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

Ferhat Erata[1,2] Moharram Challenger[1,4] Bedir Tekinerdogan[1]
Anne Monceaux[3] Eray Tuzun[5] Geylani Kardas[4]

[1]Information Technology Group, Wageningen University, The Netherlands
[2]UNIT Information Technologies R&D Ltd., Izmir, Turkey
[3]System Engineering Platforms, AIRBUS Group Innovations, Toulouse, France
[4]International Computer Institute, Ege University, Izmir, Turkey
[5]Academy Directorate, HAVELSAN Inc., Ankara, Turkey

3rd Workshop on Dependability at Izmir Institute of Technology

Acknowledgements

- Scientific and Technological Research Council of Turkey (TUBITAK), Technology and Innovation Funding Programs Directorate (TEYDEB) under project# 9140014, 9150181
- Minister for the Economy, Industry and Digital Affairs of France, Directorate-General for Enterprise (DGE) under contract# 14293020
- European Cooperation in Science and Technology (COST) Action IC1404 "Multi-Paradigm Modelling for Cyber-Physical Systems"

TÜBİTAK

DIRECTION GÉNÉRALE DES ENTREPRISES

EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

Exploitations

ITEA-ModelWriter: Synchronized Document Engineering Platform

https://itea3.org/project/modelwriter.html

ITEA-ASSUME: Affordable Safe & Secure Mobility Evolution

https://itea3.org/project/assume.html

ITEA 3

EUREKA
innovation across borders

Source codes, datasets and screencasts are available at:

https://github.com/ModelWriter/WP3

Outline

Outline

What is Traceability?

Traceability can be defined as the degree to which a relationship can be established among work products (aka. artefacts) of the development process.

What is case-based or project-based traceability configuration?

Rigorously specification the semantics of traceability elements.

Why is Reasoning about Traceability important?

Richer and precise automated traceability analysis.
Compliance and Certification in automotive and aviation industries.

Challenges of Traceability in Industry

Semantically meaningful traceability

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

Configuration of traceability (possibly dynamically)

- Traceability Semantics is often statically defined.

Challenges of Traceability in Industry

Semantically meaningful traceability

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

Configuration of traceability (possibly dynamically)

- Traceability Semantics is often statically defined.
- The semantics cannot be easily adapted for the needs of different projects.

Challenges of Traceability in Industry

Semantically meaningful traceability

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

Configuration of traceability (possibly dynamically)

- Traceability Semantics is often statically defined.
- The semantics cannot be easily adapted for the needs of different projects.
- Different traceable elements and the relation types exist in industrial settings,

Challenges of Traceability in Industry

Semantically meaningful traceability

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

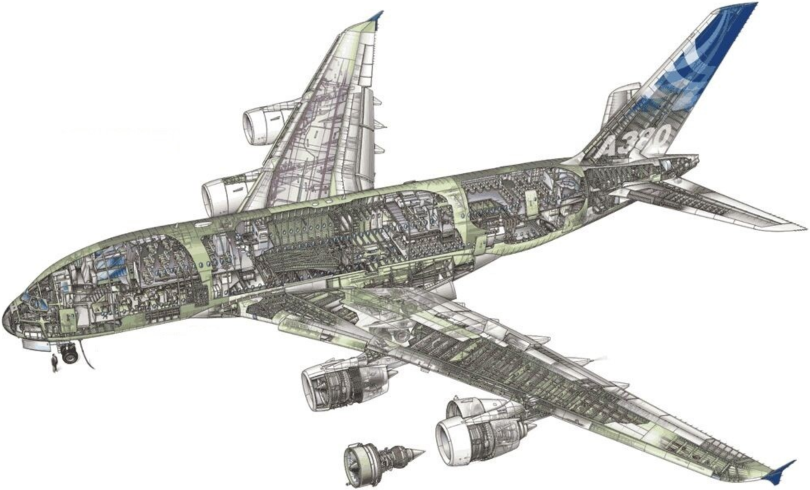
Configuration of traceability (possibly dynamically)

- Traceability Semantics is often statically defined.
- The semantics cannot be easily adapted for the needs of different projects.
- Different traceable elements and the relation types exist in industrial settings,
- Likewise, different traceability analysis scenarios exists. Several industries demands formal proofs of Traceability.

Outline

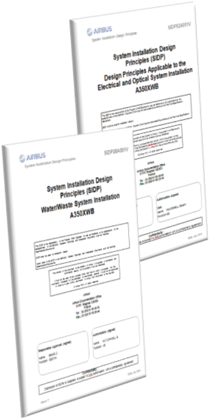
Airbus Group Innovations

System Installation Design Principles



Airbus Group Innovations

System Installation Design Principles


AIRBUS

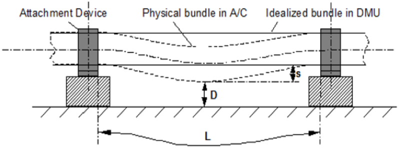
System Installation Design Principles (SIDP)
Design Principles Applicable to the Electrical and Optical System Installation
A350XWB

AIRBUS

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:



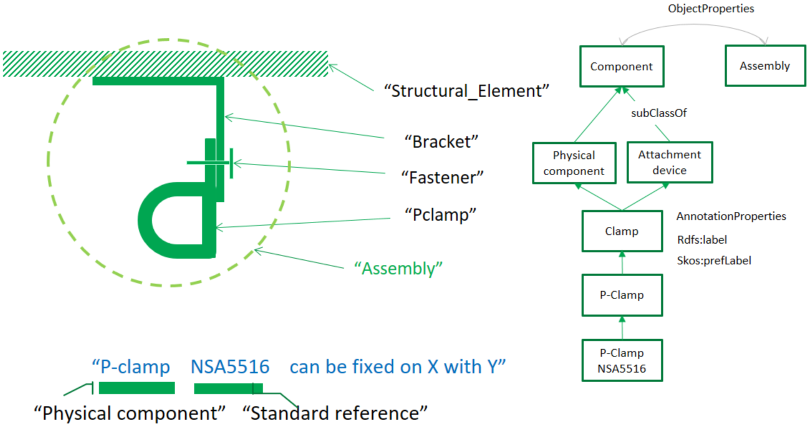
s...Sagging of bundle (real behavior of physical bundle in A/C due to gravity, ageing, etc.)
D...Required Distance
L...Actual length of a bundle segment between two Attachment Points (as designed in DMU)

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.

Airbus Group Innovations

System Installation Design Principles



Havelsan Aerospace Electronics Industry

Application Lifecycle Management

DO-178C

Software Considerations in Airborne Systems and Equipment Certification

Traceability

DO-178 requires a documented connection (called a trace) between the certification artifacts. For example, a Low Level Requirement (LLR) traces up to a High Level Requirement (HLR). A traceability analysis is then used to ensure that each *requirement* is fulfilled by the source code, that each *requirement* is tested, that each line of source code has a purpose (is connected to a requirement), and so forth. Traceability ensures the system is complete.

Traceability Analysis Activities defined in DO-178

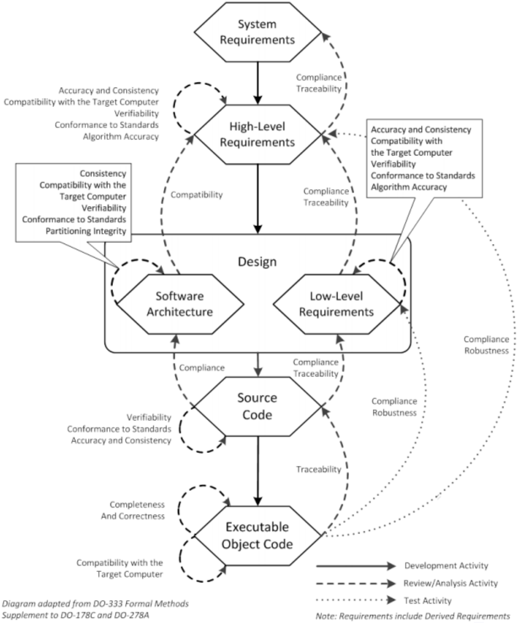
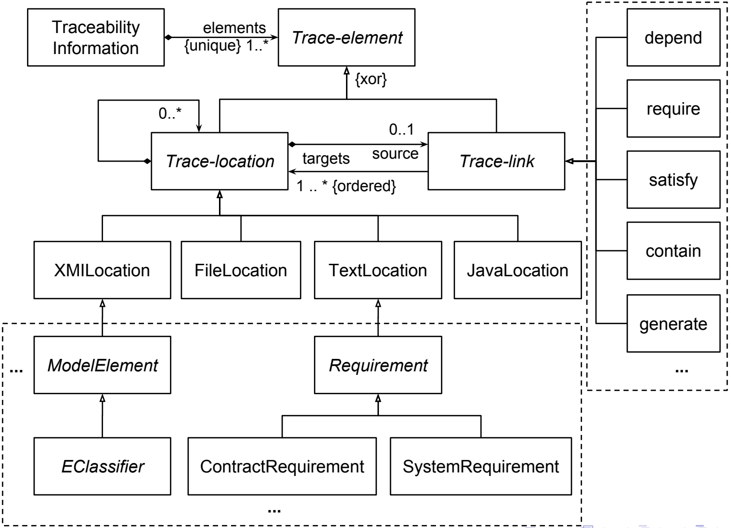


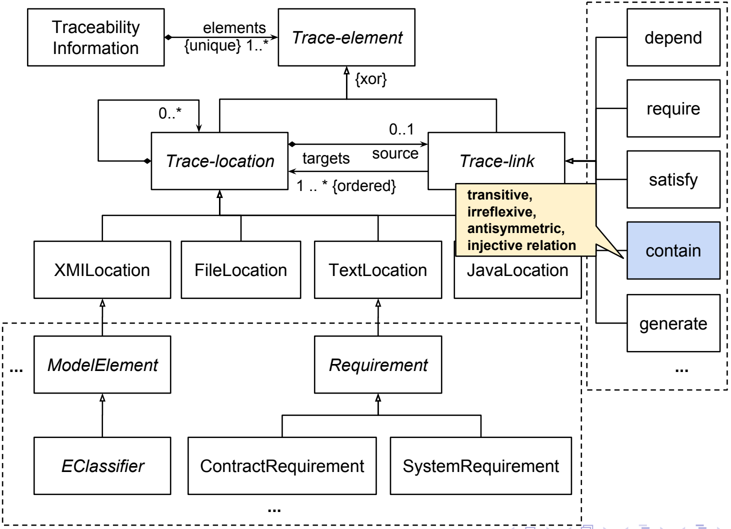
Diagram adapted from DO-333 Formal Methods Supplement to DO-178C and DO-278A

Outline

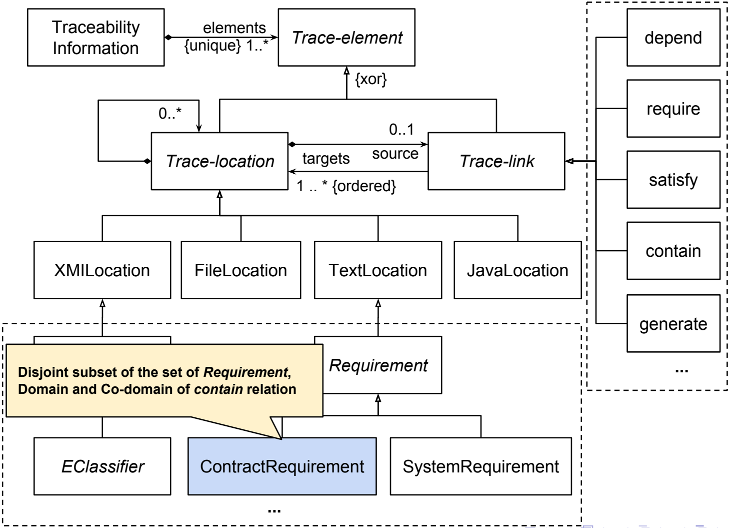
A conceptual model for traceability and its extension



Semantics of *contain relation* (represents decomposition)

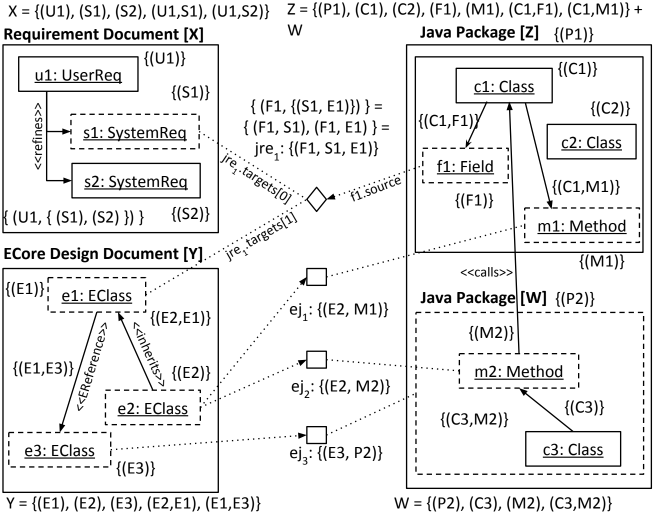


Semantics of *ContractRequirement*



Outline

Fragments of a traceability instance



First-order relational model of the traceability instance

The *universe* of traceability of the current state

$D_T : \{S_1, E_1, E_2, E_3, F_1, M_1, M_2, P_2\}$

The *type signature*

$\Sigma_T : \{R_{EJ} \sqsubseteq E \rightarrow C \sqcup M \sqcup F,\ R_{JRE} \sqsubseteq F \rightarrow S \rightarrow E\}$

The *relational model* under the signature Σ_T

$M_t : \{S = \{\langle S_1 \rangle\}, E = \{\langle E_1 \rangle, \langle E_2 \rangle, \langle E_3 \rangle\}, J = \{\langle F_1 \rangle, \langle M_1 \rangle, \langle M_2 \rangle, \langle P_2 \rangle\}, R_{EJ} = \{\langle E_2, M_1 \rangle, \langle E_2, M_2 \rangle, \langle E_3, P_2 \rangle\}, R_{JRE} = \{\langle F_1, S_1, E_1 \rangle\}\}$

First-order Relational Logic (FOL + Relational Calculus)

. Relational Join and ∼ Transpose

The *dot join* and *transpose* operators ensure a uniform way of navigation between *trace-locations* through *trace-links* in constraints.

∗ ˆ(Reflexive) Transitive Closure

Transitive Closure allows the encoding of common reachability constraints that otherwise could not be expressed in FOL, such as preventing cyclic dependencies between *trace-locations*.

Domain and Range Restrictions

The restriction operators are used to filter relations to a given domain or range.

First-order Relational Logic (FOL + Relational Calculus)

. Relational Join and ~ Transpose

$E.R_{EJ} = \{\langle E_1\rangle, \langle E_2\rangle, \langle E_3\rangle\}.\{\langle E_2, M_1\rangle, \langle E_2, M_2\rangle, \langle E_3, P_2\rangle\}$
$= \{\langle M_1\rangle, \langle M_2\rangle, \langle P_2\rangle\}$
$J.\sim R_{EJ} = \{\langle F_1\rangle, \langle M_1\rangle, \langle M_2\rangle, \langle P_2\rangle\}.\{\langle M_1, E_2\rangle, \langle M_2, E_2\rangle, \langle P_2, E_3\rangle\}$
$= \{\langle E_2\rangle, \langle E_3\rangle\}$

$*$ ^ (Reflexive) Transitive Closure

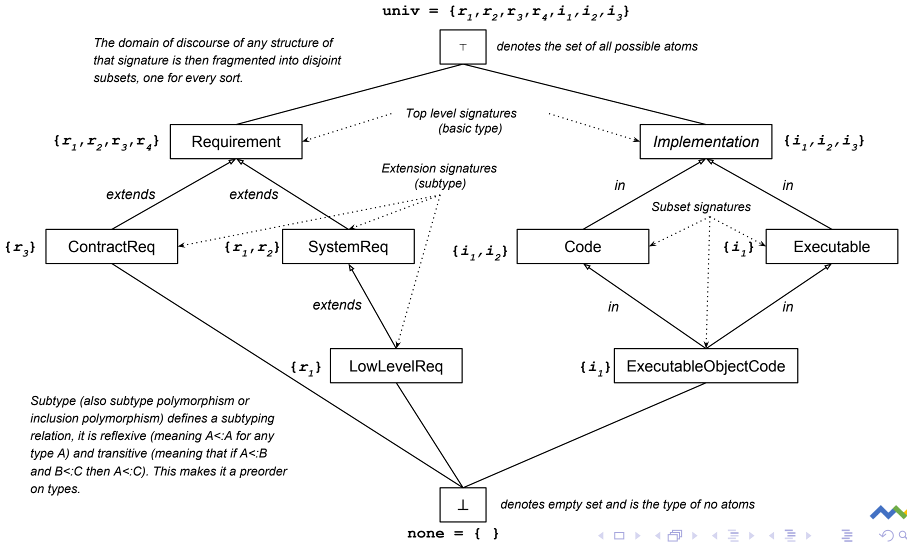
$\hat{}\{\langle M_1, E_1\rangle, \langle E_1, C_1\rangle\} = \{\langle M_1, E_1\rangle, \langle E_1, C_1\rangle, \langle M_1, C_1\rangle\}$

Domain and Range Restrictions

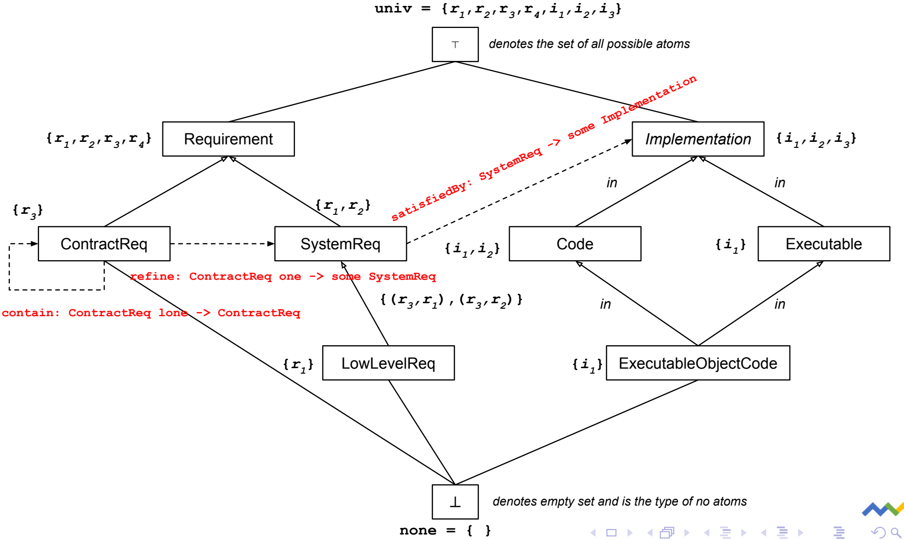
$P <: R_{JE} = \{\langle P_2\rangle\} <: \{\langle M_1, E_2\rangle, \langle M_2, E_2\rangle, \langle P_2, E_3\rangle\} = \{\langle P_2, E_3\rangle\}$

Outline

Basic Type and SubType



Relation Types



Outline

Formal Specification of an example configuration

```
abstract sig Artefact { depends: set Artefact}

-- Locate@File
one sig Specification extends Artefact {
     contract: some ContractRequirement}

-- Locate@Text
sig ContractRequirement extends Artefact {
     system: set SystemRequirement,
     contains: set ContractRequirement}

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

```
abstract sig Implementation extends Artefact {
    fulfills: lone ContractRequirement}

-- Locate@Java
sig Code, Component extends Implementation {}

-- Locate@EMF
sig Model extends Implementation {
    transforms, conforms: set Model,
    generates: set (Code ∪ Component)}

-- Semantics@SystemRequirement.satisfiedBy
fact {∀ i: Implementation | some i.~satisfiedBy}
```

Automated analysis functions over Traceability Model

Consistency Checking

The system checks whether the user model satisfies the specification or not.

Reasoning about Trace-relations

If the model is a partial (incomplete), the platform tries to complete the model with respect to the semantics declared in the specification inferring new trace-relations on the model.

Trace-elements Discovery

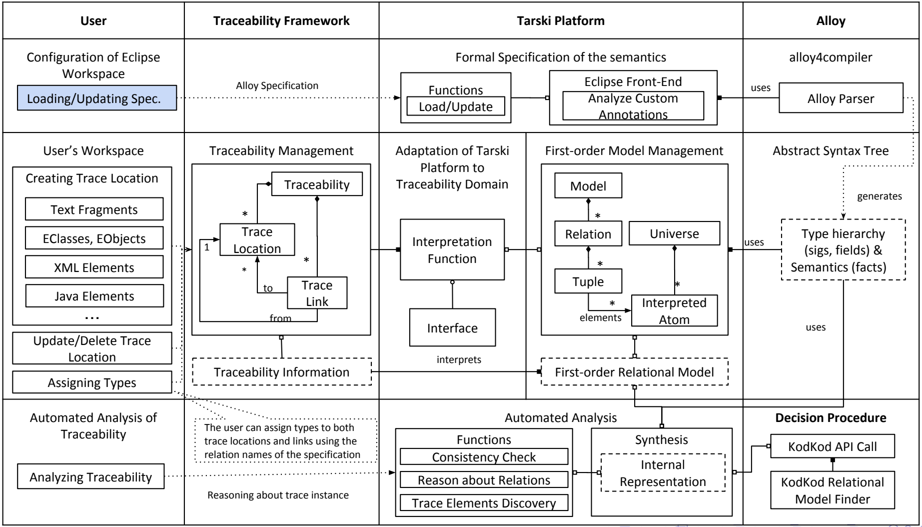
If a de-synchronization occurs on one or more ends of a *trace-link* probably caused by a change such as deletion of a trace-location, we try to repair the broken link based on the specified semantics.

Reasoning about Trace-relations

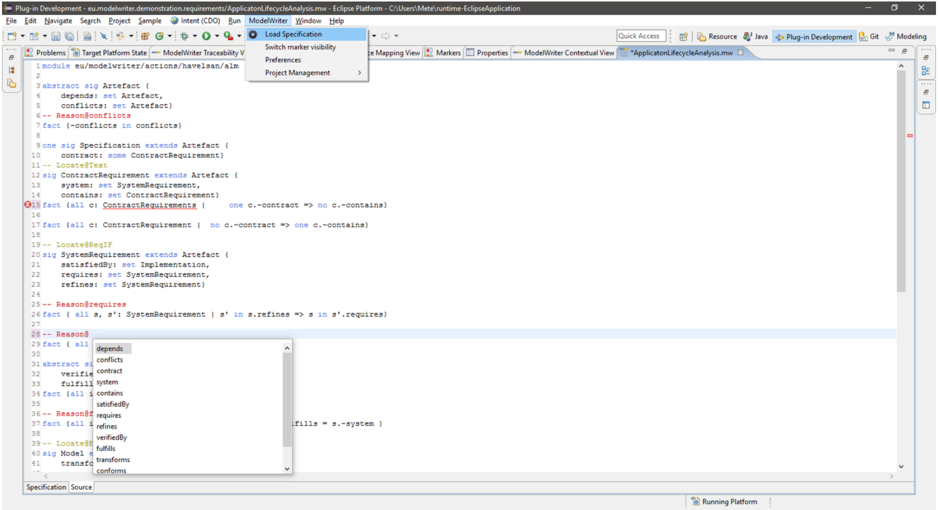
```
-- Reason@ContractRequirement.system
fact {∀ s: SystemRequirement, s': s.*~refines |
     s'.~system = s.~system}

-- Reason@SystemRequirement.requires
fact { ∀ s, s': SystemRequirement |
     s' in s.refines ⟹ s in s'.requires }

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

Outline

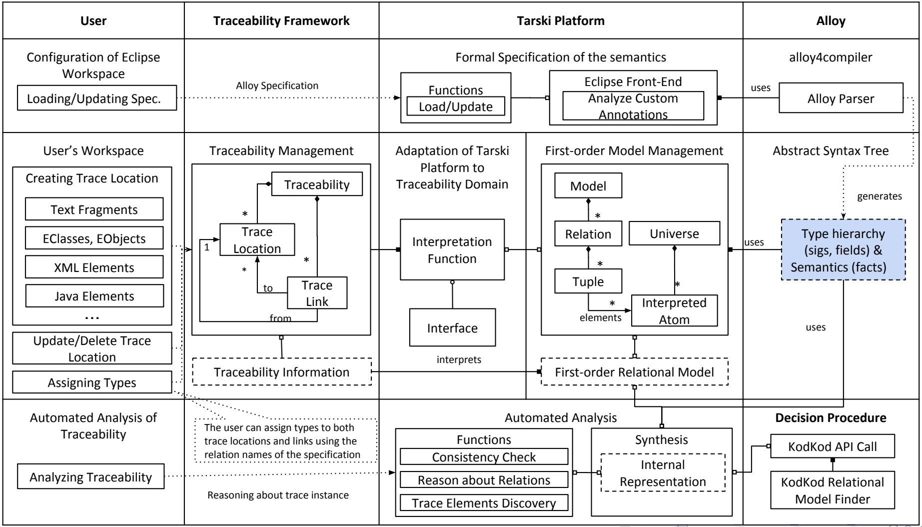
Configuration of User's Workspace



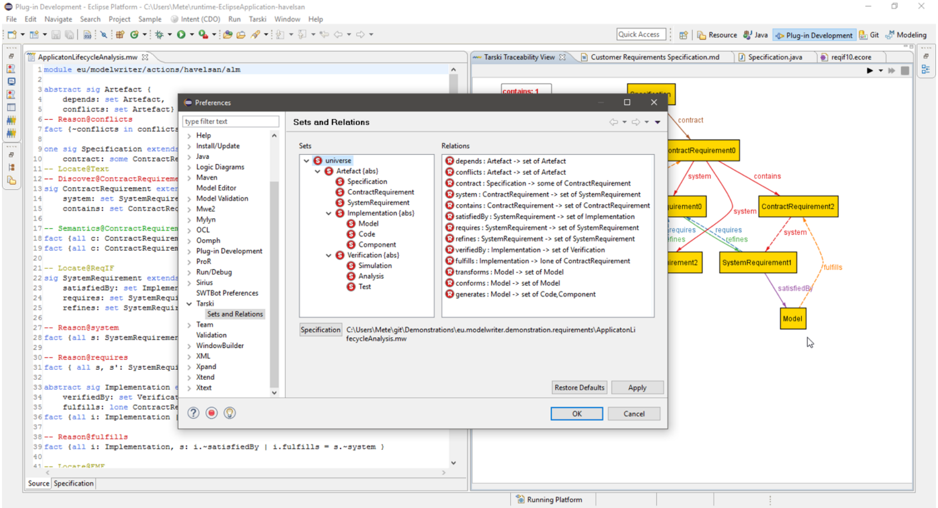
Configuration of User's Workspace



Type Hierarchy from the Specification

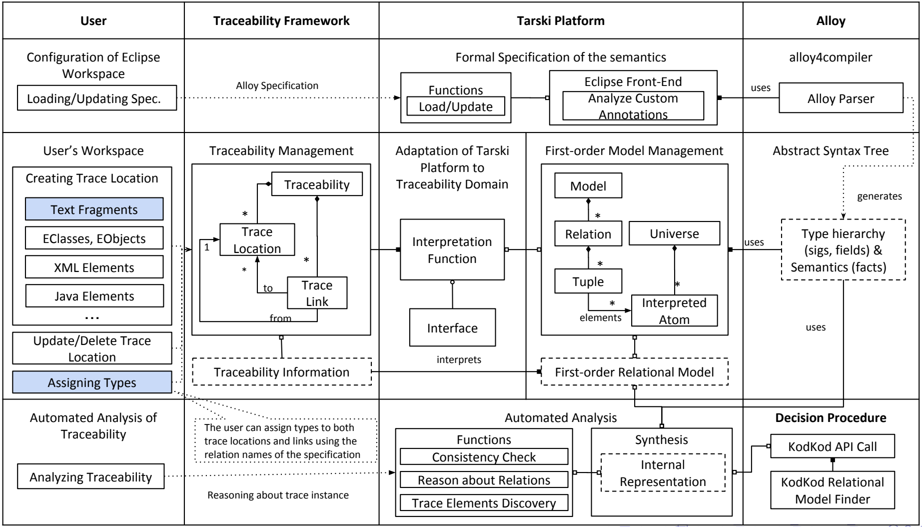


Type Hierarchy from the Specification

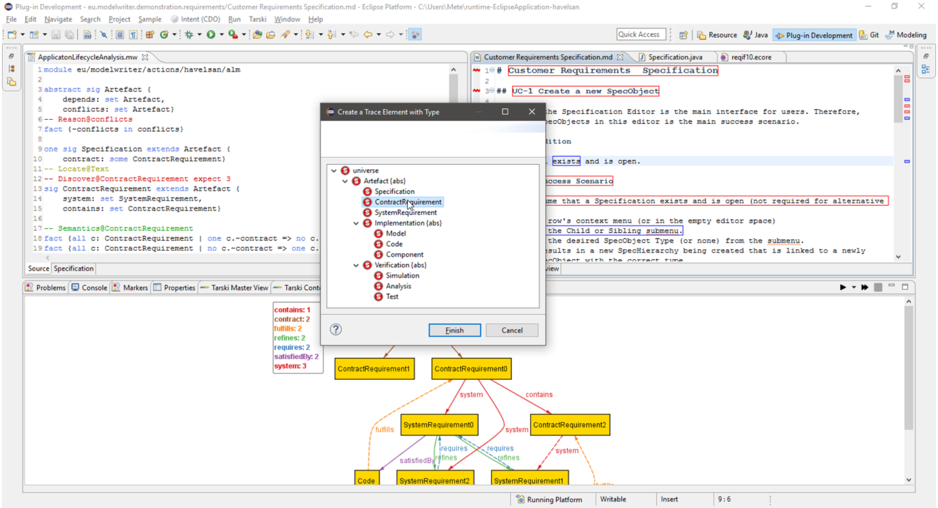


Outline

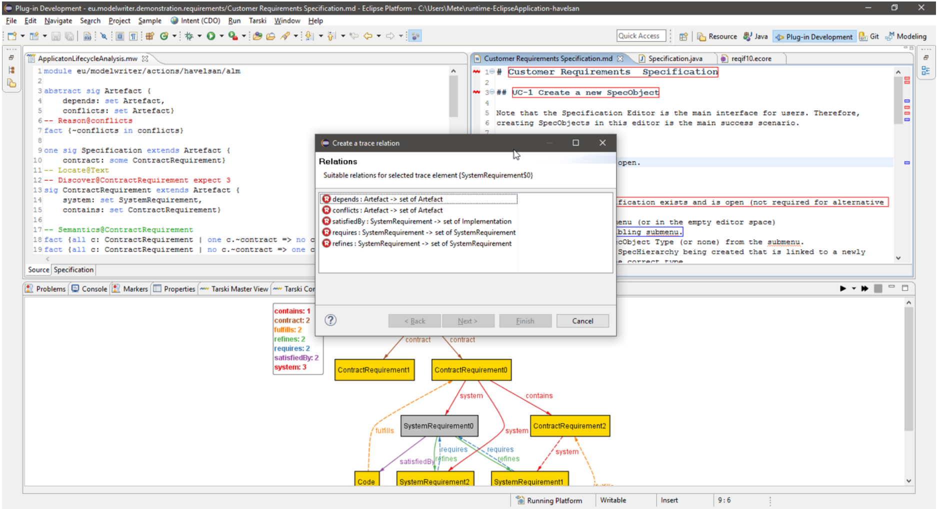
Creating *Trace-locations* and Assigning Types



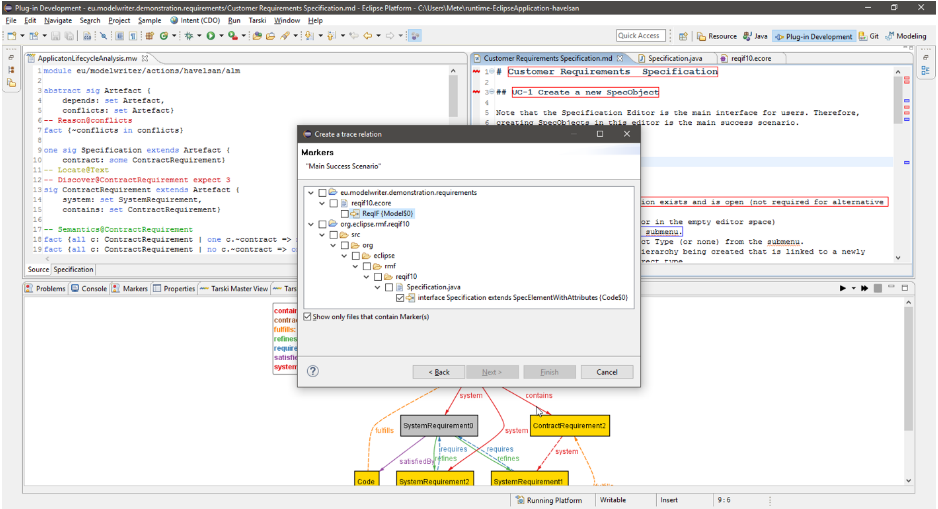
Assigning a *Sub Type* to a *Trace-location*



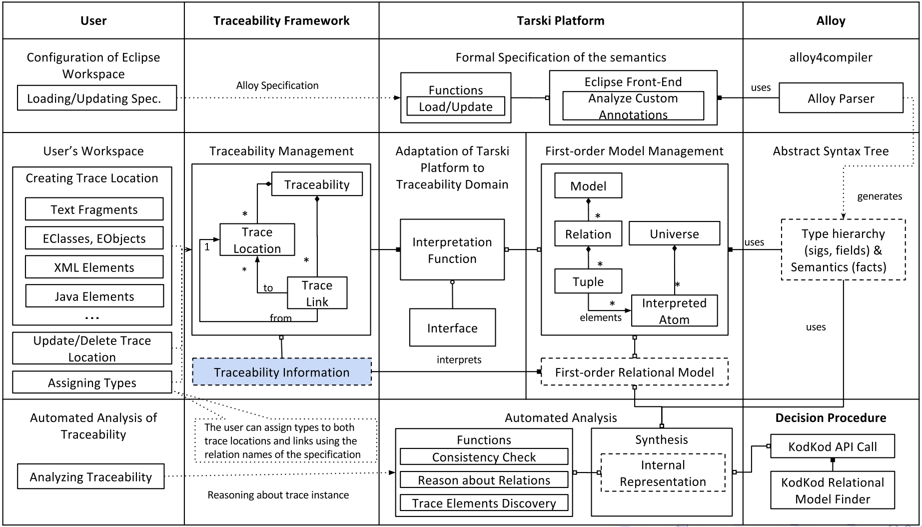
Assigning a binary *Field Type* to a *Trace-link*



Selecting a *Trace-Location* from the co-domain of the *type*



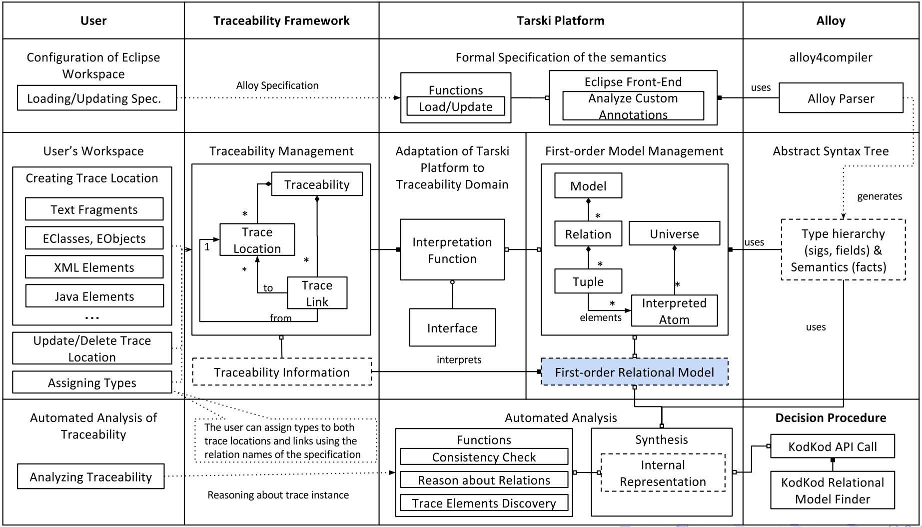
Traceability Information



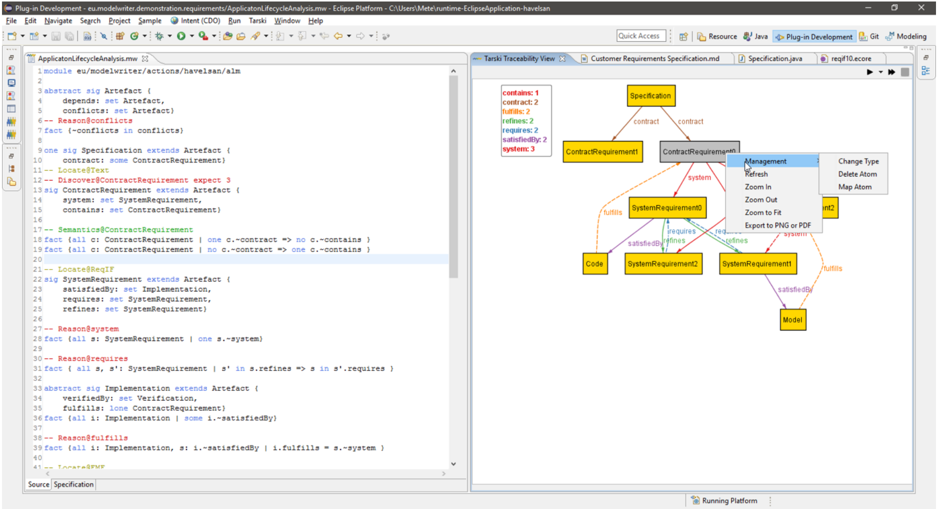
Outline

1. Introduction
   - Motivation
   - Industrial Use Cases
2. Approach
   - Traceability Domain Model
   - First-order Relational Model and Logic
   - Type Annotation and Trace-Relations
   - Formal Semantics and Automated Analysis
3. Demonstration
   - Formal Specification of Traceability Semantics
   - Traceability Management
   - First-order Model Management
   - Automated Analysis of Traceability
4. Conclusion and Future Work

First-order Relational Model

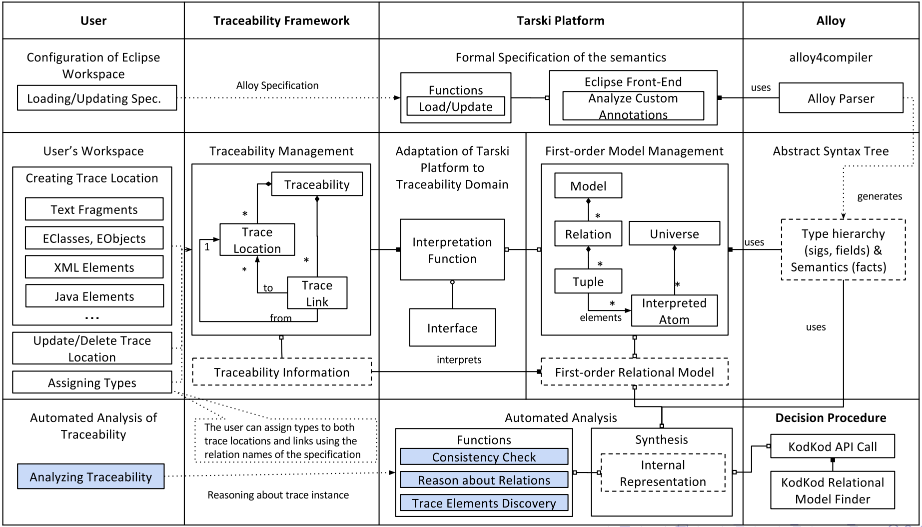


Dynamic Configuration & Model Management

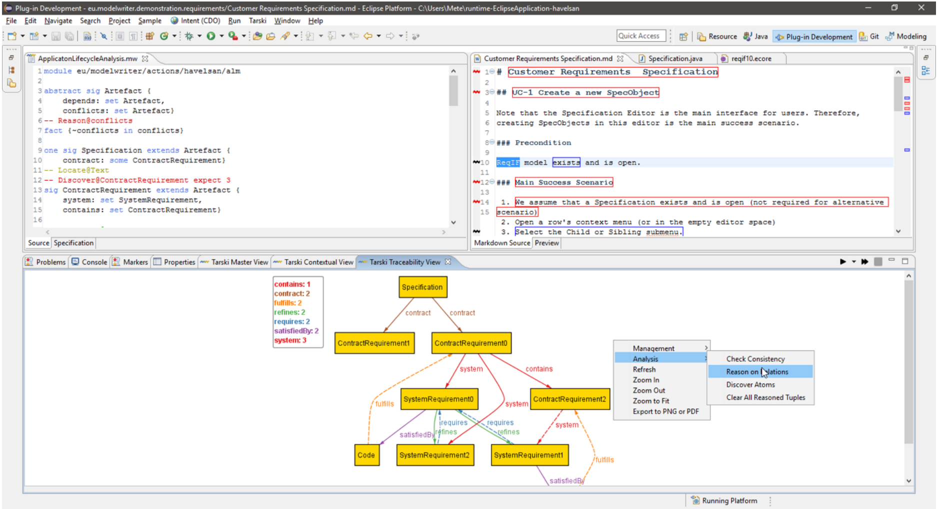


Outline

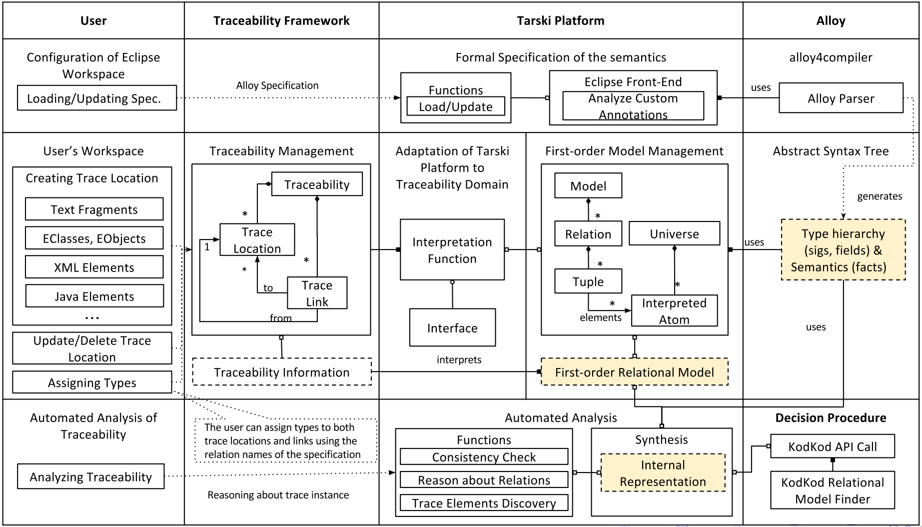
Reasoning about Trace-instance



Automated Analysis of Traceability



Synthesis of Internal Representation



- Should we consider also the temporal behavior of the traceability? Interesting analysis scenarios exist in industry
- We are not supporting ordered sets of Alloy which usually help model the dynamic behaviour.
- First-order theory of relations might be a candidate for traceability in Multi-pardigm Modeling for Cyber-physical Systems. Preliminary results shows that the approach works on the synchronization of design rules with design/installation of physical components.
- However, DPLL(T) solvers does not currently exists for this fragment of the theory.
- Alloy Language is too expressive for the domain of traceability. We're working on the formalization of a First-order theory for traceability and the development of a domain-specific language for traceability.

Modeling and Reasoning Approaches