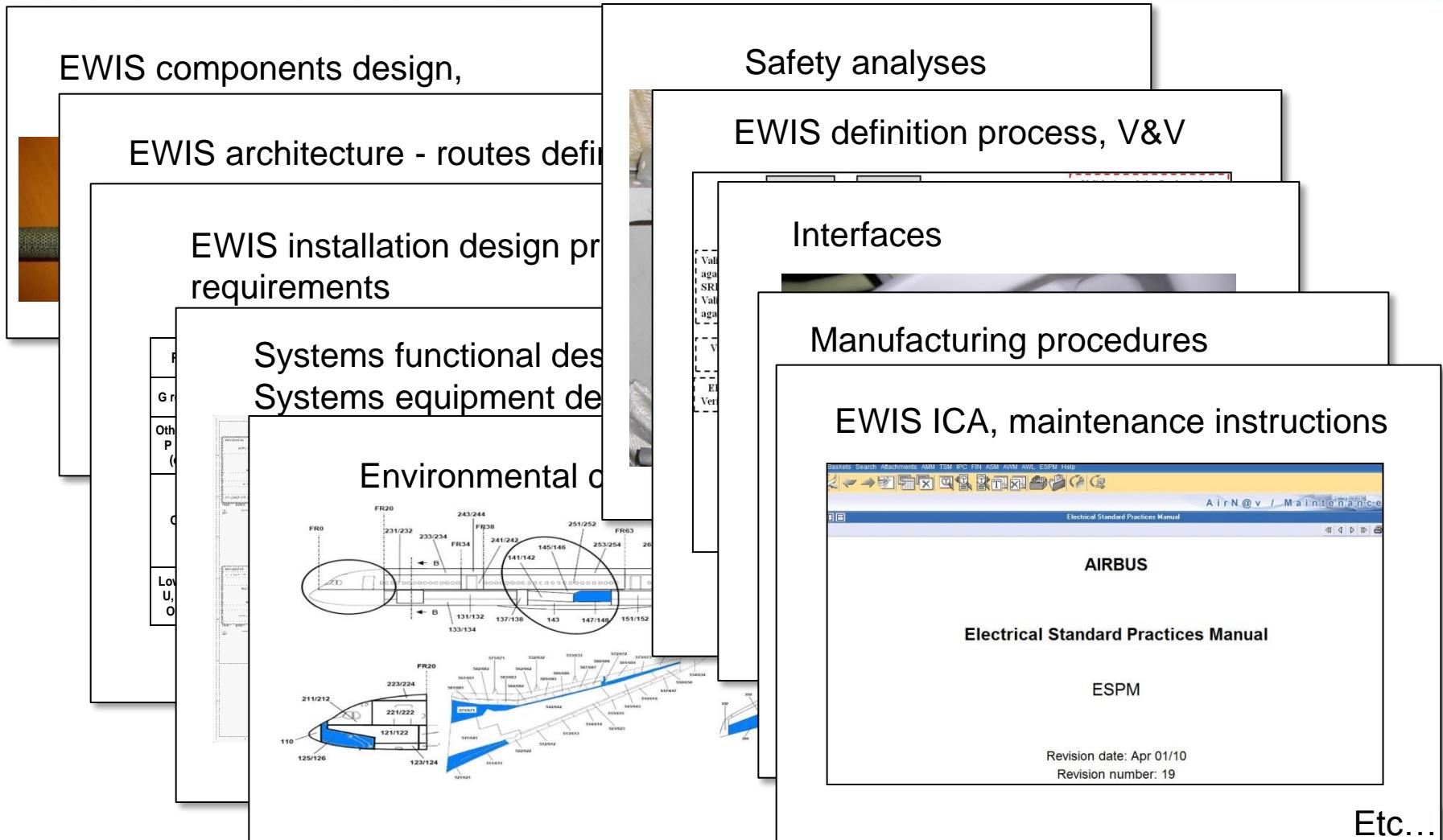
- Power distribution centers wiring
- External wiring of equipment
- Wiring of equipment not qualified to appropriate standards e.g. EUROCAE ED-14 / RTCA DO-160

cf CS 25.1701(b)&(c)



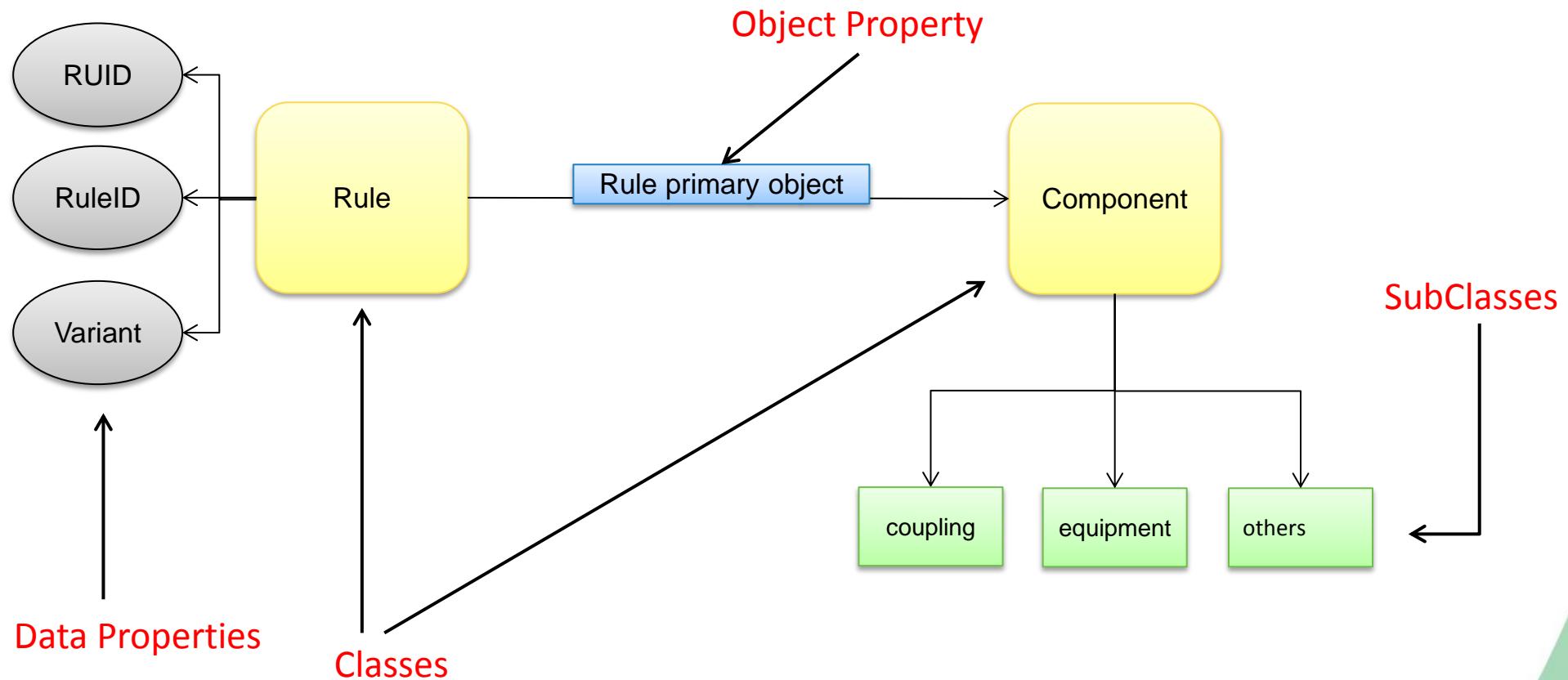
Illustrate traceability between all aspects of A/C wiring

DRAFT 3



Back-up slides

Représentation des données



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

DRAFT 3

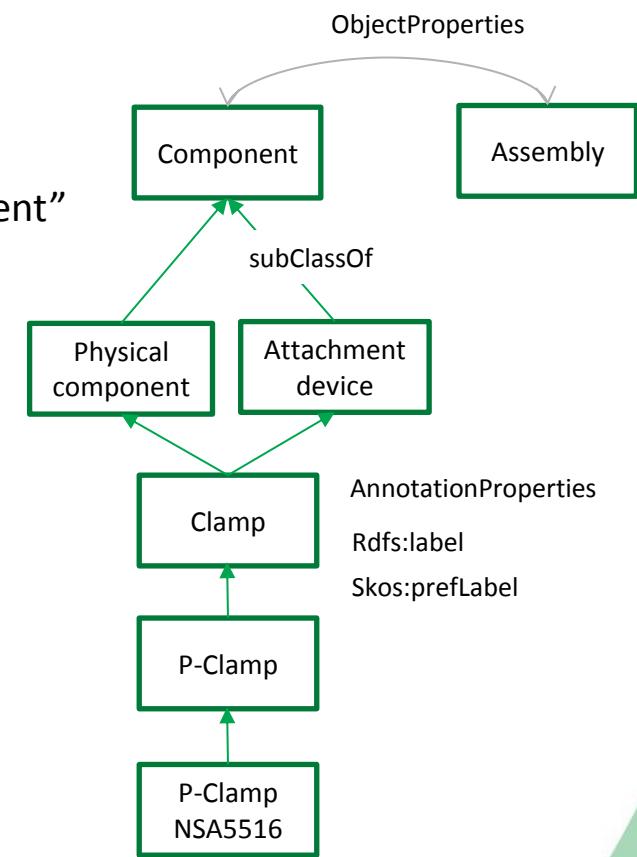
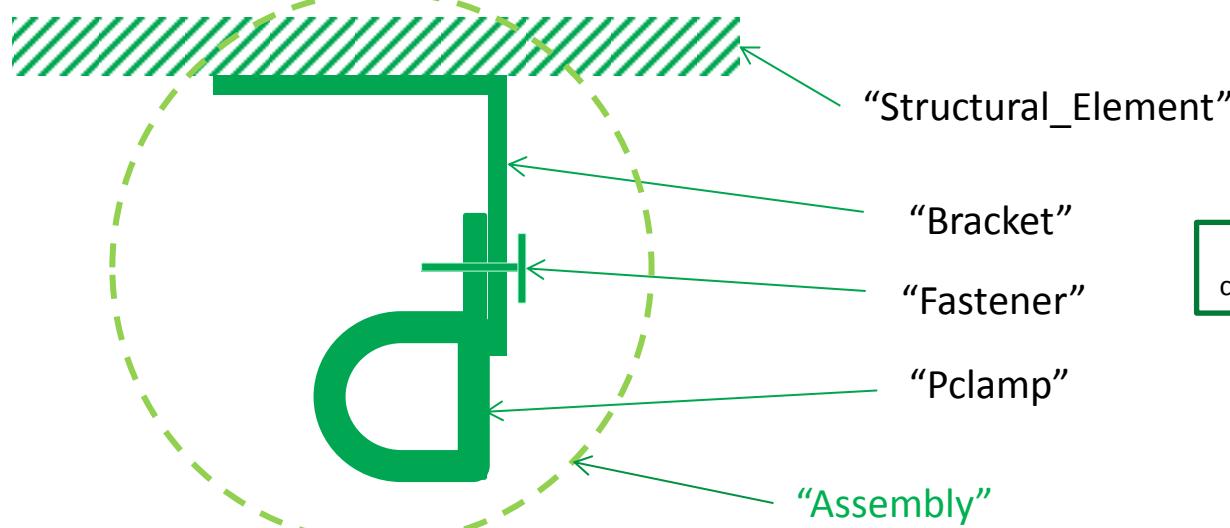
Status 1st iteration: synchronizing the SIDP KB content and text part

Model part = rule KB

Textual part = rule sentences

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

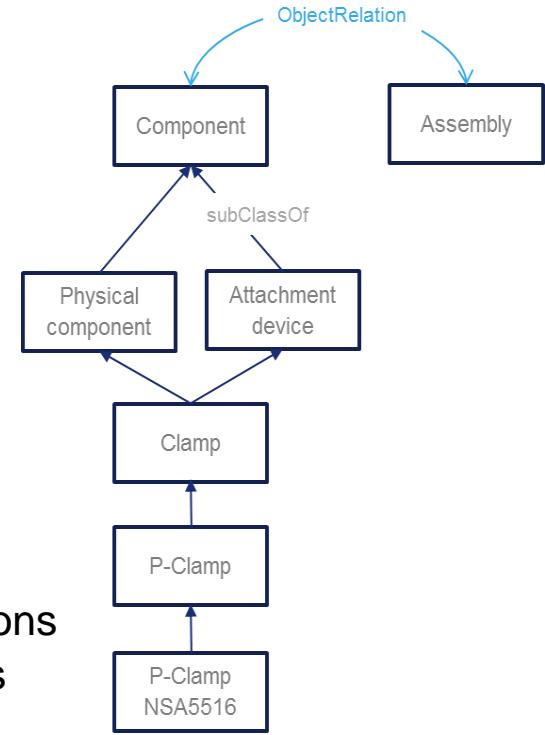
Component classes taxonomy



Component classes taxonomy

“P-clamp” NSA5516 can be fixed on X with Y

 “Physical component” “Standard reference”



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

SPARQL queries

Status : model based queries specification

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

The screenshot shows the RDFizer application interface. On the left, there's an ontology browser with tabs for 'Ontology' (selected), 'Datas', and 'Queries'. Under 'Ontology', it shows 'Uploaded' and 'Classes Tree View'. The 'Classes' tree view pane lists various classes like ActionVerb, ATA, Auxiliary, Collection, Concept, ConceptScheme, ConditionalSolution, DesignMaturity, Location, and Object (which is selected). Below these are Prep1, Prep2, Prep3, Program, Rule, Zone, and ZoneClassification. On the right, there's a 'Query' builder. It has a 'Query' section with a table where the first row contains 'RulePerComponent' and 'comp:Component' with a dropdown menu 'Select an element'. Below this is a detailed view of a query step: '2.1 Find rules applying to a "Component" type'. It includes a list of sub-points (a, b, c) and their descriptions. To the right of this is the SPARQL code for the query:

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

At the bottom of the query builder, there are buttons for 'Add', 'Delete', 'Modify', 'Save', 'Execute', and tabs for 'Query Result' and 'Ontology Details'.

DRAFT

**Thank you for your attention!
Any question ?**





Model Writer

**FEAD (Front End Accessory and Drive) &
EGR (Exhaust Gas Recycling)**

06.11.2016

SYNCHRONIZATION OF DESIGN SPECIFICATIONS WITH CAD DATA

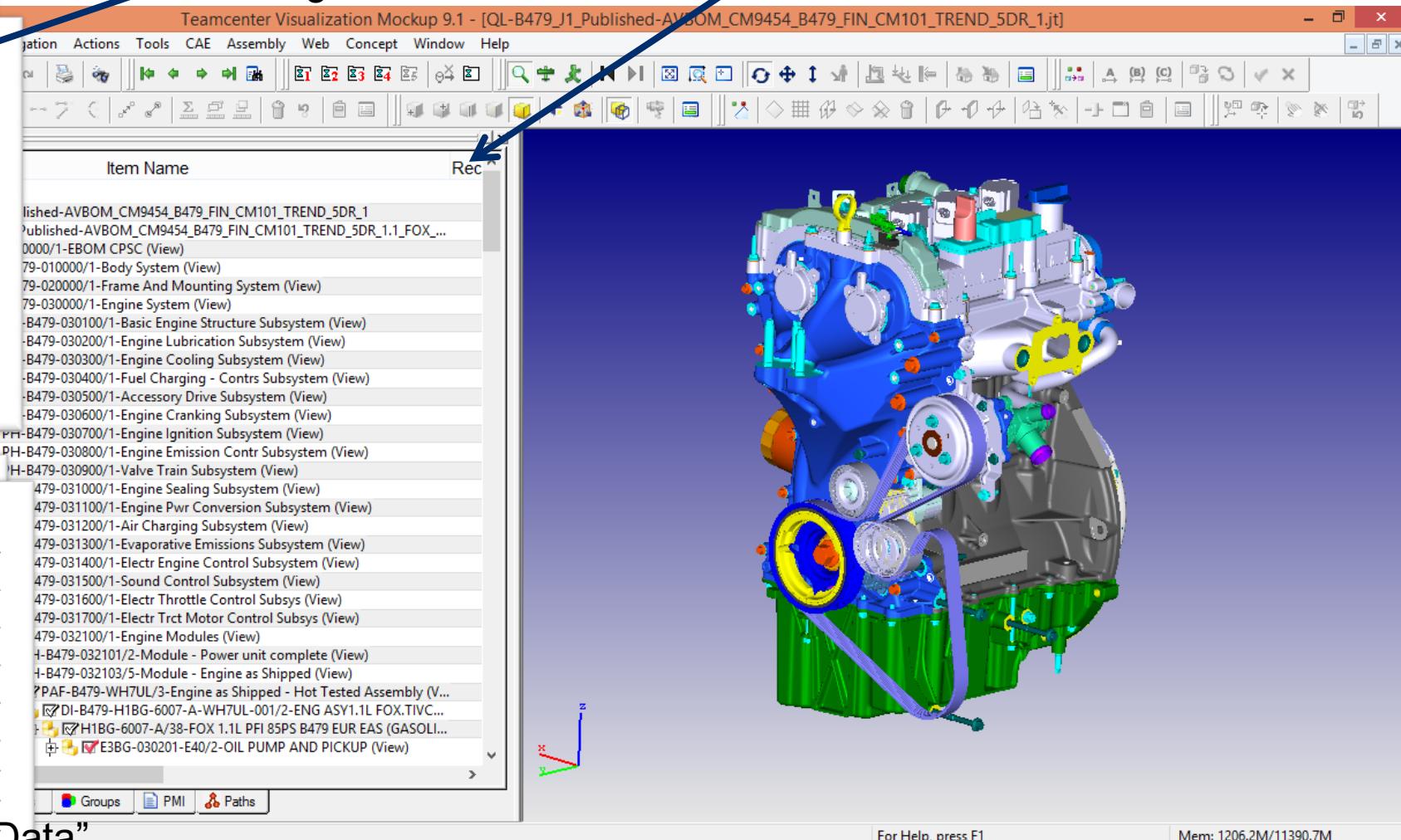
Design Rules, BoM and CAD Data

KPAC Set ID	Title	Author	Type
PT_FAST-S13	Cup Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Dryseal Pipe Thread Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Dowel and Bushing Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Fastener and Joint Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Threaded Fastener Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Wrench and Socket Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Ball Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Port Plug Design Rules Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S11	Thread Forming Screws in Plastic Saved Set	Murphy, Mark (mmurphy9)	Check List

'refix*	Base*	Suffix*	Name*	CPSC*	Engineering Commodity	Remarks (Fastener Attaching Statement)
W50025	S437	BOLT M8X30 HF 8	030501	FASTENERS (PMT400)		FROM: 6A228, TO: 6059;
W500214	S437	BOLT M6X20 HF PIL 8	030501	FASTENERS (PMT400)		FROM: 19A216, TO: 6059;
W704693	S442	BOLT M8X95 HF PIL 8	030501	FASTENERS (PMT400)		FROM: 10300; TO: 6059 & 6015;
W704693	S442	BOLT M8X95 HF PIL 8	030501	FASTENERS (PMT400)		FROM: 19D259; TO: 6059 & 6675;
W714914	S303	BOLT M8X13 HF SERR STL	030501	FASTENERS (PMT400)		FROM: 8509, TO: 6059;
C1BQ	8509	AC	CVR-ACC DRV	030501	FEAD	
CM5Q	19A216	AB	IDL ASY ACC DRV BEL	030501	FEAD	
CM5Q	6A228	BA	TENS ASY ACC DRV	030501	FEAD	
CM5Q	8C301	DA	BCI ACC DRV	030501	FEAD	

“Design Rules”

“Computer-aided Design Data”



SIEMENS COLLABORATION WITH FORD

Siemens – TeamCenter

Ford Otosan is using Siemens Team Center for sharing and uploading the internal documents. Siemens has 6 server for Ford. Ford Otosan is currently using Ford of Europe (FOE) servers.

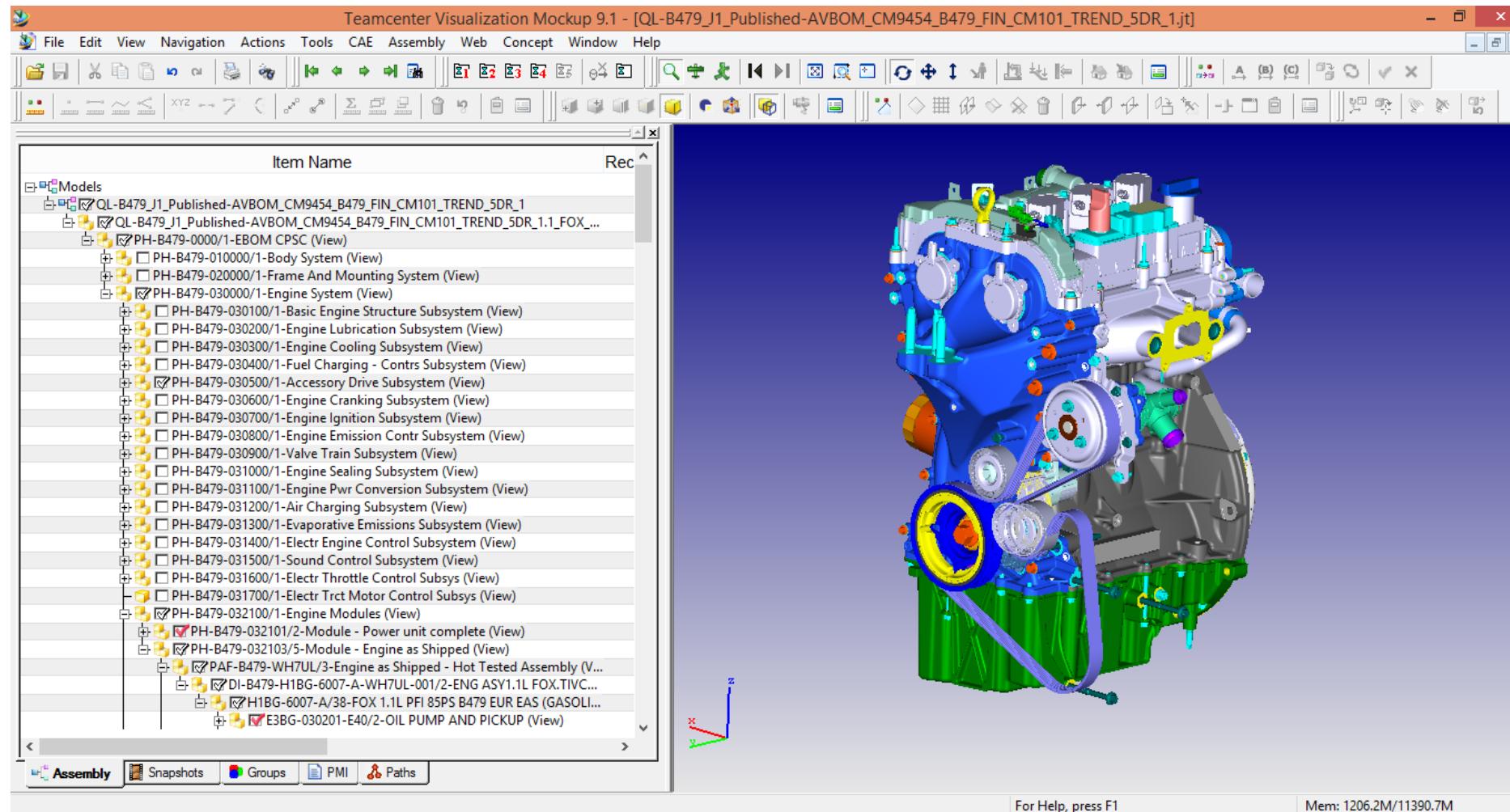
<https://www.teamcenterfoe1pe1.ford.com/tc/webclient>

The screenshot shows the Siemens TeamCenter web client interface. The top navigation bar includes links for New, Edit, View, Tools, Actions, Help, Logout, and the Siemens logo. The main content area displays a document titled "PRG-V408MCA-V408MCA PROGRAM NODE". The document details pane shows the owner as "Collin.Candi-CCOLLI34 (ccoll34)" and the last modified date as "05-Nov-2016 09:46". Below this, there are tabs for Overview, Related Datasets, Impact Analysis, Viewer, and Details. The Overview tab is selected, showing a list of available revisions: PRG-V408MCA/1-V408MCA PROGRAM NODE, PRG-V408MCA/2-V408MCA PROGRAM NODE, PRG-V408MCA/3-V408MCA PROGRAM NODE, PRG-V408MCA/4-V408MCA PROGRAM NODE, PRG-V408MCA/5-V408MCA PROGRAM NODE, PRG-V408MCA/6-V408MCA PROGRAM NODE, PRG-V408MCA/7-V408MCA PROGRAM NODE, and PRG-V408MCA/8-V408MCA PROGRAM NODE. The "Actions" section contains a "Copy" button. The "Item Properties" section at the bottom lists the owner, group ID, and last modifying user.

USER INTERFACE FOR 3D MODELS

Siemens – TeamCenter Visualization Mockup 9.1

Ford Otosan is using Siemens TeamCenter Visualization Mockup 9.1 for displaying the documents.



SEVERAL FACTS ABOUT THE USE CASE

Facts, Impact, and KPIs

- Total number of rules in the whole system? About 2000 rules
- Expected Rule Coverage in the use cases? At least 50%
- Impact in FO: Total number of systems in which the tool can be applied? 70/80 teams (disciplines)
- Principles for each disciplines (
 - EGR: Thermodynamics, Fluids Dynamic
 - FEAD: Mechanical Construction
 - Suspension: Mechanical Construction
 - Electronic: Electrical Engineering and Embedded Systems)
- Total number of items in a vehicle with respect to CPSC system? About 10000 items per vehicle
- Performance Metrics
 - Expected Response time in seconds per each validation operation? At most 2 seconds
 - Server-side High Performance Computing is need for checking consistency and reasoning processes.
- Ergonomics
 - Siemens PLM Visualization Integration



Model Writer

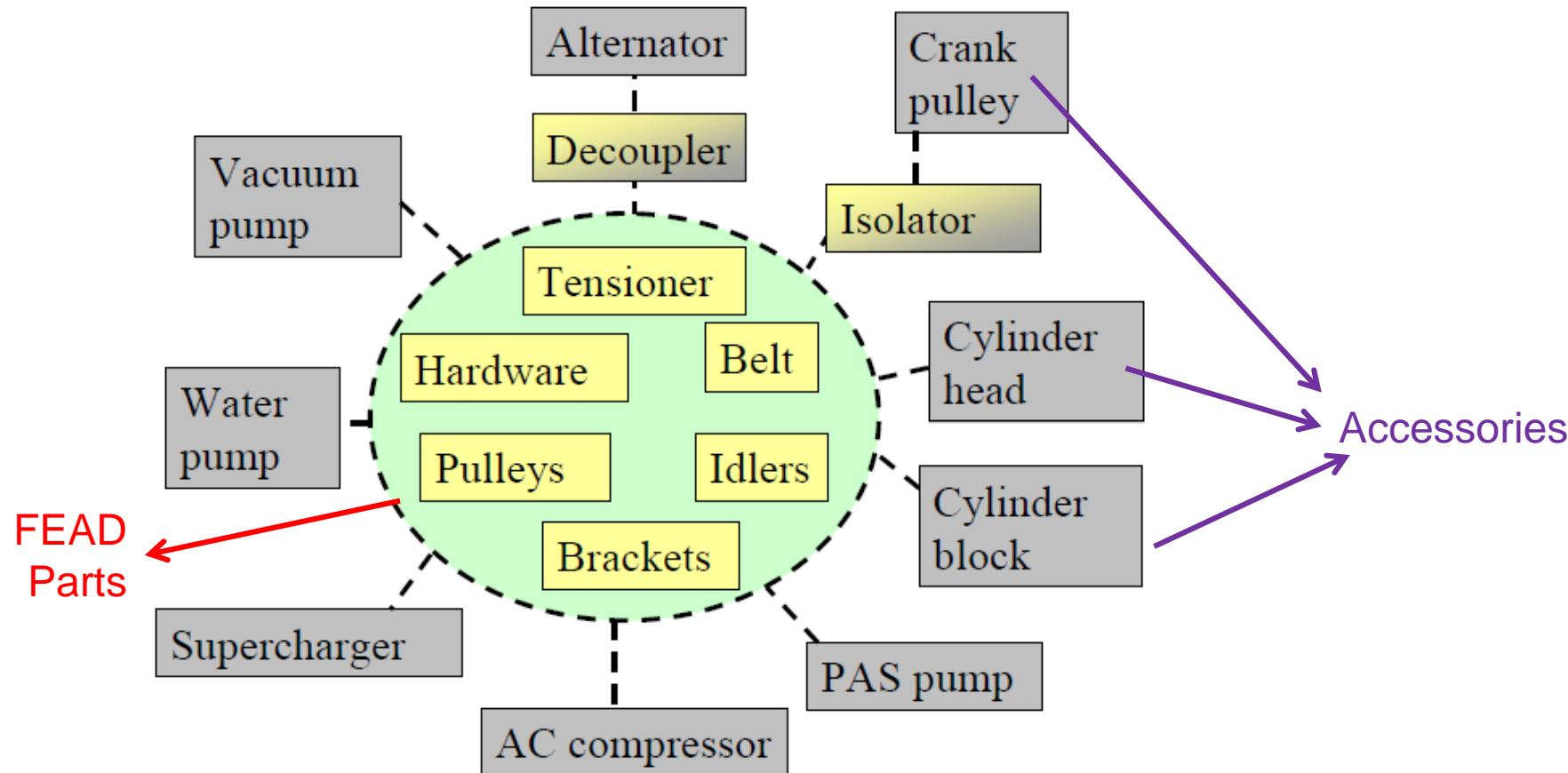
FEAD (Front End Accessory and Drive)

06.11.2016

WHAT'S FEAD?

Front End Accessory Drive Design

- The objective of good accessory drive design is to transmit power to turn the various accessories with maximum efficiency under all driving conditions, without the customer knowing the job is being done.



FEAD PARTS

Tensioner

- The function of a tensioner is to control system tension at a sufficient level to prevent belt slip under any driving condition.



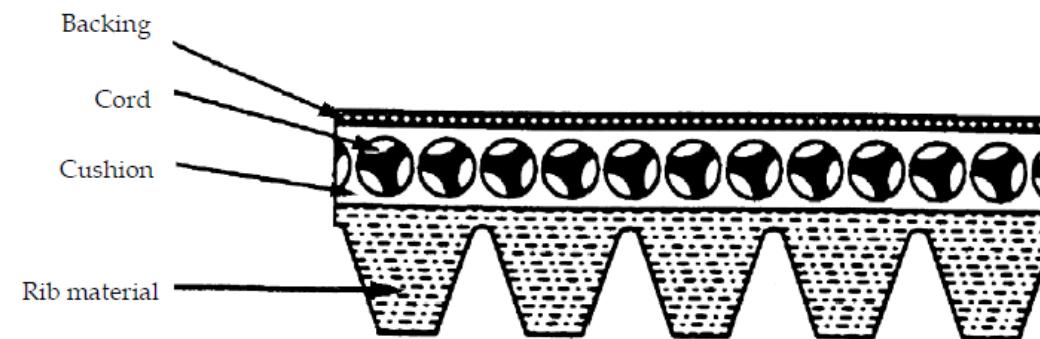
Manual

Rotary - automatic

Strut - automatic

Belt

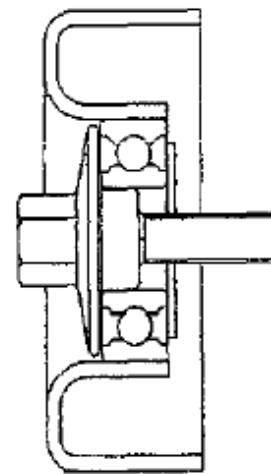
- The belt transmits the driving power from the crankshaft to the accessories.



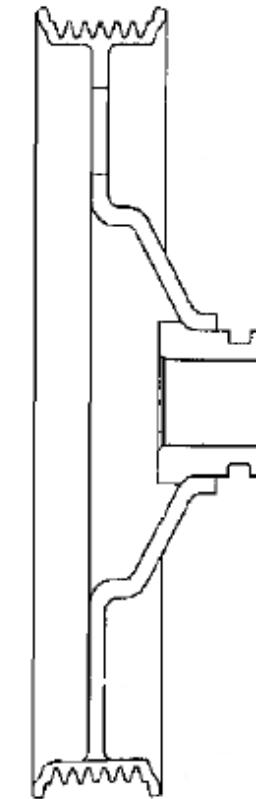
FEAD PARTS

Idler Pulleys

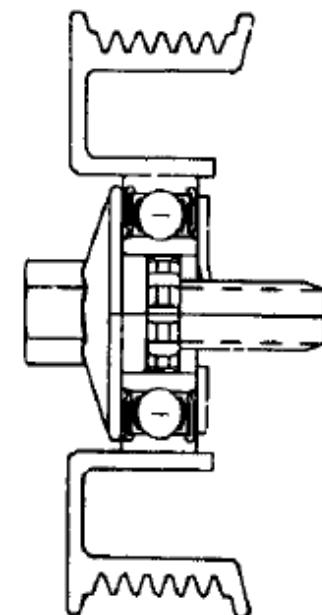
- An idler pulley is introduced into the system to guide the belt to achieve the following requirements:
 - Reposition the belt path to avoid obstacles,
 - Create more belt wrap on adjacent pulleys,
 - Control vibration on long spans.



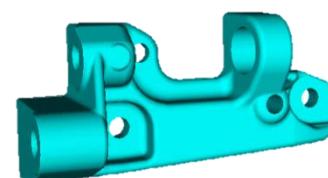
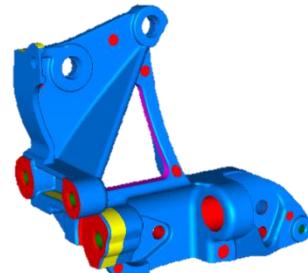
Flat idler



Grooved pulley

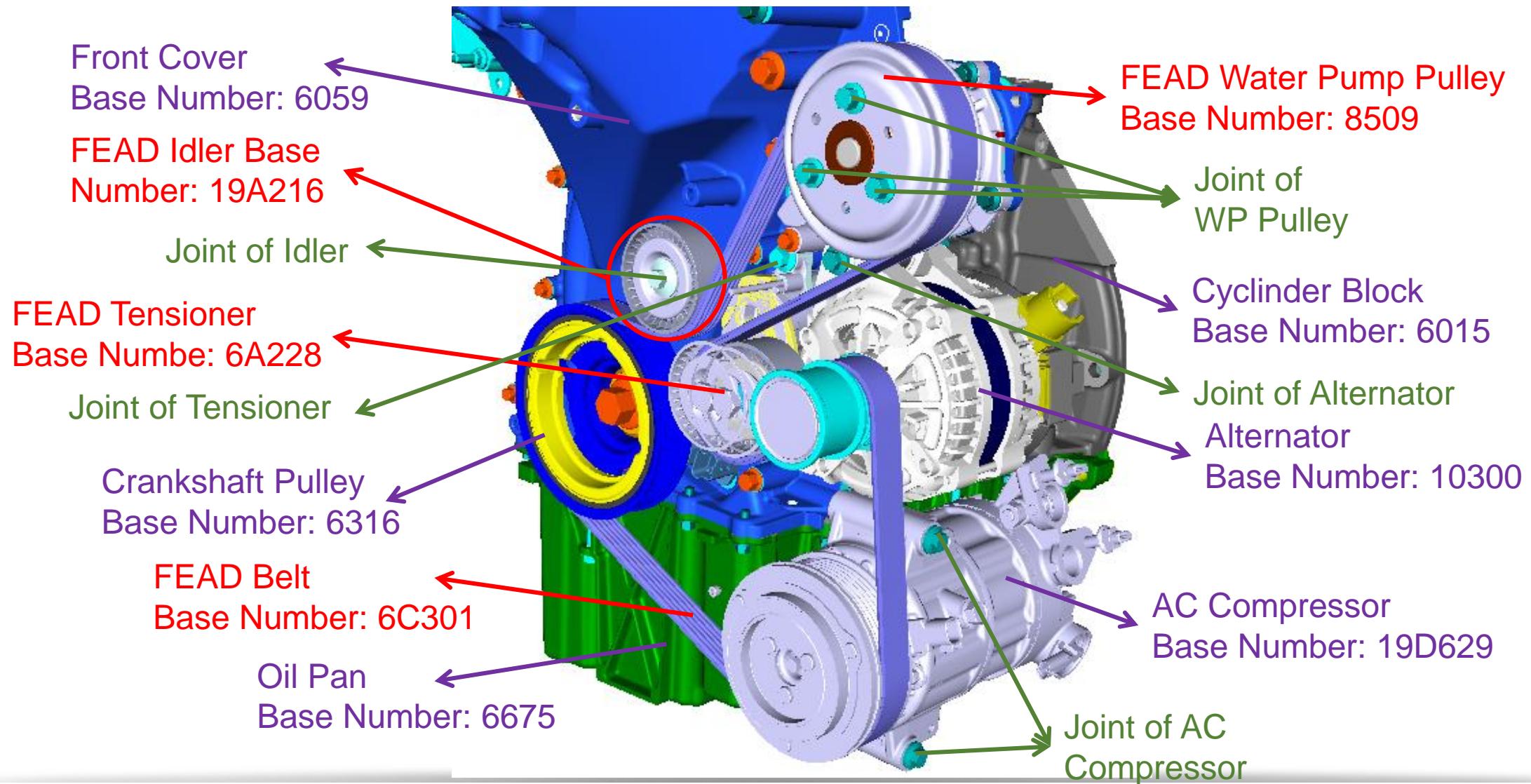


Grooved idler

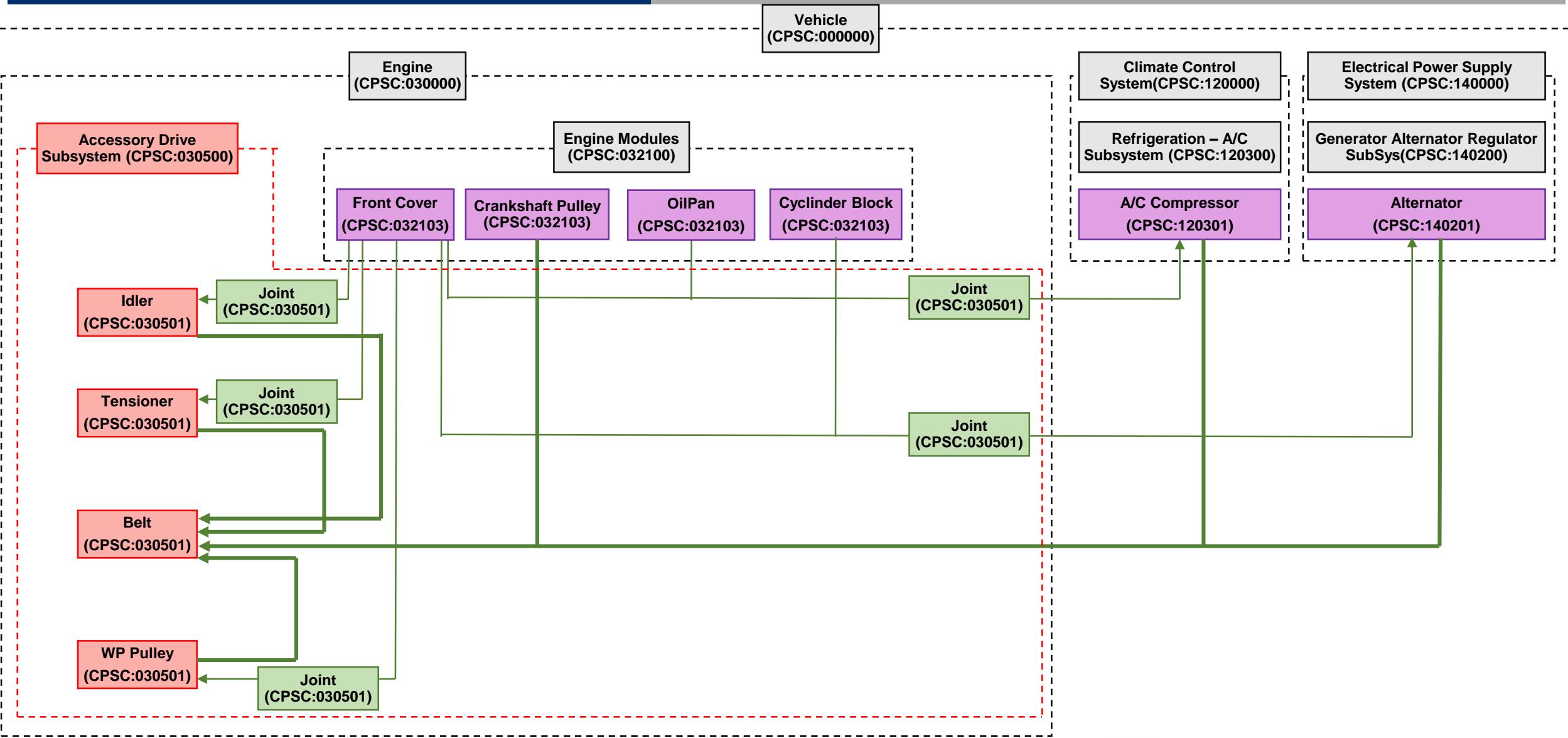


RELATION BETWEEN FEAD PARTS

Please see the FEAD part (red and green) and the other related parts (purple) layout below.



BOUNDARY DIAGRAMS OF THE PARTS



FEAD PART LIST AND REQUIRED INFORMATIONS

Prefix*	Base*	Suffix*	Name*	CPSC*	Engineering Commodity	Remarks (Fastener Attaching Statement)
	W500025	S437	BOLT M8X30 HF 8	030501	FASTENERS (PMT400)	FROM: 6A228; TO: 6059;
	W500214	S437	BOLT M6X20 HF PIL 8	030501	FASTENERS (PMT400)	FROM: 19A216; TO: 6059 ;
	W704693	S442	BOLT M8X95 HF PIL 8	030501	FASTENERS (PMT400)	FROM: 10300; TO: 6059 & 6015 ;
	W704693	S442	BOLT M8X95 HF PIL 8	030501	FASTENERS (PMT400)	FROM: 19D259; TO: 6059 & 6675 ;
	W714914	S303	BOLT M8X13 HF SERR STL	030501	FASTENERS (PMT400)	FROM: 8509; TO: 6059;
C1BQ	8509	AC	CVR-ACC DRV	030501	FEAD	
CM5Q	19A216	AB	IDL ASY ACC DRV BEL	030501	FEAD	
CM5Q	6A228	BA	TENS ASY ACC DRV	030501	FEAD	
CM5Q	6C301	BA	BEL ACC DRV	030501	FEAD	



Model Writer

EGR (Exhaust Gas Recycling) System

06.11.2016

EGR (EXHAUST GAS RECYCLING) SYSTEM

- **EGR – Exhaust Gas Recycling** is a method of controlling combustion through the addition of exhaust gases into the intake side of an internal combustion engine.

In its simplest form an EGR system comprises of the following steps

1. Exhaust gases pass through a take-off tube connected to the exhaust system.
2. The exhaust gas flow is metered through a control valve
3. A proportion of exhaust gases are passed into the combustion chamber through the engine intake

Internal EGR - Engines equipped with variable valve timing on the intake and exhaust camshafts are able to trap exhaust gasses within the cylinder by not fully expelling it during the exhaust stroke, dramatically increasing the residual gas fraction. This is called internal EGR. All the benefits of EGR are not realised by this method. However, in certain cases, internal (uncooled) EGR is preferred:

Gasoline – At low loads to reduce pumping losses while maintaining combustion stability

Diesel – At low loads and during cold start

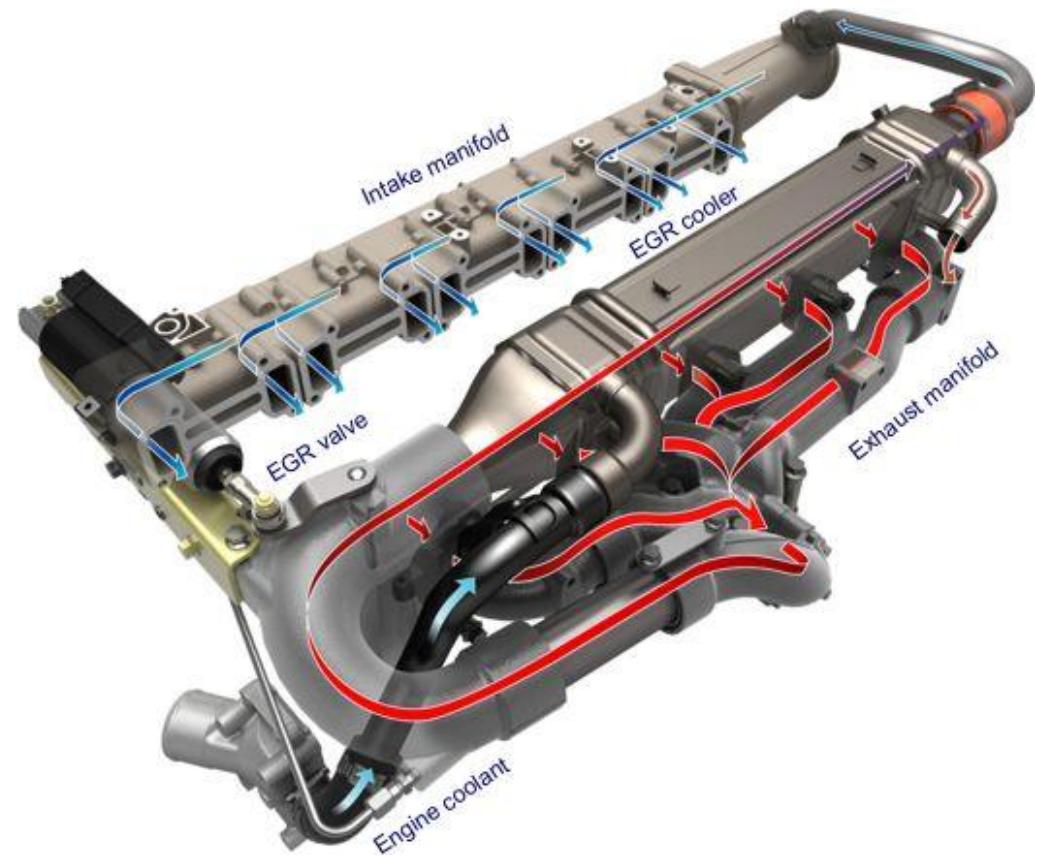
EGR Cooler Function

Diesel - EGR cooling reduces engine out NOx and increases intake charge density, allowing for increased EGR % without compromising desired air/fuel ratio.

Gasoline - For downsized boosted applications, cooled EGR not only reduces NOx but also improves fuel economy by mitigating knock and reducing or eliminating enrichment to maintain proper turbine temperatures.

EGR Valve Function

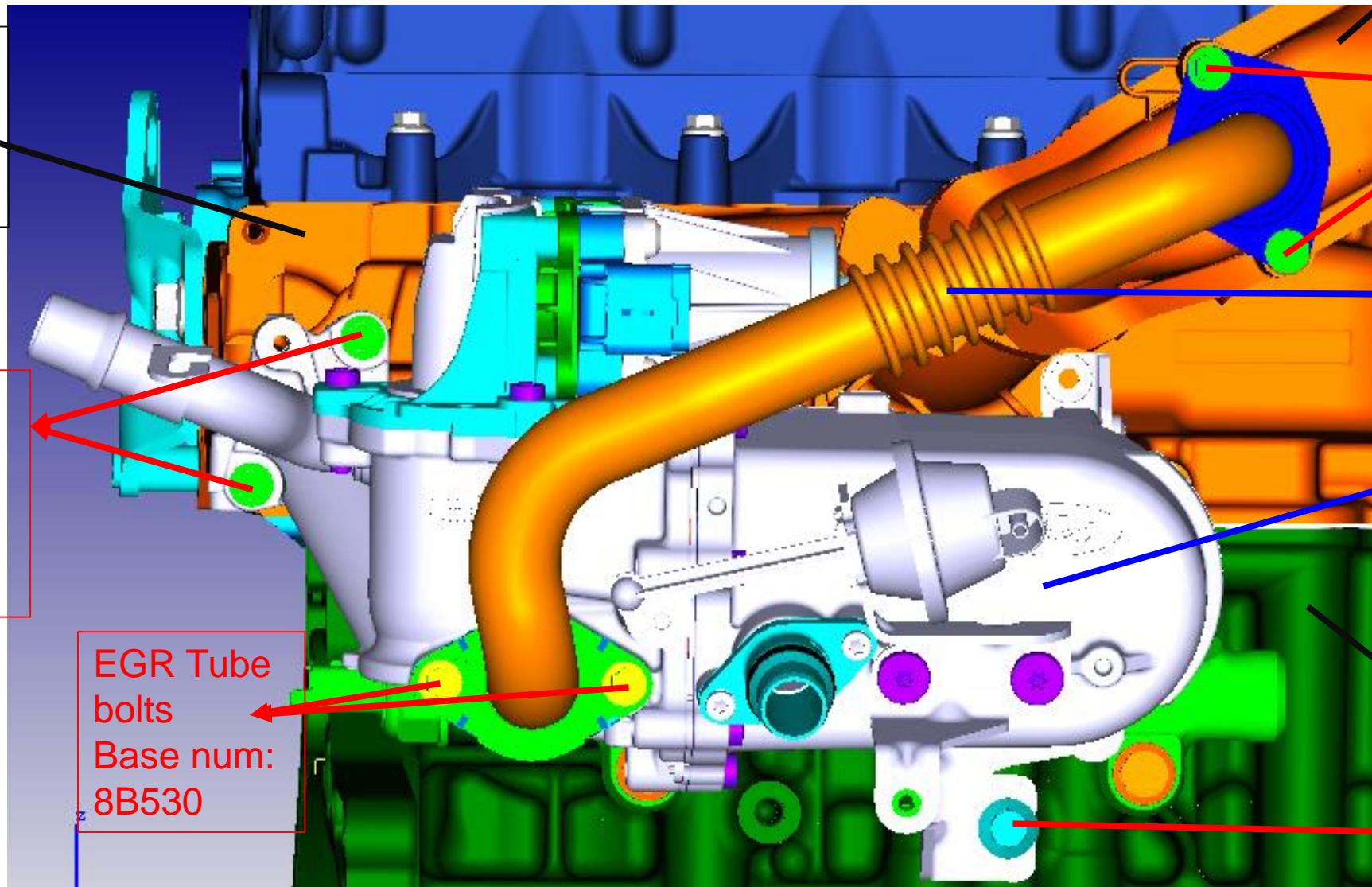
An EGR Valve is used to control the amount of EGR flow which enters the intake system. The valve assembly contains a DC electronically controlled motor, a gearbox and usually a poppet on seat valve. Depending on design and material specification, the assembly can be placed either side of the EGR cooler. Valves exposed to high temperatures will have additional channels in the casting for coolant flow.



OVERVIEW OF EGR SYSTEM

Intake manifold tube
Base num: 9S331

Cylinder
head
Base num:
6090



EGR Tube
bolts
Base num:
8B530

EGR Tube
Base num: 9D477

EGR Module
(Cooler &
Valve) base
num:
9D475

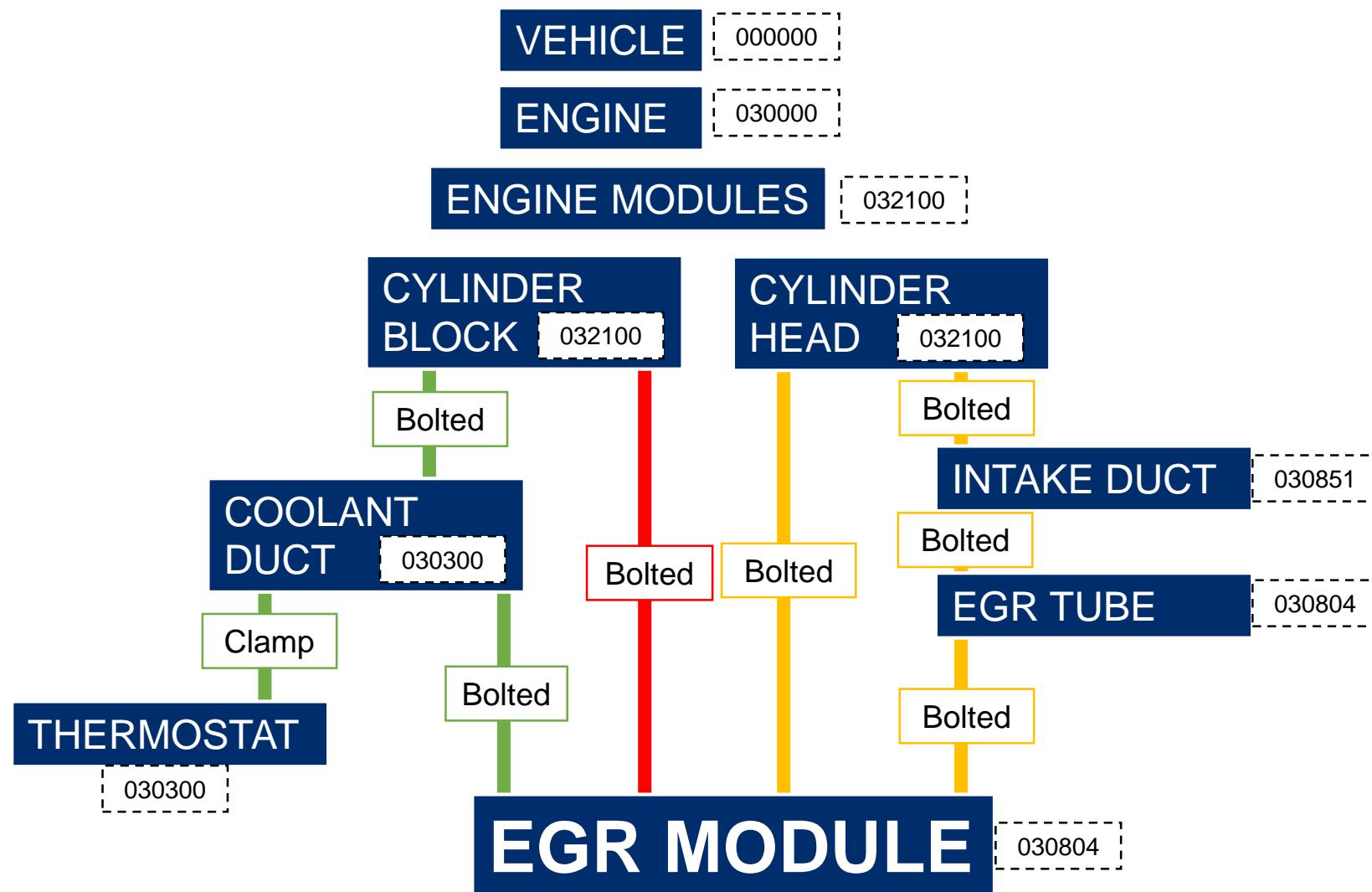
Cylinder block
Base num: 6015

EGR Tube
bolts
Base num:
8B530

EGR module bolt
Base num:c3

EGR parts
Joints
Related
parts

BOUNDARY DIAGRAMS



Joints are shown with lines.

- Liquid transfer
- Gas transfer
- Structural
- CPSC

CPSC Code is a 6-digit numerical code that is defined to identify commodities. In our example, our commodity is in ENGINE AS SHIPPED. Which means, the parts that are assembled in engine plant, there are also other engine parts that are assembled in vehicle plant afterwards, those parts usually have other unique CPSC codes.

BOM

Prefix*	Base*	Suffix*	Name*	CPSC*	Engineering Commodity	Remarks (Fastener Attaching Statement)
	W500223	S437	BOLT M8X20 HD PIL	030804	FASTENERS (PMT400)	FROM: 6A228; TO: 6059;
FM5Q	9D475	AA	VLV-ASY E/G/R	030804	DRESSED ENGINE	
7M5Q	8B530	AA	SC CVR WTR PM	030804	DRESSED ENGINE	FROM: 9D475; TO: 6090
FM5Q	9D477	AA	TUB ASY E/G/R OLET	030804	DRESSED ENGINE	
AV6Q	9S331	AA	INTK MANF TUB	030851	DRESSED ENGINE	

RELATED DESIGN RULESETS

EGR Systems Design Rules

Engine Fasteners (Bolted joints) Design Rules

Engine Sealing Design Rules

KPAC Set ID	Title	Author	Type	KPAC Set ID	Title	Author	Type
0308_ES-S1	Exhaust Gas Recirculation Design Rule Saved Set for Diesel	Ewen, Ken (kewen)	Check List	PT_FAST-S13	Cup Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S13	Dryseal Pipe Thread Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S13	Dowel and Bushing Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S13	Fastener and Joint Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S13	Threaded Fastener Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S13	Wrench and Socket Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S16	Ball Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S16	Port Plug Design Rules Saved Set	Murphy, Mark (mmurphy9)	Check List
				PT_FAST-S16	Thread Forming Screws in Plastic Saved Set	Murphy, Mark (mmurphy9)	Check List

KPAC Set ID	Title	Author	Type
0310_ES-S11	Cam Cover Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S11	Dynamic Seal Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S11	Fasteners Rules for Covers Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S11	Front Cover Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S11	Oil Pan Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S11	Rear Seal Retainer Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S11	Static Seal Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List

SYSTEM CHECK

- Three types of EGR systems are listed. We need to choose the correct system to see the correct rules.

KPAC Set ID	Title	Author	Type
0308_EG-S1:	Exhaust Gas Recirculation Design Rule Saved Set for Diesel 	Ewen, Ken (kewen)	Check List
0308_EG-S1:	Exhaust Gas Recirculation Design Rule Saved Set for Gasoline 	Ewen, Ken (kewen)	Check List
0308_EG-S1:	Exhaust Gas Recirculation Design rule Saved Set for Low Pressure 	Ewen, Ken (kewen)	Check List

The particular example we are working on is using a **High Pressure EGR System** on a **DIESEL** engine. So we will need to choose the first ruleset. The other 2 rulesets are not related to our system, they are there for **GASOLINE** and **Low Pressure EGR** engines..

SYSTEM CHECK

- We have identified the joints in our systems. EGR means Exhaust Gas Recirculation, which means it includes a lot of gas transfers. Additionally, EGR systems usually includes a cooler as well. In our example system, the cooler and the valve of the EGR system is packaged inside one module. So our system also includes coolant (liquid) transfer, to cool down the gases.
- As we have all these gas and liquid transfers, some of our related joints should be leak proof, sealed properly. To support this, there are design rules for **ENGINE SEALING**.

KPAC Set ID	Title	Author	Type
0310_ES-S1:	Cam Cover Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S1:	Dynamic Seal Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S1:	Fasteners Rules for Covers Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S1:	Front Cover Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S1:	Oil Pan Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S1:	Rear Seal Retainer Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List
0310_ES-S1:	Static Seal Design Rule Saved Set	Smith, Thomas (tsmith57)	Check List

- In this list, there are some design rulesets for specific systems. As there is no specific design ruleset for EGR systems inside here, we will choose Static Seal Design Ruleset. The others, for example Cam Cover design ruleset should be used when the related part is indeed a cam cover.

SYSTEM CHECK

- Lastly, as there are also fasteners that connects the parts to eachother, we have general rules for these fixations. As you can see on the below list there are a few different rulesets for that.

KPAC Set ID	Title	Author	Type
PT_FAST-S13	Cup Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S13	Dryseal Pipe Thread Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S13	Dowel and Bushing Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S13	Fastener and Joint Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S13	Threaded Fastener Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S13	Wrench and Socket Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S16	Ball Plug Design Rule Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S16	Port Plug Design Rules Saved Set	Murphy, Mark (mmurphy9)	Check List
PT_FAST-S19	Thread Forming Screws in Plastic Saved Set	Murphy, Mark (mmurphy9)	Check List

You can see that some special types of fasteners (like ball plug, dowel/bushing, cup plug etc. has unique design rulesets. In our example, we do not have a special feature on any of our fixing points, so we will just use the generic Fastener and Joint design rule and Threaded Fastener design ruleset.



Model Writer

Work Package 1 - Industrial Use Cases and Requirements

Integration with Application Lifecycle Management tools

Eray TÜZÜN (HAVELSAN)

Requirement Work Item

Customized Form

Attributes

WorkItem Number P

History

Discussion

Microsoft Word - TrQBSsrs.SRS

RequisitePro - The Learning Project - Traditional

Create Req... Mod. Req. [] Req. Doc Project GoTo Views Window Help Exit

Document(s) opened Doc: QBS Software Requirement

File Edit View Insert Format Tools Table Window Help

Normal Times New Roman 10 B I U

125% 1 2 3 4 5

4. Non-Functional Requirements

4.1. Safety Requirements

4.2. Security and Privacy Requirements

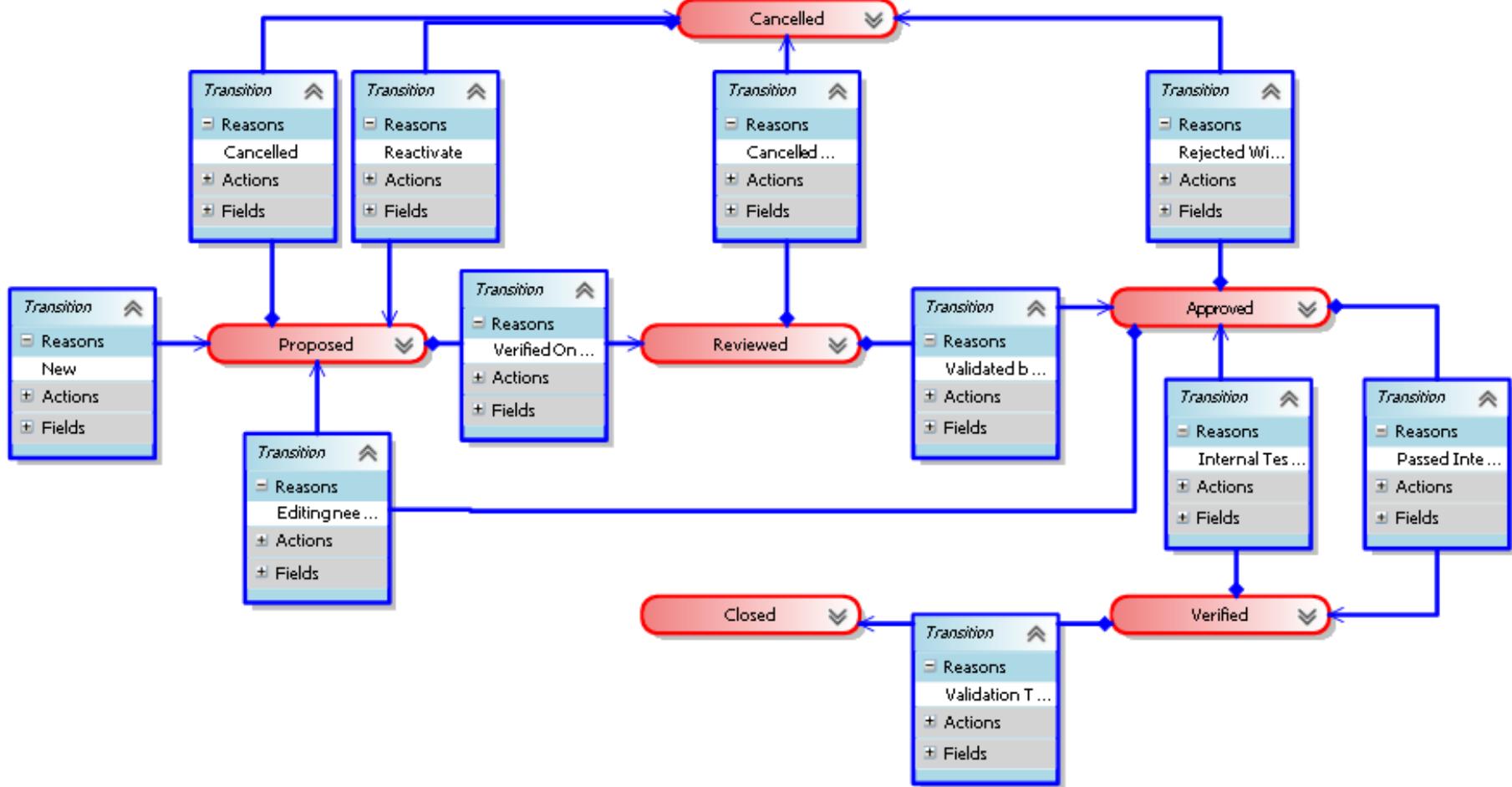
- [SR26 The system shall allow editing based upon security set up by the system administrator]
- [SR27 The system shall generate an error message stating that the current user has entered an in security and prompt the user to re-enter the information.]

4.3. Environmental Requirements

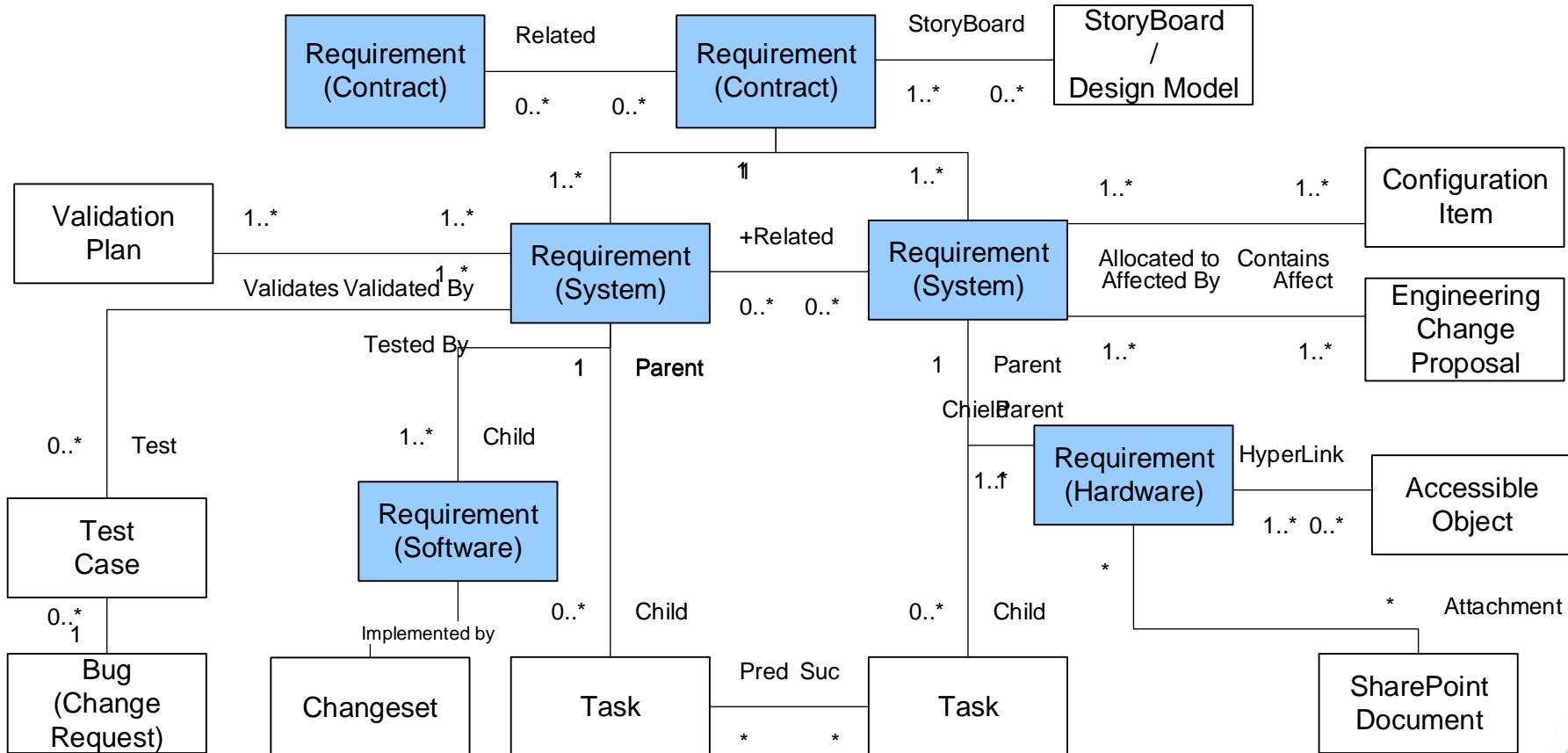
Detail environmental requirements as needed. For hardware based systems, environmental issues can include temperature, shock, humidity, radiation, etc. For software applications, environmental factors can include usage conditions, user environment, resource availability, maintenance issues, error handling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

Page 11 Sec 4 11/14 At 4.3" Ln 14 Col 1 REC TRK EXT OVR WPH

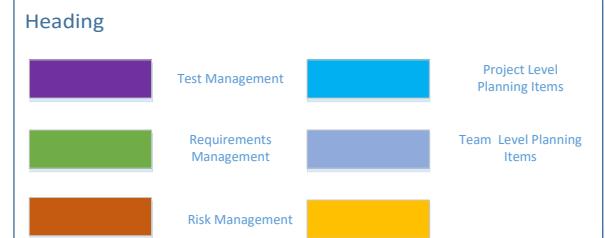
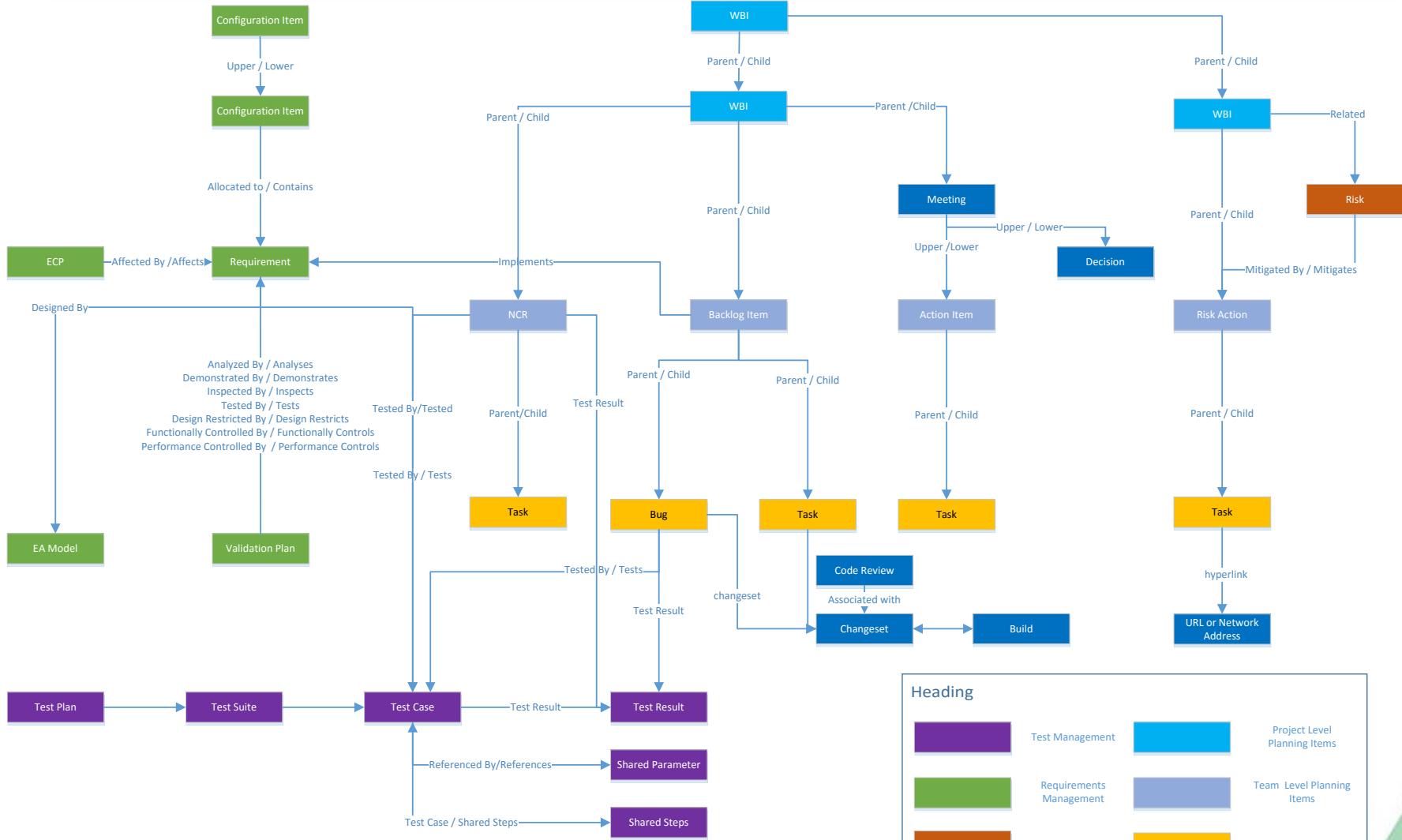
Requirement LifeCycle



Requirements Traceability



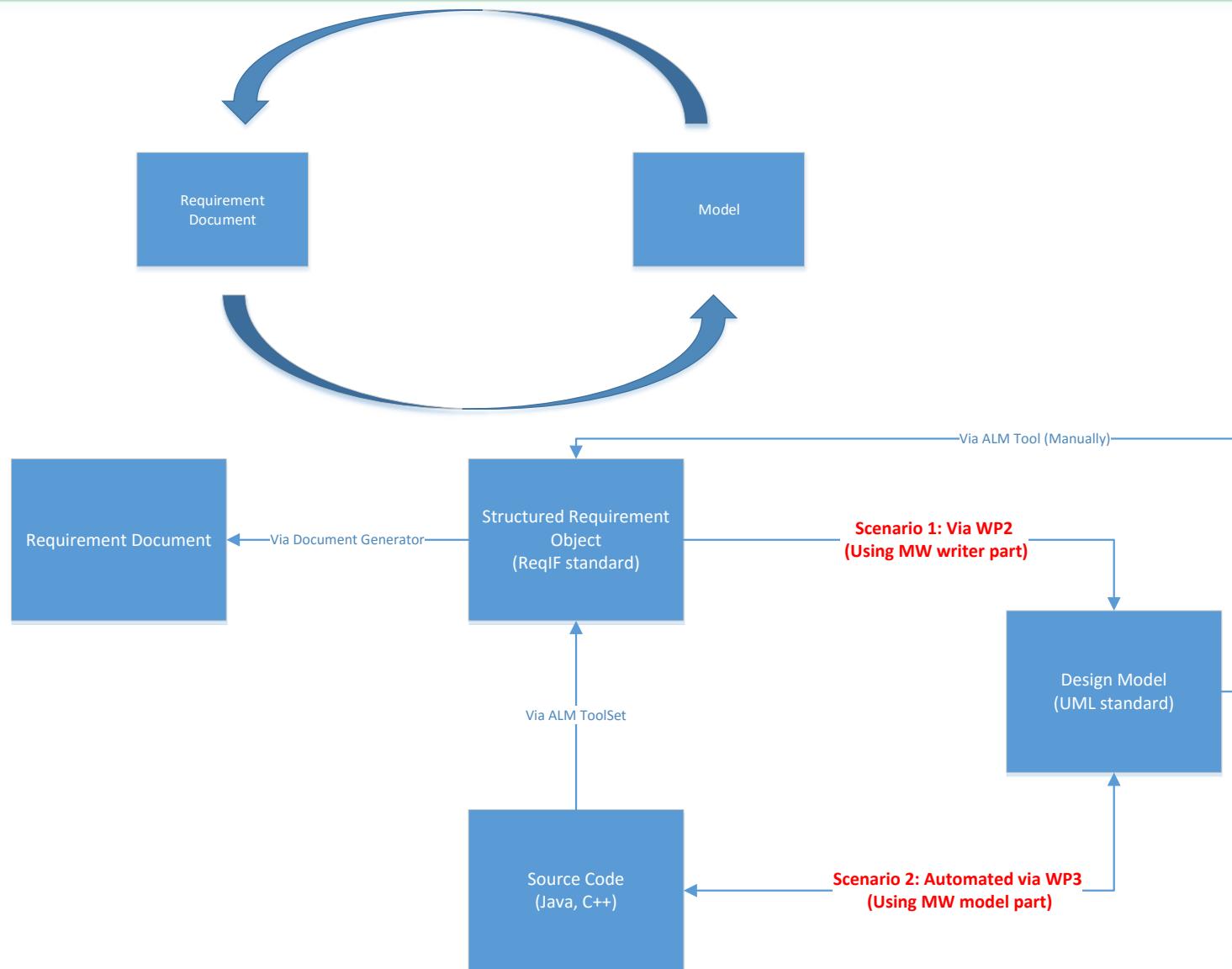
Requirements in ALM



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

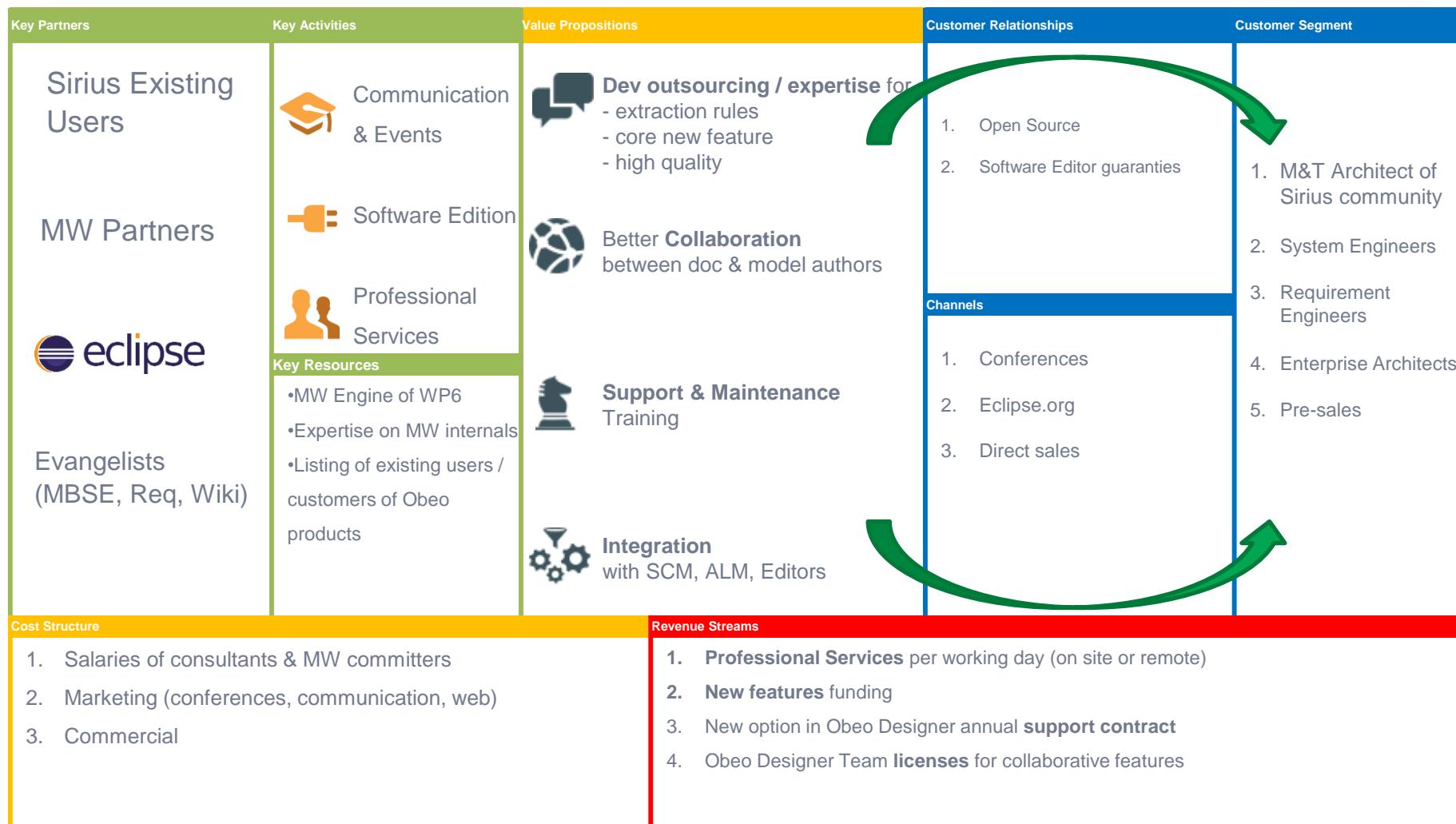


Exploitation, Dissemination, Standardisation (WP7)

Yvan Lussaud, OBEO



Business Model Canvas - Obeo



UNIT as a tool and technology provider

Integration with Siemens Teamcenter PLM visualization

Required for FORD-OTOSAN use case

Using JT Open Toolkit's C++ application programming interface for 3D product information

ISO 14306:2012 Industrial automation systems and integration -- JT file format specification for 3D visualization

Integration with IBM Rational DOORS or Esterel SCADE Suite

Required for HAVELSAN use case

using ReqIF (Requirement Interchange Format) Standard and Eclipse's Requirement Management Framework (RMF)

Obeo Sirius	<ul style="list-style-type: none">• 200 clients and about 10000 prospects• 60% of Obeo's revenue
MBSE	<ul style="list-style-type: none">• System analysis and live cycle tooling• About 50 million euros (IBM, Siemens, Dassault System, PTC, ...)
Enterprise Architecture	<ul style="list-style-type: none">• 100 million euros in France alone• 9000 people certified Togaf 9 in the world
Competitors	<ul style="list-style-type: none">• Reqifify, Recycle, DoorsNG• Assets of MW: Open source, Interoperability, not only focused on requirements

	Actual	Goal
Number of users (not counting MW participants)	0	10
Number of projects using MW outcomes	0	6
Unique visitors of MW website per month	0	300

Release of Tarski platform (WP3) – UNIT and KOCSISTEM

- Automated analysis and dynamically configurable semantics of traceability links
- Standalone and open source <https://github.com/ModelWriter/WP3>

Achievement: Synchronization of EGR & FEAD designs with specifications

- Improve review process of engineering teams
- Impact analysis of design changes

Release of M2Doc – Obeo

- ModelWriter helped us to ramp-up MS Word document creation and Java API
- New product of MS word document generation M2Doc
- Future new feature for traceability using ModelWriter mapping API

Release of Synchronization Engine and UI – Obeo

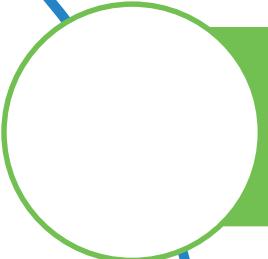
- Synchronization engine first version
- First implementation of GUI

Ontology consistency checking – CNRS and Airbus

- Identify and remove inconsistencies from the knowledge base
- Solution based on Airbus needs

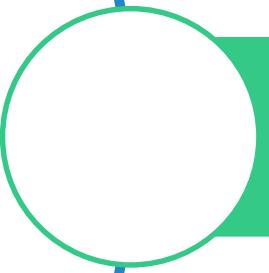
MS Word plug-in – UNIT and HISBIM

- Enable seamless integration with ModelWriter on Eclipse platform
- GUI is almost ready
- Working on integration of Eclipse platform and MS Word plug-in with respect to ModelWriter workspace

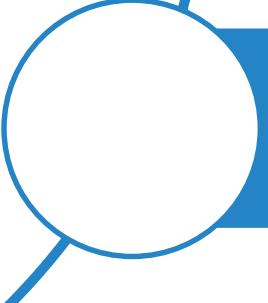


Release of Component Ontology – Airbus

- partially describes the vocabulary used in the SIDP documents and database
- Improves the SIDP model developed in first semester
- Used for semantic annotation and link creation

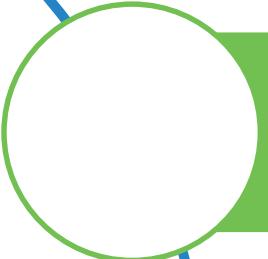


Web based services for Turkish language support – Mantis



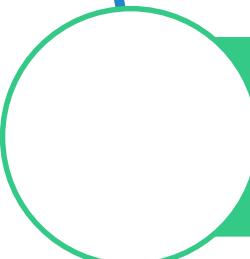
Highlight experience of industrial end users

- Airbus
- Havelsan
- Ford-Otosan



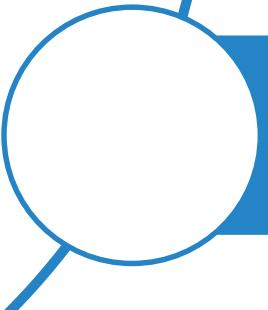
6th international ModelWriter Workshop (25 February 2015)

- Product owner review
- Integration of first release



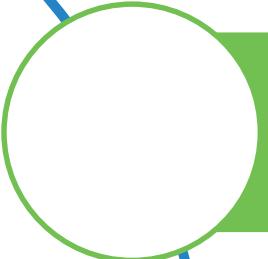
7th international ModelWriter Workshop (6 June 2016)

- At Airbus headquarter
- Brain storming and progress presentation
- Mapping base design and sketches of GUI by Obeo



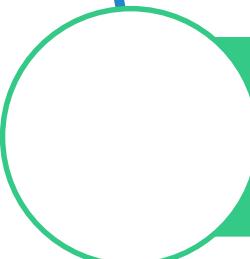
Airbus Friday TechTalk

- Demonstration of current state
- Presentation of objectives



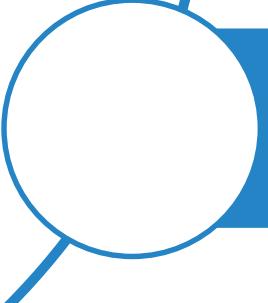
MPM4CPS (Multi-paradigm Modeling for Cyber-physical Systems) Cost Action

- UNIT and AIRBUS are Management Committee Members representing Turkey and France in this COST action



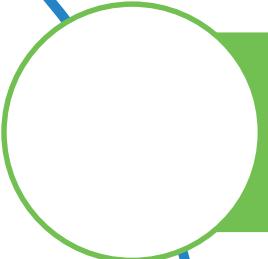
The European Research Network on Software-intensive Systems-of-Systems (SiSoS)

- UNIT is a participant in the network



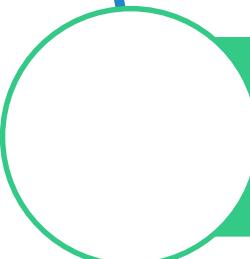
ITEA3-Assume project

- Ford-Otosan, Havelsan, KocSistem and UNIT participates in that project



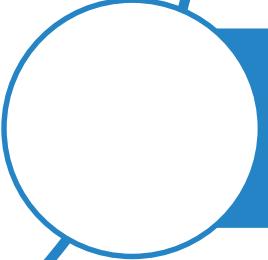
ACM Applied Computing Symposium

- Paper submitted and under review



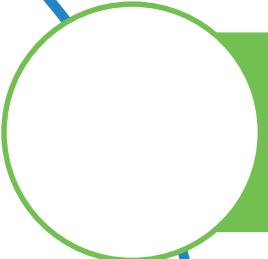
SAT/SMT/AR Summer School

- Poster presentation

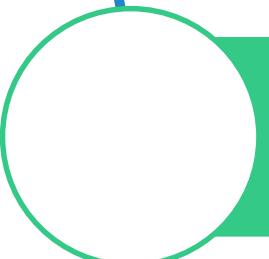


VTSA Summer School

- Poster presentation

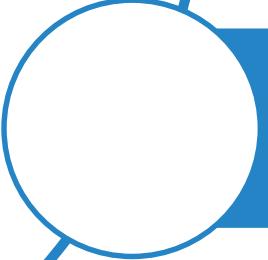


Use Obeo deployment of MW on Sirius documentation



Target developers as early adopters

- EclipseCon, SiriusCon, Eclipse Demo Camp



Move MW project to Eclipse foundation

Modular architecture

- Open source
- API
- Definition of domain model for mapping

Used standards

- EMF based of MOF
- XML serialization
- Standard ontology formats (owl, rdf, turtle, ...)

WP2 - Semantic Parsing and Generation of Documents and Documents Components

*Claire GARDENT, Bikash GYAWALI,
Anastasia SHIMORINA
CNRS / LORIA
Samuel CRUZ-LARA
University of Lorraine / LORIA*

WP2



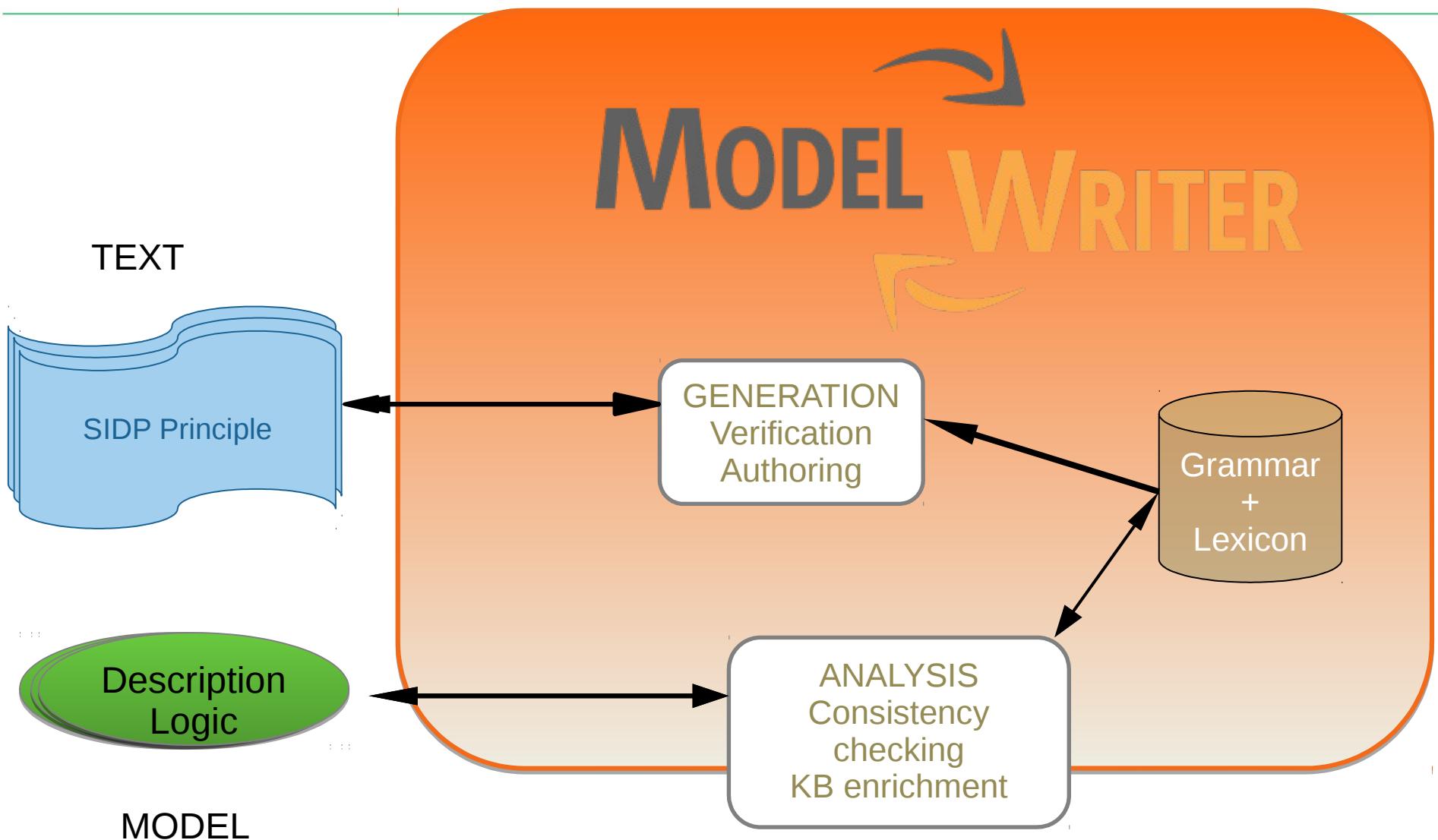
Goal: Provide tools and methods for:

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

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

Synchronising Text and Model



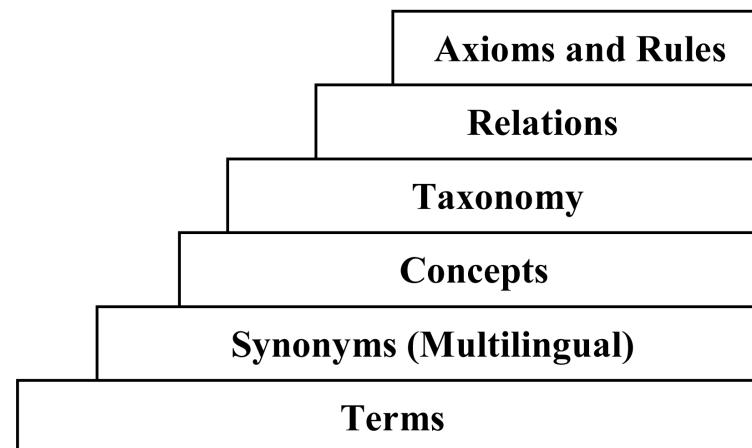
Related Work on Ontology Learning

Ontology Learning from Text

[Mädche and Staab 2000, Volker et al. 2007, Tablan et al. 2006, Zouaq and Nkambou 2008]

Restricted expressivity. Not applied to sentences (complex axioms).

Limited to : definition of new classes, creation of hierarchies between classes, definition of object and data-type properties, creation of instances, and setting of property values for instances



Related Work on Semantic Parsing

Deep parsing

[Currant et al. 2007, McCartney and Manning 2007]

First Order Logic Representations close to initial text. Trained on newspaper text (Penn Tree Bank).

==> Not easily adaptable to Description Logic and SLDP text.

Domain Specific Semantic Parsing

[Ge and Mooney 2009,Wong and Mooney 2007]

==> Require parallel text-data training corpus.

Open Domain Semantic parsing

[Kwiatkowski et al 2010, Bordes et al. 2012, Kwiatkowski et al 2013, Berant et al., 2013, Bordes et al. 2014, Wang et al. 2015]

==> Restricted to questions. Require parallel question-answer training corpus.

Related Work on Text Generation

Symbolic Approaches

[Dimitrios et al. 2007, Androtsopoulos et al. 2013, Power et al. 2010, Bontcheva et al. 2004]

Heavily dependent on hand-written modules.

Machine Learning Approaches

[Wong et al. 2007, Belz 2008, Angeli et al. 2010, Chen et al. 2008, Konstas and Lapata 2012a and 2012b]

Require parallel data-text training corpus.

Pattern-Based

[Duma et al. 2010, Blake et al. 2013, Schilder et al. 2013]

*Require large quantity of parallel or comparable text-data training corpus.
Limited Semantic Variability (set of RDF triples).*

Reversible Processing: Text <-> Model

Analysis: SIDP Rule → Description Logic

Generation: Description Logic → SIDP Rule

Verification by generation

Semantic Parsing of Complex Axioms

Pipe shall be identified by labels

$\text{Pipe} \sqsubseteq \exists \text{identificationArg2}^{-}.(\text{Identification} \sqcap \exists \text{identificationArg3}.\text{Label})$

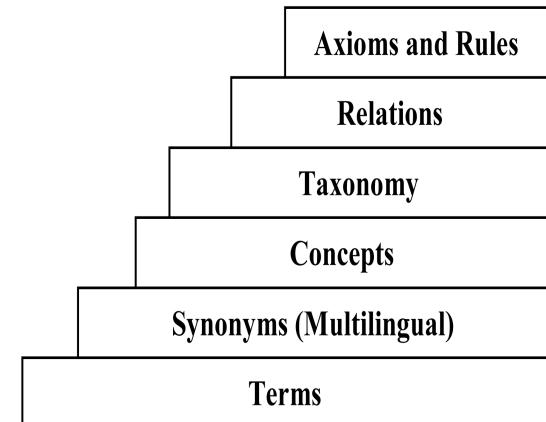
Executable Semantic Parsing on DL KBs

The output of semantic parsing is used to update a Description logic Knowledge Base and check the consistency of SIDPs
(system installation design principle)

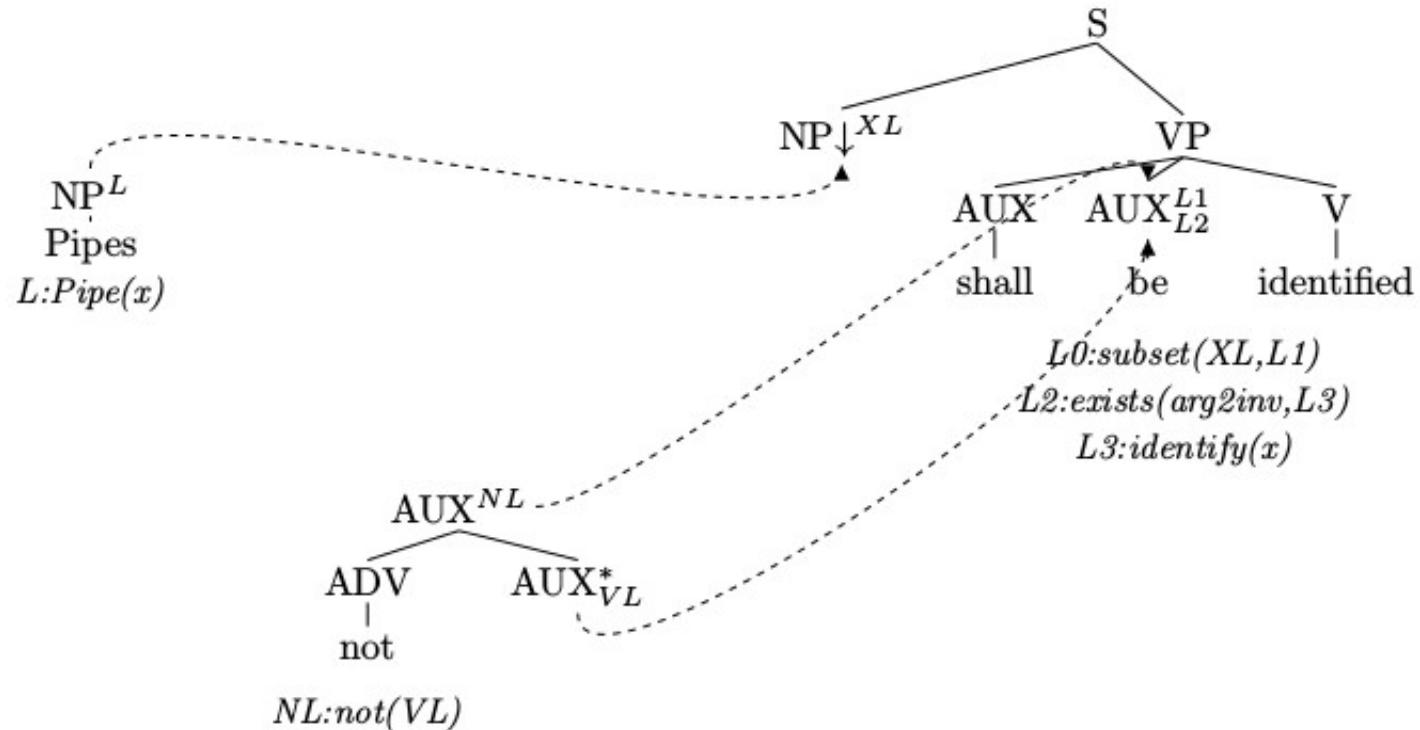
Genericity

No training corpus required.

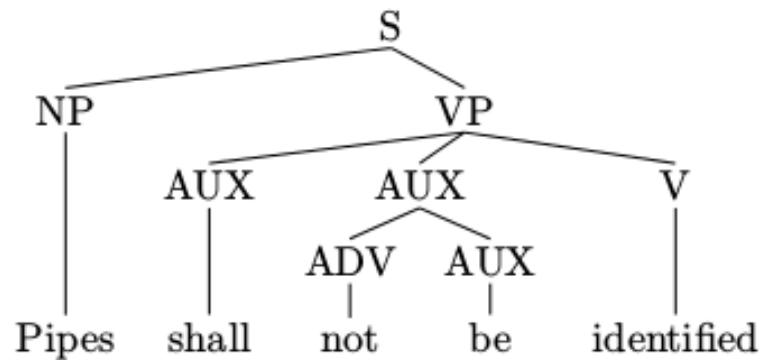
Adaptation to a new domain through grammar adaptation,
extension or induction



Grammar-Based Parsing and Generation



Grammar-Based Parsing and Generation



PL:Pipe(x) L0:subset(PL,NL) NL:not(VPL)

VPL:exists(identifyA2inv,VL) VL:Identify(x)

Pipe ⊑ ∃¬identifyA2⁻.(Identify)

Semantic Variations

Logical Operators	
<u>Only</u> S shall be used by O	$\neg S \sqsubseteq \neg \exists useA2^-.(use \sqcap \exists by.O)$
S should be used by <u>all</u> O	$O \sqsubseteq \exists by^-.(Use \sqcap \exists useA2.S)$
S shall <u>not</u> be used by O	$S \sqsubseteq \neg \exists useA2^-.(Use \sqcap \exists by.O)$
Word Order	
S shall be used by O <u>only</u>	$S \sqsubseteq \neg \exists useA2^-.(Use \sqcap \exists by.\neg O)$
<u>All</u> S shall be used by O	$S \sqsubseteq \exists useA2^-.(Use \sqcap \exists by.O)$
Arity	
S shall be used	$S \sqsubseteq \exists useA2^-.(use)$
S shall be used by O	$S \sqsubseteq \exists useA2^-.(use \sqcap \exists by.O)$
S shall be used by O on PO	$S \sqsubseteq \exists useA2^-.(use \sqcap \exists by.O \sqcap \exists on.PO)$
Sentence Structure	
S shall be used by O <u>before</u> entering connections	$(Use \sqcap \exists useA2.S \sqcap \exists by.O) \sqsubseteq \exists before.(Enter \sqcap \exists enterA2.Connections)$
Modifiers	
S shall be used <u>directly</u> by O	$S \sqsubseteq \exists useA2^-.(Use \sqcap \exists directly.(\exists by.O))$
S shall be used by O <u>between</u> C and D	$S \sqsubseteq \exists useA2^-.(Use \sqcap \exists useA3.(O \sqcap \exists betweenA1^-.(Between \sqcap \exists betweenA2.C \sqcap \exists betweenA3.D)))$

Updating the Model and Checking Consistency



The Semantic Representations output by the parser are converted to Description Logic

$l_0 : A(x)$	$\Rightarrow :A$
$l_0 : \text{exists}(R, l_1) \ l_1 : C$	$\Rightarrow \text{ObjectSomeValuesFrom}(:R \ \tau(C))$
$l_0 : \text{subset}(l_1, l_2) \ l_1 : C_1 \ l_2 : C_2$	$\Rightarrow \text{SubClassOf}(\tau(C_1) \ \tau(C_2))$
$l_0 : \text{and}(l_1, l_2) \ l_1 : C_1 \ l_2 : C_2$	$\Rightarrow \text{ObjectIntersectionOf}(\tau(C_1) \ \tau(C_2))$
$l_0 : \text{not}(l_1) \ l_1 : C$	$\Rightarrow \text{not}(\tau(C))$

E.g.,

Pipes should be identified by labels

$l_1 : \text{Pipe}(x) \ l_0 : \text{subset}(l_1, l_2) \ l_2 : \text{exists}(\text{identifyA2inv}, l_3) \ l_3 : \text{and}(l_4, l_5)$
 $l_4 : \text{Identify}(z) \ l_5 : \text{exists}(\text{by}, l_6) \ l_6 : \text{Label}(y)$

`SubClassOf(Pipe ObjectSomeValuesFrom(identifyA2inv
ObjectIntersectionOf(Identify ObjectSomeValuesFrom(by Label))))`

The formula is added to the AIRBUS KB and Hermit is used to check for consistency

Experimental Setup and Results for Parsing

Grammar: 52 trees

Lexicon: 10781 lexical entries

Parsing algorithm: CKY + Robustness mechanism to skip unknown words

Input: 991 System Installation Design Principles

	Complete Parse	Partial Parse	Failure
Simple SIDP	132	329	24
Complex SIDP	0	496	10
All SIDP	132 (13%)	825 (83%)	34 (3%)

Updating the Model using Parsing Results (Complete Parses)

CONCEPTS	Nb. of new Concepts	184
	Nb. of Existing Concepts	30
PROPERTIES	Nb. of New Properties	62
	Nb. of SIDP Axioms (from Parsing)	132
SIDP AXIOMS	Nb of Invalid Axioms	0
	Nb. of Redundant Axioms	2
ALL	Total Nb. Of Added Elements	376
	Nb. of Axioms in Initial KB	12469
	Nb. of Axioms in Enriched KB	13029

Updating the Model using Parsing Results (All Parses)



CONCEPTS

Nb. of new Concepts	667
---------------------	-----

PROPERTIES

Nb. of Existing Concepts	79
--------------------------	----

SIDP AXIOMS

Nb. of New Properties	98
-----------------------	----

Nb. of SIDP Axioms (from Parsing)	957
-----------------------------------	-----

Nb of Invalid Axioms	61
----------------------	----

Nb. of Redundant Axioms	125
-------------------------	-----

Nb. of Inconsistent Axioms	20
----------------------------	----

ALL

Nb. of added SIDP Axioms	749
--------------------------	-----

Total Nb. Of Added Elements	1514
-----------------------------	------

Nb. of Axioms in Initial KB	12469
-----------------------------	-------

Nb. of Axioms in Enriched KB	14650
------------------------------	-------

Generation Results

Grammar: 52 trees

Lexicon: 10781 lexical entries

Generation algorithm: Tabular + Polarity Filtering

Input: 957 Description Logic Axioms derived from the AIRBUS System Installation Design Principles

	Success	Failure
Simple SIDP	448	13
Complex SIDP	470	26
All SIDP	918 (96%)	39 (4%)

Verifying Parsing Results Using Generation

BLEU	< 0.33	> 0.32 and < 0.67	> 0.66
Complete Parses (S)	1	0	131 (14%)
Complete Parses (C)			
Partial (Simple)	143	117	69
Partial (Complex)	396	91	9
All Parses	540 (56%)	208 (22%)	209 (22%)

Regenerating from the DL formula derived through parsing from an SIDP:

- Produces a sentence identical to the input SIDP for complete parses
- Produces a sentence highly similar to the input SIDP in 44% of the cases for partial parses

Perspectives and Future Work

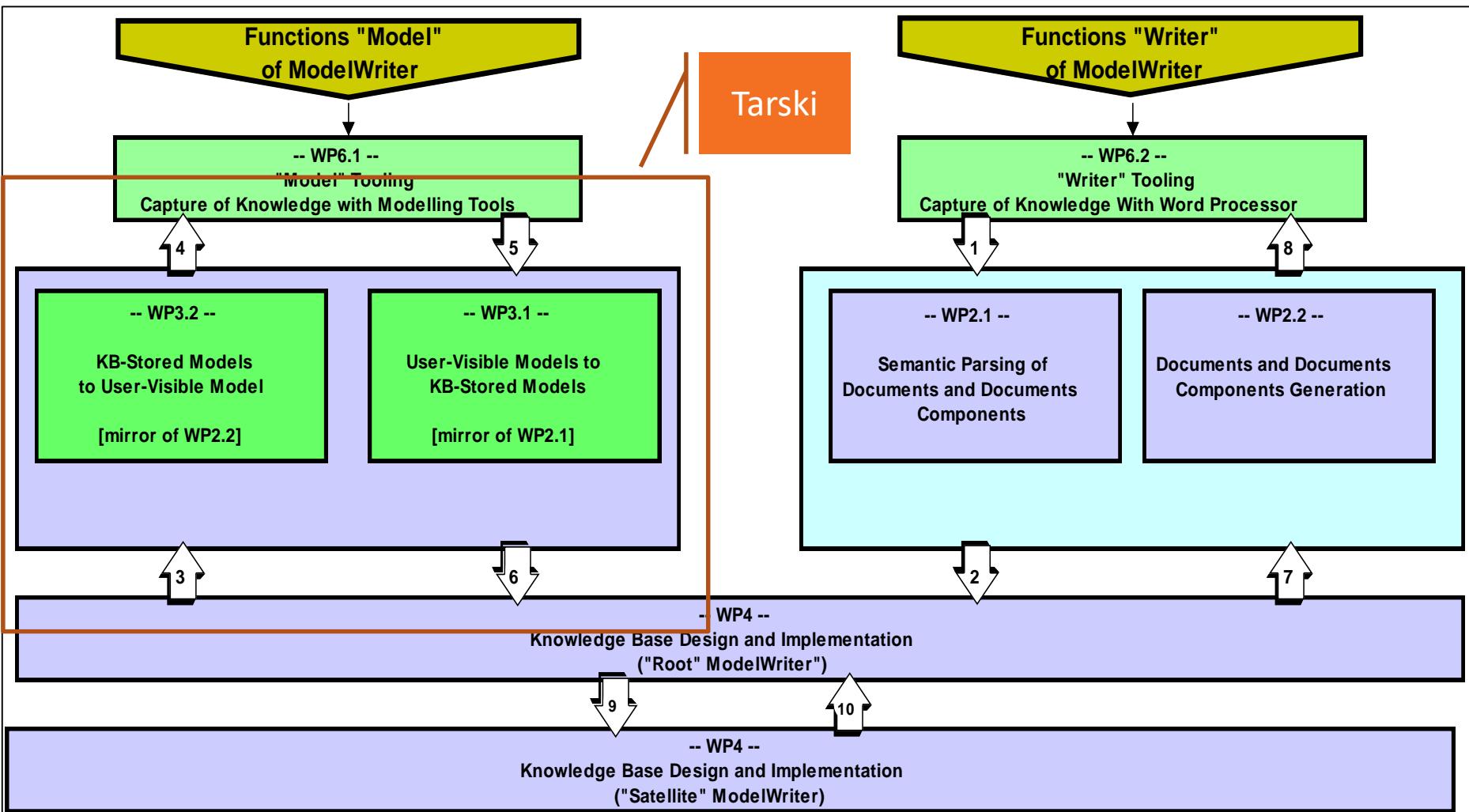
- Improve lexicon construction using chunking
- Improve coverage on complex sentences including conditions
- Querying the KB (support for AIRBUS engineers)
- Improve robustness and genericity (experiment with deep learning approaches using data expansion techniques and sequence to sequence models)

WP3 - Model to/from Knowledge Base Synchronization Mechanism

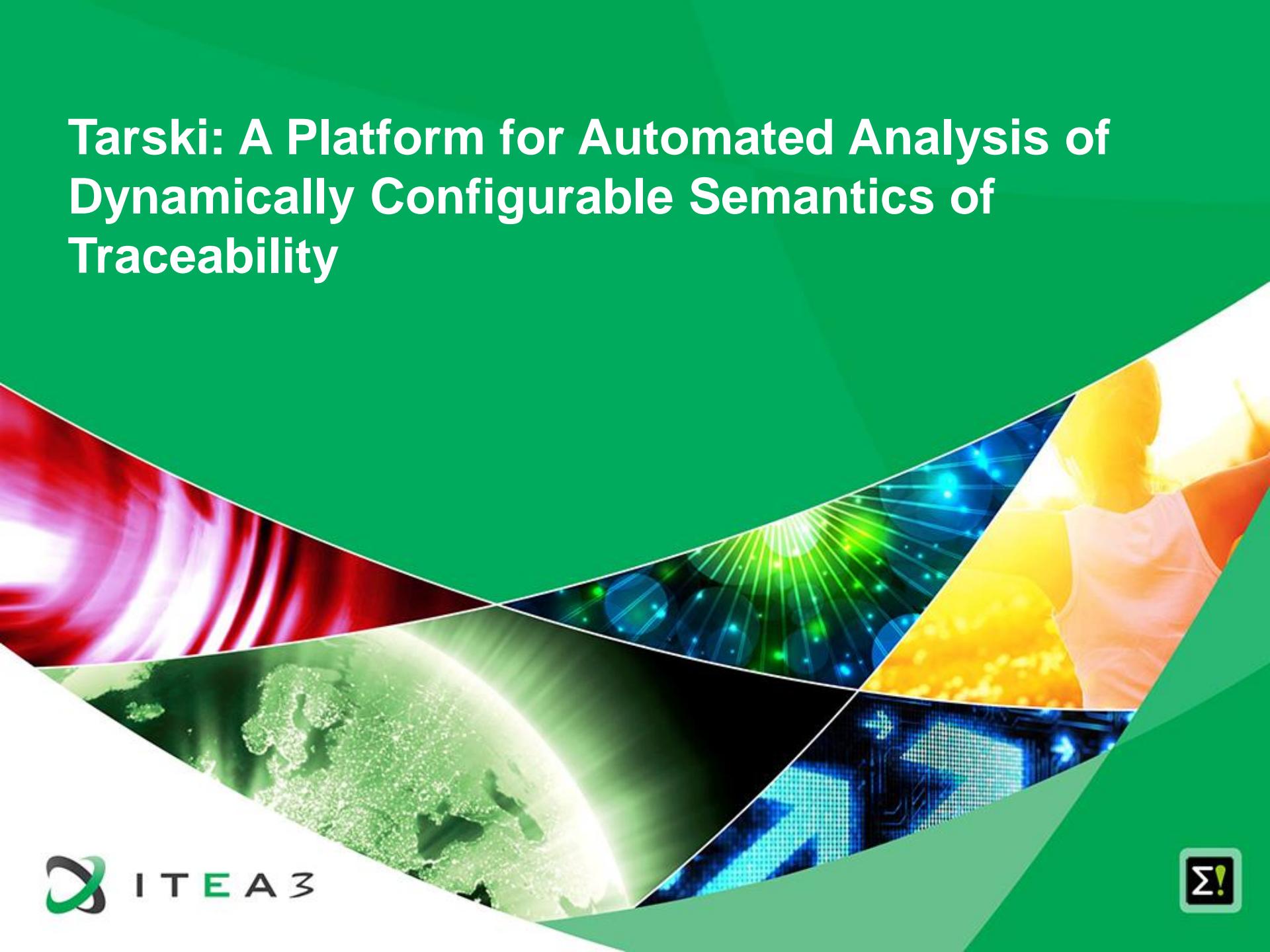
*Ferhat Erata, Work Package Leader
UNIT Information Technologies R&D Ltd.*

Technological components & interactions

Collaboration by WP interactions



Tarski: A Platform for Automated Analysis of Dynamically Configurable Semantics of Traceability

The background of the slide features a dynamic, abstract collage of various images. It includes a red and white striped pattern, a green globe showing Europe and Africa, a blue and green glowing particle field, a yellow and orange organic shape, a blue digital circuit board, and a close-up of a person's face. These images are set against a light green gradient background.

Challenges of Traceability in Industry

Semantically meaningful traceability

- traceability relations should have a rich semantic meaning instead of being simple bi-directional referential relation

Configurability of traceability (possibly dynamically)

- the semantics of traceability is often statically defined
- the semantics cannot be easily adapted for the needs of different projects.
- different traceable elements and the types of relations exist in industrial settings.

Several industries demands formal proofs of traceability

Consistency checking and repairing broken trace links

Technical Contributions @Tarski

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metu\Runtime-EclipseApplication

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

Customer Requirements Specification.md

```

1# Customer Requirements Specification
2
3## UC-1 Create a new SpecObject
4
5Note that the Specification Editor is the main interface for users. Therefore,
6creating SpecObjects in this editor is the main success scenario.
7
8### Precondition
9
10ReqIF model exists and is open.
11
12### Main Success Scenario
13
141. We assume that a Specification exists and is open (not required for alternative
15scenario)
162. Open a row's context menu (or in the empty editor space)
173. Select the Child or Sibling submenu.
4. Select the desired SpecObject Type (or none) from the submenu.
5. This results in a new SpecHierarchy being created that is linked to a newly

```

Specification Source

Problems Target Platform State ModelWriter Traceability View Console ModelWriter Source Mapping View Markers Properties ModelWriter Contextual View

contract: 1
fulfills: 2
refines: 2
requires: 2
satisfiedBy: 2
system: 3

Specification

ContractRequirement

SystemRequirement

Code

SystemRequirement0

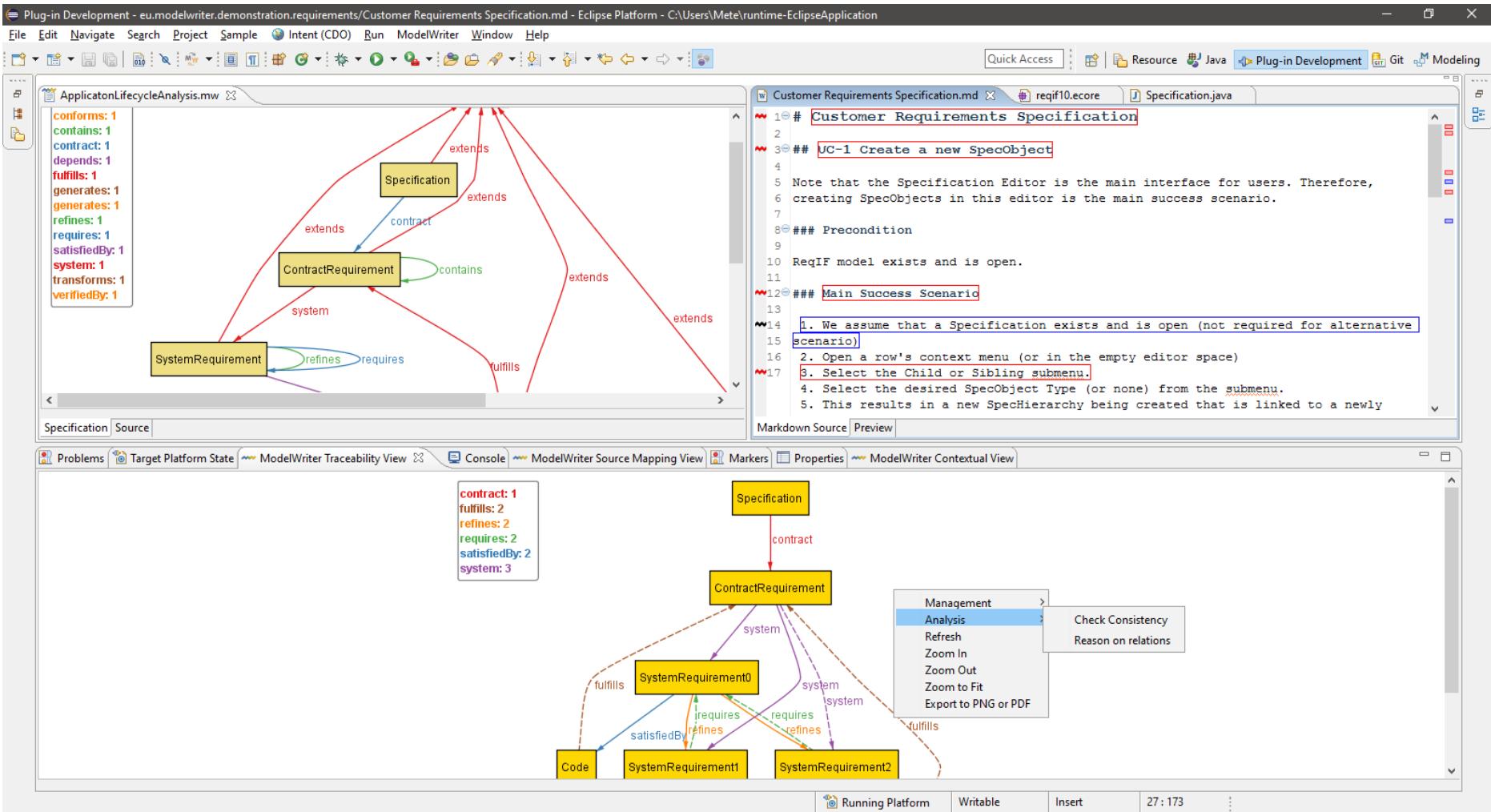
SystemRequirement1

SystemRequirement2

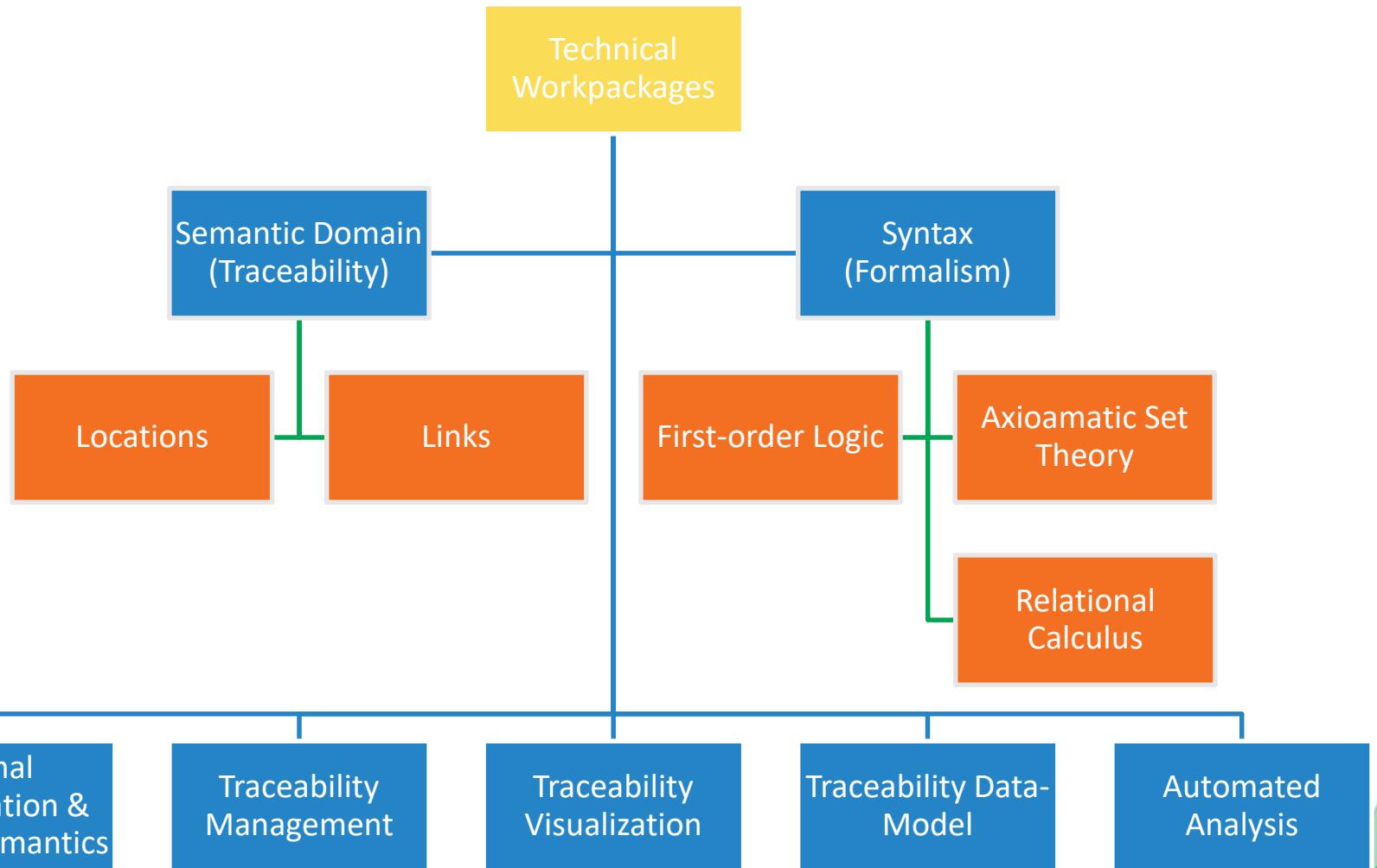
extends
contract
contains
extends
system
refines
requires
fulfills
contract
refines
requires
fulfills
satisfiedBy
refines
refines
refines
fulfills

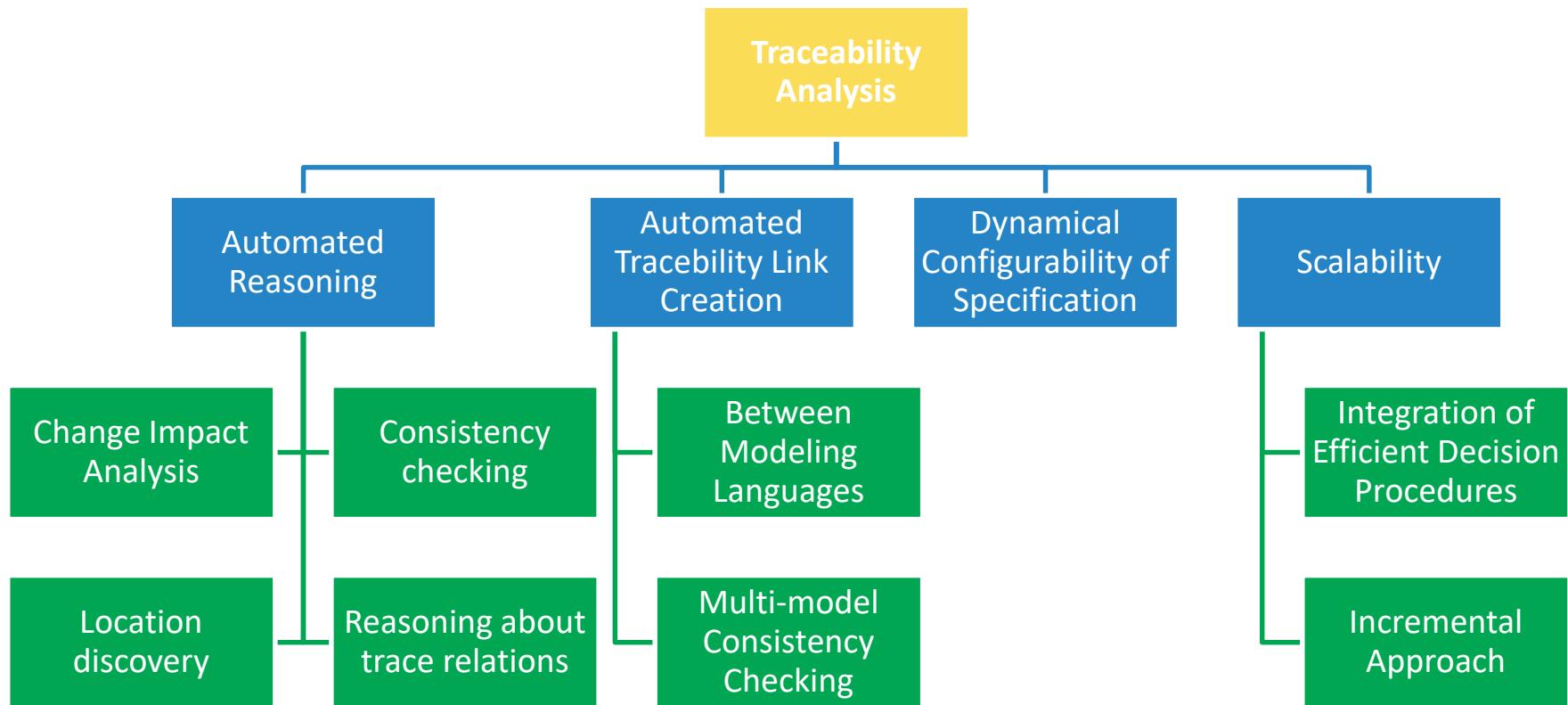
Management Analysis > Check Consistency Reason on relations

Running Platform Writable Insert 27:173

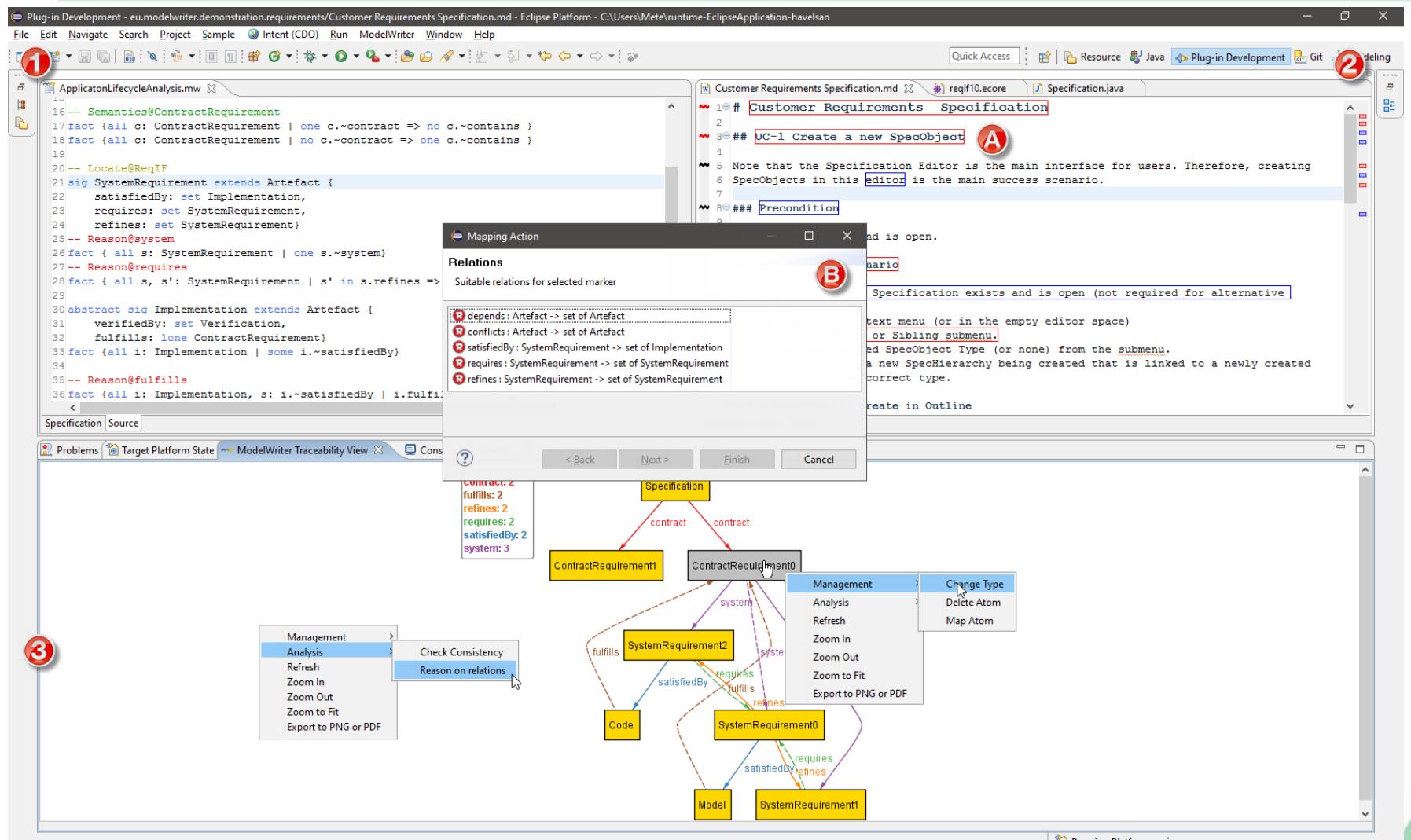


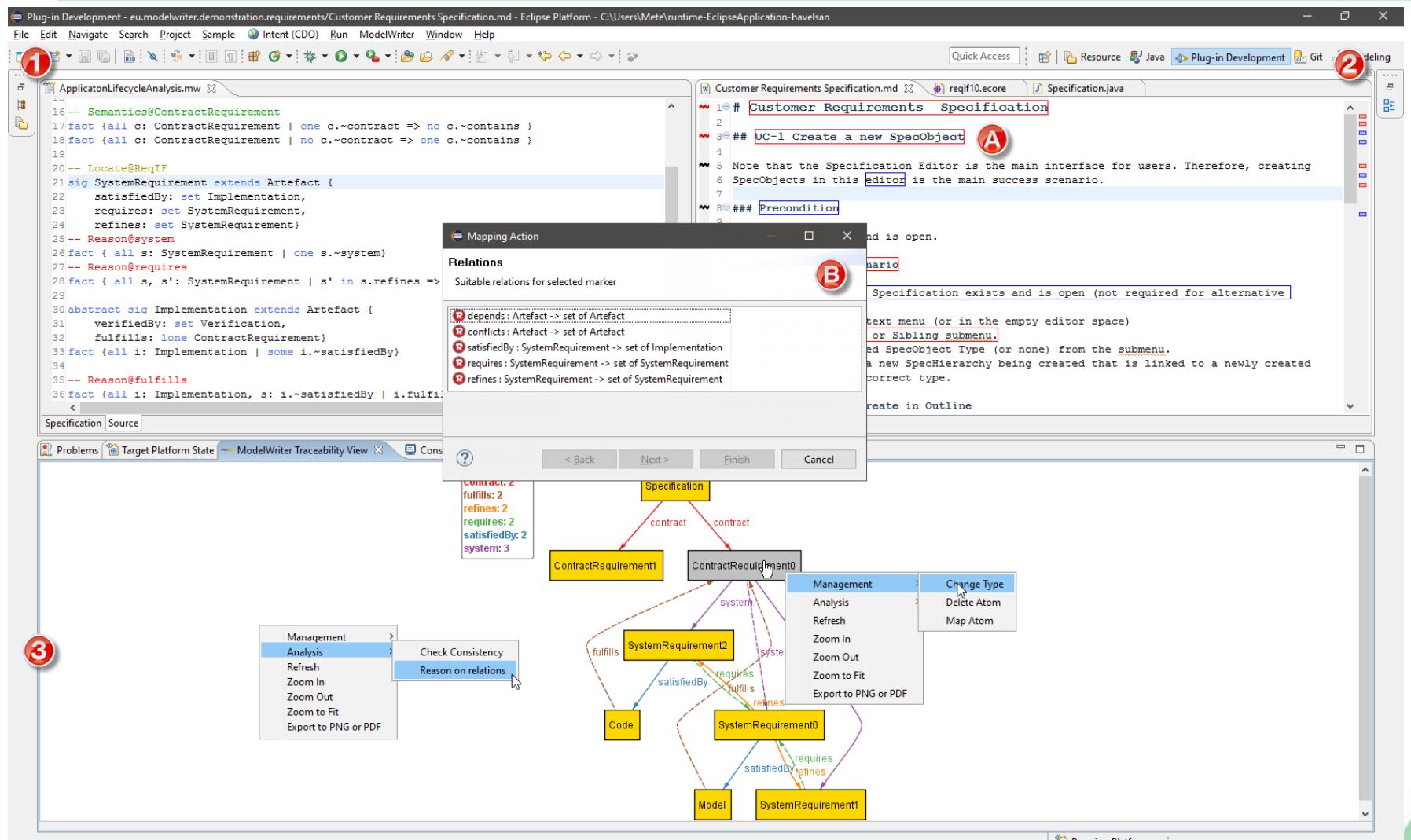
Overview of Technical Contributions @Tarski

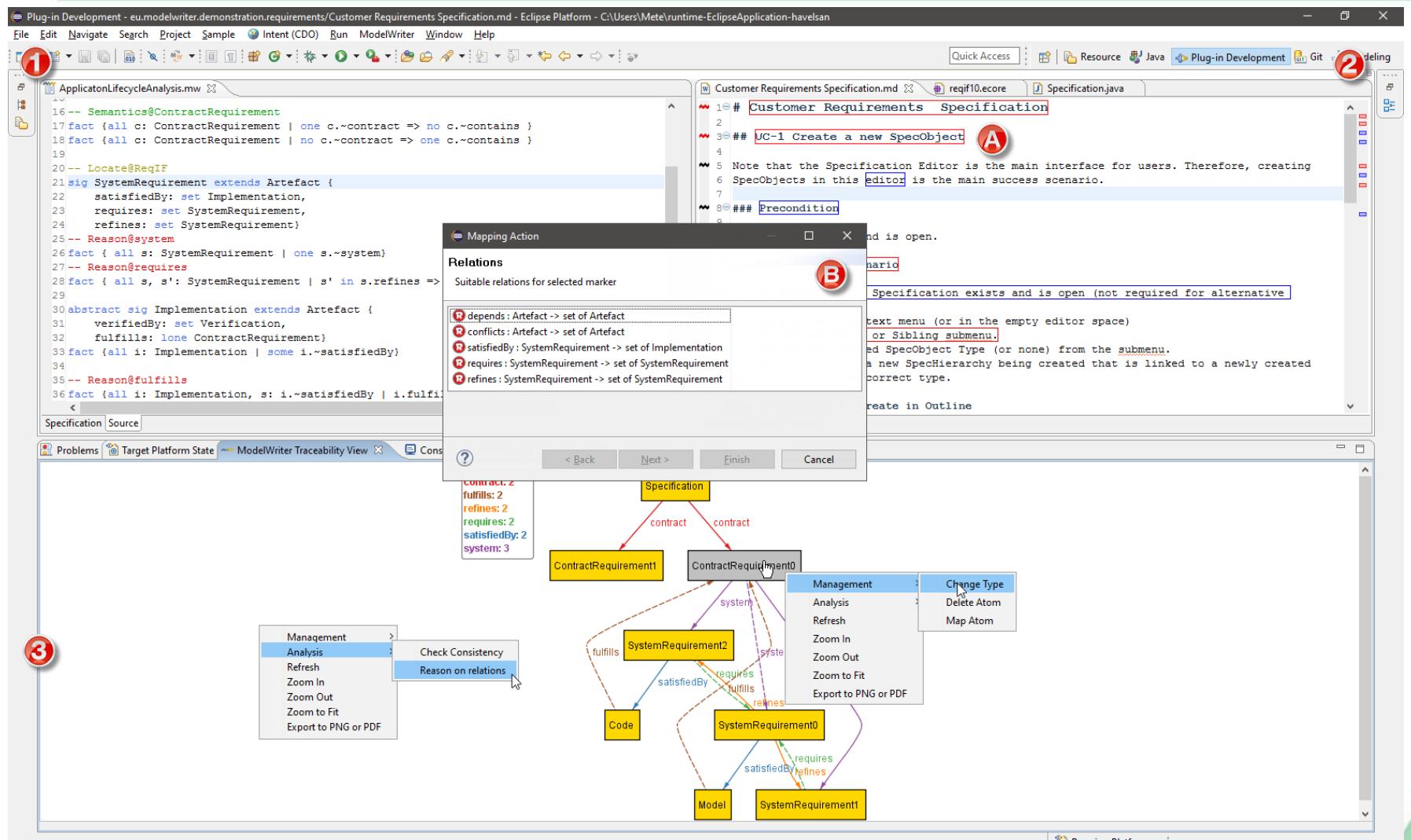




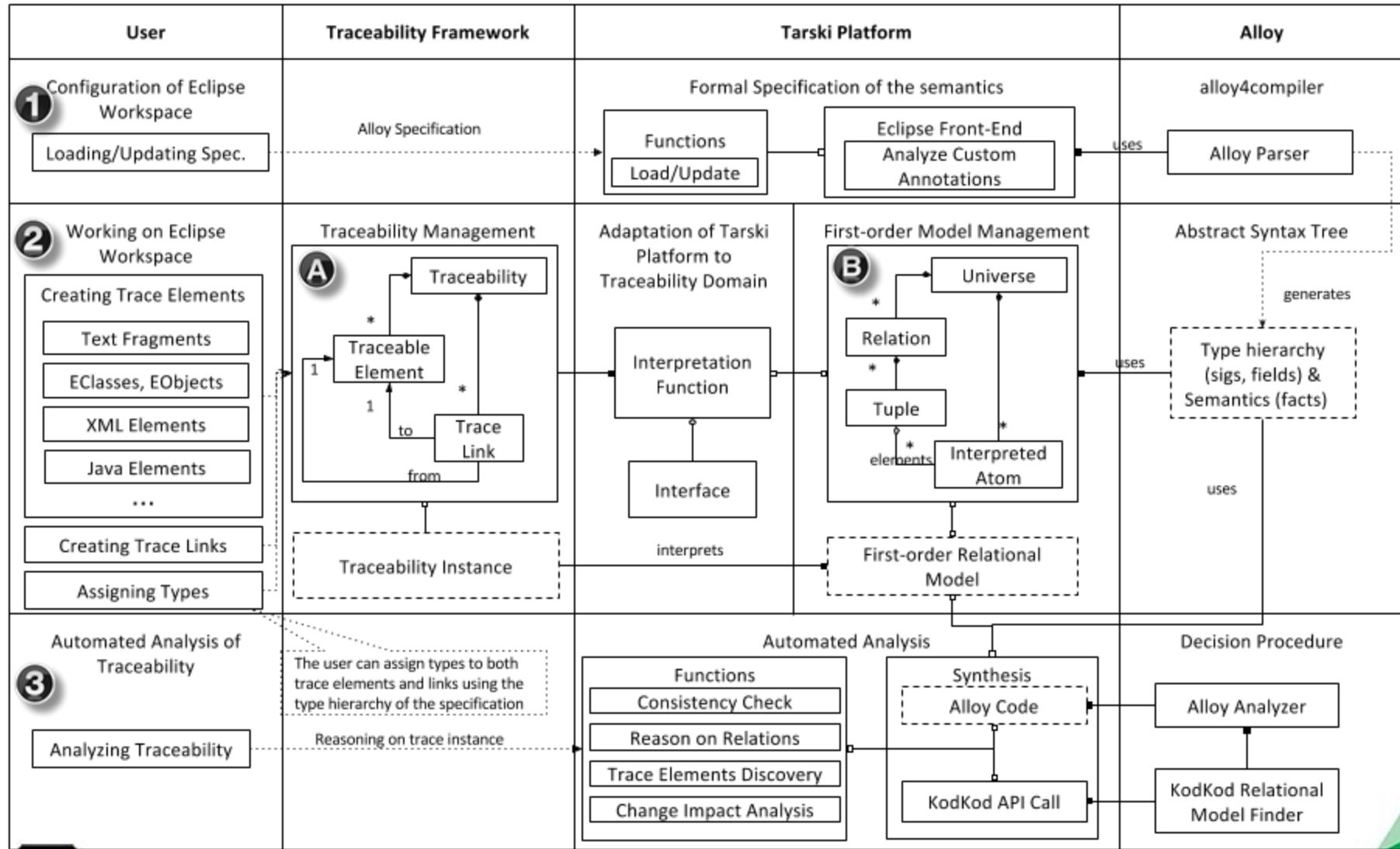
Tarski Approach

1  The screenshot shows the Eclipse Platform interface. On the left, there is a code editor for 'ApplicatonLifecycleAnalysis.mw' containing Tarski model code. On the right, there is a 'Customer Requirements Specification' document with numbered steps and annotations. A 'Mapping Action' dialog is open in the center, showing relations between artifacts. At the bottom, there is a traceability view diagram with nodes like 'Specification', 'ContractRequirement', 'SystemRequirement', 'Code', 'Model', and 'SystemRequirement'. Context menus are visible over the nodes.

2  This part of the screenshot focuses on the 'Customer Requirements Specification' document. Annotation A points to a note about creating a new SpecObject. Annotation B points to the 'Relations' section of the 'Mapping Action' dialog, which lists various relations like 'depends', 'conflicts', 'satisfiedBy', 'requires', and 'refines'.

3  This part of the screenshot shows the traceability view at the bottom. A context menu is open over a node, with options like 'Management', 'Analysis', 'Reason on relations', 'Check Consistency', and 'Change Type'. Another context menu is visible on the right side of the screen.

Tarski Approach



Types/Component Ontology derived from the specification

Plug-in Development - Eclipse Platform - C:\Users\Metu\runtime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw Customer Requirements Specification.md Specification.java reqif10.ecore

ApplicationLifecycleAnalysis.mw

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact;
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends
10 contract: some ContractRequirement
11 -- Locate@Text
12 -- Discover@ContractRequirement
13 sig ContractRequirement extends
14 system: set SystemRequirement
15 contains: set ContractRequirement
16
17 -- Semantics@ContractRequirement
18 fact {all c: ContractRequirement}
19 fact {all c: ContractRequirement}
20
21 -- Locate@ReqIF
22 sig SystemRequirement extends
23 satisfiedBy: set Implementation
24 requires: set SystemRequirement
25 refines: set SystemRequirement
26
27 -- Reason@system
28 fact {all s: SystemRequirement}
29
30 -- Reason@requires
31 fact {all s, s': SystemRequirement}
32
33 abstract sig Implementation {
34   verifiedBy: set Verification
35   fulfills: lone ContractRequirement
36 fact {all i: Implementation}
37
38 -- Reason@fulfills
39 fact {all i: Implementation, s: i.~satisfiedBy | i.fulfills = s.~system}
40
41 -- Tarski@FMP
```

Preferences

type filter text

- > Help
- > Install/Update
- > Java
- > Logic Diagrams
- > Maven
- > Model Editor
- > Model Validation
- > Mwe2
- > Mylyn
- > OCL
- > Oomph
- > Plug-in Development
- > ProR
- > Run/Debug
- > Sirius
- > SWTBot Preferences
- > Tarski
 - Sets and Relations
- > Team
- > Validation
- > WindowBuilder
- > XML
- > Xpand
- > Xtend
- > Xtext

Sets and Relations

Sets

- universe
 - Artefact (abs)
 - Specification
 - ContractRequirement
 - SystemRequirement
 - Implementation (abs)
 - Model
 - Code
 - Component
 - Verification (abs)
 - Simulation
 - Analysis
 - Test

Relations

- depends : Artefact -> set of Artefact
- conflicts : Artefact -> set of Artefact
- contract : Specification -> some of ContractRequirement
- system : ContractRequirement -> set of SystemRequirement
- contains : ContractRequirement -> set of ContractRequirement
- satisfiedBy : SystemRequirement -> set of Implementation
- requires : SystemRequirement -> set of SystemRequirement
- refines : SystemRequirement -> set of SystemRequirement
- verifiedBy : Implementation -> set of Verification
- fulfills : Implementation -> lone of ContractRequirement
- transforms : Model -> set of Model
- conforms : Model -> set of Model
- generates : Model -> set of Code, Component

Specification C:\Users\Metu\git\Demonstrations\eu.modelwriter.demonstration.requirements\ApplicationLifecycleAnalysis.mw

OK Cancel

Tarski Traceability View

Customer Requirements Specification.md

Specification.java

reqif10.ecore

contains: 1

Specification

contractRequirement0

Requirement0

ContractRequirement2

Requirement2

SystemRequirement1

Model

contract

system

contains

requires

refines

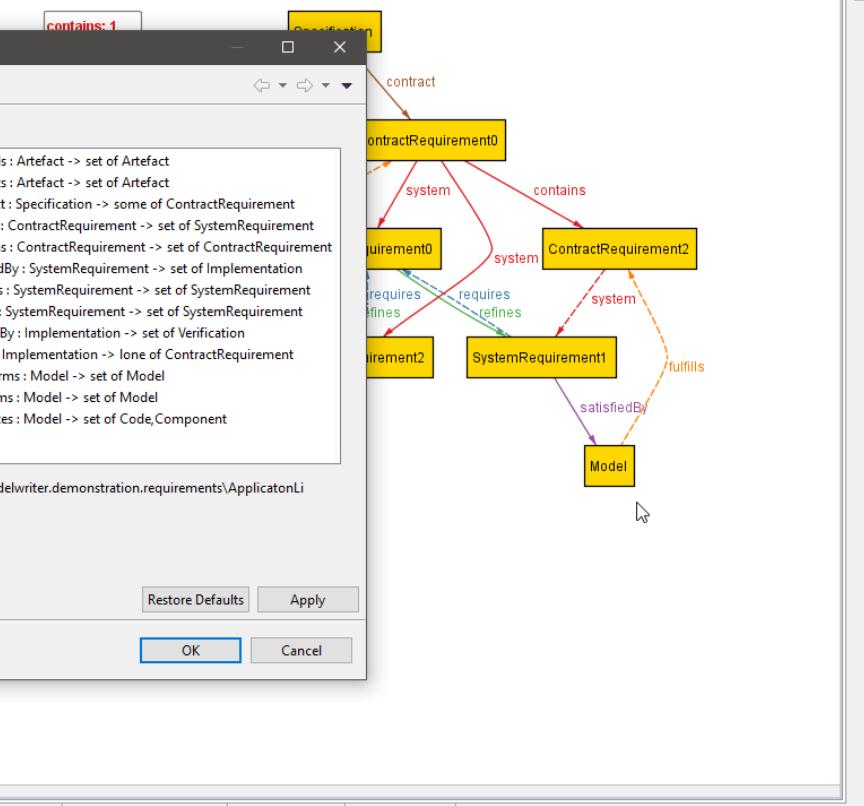
refines

system

fulfills

satisfiedBy

Running Platform



Assigning Unary Relations to a Traceable Elements



Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metu\runtime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact)
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10   contract: some ContractRequirement
11 -- Locate@Text
12 -- Discover@ContractRequirement expect 3
13 sig ContractRequirement extends Artefact {
14   system: set SystemRequirement,
15   contains: set ContractRequirement)
16
17 -- Semantics@ContractRequirement
18 fact {all c: ContractRequirement | one c.~contract => no c.
19 fact {all c: ContractRequirement | no c.~contract => one c.
<
```

Source Specification

Problems Console Markers Properties Tarsi Master View Tarsi Context

contains: 1
contract: 2
fulfills: 2
refines: 2
requires: 2
satisfiedBy: 2
system: 3

Customer Requirements Specification.md

```
1# Customer Requirements Specification
2
3## UC-1 Create a new SpecObject
4
```

Create a Trace Element with Type

universe

- Artefact (abs)
 - Specification
 - ContractRequirement
 - SystemRequirement
- Implementation (abs)
 - Model
 - Code
 - Component
- Verification (abs)
 - Simulation
 - Analysis
 - Test

Finish Cancel

ContractRequirement1

ContractRequirement0

SystemRequirement0

ContractRequirement2

Code

SystemRequirement2

SystemRequirement1

system

contains

fulfills

satisfiedBy

refines

requires

defines

Running Platform Writable Insert 9:6



Assigning Binary Relations to a Trace Link



Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metu\runtime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact)
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10   contract: some ContractRequirement)
11 -- Locate@Text
12 -- Discover@ContractRequirement expect 3
13 sig ContractRequirement extends Artefact {
14   system: set SystemRequirement,
15   contains: set ContractRequirement)
16
17 -- Semantics@ContractRequirement
18 fact {all c: ContractRequirement | one c.~contract => no c.~contains}
19 fact {all c: ContractRequirement | no c.~contract => one c.~contains}
<
Source Specification
```

Problems Console Markers Properties Tarsi Master View Tarsi Contracts

Create a trace relation

Relations

Suitable relations for selected trace element (SystemRequirement\$0)

- depends : Artefact -> set of Artefact
- conflicts : Artefact -> set of Artefact
- satisfiedBy : SystemRequirement -> set of Implementation
- requires : SystemRequirement -> set of SystemRequirement
- refines : SystemRequirement -> set of SystemRequirement

ContractRequirement1

ContractRequirement0

SystemRequirement0

ContractRequirement2

Code

SystemRequirement2

SystemRequirement1

contract

contract

contains

fulfills

system

requires

satisfiedBy

refines

refines

system

system

Running Platform Writable Insert 9:6



Selecting a range for a binary relation from an existing traceable elements

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metu\runtime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact)
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10   contract: some ContractRequirement
11   -- Locate@Text
12   -- Discover@ContractRequirement expect 3
13   sig ContractRequirement extends Artefact {
14     system: set SystemRequirement,
15     contains: set ContractRequirement)
16
17 -- Semantics@ContractRequirement
18 fact {all c: ContractRequirement | one c.~contract => ...
19 fact {all c: ContractRequirement | no c.~contract => ...
<
```

Source Specification

Problems Console Markers Properties Tarsi Master View Tars

Markers "Main Success Scenario"

Create a trace relation

Markers

Main Success Scenario

Show only files that contain Marker(s)

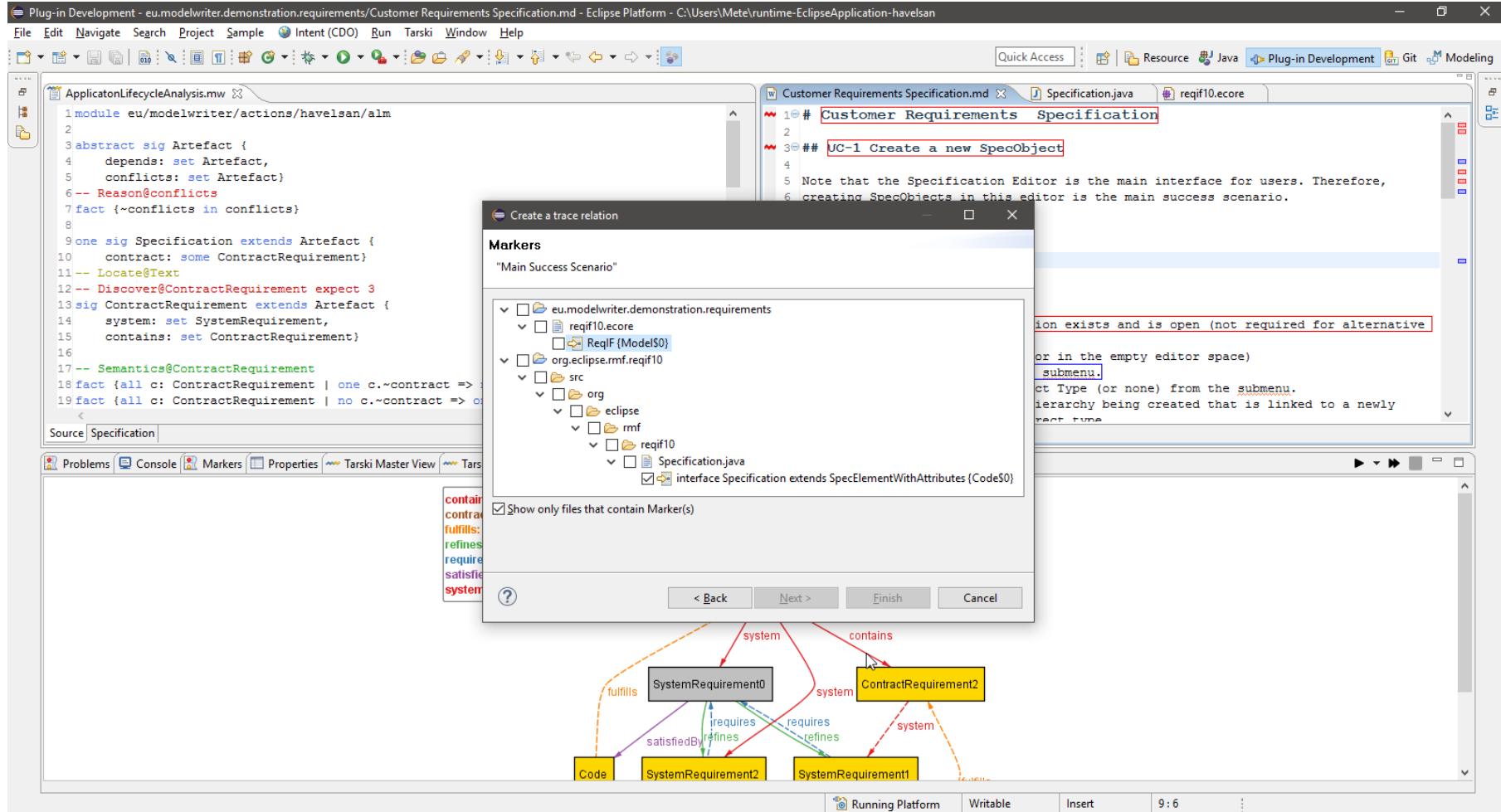
eu.modelwriter.demonstration.requirements
reqif10.ecore
ReqIF (ModelS0)
org.eclipse.emf.reqif10
src
org
eclipse
rmf
reqif10
Specification.java
interface Specification extends SpecElementWithAttributes (CodeS0)

< Back Next > Finish Cancel

SystemRequirement0
ContractRequirement2
Code
SystemRequirement2
SystemRequirement1

fulfills
refines
requires
satisfies
system
contains
fulfills
refines
requires
satisfies
system
system
system
system

Running Platform Writable Insert 9:6



The screenshot shows the Eclipse Platform interface with several open windows. The left window displays a Metamodel (ApplicationLifecycleAnalysis.mw) containing UML-like code. The right window shows a 'Customer Requirements Specification.md' file with some text and a note about the Specification Editor. A central dialog box titled 'Create a trace relation' lists 'Markers' under 'Main Success Scenario'. Below it is a diagram showing relationships between 'SystemRequirement0', 'ContractRequirement2', 'Code', 'SystemRequirement2', and 'SystemRequirement1'. Relationships include 'fulfills', 'refines', 'requires', 'satisfies', and 'system'.

Automated Analysis of Traceability

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metu\runtime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

```

1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact)
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10   contract: some ContractRequirement
11 -- Locate@Text
12 -- Discover@ContractRequirement expect 3
13 sig ContractRequirement extends Artefact {
14   system: set SystemRequirement,
15   contains: set ContractRequirement
16

```

Customer Requirements Specification.md

```

1# Customer Requirements Specification
2
3## UC-1 Create a new SpecObject
4
5 Note that the Specification Editor is the main interface for users. Therefore,
6 creating SpecObjects in this editor is the main success scenario.
7
8### Precondition
9
10 Reqls model exists and is open.
11
12### Main Success Scenario
13
14 1. We assume that a Specification exists and is open (not required for alternative
15 scenario)
16 2. Open a row's context menu (or in the empty editor space)
17 3. Select the Child or Sibling submenu.

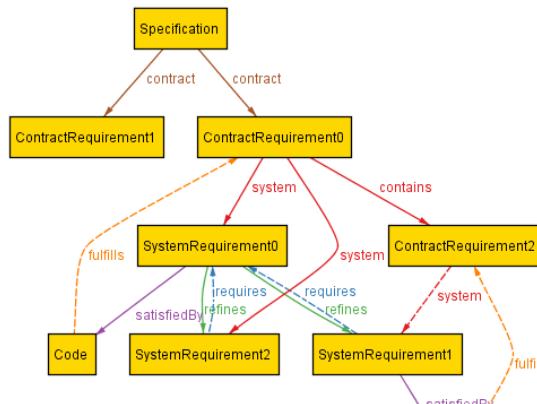
```

Specification

Source Specification

Problems Console Markers Properties Tarsi Master View Tarsi Contextual View Tarsi Traceability View

Management Analysis > Check Consistency Reason on Violations Discover Atoms Clear All Reasoned Tuples Refresh Zoom In Zoom Out Zoom to Fit Export to PNG or PDF



The diagram illustrates a traceability graph with the following entities and their relationships:

- Specification** (yellow box) has two outgoing solid arrows labeled "contract" pointing to **ContractRequirement1** and **ContractRequirement0**.
- ContractRequirement1** and **ContractRequirement0** both have solid arrows labeled "system" pointing to **SystemRequirement0**.
- SystemRequirement0** has a solid arrow labeled "contains" pointing to **ContractRequirement2**.
- ContractRequirement0** has a dashed arrow labeled "fulfills" pointing to **Code**.
- Code** has a dashed arrow labeled "satisfiedBy" pointing to **SystemRequirement2**.
- SystemRequirement2** has a dashed arrow labeled "refines" pointing to **SystemRequirement0**.
- SystemRequirement0** has a dashed arrow labeled "requires" pointing to **SystemRequirement1**.
- SystemRequirement1** has a dashed arrow labeled "refines" pointing to **SystemRequirement0**.
- SystemRequirement0** has a dashed arrow labeled "requires" pointing to **SystemRequirement2**.
- SystemRequirement2** has a dashed arrow labeled "satisfiedBy" pointing to **SystemRequirement1**.
- SystemRequirement1** has a dashed arrow labeled "fulfills" pointing to **Code**.

Dynamical Configuration & Model Management

Plug-in Development - eu.modelwriter.demonstration.requirements/ApplicationLifecycleAnalysis.mw - Eclipse Platform - C:\Users\Mete\runtime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4     depends: set Artefact,
5     conflicts: set Artefact}
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10    contract: some ContractRequirement
11    -- Locate@Text
12    -- Discover@ContractRequirement expect 3
13    sig ContractRequirement extends Artefact {
14        system: set SystemRequirement,
15        contains: set ContractRequirement}
16
17    -- Semantics@ContractRequirement
18    fact {all c: ContractRequirement | one c.~contract => no c.~contains }
19    fact {all c: ContractRequirement | no c.~contract => one c.~contains }
20
21    -- Locate@ReqIF
22    sig SystemRequirement extends Artefact {
23        satisfiedBy: set Implementation,
24        requires: set SystemRequirement,
25        refines: set SystemRequirement}
26
27    -- Reason@system
28    fact {all s: SystemRequirement | one s.~system}
29
30    -- Reason@requires
31    fact { all s, s': SystemRequirement | s' in s.refines => s in s'.requires }
32
33 abstract sig Implementation extends Artefact {
34     verifiedBy: set Verification,
35     fulfills: lone ContractRequirement}
36 fact {all i: Implementation | some i.~satisfiedBy}
37
38    -- Reason@fulfills
39 fact {all i: Implementation, s: i.~satisfiedBy | i.fulfills = s.~system }
40
41 -- Tarski@FMP
42
```

Specification

ContractRequirement1

ContractRequirement2

SystemRequirement0

Code

SystemRequirement2

SystemRequirement1

Model

Customer Requirements Specification.md

Specification.java

reqif10.ecore

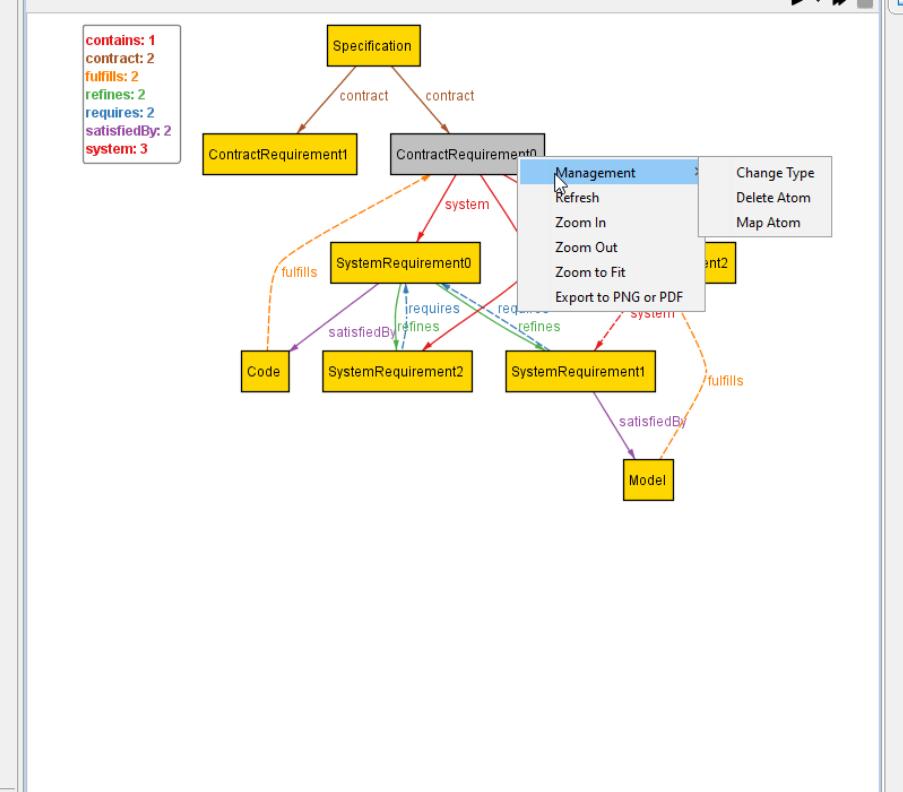
Management

- Refresh
- Zoom In
- Zoom Out
- Zoom to Fit
- Export to PNG or PDF

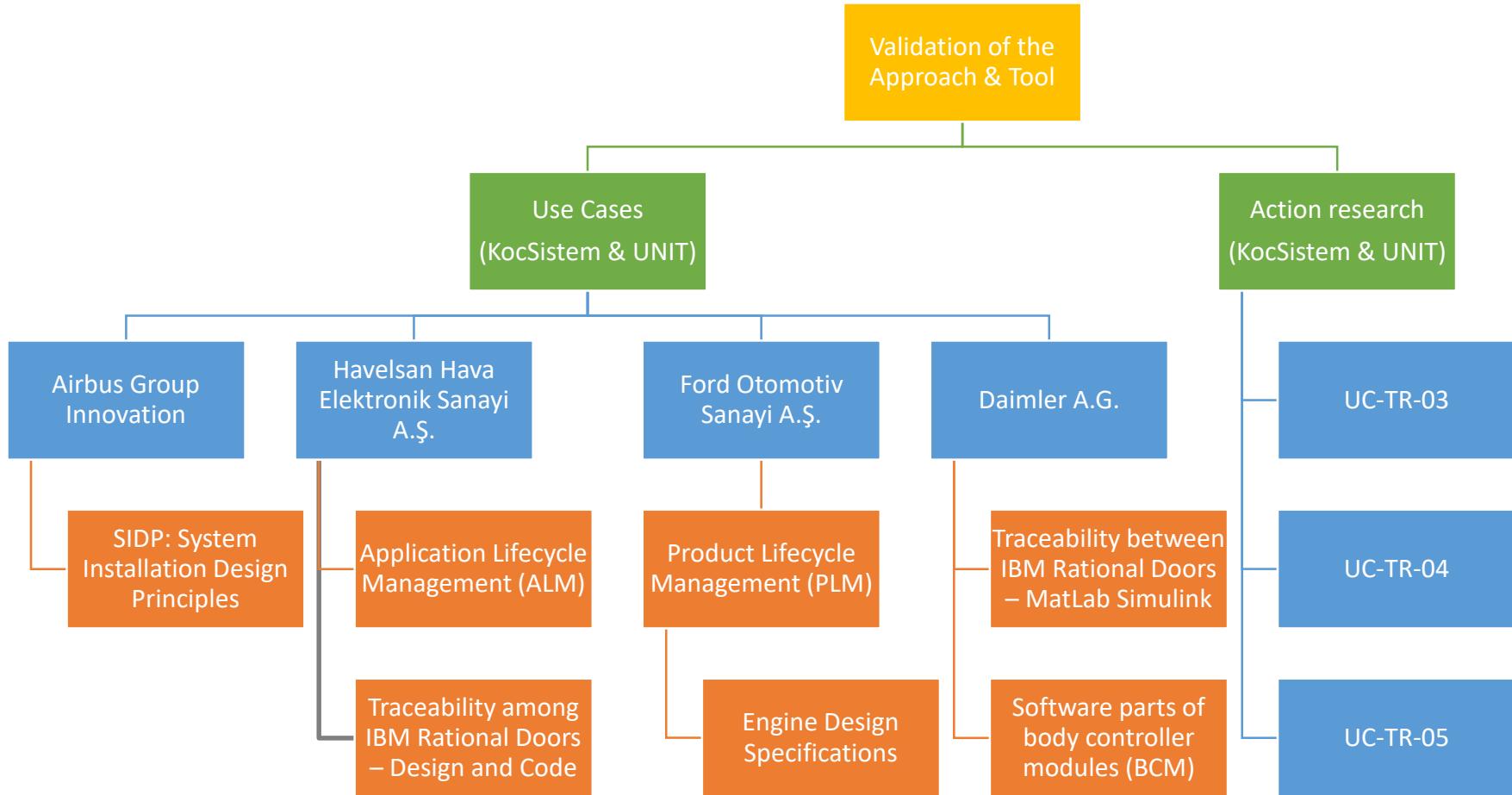
Change Type

Delete Atom

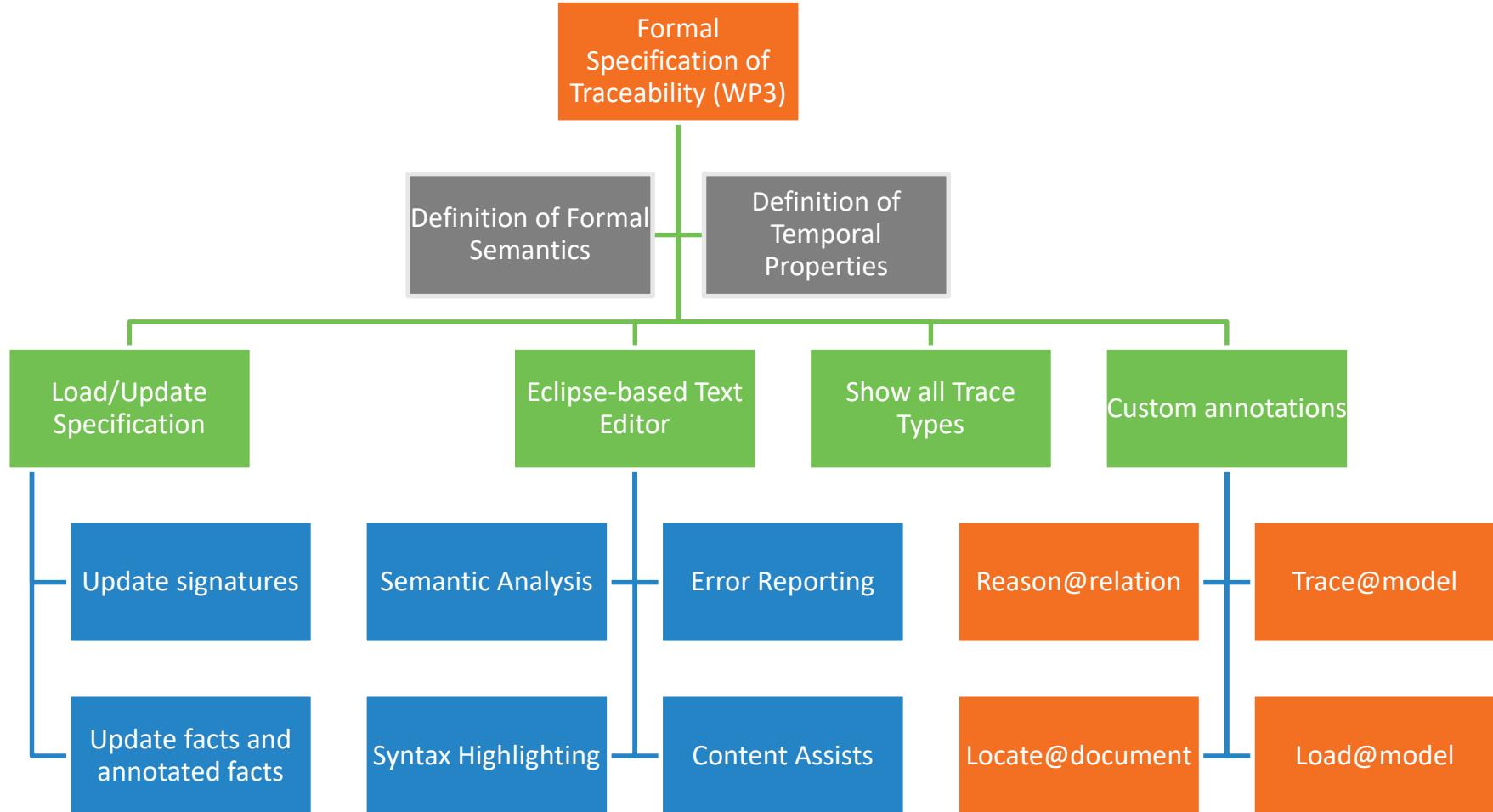
Map Atom



Validation of the Approach and Tool



Formal Specification of Traceability (WP3) [UNIT]



Demonstration Textual Editor in Action



Plug-in Development - eu.modelwriter.demonstration.requirements/ApplicatonLifecycleAnalysis.mw - Eclipse Platform - C:\Users\Metu\runtime-EclipseApplication

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

Load Specification
Switch marker visibility
Preferences
Project Management >

Problems Target Platform State ModelWriter Traceability View

Specification Mapping View Markers Properties ModelWriter Contextual View *ApplicatonLifecycleAnalysis.mw

1 module eu.modelwriter/actions/havelsan/alm

2

3 abstract sig Artefact {

4 depends: set Artefact,

5 conflicts: set Artefact)

6 -- Reason@conflicts

7 fact {~conflicts in conflicts}

8

9 one sig Specification extends Artefact {

10 contract: some ContractRequirement)

11 -- Locate@Text

12 sig ContractRequirement extends Artefact {

13 system: set SystemRequirement,

14 contains: set ContractRequirement)

15 fact {all c: ContractRequirements | one c.~contract => no c.~contains}

16

17 fact {all c: ContractRequirement | no c.~contract => one c.~contains}

18

19 -- Locate@ReqIF

20 sig SystemRequirement extends Artefact {

21 satisfiedBy: set Implementation,

22 requires: set SystemRequirement,

23 refines: set SystemRequirement}

24

25 -- Reason@requires

26 fact { all s, s': SystemRequirement | s' in s.refines => s in s'.requires}

27

28 -- Reason@

29 fact { all i

30

31 abstract sig

32 verify

33 fulfill

34 fact {all i

35 contains

36 satisfiedBy

37 fact {all i

38 requires

39 refines

40 verifiedBy

41 fulfills

42 transforms

43 conforms

44 <

45 depends

46 conflicts

47 contract

48 fulfill

49 contains

50 satisfiedBy

51 requires

52 refines

53 verifiedBy

54 fulfills

55 transforms

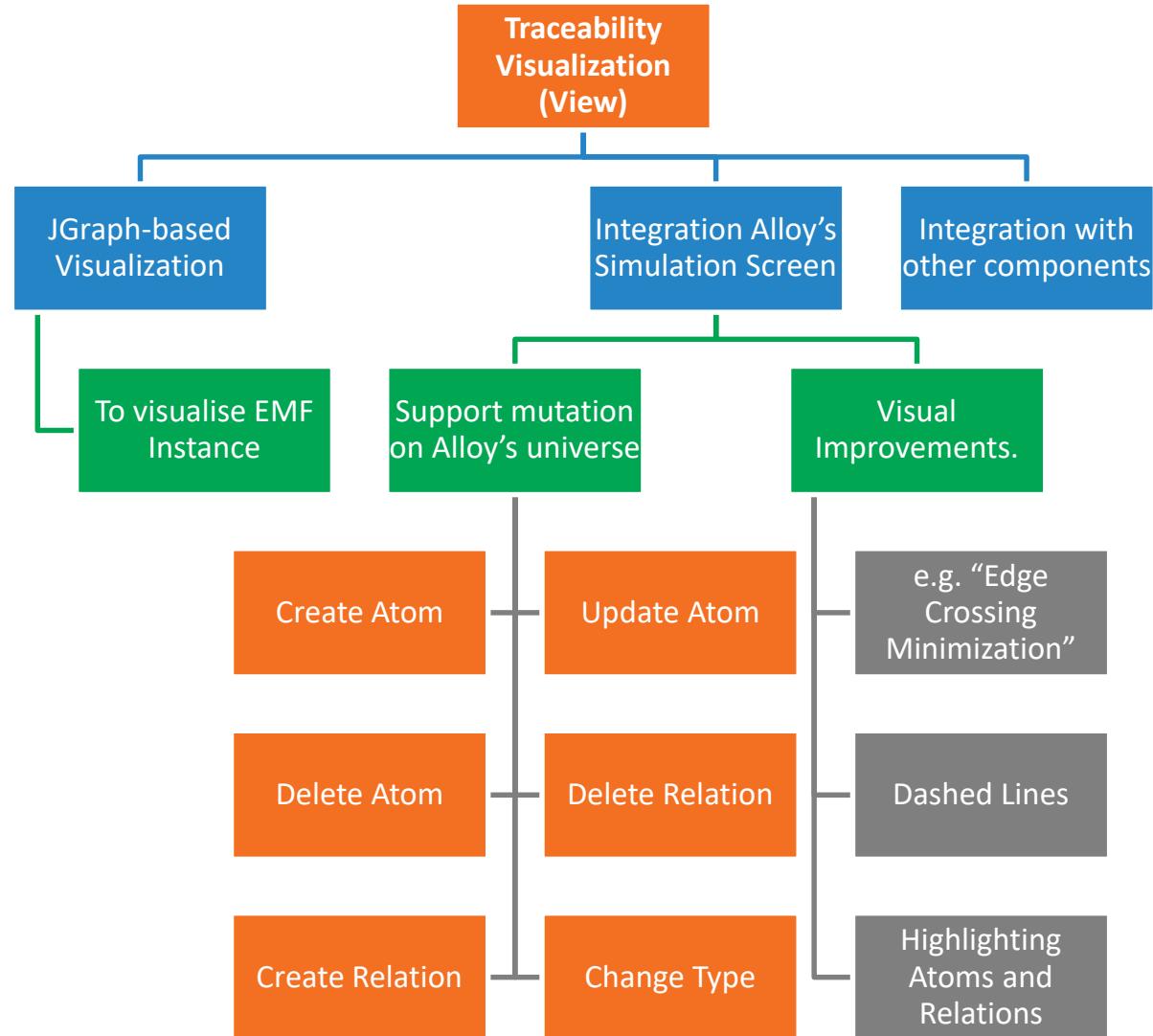
56 conforms

57 .fills = s.~system }

Specification Source

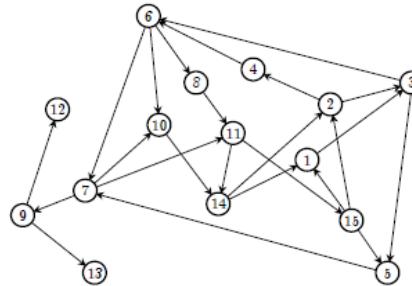
Running Platform

Traceability Visualization/View (WP3) [KoçSistem]

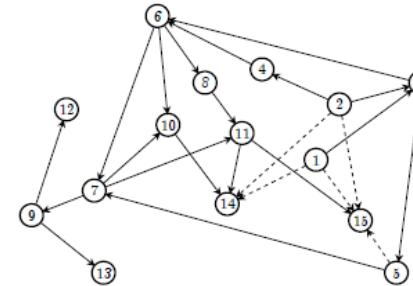


Layered/Hierarchical Graph Drawing

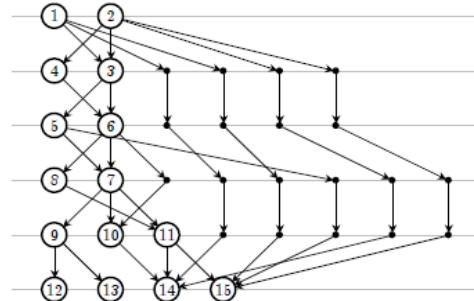
Sugiyama Framework



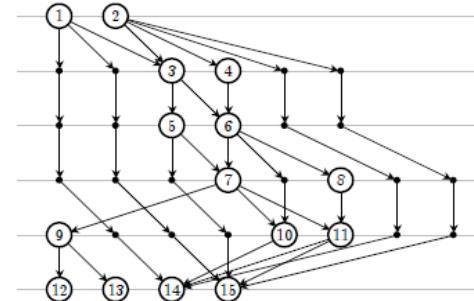
(a) Input digraph, G .



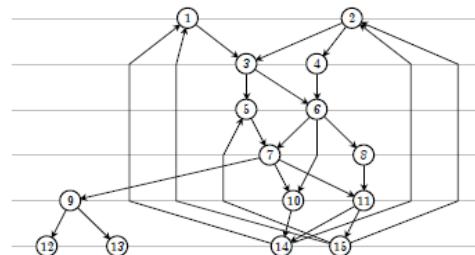
(b) Cycles removed.



(c) After leveling.



(d) Edge crossings minimized.



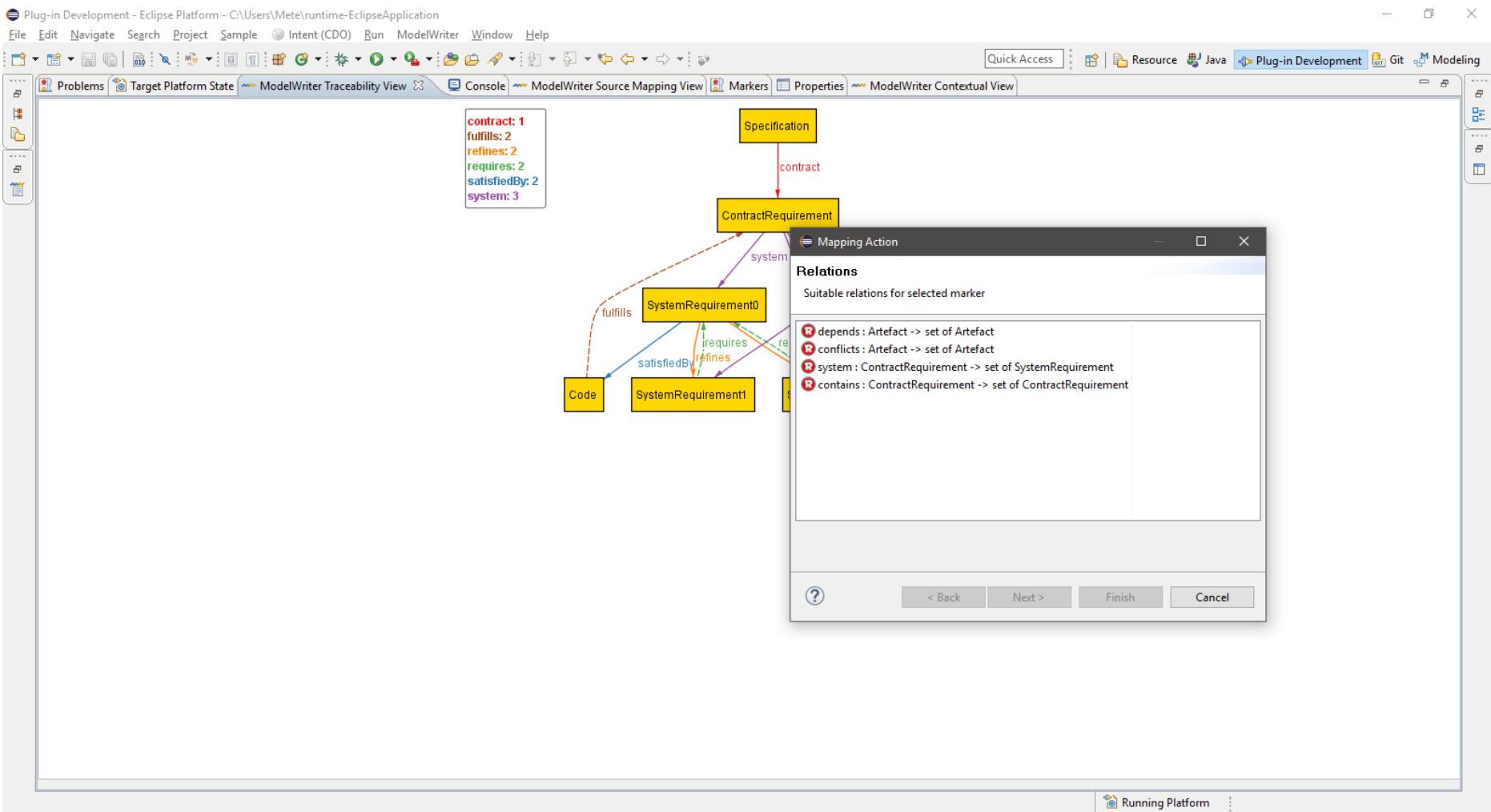
(e) Edges straightened.

Demonstration Visualization in Action

Plug-in Development - Eclipse Platform - C:\Users\mete\runtime-EclipseApplication

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

Problems Target Platform State ModelWriter Traceability View Console ModelWriter Source Mapping View Markers Properties ModelWriter Contextual View



The diagram illustrates a traceability relationship between several artifacts:

- Specification** (yellow box) has a red arrow labeled "contract" pointing to **ContractRequirement**.
- ContractRequirement** (yellow box) has a blue arrow labeled "system" pointing to **SystemRequirement0**.
- SystemRequirement0** (yellow box) has a dashed blue arrow labeled "fulfills" pointing to **Code**.
- SystemRequirement0** (yellow box) has a dashed green arrow labeled "satisfiedBy" pointing to **SystemRequirement1**.
- SystemRequirement1** (yellow box) has a dashed green arrow labeled "refines" pointing to **SystemRequirement0**.
- SystemRequirement1** (yellow box) has a dashed orange arrow labeled "requires" pointing to **Code**.

A tooltip on the left side of the diagram lists the following counts for each relationship type:

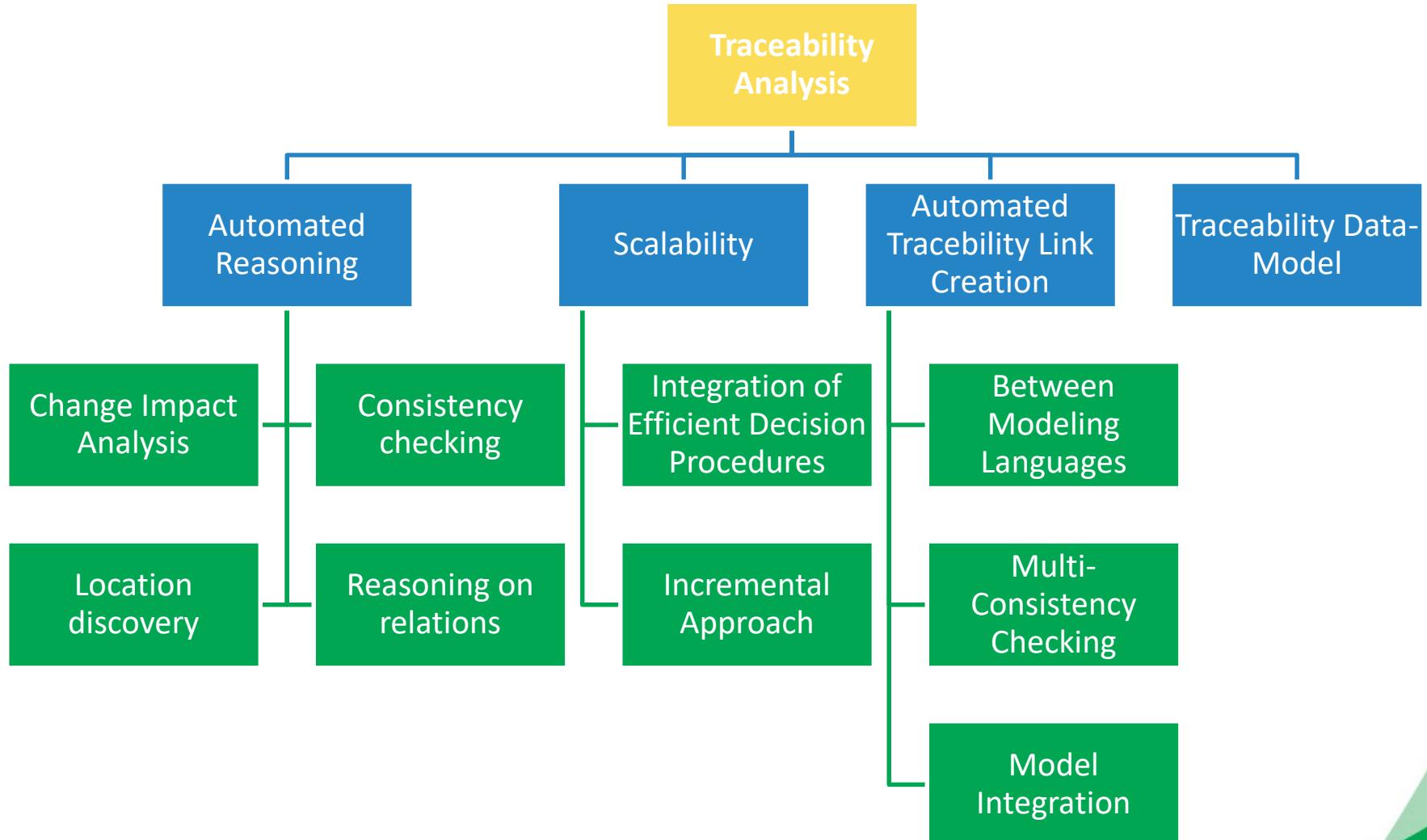
- contract: 1
- fulfills: 2
- refines: 2
- requires: 2
- satisfiedBy: 2
- system: 3

A modal dialog titled "Mapping Action" is open, showing a list of suitable relations for the selected marker:

- depends : Artefact -> set of Artefact
- conflicts : Artefact -> set of Artefact
- system : ContractRequirement -> set of SystemRequirement
- contains : ContractRequirement -> set of ContractRequirement

Buttons at the bottom of the dialog include: ? (Help), < Back, Next >, Finish, and Cancel.

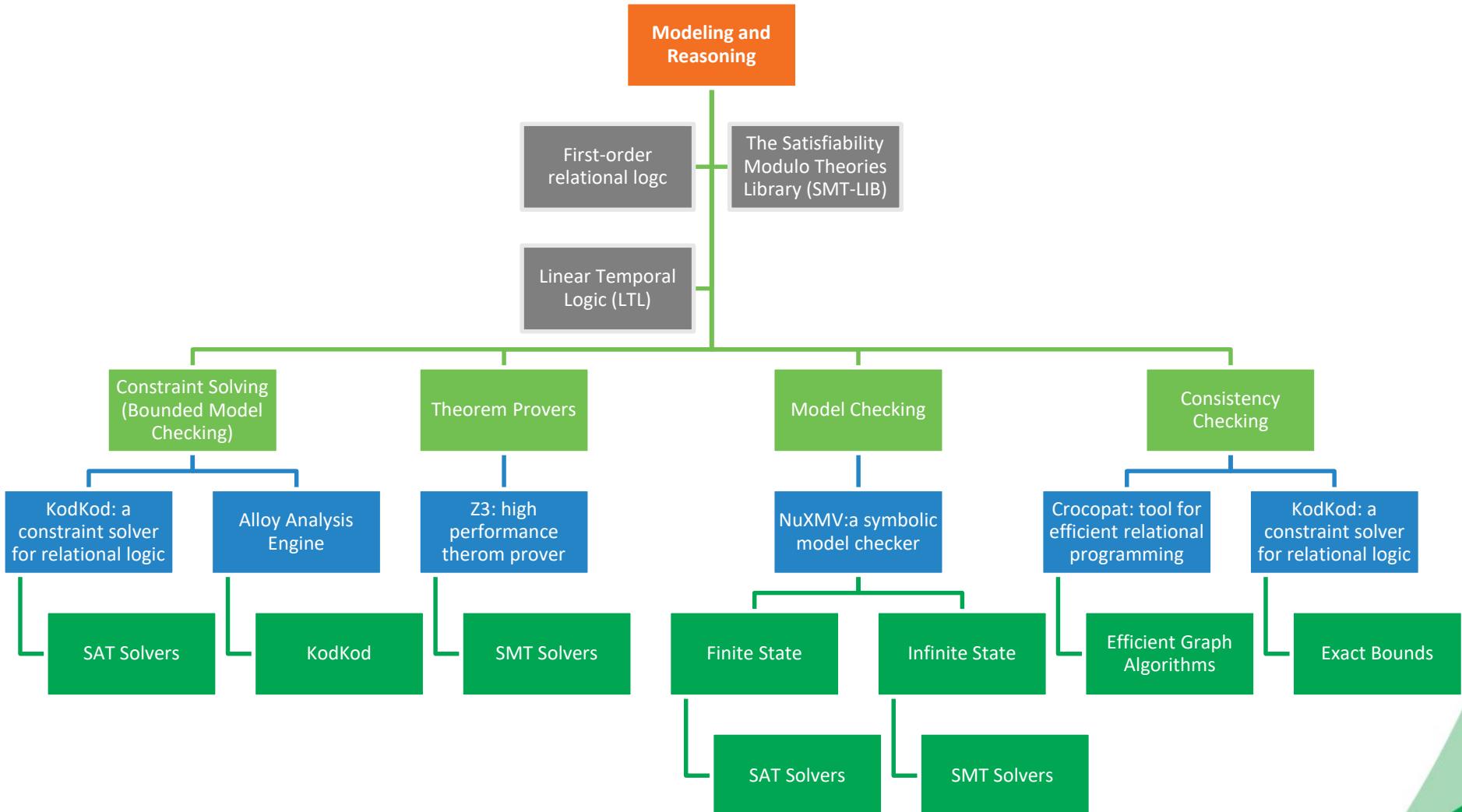
Traceability Analysis (WP4 & WP3) [UNIT]



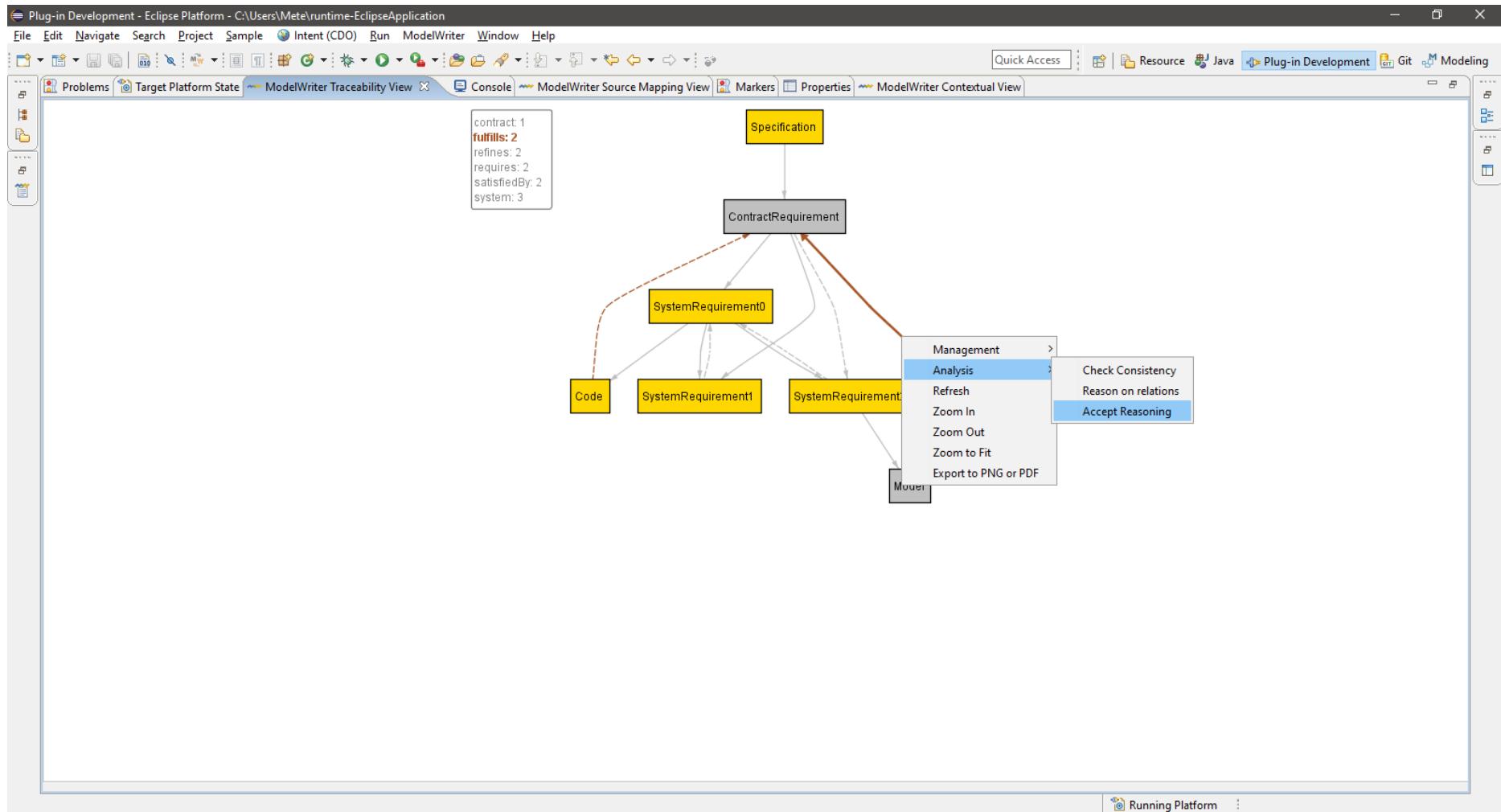
Modeling and Reasoning Approaches (WP3) [UNIT]



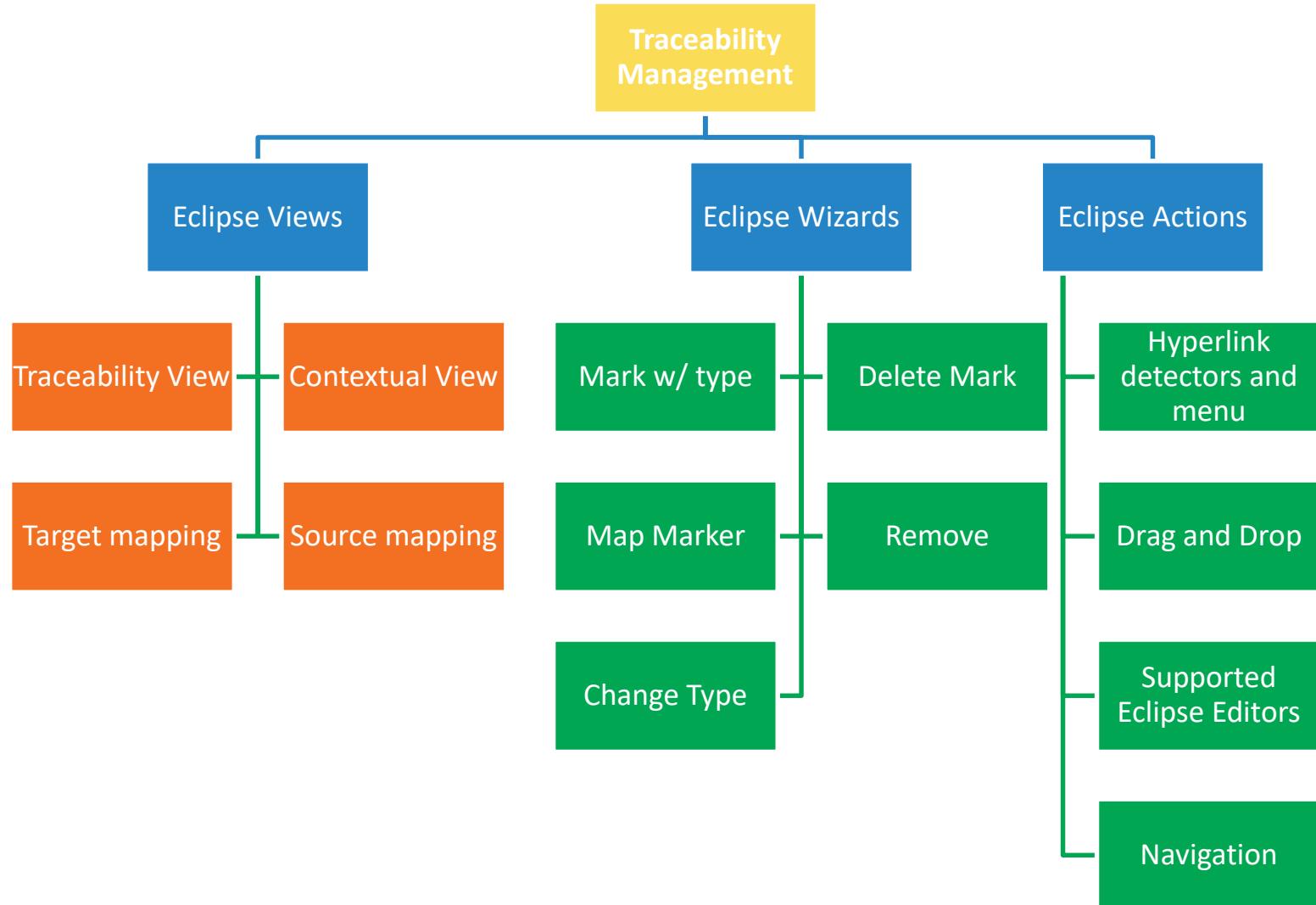
ITEA3



Demonstration Traceability Analysis in Action



Traceability Management (WP6) [KoçSistem]



Demonstration

Traceability Management in Action

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Mete\Runtime\EclipseApplication

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

ApplicationLifecycleAnalysis.mw

```

    graph TD
        Artfact[Artefact] -- conflicts --> SystemRequirement1[SystemRequirement1]
        Artfact -- depends --> SystemRequirement2[SystemRequirement2]
        SystemRequirement1 -- extends --> Specification[Specification]
        SystemRequirement2 -- extends --> ContractRequirement[ContractRequirement]
        Specification -- contract --> ContractRequirement
        SystemRequirement1 -- system --> ContractRequirement
        ContractRequirement -- contains --> SystemRequirement1
        SystemRequirement1 -- refines --> SystemRequirement2
        SystemRequirement2 -- refines --> SystemRequirement1
        SystemRequirement2 -- fulfills --> Code[Code]
    
```

Extends: 11
Conflicts: 1
Conforms: 1
Contains: 1
Contract: 1
Depends: 1
Fulfils: 1
Generates: 1
Generates: 1
Refines: 1
Requires: 1
SatisfiedBy: 1
System: 1
Transforms: 1
VerifiedBy: 1

Customer Requirements Specification.md

reqif0.ecore

Specification.java

Quick Access

Resource Java Plug-in Development Git Modeling

Contextual Menu (right-clicked on Specification)

- Undo Typing** Ctrl+Z
- Revert File**
- Save**
- Open With**
- Show In**
- Preview at "UC-1 Create a new SpecObject"**
- Cut** Ctrl+X
- Copy** Ctrl+C
- Paste** Ctrl+V
- Quick Fix** Ctrl+1
- Shift Right**
- Shift Left**
- Add to Snippets...**
- Run As**
- Debug As**
- Validate**
- Github**
- ModelWriter**
 - Team**
 - Compare With**
 - Replace With**
 - WikiText**
 - Markup Language**
 - Preferences...**
 - Remove from Context** Ctrl+Alt+Shift+Down

Specification | Source

Problems Target Platform State ModelWriter Traceability View Console ModelWriter Source Mapping View Markers Properties ModelWriter Contextual

Specification Contextual Menu (right-clicked on Specification)

- contract:** 1
- fulfills:** 2
- refines:** 2
- requires:** 2
- satisfiedBy:** 2
- system:** 3

Code Contextual Menu (right-clicked on Code)

- fulfils**

SystemRequirement Contextual Menu (right-clicked on SystemRequirement1)

- requires**
- satisfiedBy**
- refines**

SystemRequirement Contextual Menu (right-clicked on SystemRequirement2)

- refines**
- fulfills**

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

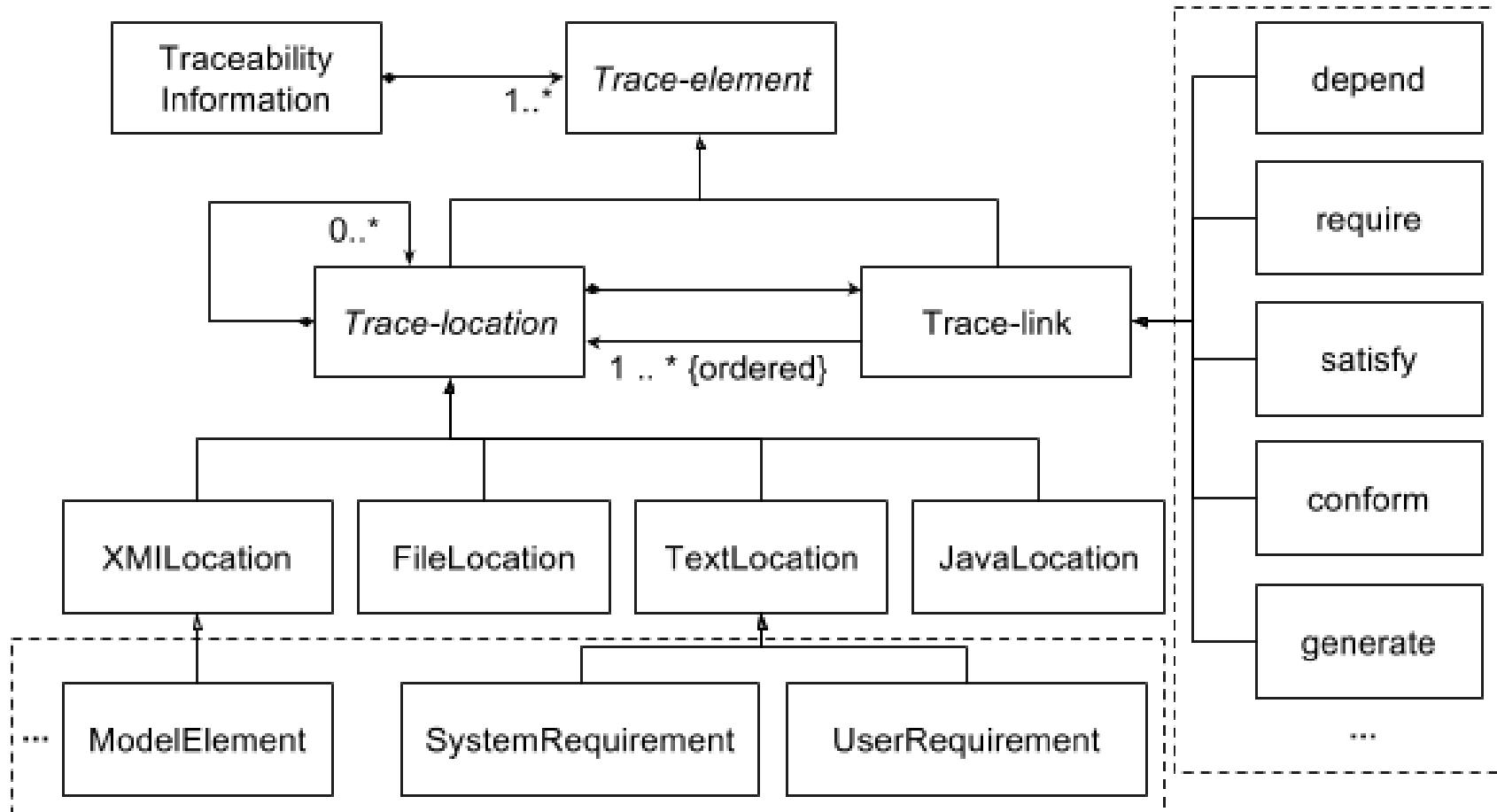
WP4 – Knowledge base Design and Implementation

*Prof. Dr. Geylani Kardas
(KocSistem)*

- Design and implement the ModelWriter's:
 - federated Knowledge Base itself, hosting multiple formalisms.
 - bi-directional text-model synchronization mechanism.
 - required API
 - 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.
 - 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).

ModelWriter Core Model Approach

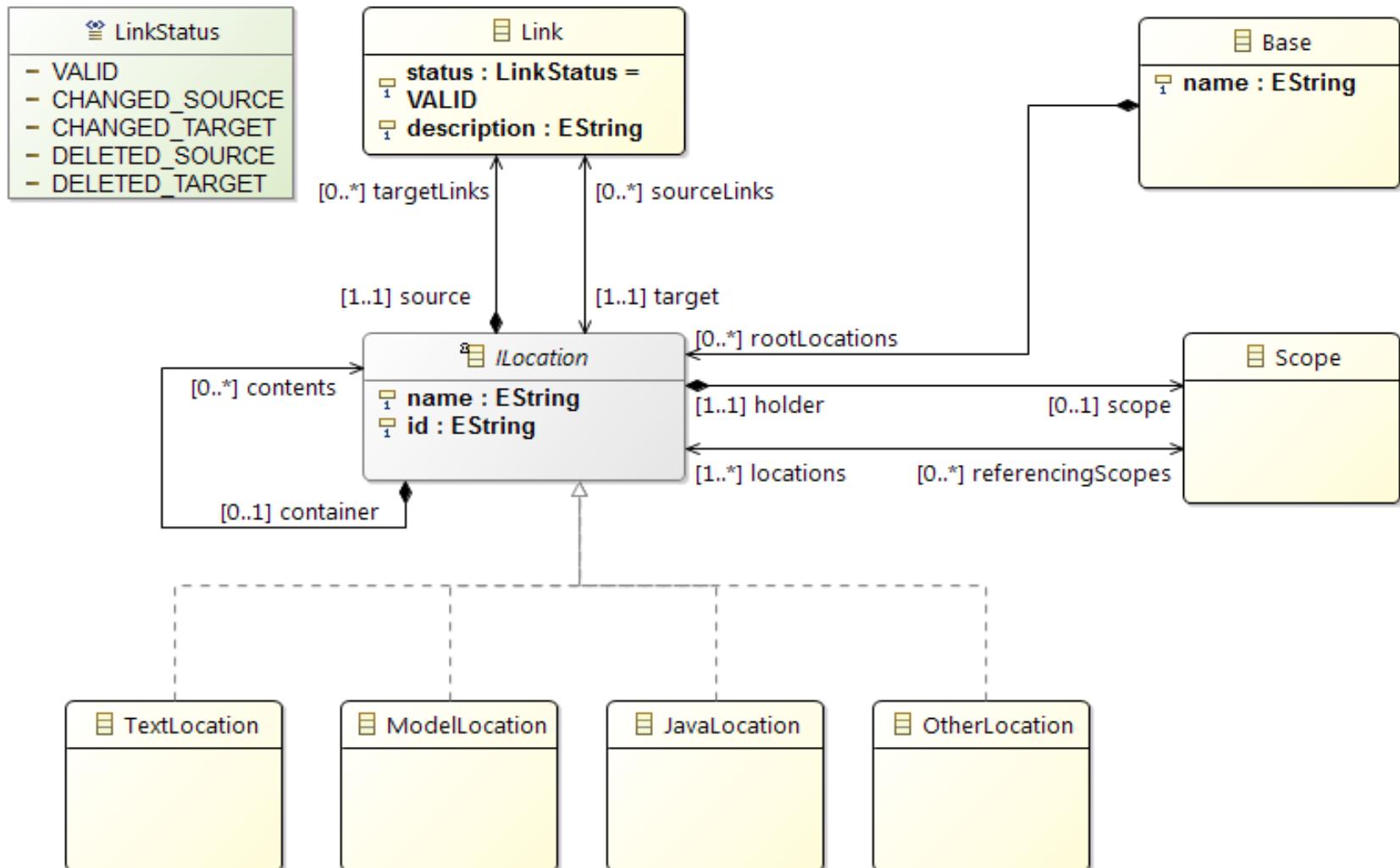


ModelWriter Core Model

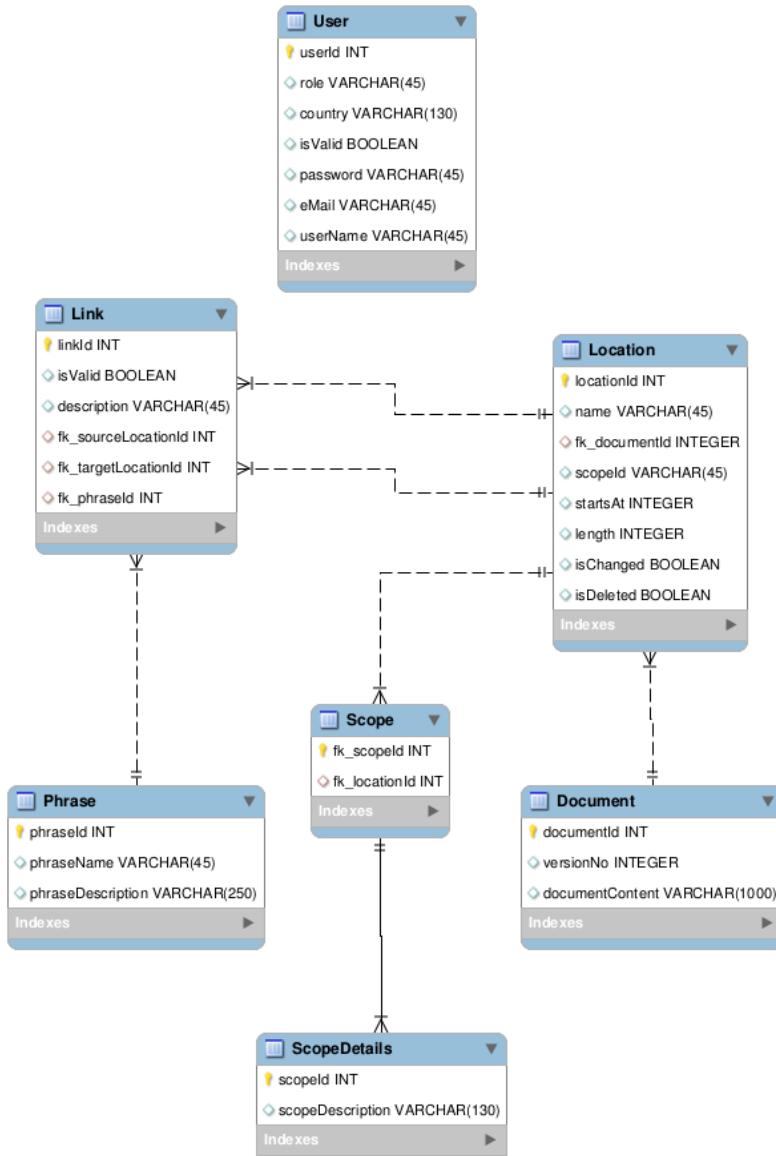
State-of-the-art

	SotA Tools & Approaches	Consideration of Different Artefacts/ Heterogeneity of artefacts (internal or external models)	Traceability Management		
			Approach (Management of Traceability)	Definition of Formal Semantic for trace-links	Dynamic Configuration of Semantics of trace-links
Industrial Tools, Methods & Standards	SysML ¹	UML Elements	UML Profiling mechanism	-	-
	ReqIF ²	Textual Requirements	Definition of XML Schema and extending its Data-Model	-	-
	IBM Doors ³	Arbitrary between model elements	Creation of Relation Types	Transitivity of relations	-
	ModelWriter	Arbitrary between any model element or textual requirements	Basic Traceability Model extended with a Formal Specification	FORL (First-order Relational Logic)	+

ModelWriter Core Model Implemented by OBEO



Knowledge-base Design

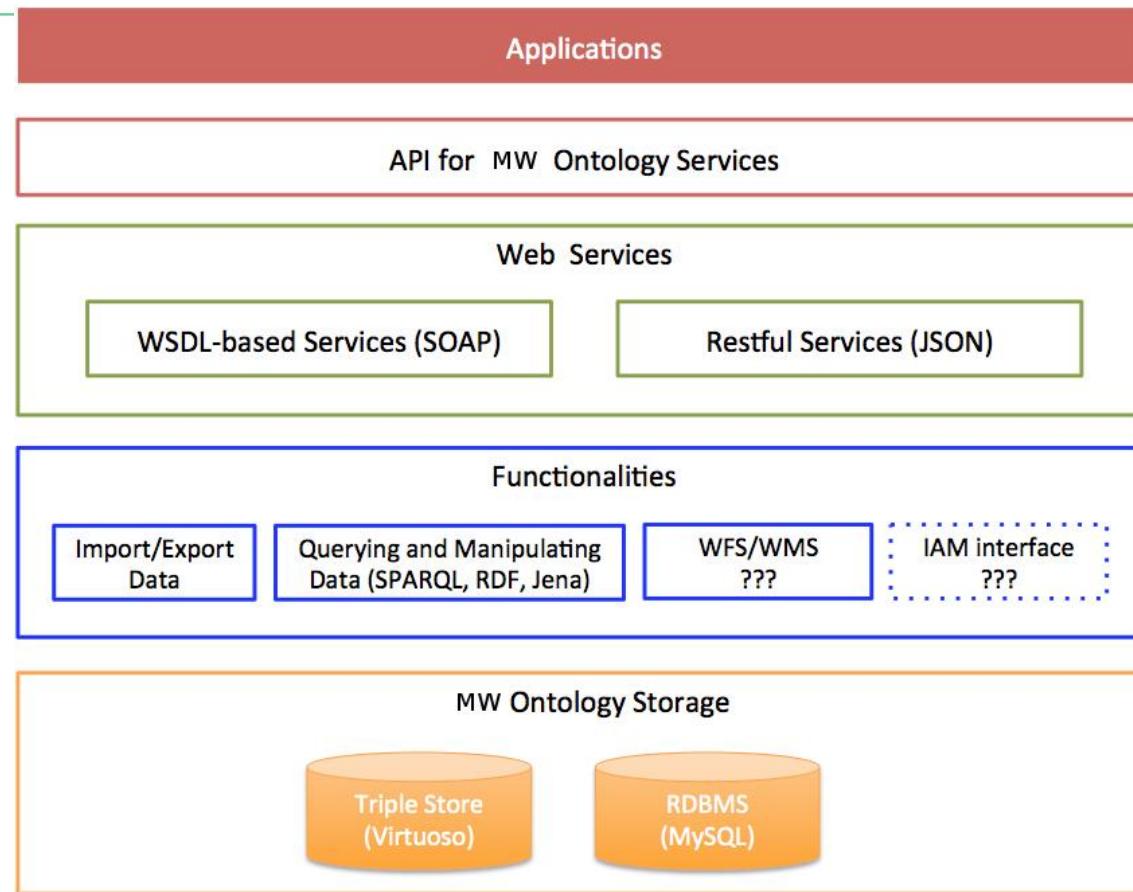


The class diagram of the knowledge base data structure representation.

MANTIS has prepared D4.1.1 Knowledge Base Design Document.

This document still is open to edit and to add contributions.

Knowledge-base Implementation Ontology Infrastructure



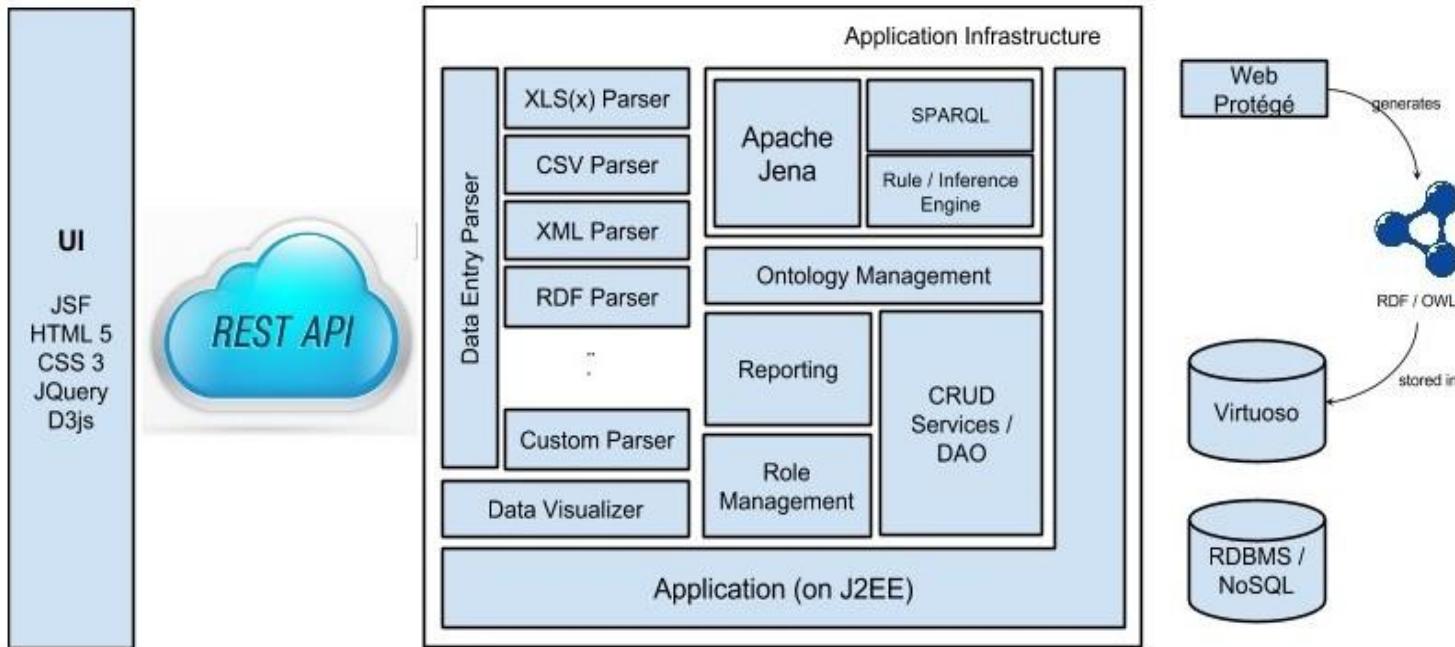
WMS/WFS services: standardized by Open Geospatial Consortium to use geographical attributes.
IAM interface for identification and access management. Current platform uses OAuth2 authentication system (<http://oauth.net/2/>)



Web Feature Services (WFS)/Web Map Services (WMS) may be partially implemented.
Identity and access management (IAM) services may not be implemented at all.

Knowledge-base Implementation

Ontology Infrastructure



The requests that arrives via web services are stored as triple in the Virtuoso Ontology Database and can be queried.

All processes that are based on ontology are done in Ontology Management Module (OMM). OMM is set up on parent Jena framework and inference engine can be utilized efficiently with SPARQL queries.

Also by using Web Protégé ontology can be generated in RDF/OWL formats and that ontology can be stored in Virtuoso.

Ontology Issues and Services

Ontology Issues

CRUD operations on ontologies as RESTFUL services

Using a sample design document, Mantis designed a document ontology

Ontology is hosted on Mantis servers

Manual RDF export

Automatically RDF export (working on)

Ontology Services

insertTriple: This method inserts a new triple into an ontology.

ImportIntoOntology: This method imports triples into an ontology

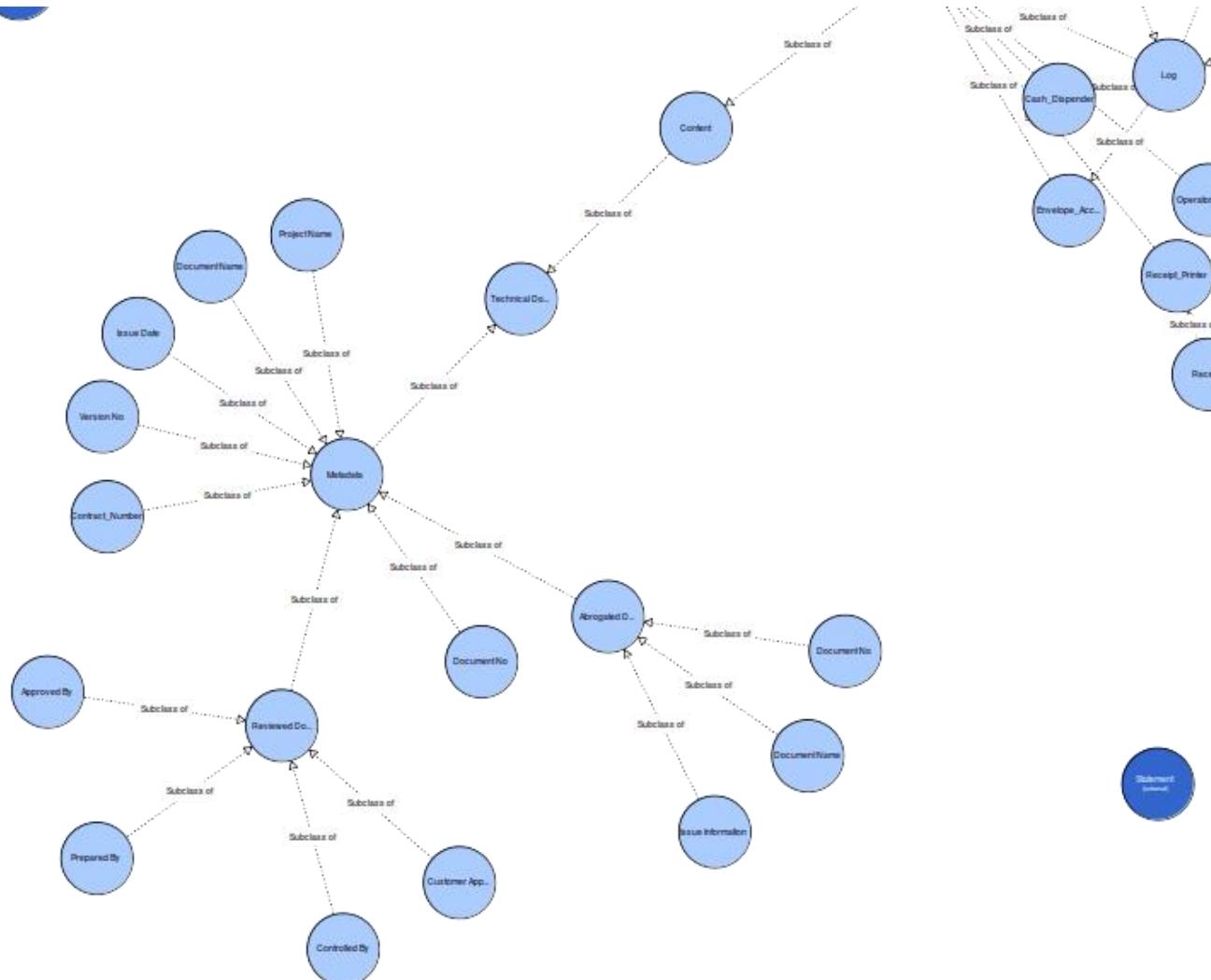
exportOntology: This method exports a specific ontology

executeQuery: This method executes specific query

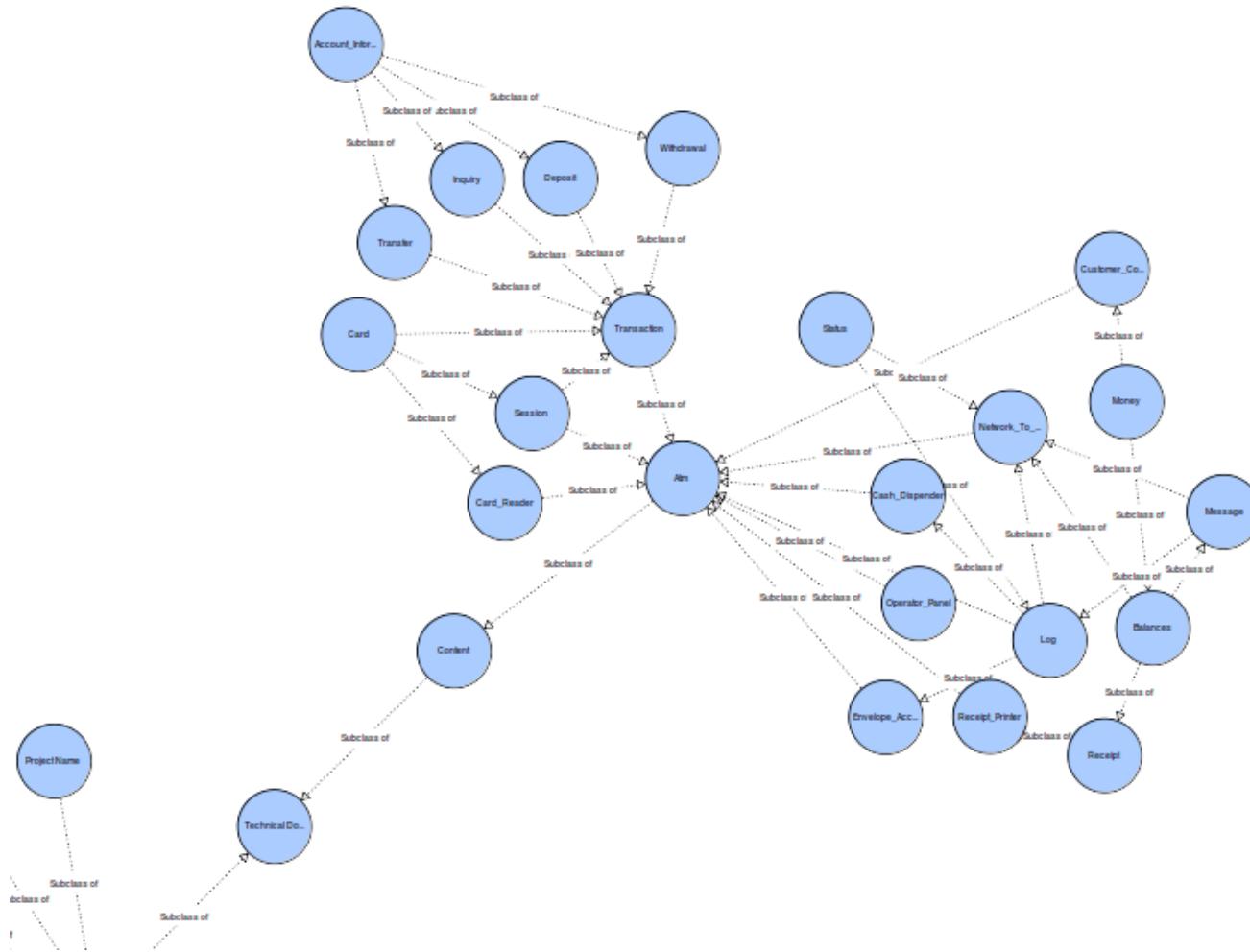
dropOntology: This method drops a specific ontology

removeTriple: This method removes specific triple(s)

Sample Document Ontology Model



Sample Document Ontology Model Instance



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

User Interfaces and Integration (WP6)

Yvan Lussaud, OBEO



Software stack

Integration GUI

GUI

GUI

Semantic annotation

Connector (Text, EMF, Ontology,...)

Ontology

Specific Locations (Text, EMF, Ontology,...)

Mapping API

Storage (XMI)

Components

- Mapping API and implementation (EMF/XMI)
- Text and model (EMF and ontology) connectors
- Semantic annotation

Continuous integration

- Travis <https://travis-ci.org/ModelWriter/Source>
- Nightly and release builds
- Eclipse update sites for each build

Next steps

- Develop synchronization API and GUI
- Integrate semantic parsing and consistency checking

Software stack

Integration GUI

Automated reasoning about locations and links

Semantic GUI

Semantic annotation

Consistency checking

Semantic parsing

Ontology

Mapping GUI

Connectors
(Text, EMF, Ontology,...)

Specific Locations
(Text, EMF, Ontology,...)

Synchronization

Mapping API

Storage (XMI)

Code

- Release 1: 19K line of codes and 13K lines of comment
- Release 2: 30K line of codes and 20K lines of comment

Commits

- Release 1: 60 commits
- Release 2: 88 commits

Tests

- Release 1: 403 unit tests
- Release 2: 688 unit tests + some RCPTT tests

Eclipse update sites

- Two update sites one for EPL features and one for GPL features
- <http://modelwriter.obeo.fr/download>
- Eclipse: Luna, Mars, Neon

Mapping features

- Text, EMF Model full support and ontology partial support
- Mapping base saved to XMI
- EMF/XMI mapping base creation wizard

Semantic features

- Main ontology format supported (RDF, OWL, Turtle, ...)
- Semantic annotation and mapping creation from annotation
- Text extraction from .doc and .docx files

Deliverables for WP6

Kind	Name	Status
Software	D6.3.1-2 Writer enhancements (release 2)	Done
Software	D6.4.2-2 User Interface for the Writer part (release 2)	Done
Software	D6.4.3-2 IDE-integrated User Interface to handle Sync issues (major release 2)	Done
Document	D6.5.1-2 Acceptance Test Plan (release 2)	Merged
Software	D6.5.2-2 Automated Acceptance Tests (release 2)	Done
Document	D6.6.1-2 Acceptance Test Procedures (release 2)	Merged
Software	D6.7.1-2 ModelWriter major release (release 2)	Done
Software	D6.7.2-2 WP2 Integration in ModelWriter and Documentation (release 2)	Done
Document	D6.8.1-2 Evaluation report (release 2)	Coming soon

Completion/quick outline

- + limit interactions to a minimum
- - Need to know what can be targeted (not possible for text)
- - Project possible targets out of their editing form

...resolving impacts in terms of each component's architecture. Described architecture defines basic structure and organization for the next research component: Work Packages. It defines also the interactions between:

del" S Work Packages

VP6.1 - Selection an

VP3 - storing and re
" side

VP6.2 - selection an

VP2.1 - storing text

VP2.2 - generating b
" side

VP4 - the repository

i.e. semantic/syntac
defining the knowled

knowledge base is th
ents, the architect

Select Target Location

Filter

MW Ontologie

WorkPackage

proposed location 12

Location 2

proposed location 21

Illustrating the architectural design details, let us review the objectives of each package.

WP6

Ergonomics

Goals

- No projection / get elements from the editor
- Works with text
- Works with graphics

Palette

- Eclipse view
- Add element from editors to the palette
- Link to elements in the palette

Possible improvements

- Quick outline with elements of the palette ?
- Use current selection and an action on element from the palette ?

5 Demonstrations

Ferhat Erata (UNIT, WP3 Leader)

Emre Kirmizi (UNIT, Technical Contact)

Dr. Claire Gardent (CNRS/LORIA, WP2 Leader)

Bikash Gyawali (CNRS/LORIA, Technical Contact)

Anastasia Shimorina (CNRS/LORIA, Technical Contact)

Dr. Geylani Kardas (KOCSISTEM, Consultant)

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 - Word". The ribbon menu is visible at the top, showing tabs like FILE, HOME, INSERT, DESIGN, etc. The main content area contains the following text:

Library Management System

GLOSSARY

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

1.2 ...

REQUIREMENTS

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

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

1.3 ...

At the bottom, the status bar shows "PAGE 1 OF 1 76 WORDS ENGLISH (UNITED STATES) %80".

What is a text? (document file formats)

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



Java - PropertyPage/test/document.xml - Eclipse

File Edit Source Navigate Search Project Sample Run Window Help

Sample Plain Text File document.xml

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

Design Source

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

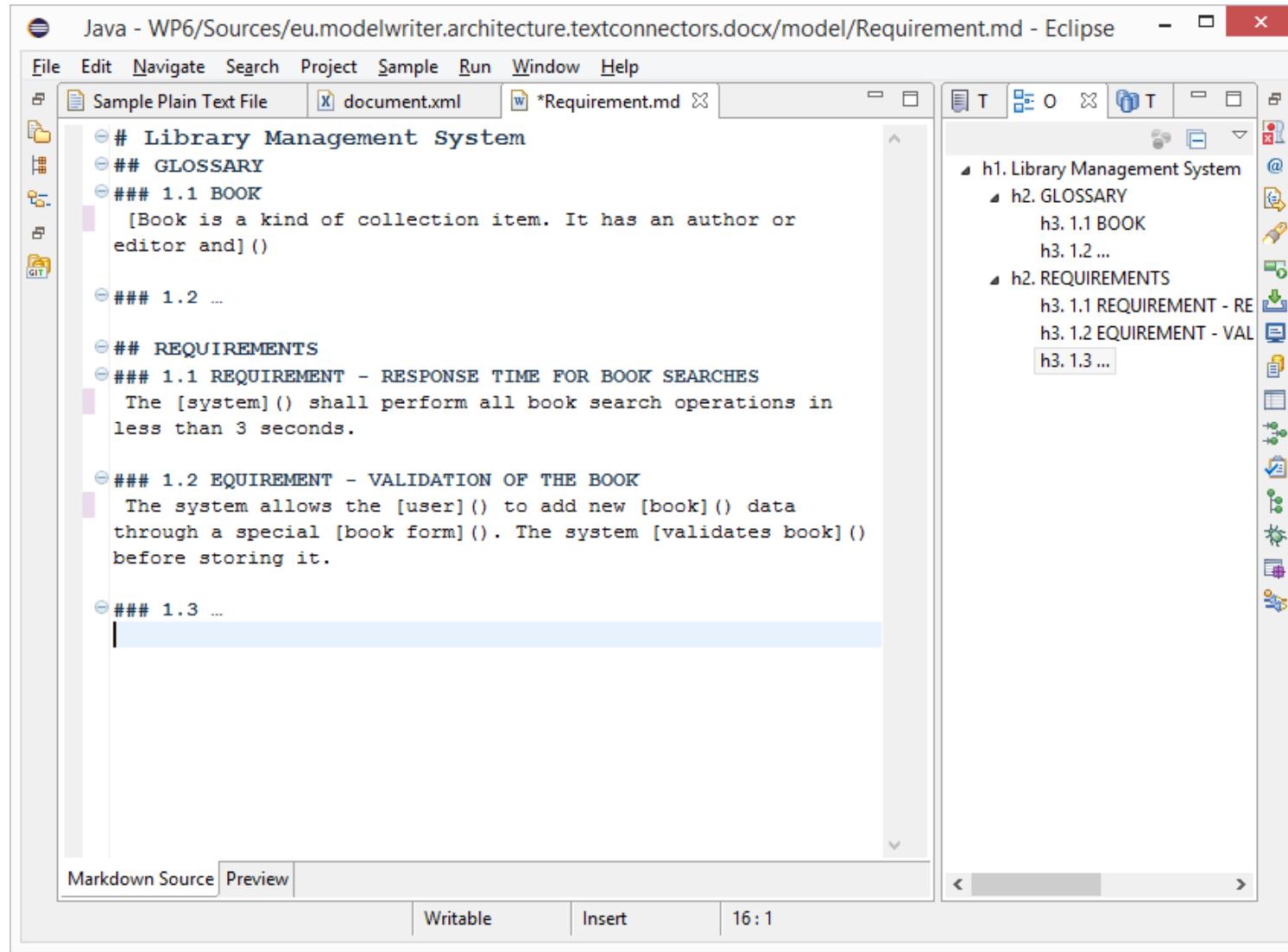
Property	Value
http://schemas.openxmlformats.org/wordprocessingml/2006/main w:val	Heading1

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 is color-coded for syntax highlighting. 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: 'http://schemas.openxmlformats.org/wordprocessingml/2006/main w:val' with a value of 'Heading1'. The status bar at the bottom shows the path 'w:document/w:body/w:pPr/w:pStyle/w:val', the status 'Writable', the 'Smart Insert' feature, and the current position '33 : 38'.

What is a text? (.md source file)

text/markdown (ICANN Standard)



The screenshot shows the Eclipse IDE interface with a .md file open. The left pane displays the content of the file, which is a structured text document in Markdown format. The right pane shows a tree view of the document's structure.

File menu: File, Edit, Navigate, Search, Project, Sample, Run, Window, Help

Open files: Sample Plain Text File, document.xml, *Requirement.md

Content:

```
# 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 ...
```

Tree View:

- 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 ...

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
```

Content:

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 ...

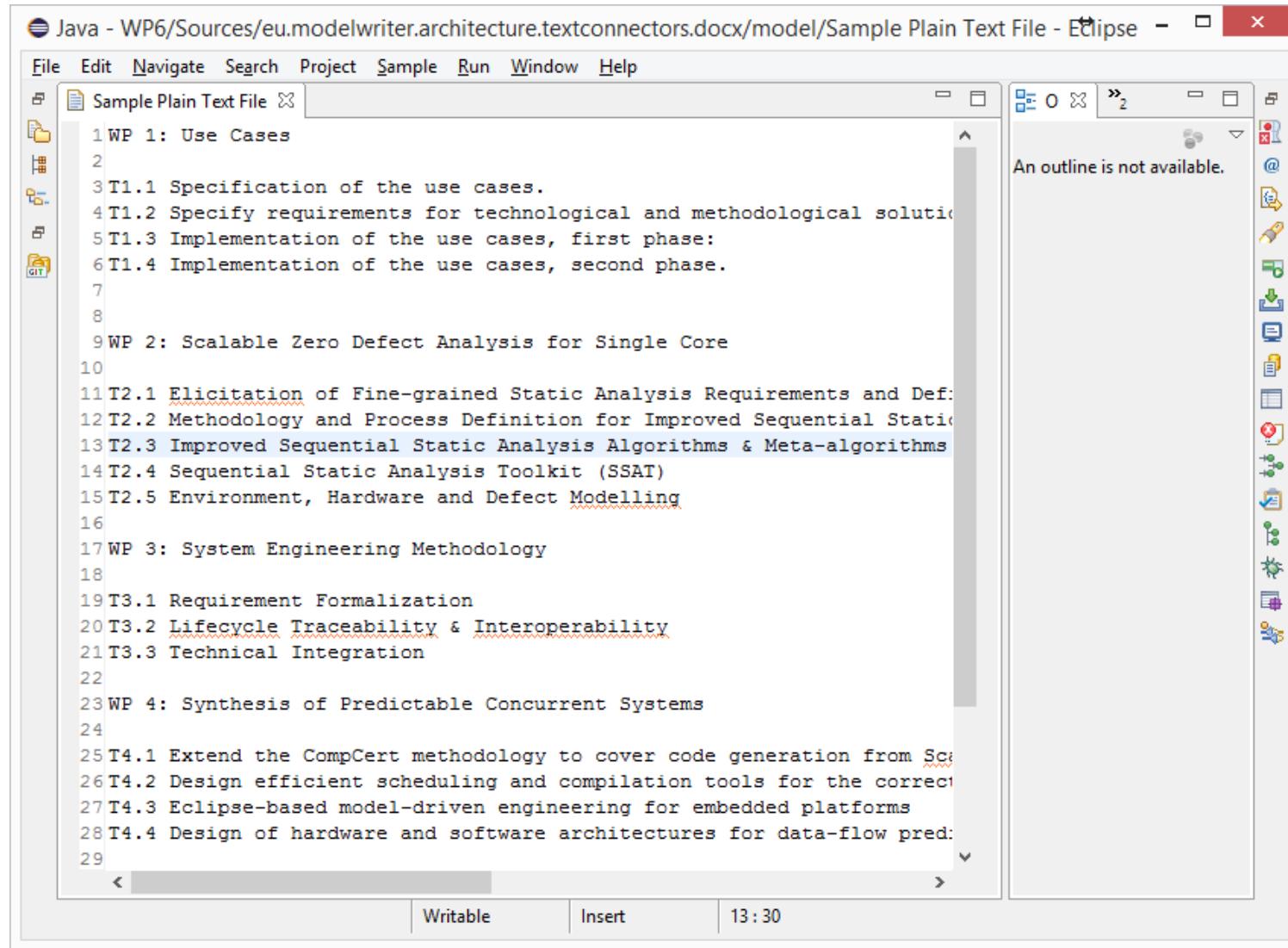
Markdown Source Preview

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)



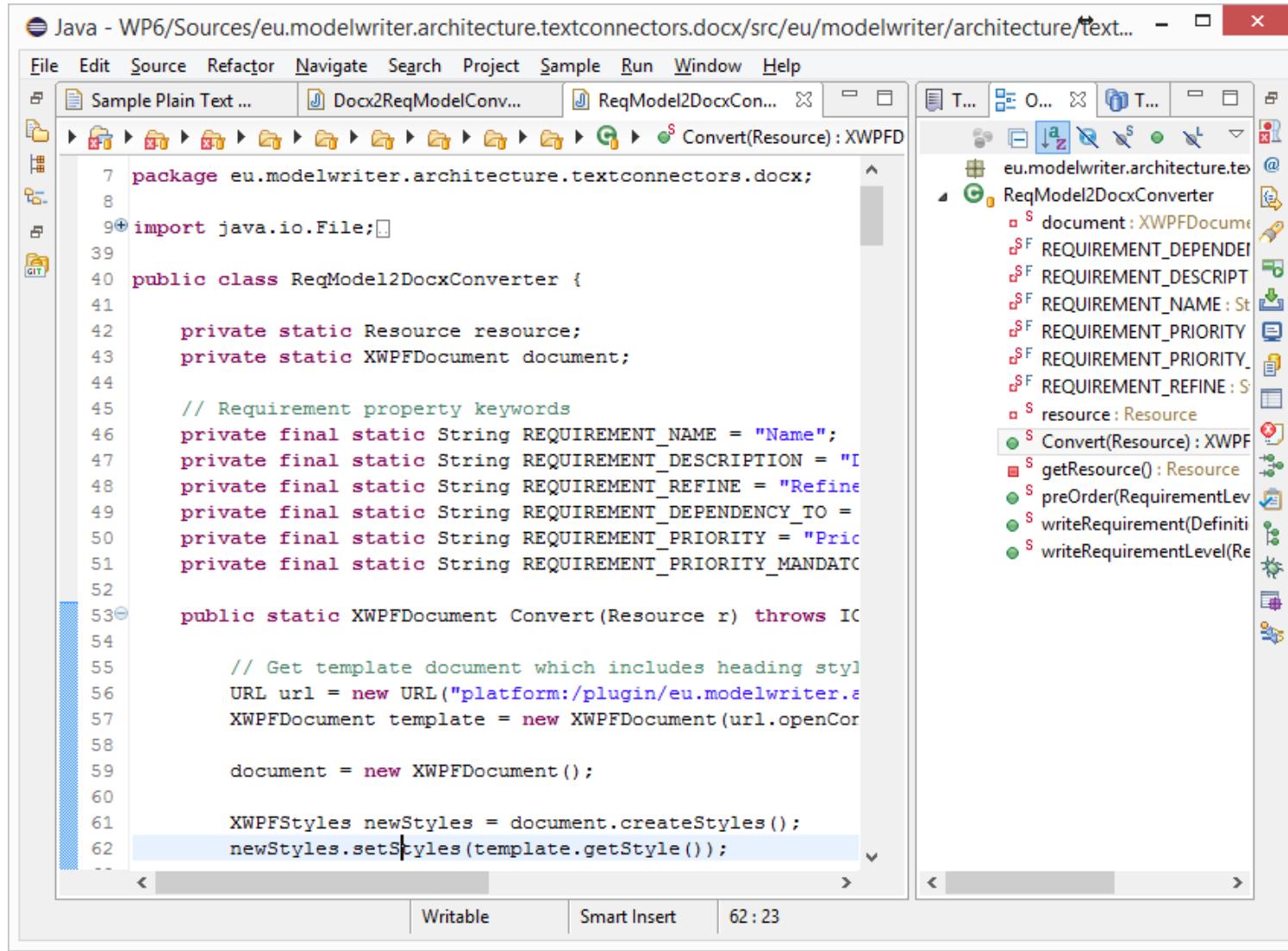
The screenshot shows the Eclipse IDE interface with a text editor open. The title bar reads "Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/model/Sample Plain Text File - Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Sample, Run, Window, and Help. The left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

Java Code:

```
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
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62 }
```

UML Class Diagram:

```
class ReqModel2DocxConverter {
    +S document : XWPFDocument
    +SF REQUIREMENT_DEPENDENCY_TO : Resource
    +SF REQUIREMENT_DESCRIPTION : String
    +SF REQUIREMENT_NAME : String
    +SF REQUIREMENT_PRIORITY : String
    +SF REQUIREMENT_PRIORITY_MANDATORY : Boolean
    +SF REQUIREMENT_REFINE : String
    +S resource : Resource
    +S Convert(Resource) : XWPFDocument
    +S getResource() : Resource
    +S preOrder(RequirementLevel)
    +S writeRequirement(Definition)
    +S writeRequirementLevel(RequirementLevel)
}
```

What is a model?

Everything is a model! (ReqIF Standard)

Requirements Interchange Format



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

File Edit Search Requirements fmStudio Window Help

Quick Access

My.reqif Requirements Document

Outline

ID Name Description

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

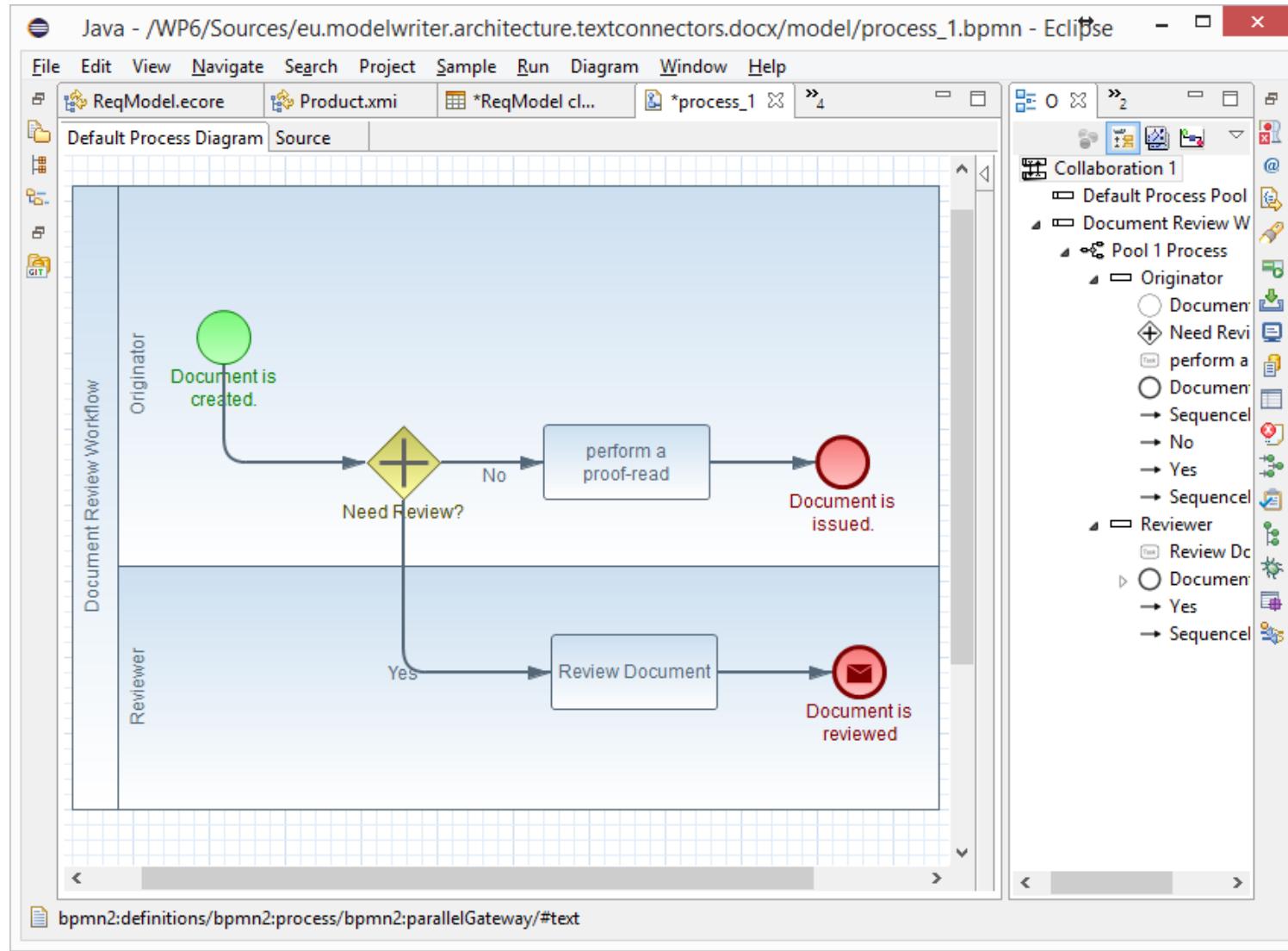
Properties

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

Standard Attributes All Attributes

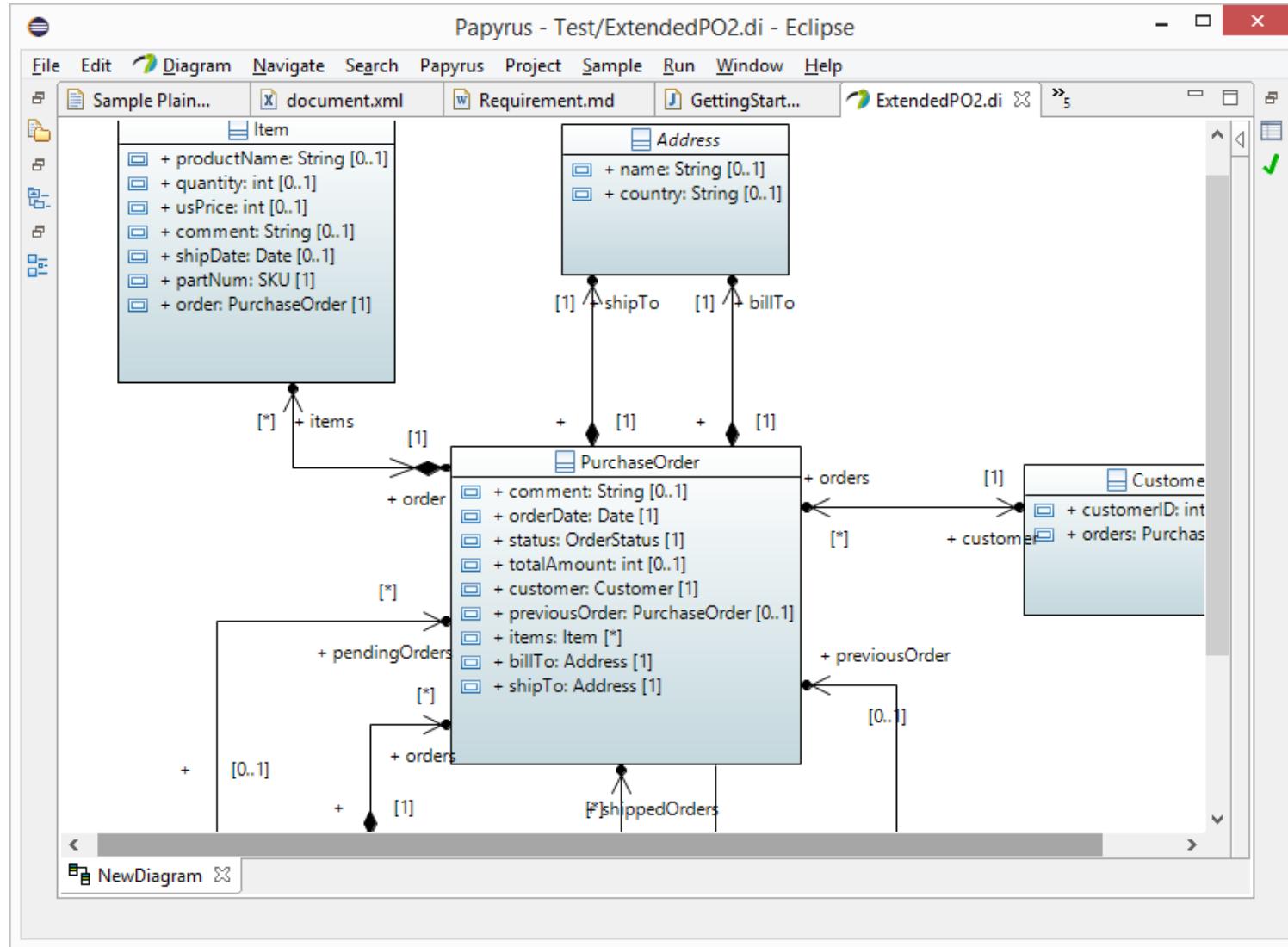
Everything is a model! (BPMN Standard)

Business Process Model & Notation



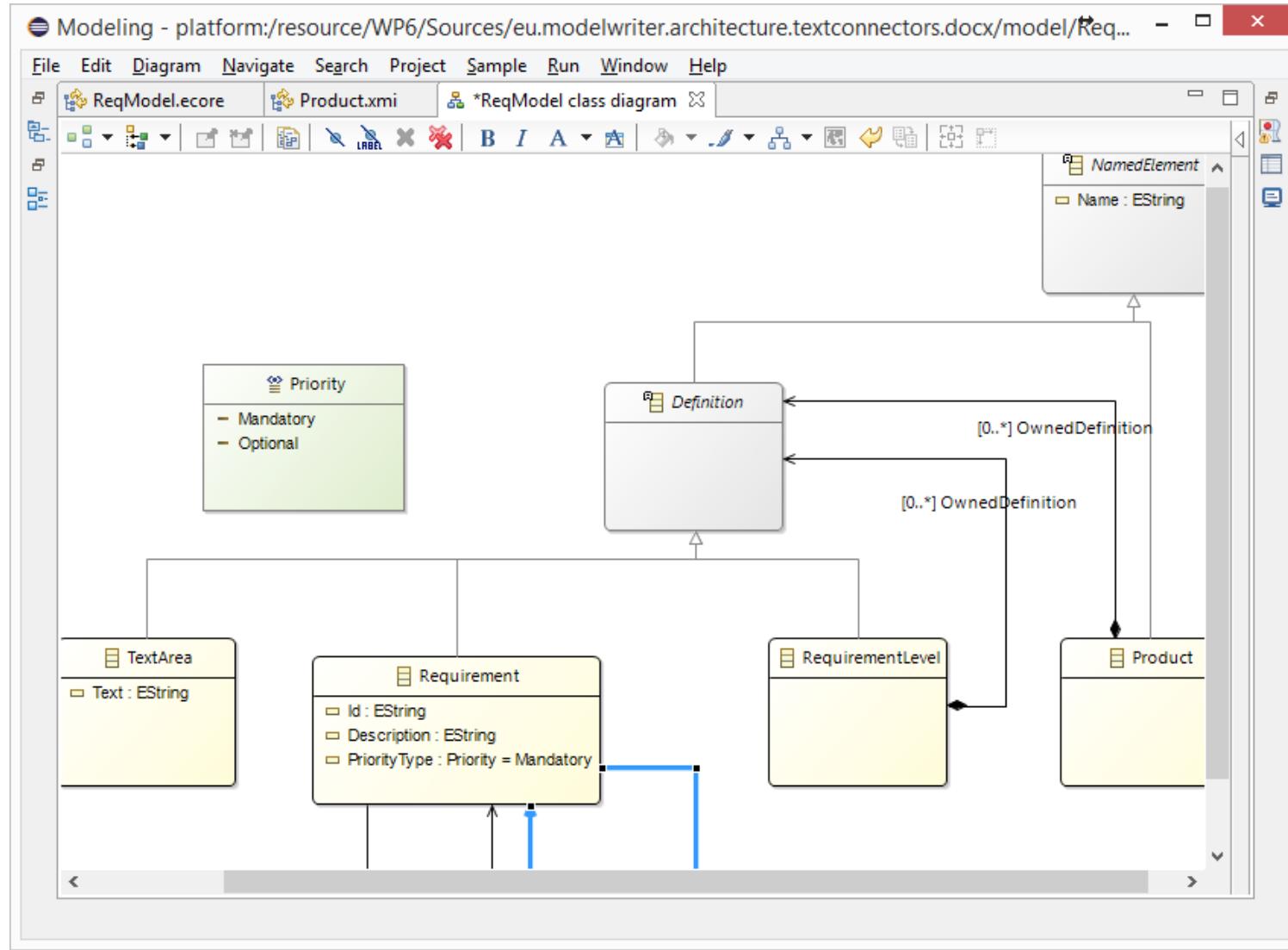
Everything is a model! (UML Standard)

UML Modeling Languages



Everything is a model!

Eclipse Modeling Framework (EMF)



Everything is a model!

Tree-based or Tabular Representations



The screenshot shows a modeling environment with the following components:

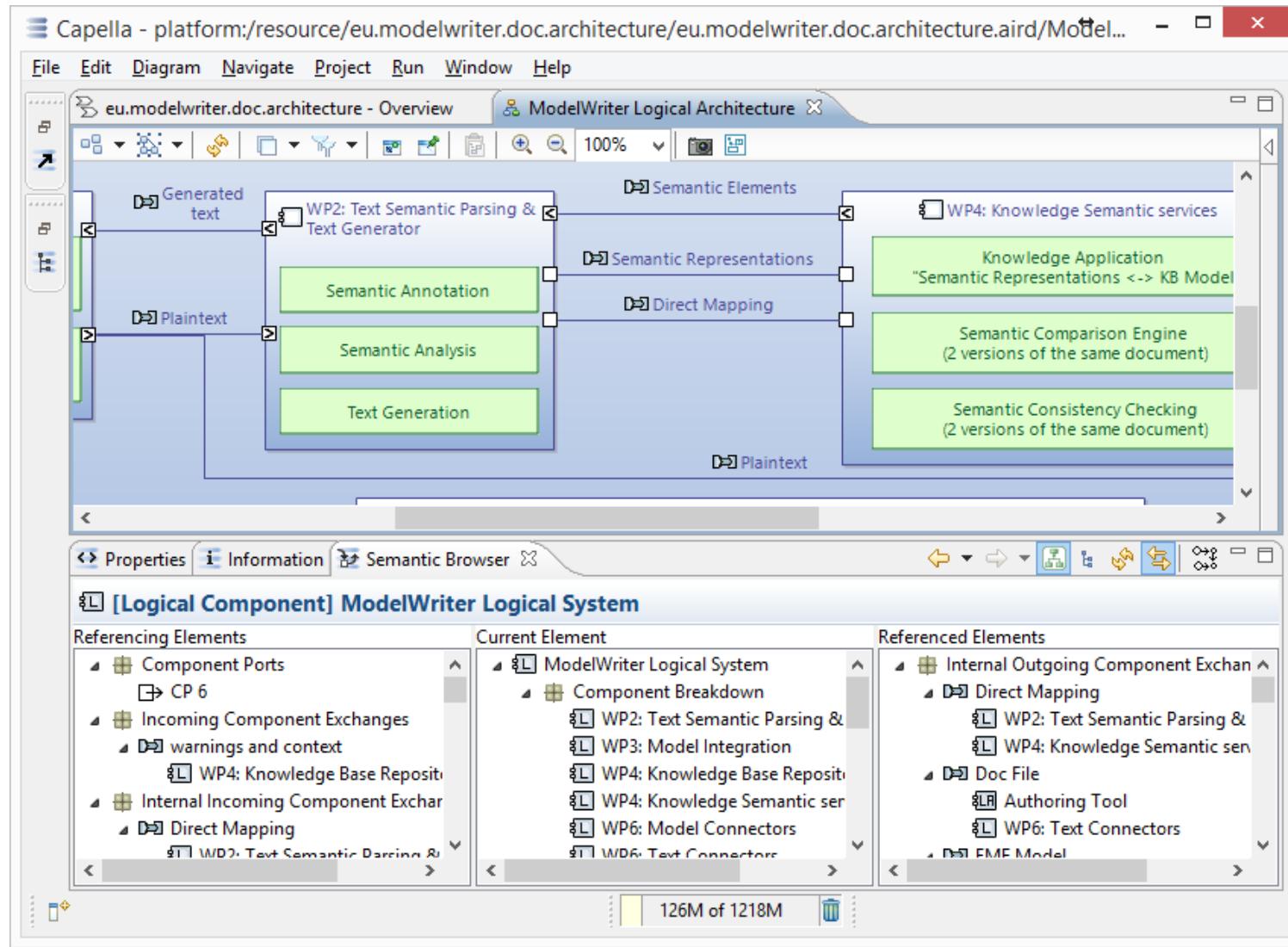
- Top Bar:** File, Edit, Navigate, Search, Project, DTable, Sample, Run, Window, Help.
- Left Sidebar:** Shows a tree view of model elements:
 - ReqModel.ecore
 - Product.xmi
 - NamedElement
 - Name : EString
 - Product -> NamedElement
 - OwnedDefinition : Definition
 - Definition -> NamedElement
 -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

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

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

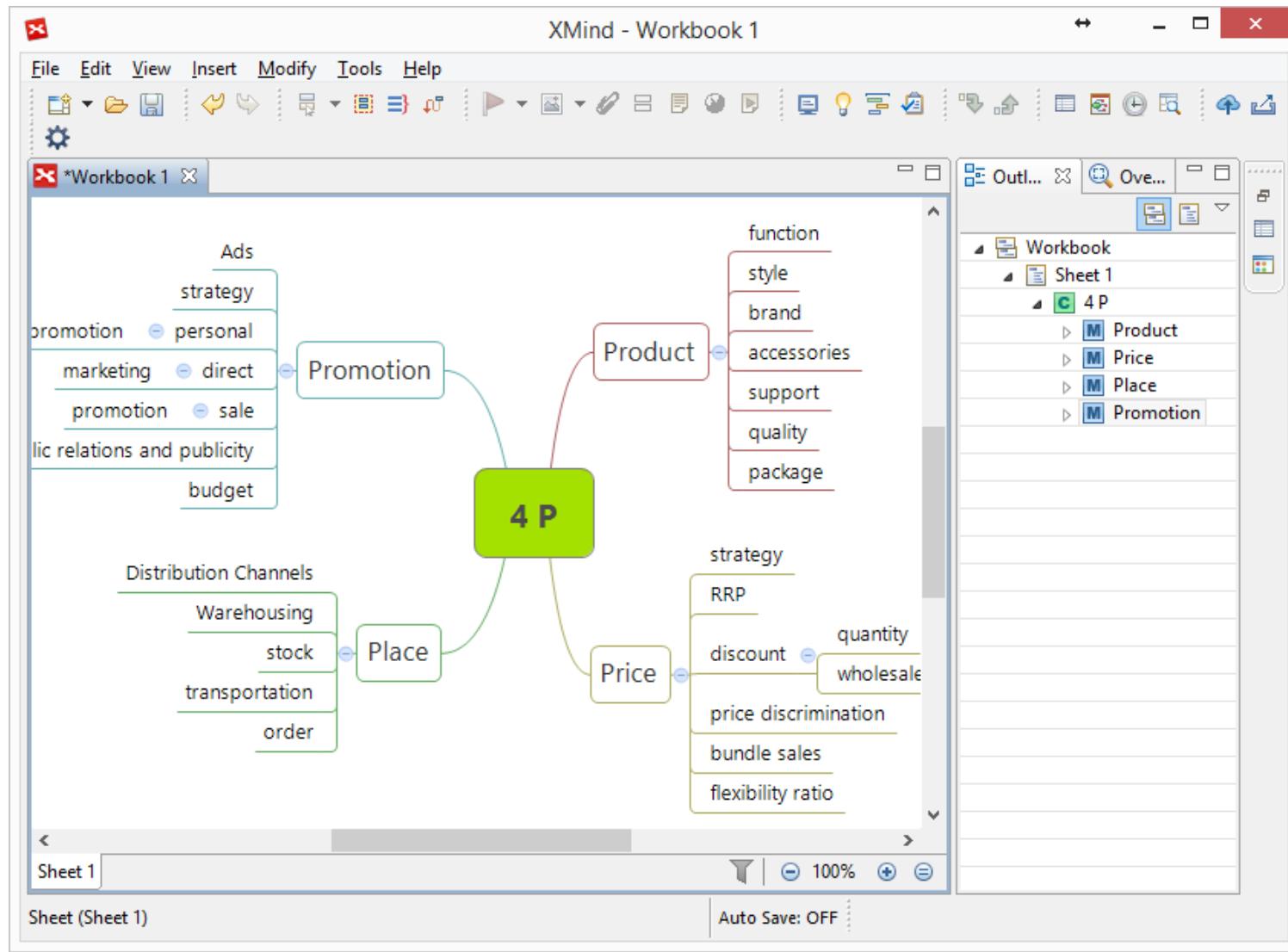
Everything is a model!

Software/System Architecture Design



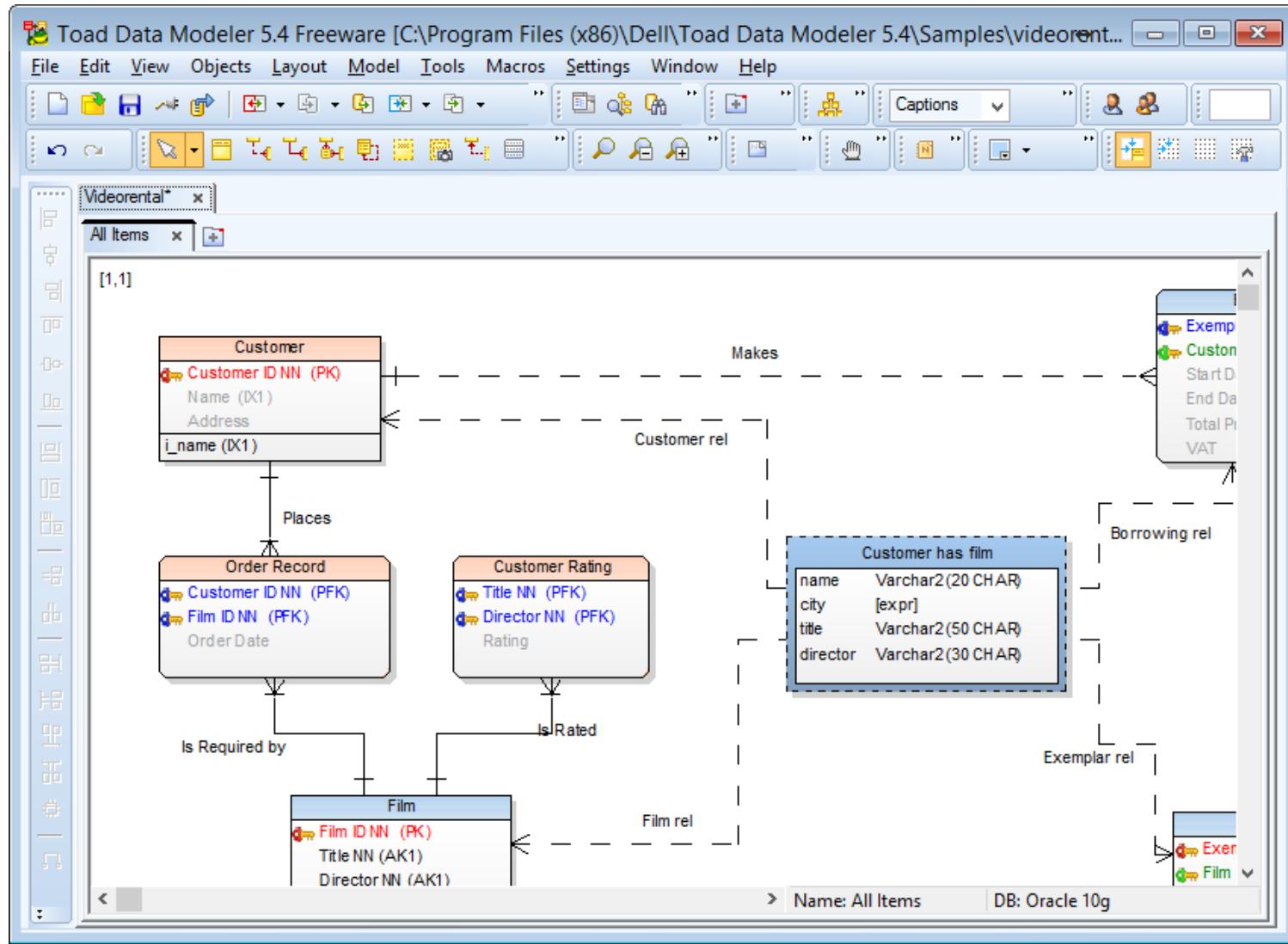
Everything is a model!

Topic Maps, Mind Maps, Vocabularies ...



Everything is a model!

Databases (ER, IDEF1.x)



Everything is a model! (Textual Lang.)

Domain Specific Languages



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

File Edit Navigate Search Project Sample Run Window Help

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

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

Writable Insert 11:9

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

Even Programming Languages (ASTs)

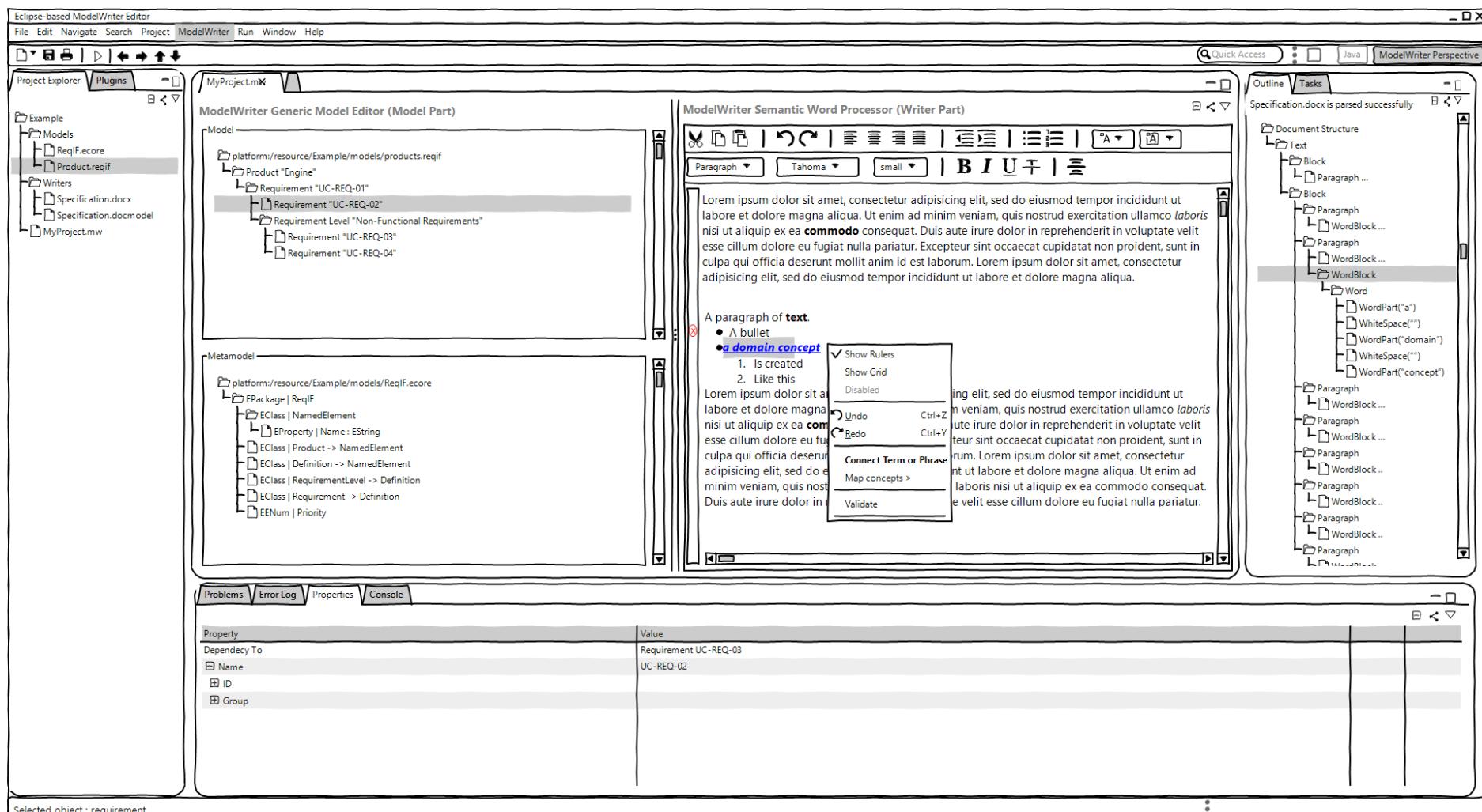
A screenshot of an IDE interface. On the left, a code editor shows Java code for a class named ReqModel2DocxConverter. The code includes imports for java.io.File and XWPFD, and defines a static Resource resource and XWPFDocument document. It contains several static final String variables for requirement properties like REQUIREMENT_NAME, REQUIREMENT_DESCRIPTION, REQUIREMENT_REFINE, REQUIREMENT_DEPENDENCY_TO, REQUIREMENT_PRIORITY, and REQUIREMENT_PRIORITY_MANDATORY. The class has a static method Convert(Resource r) that creates an XWPFDocument and sets its styles from a template. On the right, a tree-based AST (Abstract Syntax Tree) viewer shows the structure of the ReqModel2DocxConverter class. It lists nodes for document, REQUIREMENT_DEPENDENCY_TO, REQUIREMENT_DESCRIPTION, REQUIREMENT_NAME, REQUIREMENT_PRIORITY, REQUIREMENT_REFINE, and resource, along with their corresponding methods like Convert(Resource), getResource(), preOrder(RequirementLevel), and writeRequirement(RequirementLevel).

```
Java - WP6/Sources/eu.modelwriter.architecture.textconnectors.docx/src/eu/modelwriter/architecture/text... - □ ×
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;
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 I
26
27         // Get template document which includes heading styl
28         URL url = new URL("platform:/plugin/eu.modelwriter.a
29         XWPFDocument template = new XWPFDocument(url.openCor
30
31         document = new XWPFDocument();
32
33         XWPFFormats newFormats = document.createStyles();
34         newFormats.setFormats(template.getStyle());
35
36
37
38
39
39+     public static void main(String[] args) {
40
41         ReqModel2DocxConverter converter = new ReqModel2DocxC
42
43         Resource resource = converter.getResource();
44
45         Requirement requirement = resource.getRequirement("Name
46
47         requirement.setDescription("Initial requirement descri
48
49         requirement.setPriority("High");
50
51         requirement.setRefinement("Refinement 1");
52
53         converter.writeRequirement(requirement);
54
55
56
57
58
59
59+     }
59+ }
```

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

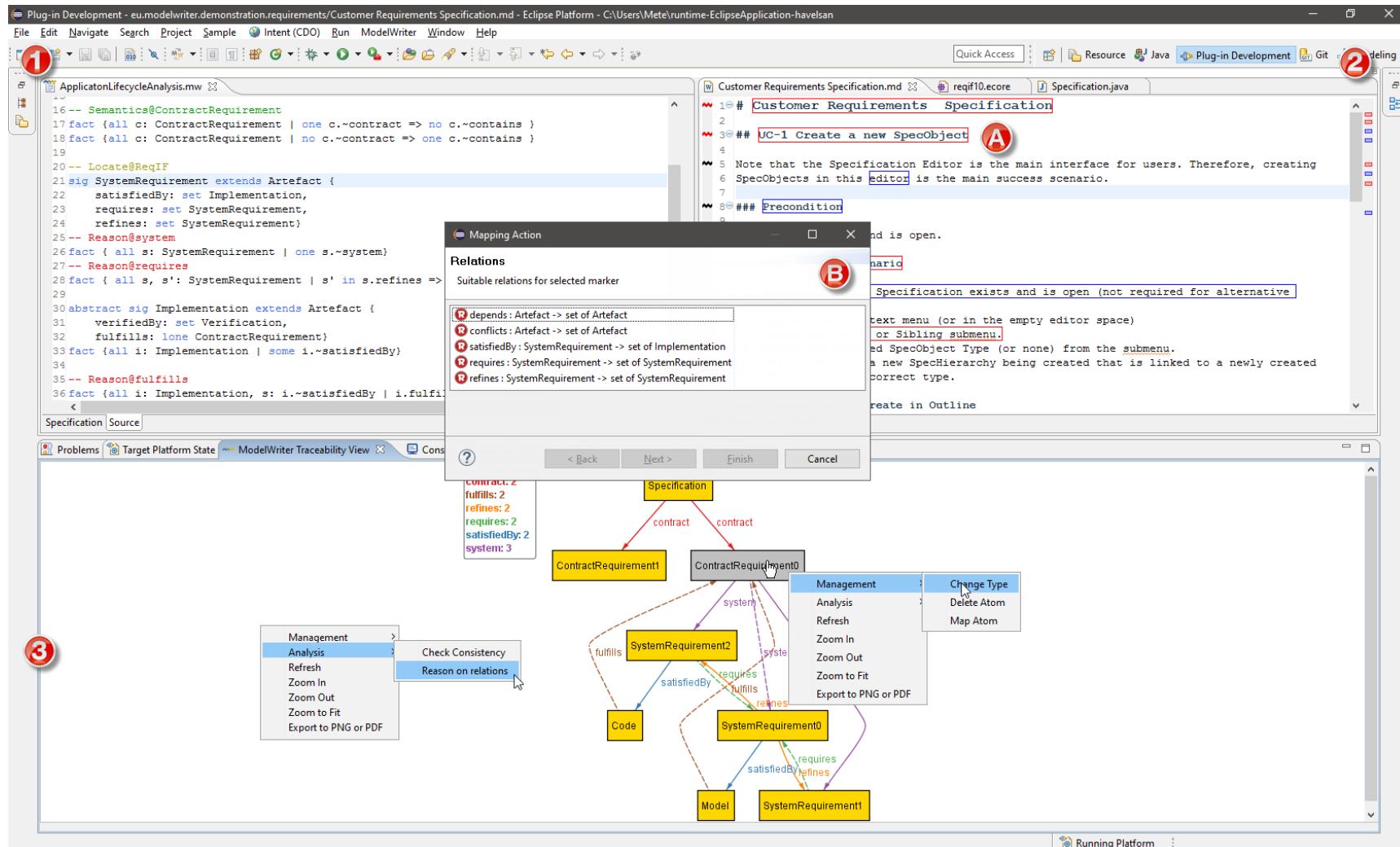


ModelWriter – The Solution



Text & Model-Synchronized Document Engineering Platform

Solution – Knowledge Capture



Text & Model-Synchronized Document Engineering Platform

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
Search Link
2 P... Add Link
ABS... Remove Link

Generate Links

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>81.8</j:0:hasOffset>
  <j:0:isSameAs>http://airbus-group/opp-function#Attach</j:0:isSameAs>
  <j:0:hasValue>attach</j:0:hasValue>
</rdf:Description>
<rdf:Description rdf:about="http://ModelWriter/TxtDocument/attached709">
  <j:0:hasOffset>709</j:0:hasOffset>
  <j:0:isMorphologySimilarTo>http://airbus-group/opp-function#Attach</j:0:isMorphologySim
  <j:0:hasValue>attached</j:0:hasValue>
</rdf:Description>
</rdf:RDF>
```

Text & Model-Synchronized Document Engineering Platform

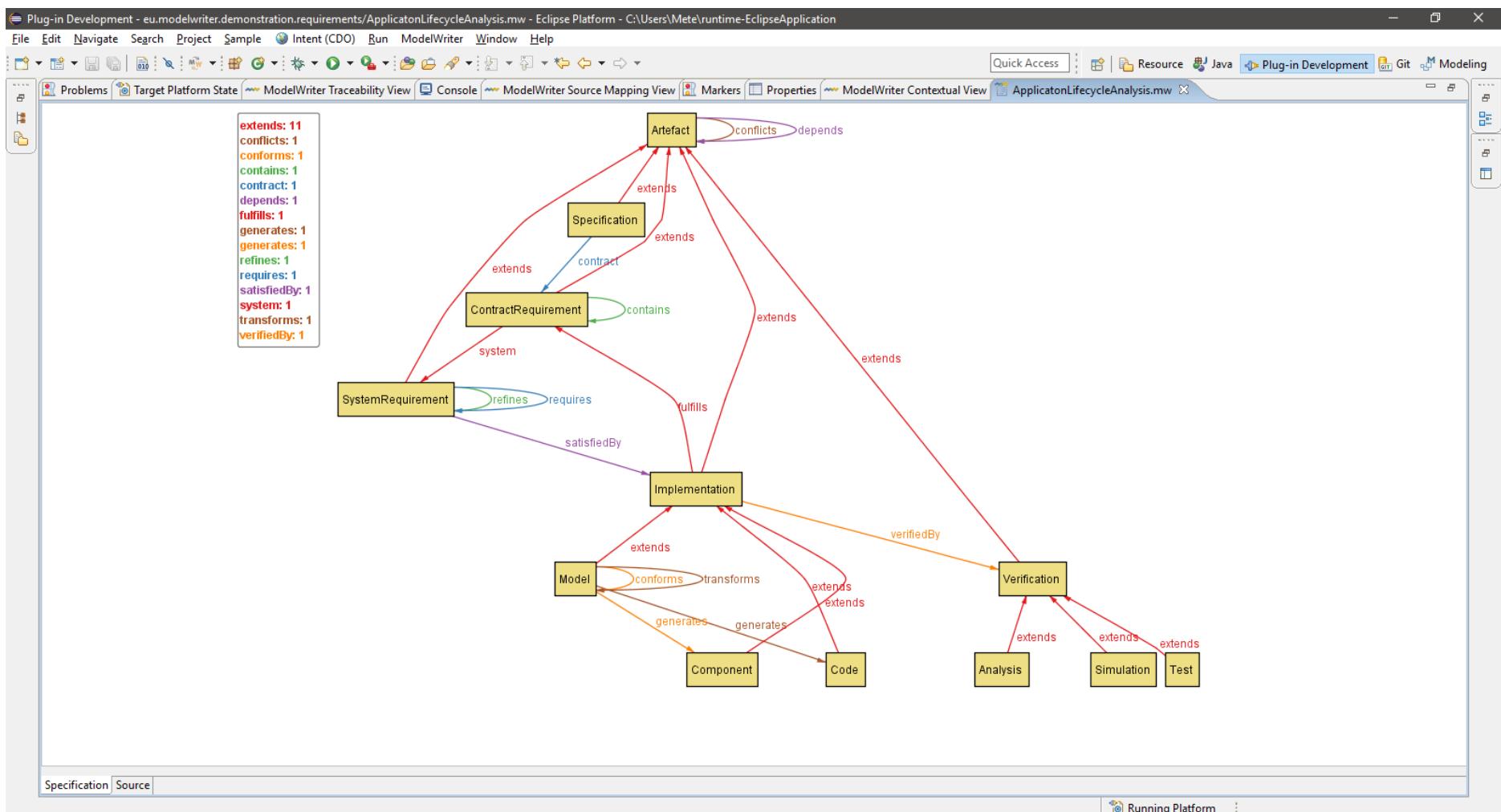


Synchronization is maintained!

What about configuration/formalization of the platform?



Configuration: Havelsan example



A Formal Specification Model to configure the ModelWriter

Is it possible to vizualize the trace links?



Traceability: Havelsan example

Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - Eclipse Platform - C:\Users\Metz\runTime-EclipseApplication-havelsan

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

Quick Access Resource Java Plug-in Development Git Modeling

ApplicationLifecycleAnalysis.mw

```
1 module eu.modelwriter/actions/havelsan/alm
2
3 abstract sig Artefact {
4   depends: set Artefact,
5   conflicts: set Artefact)
6 -- Reason@conflicts
7 fact {~conflicts in conflicts}
8
9 one sig Specification extends Artefact {
10   contract: some ContractRequirement)
11 -- Locate@Text
12 -- Discover@ContractRequirement expect 3
13 sig ContractRequirement extends Artefact {
14   system: set SystemRequirement,
15   contains: set ContractRequirement)
16
```

Customer Requirements Specification.md

```
10 # Customer Requirements Specification
11
12 ## UC-1 Create a new SpecObject
13 Note that the Specification Editor is the main interface for users. Therefore,
14 creating SpecObjects in this editor is the main success scenario.
15
16 ### Precondition
17 ReqIF model exists and is open.
18
19 ## Main Success Scenario
20
21 1. We assume that a Specification exists and is open (not required for alternative
22 scenario)
23 2. Open a row's context menu (or in the empty editor space)
24 3. Select the Child or Sibling submenu.
```

Specification

ContractRequirement1

ContractRequirement0

SystemRequirement0

SystemRequirement2

Code

ContractRequirement2

Management Analysis >

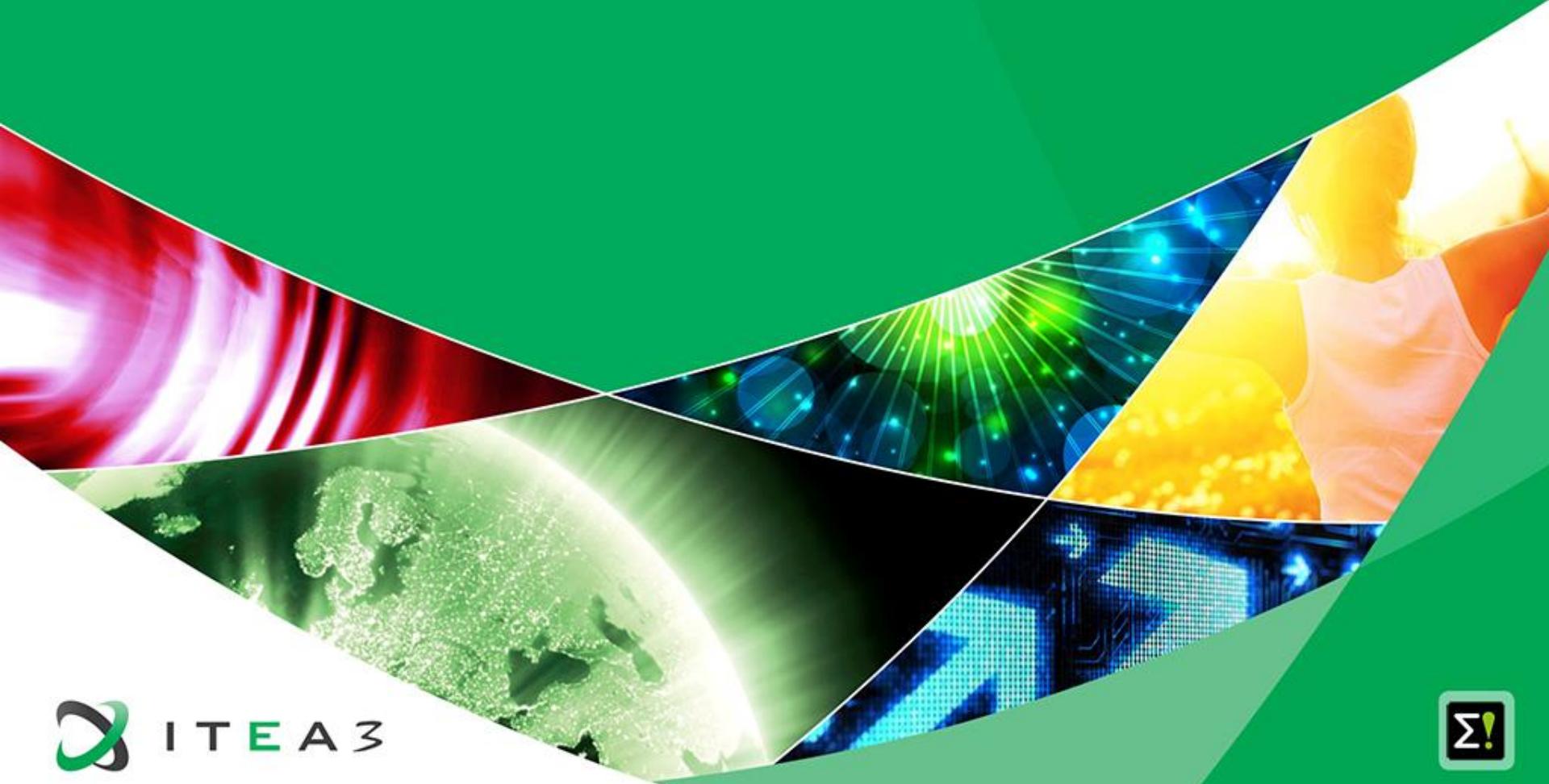
- Check Consistency
- Reason on Solutions
- Discover Atoms
- Clear All Reasoned Tuples

Refinement Traceability View

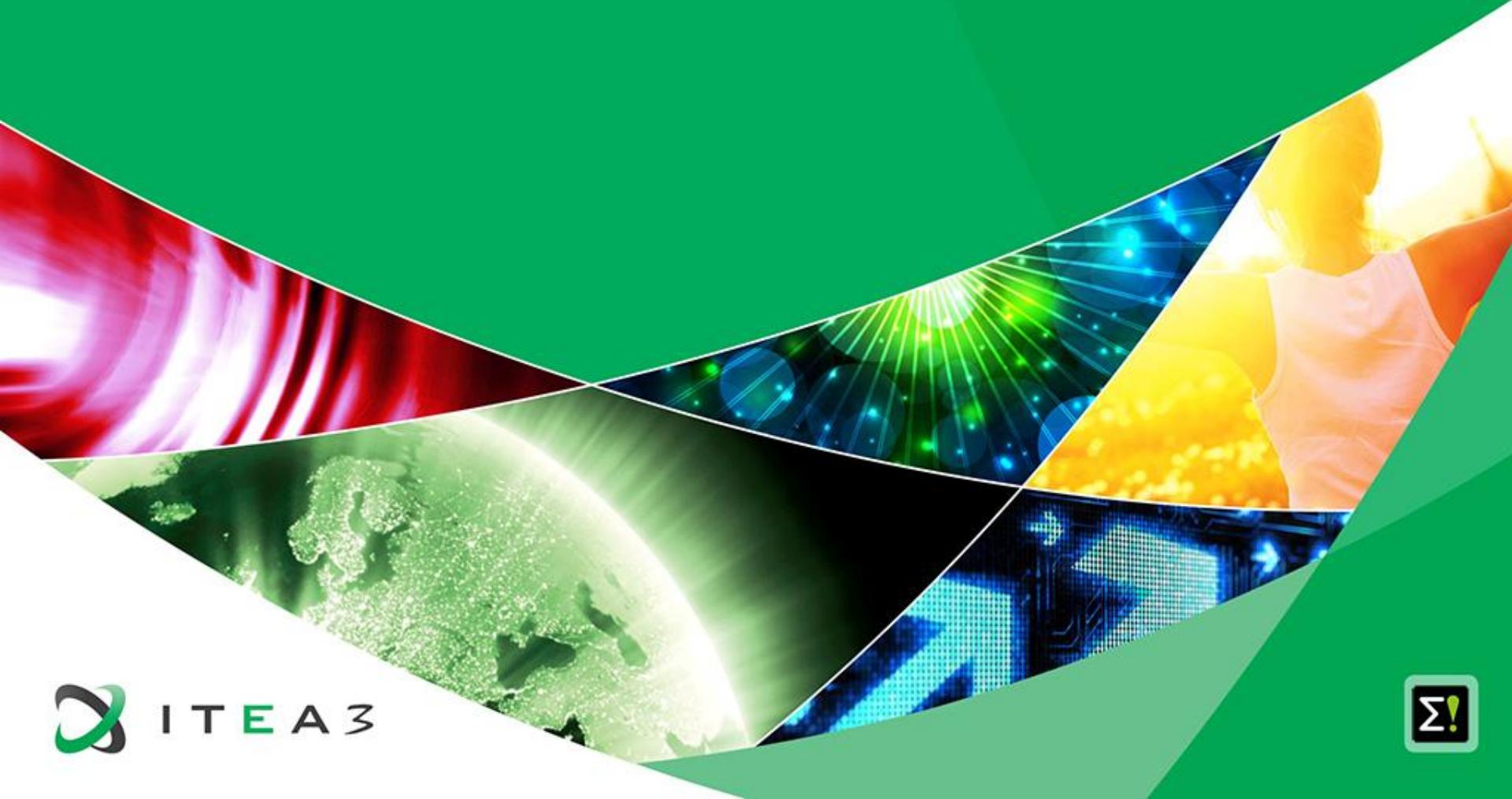
```
graph TD; Specification -- contract --> ContractRequirement1; Specification -- contract --> ContractRequirement0; ContractRequirement0 -- system --> SystemRequirement0; ContractRequirement0 -- system --> ContractRequirement2; SystemRequirement0 -- requires --> SystemRequirement2; SystemRequirement0 -- refines --> SystemRequirement1; SystemRequirement1 -- satisfiesBy --> SystemRequirement2; SystemRequirement1 -- fulfills --> ContractRequirement2; SystemRequirement2 -- fulfills --> ContractRequirement1
```

A Formal Specification Model to configure the ModelWriter

Airbus Use Cases



Ford-Otosan Use Case



Havelsan Use Case

