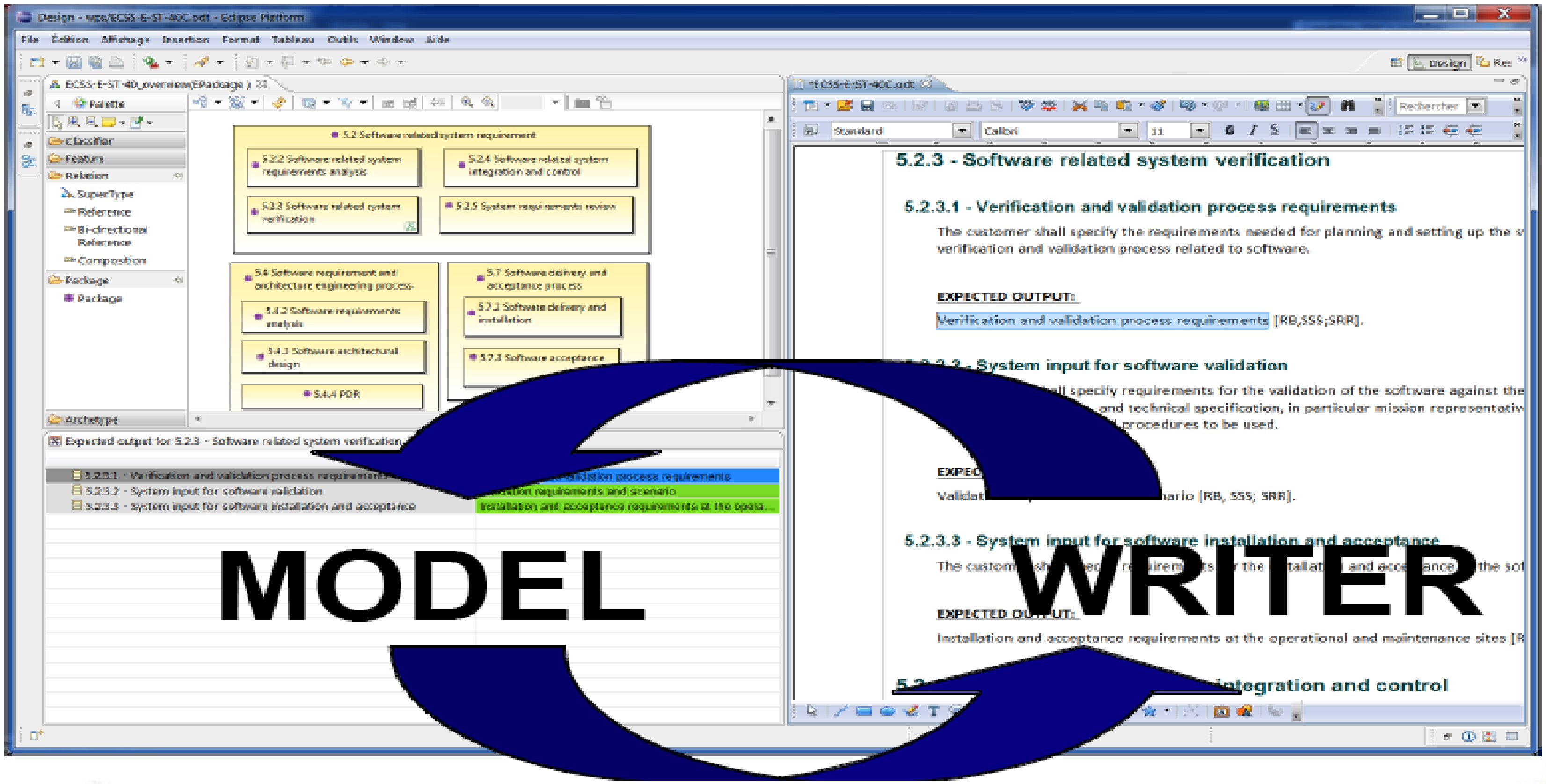
ModelWriter

Text & Model-Synchronized Document Engineering Platform


Design - wps/ECSS-E-ST-40C.odt - Eclipse Platform
File Édition Affichage Insertion Format Tableau Outils Window Aide
ECSS-E-ST-40_overview(EPackage)
Palette
Classifier
Feature
Relation
SuperType
Reference
Bi-directional Reference
Composition
Package
Package
Archetype
5.2 Software related system requirement
5.2.2 Software related system requirements analysis
5.2.4 Software related system integration and control
5.2.3 Software related system verification
5.2.5 System requirements review
5.4 Software requirement and architecture engineering process
5.7 Software delivery and acceptance process
5.4.2 Software requirements analysis
5.7.2 Software delivery and installation
5.4.3 Software architectural design
5.7.3 Software acceptance
5.4.4 PDR
*ECSS-E-ST-40C.odt
Standard
Calibri
11
Rechercher
Design
MODEL
WRITER

Expected output for 5.2.3 - Software related system verification

5.2.3.1 - Verification and validation process requirements	[illegible]alidation process requirements
5.2.3.2 - System input for software validation	[illegible]ation requirements and scenario
5.2.3.3 - System input for software installation and acceptance	Installation and acceptance requirements at the opera...

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 s
verification and validation process related to software.

EXPECTED OUTPUT:

Verification and validation process requirements [RB,SSS;SRR].

[illegible] - System input for software validation

[illegible] all specify requirements for the validation of the software against the
[illegible] and technical specification, in particular mission representativ
[illegible] procedures to be used.

EXPEC[illegible]

Validat[illegible] ario [RB, SSS; SRR].

5.2.3.3 - System input for software installation and acceptance

The custom[illegible] sh[illegible] re[illegible] ire[illegible] ts [illegible] the [illegible] tallat[illegible] and acce[illegible] ance [illegible] the sof

EXPECTED OU[illegible]UT:

Installation and acceptance requirements at the operational and maintenance sites [R

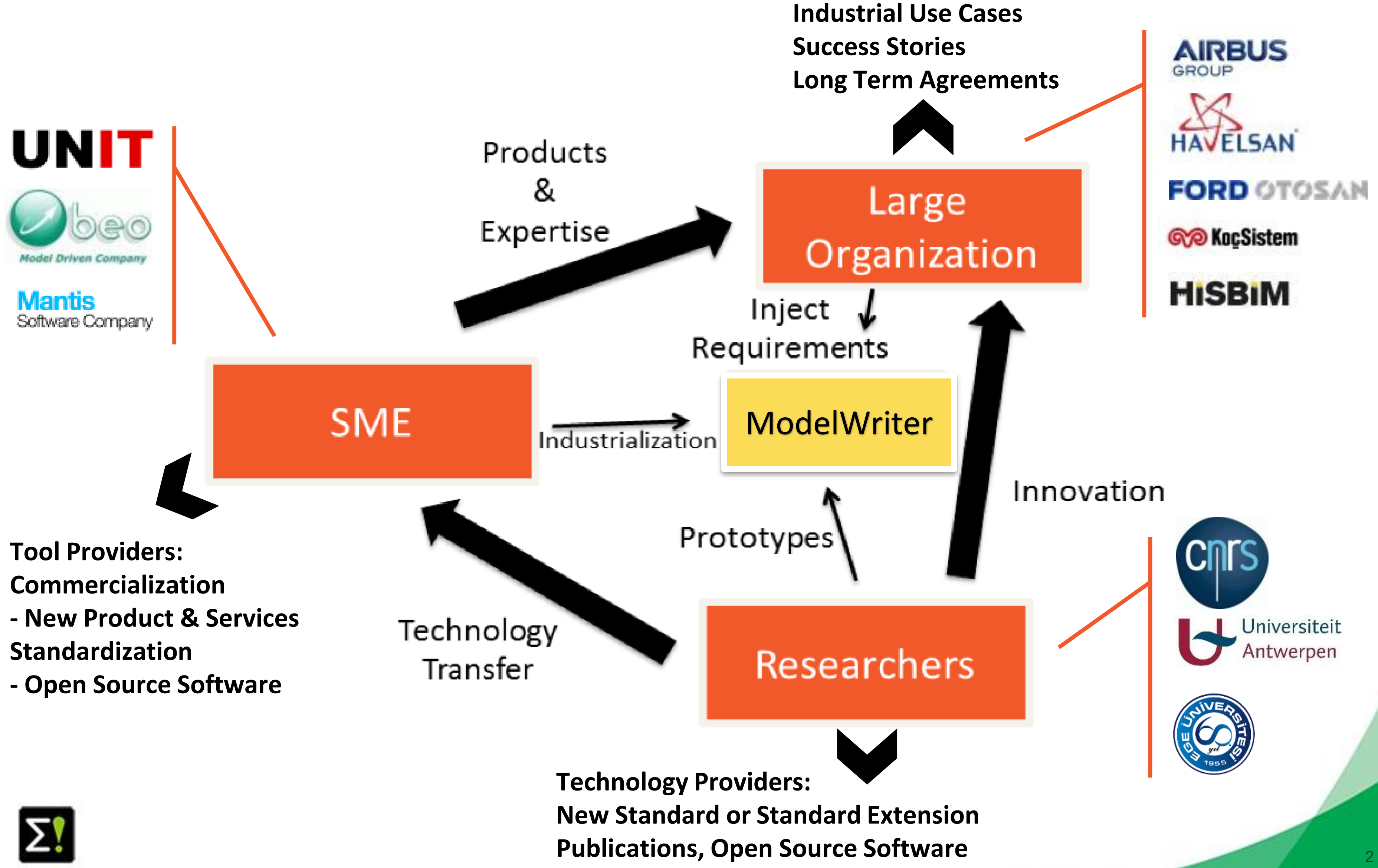
5.2[illegible] integration and control

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

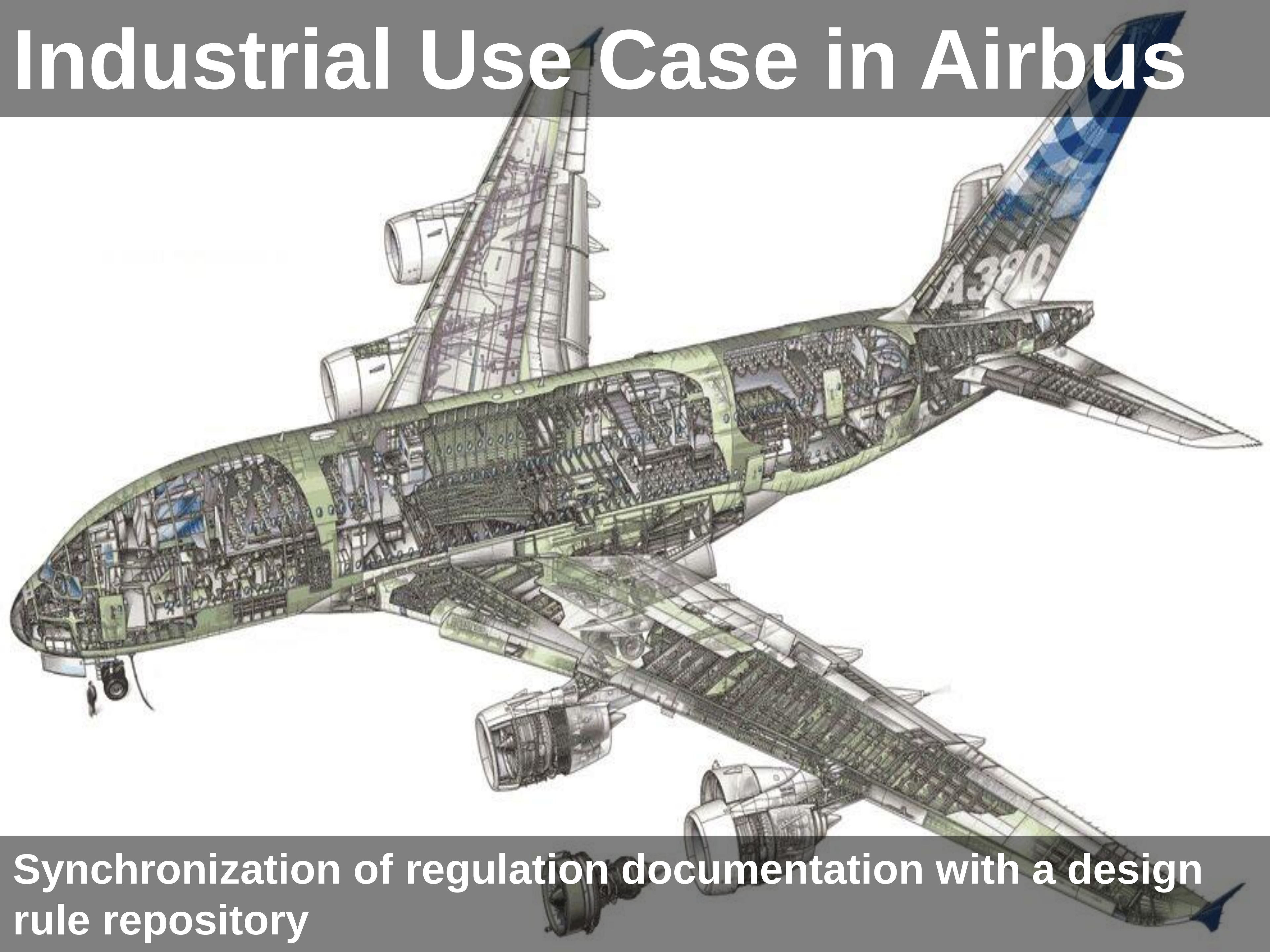
ITEA 3

Industrialization Triangle in ModelWriter

Open Source Software

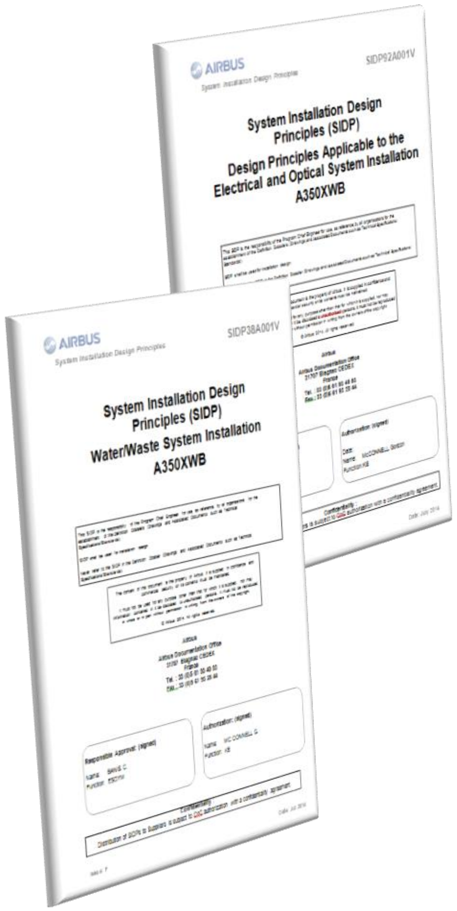


Industrial Use Case in Airbus



Synchronization of regulation documentation with a design rule repository

SIDP: System Installation Design Principles



AIRBUS

System Installation Design Principles

SIDP92A001V

System Installation Design Principles (SIDP)

Design Principles Applicable to the Electrical and Optical System Installation

A350XWB

AIRBUS

System Installation Design Principles

SIDP38A001V

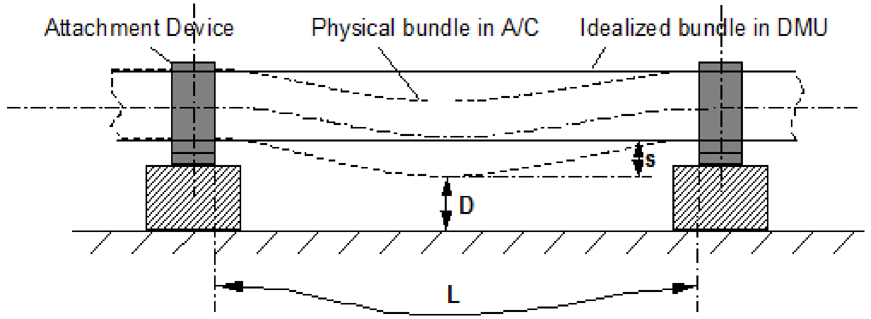
System Installation Design Principles (SIDP)

Water/Waste System Installation

A350XWB

SIDP92A001V-A-784

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

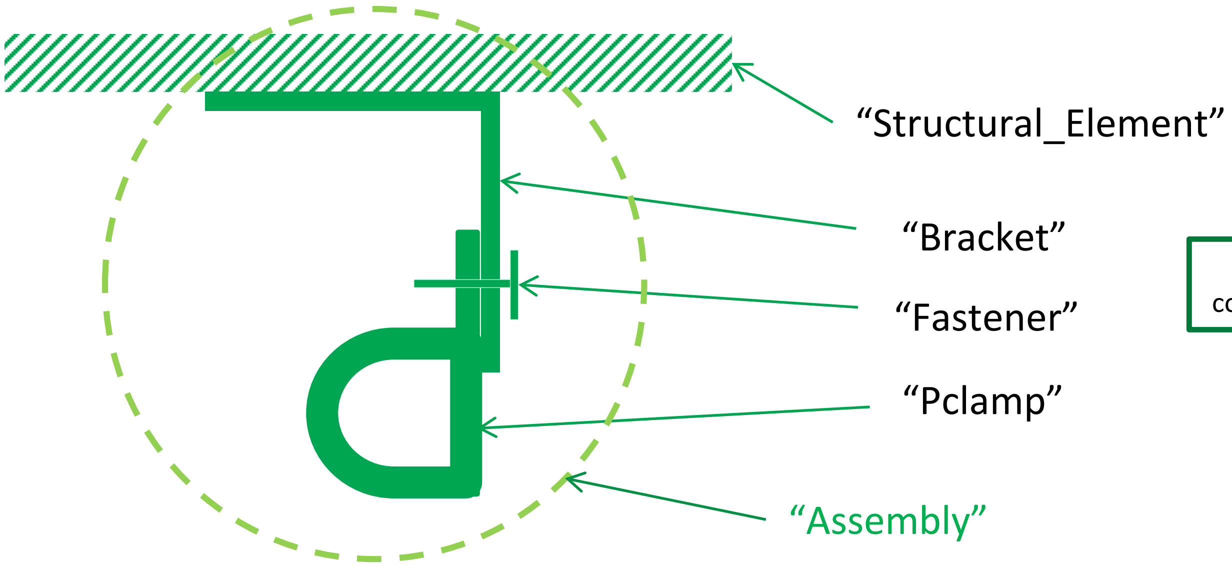


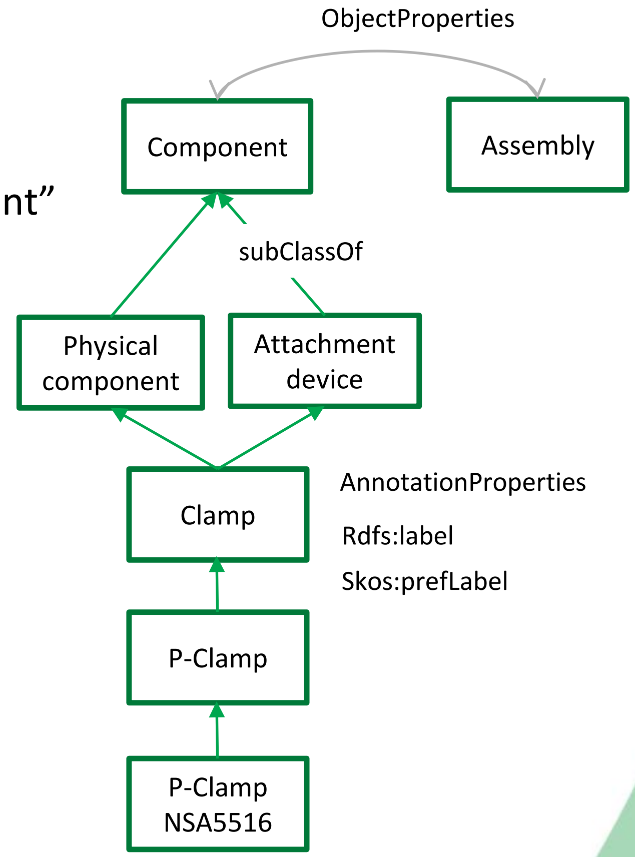
s...Sagging of bundle (real behavior of physical bundle in A/C due to gravity, ageing, etc.)
D...Required Distance
L...Actual length of a bundle segment between two Attachment Points (as designed in DMU)

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.

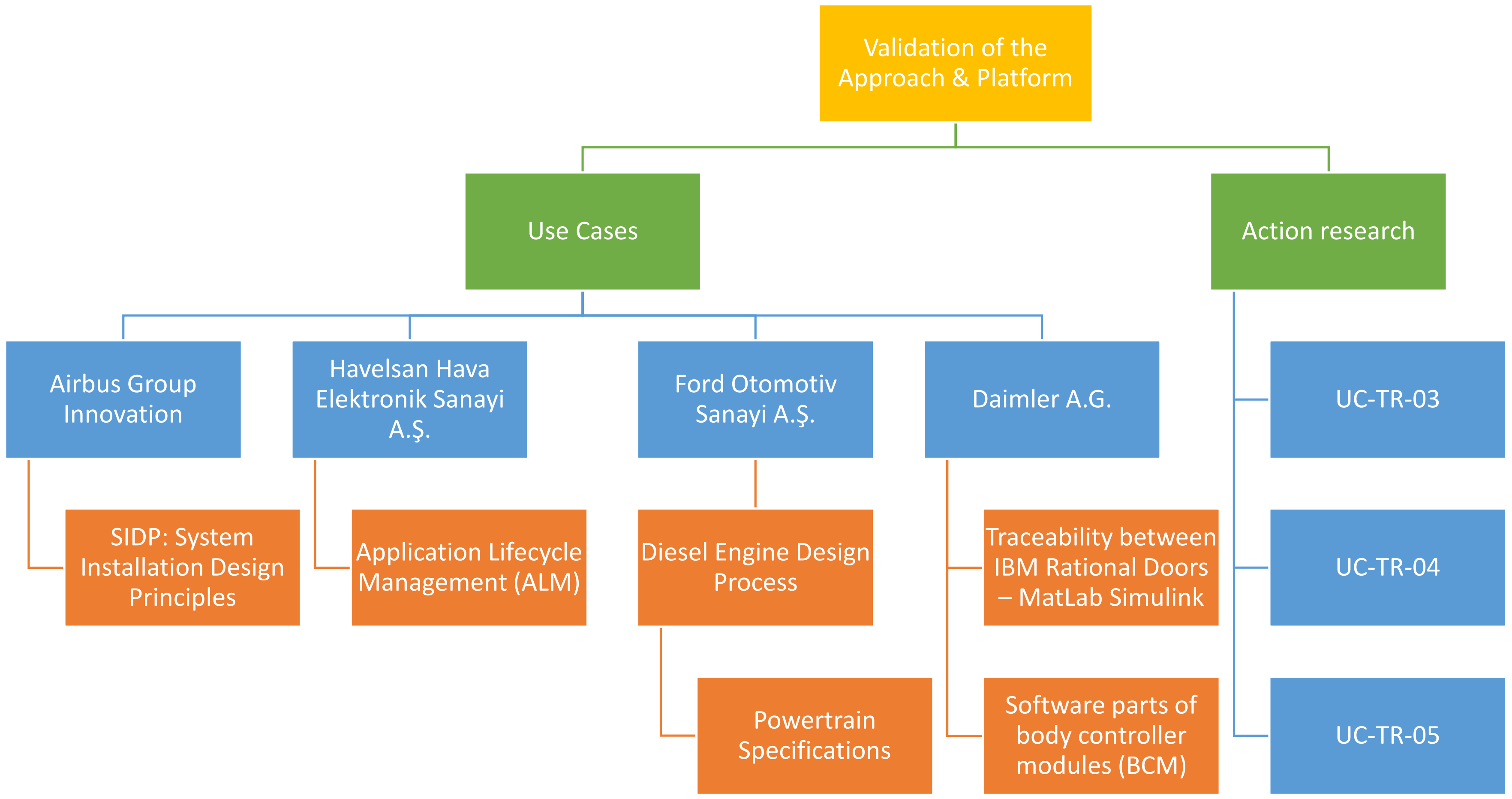
Component classes taxonomy





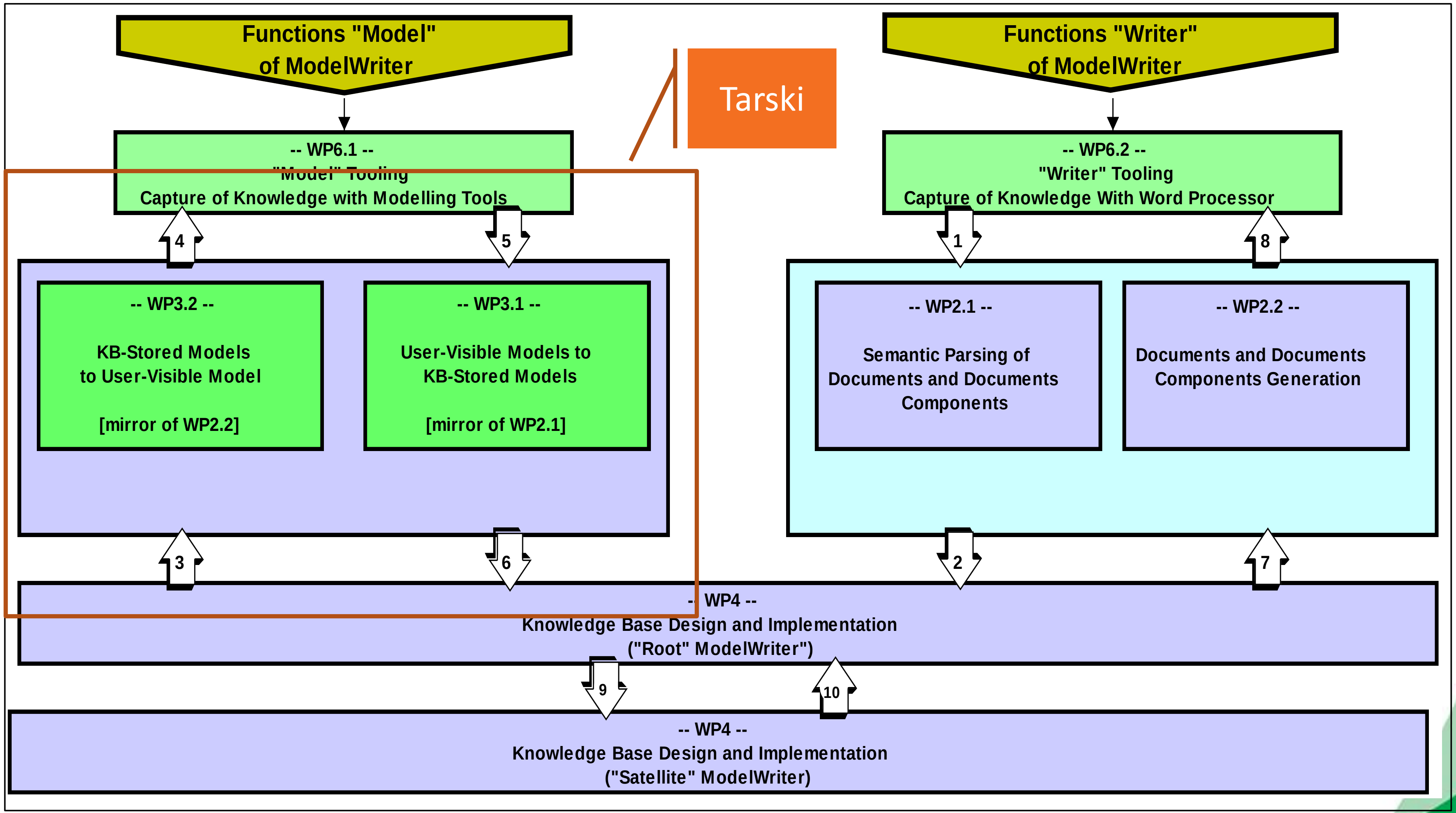
$\underbrace{\text{"P-clamp}}_{\text{"Physical component"}}\ \underbrace{\text{NSA5516}}_{\text{"Standard reference"}}$ can be fixed on X with Y"

Industrial Use Cases


Validation of the Approach & Platform
Use Cases
Action research
Airbus Group Innovation
Havelsan Hava Elektronik Sanayi A.Ş.
Ford Otomotiv Sanayi A.Ş.
Daimler A.G.
UC-TR-03
SIDP: System Installation Design Principles
Application Lifecycle Management (ALM)
Diesel Engine Design Process
Traceability between IBM Rational Doors – MatLab Simulink
UC-TR-04
Powertrain Specifications
Software parts of body controller modules (BCM)
UC-TR-05

Technological components & interactions

Colloboration by WP interactions


Functions "Model" of ModelWriter
Functions "Writer" of ModelWriter
Tarski
-- WP6.1 -- "Model" Tooling Capture of Knowledge with Modelling Tools
-- WP6.2 -- "Writer" Tooling Capture of Knowledge With Word Processor
-- WP3.2 -- KB-Stored Models to User-Visible Model [mirror of WP2.2]
-- WP3.1 -- User-Visible Models to KB-Stored Models [mirror of WP2.1]
-- WP2.1 -- Semantic Parsing of Documents and Documents Components
-- WP2.2 -- Documents and Documents Components Generation
-- WP4 -- Knowledge Base Design and Implementation ("Root" ModelWriter")
-- WP4 -- Knowledge Base Design and Implementation ("Satellite" ModelWriter)
1
2
3
4
5
6
7
8
9
10

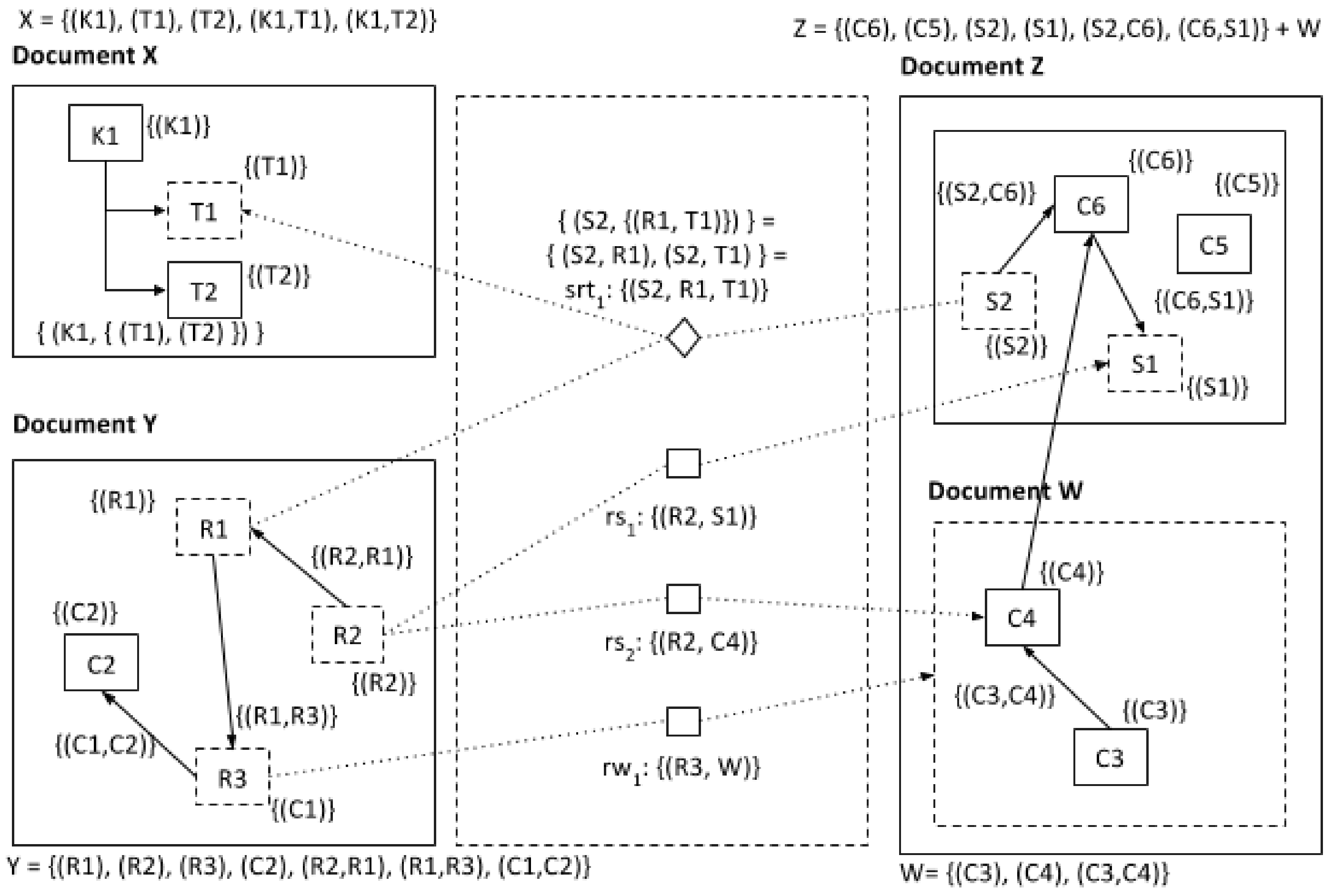
Tarski: A Platform for Automated Analysis of Dynamically Configurable Semantics of Traceability

Ferhat Erata[1,2] and Bedir Tekinerdogan[2]

[1] *UNIT Information Technologies R&D Ltd*

[2] *Information Technology Group, Wageningen University*

Tarski: A Platform for Automated Analysis of Traceability using Constraint Solvers



Univ of Traceability = {(T1), (S2), (S1), (R1), (R2), (R3), (R2,S1), (R3,W), (S2,T1,R1), (W)}

Formalization = `rs: R ->(S + C)` **and** `rw: R -> W` **and** `srt: S -> T -> R`

Challenges of Traceability in Industry

Semantically meaningful traceability

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

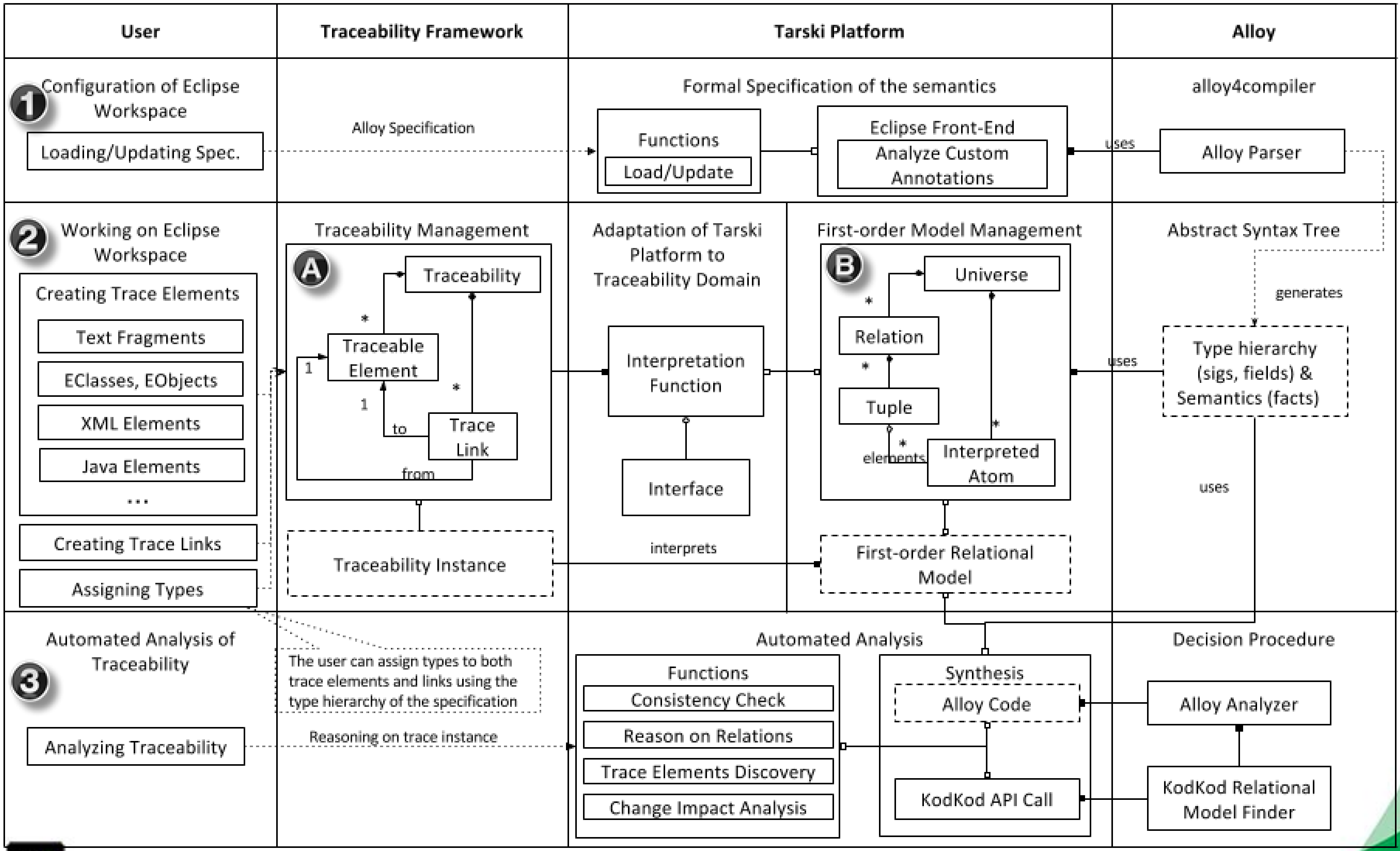
Configurability of traceability (possibly dynamically)

- the semantics of traceability elements 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.

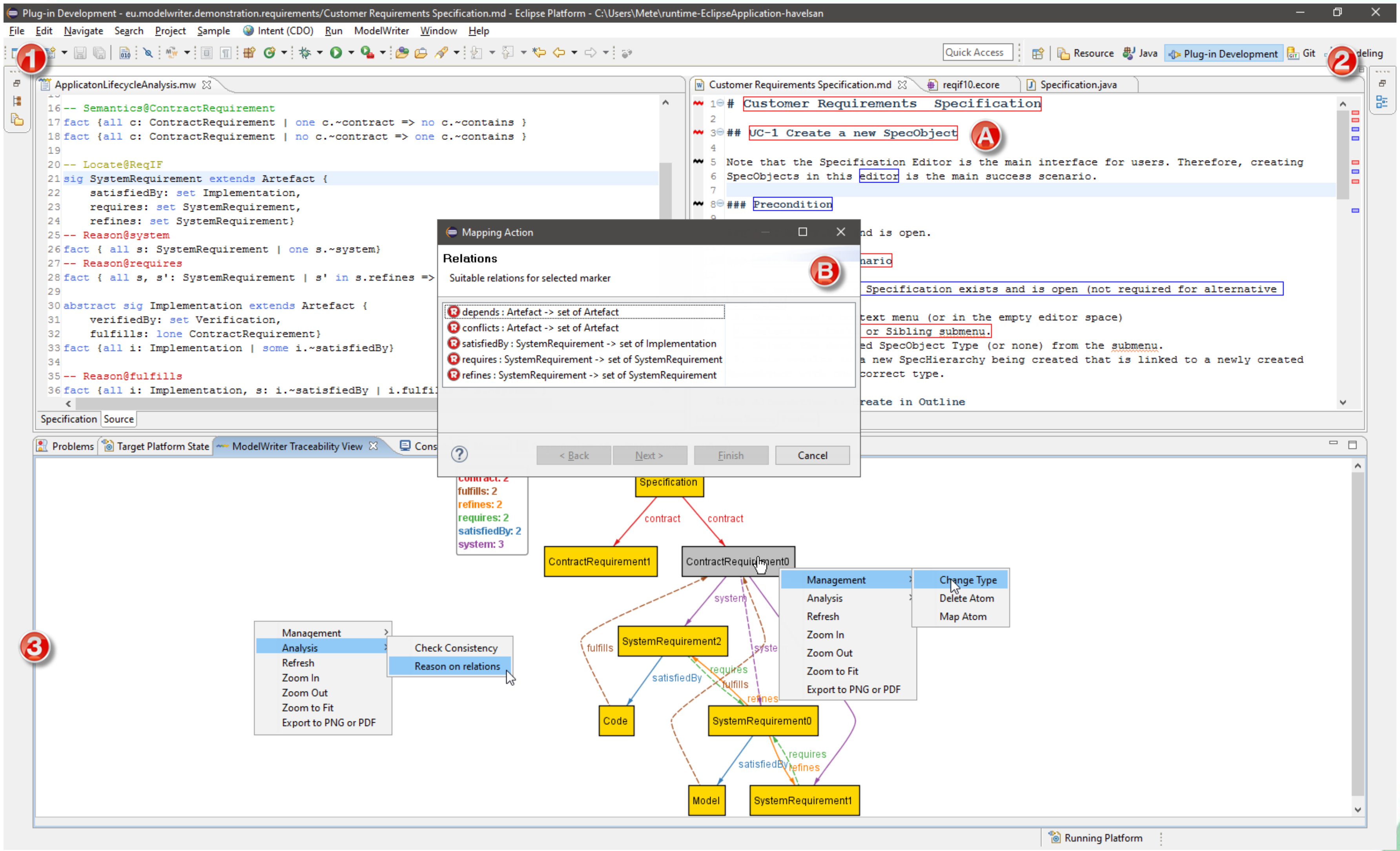
Several industries demands formal proofs of traceability

Consistency checking and repairing broken trace links

Tarski Approach



Tarski Approach



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

ApplicatonLifecycleAnalysis.mw

```
-- Semantics@ContractRequirement
fact {all c: ContractRequirement | one c.~contract => no c.~contains }
fact {all c: ContractRequirement | no c.~contract => one c.~contains }

-- Locate@ReqIF
sig SystemRequirement extends Artefact {
    satisfiedBy: set Implementation,
    requires: set SystemRequirement,
    refines: set SystemRequirement}
-- Reason@system
fact { all s: SystemRequirement | one s.~system}
-- Reason@requires
fact { all s, s': SystemRequirement | s' in s.refines =>

abstract sig Implementation extends Artefact {
    verifiedBy: set Verification,
    fulfills: lone ContractRequirement}
fact {all i: Implementation | some i.~satisfiedBy}

-- Reason@fulfills
fact {all i: Implementation, s: i.~satisfiedBy | i.fulfi
```

Customer Requirements Specification.md

```
# Customer Requirements Specification

## UC-1 Create a new SpecObject

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

### Precondition
nd is open.
nario
Specification exists and is open (not required for alternative
text menu (or in the empty editor space)
 or Sibling submenu.
ed SpecObject Type (or none) from the submenu.
a new SpecHierarchy being created that is linked to a newly created
correct type.

reate in Outline
```

Mapping Action

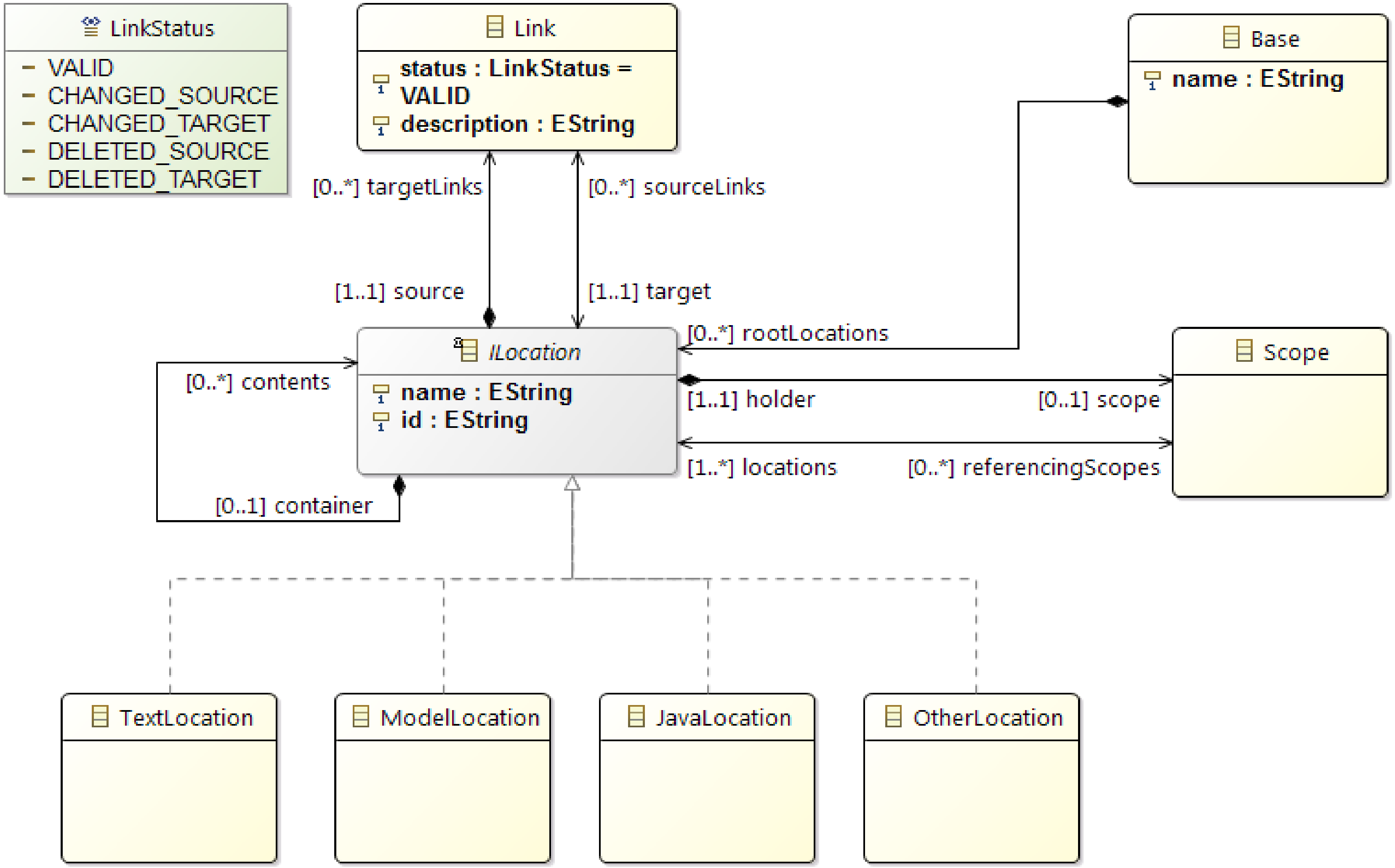
Relations

Suitable relations for selected marker

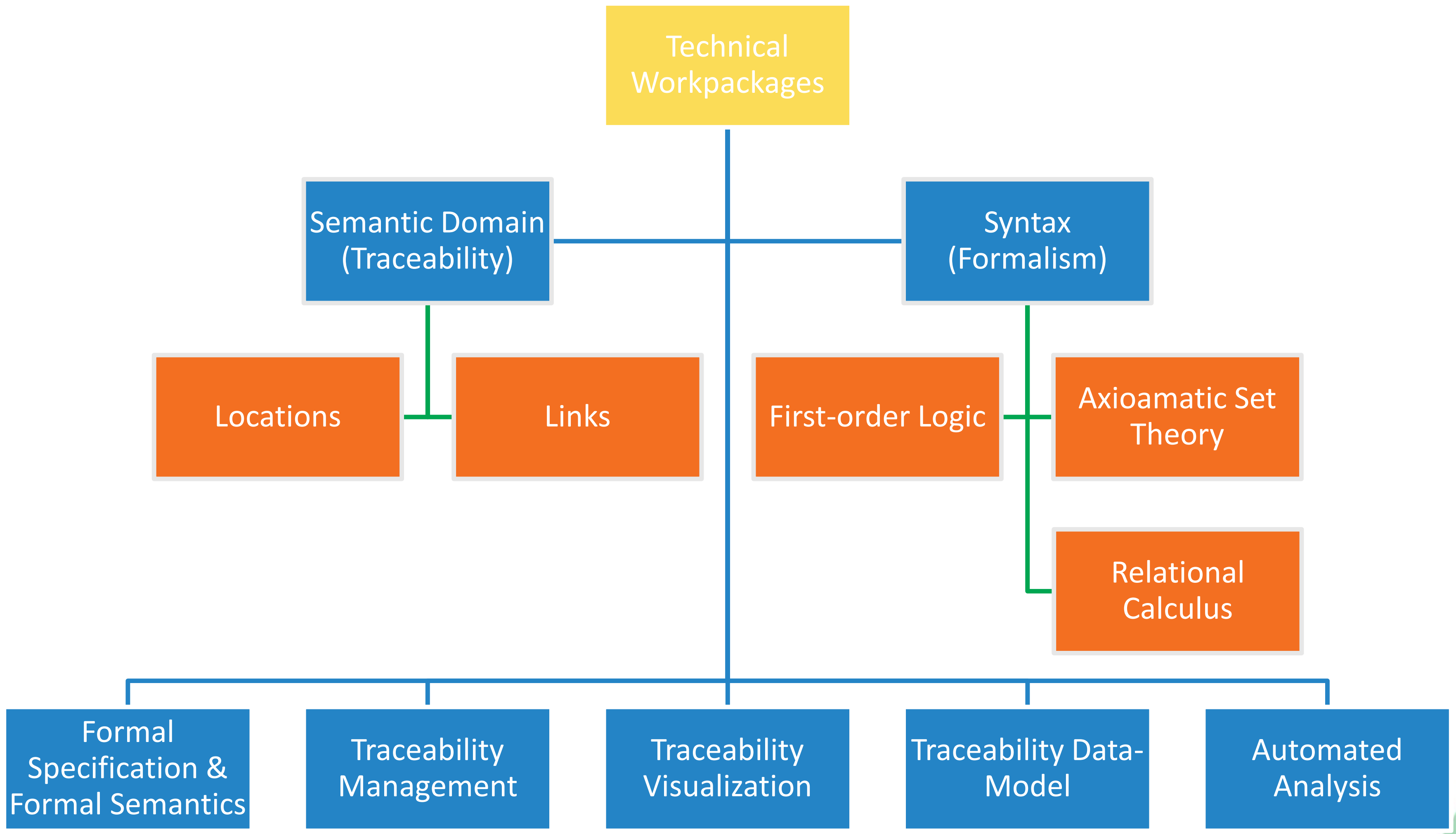
- depends : Artefact -> set of Artefact
- conflicts : Artefact -> set of Artefact
- satisfiedBy : SystemRequirement -> set of Implementation
- requires : SystemRequirement -> set of SystemRequirement
- refines : SystemRequirement -> set of SystemRequirement

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

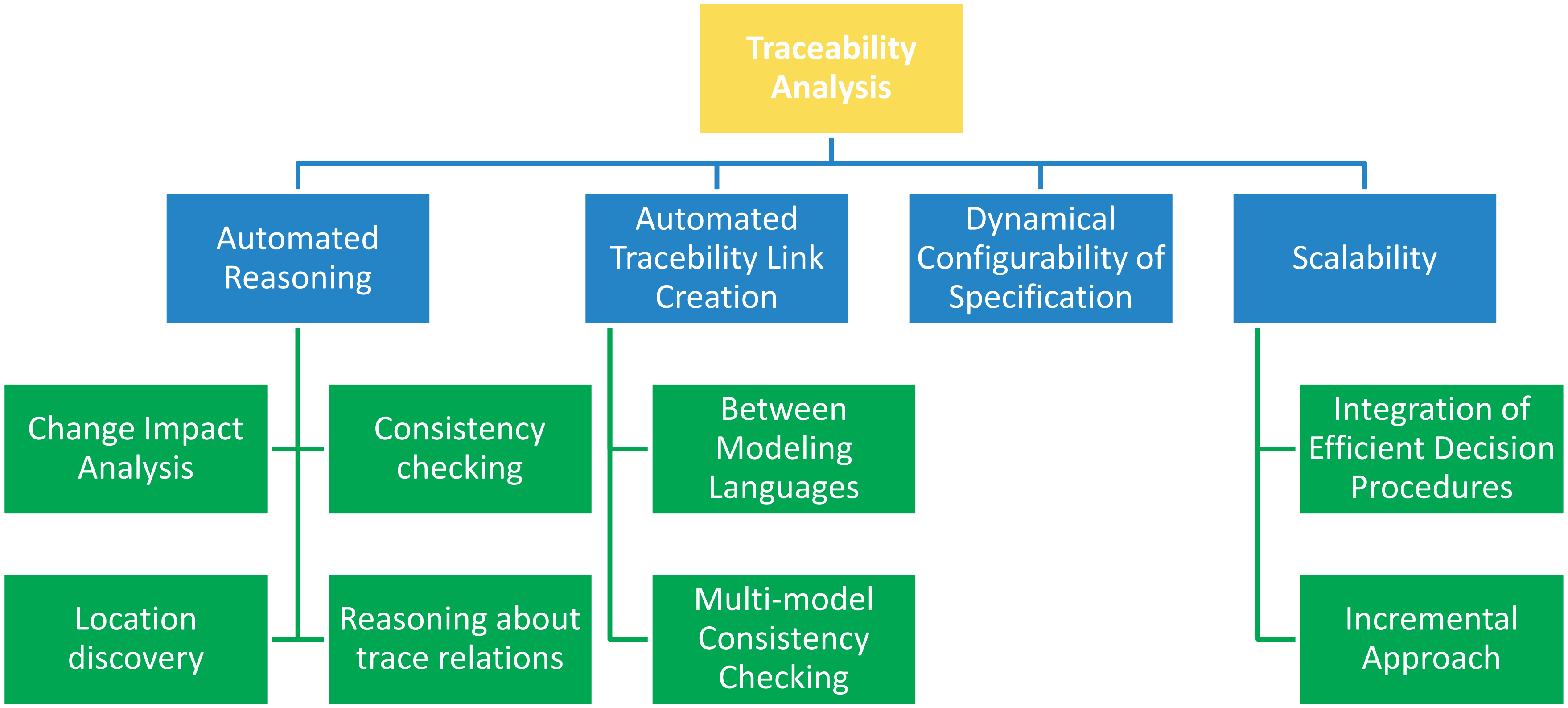
Tarski
Traceability Domain (designed by OBEO)



Overview of Technical Contributions @Tarski

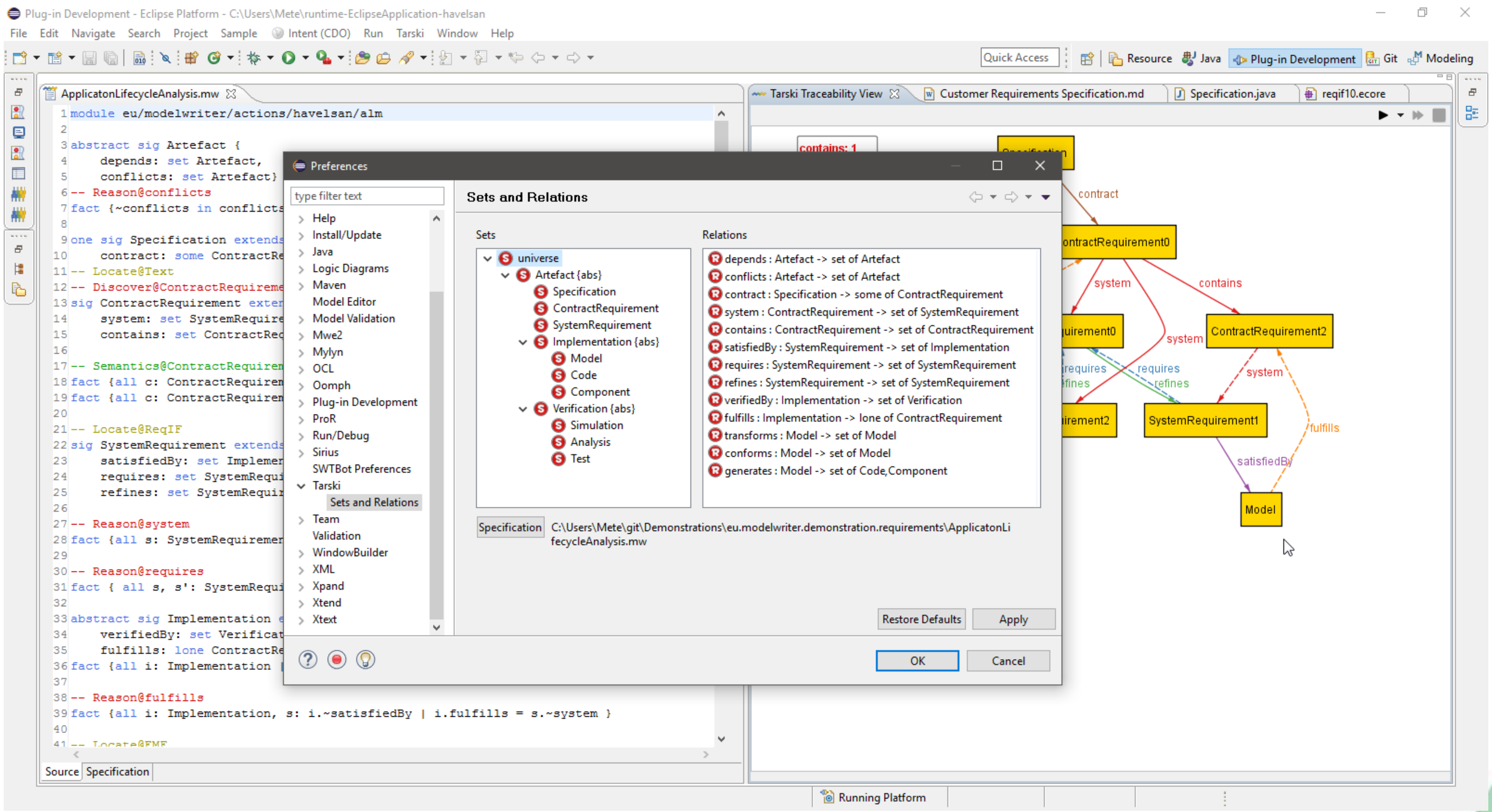


Traceability Analysis


Traceability Analysis
Automated Reasoning
Automated Tracebility Link Creation
Dynamical Configurability of Specification
Scalability
Change Impact Analysis
Consistency checking
Location discovery
Reasoning about trace relations
Between Modeling Languages
Multi-model Consistency Checking
Integration of Efficient Decision Procedures
Incremental Approach

Types/Component Ontology derived from the specification

EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY



```
module eu/modelwriter/actions/havelsan/alm

abstract sig Artefact {
    depends: set Artefact,
    conflicts: set Artefact}
-- Reason@conflicts
fact {~conflicts in conflicts

one sig Specification extends
    contract: some ContractRe
-- Locate@Text
-- Discover@ContractRequireme
sig ContractRequirement exter
    system: set SystemRequire
    contains: set ContractReq

-- Semantics@ContractRequiren
fact {all c: ContractRequiren
fact {all c: ContractRequiren

-- Locate@ReqIF
sig SystemRequirement extends
    satisfiedBy: set Implemer
    requires: set SystemRequi
    refines: set SystemRequir

-- Reason@system
fact {all s: SystemRequiremer

-- Reason@requires
fact { all s, s': SystemRequi

abstract sig Implementation e
    verifiedBy: set Verificat
    fulfills: lone ContractRe
fact {all i: Implementation |

-- Reason@fulfills
fact {all i: Implementation, s: i.~satisfiedBy | i.fulfills = s.~system }

-- Locate@EMF
```

Sets and Relations

Sets

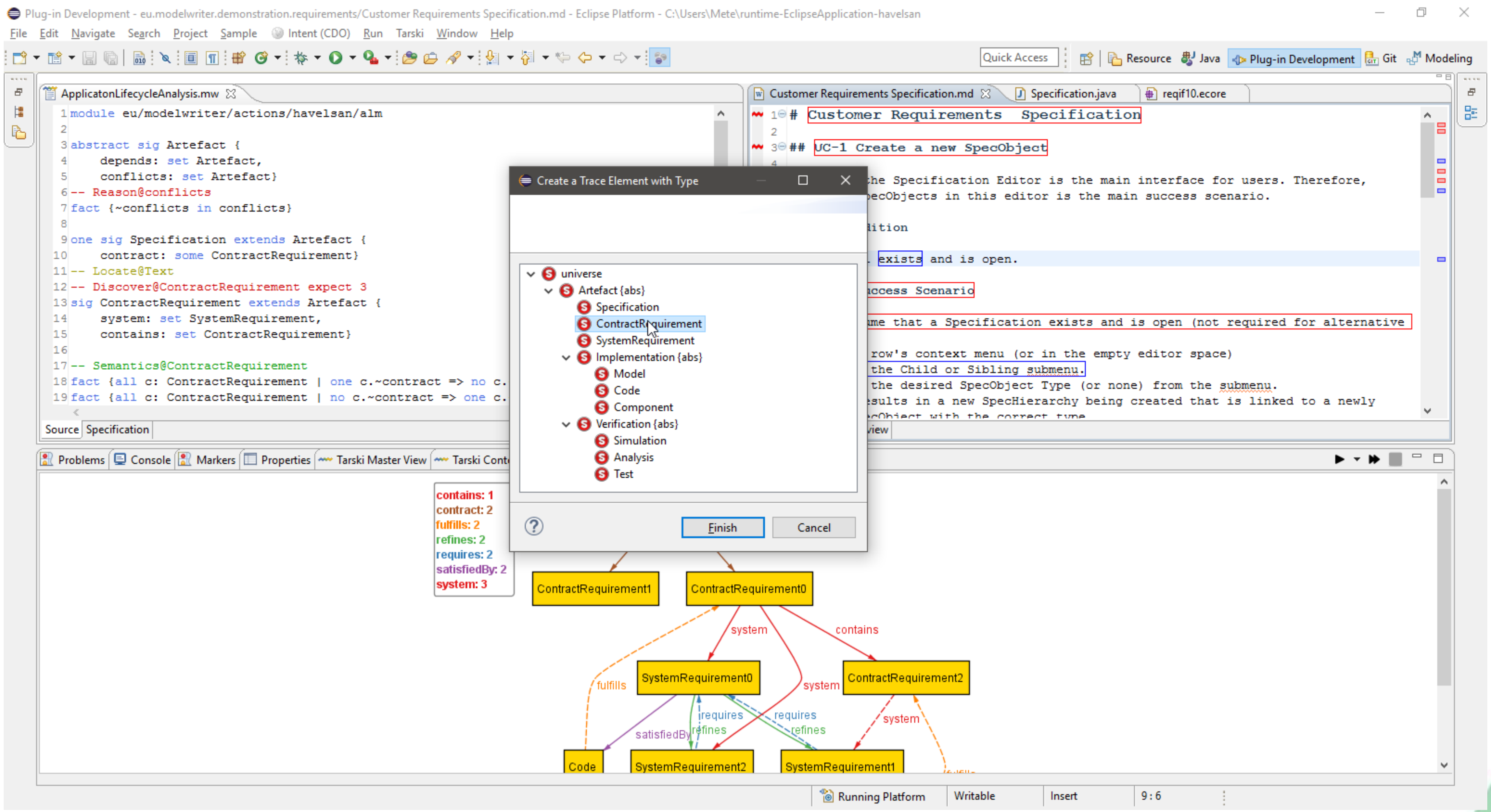
- universe
  - Artefact {abs}
    - Specification
    - ContractRequirement
    - SystemRequirement
    - Implementation {abs}
      - Model
      - Code
      - Component
    - Verification {abs}
      - Simulation
      - Analysis
      - Test

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\Mete\git\Demonstrations\eu.modelwriter.demonstration.requirements\ApplicatonLifecycleAnalysis.mw

Assigning Unary Relations to a Traceable Elements


Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - 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
ApplicatonLifecycleAnalysis.mw
Customer Requirements Specification.md
Specification.java
reqif10.ecore
Source
Specification
Problems
Console
Markers
Properties
Tarski Master View
Tarski Cont
Running Platform
Writable
Insert
9 : 6

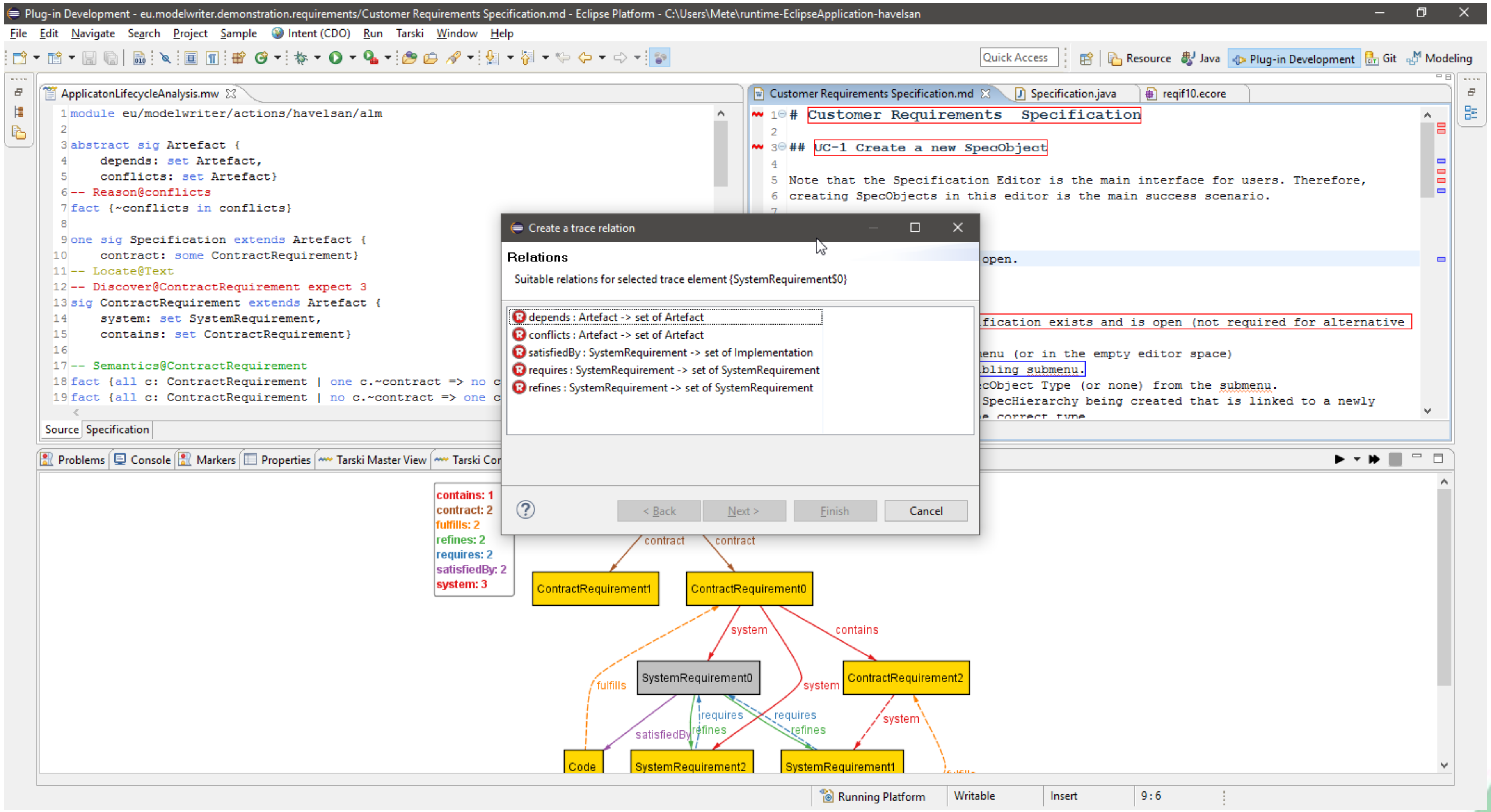
```
module eu/modelwriter/actions/havelsan/alm

abstract sig Artefact {
    depends: set Artefact,
    conflicts: set Artefact}
-- Reason@conflicts
fact {~conflicts in conflicts}

one sig Specification extends Artefact {
    contract: some ContractRequirement}
-- Locate@Text
-- Discover@ContractRequirement expect 3
sig ContractRequirement extends Artefact {
    system: set SystemRequirement,
    contains: set ContractRequirement}

-- Semantics@ContractRequirement
fact {all c: ContractRequirement | one c.~contract => no c.
fact {all c: ContractRequirement | no c.~contract => one c.
```

```
# Customer Requirements  Specification

## UC-1 Create a new SpecObject

he Specification Editor is the main interface for users. Therefore,
ecObjects in this editor is the main success scenario.

ition

 exists and is open.

ccess Scenario

me that a Specification exists and is open (not required for alternative

 row's context menu (or in the empty editor space)
 the Child or Sibling submenu.
 the desired SpecObject Type (or none) from the submenu.
sults in a new SpecHierarchy being created that is linked to a newly
```

Create a Trace Element with Type

- universe
  - Artefact {abs}
    - Specification
    - ContractRequirement
    - SystemRequirement
    - Implementation {abs}
      - Model
      - Code
      - Component
    - Verification {abs}
      - Simulation
      - Analysis
      - Test

Assigning Binary Relations to a Trace Link



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

ApplicatonLifecycleAnalysis.mw

```
module eu/modelwriter/actions/havelsan/alm

abstract sig Artefact {
    depends: set Artefact,
    conflicts: set Artefact}
-- Reason@conflicts
fact {~conflicts in conflicts}

one sig Specification extends Artefact {
    contract: some ContractRequirement}
-- Locate@Text
-- Discover@ContractRequirement expect 3
sig ContractRequirement extends Artefact {
    system: set SystemRequirement,
    contains: set ContractRequirement}

-- Semantics@ContractRequirement
fact {all c: ContractRequirement | one c.~contract => no c
fact {all c: ContractRequirement | no c.~contract => one c
```

Customer Requirements Specification.md

```
# Customer Requirements  Specification

## UC-1 Create a new SpecObject

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

open.

fication exists and is open (not required for alternative

enu (or in the empty editor space)
bling submenu.
cObject Type (or none) from the submenu.
SpecHierarchy being created that is linked to a newly
e correct type
```

Create a trace relation

Relations

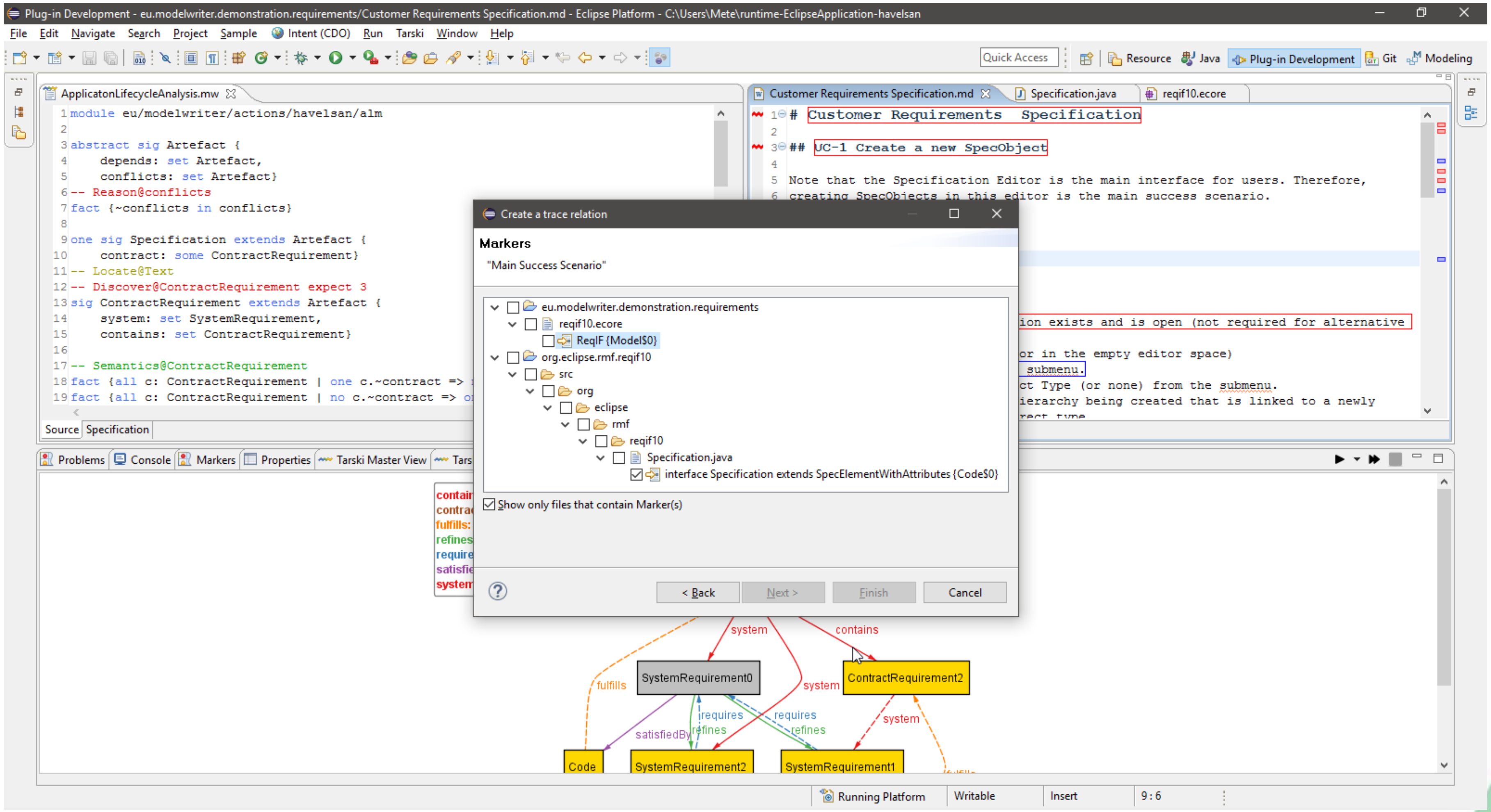
Suitable relations for selected trace element {SystemRequirement$0}

- 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

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

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

EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY



Plug-in Development - eu.modelwriter.demonstration.requirements/Customer Requirements Specification.md - 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
ApplicatonLifecycleAnalysis.mw
Customer Requirements Specification.md
Specification.java
reqif10.ecore
Source Specification
Problems Console Markers Properties Tarski Master View Tars
Running Platform Writable Insert 9 : 6

```
module eu/modelwriter/actions/havelsan/alm

abstract sig Artefact {
    depends: set Artefact,
    conflicts: set Artefact}
-- Reason@conflicts
fact {~conflicts in conflicts}

one sig Specification extends Artefact {
    contract: some ContractRequirement}
-- Locate@Text
-- Discover@ContractRequirement expect 3
sig ContractRequirement extends Artefact {
    system: set SystemRequirement,
    contains: set ContractRequirement}

-- Semantics@ContractRequirement
fact {all c: ContractRequirement | one c.~contract =>
fact {all c: ContractRequirement | no c.~contract => o
```

```
# Customer Requirements Specification

## UC-1 Create a new SpecObject

Note that the Specification Editor is the main interface for users. Therefore,
creating SpecObjects in this editor is the main success scenario.
ion exists and is open (not required for alternative
or in the empty editor space)
 submenu.
ct Type (or none) from the submenu.
ierarchy being created that is linked to a newly
rect type
```

Create a trace relation

Markers

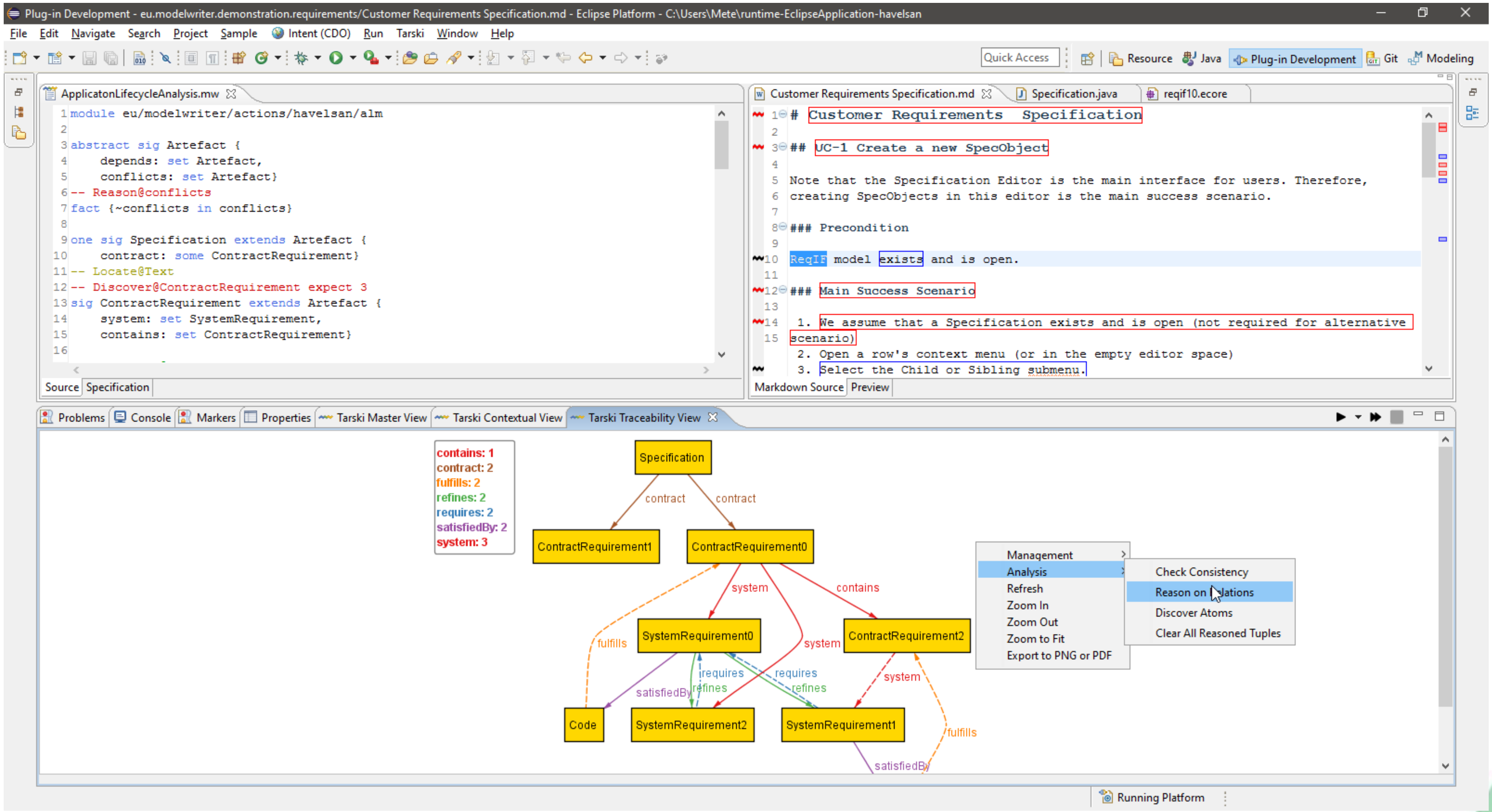
"Main Success Scenario"

- ☐ eu.modelwriter.demonstration.requirements
  - ☐ reqif10.ecore
    - ☐ ReqIF {Model$0}
- ☐ org.eclipse.rmf.reqif10
  - ☐ src
    - ☐ org
      - ☐ eclipse
        - ☐ rmf
          - ☐ reqif10
            - ☐ Specification.java
              - ☑ interface Specification extends SpecElementWithAttributes {Code$0}

☑ Show only files that contain Marker(s)

< Back | Next > | Finish | Cancel

Automated Analysis of Traceability

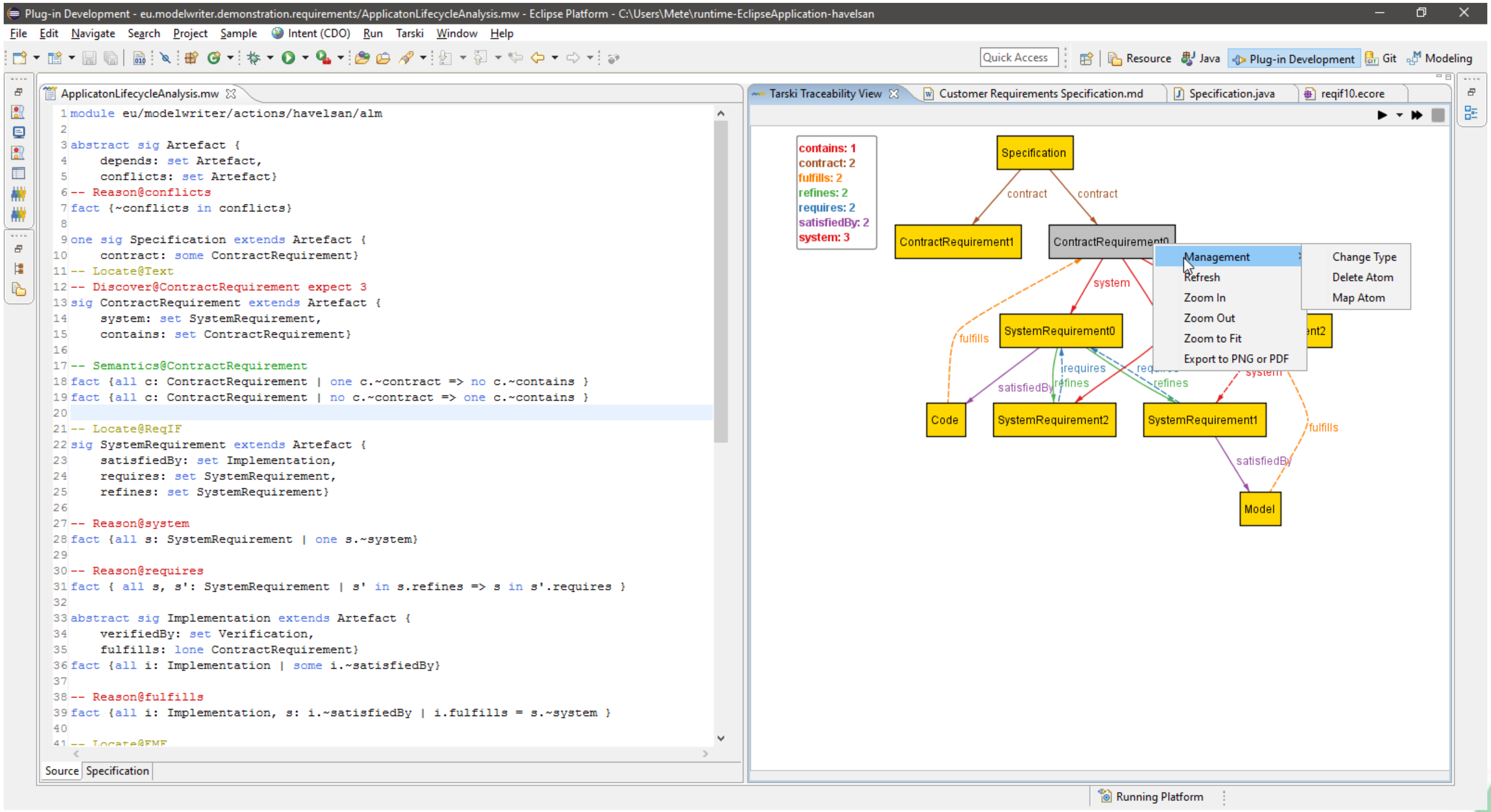


```
module eu/modelwriter/actions/havelsan/alm

abstract sig Artefact {
    depends: set Artefact,
    conflicts: set Artefact}
-- Reason@conflicts
fact {~conflicts in conflicts}

one sig Specification extends Artefact {
    contract: some ContractRequirement}
-- Locate@Text
-- Discover@ContractRequirement expect 3
sig ContractRequirement extends Artefact {
    system: set SystemRequirement,
    contains: set ContractRequirement}
```

```
# Customer Requirements  Specification

## UC-1 Create a new SpecObject

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

### Precondition

ReqIF model exists and is open.

### Main Success Scenario

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

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

Dynamical Configuration & Model Management


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

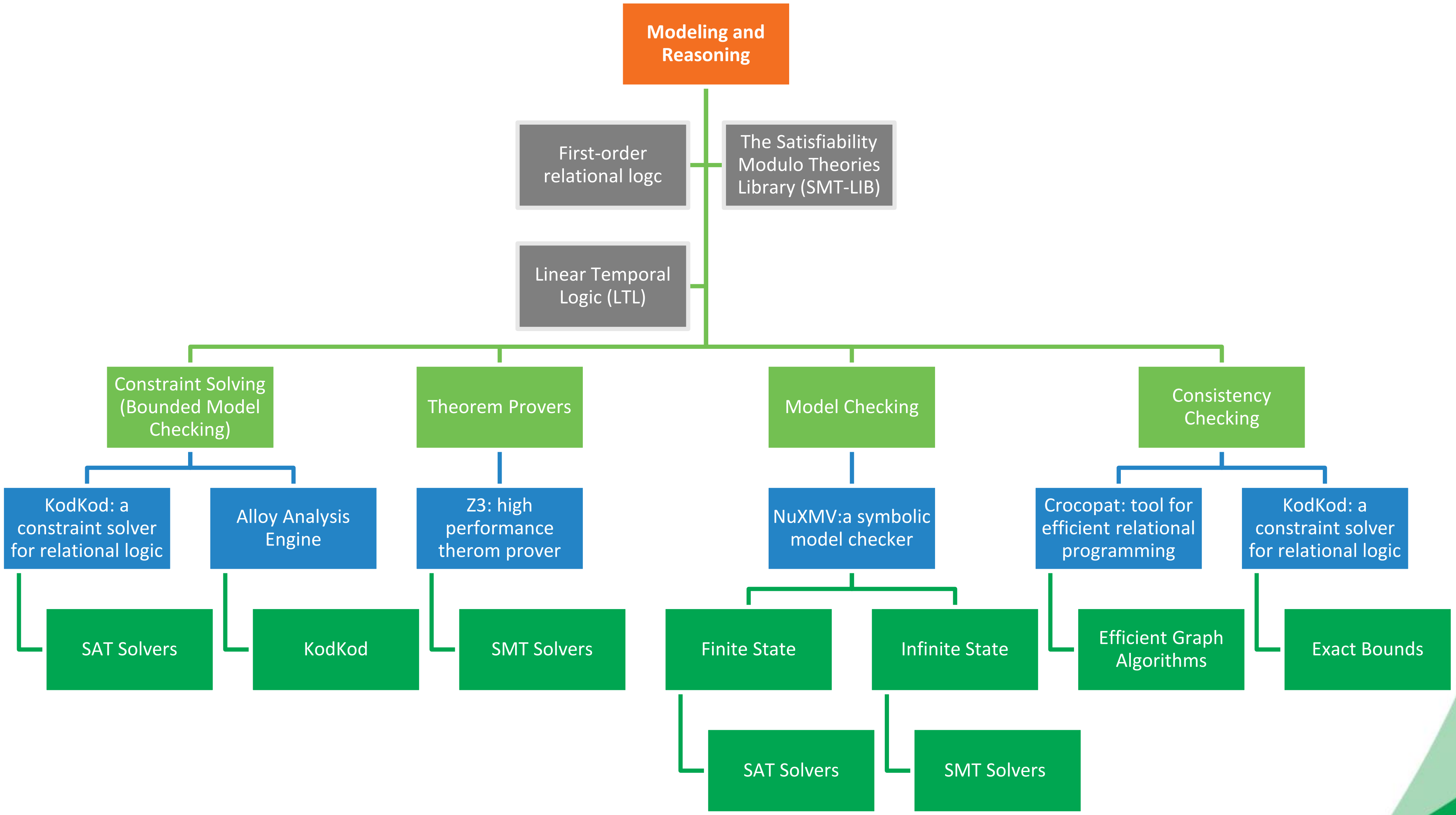
ApplicatonLifecycleAnalysis.mw

```
module eu/modelwriter/actions/havelsan/alm

abstract sig Artefact {
    depends: set Artefact,
    conflicts: set Artefact}
-- Reason@conflicts
fact {~conflicts in conflicts}

one sig Specification extends Artefact {
    contract: some ContractRequirement}
-- Locate@Text
-- Discover@ContractRequirement expect 3
sig ContractRequirement extends Artefact {
    system: set SystemRequirement,
    contains: set ContractRequirement}

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

-- Locate@ReqIF
sig SystemRequirement extends Artefact {
    satisfiedBy: set Implementation,
    requires: set SystemRequirement,
    refines: set SystemRequirement}

-- Reason@system
fact {all s: SystemRequirement | one s.~system}

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

abstract sig Implementation extends Artefact {
    verifiedBy: set Verification,
    fulfills: lone ContractRequirement}
fact {all i: Implementation | some i.~satisfiedBy}

-- Reason@fulfills
fact {all i: Implementation, s: i.~satisfiedBy | i.fulfills = s.~system }

-- Locate@EMF
```

Discussion

- First-order theory of relations to be a solution for traceability in MPM4CPS?
  - Preliminary results shows that the approach works on the synchronization of design rules with design/installation of physical components
- Currently, DPLL(T) solver does not exists for the theory
- What about other theories and combination of theories?
- Should we consider also the temporal behavior of the traceability?

Modeling & Reasoning Approaches



Modeling and Reasoning
First-order relational logc
The Satisfiability Modulo Theories Library (SMT-LIB)
Linear Temporal Logic (LTL)
Constraint Solving (Bounded Model Checking)
Theorem Provers
Model Checking
Consistency Checking
KodKod: a constraint solver for relational logic
Alloy Analysis Engine
Z3: high performance therom prover
NuXMV:a symbolic model checker
Crocopat: tool for efficient relational programming
KodKod: a constraint solver for relational logic
SAT Solvers
KodKod
SMT Solvers
Finite State
Infinite State
Efficient Graph Algorithms
Exact Bounds
SAT Solvers
SMT Solvers