Qualisist Tool Training

Mauricio Alferez, Angelo Rizzi and Alvaro Veizaga
SnT, University of Luxembourg
mauricio.alferez@uni.lu / angelo.rizzi@uni.lu / alvaro.veizaga@uni.lu

Qualisist
A joint project of SnT, escent and Clearstream

Challenges Addressed by Qualisist

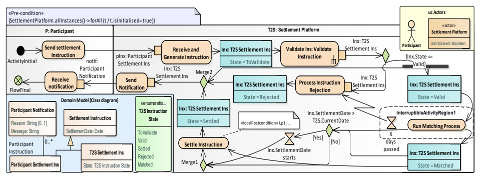
Production of high-quality requirements and models

Generation of a full deliverable in a single tool

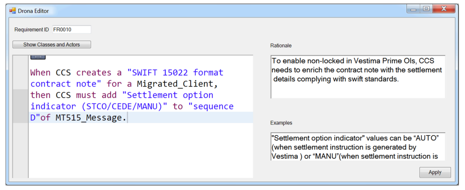
Automated generation of acceptance criteria

The Qualisist Solution

1. Modeling support



2. Requirements authoring support



Requirement ID FR0010

```
When CCS creates a "SWIFT 15022 format
contract note" for a Migrated_Client,
then CCS must add "Settlement option
indicator (STCO/CEDE/MANU)" to "sequence
D"of MT515_Message.
```

Rationale

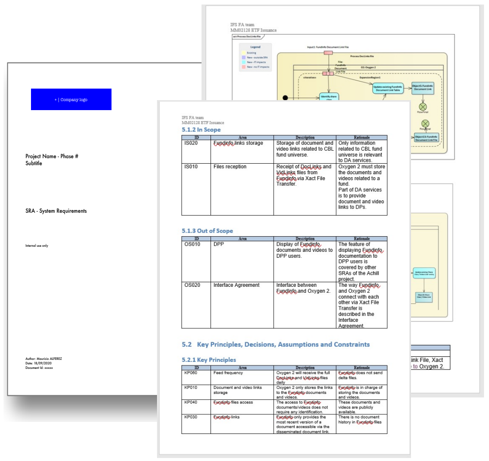
To enable non-locked in Vestima Prime OIs, CCS needs to enrich the contract note with the settlement details complying with swift standards.

Examples

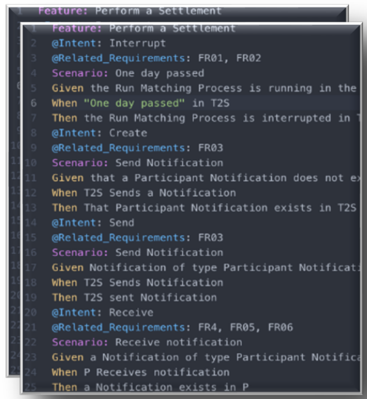
"Settlement option indicator" values can be "AUTO" (when settlement instruction is generated by Vestima) or "MANU"(when settlement instruction is

3. Requirements-to-model reconciliation support

4. Full deliverable generation



5. Gherkin test scenarios generation



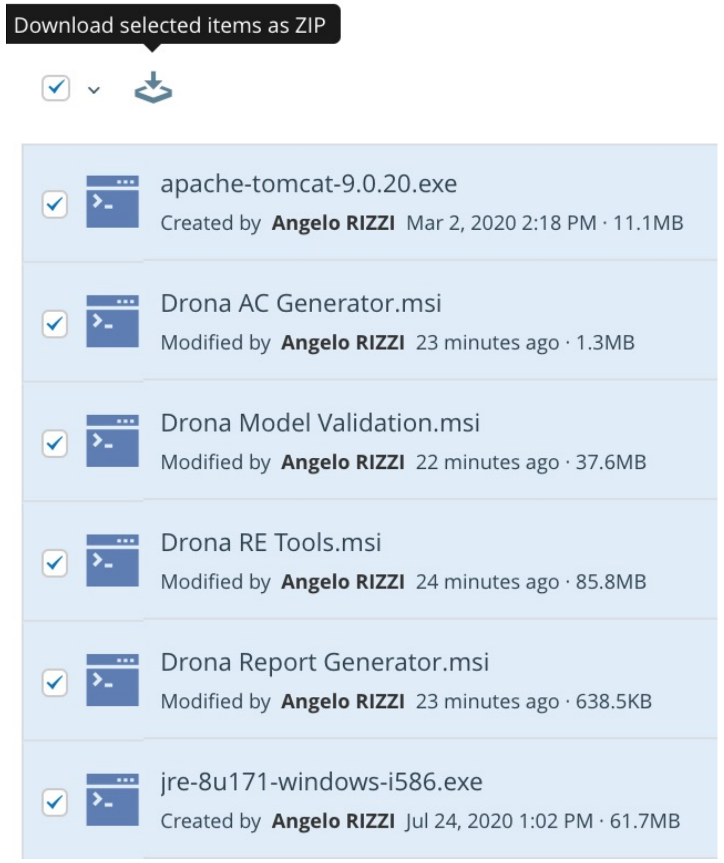
```
Feature: Perform a Settlement
@Intent: Interrupt
@Related_Requirements: FR01, FR02
Scenario: One day passed
Given the Run Matching Process is running in the
When "One day passed" in T2S
Then the Run Matching Process is interrupted in
@Intent: Create
@Related_Requirements: FR03
Scenario: Send Notification
Given that a Participant Notification does not e
When T2S Sends a Notification
Then That Participant Notification exists in T2S
@Intent: Send
@Related_Requirements: FR03
Scenario: Send Notification
Given Notification of type Participant Notificat
When T2S Sends Notification
Then T2S sent Notification
@Intent: Receive
@Related_Requirements: FR4, FR05, FR06
Scenario: Receive notification
Given a Notification of type Participant Notifica
When P Receives notification
Then a Notification exists in P
```

Agenda

0. **Installation and configuration**
1. Modelling Support
2. Requirements authoring support
3. Requirements-to-Model reconciliation support
4. Full deliverable generation
5. Gherkin test Scenarios generation

Installation and Configuration

- Download the latest installers available at https://dropit.uni.lu/invitations?share=ce52a5ed37e4c39b90a0



apache-tomcat-9.0.20.exe
Created by **Angelo RIZZI** Mar 2, 2020 2:18 PM · 11.1MB

Drona AC Generator.msi
Modified by **Angelo RIZZI** 23 minutes ago · 1.3MB

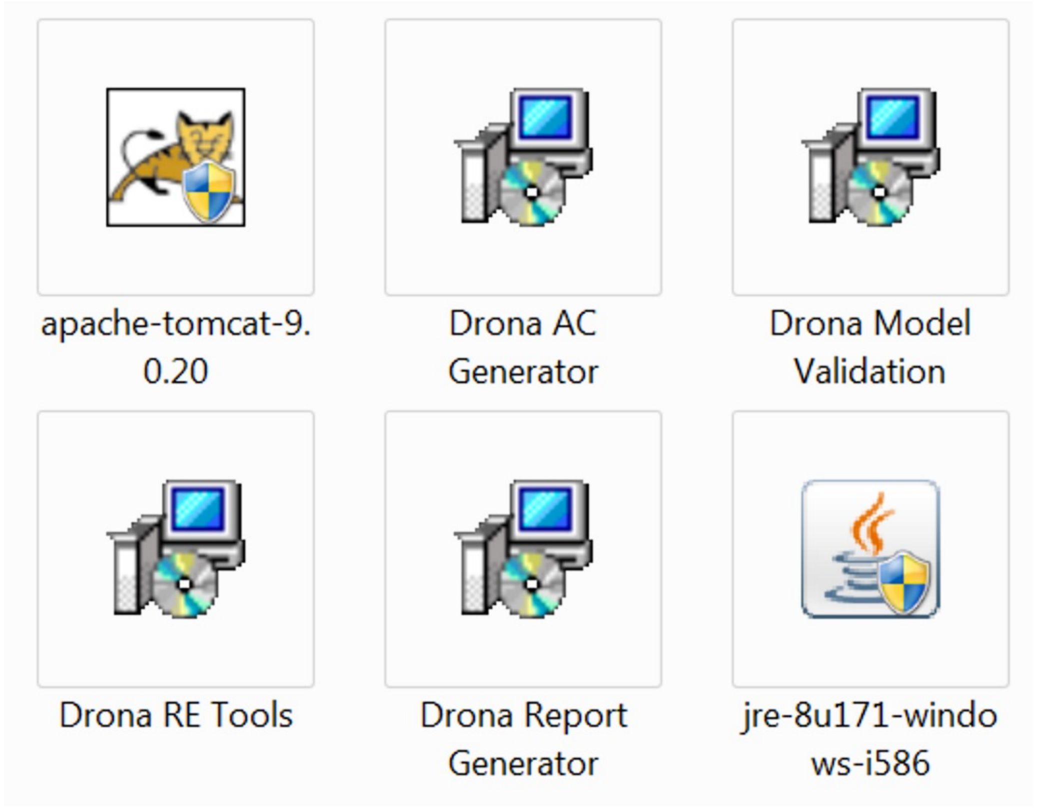
Drona Model Validation.msi
Modified by **Angelo RIZZI** 22 minutes ago · 37.6MB

Drona RE Tools.msi
Modified by **Angelo RIZZI** 24 minutes ago · 85.8MB

Drona Report Generator.msi
Modified by **Angelo RIZZI** 23 minutes ago · 638.5KB

jre-8u171-windows-i586.exe
Created by **Angelo RIZZI** Jul 24, 2020 1:02 PM · 61.7MB

- Extract the installers to your machine

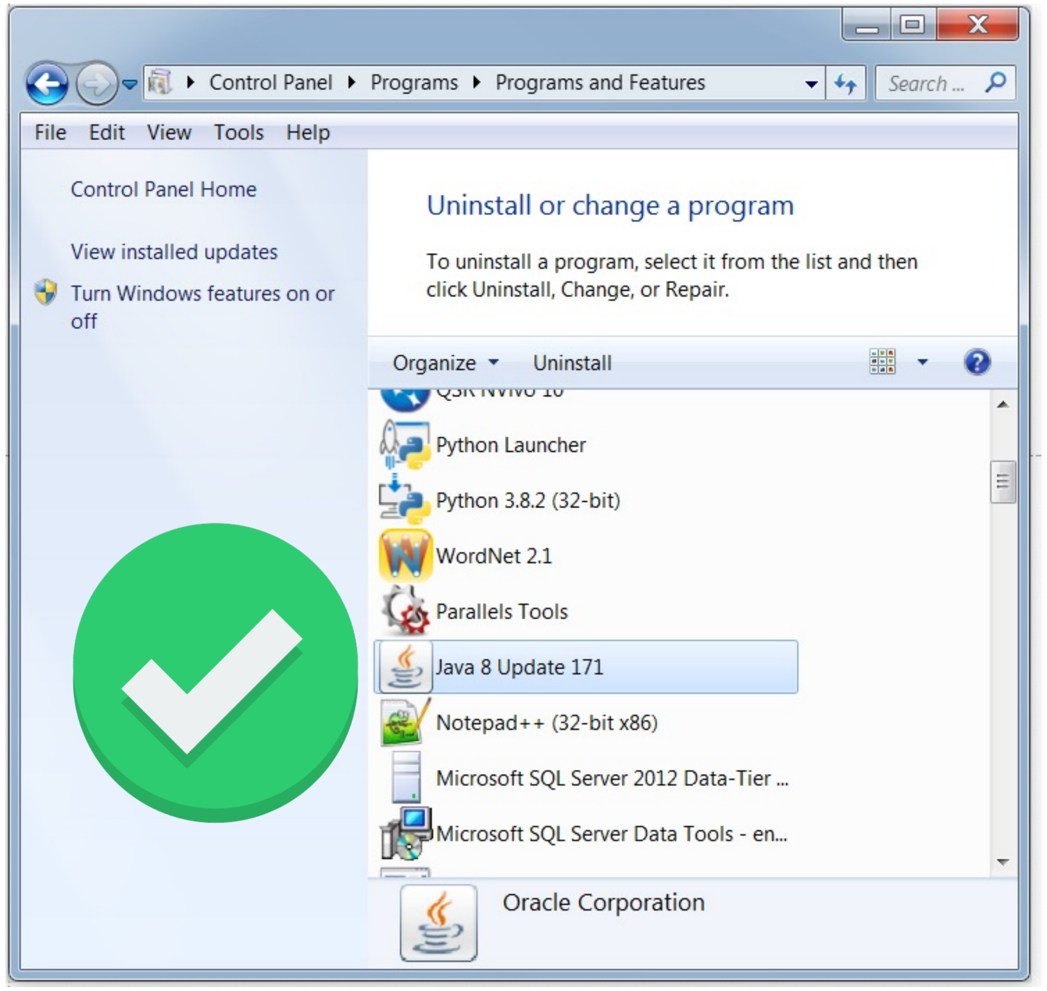


Installation of Java

- Check if Java 8 is installed on your computer.

  Start→Control Panel→Programs and Features.

- If Java is not listed in Programs and Features, install Java SE 8 using the file jre-8u171-windows-i586.exe (See the next slide)



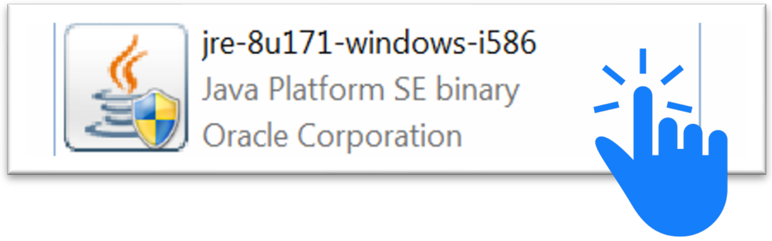
Uninstall or change a program

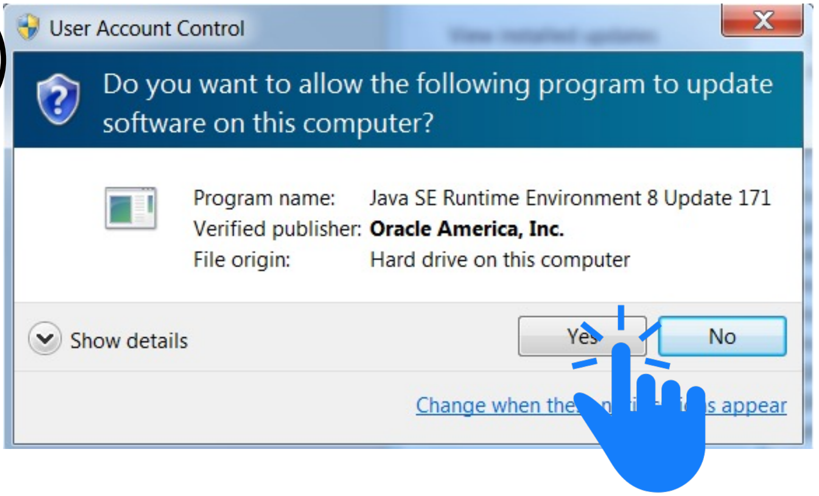
To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

QSR NVivo 10
Python Launcher
Python 3.8.2 (32-bit)
WordNet 2.1
Parallels Tools
Java 8 Update 171
Notepad++ (32-bit x86)
Microsoft SQL Server 2012 Data-Tier ...
Microsoft SQL Server Data Tools - en...

Oracle Corporation

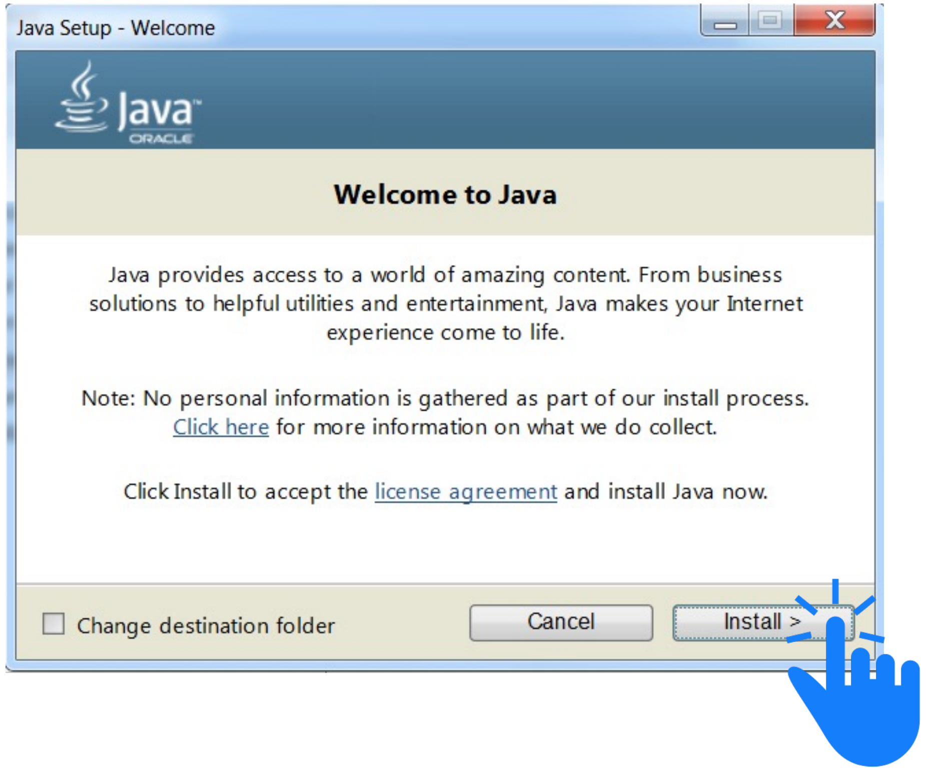
Installation of Java

①

jre-8u171-windows-i586
Java Platform SE binary
Oracle Corporation

②

User Account Control

Do you want to allow the following program to update software on this computer?

Program name: Java SE Runtime Environment 8 Update 171
Verified publisher: **Oracle America, Inc.**
File origin: Hard drive on this computer

③

Java Setup - Welcome

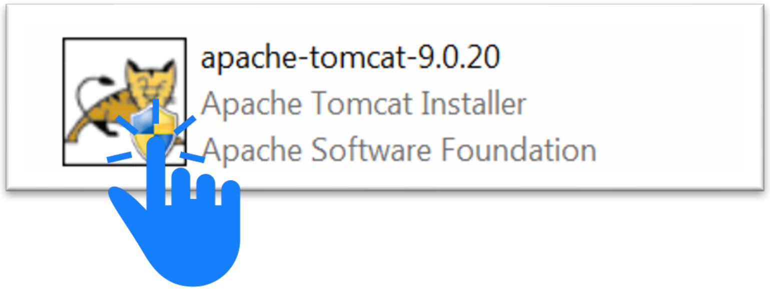
Welcome to Java

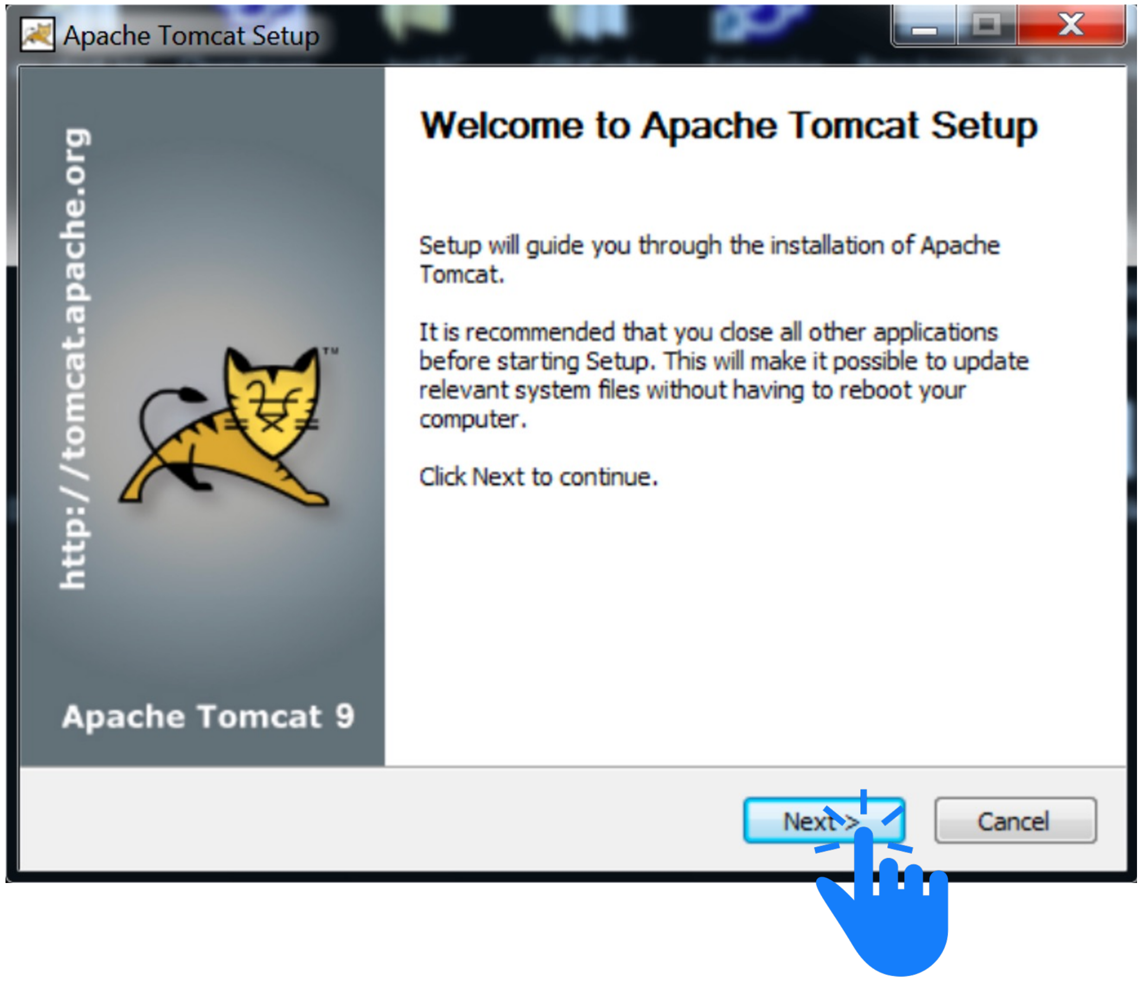
Java provides access to a world of amazing content. From business solutions to helpful utilities and entertainment, Java makes your Internet experience come to life.

Note: No personal information is gathered as part of our install process. Click here for more information on what we do collect.

Click Install to accept the license agreement and install Java now.

Tomcat Installation and Configuration


apache-tomcat-9.0.20
Apache Tomcat Installer
Apache Software Foundation


Apache Tomcat Setup

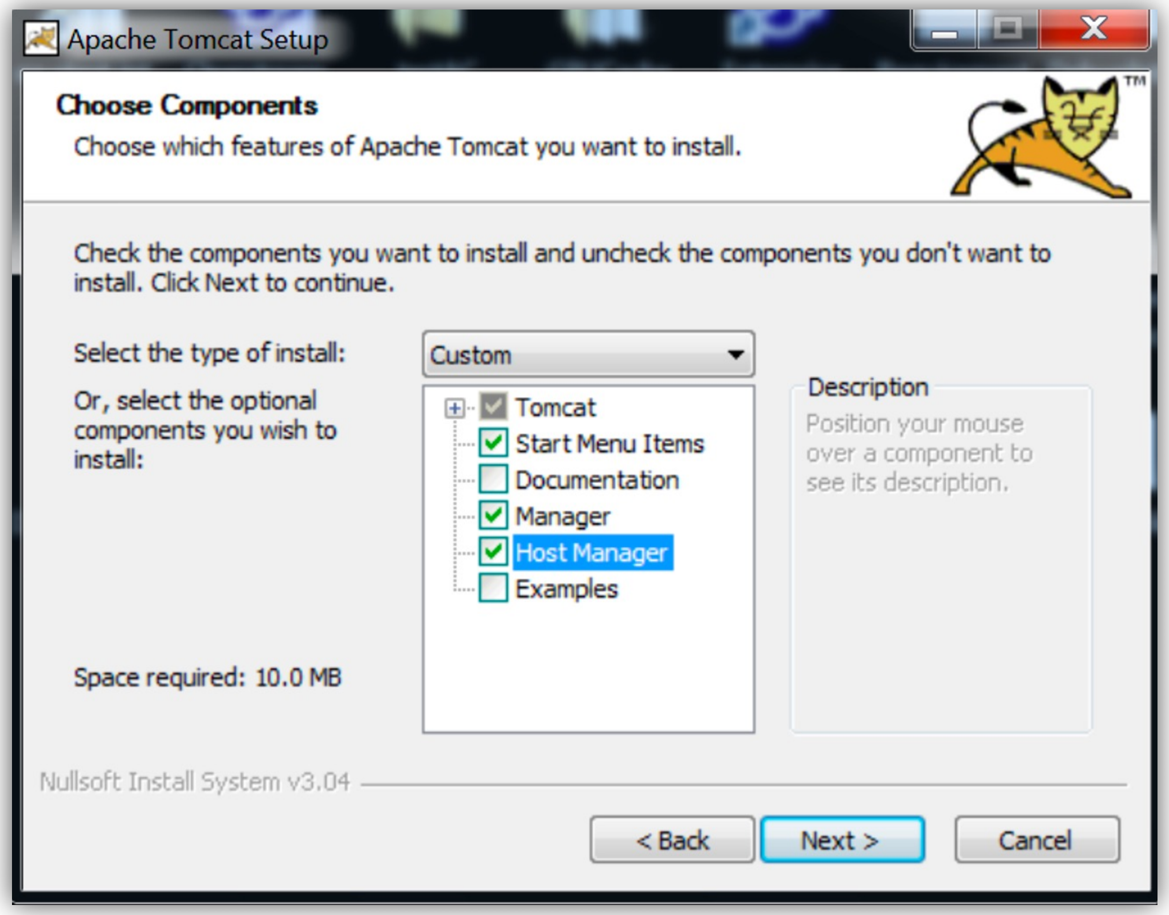
Welcome to Apache Tomcat Setup

Setup will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Tomcat Installation and Configuration



Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Custom

Or, select the optional components you wish to install:

- [✓] Tomcat
- [✓] Start Menu Items
- [ ] Documentation
- [✓] Manager
- [✓] Host Manager
- [ ] Examples

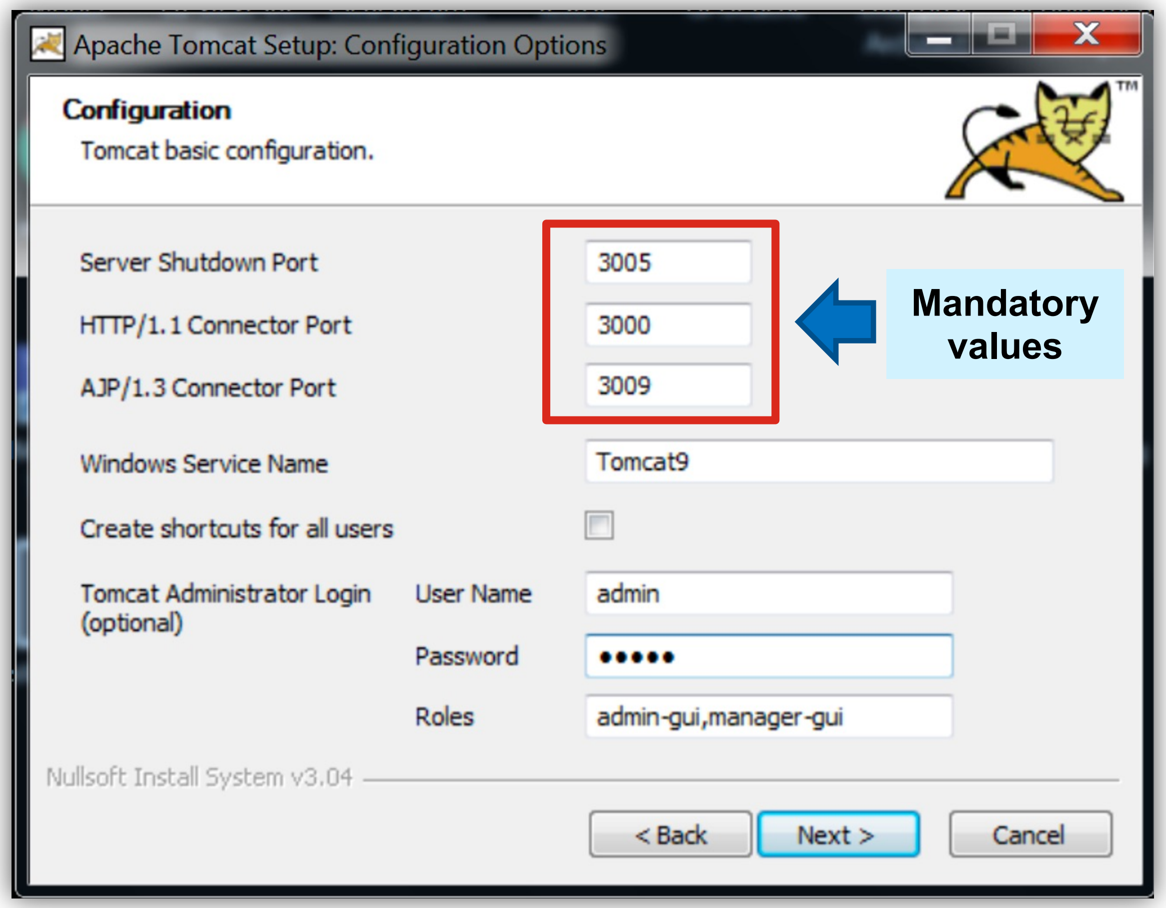
Description

Position your mouse over a component to see its description.

Space required: 10.0 MB

Nullsoft Install System v3.04

< Back | Next > | Cancel



Apache Tomcat Setup: Configuration Options

Configuration

Tomcat basic configuration.

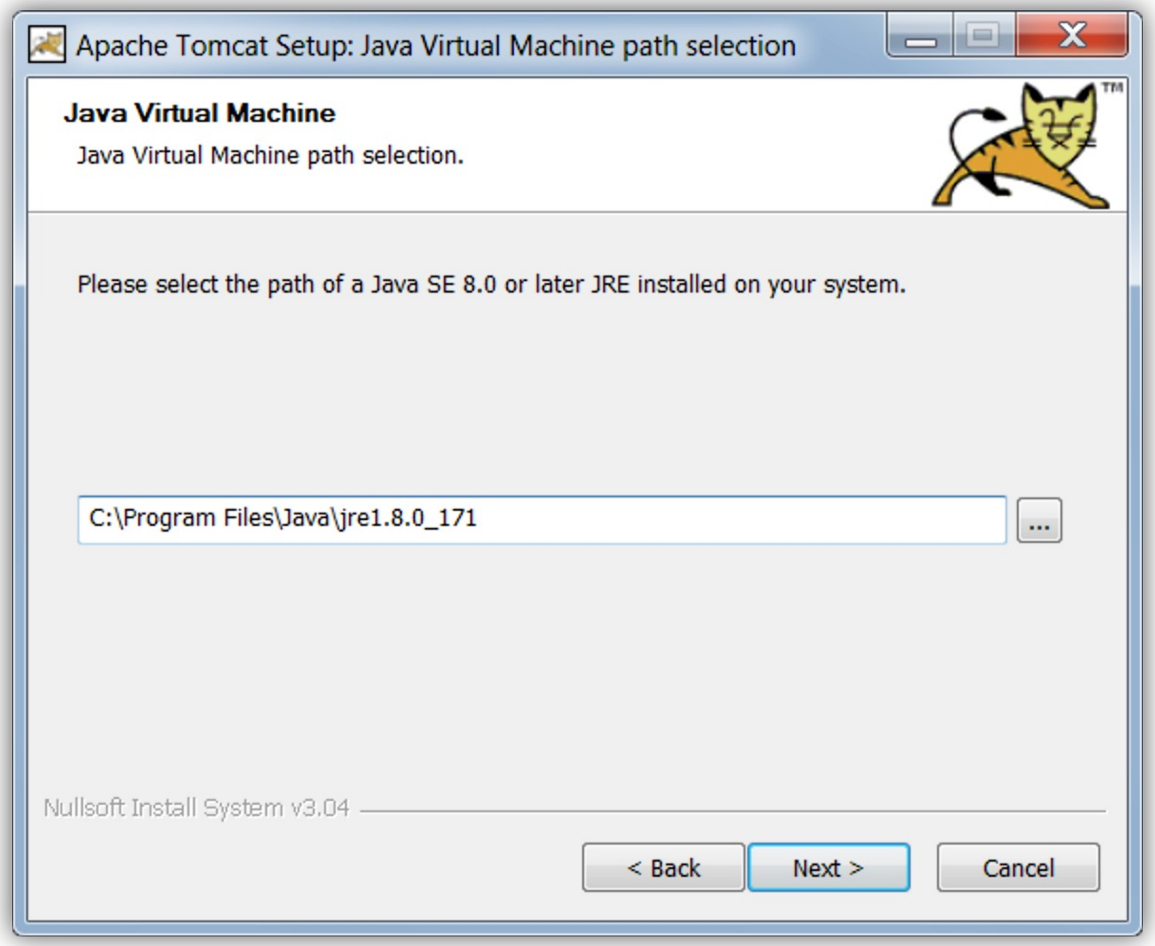
Server Shutdown Port		3005
HTTP/1.1 Connector Port		3000
AJP/1.3 Connector Port		3009
Windows Service Name		Tomcat9
Create shortcuts for all users		☐
Tomcat Administrator Login (optional)	User Name	admin
	Password	●●●●●
	Roles	admin-gui,manager-gui

Mandatory values

Nullsoft Install System v3.04

< Back | Next > | Cancel

Tomcat Installation and Configuration


Apache Tomcat Setup: Java Virtual Machine path selection

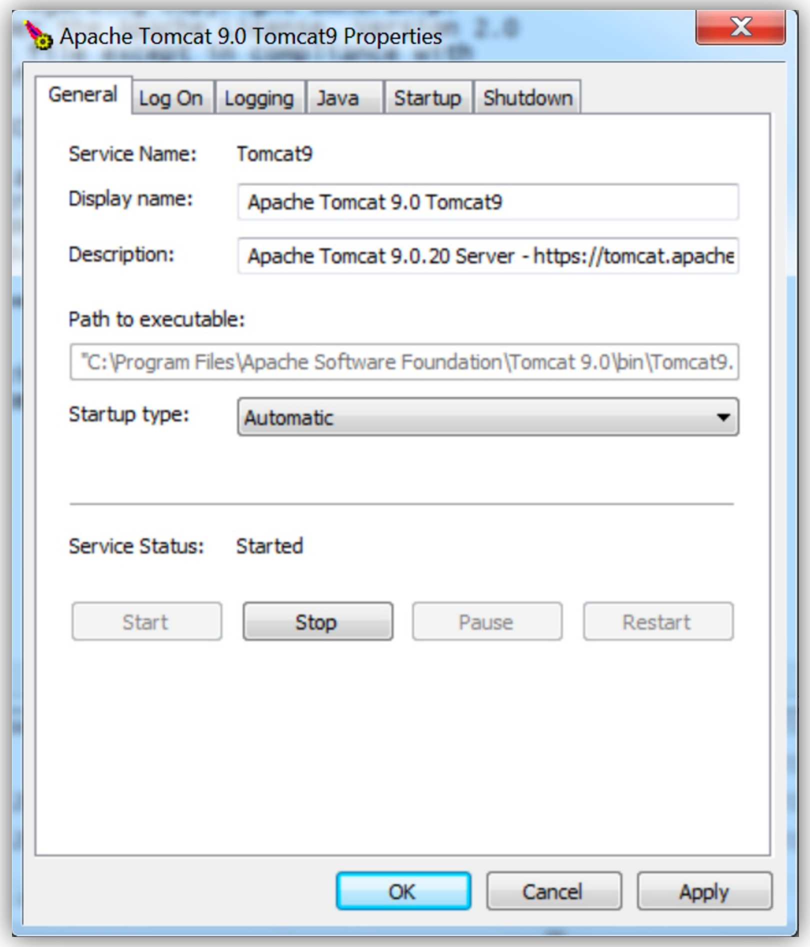
Java Virtual Machine

Java Virtual Machine path selection.

Please select the path of a Java SE 8.0 or later JRE installed on your system.

C:\Program Files\Java\jre1.8.0_171

Nullsoft Install System v3.04


Apache Tomcat 9.0 Tomcat9 Properties

Service Name: Tomcat9

Display name: Apache Tomcat 9.0 Tomcat9

Description: Apache Tomcat 9.0.20 Server - https://tomcat.apache

Path to executable:

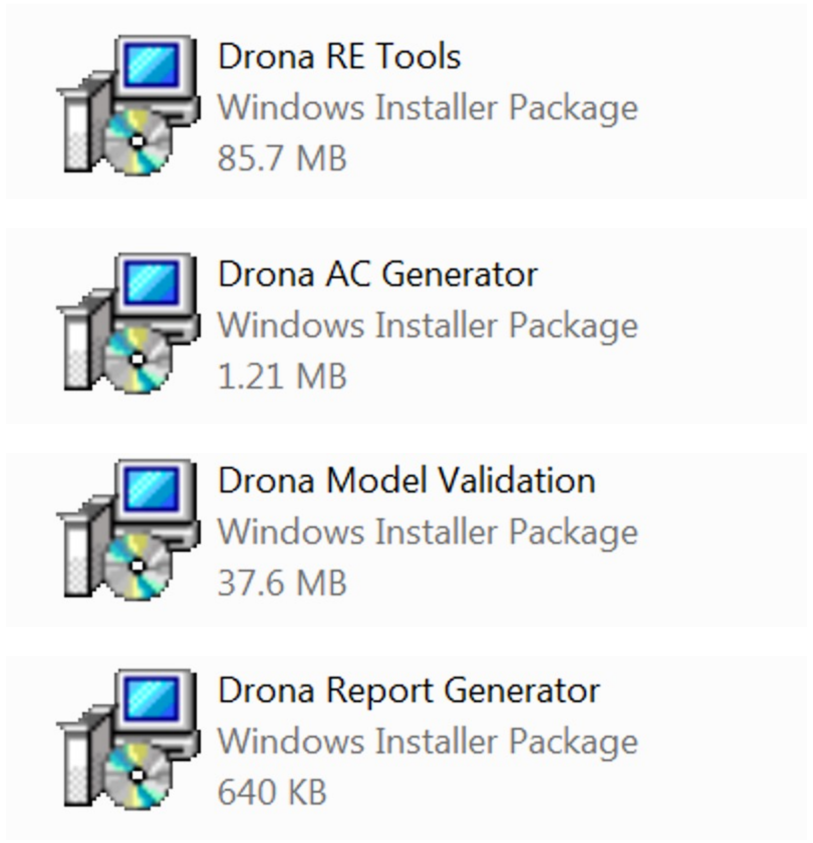
"C:\Program Files\Apache Software Foundation\Tomcat 9.0\bin\Tomcat9.

Startup type: Automatic

Service Status: Started

Qualisist Add-Ins Installation

1. Qualisist Requirement Editing Tools
2. Qualisist Acceptance Criteria Generator
3. Qualisist Validation Rules
4. Qualisist Report Generator


Drona RE Tools
Windows Installer Package
85.7 MB

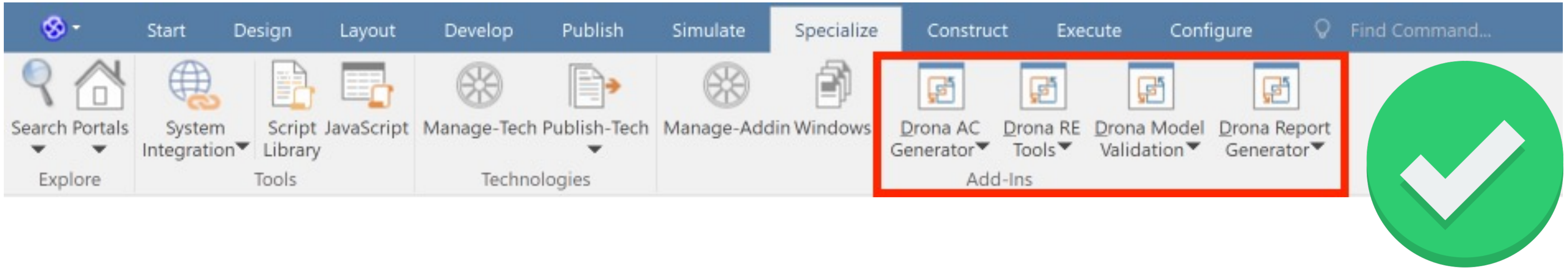
Drona AC Generator
Windows Installer Package
1.21 MB

Drona Model Validation
Windows Installer Package
37.6 MB

Drona Report Generator
Windows Installer Package
640 KB

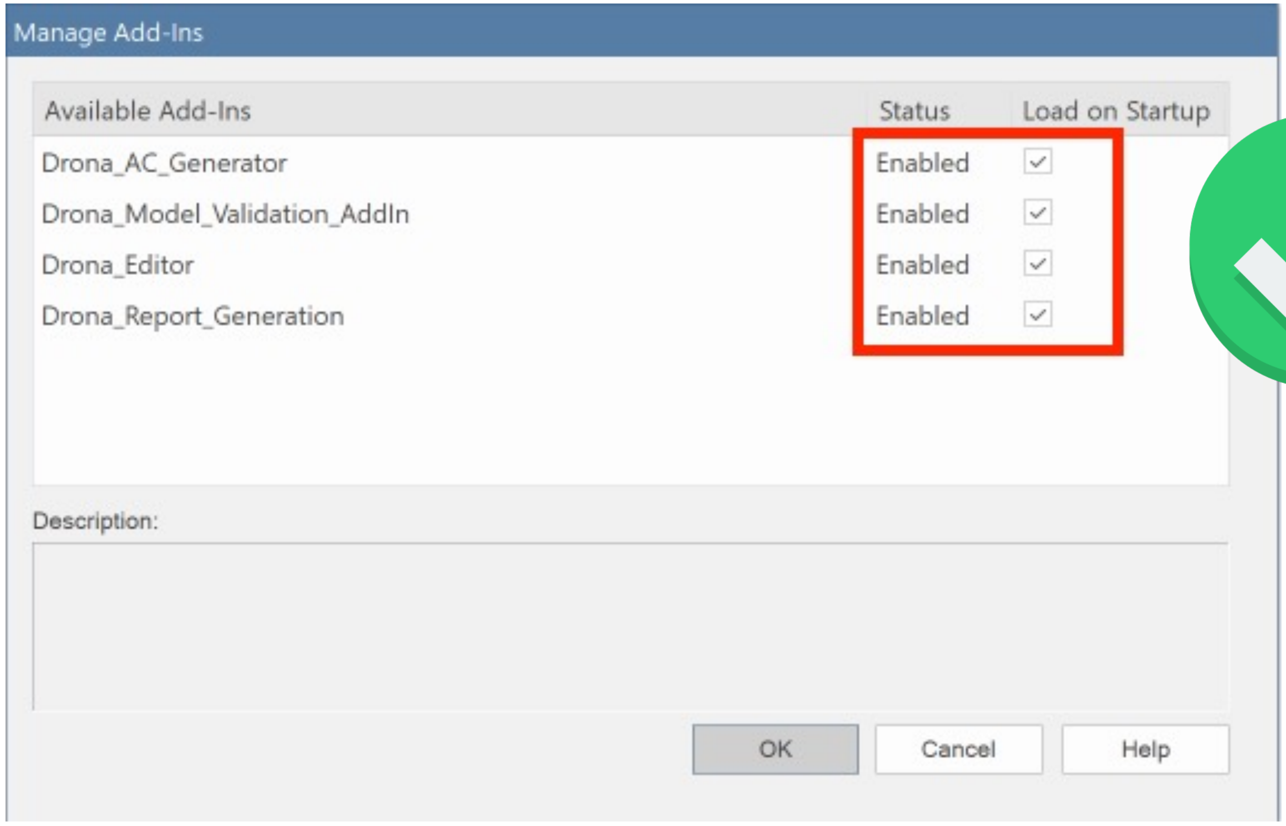
Qualisist Add-Ins Configuration

Open Enterprise Architect and go to Specialize → Add-Ins
You will see the icons shown below



Qualisist Add-Ins Configuration

Go to: Specialize → Add-Ins → Manage Add-Ins
Make sure all the Qualisist Add-Ins will load on startup



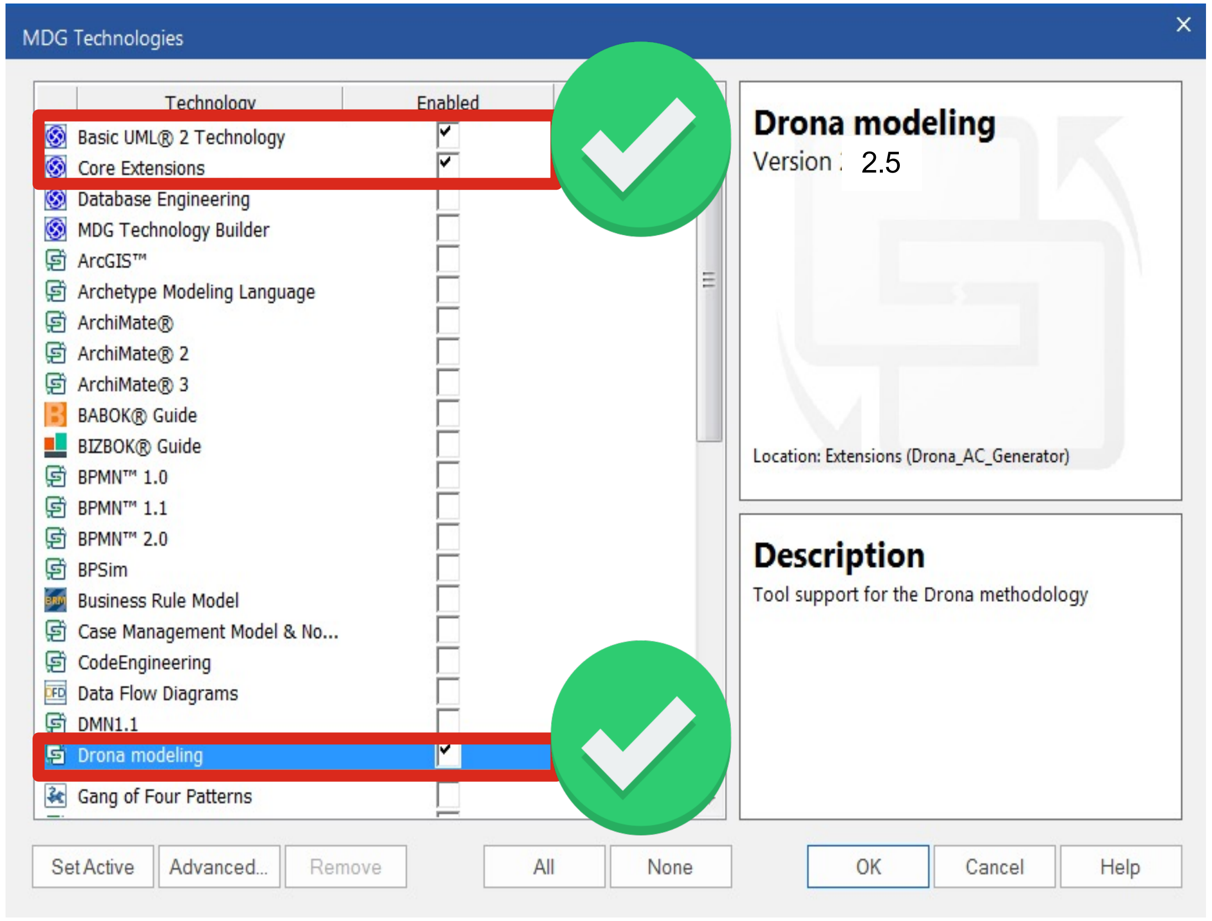
Manage Add-Ins

Available Add-Ins	Status	Load on Startup
Drona_AC_Generator	Enabled	✓
Drona_Model_Validation_AddIn	Enabled	✓
Drona_Editor	Enabled	✓
Drona_Report_Generation	Enabled	✓

Description:

Qualisist Add-Ins Configuration

- Go to: Specialize → Technologies → Manage
- Make sure that at least the following MDG (Model-Driven Generation) are selected:
  - (1) Basic UML2 Technology
  - (2) Core Extensions, and
  - (3) Qualisist modelling


MDG Technologies

Technology	Enabled
Basic UML® 2 Technology	✓
Core Extensions	✓
Database Engineering	
MDG Technology Builder	
ArcGIS™	
Archetype Modeling Language	
ArchiMate®	
ArchiMate® 2	
ArchiMate® 3	
BABOK® Guide	
BIZBOK® Guide	
BPMN™ 1.0	
BPMN™ 1.1	
BPMN™ 2.0	
BPSim	
Business Rule Model	
Case Management Model & No...	
CodeEngineering	
Data Flow Diagrams	
DMN1.1	
Drona modeling	✓
Gang of Four Patterns	

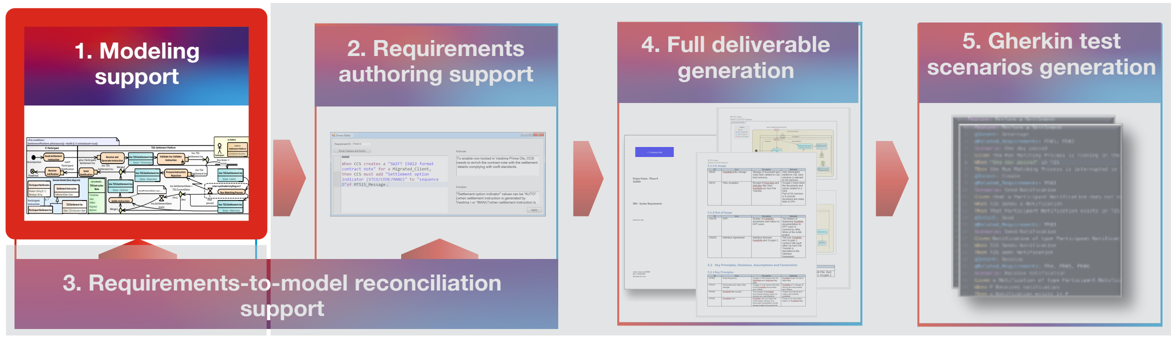
Drona modeling
Version : 2.5
Location: Extensions (Drona_AC_Generator)

Description
Tool support for the Drona methodology

Agenda

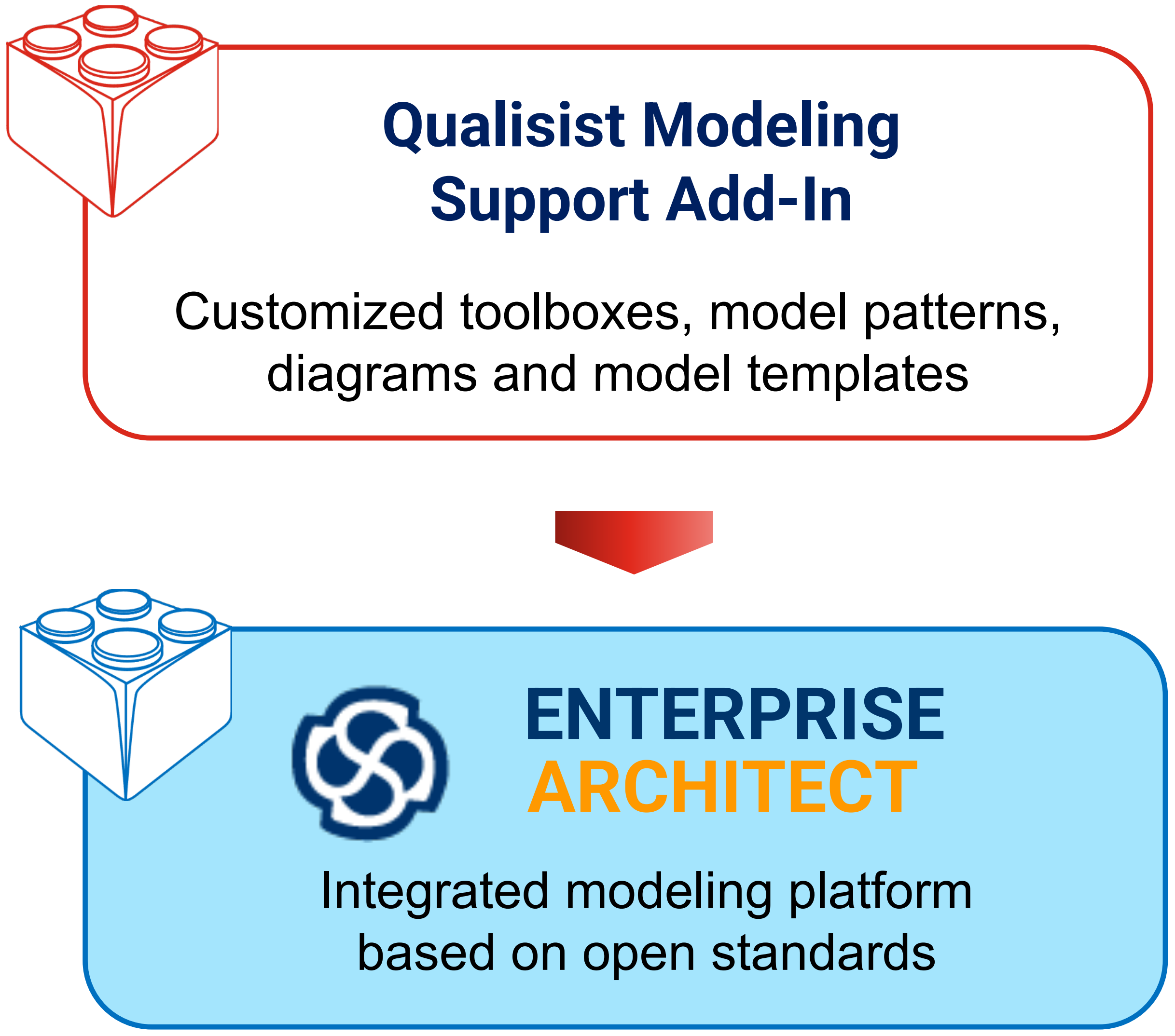
0. Installation and Configuration
1. **Modelling Support**
2. Requirements authoring support
3. Requirements-to-Model reconciliation support
4. Full deliverable generation
5. Gherkin test Scenarios generation

The Qualisist Solution


1. Modeling support
2. Requirements authoring support
4. Full deliverable generation
5. Gherkin test scenarios generation
3. Requirements-to-model reconciliation support

Qualisist Modeling Tool Support

- Full integration into the Enterprise Architect modelling platform
- Extension and customization of the modeling functionality of Enterprise Architect



Qualisist Modeling Support Add-In

Customized toolboxes, model patterns, diagrams and model templates

ENTERPRISE ARCHITECT

Integrated modeling platform based on open standards

The Unified Modelling Language (UML)

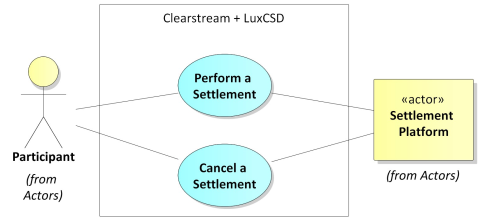
- UML [1] is a standard modelling language intended to be used for
  - modelling business and similar processes,
  - analysis, design, and implementation of software-based systems
- Qualisist proposes a UML-based methodology and tool-support
- In Qualisist, the software requirements are documented using
  - A subset of the UML, and
  - a controlled natural language for requirements

[1] https://www.omg.org/spec/UML/2.5/

Types of UML Diagrams used in Qualisist

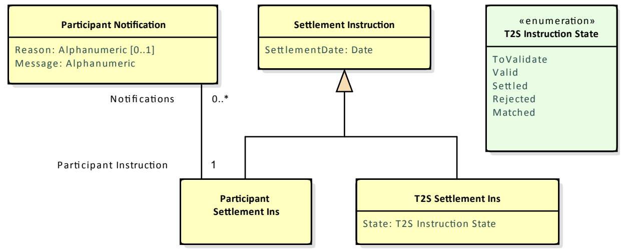
Use Case Diagrams (UCD)

- Use case diagrams express the expectations of the customers/stakeholders



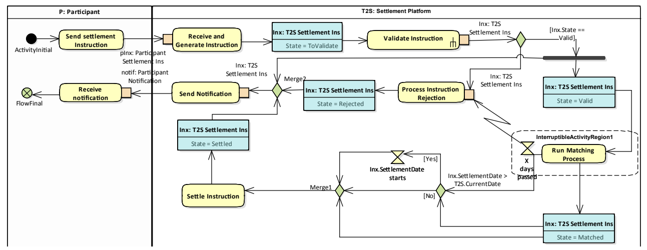
Class diagram (CD)

- In Qualisist, we use CDs to represents **Domain Models**
- A Domain Model includes concepts of a domain, their attributes, and the relations between them

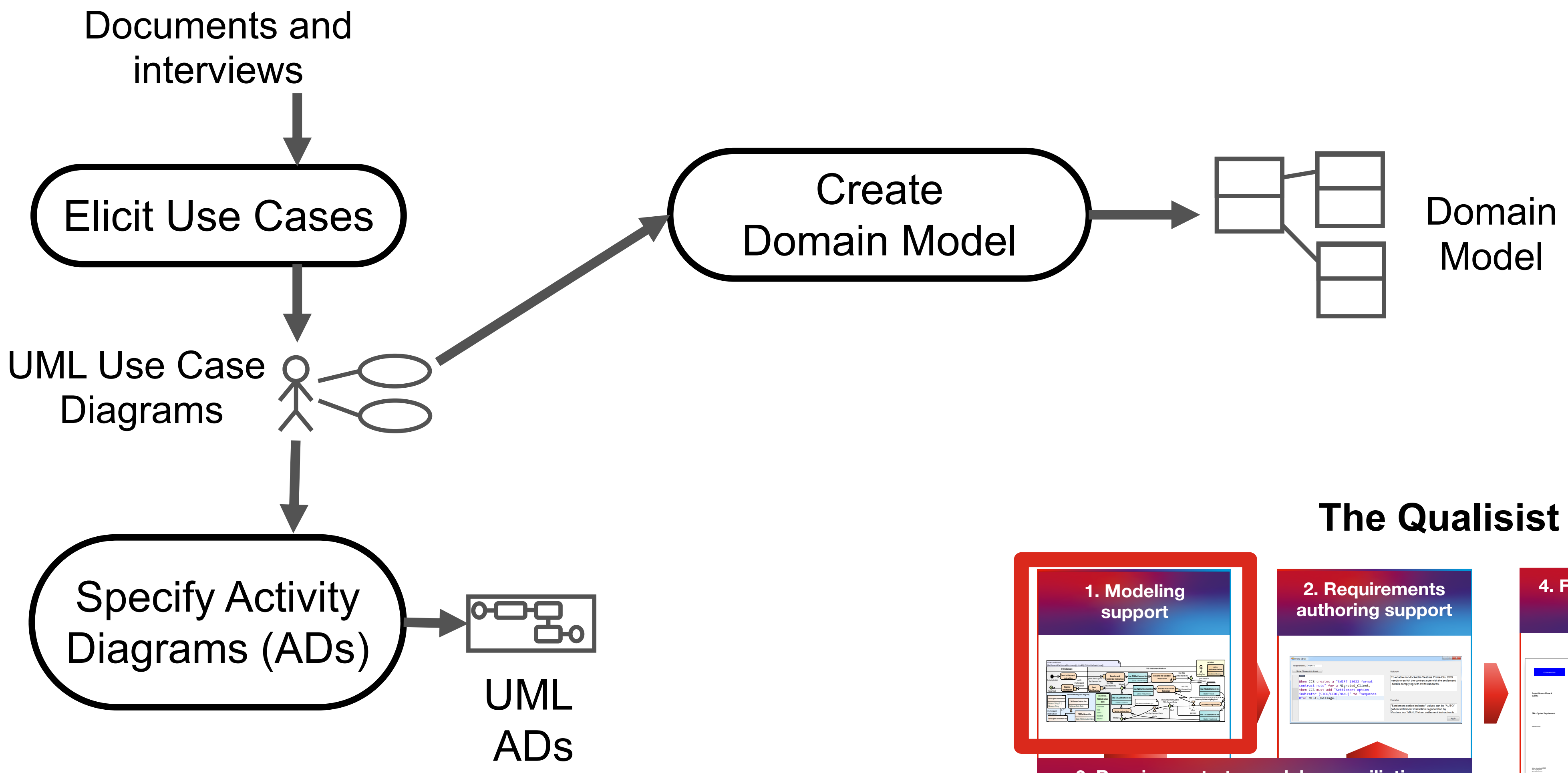


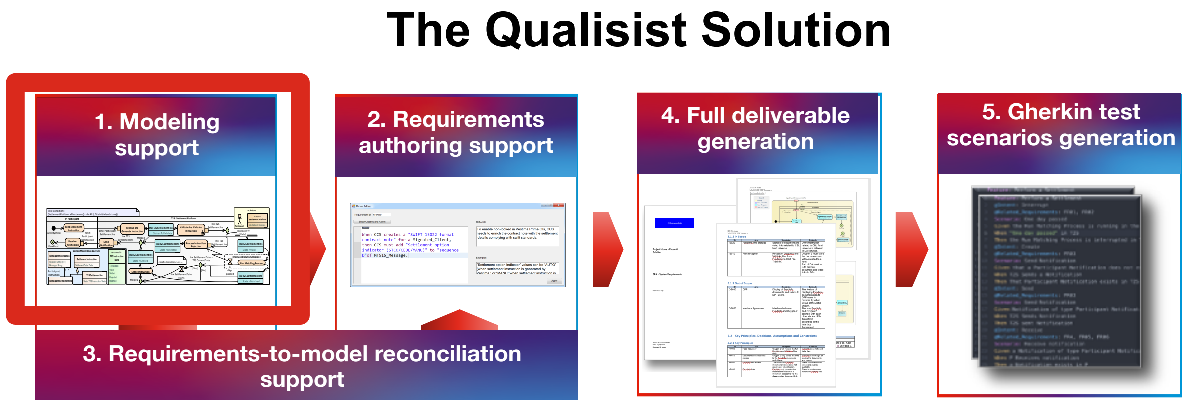
Activity Diagram (AD)

- Workflows of stepwise actions
- Support for choice, iteration and concurrency



Qualisist Modeling Methodology

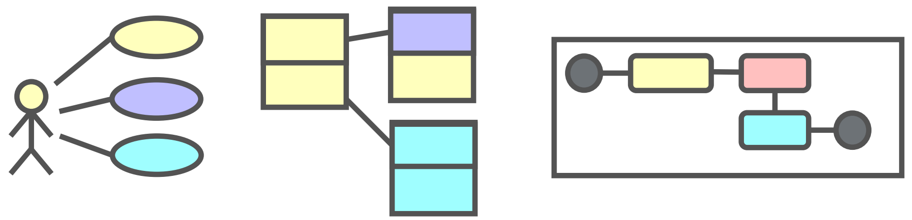



The Qualisist Solution
1. Modeling support
2. Requirements authoring support
3. Requirements-to-model reconciliation support
4. Full deliverable generation
5. Gherkin test scenarios generation

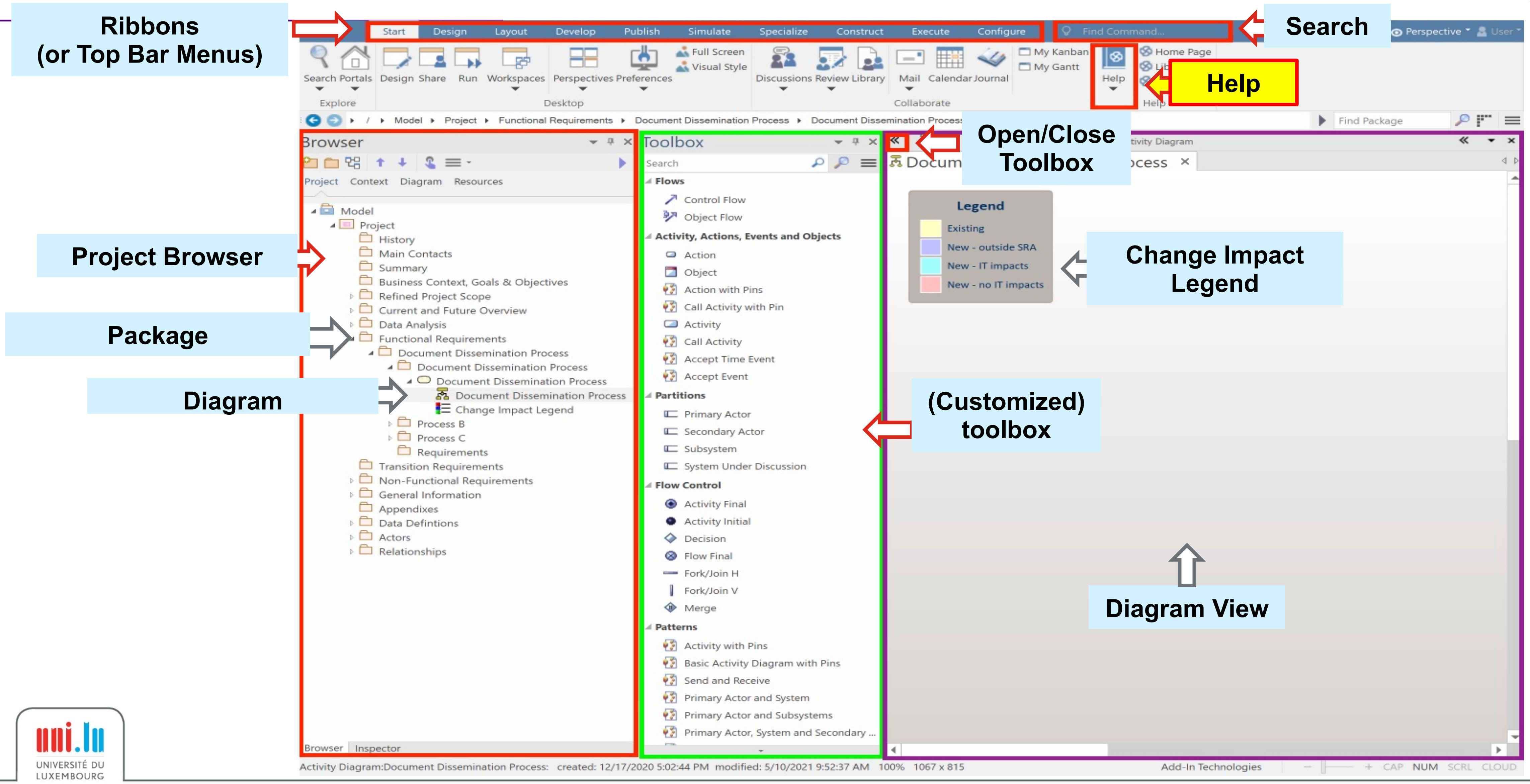
Change Impact Classifications for All UML Elements

- Proposed by Clearstream
- Applied to any element in UML diagrams
- **Existing:** Qualisist applies it to new elements
- **New – IT impacts**. Qualisist applies it to an element *E* when there is at least one requirement traced to *E*
- **New – outside SRA** and **New – no IT impact**. Applied by business analysts.

🟨	Existing
🟪	New - outside SRA
🟦	New - IT impacts
🟥	New - no IT impacts



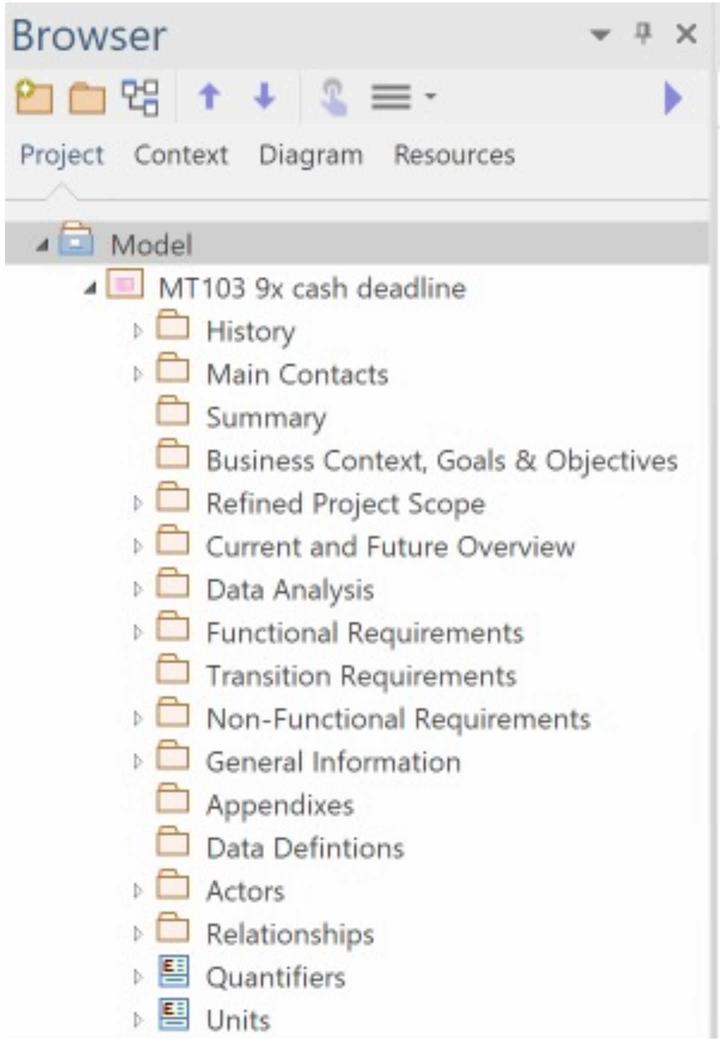
Main User Interface



Activity Diagram:Document Dissemination Process: created: 12/17/2020 5:02:44 PM modified: 5/10/2021 9:52:37 AM 100% 1067 x 815

Qualisist SRA Model Pattern

- Qualisist provides a custom model pattern named "Qualisist SRA Model".
- A Qualisist SRA model is organized in packages that represents the sections of a Software Requirements Analysis (SRA) documents according to Clearstream IFS.


Browser

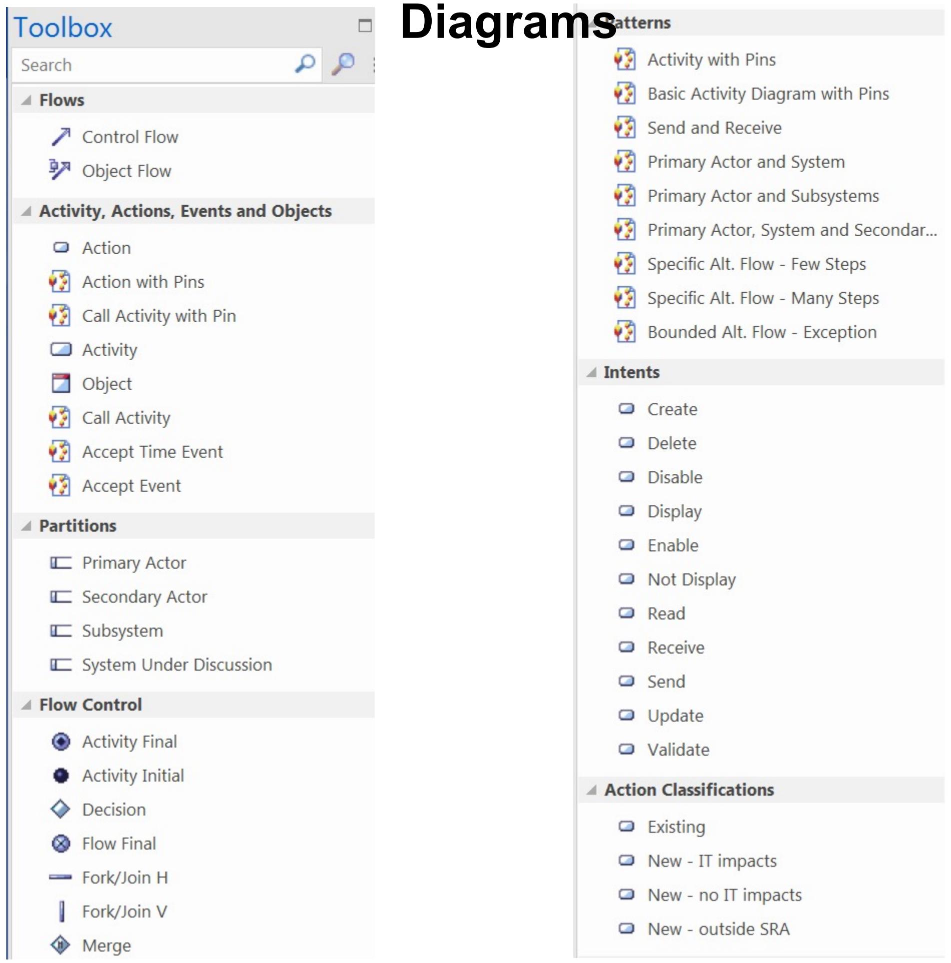
Project Context Diagram Resources

- Model
  - MT103 9x cash deadline
    - History
    - Main Contacts
    - Summary
    - Business Context, Goals & Objectives
    - Refined Project Scope
    - Current and Future Overview
    - Data Analysis
    - Functional Requirements
    - Transition Requirements
    - Non-Functional Requirements
    - General Information
    - Appendixes
    - Data Defintions
    - Actors
    - Relationships
    - Quantifiers
    - Units

Qualisist Modelling Toolboxes

- Each type of diagram has a different toolbox that defines the available elements according to Qualisist.
- Toolboxes provide quick access to:
  - The most recurring elements in the Qualisist methodology
  - Modeling patterns

Example: Toolbox for Qualisist Activity Diagrams



Toolbox

Search

Flows
- Control Flow
- Object Flow

Activity, Actions, Events and Objects
- Action
- Action with Pins
- Call Activity with Pin
- Activity
- Object
- Call Activity
- Accept Time Event
- Accept Event

Partitions
- Primary Actor
- Secondary Actor
- Subsystem
- System Under Discussion

Flow Control
- Activity Final
- Activity Initial
- Decision
- Flow Final
- Fork/Join H
- Fork/Join V
- Merge

Patterns
- Activity with Pins
- Basic Activity Diagram with Pins
- Send and Receive
- Primary Actor and System
- Primary Actor and Subsystems
- Primary Actor, System and Secondar...
- Specific Alt. Flow - Few Steps
- Specific Alt. Flow - Many Steps
- Bounded Alt. Flow - Exception

Intents
- Create
- Delete
- Disable
- Display
- Enable
- Not Display
- Read
- Receive
- Send
- Update
- Validate

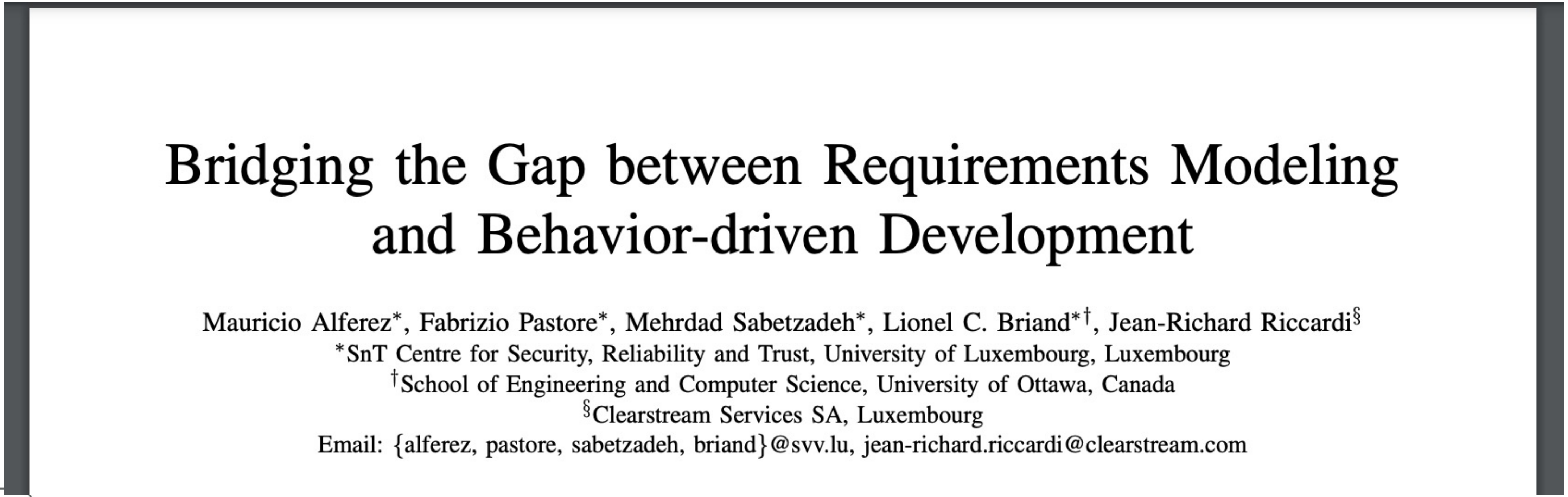
Action Classifications
- Existing
- New - IT impacts
- New - no IT impacts
- New - outside SRA

Modeling Support Resources

- Modelling from Sparx Systems
  - https://sparxsystems.com/resources/user-guides/15.2/index.html#modeling
- Official UML specification (for advanced users)
  - https://www.omg.org/spec/UML/2.5/

More Information about Qualisist Modeling Approach

https://orbilu.uni.lu/handle/10993/39710 (Chapter III)



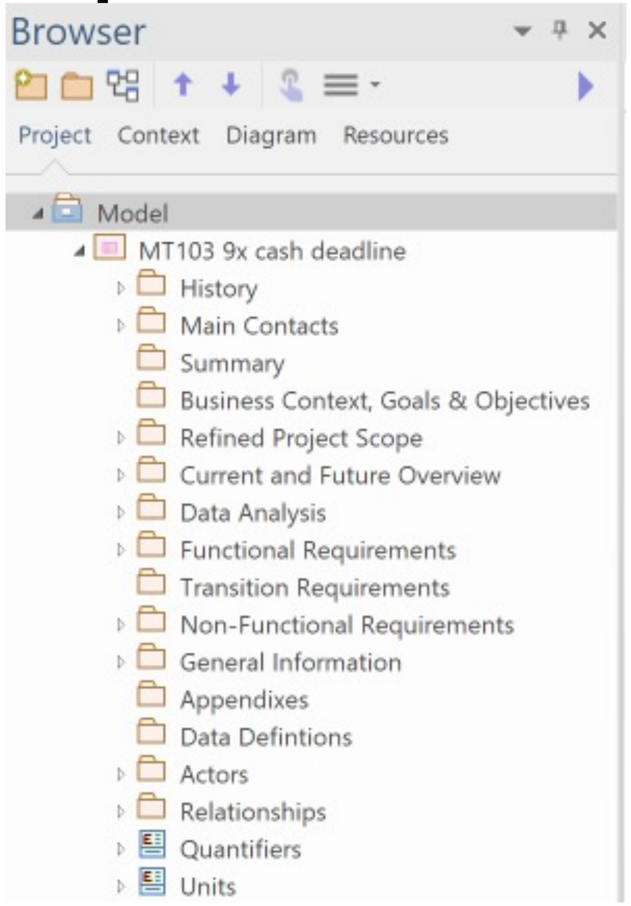
Bridging the Gap between Requirements Modeling and Behavior-driven Development

Mauricio Alferez[*], Fabrizio Pastore[*], Mehrdad Sabetzadeh[*], Lionel C. Briand[*][†], Jean-Richard Riccardi[§]
[*]SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
[†]School of Engineering and Computer Science, University of Ottawa, Canada
[§]Clearstream Services SA, Luxembourg
Email: {alferez, pastore, sabetzadeh, briand}@svv.lu, jean-richard.riccardi@clearstream.com

Practice 1: Create a Qualisist SRA Model Using the Wizard

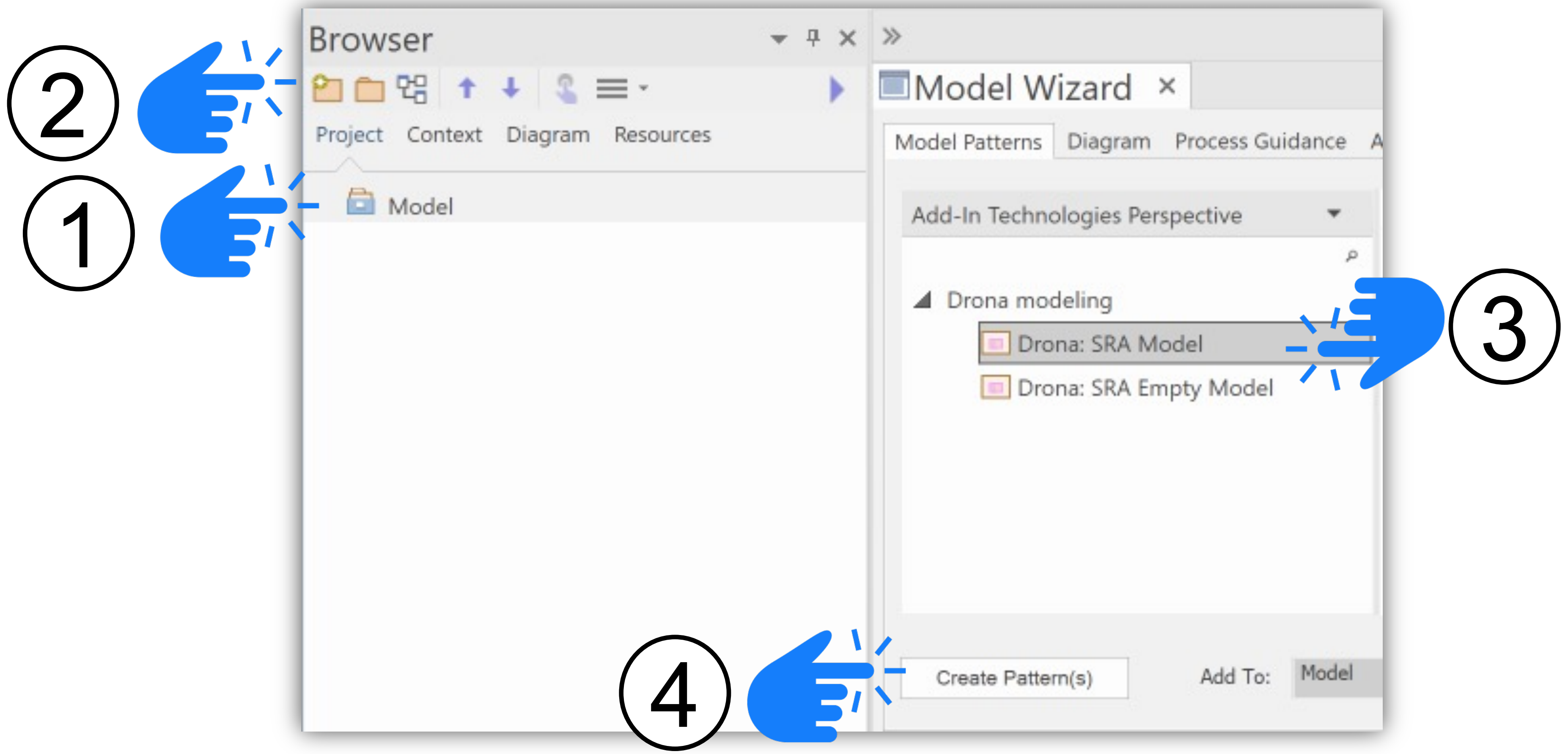
- **Goal:** Learn to create and edit an SRA model fast
- **Tasks:**
  1. Create a “Qualisist SRA Model” using the model wizard
  2. Open each package and compare with the SRA sections

Expected Result:



- Model
  - MT103 9x cash deadline
    - History
    - Main Contacts
    - Summary
    - Business Context, Goals & Objectives
    - Refined Project Scope
    - Current and Future Overview
    - Data Analysis
    - Functional Requirements
    - Transition Requirements
    - Non-Functional Requirements
    - General Information
    - Appendixes
    - Data Defintions
    - Actors
    - Relationships
    - Quantifiers
    - Units

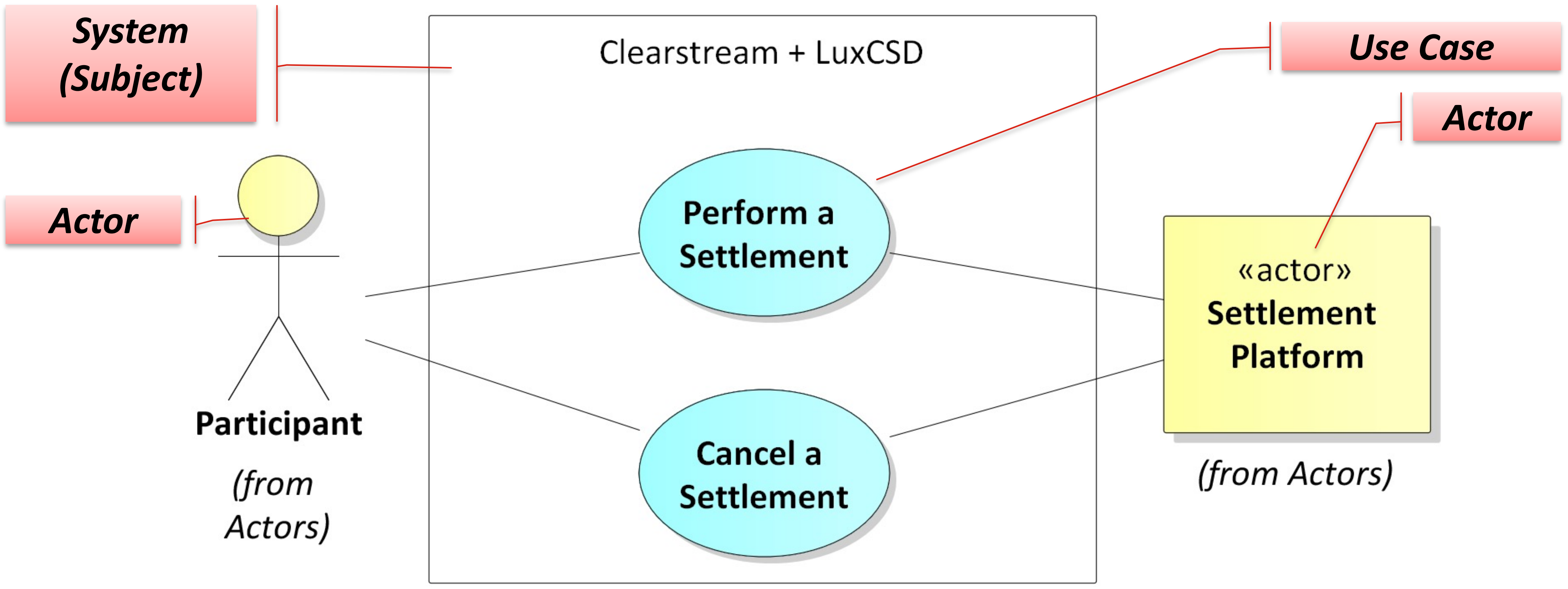
Steps to Create a Qualisist SRA Model Using the Wizard



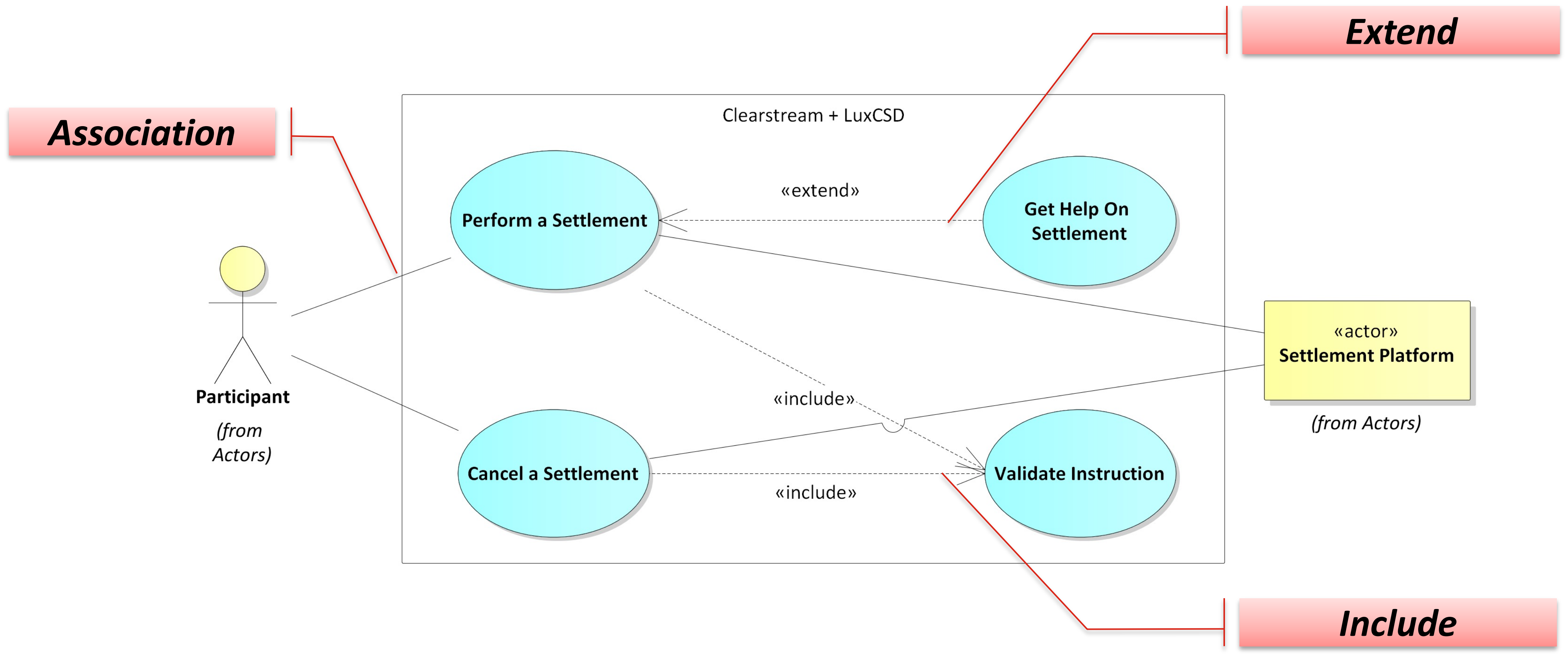
UML Use Case Diagrams

- We can use a Use Case diagram to answer the following questions:
  - What is being described? (The system)
  - Who interacts with the system? (The actors)
  - What can the actors do? (The use cases)

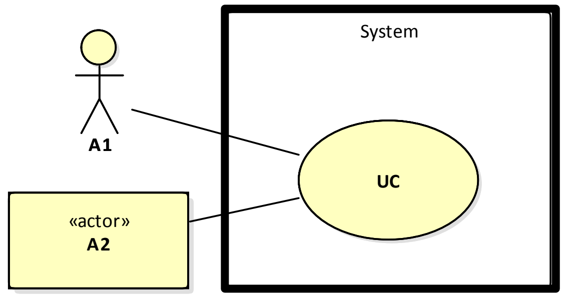
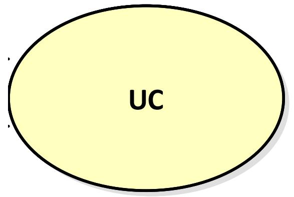
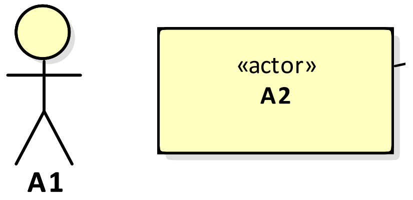
Example UML Use Case Diagram



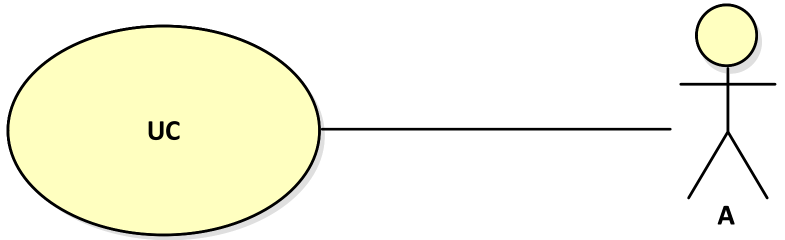
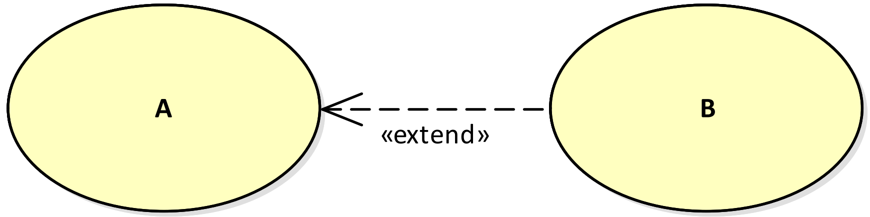
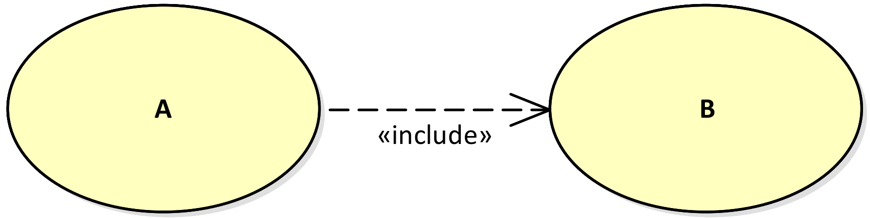
Example UML Use Case Diagram (Relationships)



Use Case Diagrams Notation used in Qualisist (1/2)

Name	Notation	Description
System Called “Subject” in UML (Boundary box notation)		Boundaries between the system and the users of the system
Use case		Unit of functionality of the system
Actor (Stickman notation and Class notation)		Role of the users of the system

Use Case Diagrams Notation used in Qualisist (2/2)

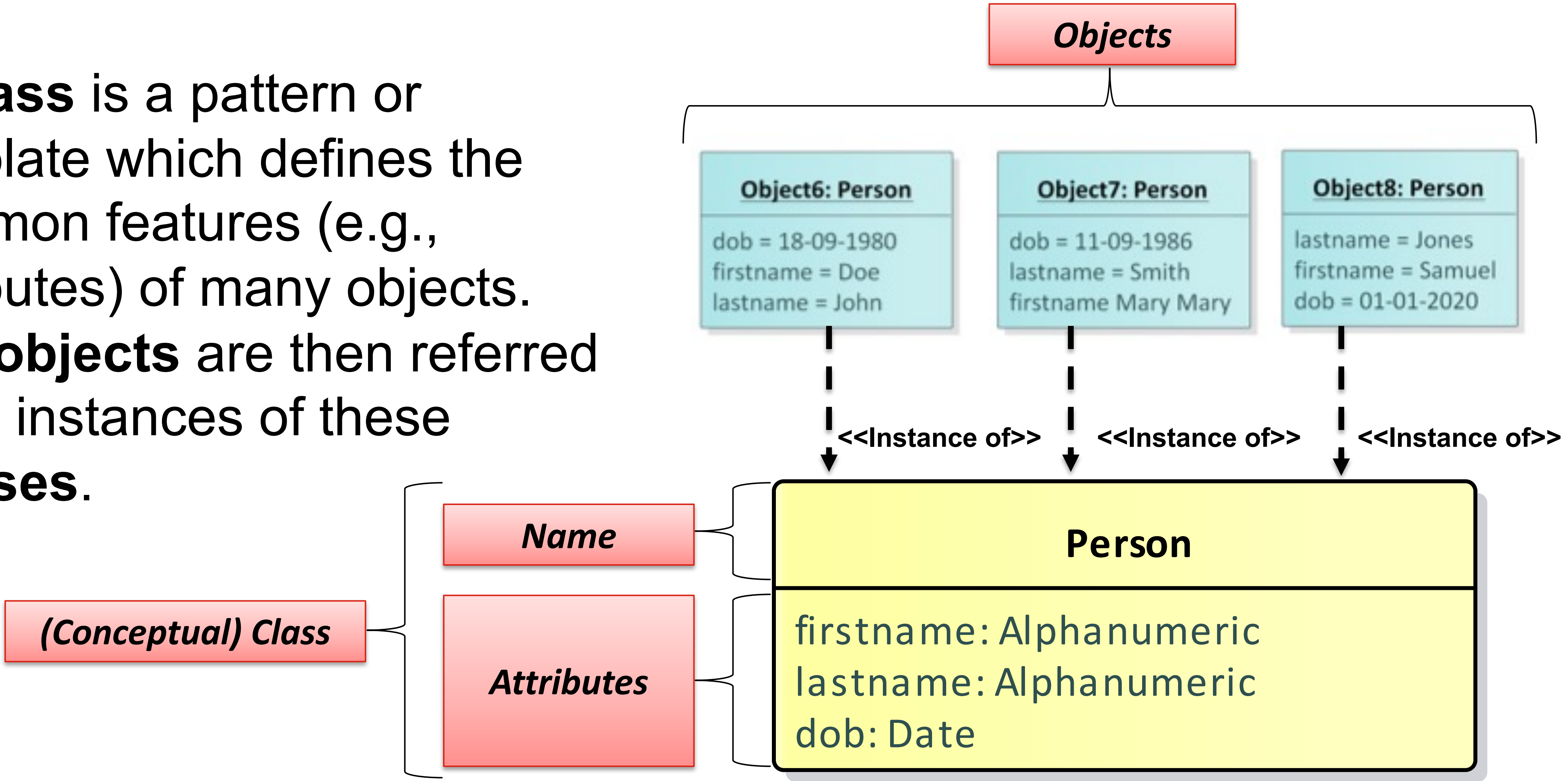
Name	Notation	Description
Association (In Use Case Diagrams)		Relationship between use cases and actors
Extend relationship		B extends A: optional use of use case B by use case A
Include relationship		A includes B: required use of use case B by use case A

Domain Models in Qualisist

- A **domain model** is a visual representation of:
  - Conceptual classes (meaningful real-world concepts or entities pertinent to the domain)
  - Associations between conceptual classes
  - Attributes of conceptual classes
- In Qualisist, domain models are expressed as UML class diagrams (CDs)

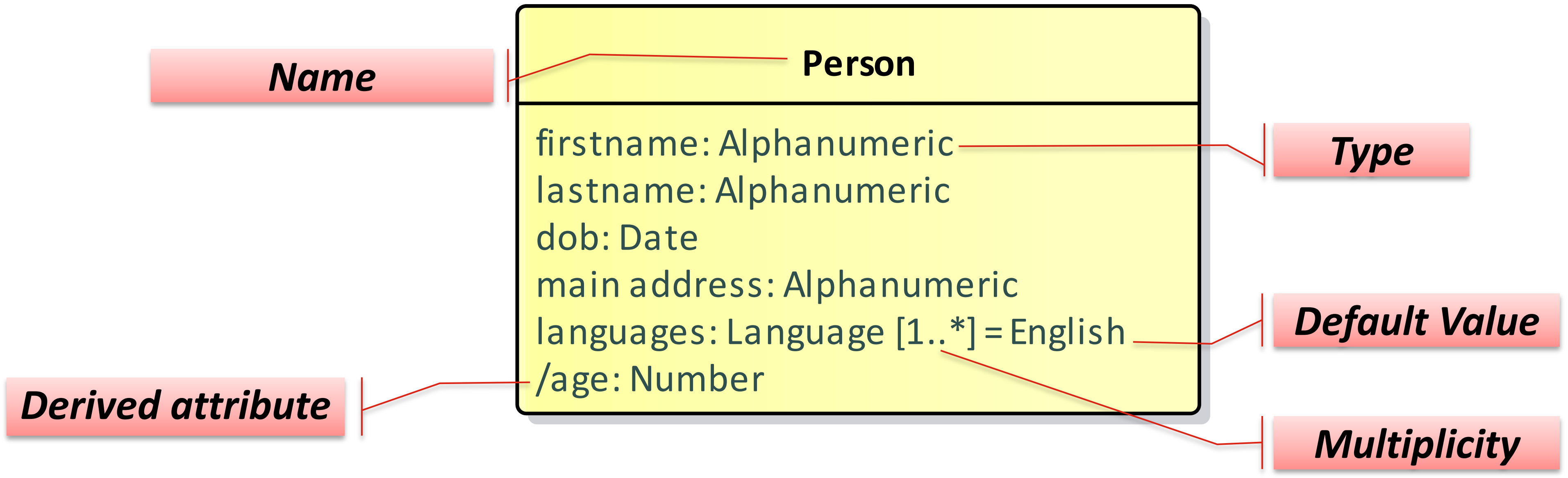
Conceptual Class and Objects

- A **Class** is a pattern or template which defines the common features (e.g., attributes) of many objects. The **objects** are then referred to as instances of these **classes**.

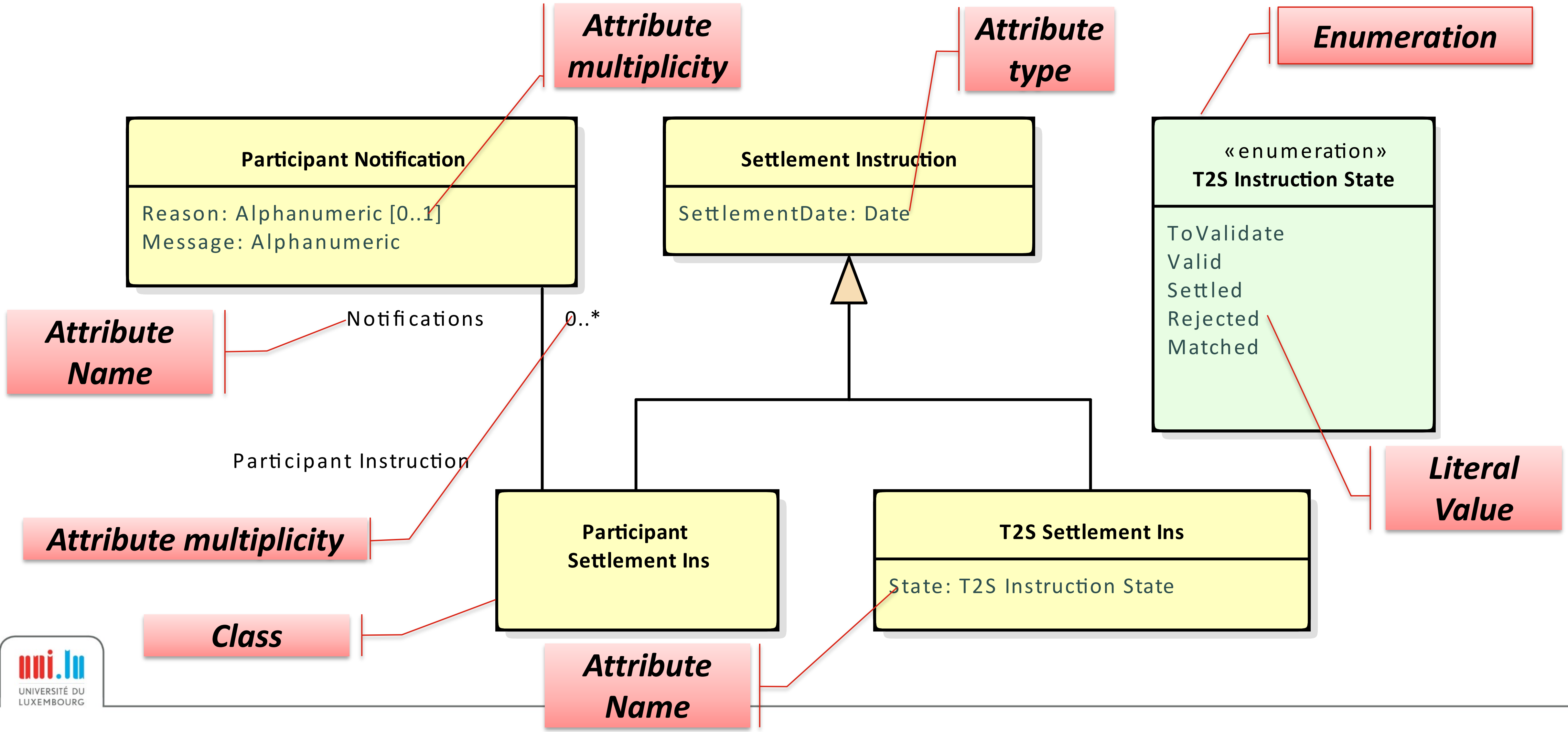


Basic Syntax for Attributes in a Class

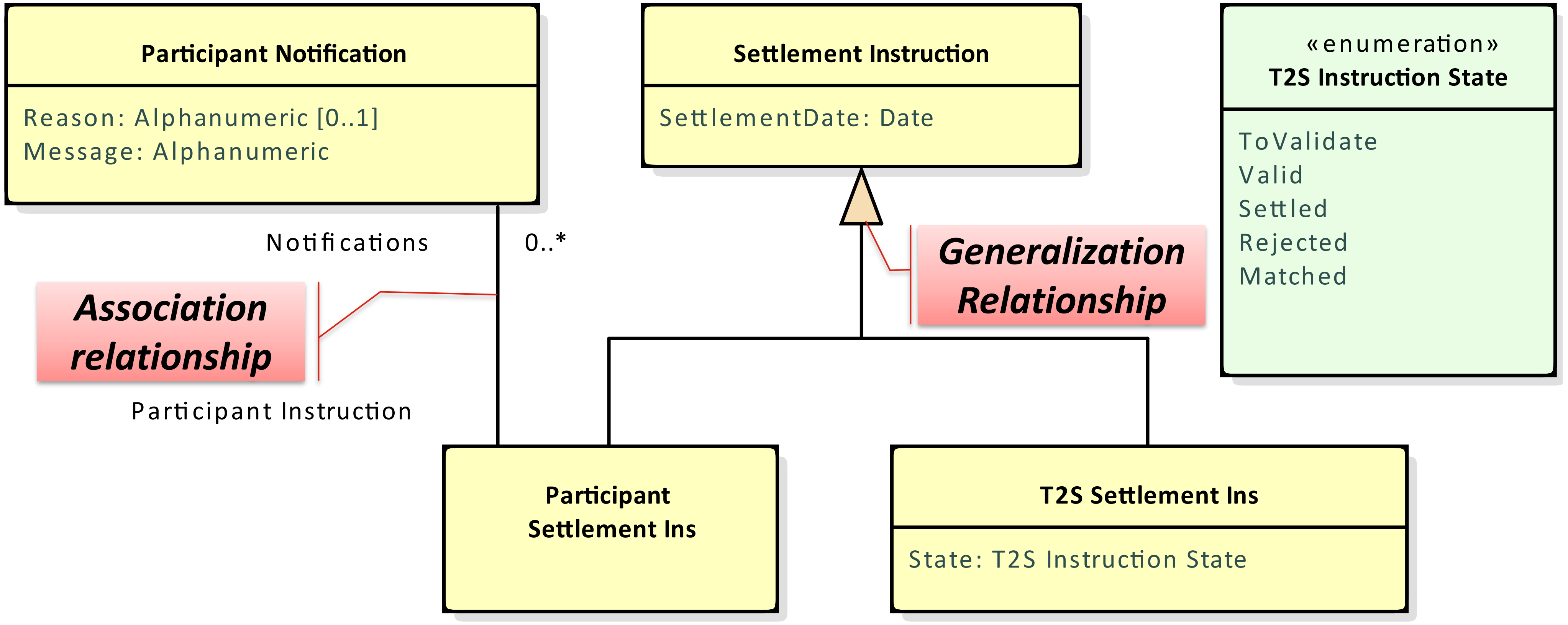
[/] Name [: Type] [Multiplicity] [= Default Value]



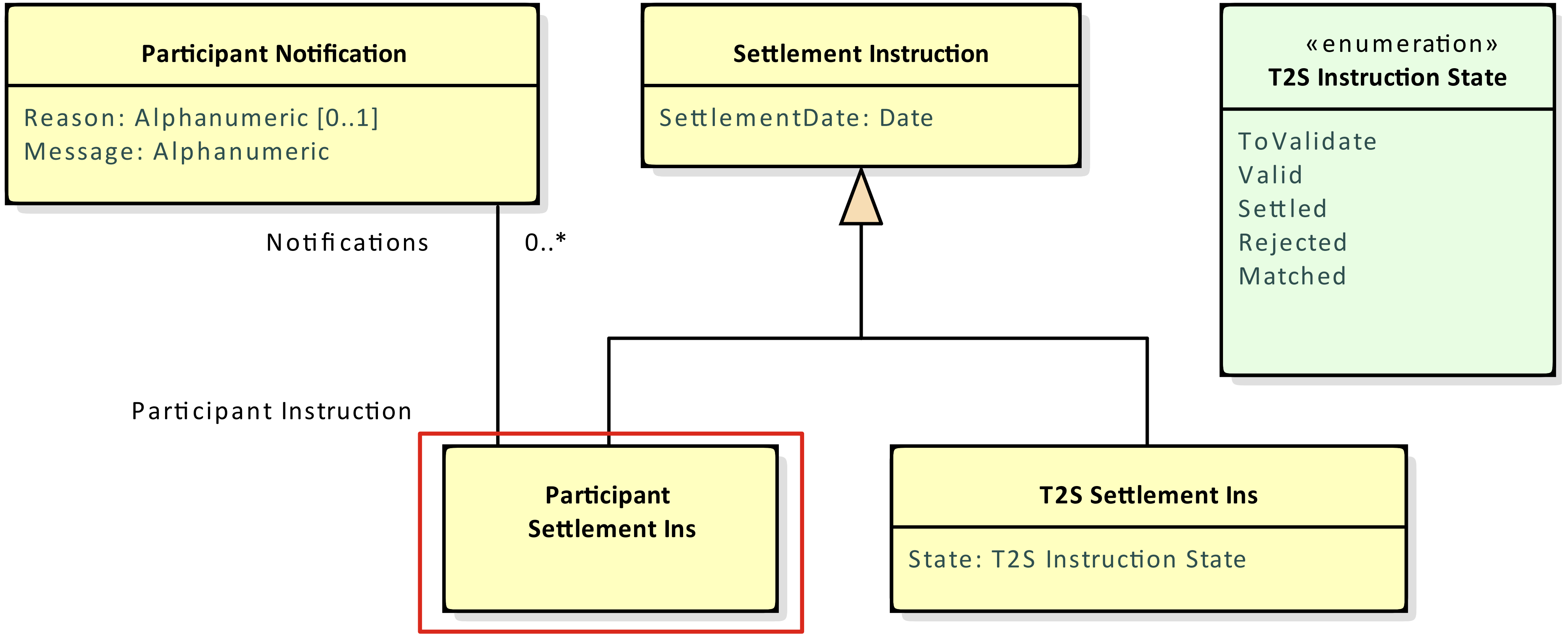
Example Domain Model



Example Domain Model (Relationships)



How Many Attributes does Participant Settlement Ins have?



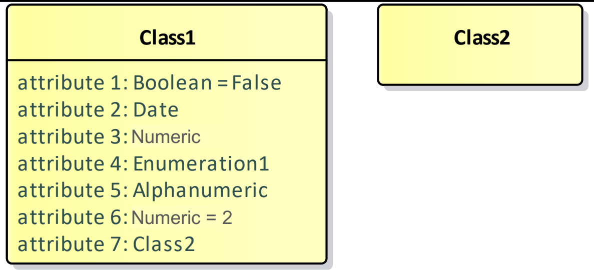
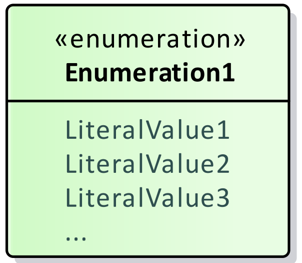
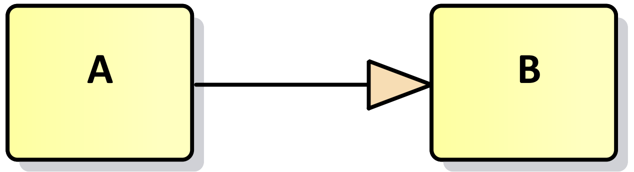
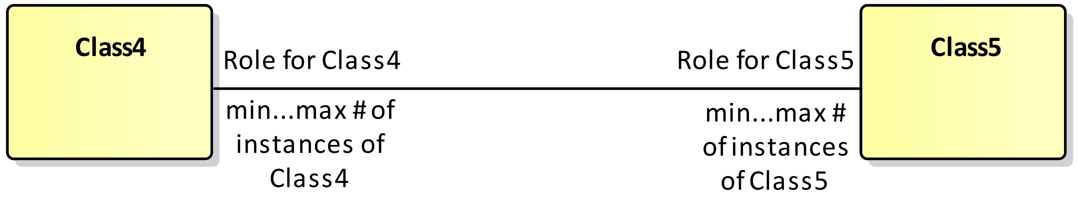
Predefined Attribute Types

- According to Clearstream, Qualisist should support four types of data attributes:
  - **Boolean**: Contain the value either true or false
  - **Date**: Contain a timestamp
  - **Alphanumeric**: Contain either numbers and/or alphabetical characters
  - **Numeric**: Contain only numbers (either integers or decimals)

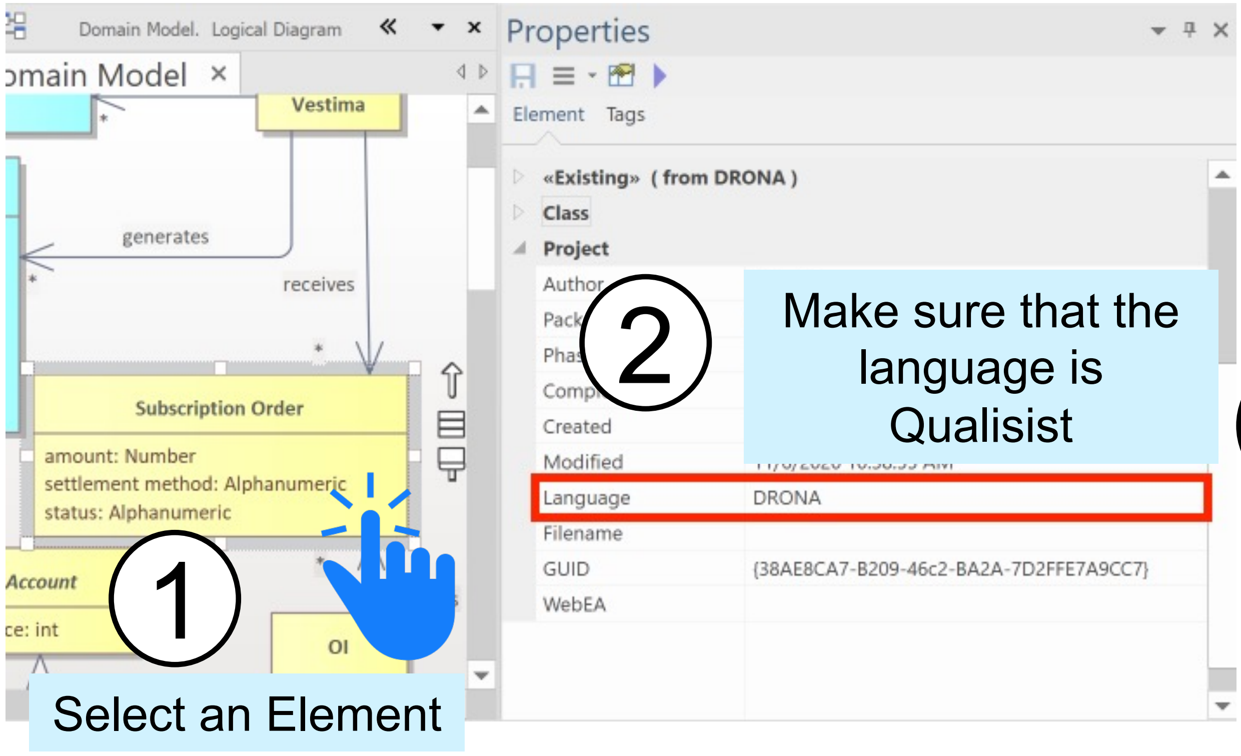
Points of Attention in Domain Modeling

- Use common terminology from your business domain
- Domain Models are built together with other diagrams to provide descriptions of the types that they use
- Use singular nouns for the names of classes, e.g., use Instruction/Account instead of Instruction**s**/Account**s**
- Use the attribute types predefined by Clearstream in Qualisist (i.e., Boolean, Date, Alphanumeric, Numeric)

Domain Models Notation used in Qualisist

Name	Notation	Description
Class	 Class1 attribute 1: Boolean = False attribute 2: Date attribute 3: Numeric attribute 4: Enumeration1 attribute 5: Alphanumeric attribute 6: Numeric = 2 attribute 7: Class2 Class2	Description of the structure and behavior of a set of objects
Enumeration	 «enumeration» Enumeration1 LiteralValue1 LiteralValue2 LiteralValue3 ...	A type that has a limited number of values.
Generalization		The specialising or sub-type (A) inherits attributes and associations of the general or base type (B)
Association	 Class4 Role for Class4 min...max # of instances of Class4 Role for Class5 min...max # of instances of Class5 Class5	Relationship between classes

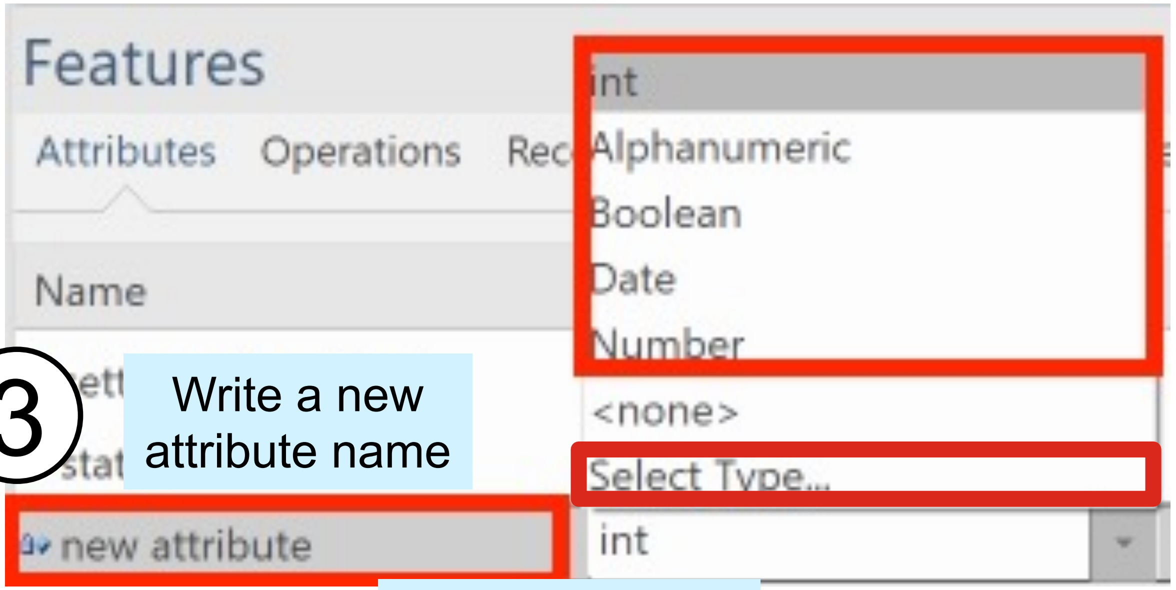
Steps to Create Attributes Using Predefined Types



Language: DRONA

1 Select an Element

2 Make sure that the language is Qualisist



3 Write a new attribute name

4 Select one predefined type among Alphanumeric, Boolean, Date, and Number

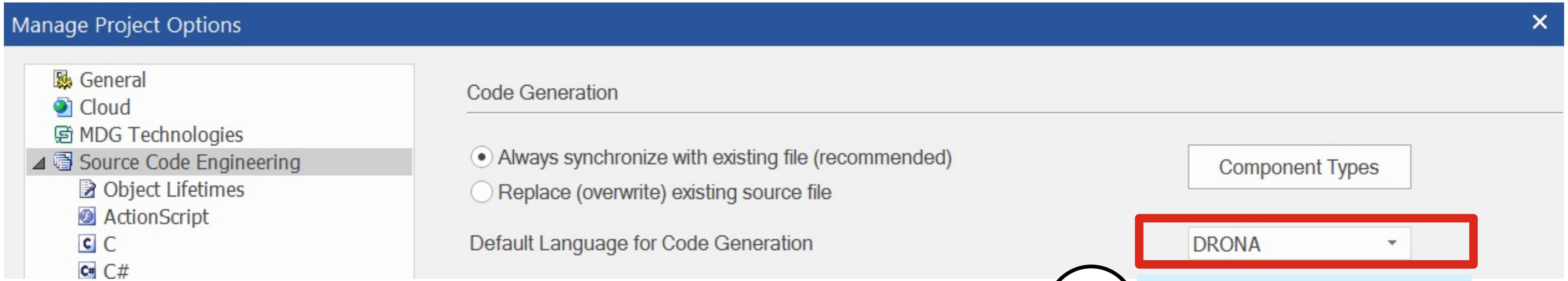
Or

Select an existing Enumeration or Class from the diagram

Steps to Define a Default Language for New Classes

- Follow the news steps if Qualisist does not appear as the default language

1 Go to Configure → Options → Source Code Engineering → Code Generation section → Default Language for Code Generation


Manage Project Options

Code Generation

(● Always synchronize with existing file (recommended))
○ Replace (overwrite) existing source file

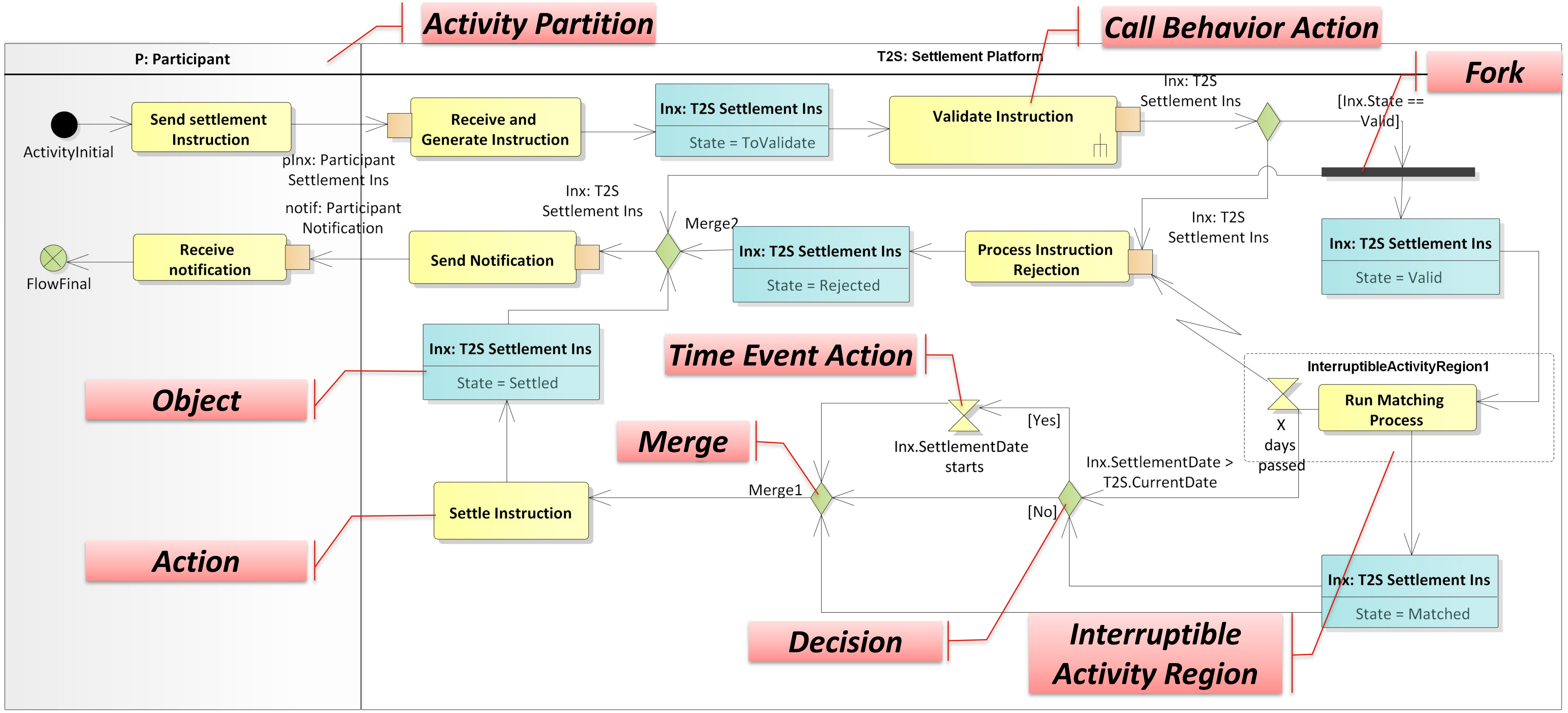
Default Language for Code Generation: DRONA

2 Make sure that the language is Qualisist

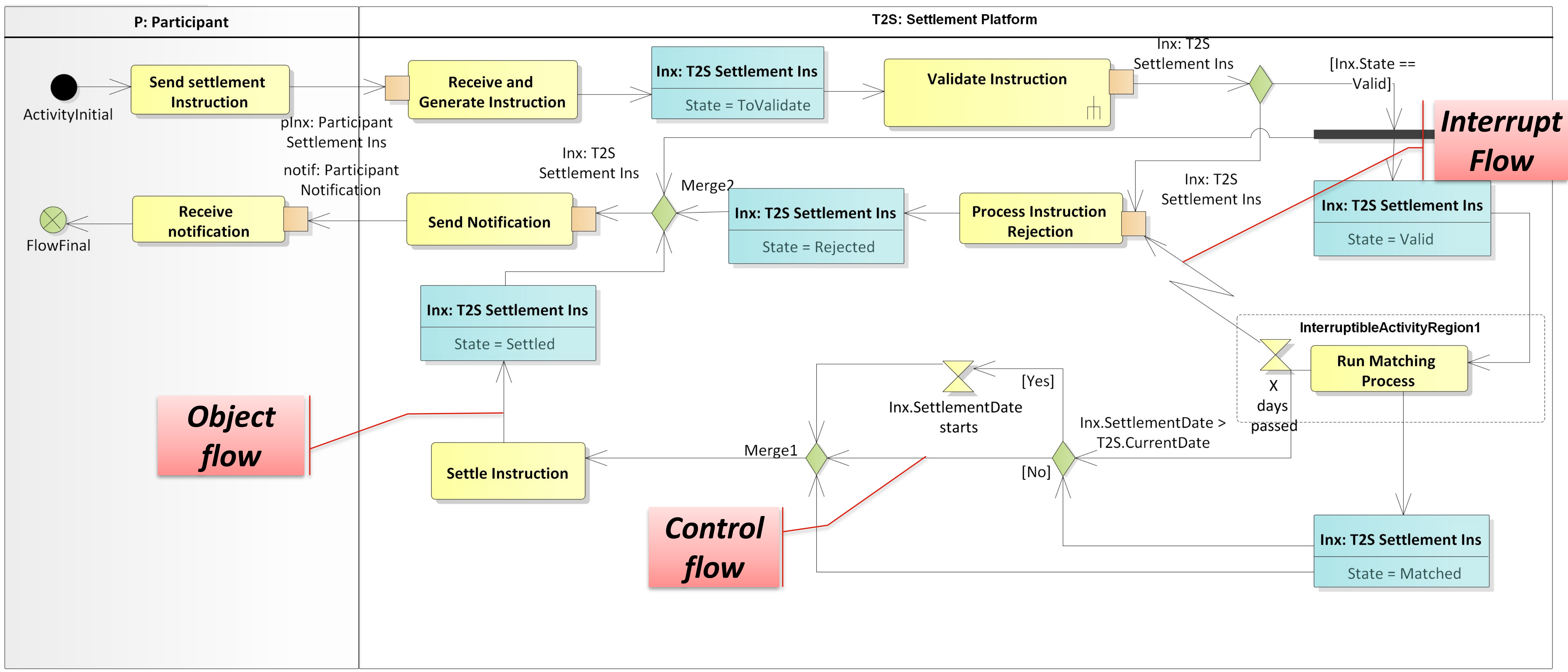
Activity Diagrams

- An activity diagram is a directed graph composed of Nodes and Edges
- Control flow and object flow define the execution order
- In Qualisist, Activity diagrams should
  - Be created along with the domain model
  - Include control flow and object flow
  - Be annotated with Pre- and Post-conditions
  - Include Activity Partitions

Example Activity Diagram (Nodes)

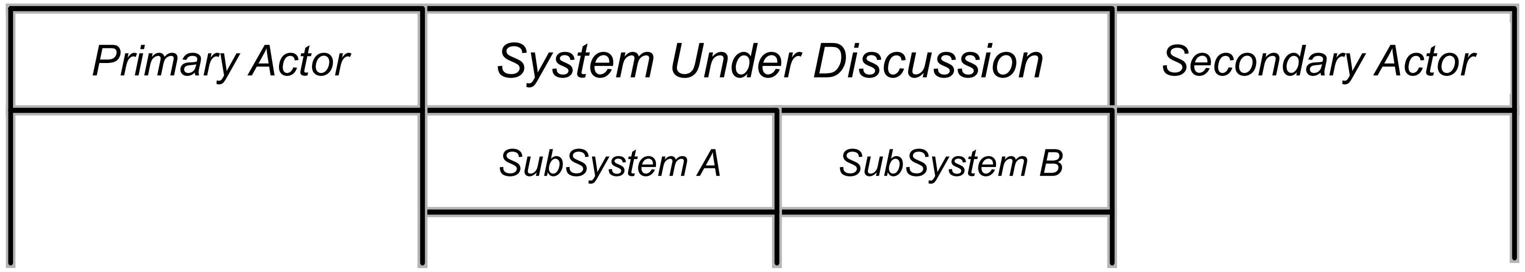


Example Activity Diagram (Edges)

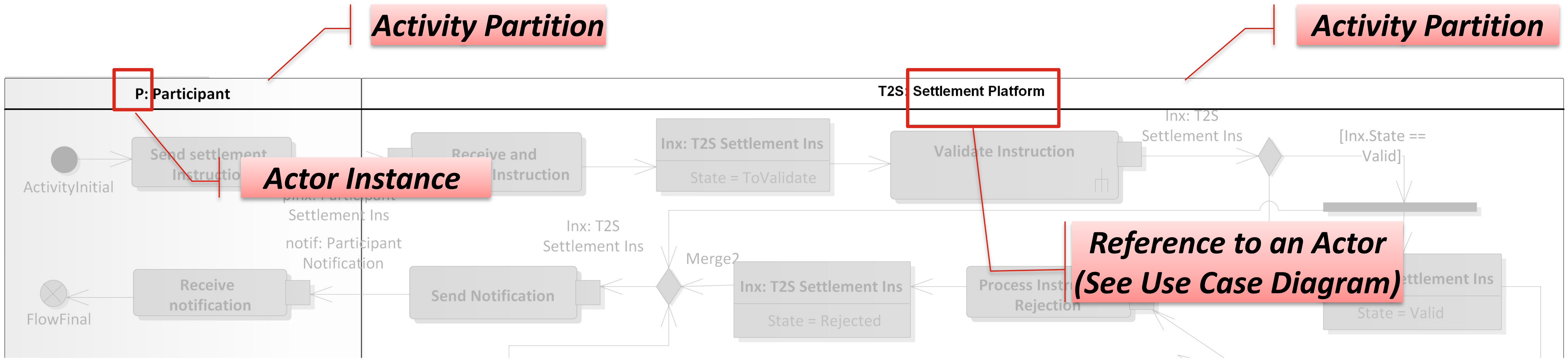


Activity Partition

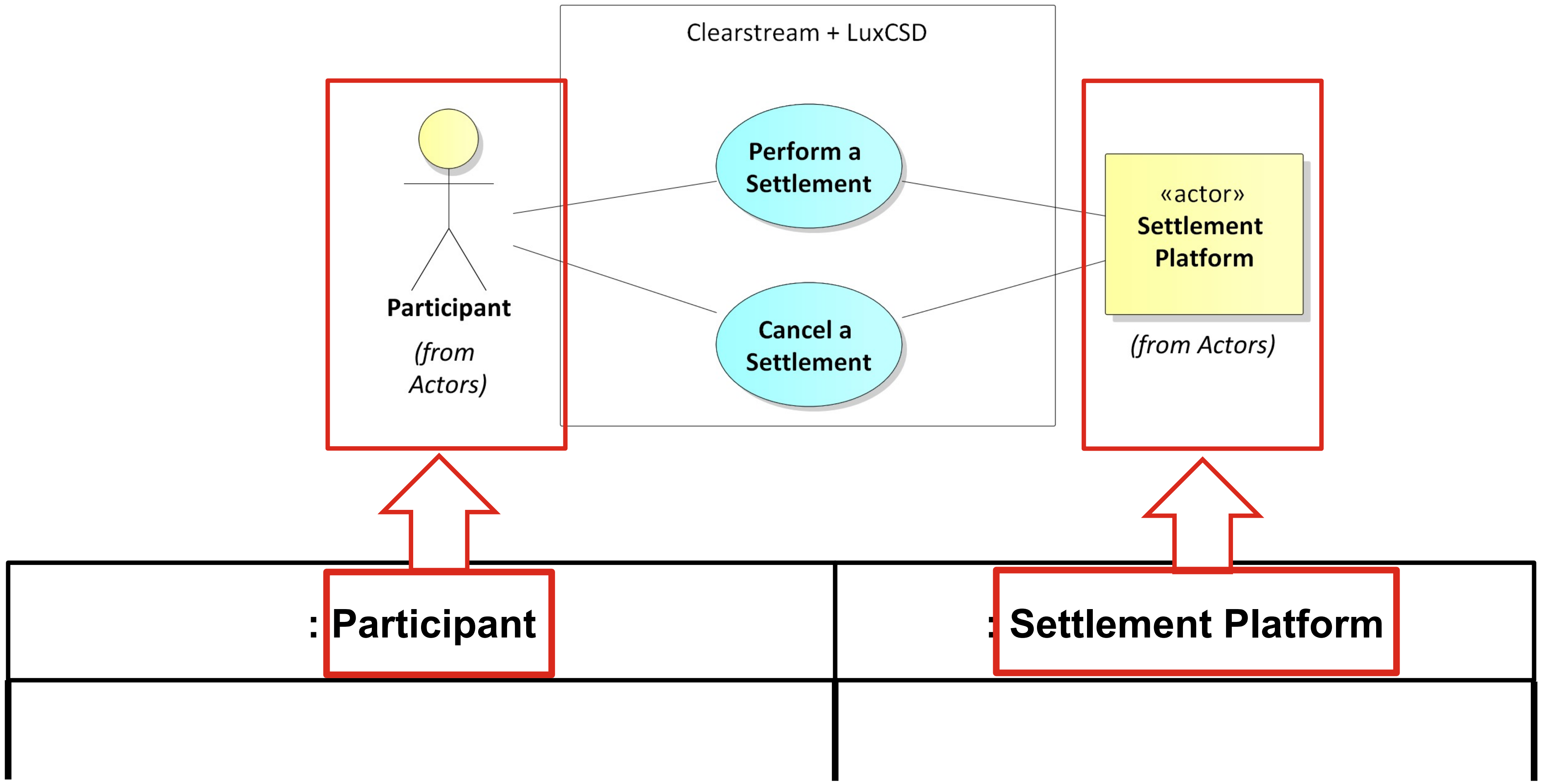
- Allows the grouping of nodes and edges of an Activity due to responsibilities
- Makes the activity diagram more structured
- In Qualisist, each Activity Partition must correspond to an Actor from the Use Case diagram



Example Activity Partitions Related to Actors

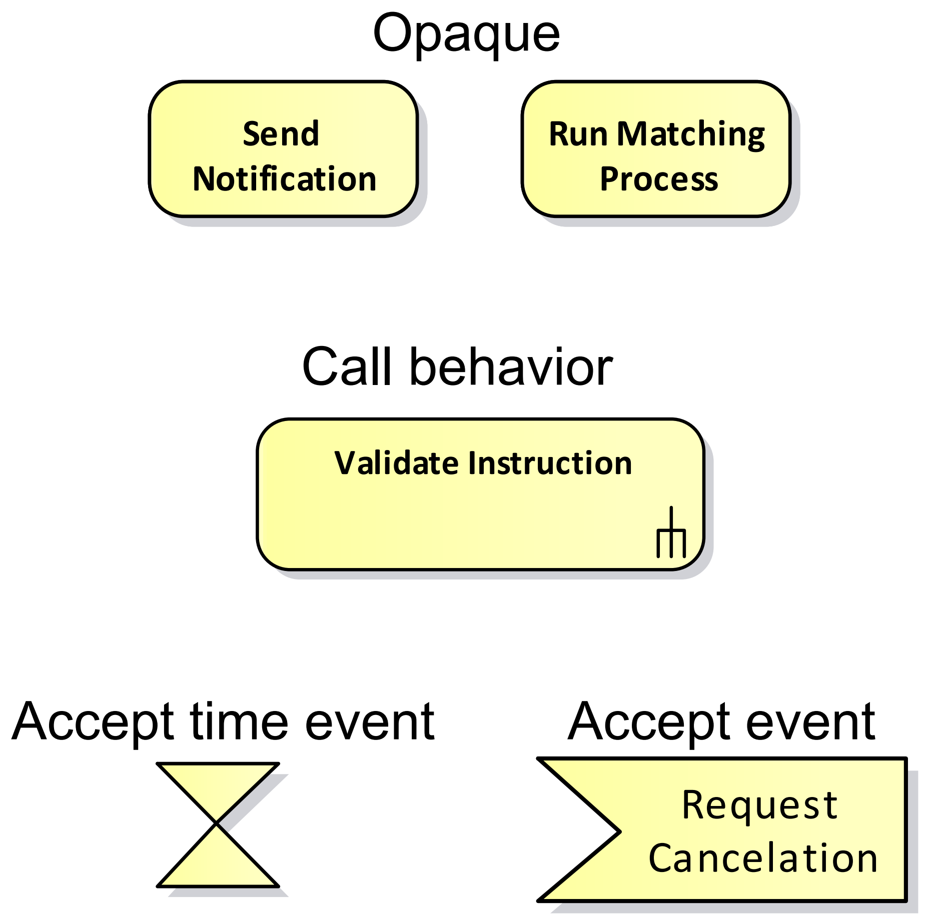


Question: Which Actors are Referenced in the Example?

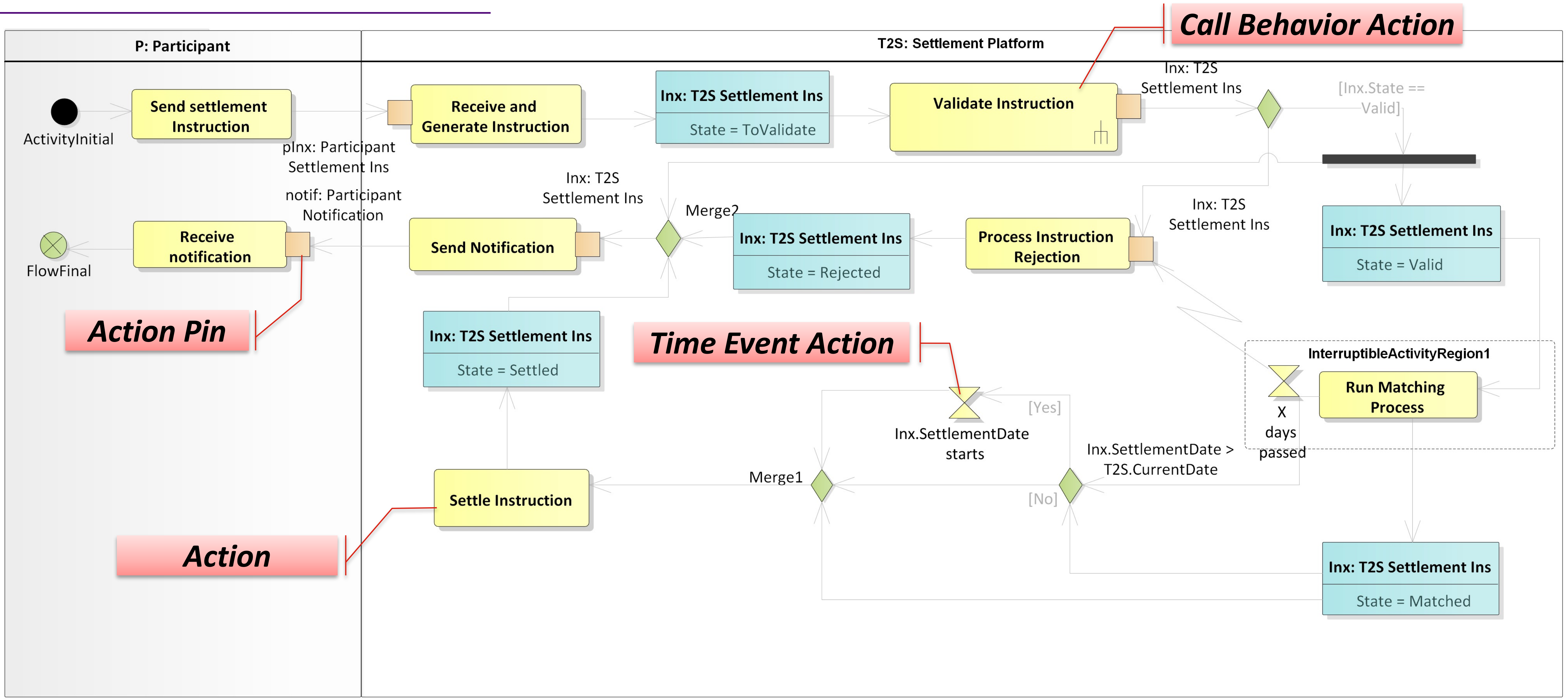


Action

- **Basic element** to specify user-defined behavior
- Process input values to produce output values
- Special notation for predefined types of actions, for example:
  - Opaque Actions
    - Atomic behavior
  - Call behavior action
    - Behavior is described in another AD
  - Event-based actions

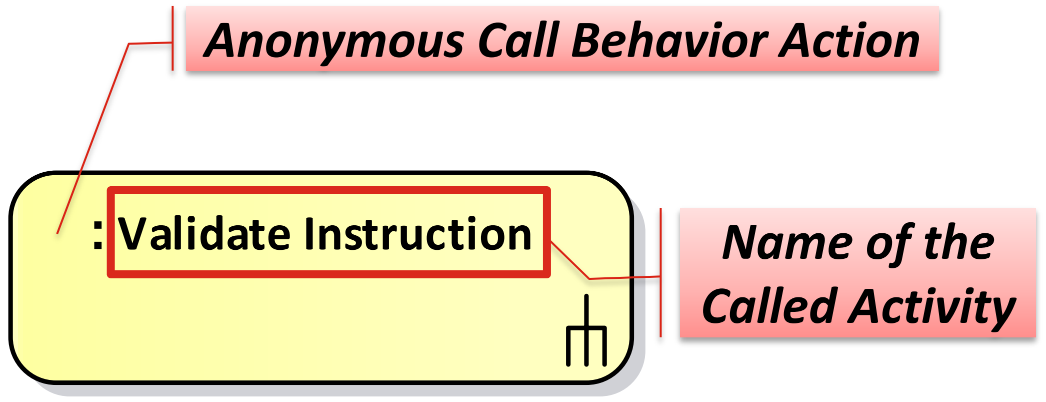


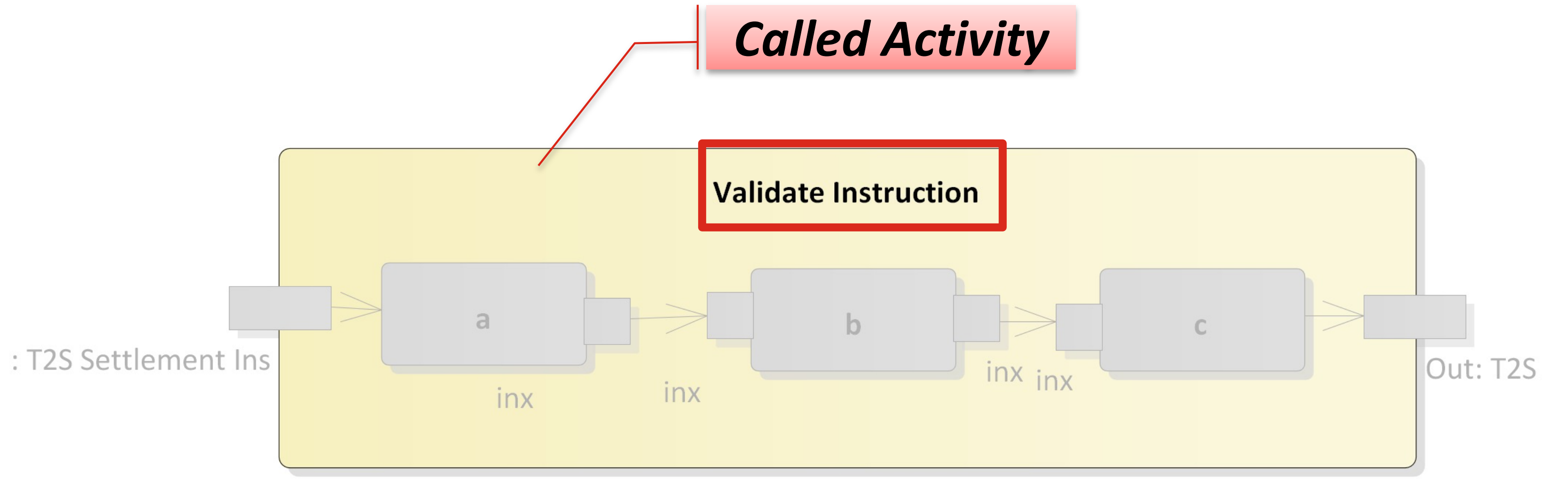
Example of Actions in Activity Partitions



Call Behavior Action

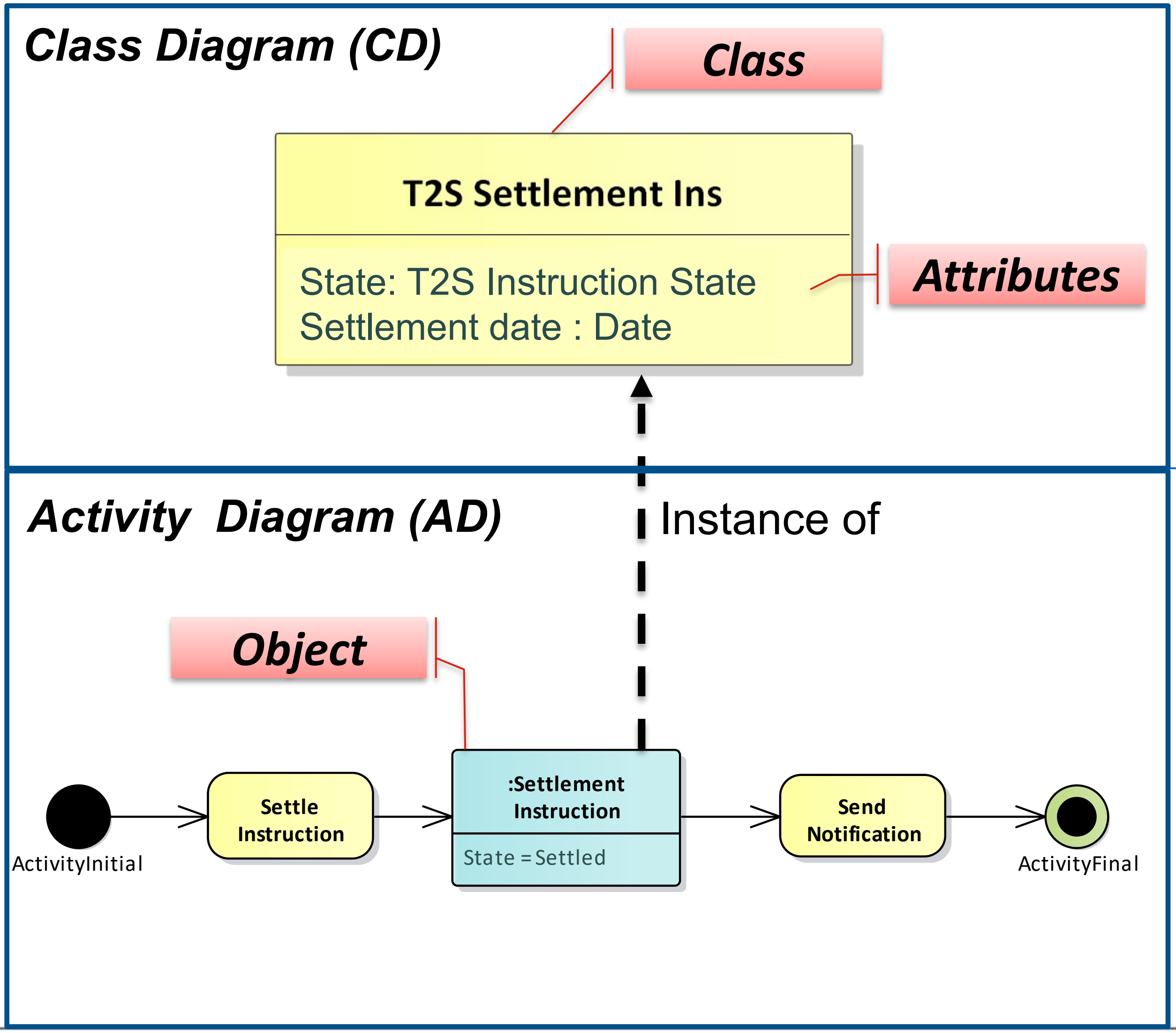
- The execution of an Action calls an Activity
- In Qualisist, an Activity Diagram specifies the behavior of the called Activity
- Advantages:
  - Model becomes clearer
  - Reusability





Object (1/3)

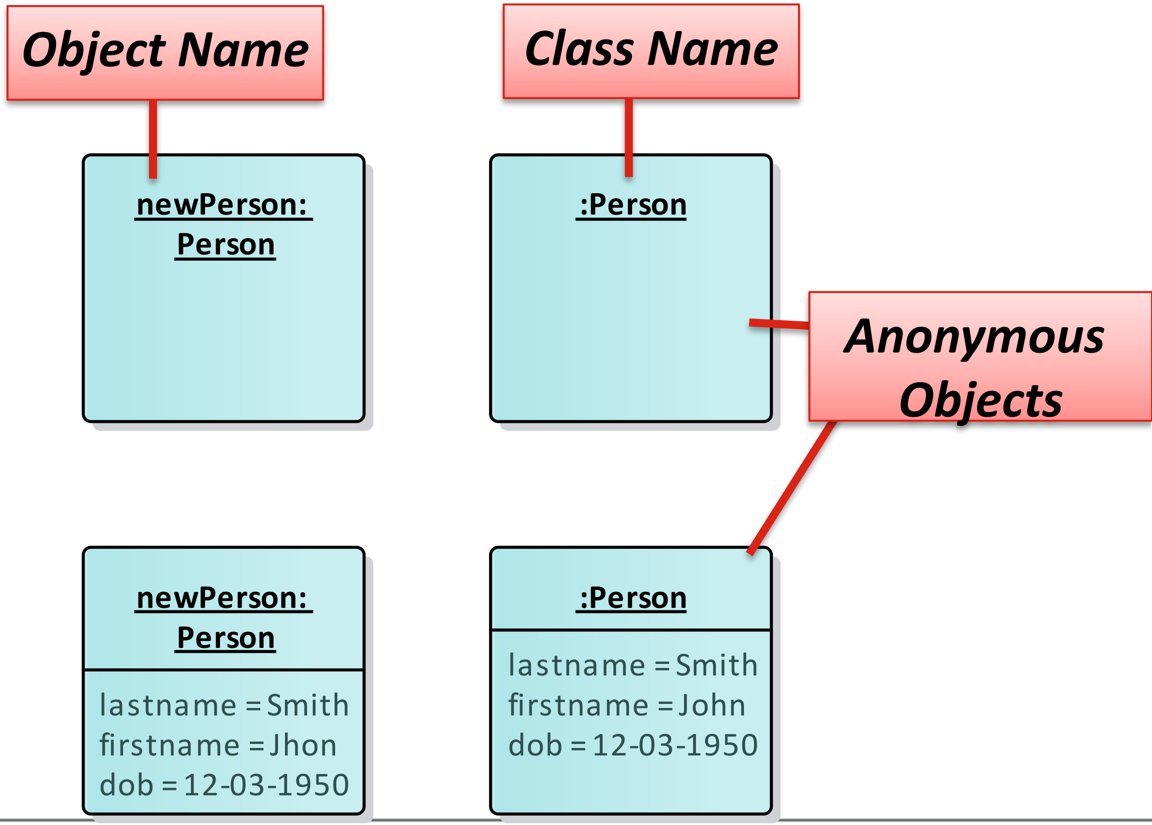
- **Object** is an **instance** of a **class**
- Not all the properties of the Class have to be represented in the Object
  - **Example:** The T2S Settlement Ins Class in the CD has two properties (State and Settlement date).



Object (2/3)

- **Objects** could have a **name** or be **anonymous**
- **Object name** allows to distinguish the instance from other instances

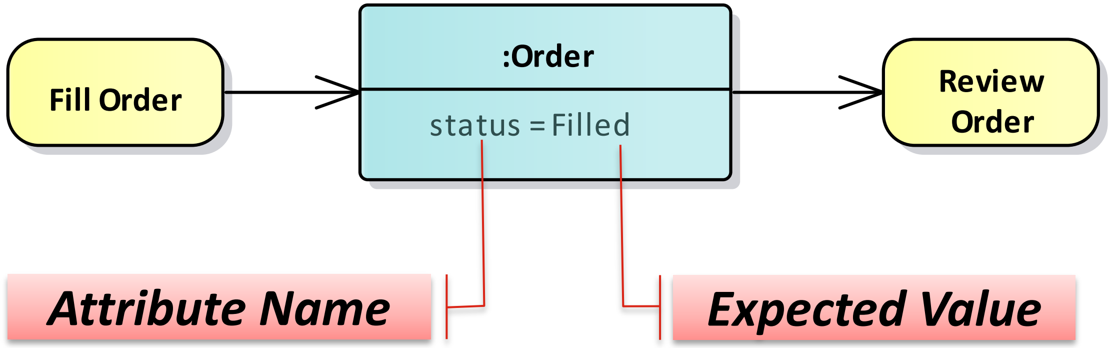
- **Example:** Notation for an Object of the Person class



Object (3/3)

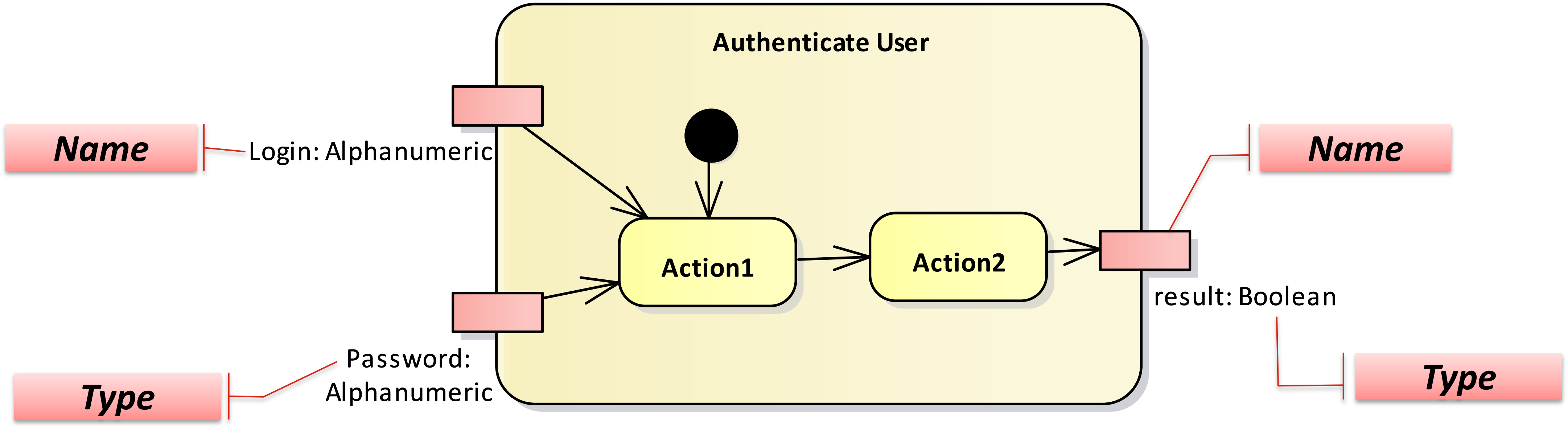
- Is the source and target of an object flow edge
- At run-time, an Object can have specific values for its attributes or exist in a particular state.

- **Example:** The value of the attribute *status* of an *Order* must have the value *Filled* after the execution of the action *Fill Order*



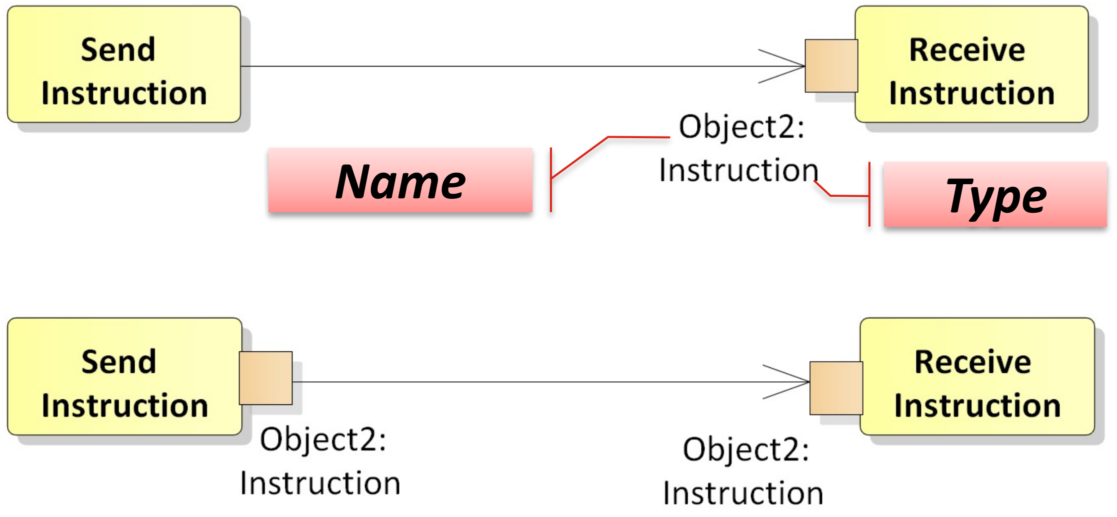
Representations of an Object (1/4)

- For Activities
  - Activity Parameter Node

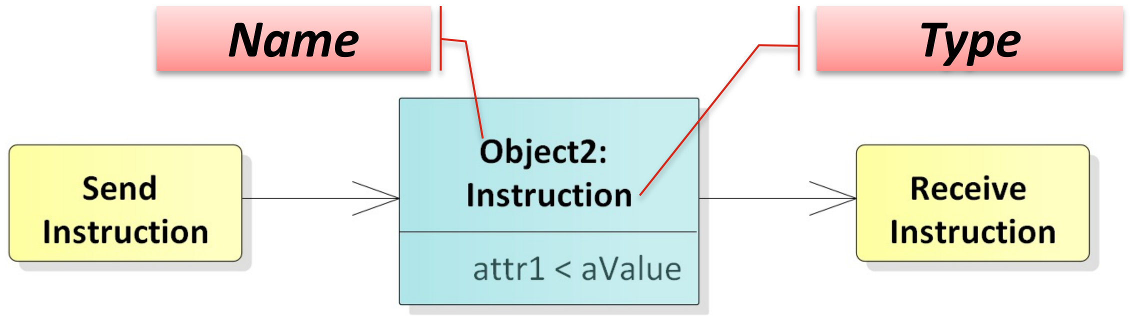


Representations of an Object (2/4)

- For actions
  - Action Pin



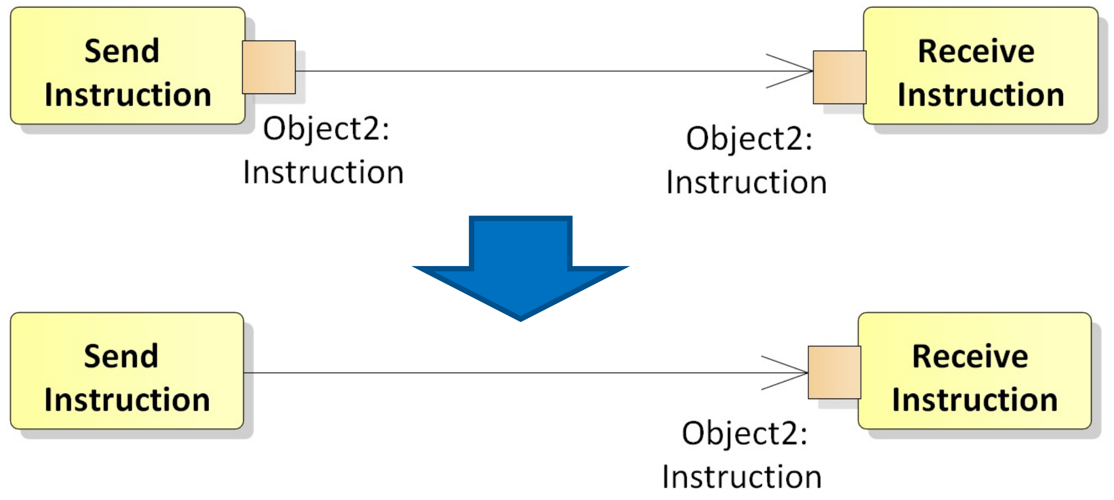
  - Object Node (Rectangle Notation)



Representations of an Object (3/4)

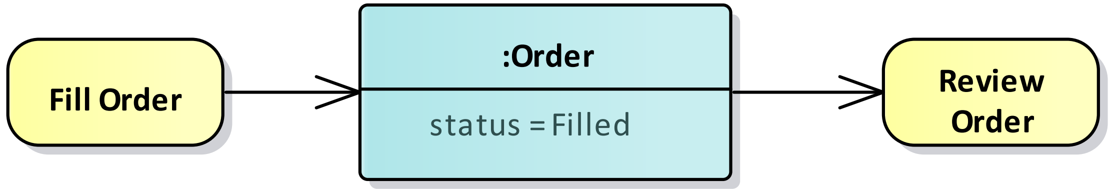
- **Action Pin**
  - **Input Pin** provides values to the Action, whereas an **Output Pin** contains the results from that Action
  - Useful to Save space in the diagram
  - In Qualisist, use action pins when there are no object state changes

- **Example:** We omitted the output pin of *Send Instruction* because it is the same object received by Received Instruction

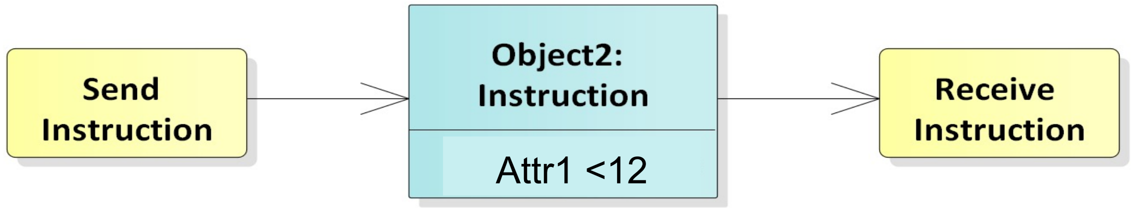


Representations of an Object (4/4)

- **Object Node (Rectangle notation)**
  - Useful to model the varying behavior of objects at run-time
  - Run state is defined in three parts
    1. Attribute name,
    2. Operator, e.g., =, <, !=, or any other user-defined operator, and
    3. Value, e.g., a number, a literal value



The status value must be equal to *Filled* after executing the action Fill Order

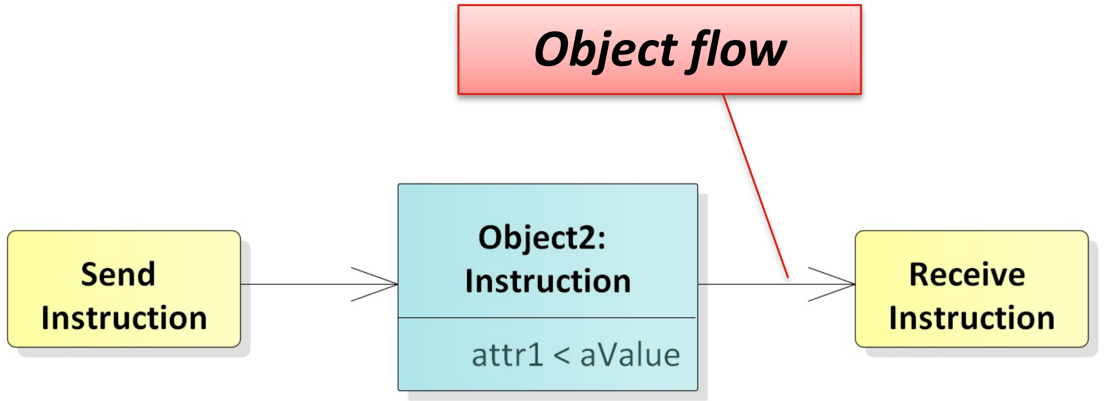


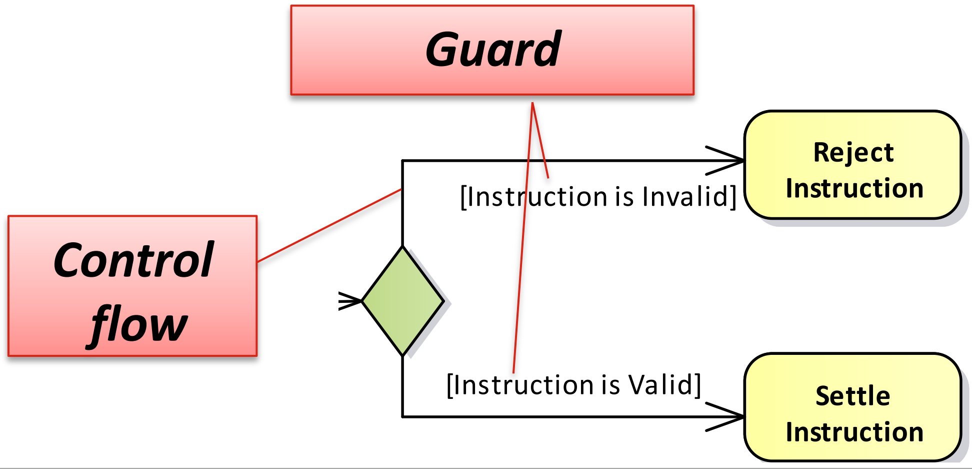
attr1 must not be greater than 12 after executing Send Instruction

Edge

- Connect nodes
- Express the execution order
- Types
  - Control flow edges
    - Define the order between nodes
  - Object flow edges
    - Used to exchange data or objects
    - Express a data dependency between nodes
- Guard (condition)
  - Control and object flow only continue if guards in square brackets evaluate to true

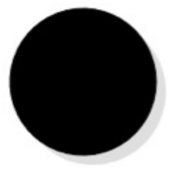
- **Examples:**

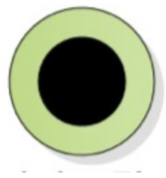


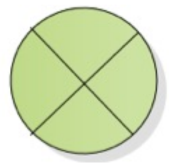


Beginning and Termination of Activities

- Initial node
  - Starts the execution of an activity
  - Provides tokens at all outgoing edges
  - Keeps tokens until the successive nodes accept them
- Activity final node
  - Ends all flows of an activity
  - First token that reaches the activity final node terminates the entire activity
  - Other control and object tokens are deleted
- Flow final node
  - Ends one execution path of an activity
  - All other tokens of the activity remain unaffected



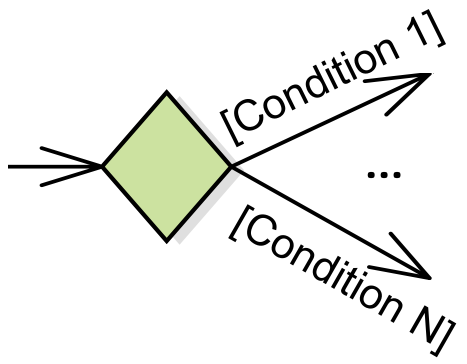




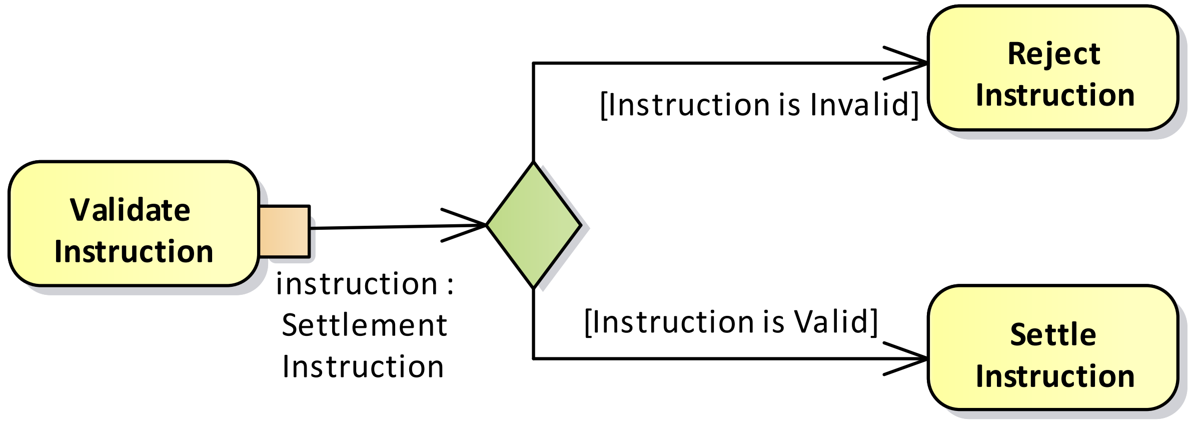
Alternative Paths – Decision Node

- Use to define alternative branches
- Outgoing edges have guards
  - Syntax: [Boolean expression]
  - Token takes **one** branch
  - <u>Guards must be mutually exclusive</u>

- **Notation:**



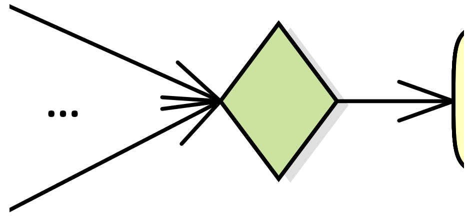
- **Example:**

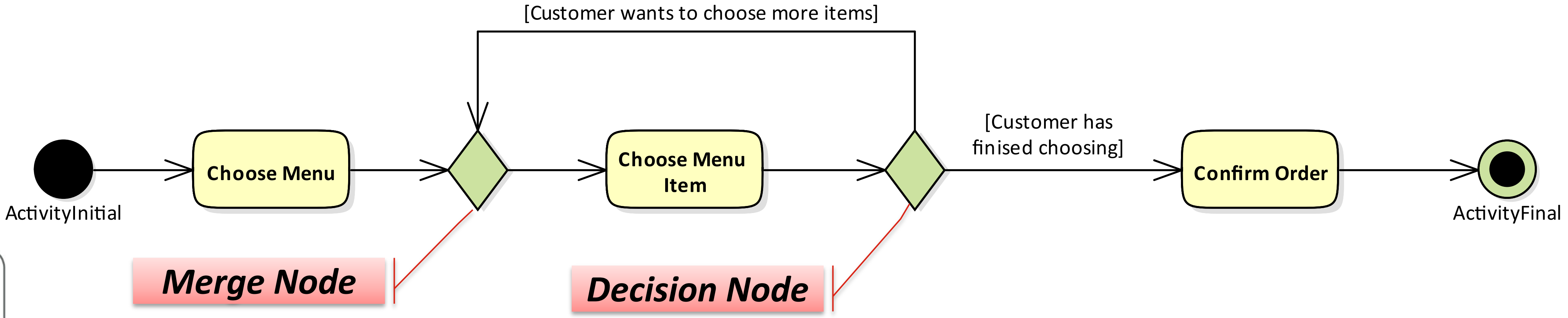


Alternative Paths – Merge Node

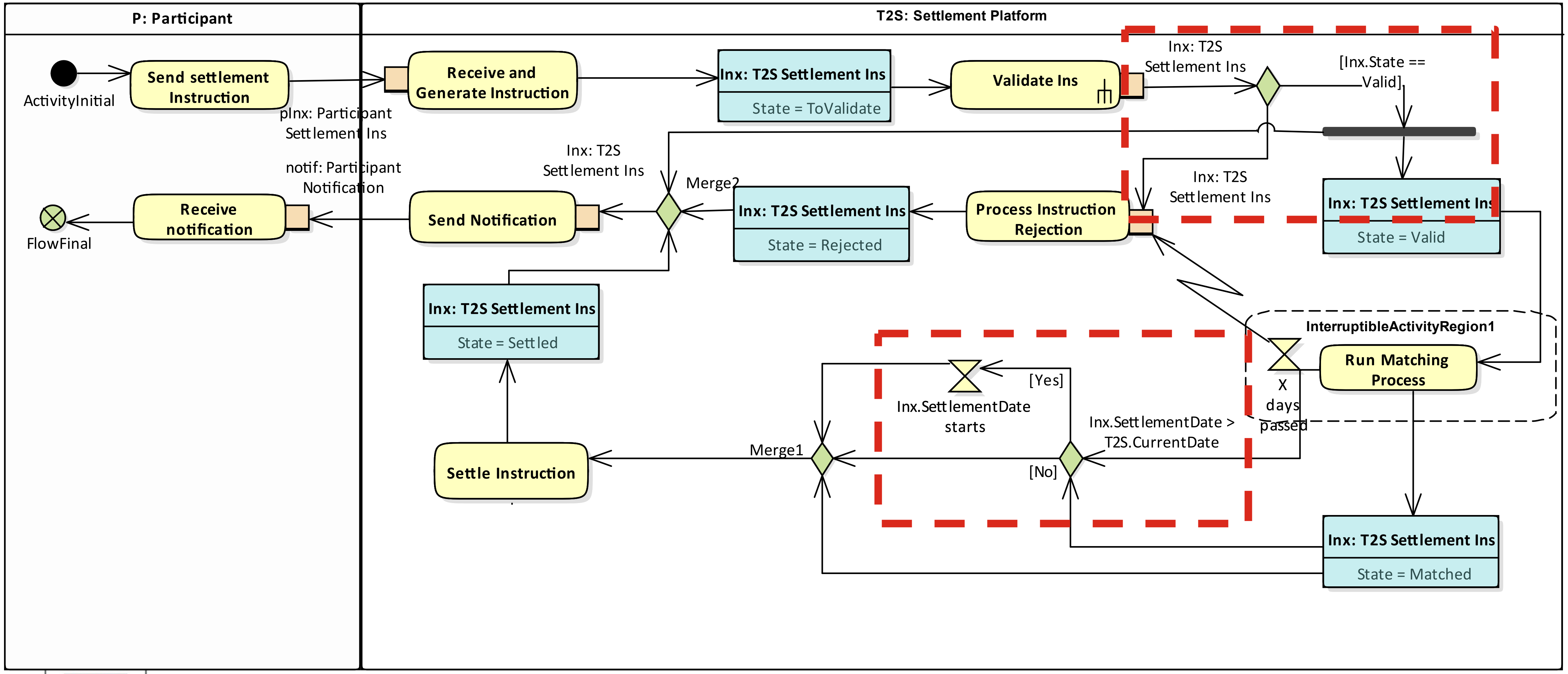
- To bring **alternative** sub-paths together
- **Passes** token to the next node
- **Example:** Decision and merge nodes used to model loops

- **Notation:**



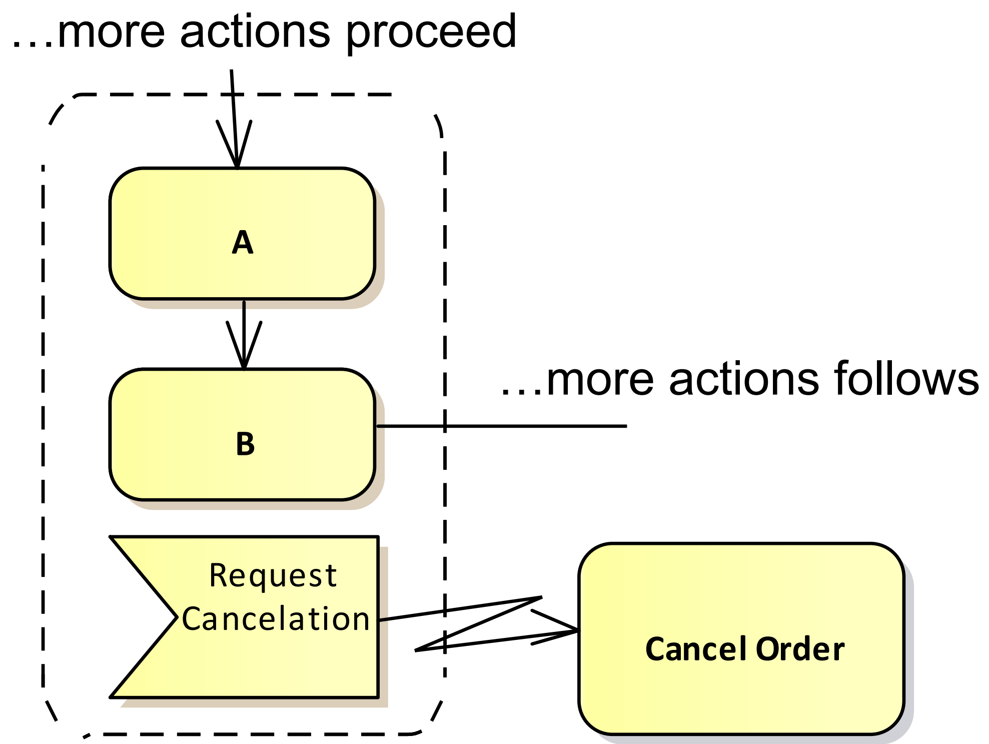


Question: What do the two Decision Nodes mean?

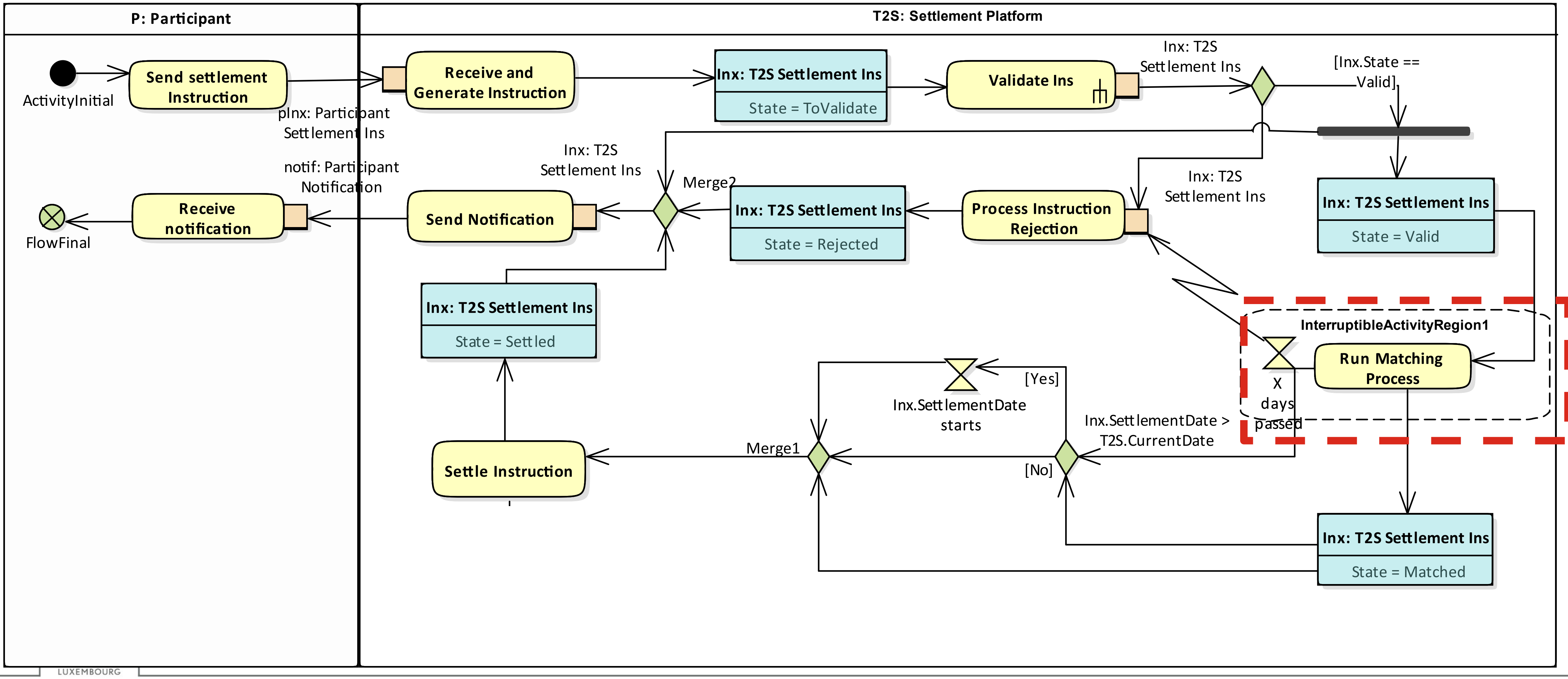


Exception Handling– Interruptible Activity Region

- Define a group of actions whose execution is to be terminated immediately if a specific event occurs. In that case, some other behavior is executed
- **Example:** If a **Cancel Request** occurs while **A** and **B** are executed
  - Exception handling is activated
  - All control tokens within the dashed rectangle are deleted
  - Action **Cancel Order** is activated and executed
  - No jumping back to the regular execution



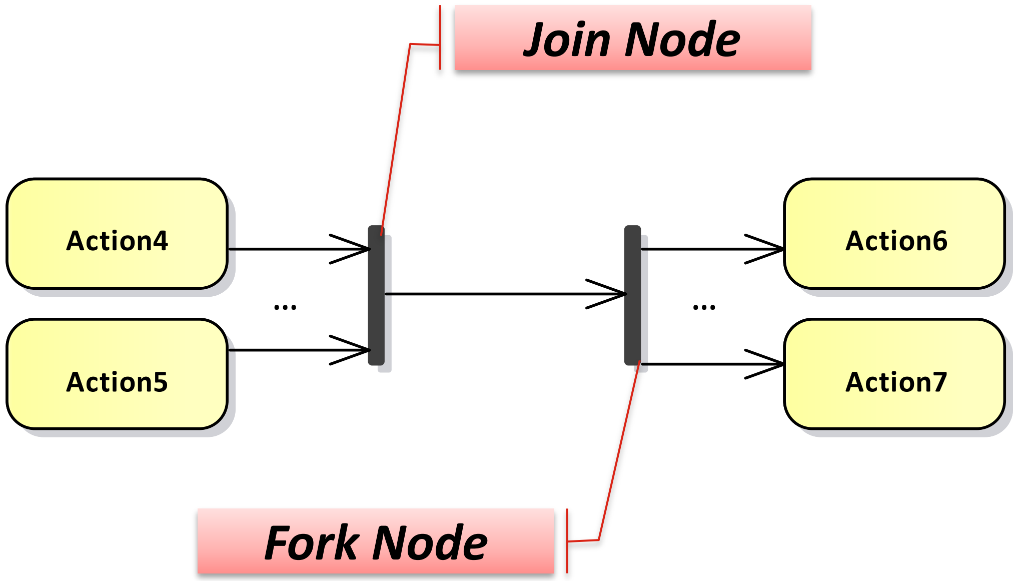
What does the Interruptible Activity Region Mean?



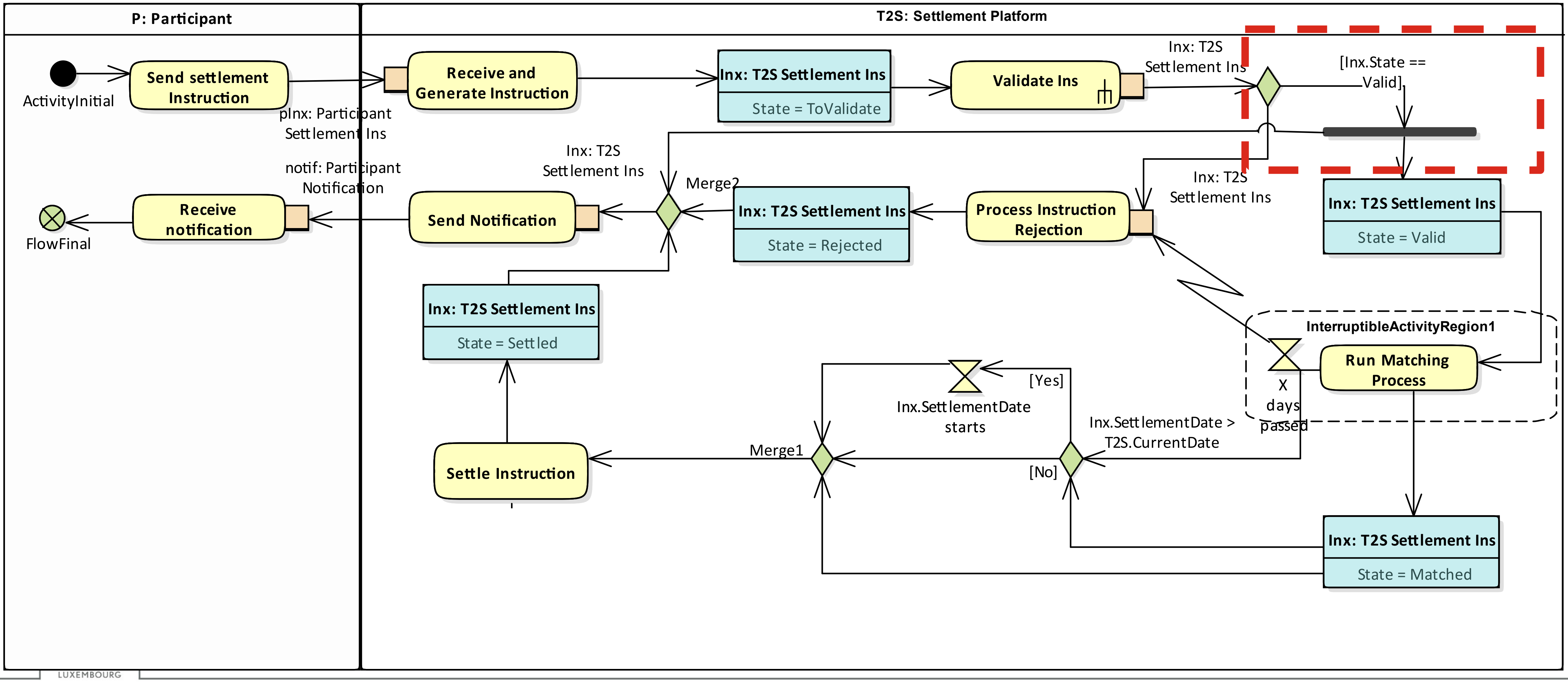
Concurrent Flows – Fork and Join Nodes

- Fork Node
  - **Splits** a flow into concurrent sub-flows.
  - **Duplicates** token for all outgoing edges
  - Actions can be executed in any order.
- Join Node

  Synchronizes concurrent sub-flows. This means:
  - **Wait** until tokens are present at all incoming edges
  - **Merge** all control tokens into one token and passes it on
  - **Pass** on all object tokens



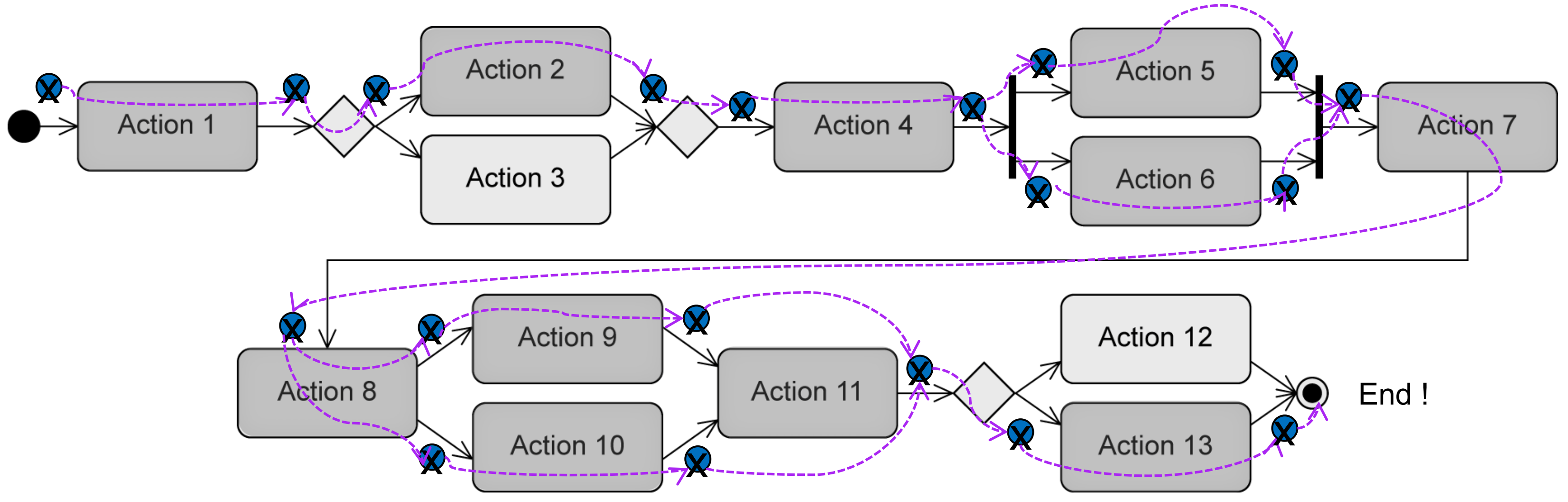
Describe What Happens in the Two Concurrent Flows



Tokens (1/2)

- Mechanism that grants the execution permission to actions
- Not explicitly represented in the diagrams
- If an action receives a token, the action can be executed
- An action passes a token to the subsequent action when is has completed its execution
- Guards can prevent the passing of a token
  - Tokens are stored in the previous node
- Control token and object token
  - Control token: “execution permission" for a node
  - Object token: transport data + “execution permission”

Tokens (2/2)



… all outgoing edges of all initial nodes are assigned a token….

… if all incoming edges of an action have a token, the action is activated and is ready for execution

… before the execution, the action consumes one token from every incoming edge;
after the execution, the action passes one token to every outgoing edge

… a decision node passes the token to **one** outgoing edge (depending on the result of the evaluation of the guard)

… a merge node individually passes each token it gets to its outgoing edge

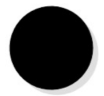
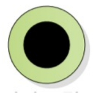
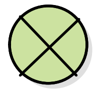
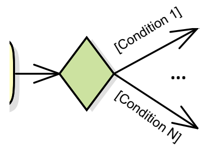
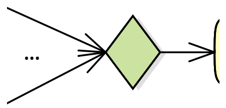
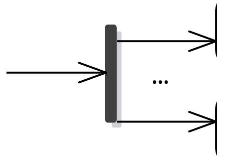
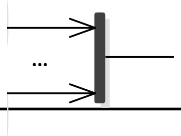
… a parallelization node duplicates an incoming token for **all** outgoing edges

… a synchronization node waits until all incoming edges have a token, merges them to a single token and passes it to its outgoing edge

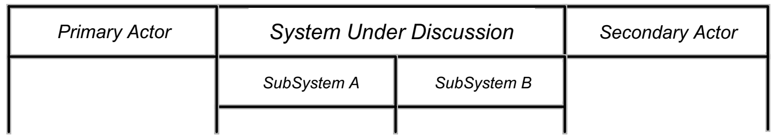
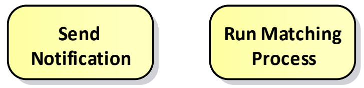
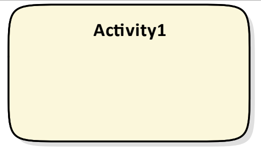
… the first token that reaches the activity final node terminates the entire activity

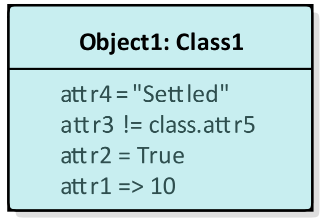
[1] Taken from: UML @ Classroom: An Introduction to Object-Oriented Modeling

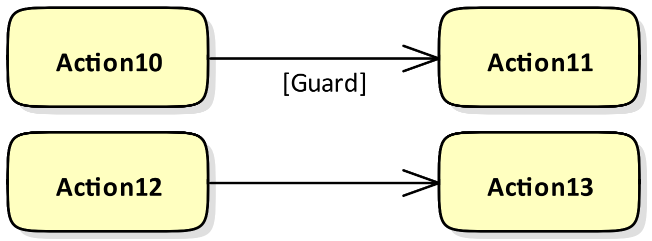
Activity Diagrams Notation used in Qualisist (1/4)

Name	Notation	Description
Initial node		Start of the execution of an activity
Activity final node		End of ALL execution flows of an activity
FlowFinalNode		End of One execution flow of an activity
DecisionNode		Chooses between outgoing flows
MergeNode		Brings together multiple flows **without synchronization.**
ForkNode		**Splits** a flow into multiple concurrent flows.
JoinNode		**Synchronizes multiple flows** Note: all actions linked to ingoing flows must be completed before execution continues

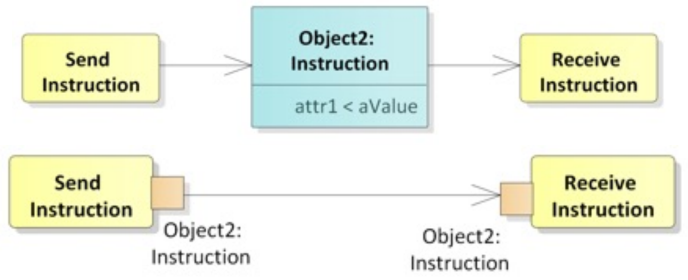
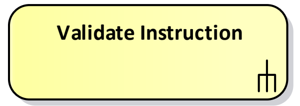
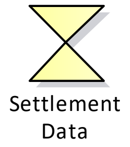
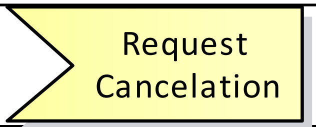
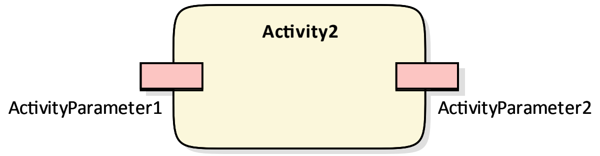
Activity Diagrams Notation used in Qualisist (2/4)

Name	Notation	Description
Activity Partitions	 Primary Actor System Under Discussion Secondary Actor SubSystem A SubSystem B	Grouping of nodes and edges within an activity
Action		Represents an action (atomic)
Activity		Represents an activity (can be broken down further)

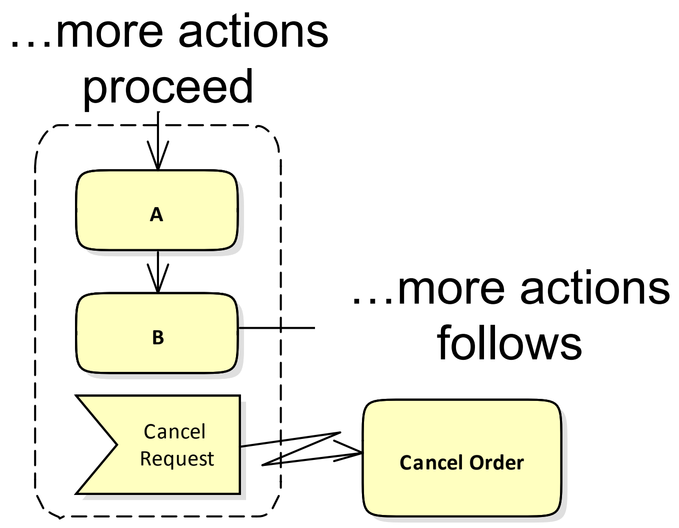
Object	 Object1: Class1
	attr4 = "Settled"
	attr3 != class.attr5
	attr2 = True
	attr1 => 10
		Contains data or objects

Control Flow		Define the execution order between nodes. The flow only continues if guards (conditions) in square brackets evaluate to true

Activity Diagrams Notation used in Qualisist (3/4)

Name	Notation	Description
Object Flow	 Send Instruction Object2: Instruction attr1 < aValue Receive Instruction Send Instruction Object2: Instruction Object2: Instruction Receive Instruction	Used to exchange data or objects. Express a data dependency between nodes
Call Behavior Action		Action A refers to Activity1
Accept Time Event		Wait for a time event
Accept Event		Wait for an event
Activity Parameter Node		Contains data and objects as input and output parameters

Activity Diagrams Notation used in Qualisist (4/4)

Name	Notation	Description
Interruptible activity region	 …more actions proceed A B …more actions follows Cancel Request Cancel Order	Flow continues on a different path if event **`Cancel Request`** is detected

Practice 2: Create UML Use Case, Class and Activity Diagrams (1/2)

- **Goal:** Learn to create a model that includes a UML use case, class and activity diagram according to the Qualisist modelling methodology.

Use_Cases exercises

Microsoft
Word Document

- **Tasks:**
  1. Open **MT103 9x Cash Deadline** Qualisist project
  2. Create a Use Case in the relevant package (open the file **"Use_Cases exercises"**)
  3. Based on the proposed solution (see next slide) create the To-Be Activity diagram (as a basis copy the As-Is activity diagram) and update domain model
  4. Discuss about the different models created by the participants

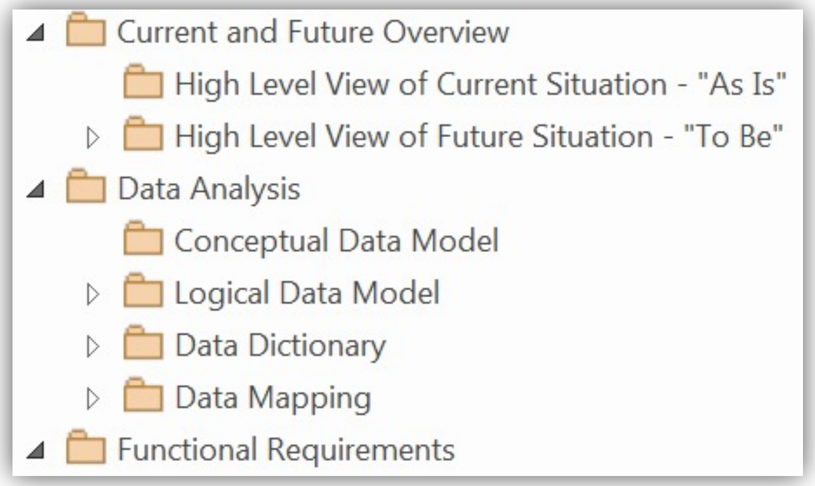
Practice 2: Create UML Use Case, Class and Activity Diagrams (2/2)

Context: *Currently, when an OI sends a subscription order with FOPP/Immediate settlement method, Vestima immediately generates and settles 9x internal cash transfer with the current business day and time as value date. As soon as 9x is settled, Vestima generates 90 instruction. However, the settlement of 90 instruction is subject to the cash instruction deadlines. If the cash instruction deadline is passed, the 90 instruction will only settle on the next available cash processing date as value date. Thus, clients complain having their accounts being debited whereas the actual payment is done on the next business day.*

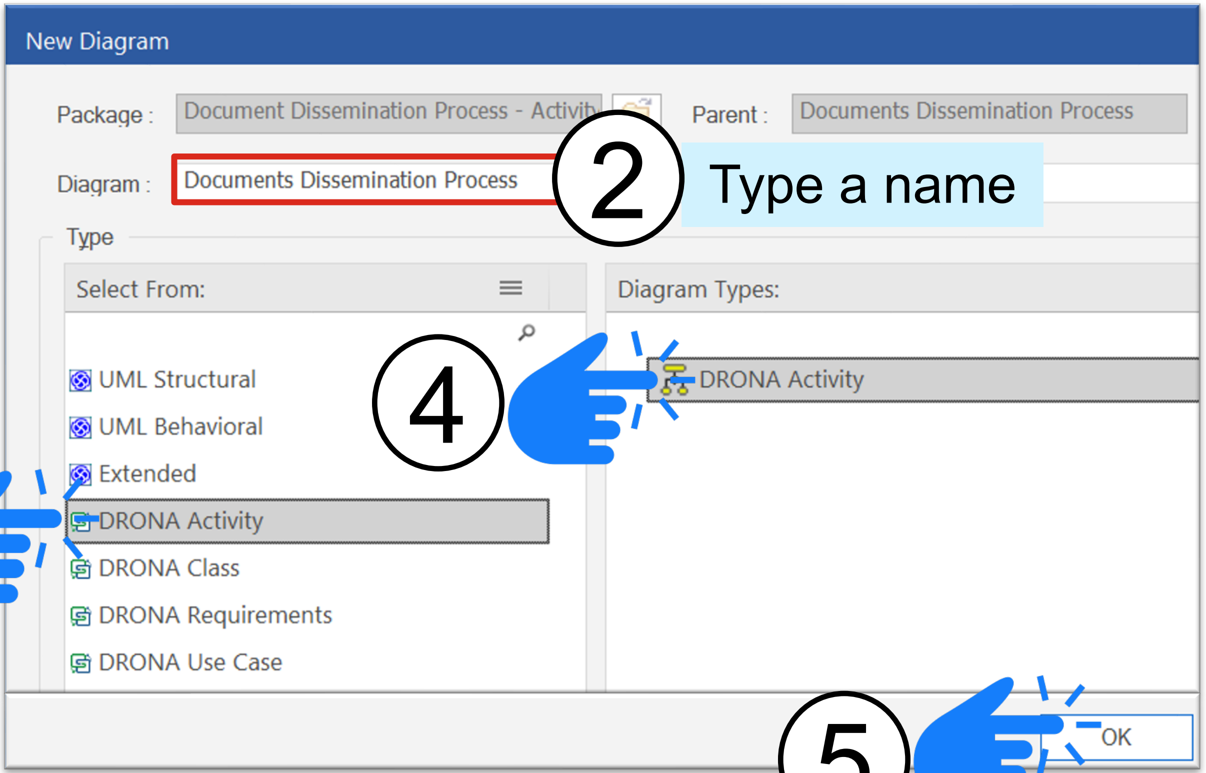
Solution: *In order to align settlement dates of 9x and 90inx, 9x instruction must become subject to cash deadlines.*
NCCIP maintains information about cash deadlines. In order to retrieve cash deadline from NCCIP before generating 9x, Vestima will create 90inx (that will serve as a request to retrieve the cash deadline) where Vestima will add 15 minutes to the "Receive timestamp" field and send this inx to NCCIP. Based on the inx, NCCIP will provide Vestima with "Expected value date" that takes the cash deadlines into consideration.
If Vestima receives expected value date from NCCIP, then Vestima must generate 9x with the expected value date.
In case Vestima receives an error message from NCCIP (such message header will contain 400, 401, 404, 500, 503 in the response) or does not receive any response from NCCIP within 15 seconds, then Vestima must by default generate 9x with the current business date as value date.

Steps to Create a Qualisist Diagram

1 Right click on any package



- Current and Future Overview
  - High Level View of Current Situation - "As Is"
  - High Level View of Future Situation - "To Be"
- Data Analysis
  - Conceptual Data Model
  - Logical Data Model
  - Data Dictionary
  - Data Mapping
- Functional Requirements



New Diagram

Package : Document Dissemination Process - Activity

Parent : Documents Dissemination Process

Diagram : Documents Dissemination Process

2 Type a name

Type

Select From:	Diagram Types:
UML Structural	DRONA Activity
UML Behavioral	
Extended	
DRONA Activity	
DRONA Class	
DRONA Requirements	
DRONA Use Case	

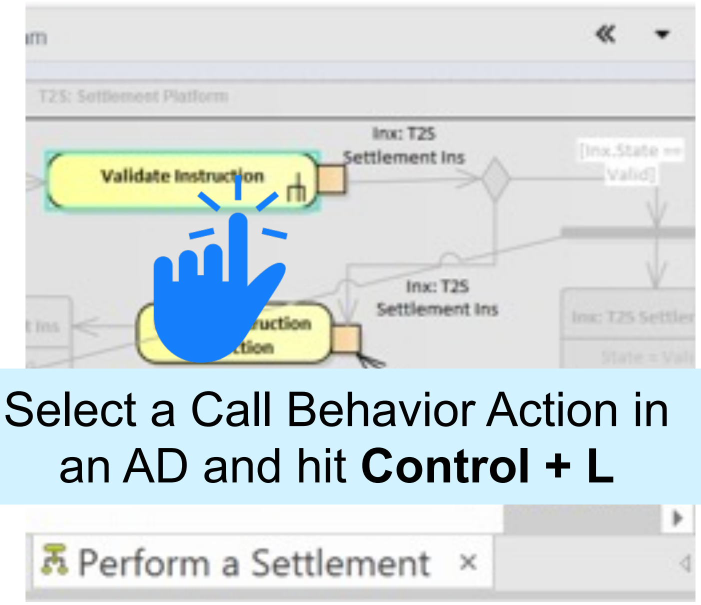
3

4

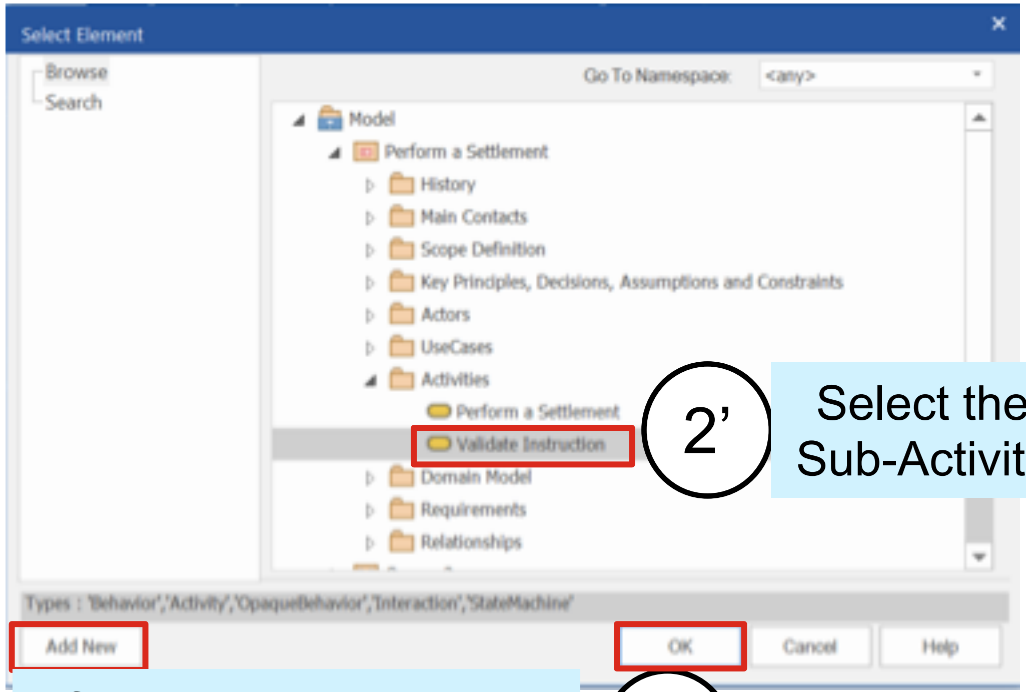
5

OK

Steps to Select a Sub-Activity (1/2)



1 Select a Call Behavior Action in an AD and hit **Control + L**



Types : 'Behavior','Activity','OpaqueBehavior','Interaction','StateMachine'

2' Select the Sub-Activity

2 If you have not created the sub-activity, hit **Add New** (**See the next Slide**)

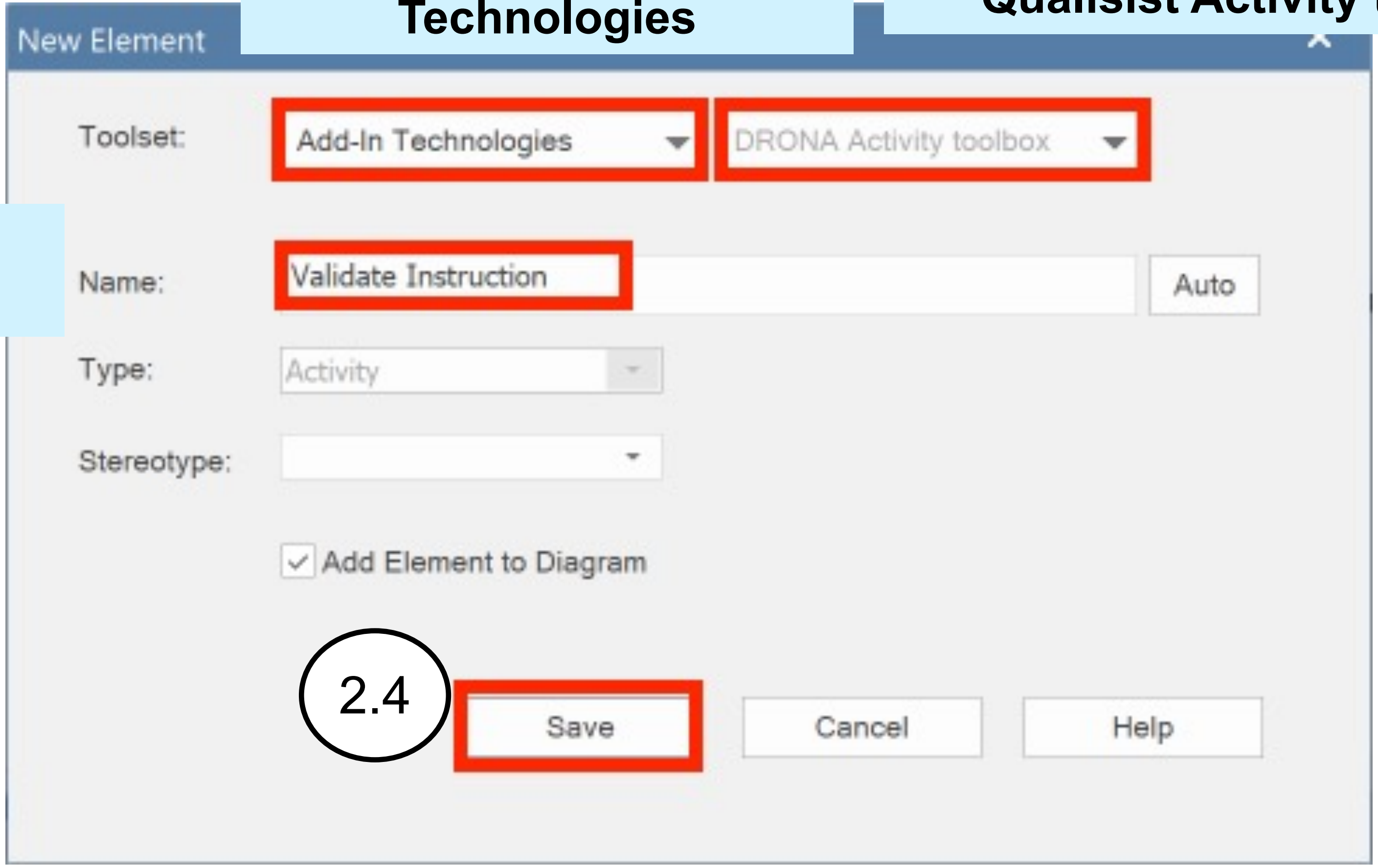
3 Hit **OK**

Steps to Assign a Call Sub-Activity (2/2)

2.2 Click on **Toolset → Specialized → Add-In Technologies**

Select **Qualisist Activity toolbox** 2.3

2.1 Name the new Activity



New Element

Toolset: Add-In Technologies | DRONA Activity toolbox

Name: Validate Instruction

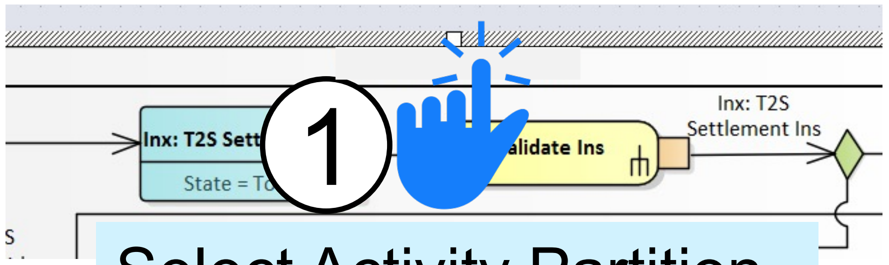
Type: Activity

Stereotype:

✓ Add Element to Diagram

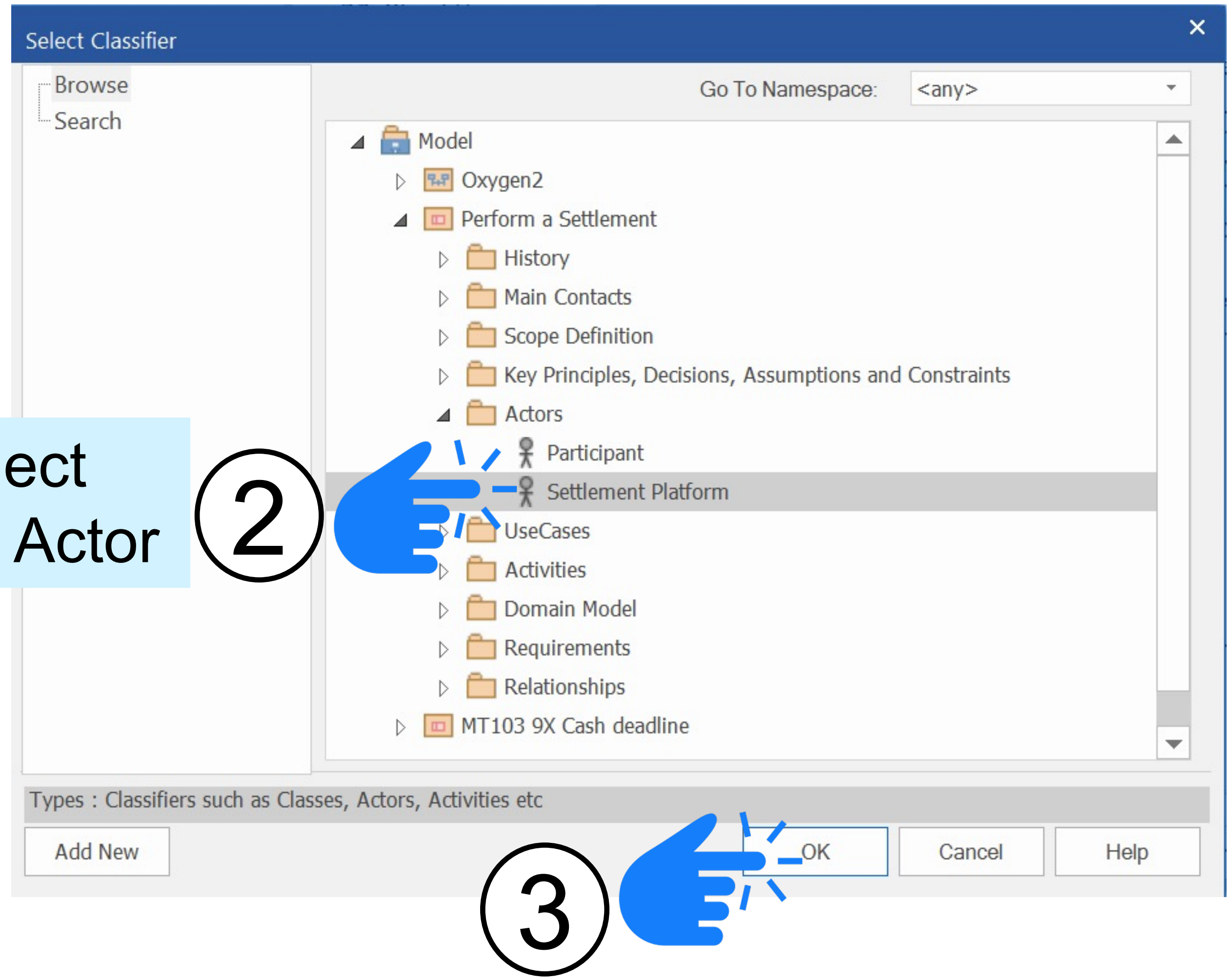
2.4

Steps to Reference an Actor from an Activity Partition



Select Activity Partition and hit **Control + L**

Select the Actor


Select Classifier

- Browse
- Search

Go To Namespace: <any>

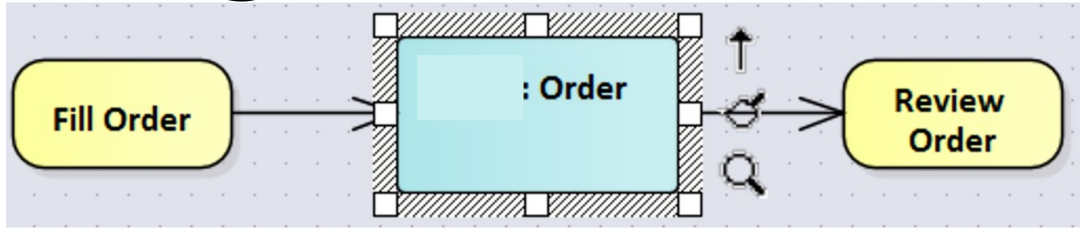
- Model
  - Oxygen2
  - Perform a Settlement
    - History
    - Main Contacts
    - Scope Definition
    - Key Principles, Decisions, Assumptions and Constraints
    - Actors
      - Participant
      - Settlement Platform
    - UseCases
    - Activities
    - Domain Model
    - Requirements
    - Relationships
  - MT103 9X Cash deadline

Types : Classifiers such as Classes, Actors, Activities etc

Add New | OK | Cancel | Help

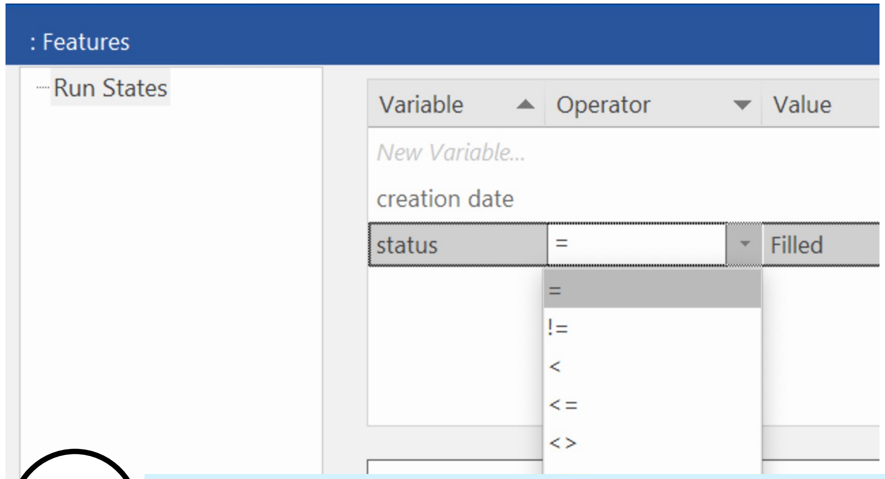
Steps to Model the Varying Behavior of Objects at Run-time

1 Select an Object

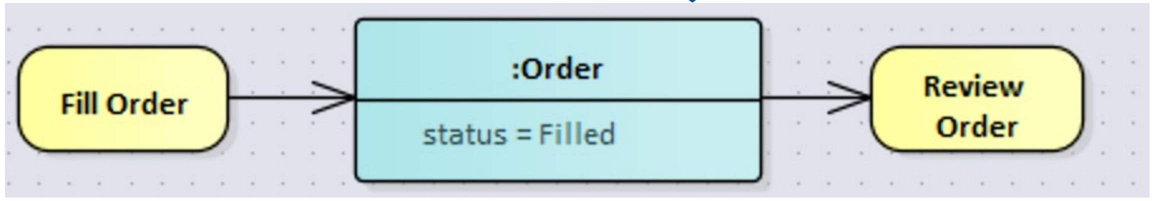


2 Hit **Control + Shift + R**

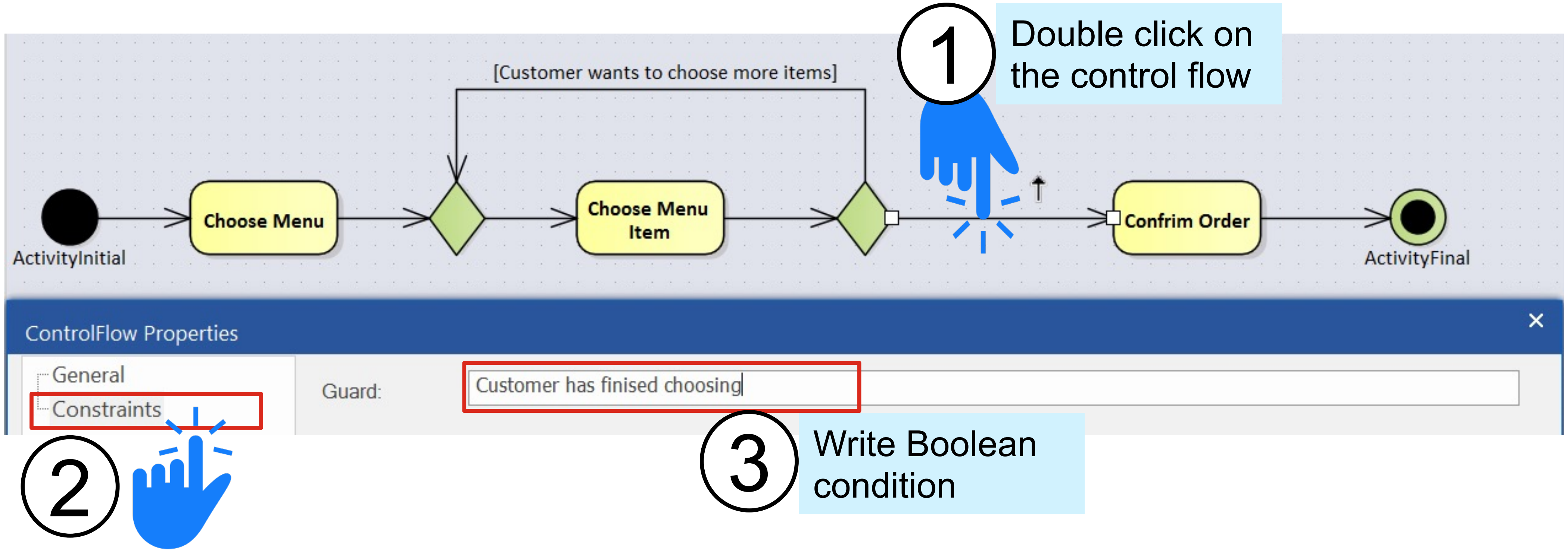
3 Select the run-time value



4 Close the above windows



Steps to Use Guards In Control Flows

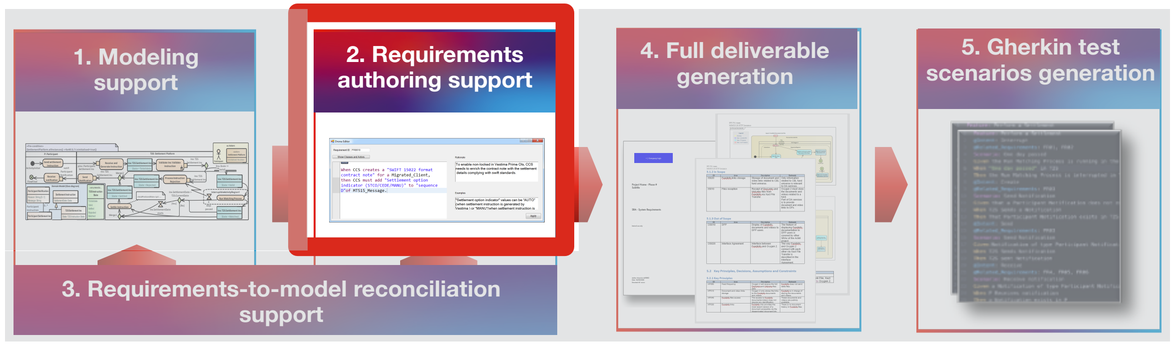


Guard: Customer has finised choosing

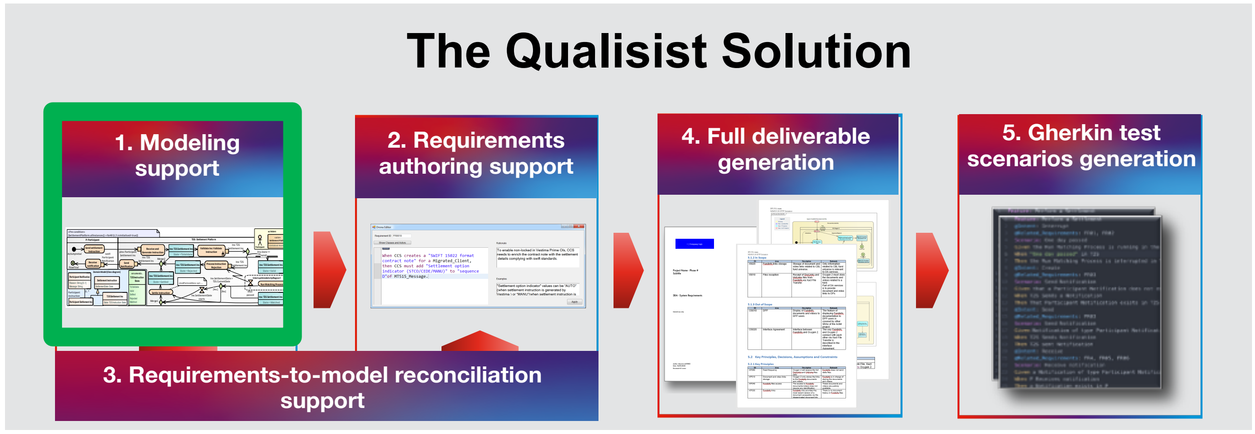
Agenda

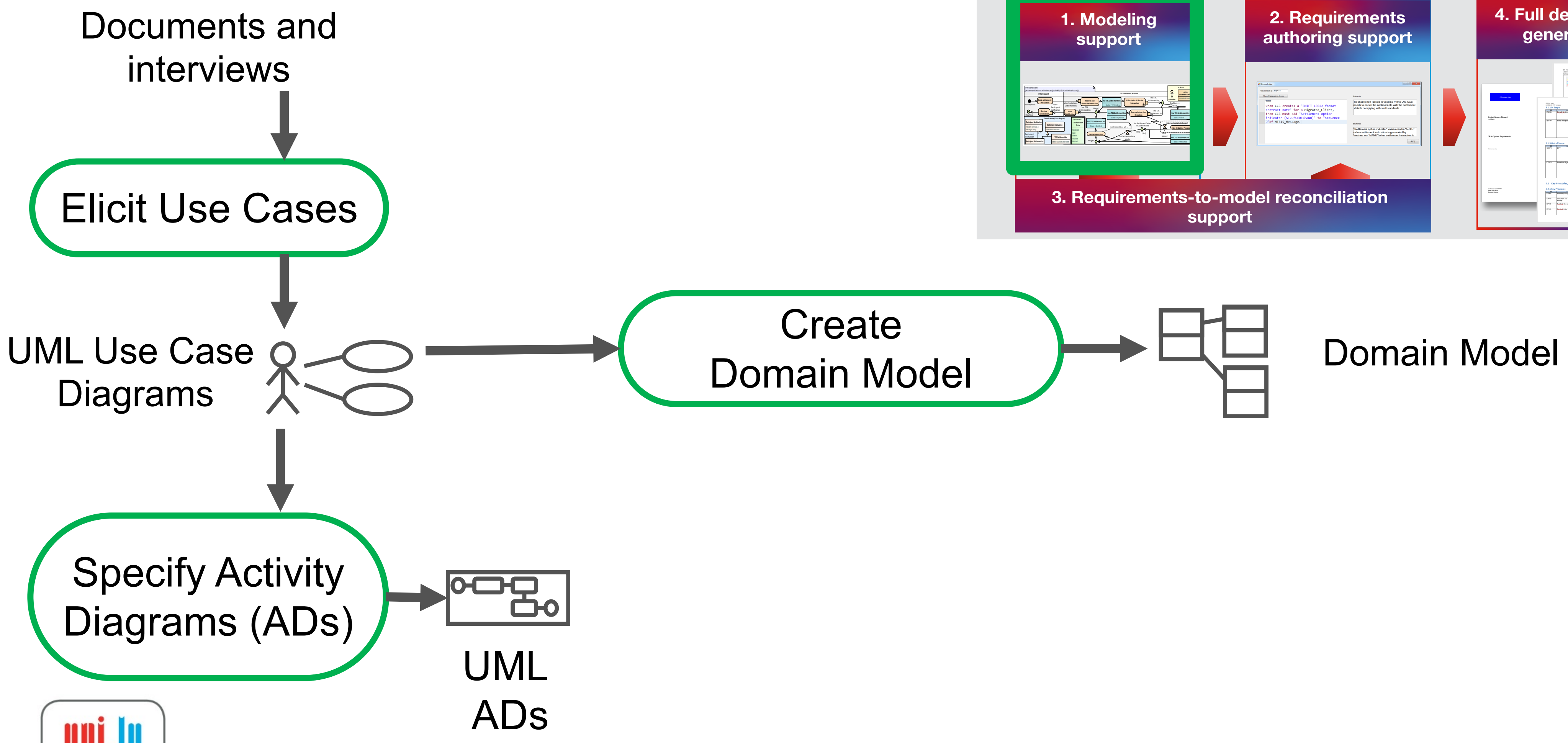
0. Installation and Configuration
1. Modelling Support
2. **Requirements authoring support**
3. Requirements-to-Model reconciliation support
4. Full deliverable generation
5. Gherkin test Scenarios generation

The Qualisist Solution

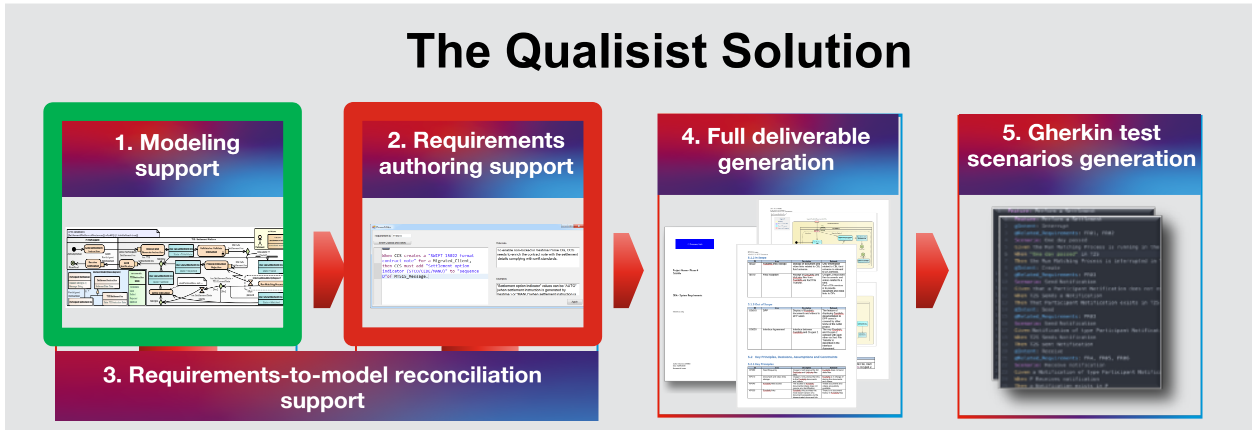

1. Modeling support
2. Requirements authoring support
4. Full deliverable generation
5. Gherkin test scenarios generation
3. Requirements-to-model reconciliation support

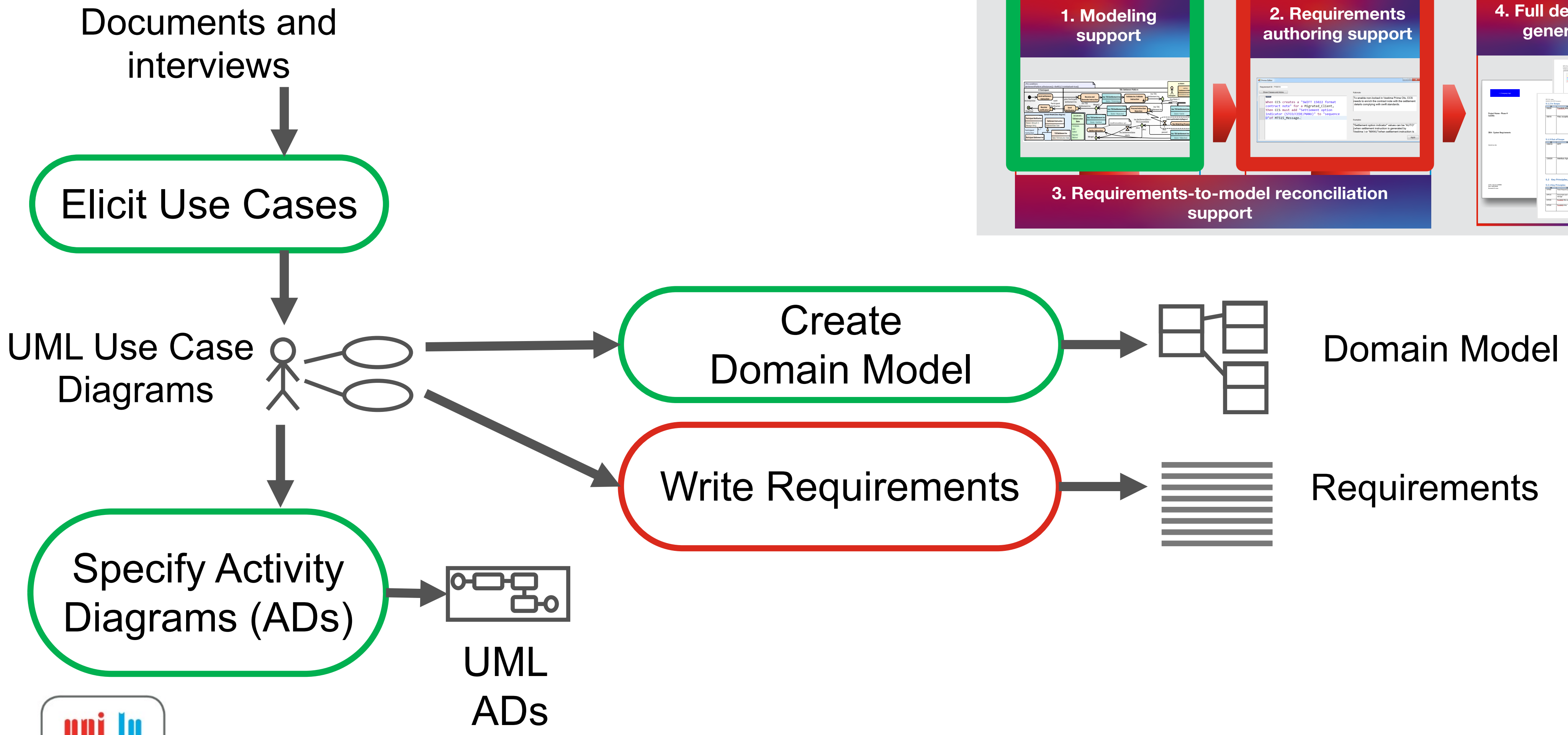
Recall the Qualisist Modeling Methodology


The Qualisist Solution
1. Modeling support
2. Requirements authoring support
3. Requirements-to-model reconciliation support
4. Full deliverable generation
5. Gherkin test scenarios generation



Write Requirements


The Qualisist Solution
1. Modeling support
2. Requirements authoring support
3. Requirements-to-model reconciliation support
4. Full deliverable generation
5. Gherkin test scenarios generation

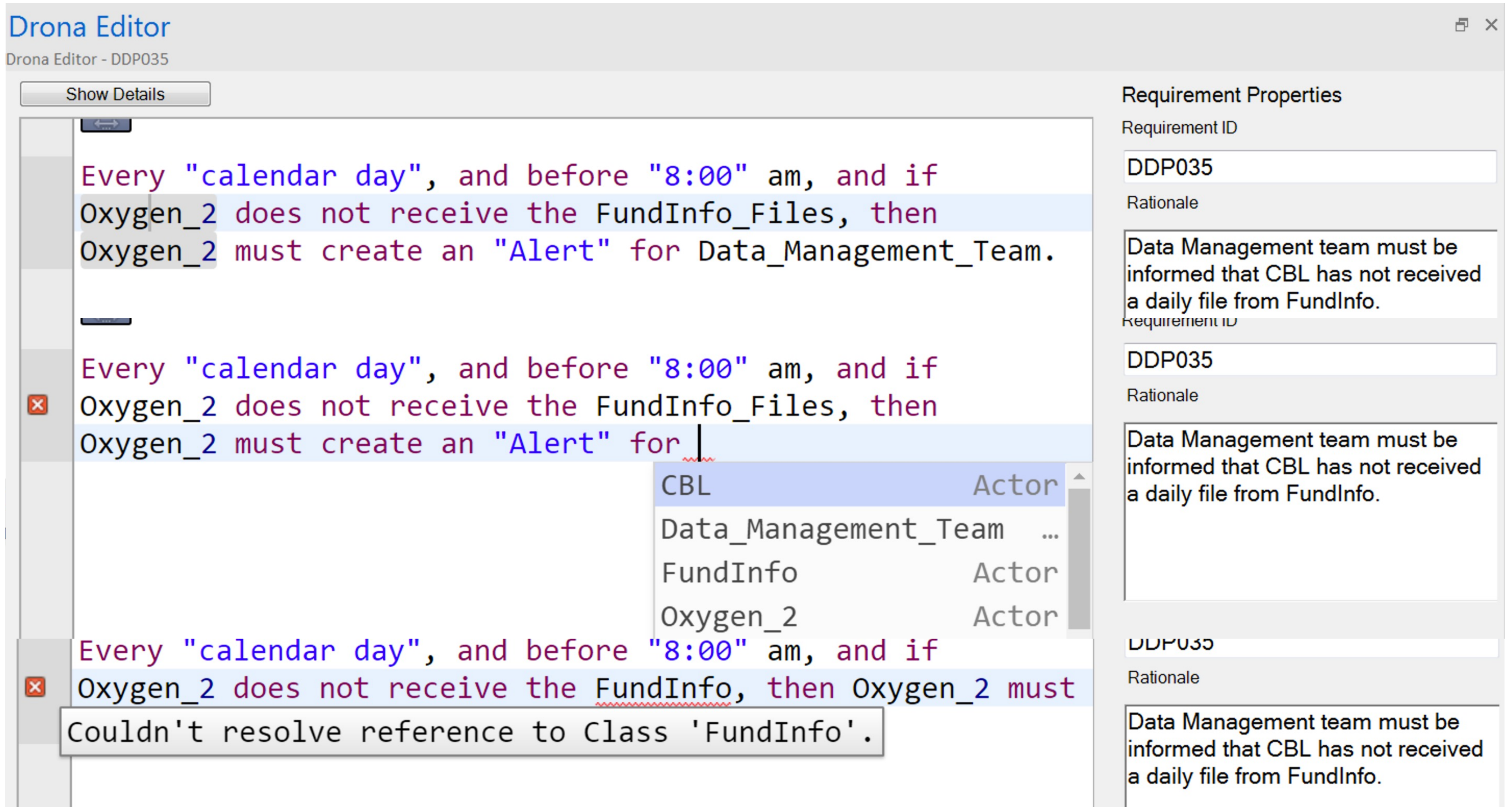


Rimay – A Language for Writing Requirements

- We systematically developed a textual language named **Rimay** intended at helping analysts write functional requirements
- We used:
  - 15 SRA's from Clearstream
  - 3215 natural language functional requirements
- Editor integrated in Enterprise Architect
  - Allows to trace between text requirements and models
  - Enables Requirements-to-Models consistency checking

Textual Requirements Support

- Requirement syntax check
- Autocompletion
- Instant feedback about requirements errors


Drona Editor

Drona Editor - DDP035

Show Details

```
Every "calendar day", and before "8:00" am, and if
Oxygen_2 does not receive the FundInfo_Files, then
Oxygen_2 must create an "Alert" for Data_Management_Team.

Every "calendar day", and before "8:00" am, and if
Oxygen_2 does not receive the FundInfo_Files, then
Oxygen_2 must create an "Alert" for
                                    CBL                  Actor
                                    Data_Management_Team ...
                                    FundInfo             Actor
                                    Oxygen_2             Actor
Every "calendar day", and before "8:00" am, and if
Oxygen_2 does not receive the FundInfo, then Oxygen_2 must
Couldn't resolve reference to Class 'FundInfo'.
```

Requirement Properties

Requirement ID: DDP035

Rationale: Data Management team must be informed that CBL has not received a daily file from FundInfo.

Requirement ID: DDP035

Rationale: Data Management team must be informed that CBL has not received a daily file from FundInfo.

DDP035

Rationale: Data Management team must be informed that CBL has not received a daily file from FundInfo.

Overall Syntax of a Requirement (1/2)

```
REQUIREMENT: SCOPE? CONDITION_STRUCTURE?
             ARTICLE? ACTOR MODAL_VERB not? SYSTEM_RESPONSE.
SCOPE: For MODIFIER? TEXT (and MODIFIER? TEXT),
```

```
MODIFIER: ARTICLE | QUANTIFIER
ARTICLE: a|an|the
QUANTIFIER: each, all, none, only one, any, …
```

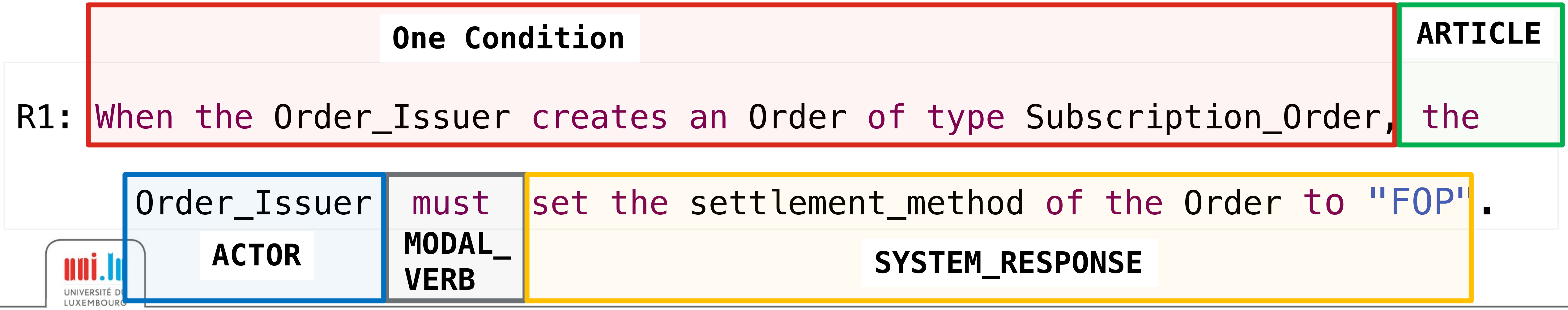
Example of a requirement with scope and without a conditions

```
R2: For each "line of the File", System must
check that Share_Class_Identifier.Value contains "line.ISIN”.
```

Overall Syntax of a Requirement (2/2)

```
REQUIREMENT: SCOPE? CONDITION_STRUCTURES?
                    ARTICLE? ACTOR MODAL_VERB not? SYSTEM_RESPONSE.
CONDITION_STRUCTURES: CONDITION_STRUCTURE ( (,|and|or|,or|,and)
                             CONDITION_STRUCTURE )*, then?
```

- **Example:** A requirement with a condition and without scope:


R1: When the Order_Issuer creates an Order of type Subscription_Order, the Order_Issuer must set the settlement_method of the Order to "FOP".

Condition Structures (When)

```
CONDITION_STRUCTURE: WHEN | IF | TEMPORAL …
WHEN: When TRIGGER
IF : If PRECONDITIONS | TRIGGER
TEMPORAL : (Before | After | Every) TRIGGER | TIME
```

- **Example:**

$$\texttt{When}\ \overbrace{\underbrace{\texttt{SystemB}}_{}\!{}^{\texttt{ACTOR}}\ \overbrace{\texttt{receives an "email alert" from SystemA}}^{\texttt{ACTIONS EXPRESSSION}}}^{\texttt{TRIGGER}}$$

Condition Structures (If) (1/2)

```
CONDITION_STRUCTURE : WHEN | IF | TEMPORAL …
WHEN: When TRIGGER
IF : If PRECONDITIONS | TRIGGER
TEMPORAL : (Before | After | Every) TRIGGER | TIME
```

- **Example (Non itemized preconditions):**

$$\texttt{If}\ \overbrace{\overbrace{\texttt{Instruction.description contains a "Keyword"}}^{\texttt{CONDITION}}\ \texttt{or}\ \overbrace{\texttt{Instruction.record is "Live"}}^{\texttt{CONDITION}}}^{\texttt{CONDITIONS\_EXPRESSION}}$$

Condition Structures (If) (2/2)

- **Example (Itemized preconditions):**

```
 If the following conditions are satisfied:
HYPHEN                                     CONDITION

  -  the "Instruction" has the properties described in "Section Y",
  -  the "Instruction" has the properties: "Status and Settlement Date",
  -  the Instruction.Settlement_Date conforms to the standard "ISO8601",
  -  the Transaction.Amount is less than or equal to "Y Value",
  -  the "Transaction Type" of Settlement_Request is equal to "Z Value" and
  -  the "Account Number" field  contains "0000"
```

Condition Structures (Temporal)

```
CONDITION_STRUCTURE: WHEN | IF | TEMPORAL …
WHEN: When TRIGGER
IF: If PRECONDITIONS | TRIGGER
TEMPORAL: (Before | After | Every) TRIGGER
```

- **Example:**

$$\text{Before } \overbrace{\overbrace{\text{SystemA}}^{\text{ACTOR}}\ \overbrace{\text{sends an "Instruction" to SystemB}}^{\text{ACTIONS EXPRESSSION}}}^{\text{TRIGGER}}$$

Two or More Condition Structures

```
CONDITION_STRUCTURES: CONDITION_STRUCTURE
           ( (,|and|or|,or|,and) CONDITION_STRUCTURE )*, then?
CONDITION_STRUCTURE: WHEN | IF | TEMPORAL
```

$$\overbrace{\overbrace{\textbf{Before}\ \text{"8:00 am",}}^{\textbf{TEMPORAL}}\ \overbrace{\textbf{every}\ \text{"calendar day",}}^{\textbf{TEMPORAL}}\ \overbrace{\textbf{if}\ \text{System}\ \textbf{does not receive the}\ \text{"X" File,}}^{\textbf{If}}}^{\textbf{CONDITION STRUCTURES (More than one)}}\ \textbf{then}$$

System **must create an** ”Alert"

Types of System Response (1/3)

The $\overbrace{\texttt{User}}^{\textbf{ACTOR}}$ must $\overbrace{\texttt{upload the "excel file" to the "SystemA"}}^{\textbf{ATOMIC SYSTEM RESPONSE}}$

The $\overbrace{\texttt{SystemA}}^{\textbf{ACTOR}}$ must $\overbrace{\texttt{create an "Confirmation Message” (referred to as MsgI)}}^{\textbf{SYSTEM RESPONSE EXPRESSION}}$
and send MsgI to SystemB

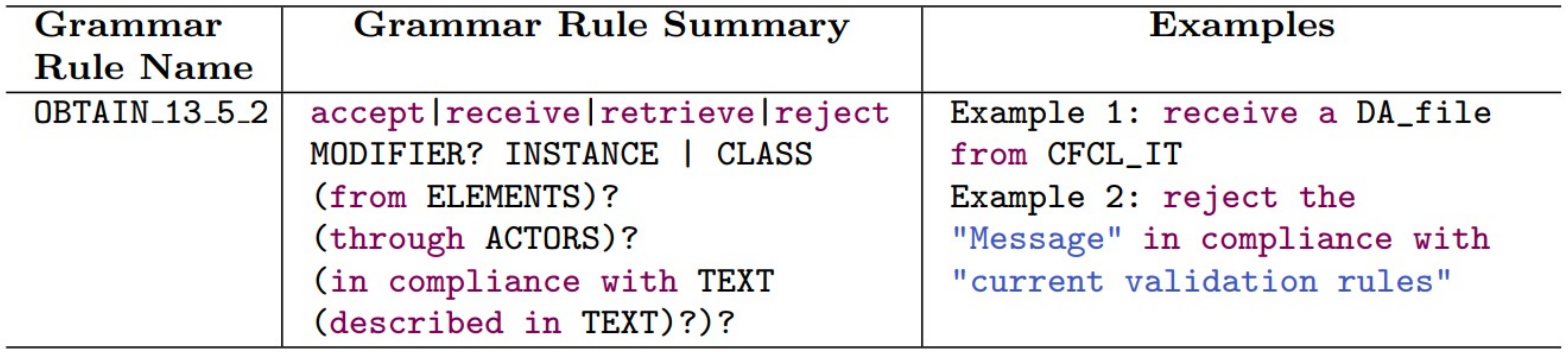
The $\overbrace{\texttt{SystemA}}^{\textbf{ACTOR}}$ must do the following actions in sequence:

$\overbrace{}^{\textbf{INT}}$

$$\left.\begin{array}{ll} 1 & \texttt{create an "Instruction"} \\ 2 & \texttt{send "Instruction" to SystemB} \end{array}\right\} \textbf{ATOMIC\_SYSTEM\_RESPONSE(s)}$$

Types of Atomic System Responses

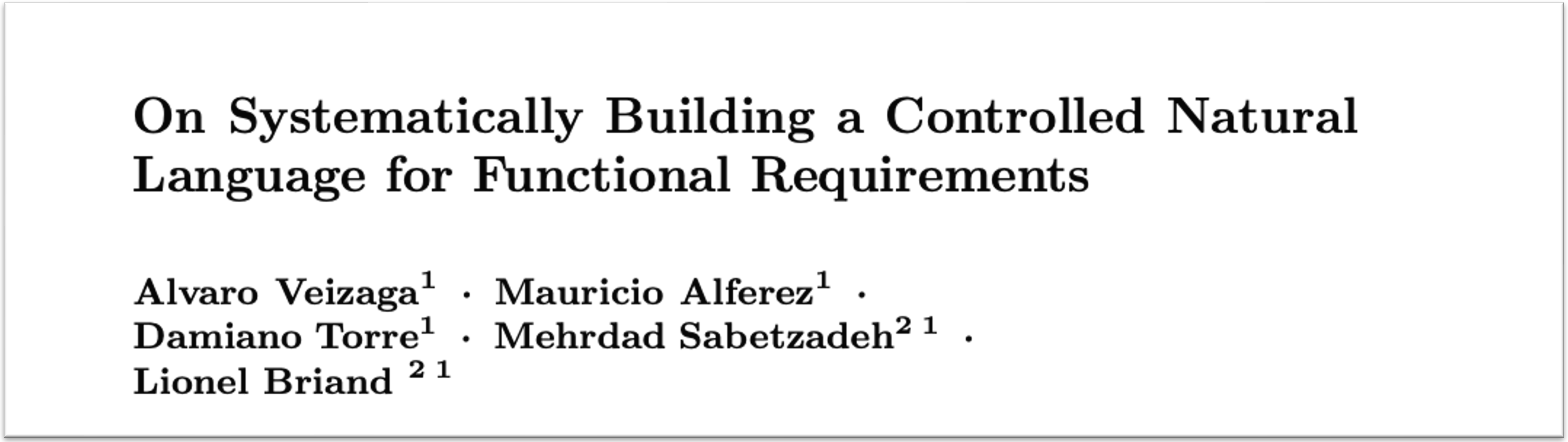
- There are 48 Grammar Rules to specify atomic system responses
- You don’t have to memorize them, just use them
- **Example:**



Grammar Rule Name	Grammar Rule Summary	Examples
OBTAIN_13_5_2	accept|receive|retrieve|reject MODIFIER? INSTANCE | CLASS (from ELEMENTS)? (through ACTORS)? (in compliance with TEXT (described in TEXT)?)?	Example 1: receive a DA_file from CFCL_IT Example 2: reject the "Message" in compliance with "current validation rules"

More Information about Rimay

- https://orbilu.uni.lu/handle/10993/46388



On Systematically Building a Controlled Natural Language for Functional Requirements

Alvaro Veizaga[1] · Mauricio Alferez[1] · Damiano Torre[1] · Mehrdad Sabetzadeh[2 1] · Lionel Briand [2 1]

Practice 3: Writing Textual Functional Requirements

Tasks:

1. Get familiar with the Qualisist Requirements Editor (lead by instructors)
2. Preparatory examples (lead by instructors)
3. Divide into groups and rewrite a list of poorly-written requirements in the Qualisist editor (open the file **"Poorly Written Requirements"**) - 10 minutes
4. Discuss the results with other groups – 30 minutes
5. *Homework:* Rewrite the requirements specified in "**MT103 9x Cash Deadline" SRA** using the Qualisist Editor (See below the SRA provided by Elene).
6. Discuss the results on the next training day – 30 minutes

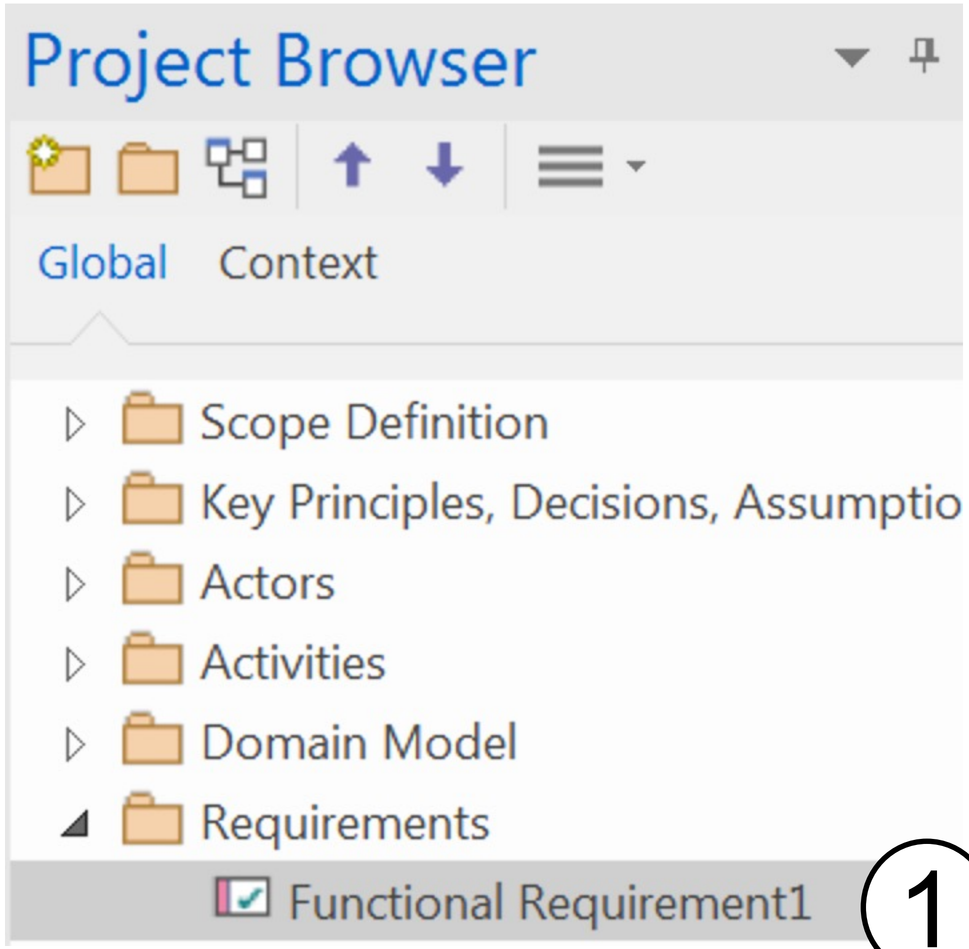
Poorly Written Requirements

Microsoft
Word Document

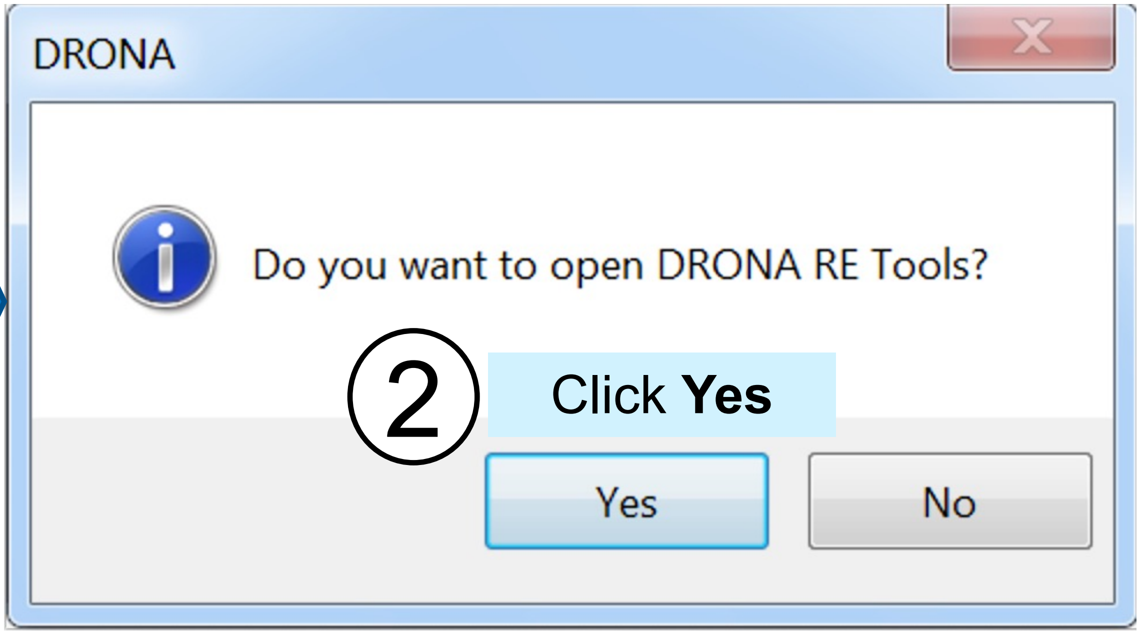
MT103 9x Cash Deadline

Microsoft
Word Document

Steps to Open the Qualisist Requirements Editor



1 Double click on a requirement


DRONA

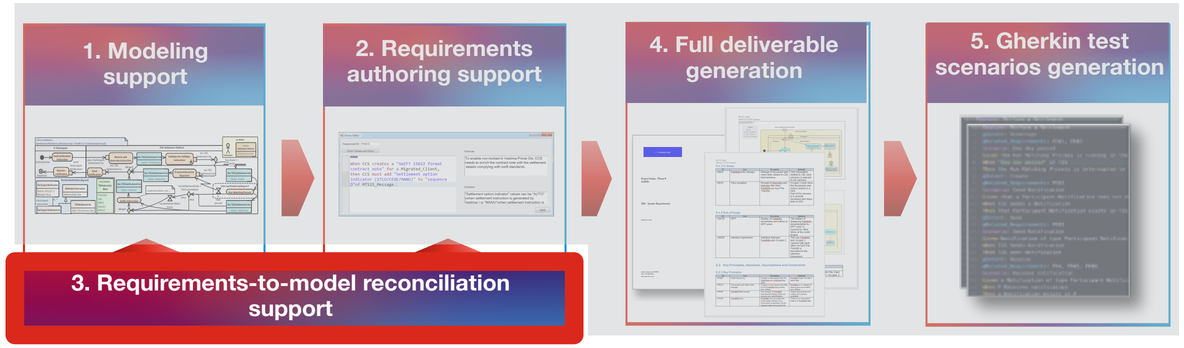
Do you want to open DRONA RE Tools?

2 Click **Yes**

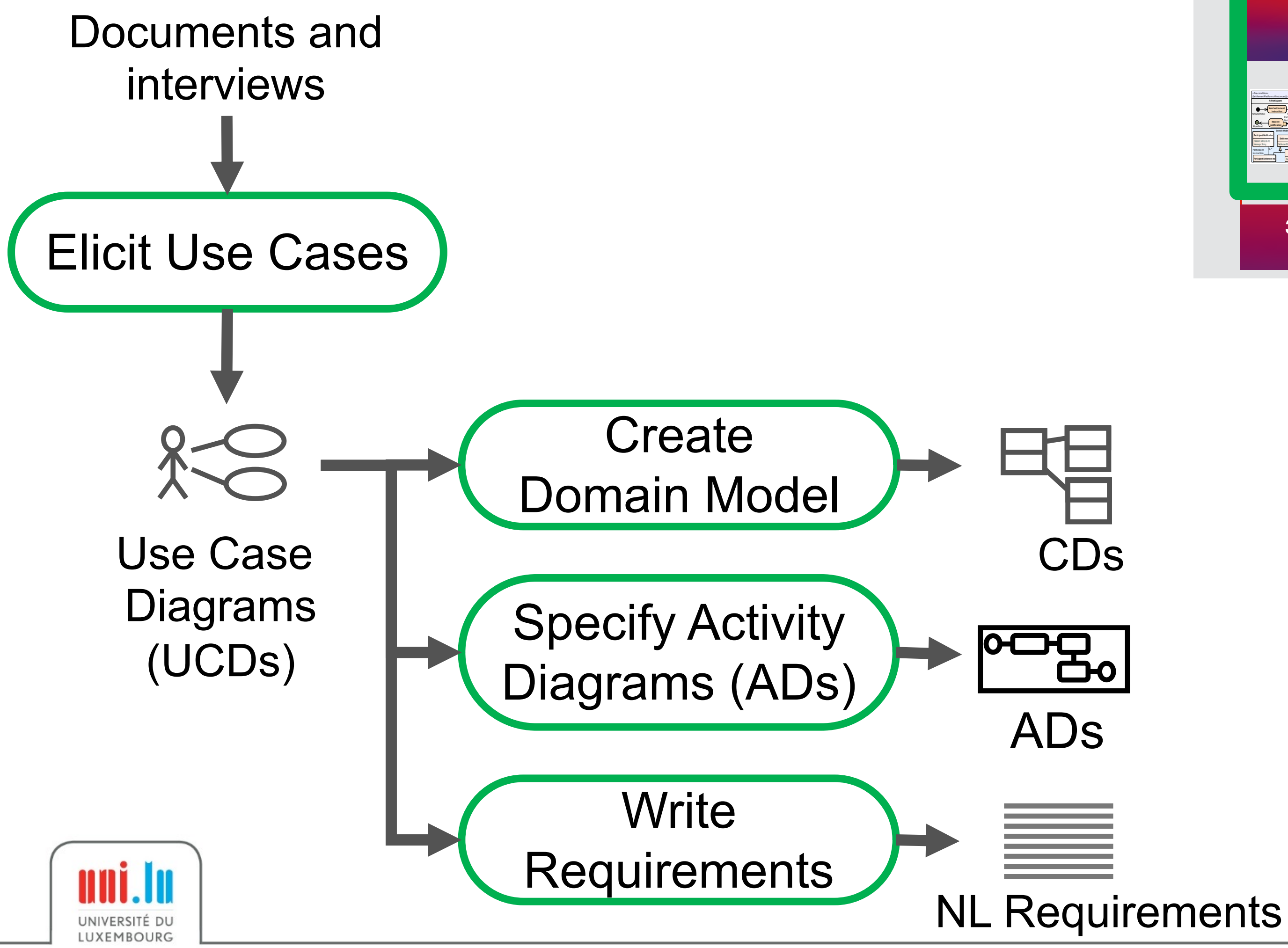
Agenda

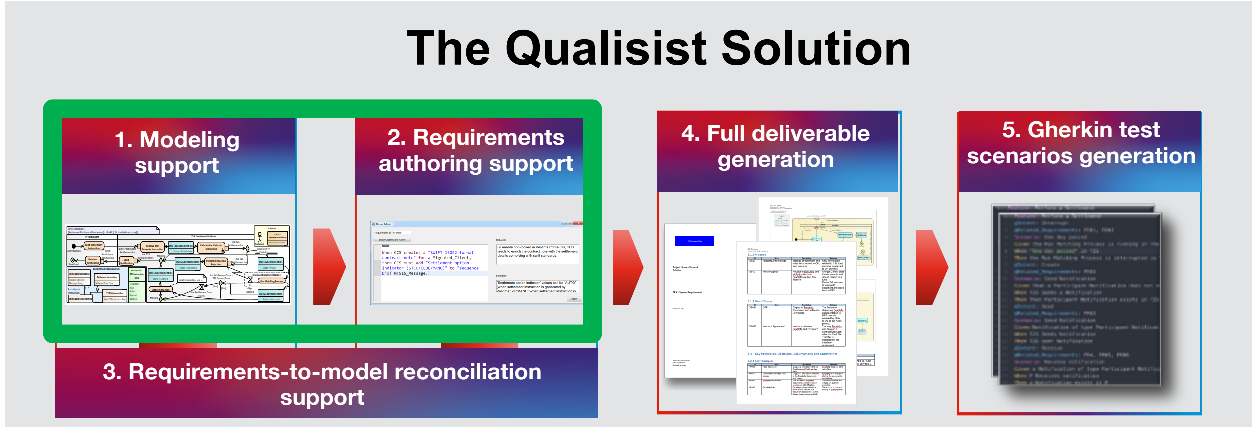
0. Installation and Configuration
1. Modelling Support
2. Requirements authoring support
3. **Requirements-to-Model reconciliation support**
4. Full deliverable generation
5. Gherkin test Scenarios generation

The Qualisist Solution


1. Modeling support
2. Requirements authoring support
4. Full deliverable generation
5. Gherkin test scenarios generation
3. Requirements-to-model reconciliation support

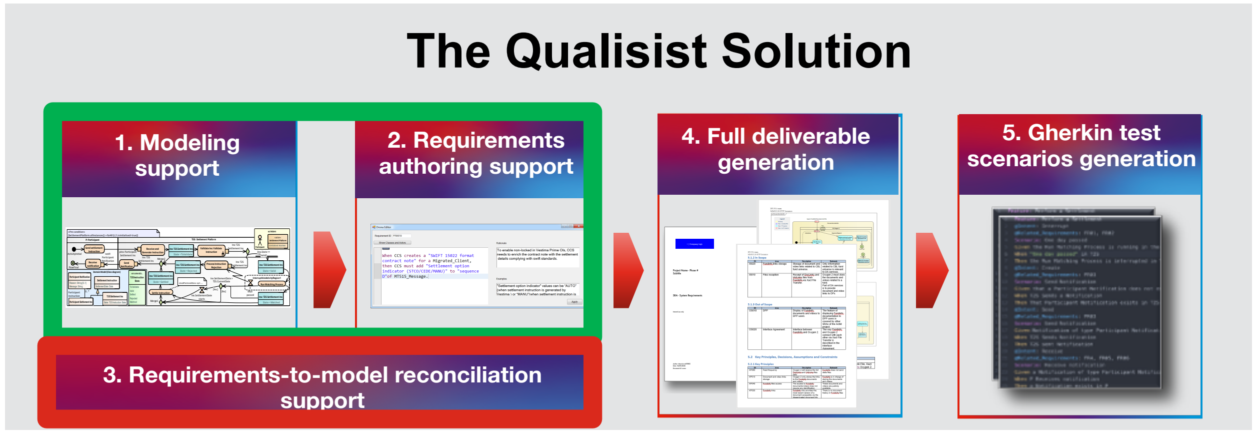
Recall the Qualisist Modeling Methodology

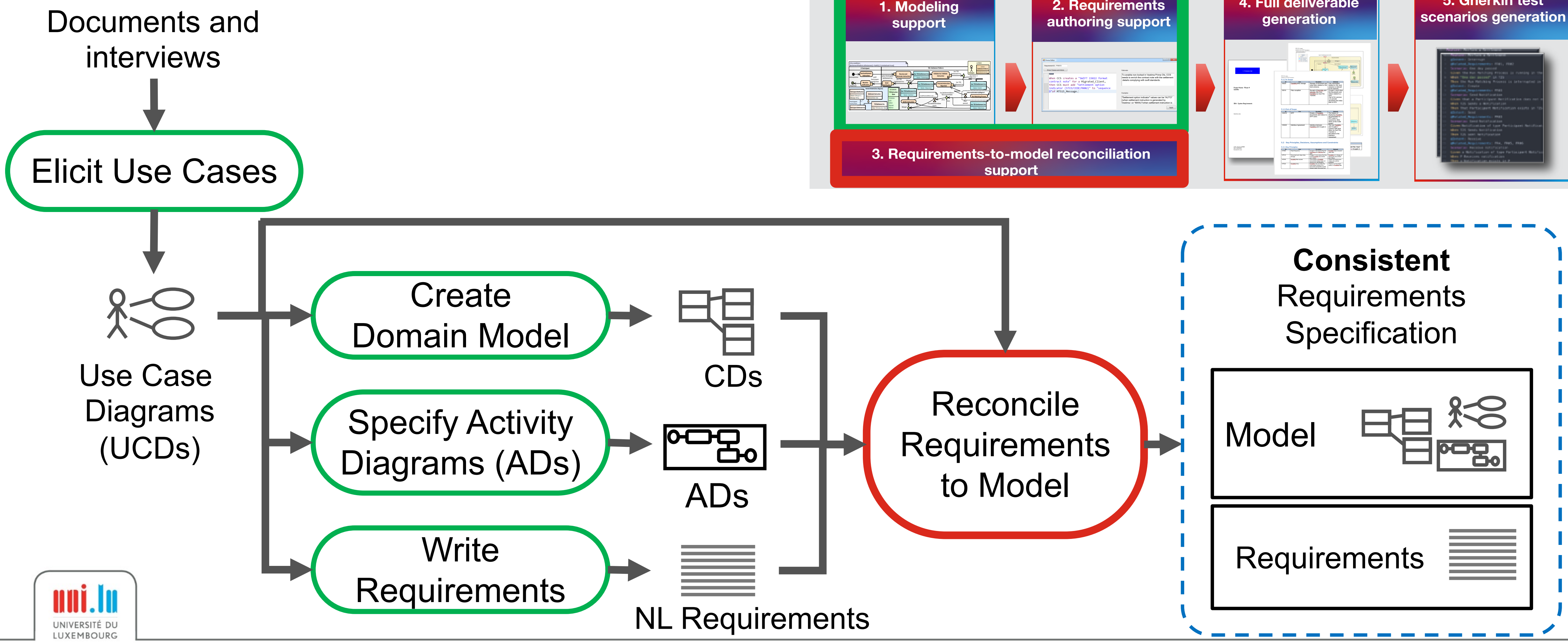




The Qualisist Solution
1. Modeling support
2. Requirements authoring support
3. Requirements-to-model reconciliation support
4. Full deliverable generation
5. Gherkin test scenarios generation

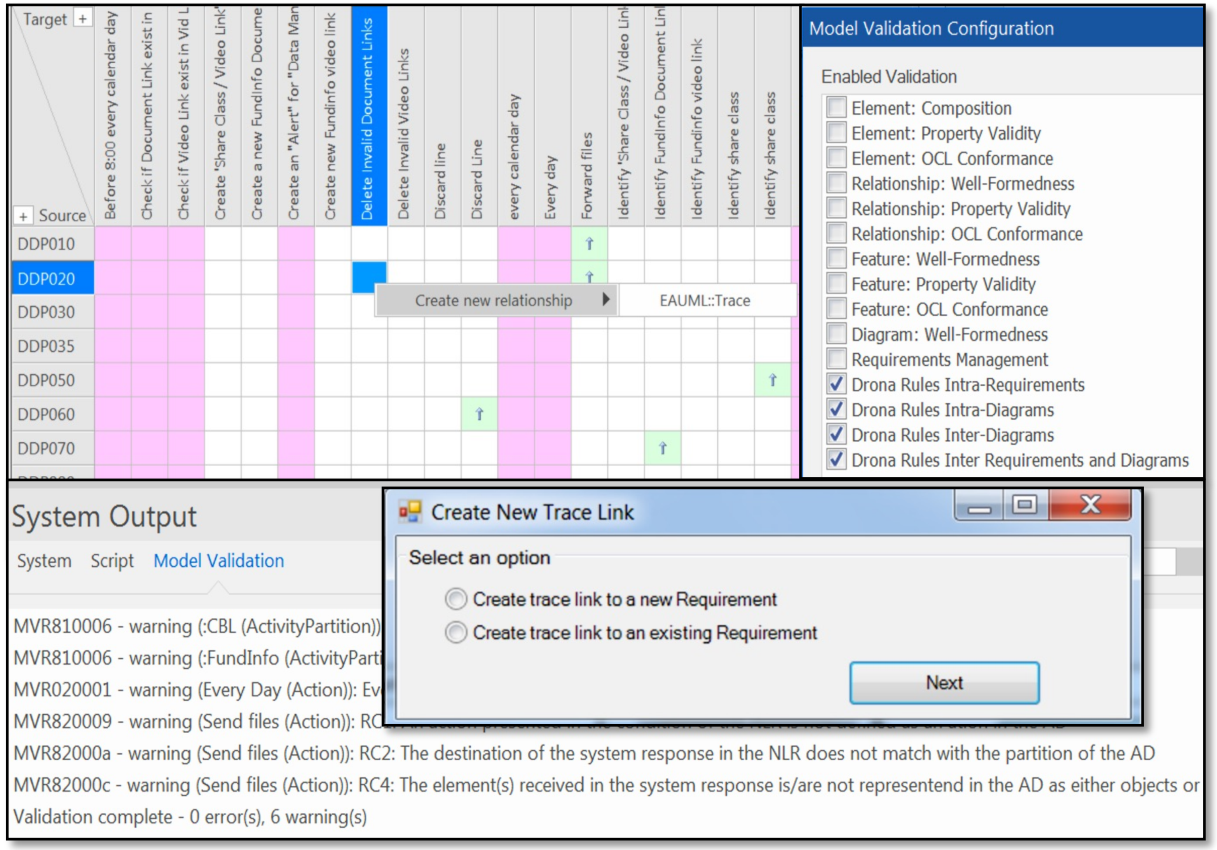
Reconcile Requirements to Model


The Qualisist Solution
1. Modeling support
2. Requirements authoring support
3. Requirements-to-model reconciliation support
4. Full deliverable generation
5. Gherkin test scenarios generation



Reconciliation Support

- Assistance to create trace links between textual requirements and models
- Consistency checking
- Proposal of recommendations for Model enrichment



Source \ Target	Before 8:00 every calendar day	Check if Document Link exist in	Check if Video Link exist in Vid L	Create 'Share Class / Video Link'	Create a new FundInfo Docume	Create an "Alert" for "Data Man	Create new Fundinfo video link	Delete Invalid Document Links	Delete Invalid Video Links	Discard line	Discard Line	every calendar day	Every day	Forward files	Identify 'Share Class / Video Link	Identify FundInfo Document Link	Identify Fundinfo video link	Identify share class	Identify share class
DDP010														↑					
DDP020														↑					
DDP030																			
DDP035																			
DDP050																			↑
DDP060											↑								
DDP070																↑			

Model Validation Configuration

Enabled Validation

- [ ] Element: Composition
- [ ] Element: Property Validity
- [ ] Element: OCL Conformance
- [ ] Relationship: Well-Formedness
- [ ] Relationship: Property Validity
- [ ] Relationship: OCL Conformance
- [ ] Feature: Well-Formedness
- [ ] Feature: Property Validity
- [ ] Feature: OCL Conformance
- [ ] Diagram: Well-Formedness
- [ ] Requirements Management
- [✓] Drona Rules Intra-Requirements
- [✓] Drona Rules Intra-Diagrams
- [✓] Drona Rules Inter-Diagrams
- [✓] Drona Rules Inter Requirements and Diagrams

System Output

```
MVR810006 - warning (:CBL (ActivityPartition))
MVR810006 - warning (:FundInfo (ActivityParti
MVR020001 - warning (Every Day (Action)): Ev
MVR820009 - warning (Send files (Action)): RC
MVR82000a - warning (Send files (Action)): RC2: The destination of the system response in the NLR does not match with the partition of the AD
MVR82000c - warning (Send files (Action)): RC4: The element(s) received in the system response is/are not representend in the AD as either objects or
Validation complete - 0 error(s), 6 warning(s)
```

Create New Trace Link

Select an option

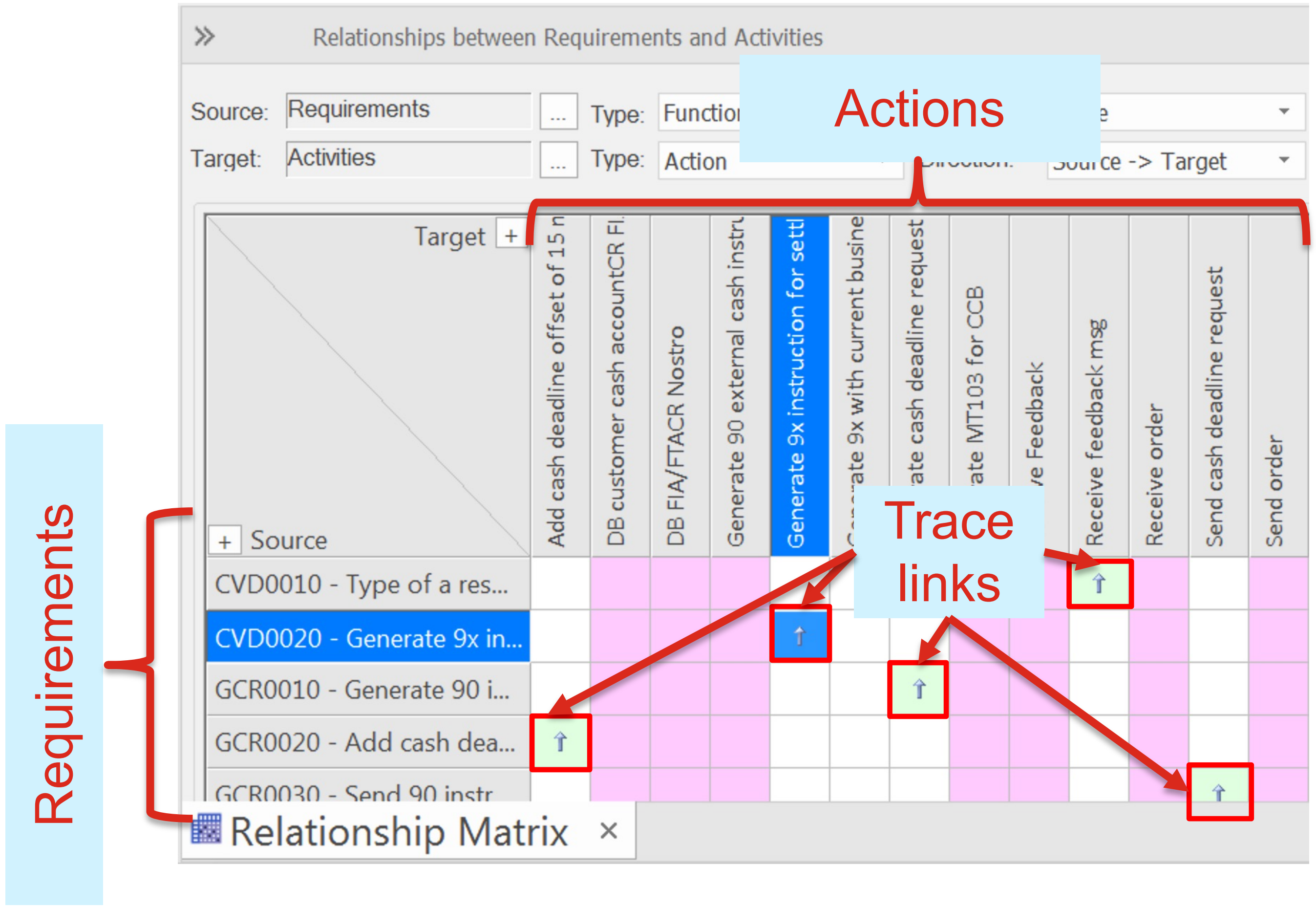
- ○ Create trace link to a new Requirement
- ○ Create trace link to an existing Requirement

Trace Links

- Models typically include trace links between model elements
- Trace links are mainly used in UML for tracking requirements and changes across models
- Trace links between requirements and AD Actions are sufficient in Qualisist as the AC generation is driven by the control flow captured by actions

Relationship Matrix

- **Relationship Matrix** allows to create, edit and view the relationships between, for example, the Requirements and Actions
- **Example** of trace links between Requirements (Rows) and Actions (Columns)



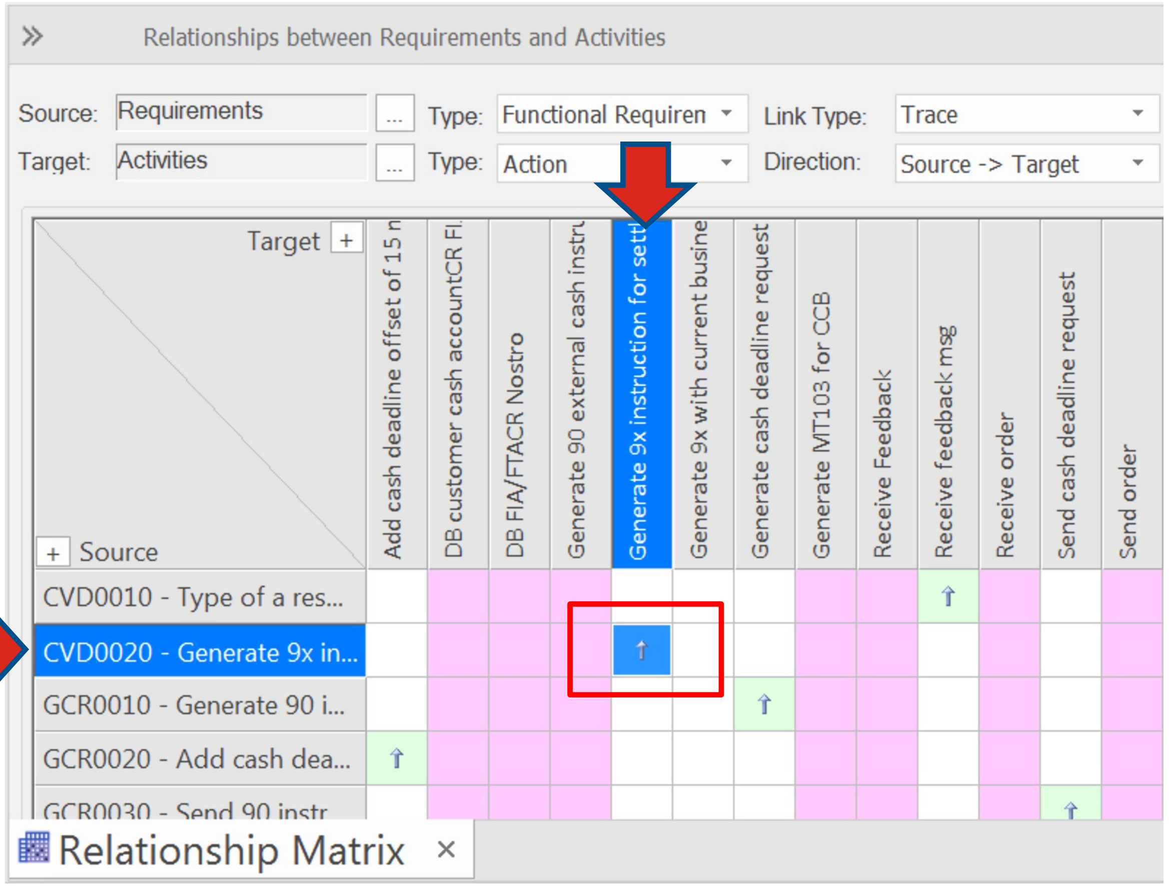
Relationships between Requirements and Activities

Source: Requirements ... Type: Functio
Target: Activities ... Type: Action
Direction: Source -> Target

Target / Source	Add cash deadline offset of 15 n	DB customer cash accountCR FI	DB FIA/FTACR Nostro	Generate 90 external cash instru	Generate 9x instruction for settl	ate 9x with current busine	ate cash deadline request	ate MT103 for CCB	e Feedback	Receive feedback msg	Receive order	Send cash deadline request	Send order
CVD0010 - Type of a res...										↑			
CVD0020 - Generate 9x in...					↑								
GCR0010 - Generate 90 i...							↑						
GCR0020 - Add cash dea...	↑												
GCR0030 - Send 90 instr...												↑	

Trace Links in Qualisist

- An arrow means that the Requirement is traced to the Action, and vice versa
- **Example:** Requirement *CVD0030 is traced to the* Action *Generate 9x instruction for settlement with expected value date*



Relationships between Requirements and Activities

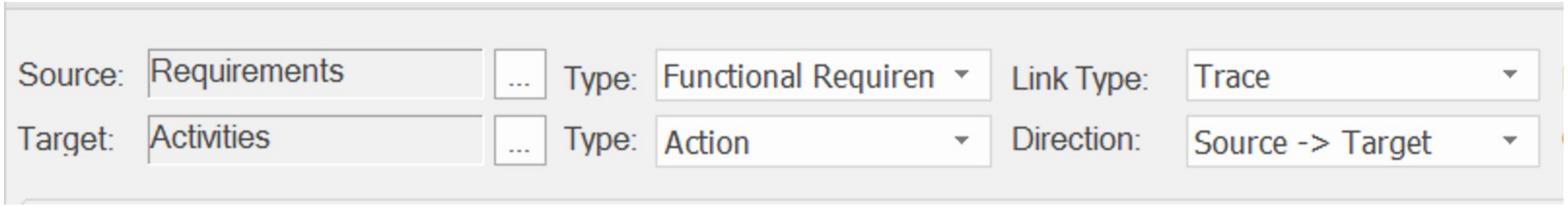
Source: Requirements ... Type: Functional Requiren Link Type: Trace
Target: Activities ... Type: Action Direction: Source -> Target

Target / Source	Add cash deadline offset of 15 n	DB customer cash accountCR FI	DB FIA/FTACR Nostro	Generate 90 external cash instru	Generate 9x instruction for sett	Generate 9x with current busine	Generate cash deadline request	Generate MT103 for CCB	Receive Feedback	Receive feedback msg	Receive order	Send cash deadline request	Send order
CVD0010 - Type of a res...										↑			
CVD0020 - Generate 9x in...					↑								
GCR0010 - Generate 90 i...							↑						
GCR0020 - Add cash dea...	↑												
GCR0030 - Send 90 instr												↑	

Relationship Matrix ×

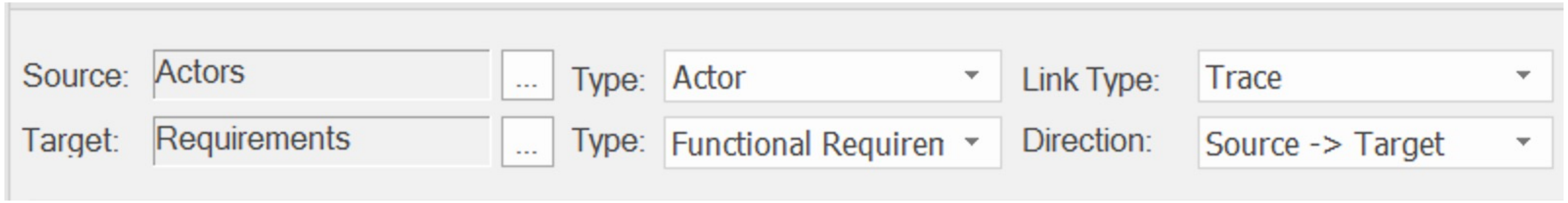
Trace Links in Qualisist

- Qualisist SRA Template provides two predefined Relationship Matrix configurations:
  - Requirements to Actions,



Source:	Requirements	...	Type:	Functional Requiren	Link Type:	Trace
Target:	Activities	...	Type:	Action	Direction:	Source -> Target

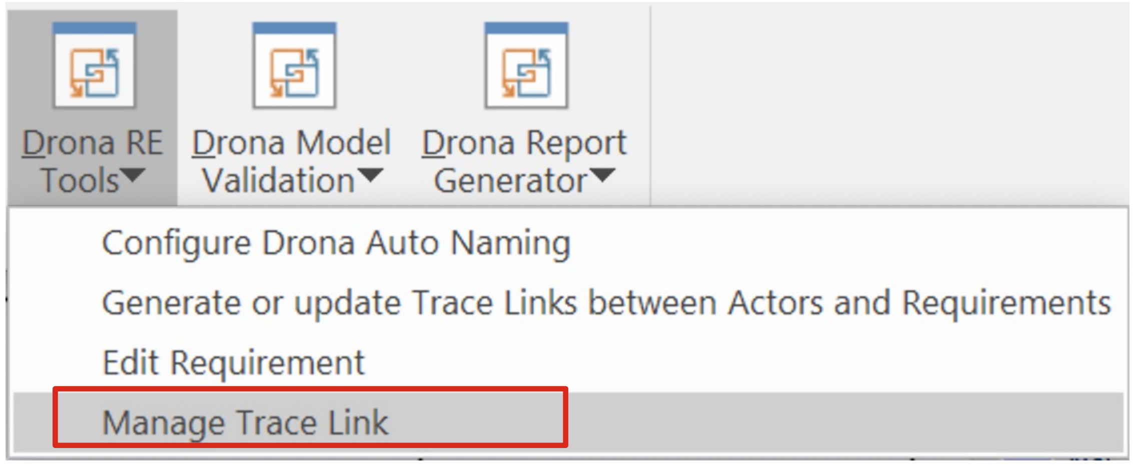
  - Requirements to Actors (represented in Activity Partitions)

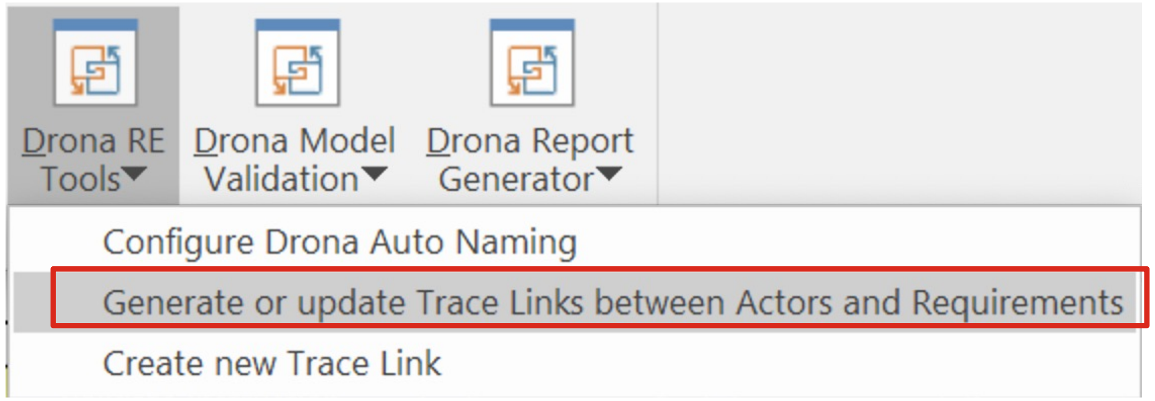


Source:	Actors	...	Type:	Actor	Link Type:	Trace
Target:	Requirements	...	Type:	Functional Requiren	Direction:	Source -> Target

Assistance to Create Trace Links

- Qualisist assist users to manage (create, delete and update) trace links between requirements and actions
- Generate or update trace links between actors and requirements

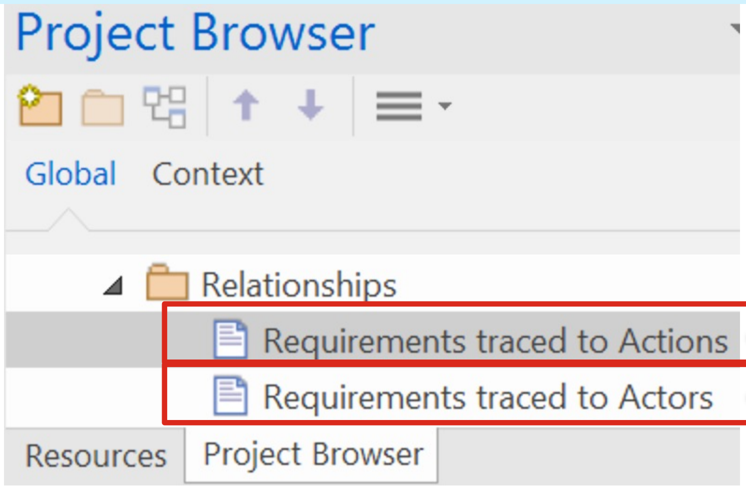



Drona RE Tools
Drona Model Validation
Drona Report Generator
Configure Drona Auto Naming
Generate or update Trace Links between Actors and Requirements
Create new Trace Link

Steps to Open a Preconfigured Relationship Matrix

securityandtrust.lu

1 Doble click on one of the two Matrix Specifications from the package **Relationships**

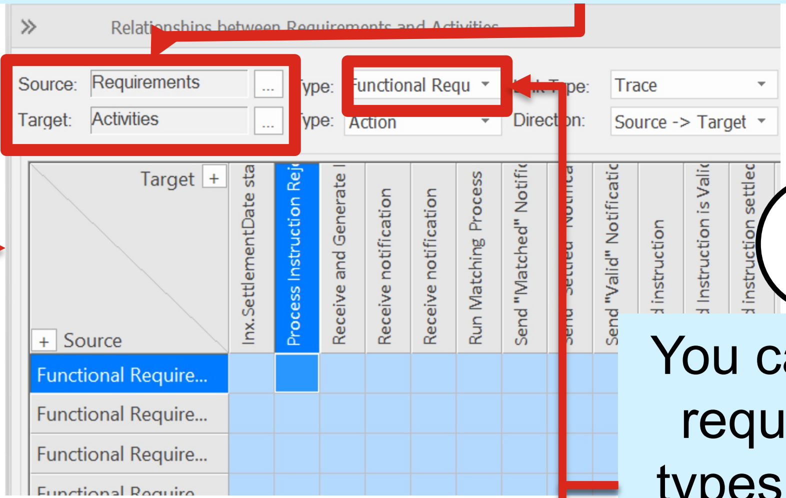


opens

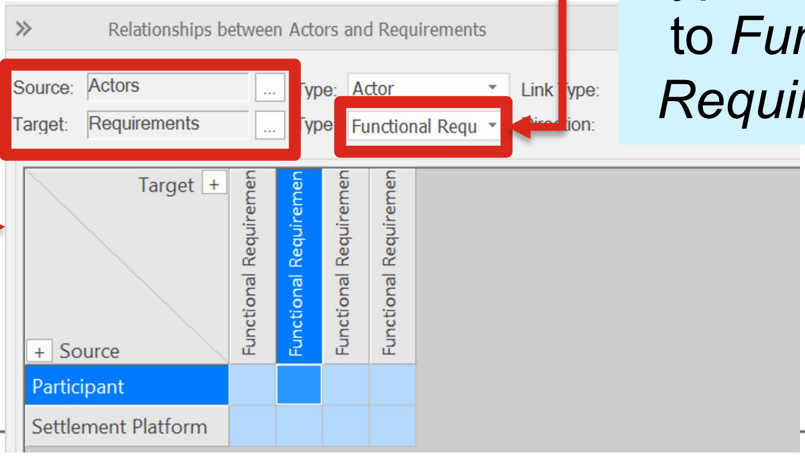
opens

2 You will see a Relationship Matrix

3 You can select other Packages in the fields **Source** and **Target**


Relationships between Requirements and Activities

Source: Requirements
Target: Activities
Type: Functional Requ
Type: Action
Link Type: Trace
Direction: Source -> Target

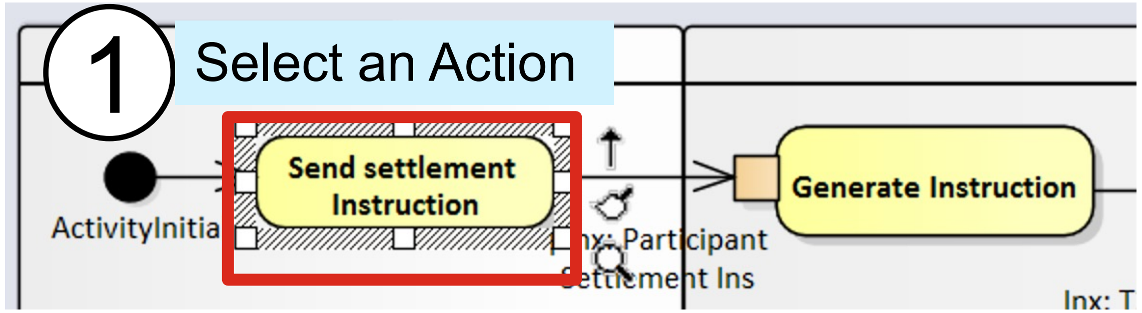

Relationships between Actors and Requirements

Source: Actors
Target: Requirements
Type: Actor
Type: Functional Requ
Link Type:
Direction:

4 You can select requirement types different to *Functional Requirements*

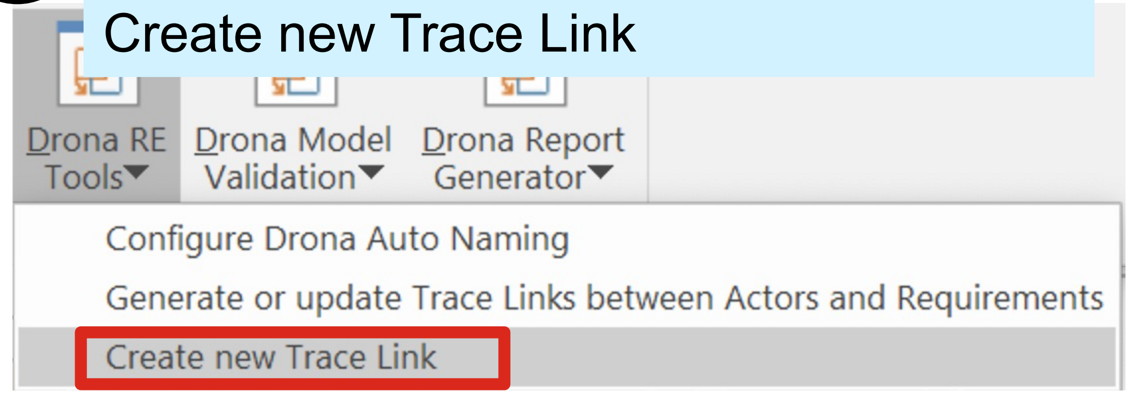
UNIVERSITÉ DU LUXEMBOURG

Steps to Create Trace Links

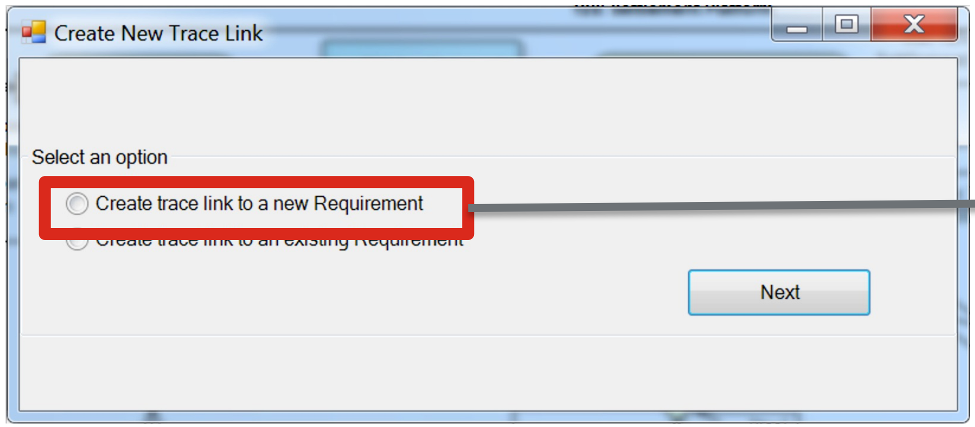


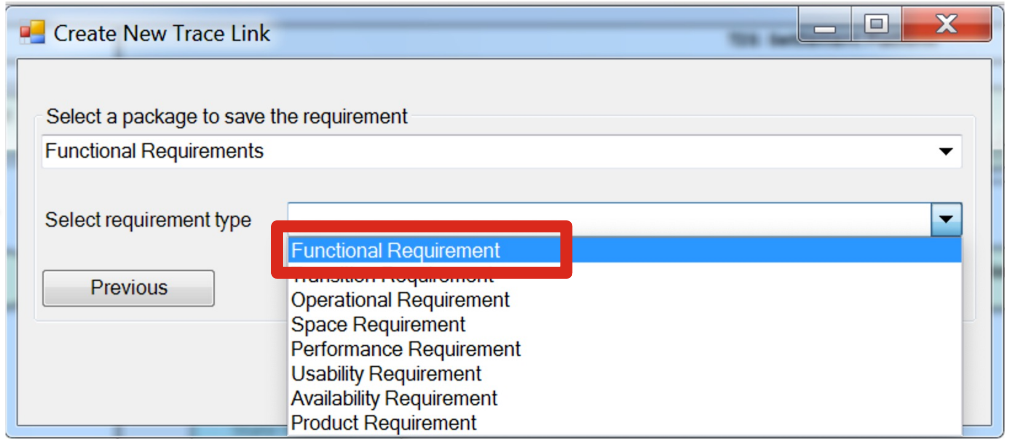
1 Select an Action

2 Go to: Specialize → Qualisist RE Tools → Create new Trace Link


Drona RE Tools
Drona Model Validation
Drona Report Generator
Configure Drona Auto Naming
Generate or update Trace Links between Actors and Requirements
Create new Trace Link

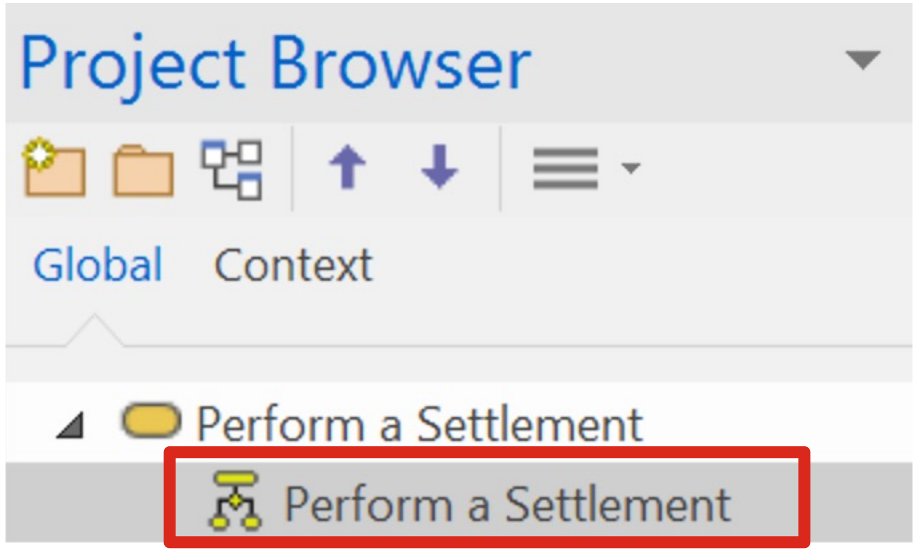
3 Select an alternative and follow the instructions


Create New Trace Link
Select an option
Create trace link to a new Requirement
Create trace link to an existing Requirement
Next

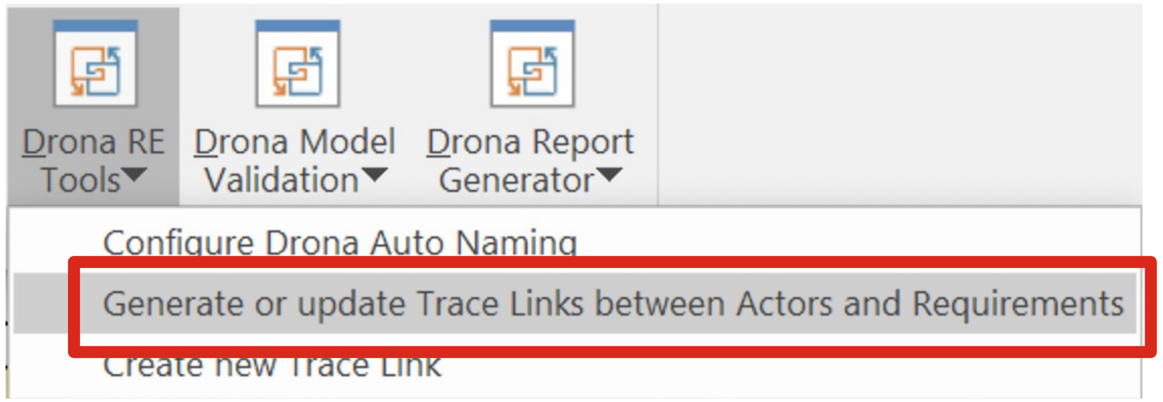


Steps to Generate and Synchronize Traces between Actors and Requirements

1 Select an Activity Diagram



2 Select Specialize → Qualisist RE Tools → Generate or update trace links between Actors and Requirements


Drona RE Tools
Drona Model Validation
Drona Report Generator
Configure Drona Auto Naming
Generate or update Trace Links between Actors and Requirements
Create new Trace Link

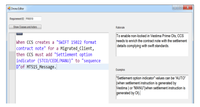
Leveraging Natural-language Requirements for Deriving Better AC from Models

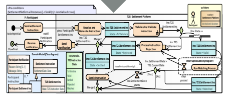
- Generating AC exclusively from models would miss critical information that is available only in NL requirements
- We need to simultaneously consider both models and NL requirements to be able to generate good AC
- Reconciliation of the information content in models and NL requirements is necessary for deriving precise and complete AC.
- Qualisist provides such an automated reconciliation approach and tool.

Main Goal

Requirements Analysis

Business Analysts



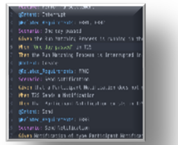


NL Requirements

Acceptance Testing

Test Engineers

Acceptance Criteria

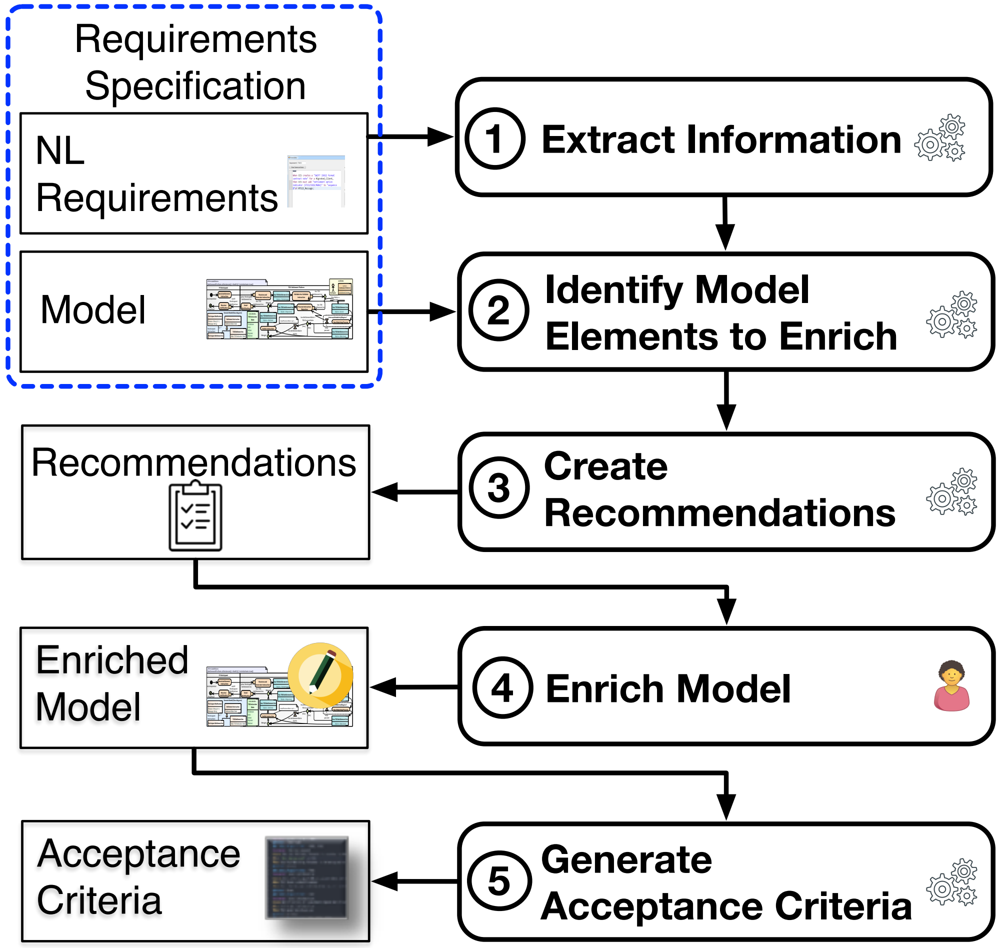


Main Goal

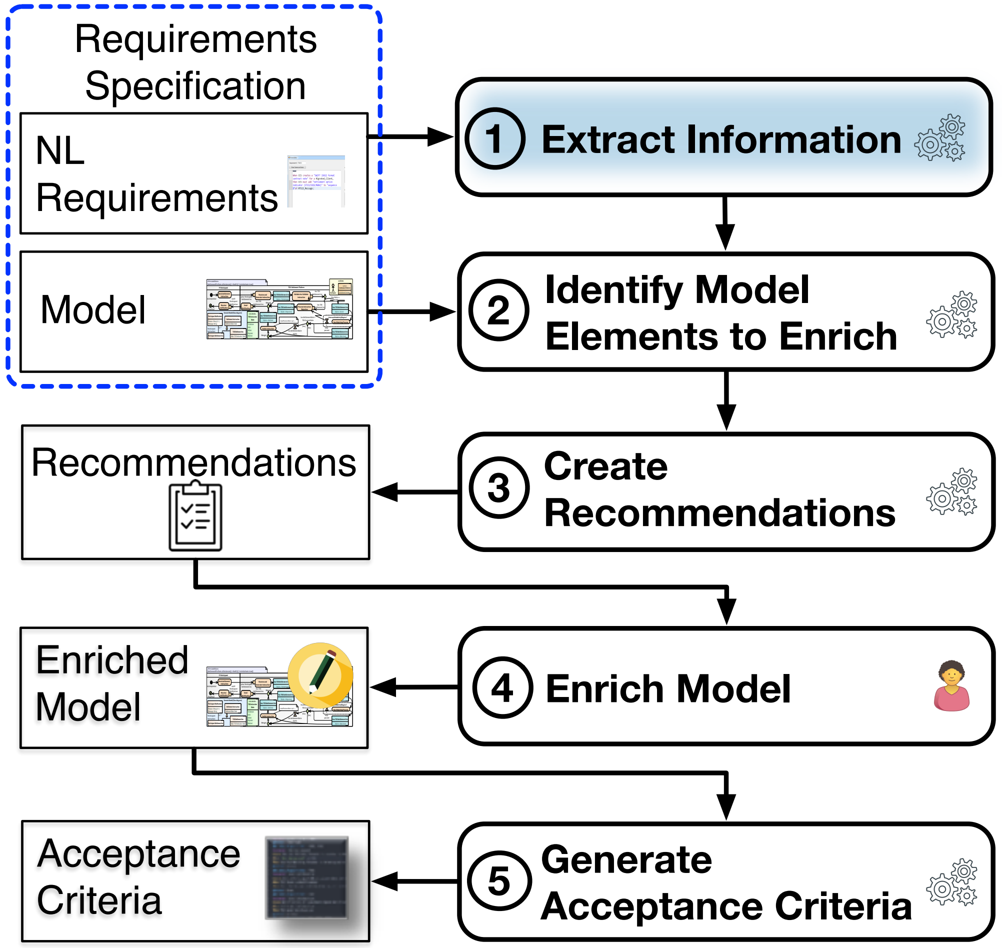
Enrich models with information extracted from NL requirements in order to generate better AC

- Define a set of 13 information extraction rules
- Propose a systematic method that generates recommendations to improve the models with the extracted information

Our Approach



Extract Information



Extract Information

RQ1: How can we extract AC-related information from NL requirements?

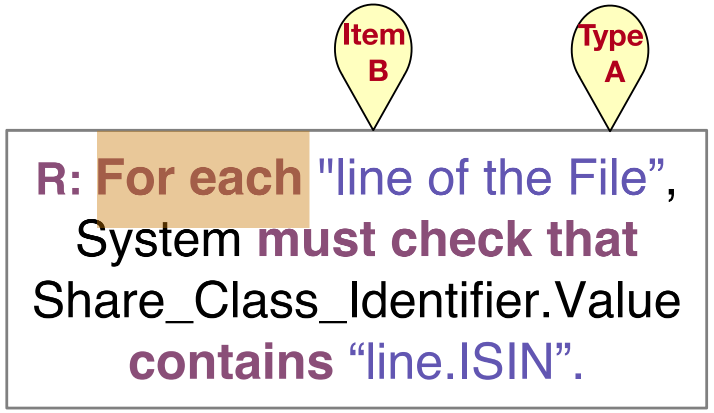
13 rules to extract AC relevant information content from NL requirements

- Derived from manual analysis of overlaps between metamodels element and the element types in Rimay

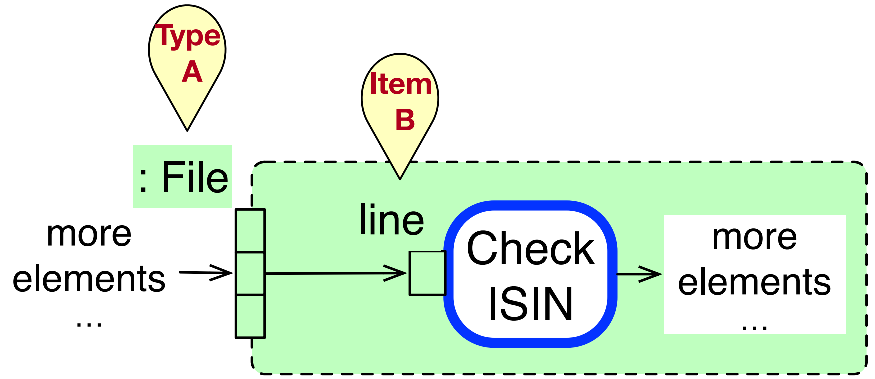
Category	#
Scope	1
Condition Structure	7
Actor	2
System Response	3

Rule Scope S1

- **S1**: If a prepositional phrase starts by "for each", and further mentions: the type ***A*** of the collection that will be iterated over and an item ***B*** in the collection, then extract ***A*** and ***B***.


Item B
Type A

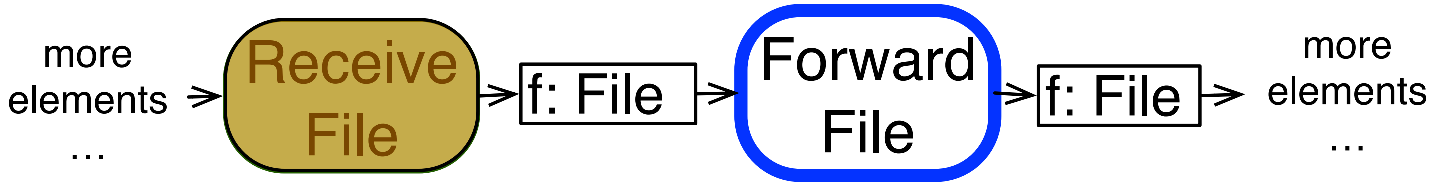
R: For each "line of the File", System **must check that** Share_Class_Identifier.Value **contains** "line.ISIN".



Rule Condition Structure C1

- **C1**: If the verb phrase ***A*** in a When structure does not match the name of any of the actions preceding the traced action, then extract ***A***.

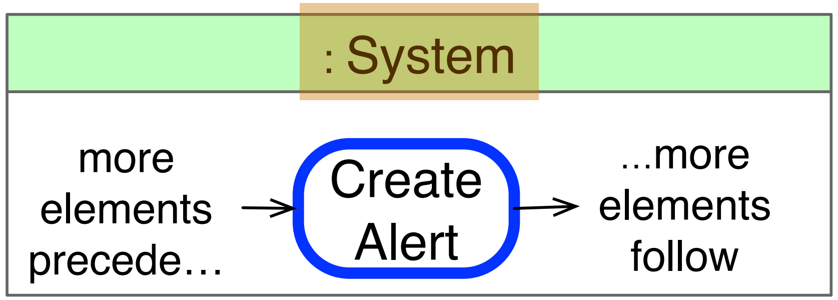
R: When Transfer_System **receives** a File, Transfer_System **must forward** the File **to** System.



Rule Actor A1

- **A1:** If an actor ***A*** in an NL requirement does not match the name of any UML actor linked to the activity partition of the traced action, then extract ***A***.

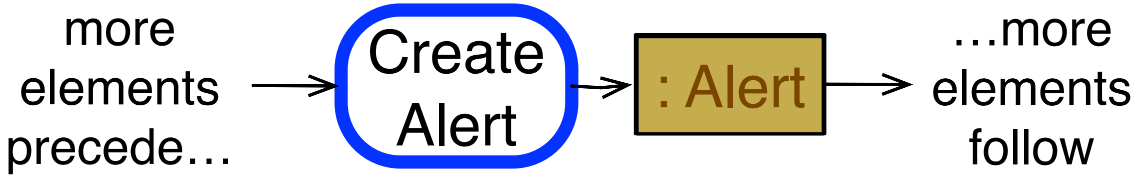
R: Before "8:00 am", **every** "calendar day", **if** System **does not receive the** File,
then System **must create an** "Alert".



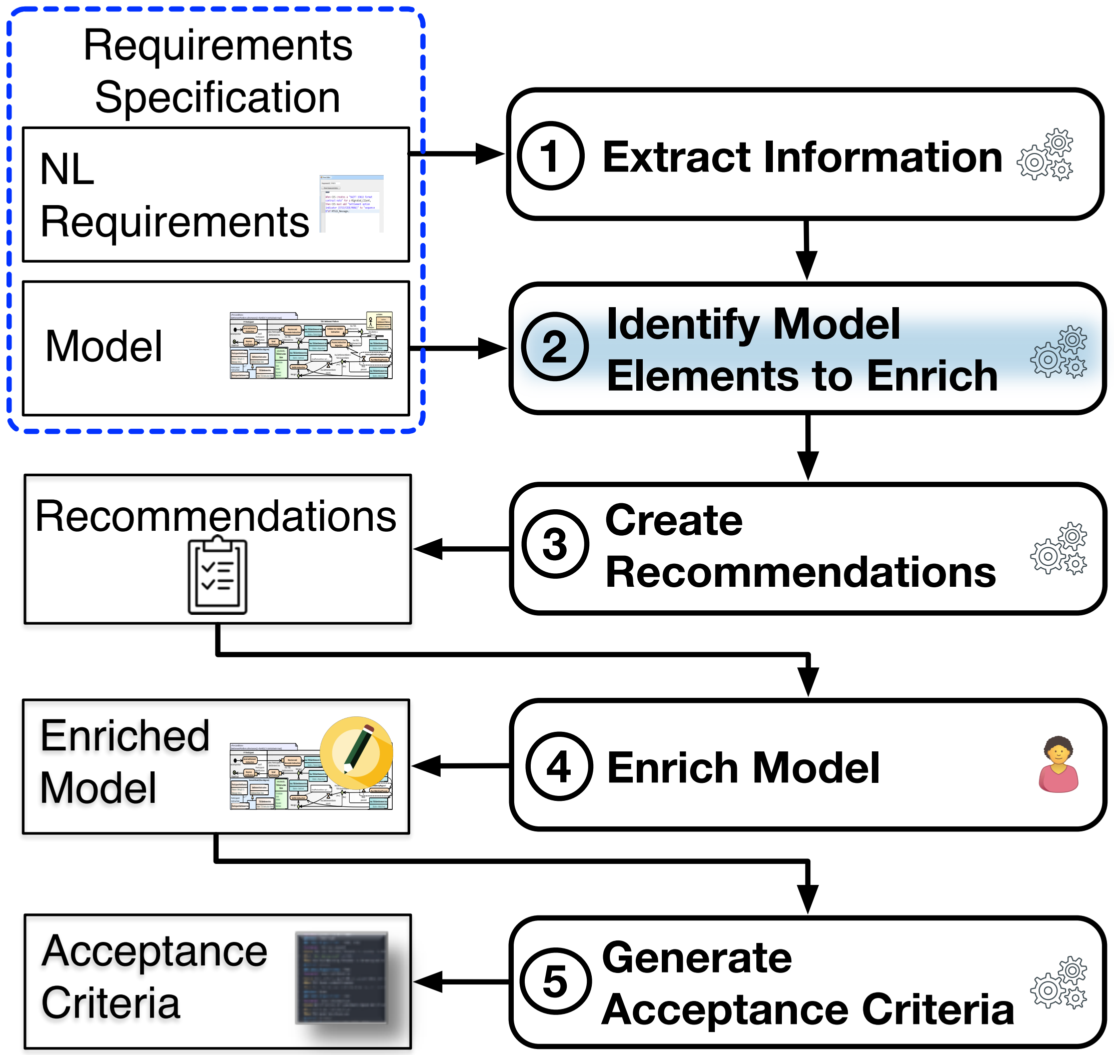
Rule System Response SR1

- **SR1:** If a system response creates data ***A*** (e.g., Report, Instruction, Alarm), then extract ***A***.

R: Before "8:00 am", **every** "calendar day", **if** System **does not receive the** File, **then** System **must create an** "Alert".

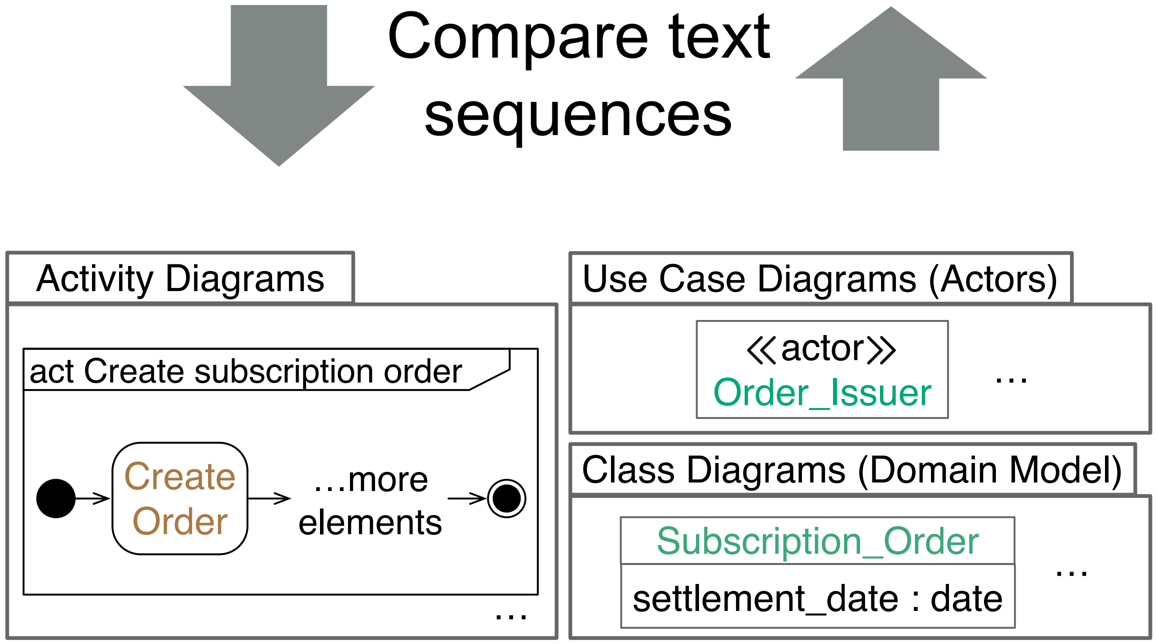


Our Approach

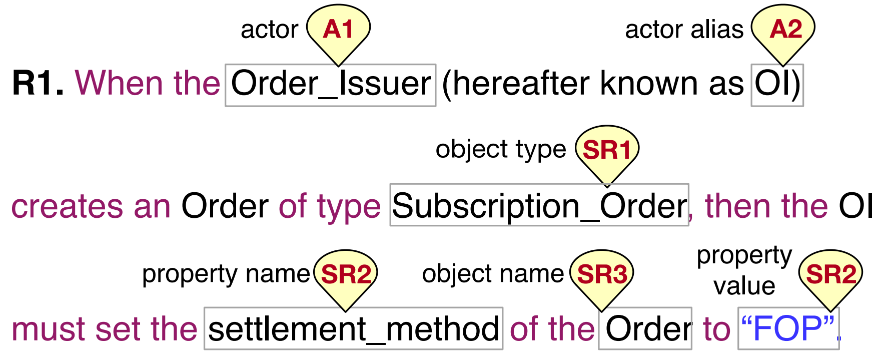


Identify Models Elements to Enrich

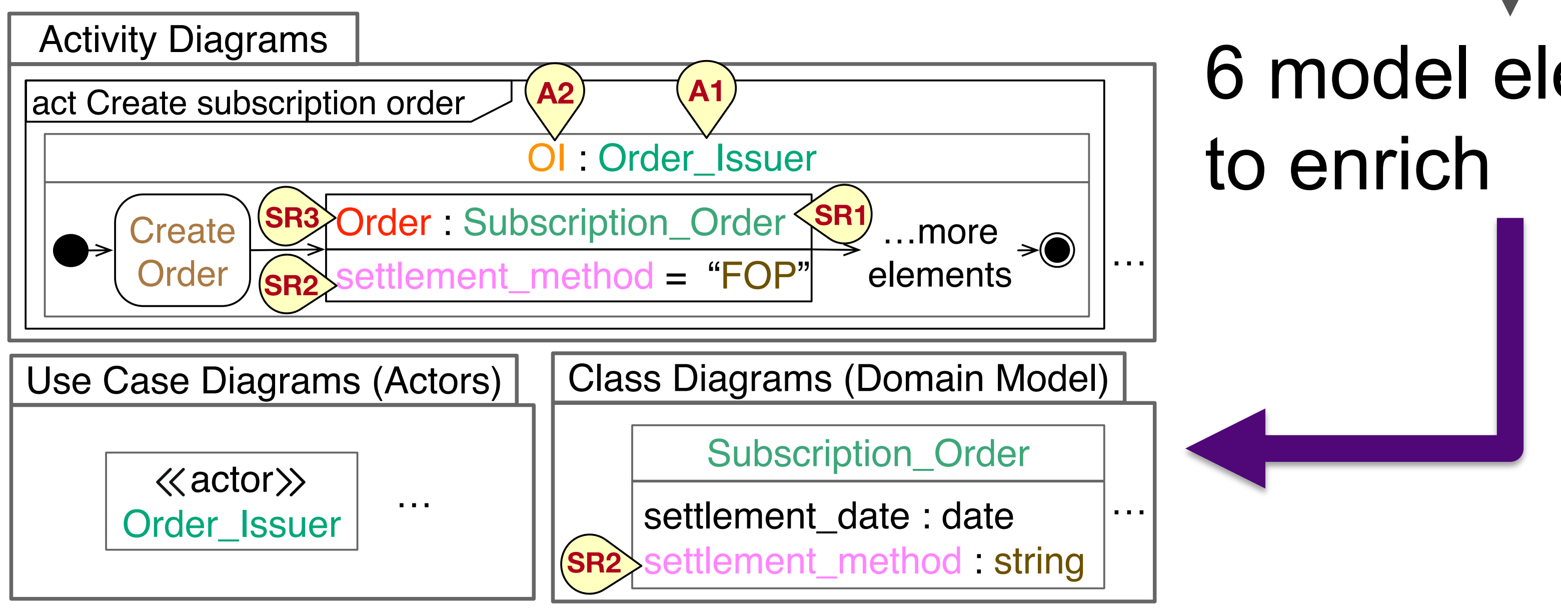
Requirement. When the Order_Issuer (hereafter known as OI) creates an Order of type Subscription_Order, then the OI must set the settlement_method of the Order to "FOP".



Identify Models Elements to Enrich



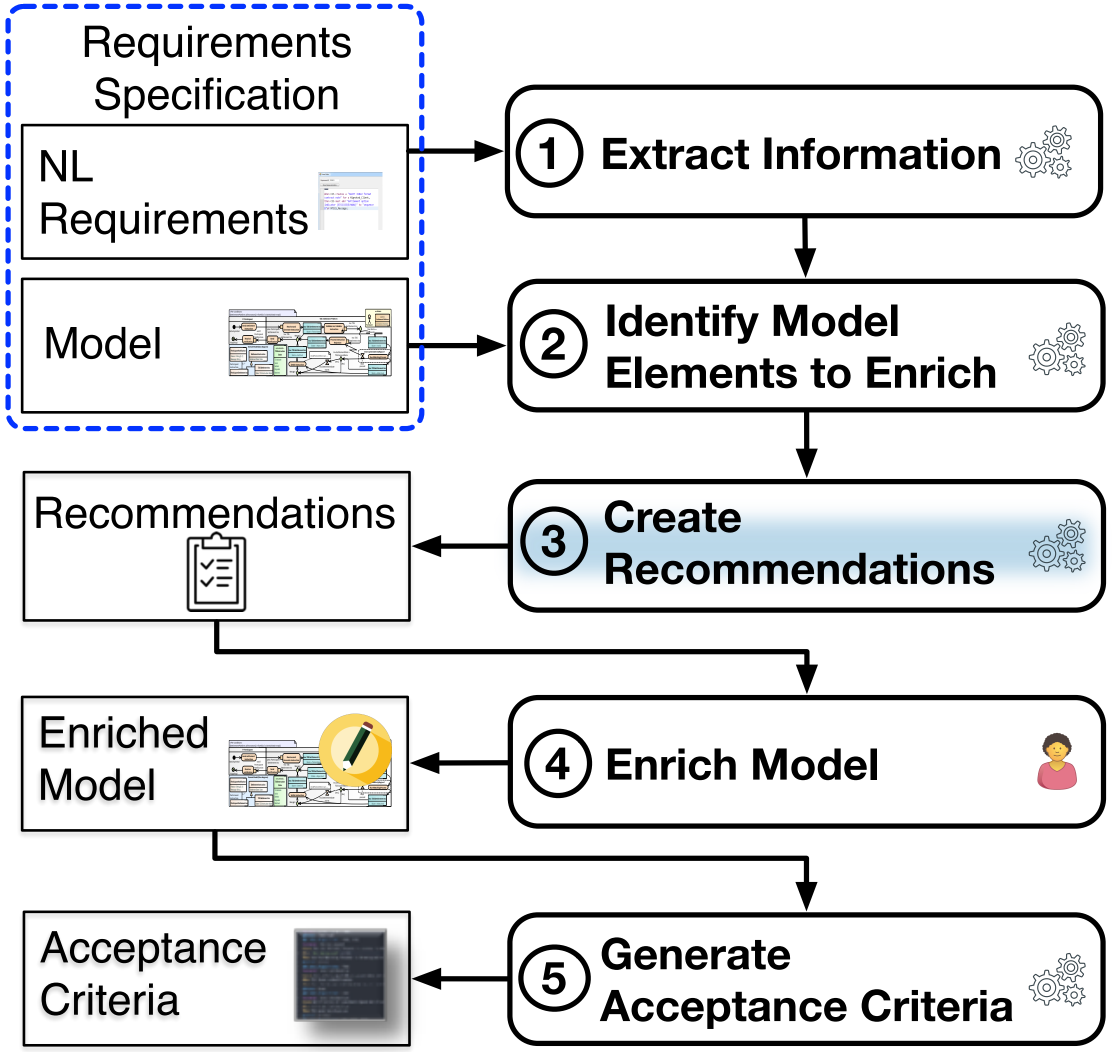
R1. When the Order_Issuer (hereafter known as OI) creates an Order of type Subscription_Order, then the OI must set the settlement_method of the Order to "FOP".



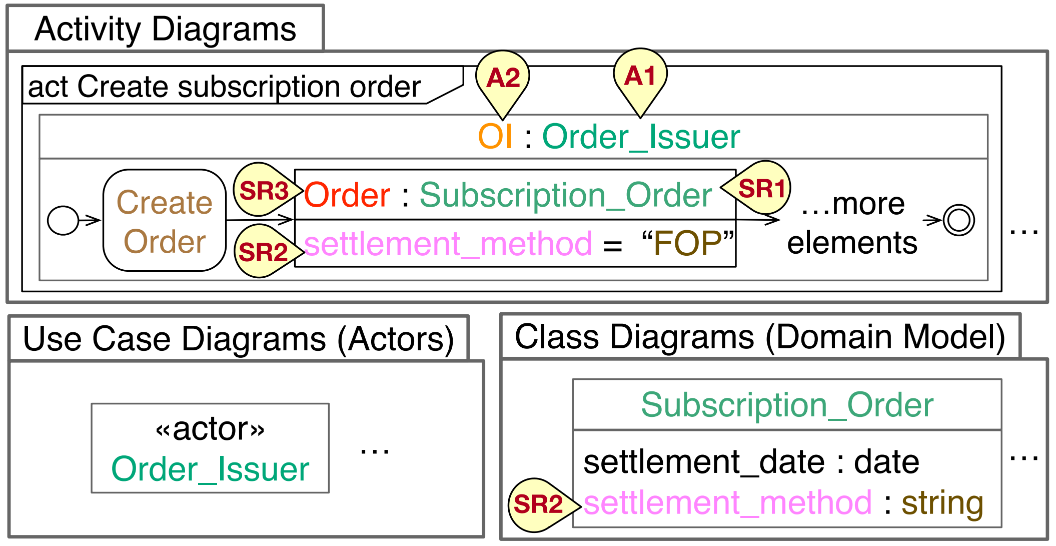
Activity Diagrams
act Create subscription order
A2
A1
OI : Order_Issuer
Create Order
SR3
Order : Subscription_Order
SR1
SR2
settlement_method = "FOP"
…more elements
…
Use Case Diagrams (Actors)
«actor»
Order_Issuer
…
Class Diagrams (Domain Model)
Subscription_Order
settlement_date : date
SR2
settlement_method : string
…

6 model elements to enrich

Our Approach



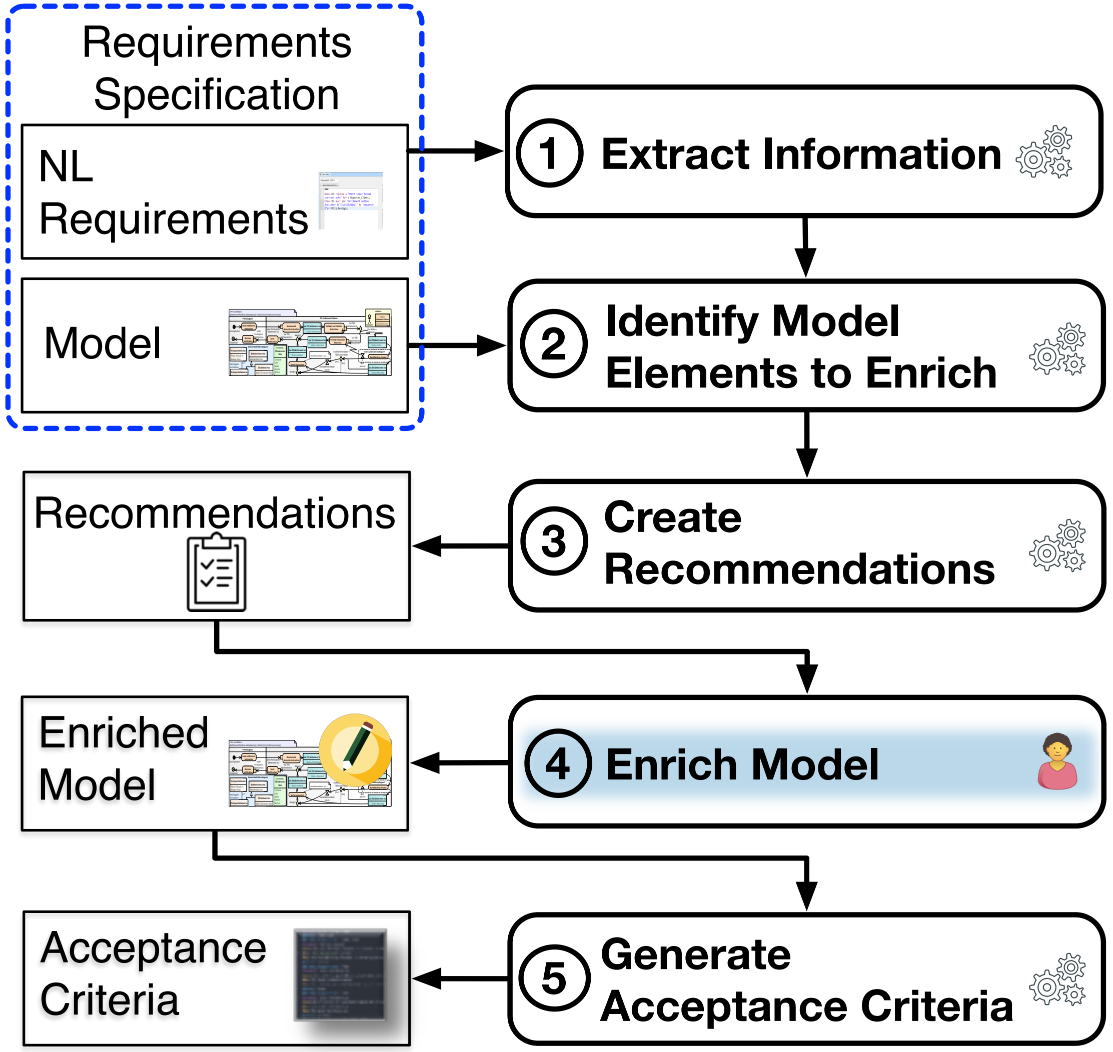
Create Recommendations


Activity Diagrams
act Create subscription order
A2
A1
OI : Order_Issuer
Create Order
SR3
Order : Subscription_Order
SR1
SR2
settlement_method = "FOP"
…more elements
…
Use Case Diagrams (Actors)
«actor» Order_Issuer
…
Class Diagrams (Domain Model)
Subscription_Order
settlement_date : date
SR2
settlement_method : string
…

Recommendations on how to enrich the model elements

ID	Description	Rule
Rec. 4	Add the property "settlement_method" to the object node of type "Subscription Order"	SR2
Rec. 5	Set the "settlement_method" property's value to "FOP"	SR2
…	…	…

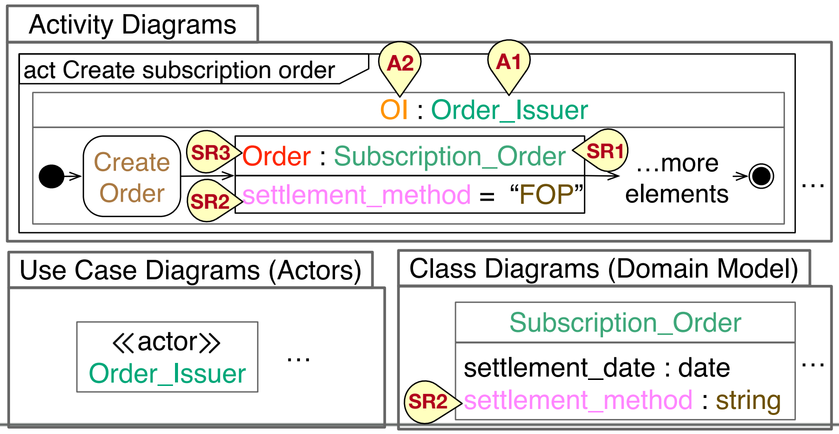
Our Approach



Enrich Model

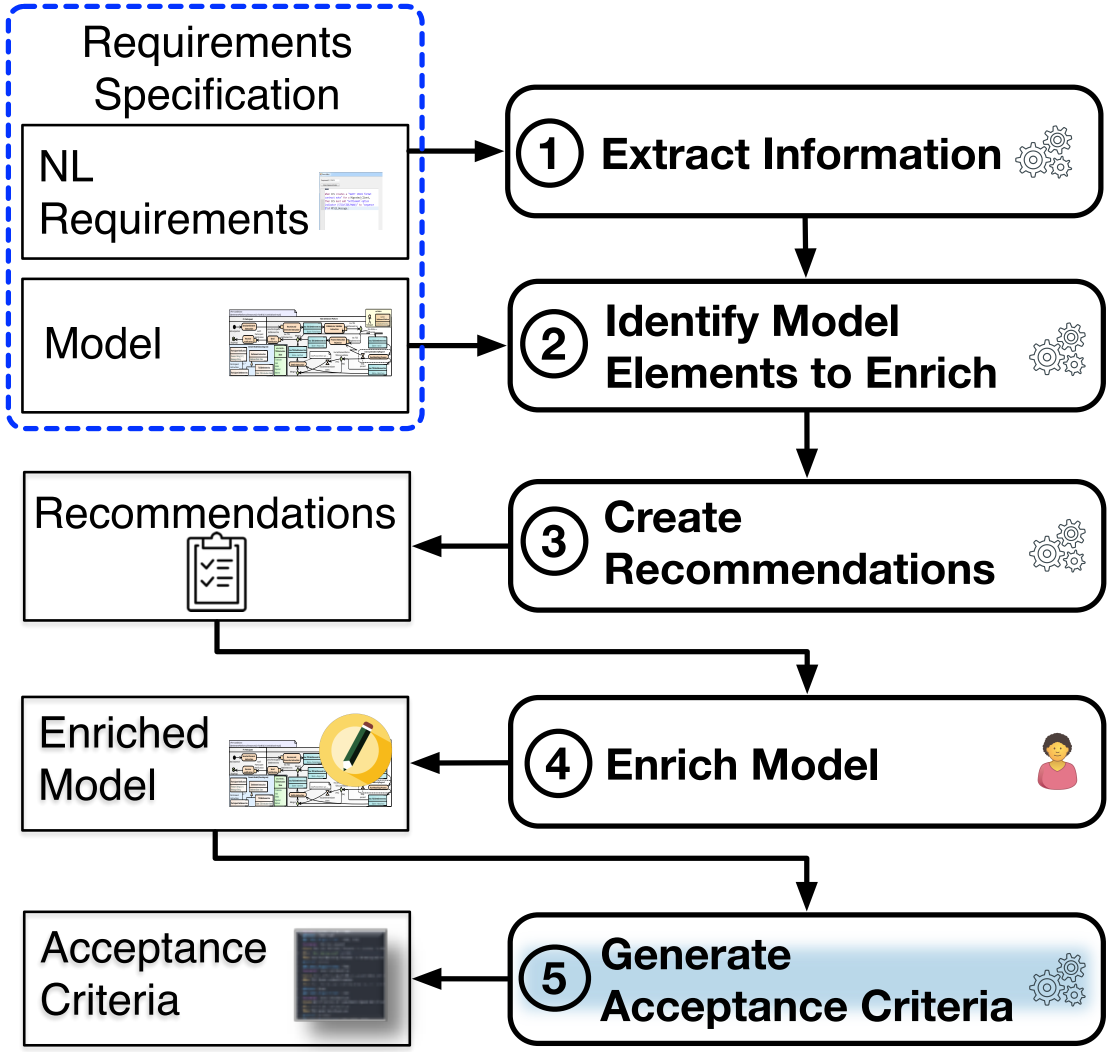
RQ2: How can we systematically enrich models with the (AC-related) information from NL requirements?

ID	Description	Rule
Rec. 4	Add the property "settlement method" to the object node of type "Subscription Order"	SR2
Rec. 5	Set the "settlement_method" property's value to "FOP"	SR2
…	…	…

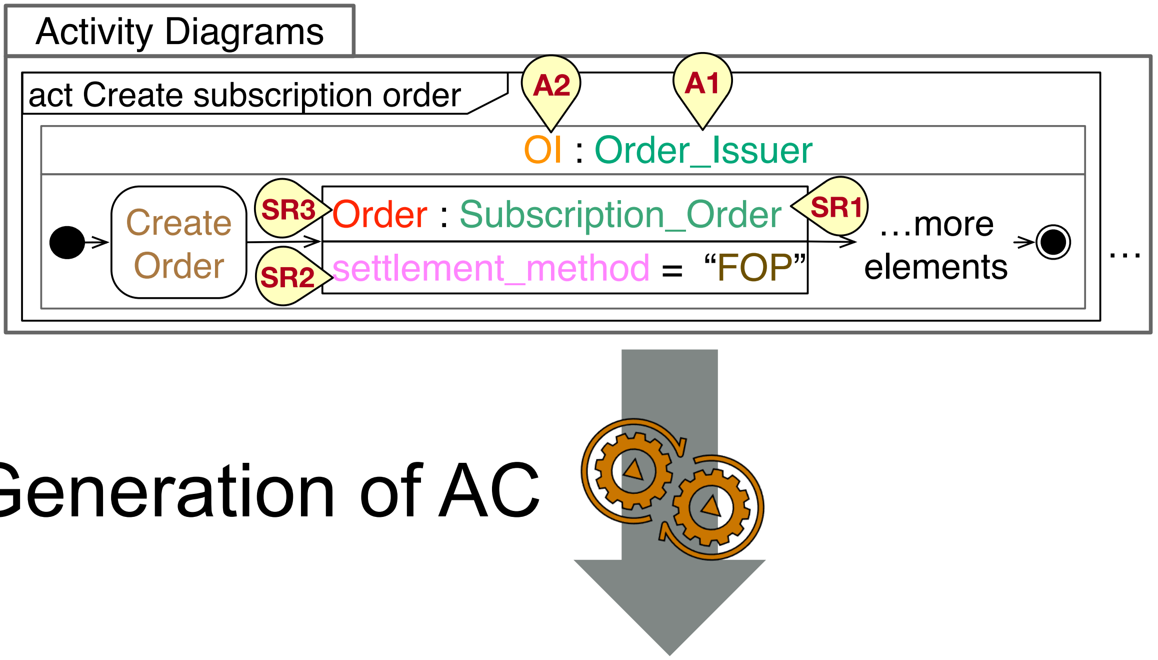

Activity Diagrams
act Create subscription order
A2
A1
OI : Order_Issuer
Create Order
SR3
Order : Subscription_Order
SR1
SR2
settlement_method = "FOP"
...more elements
…
Use Case Diagrams (Actors)
«actor» Order_Issuer
…
Class Diagrams (Domain Model)
Subscription_Order
settlement_date : date
SR2
settlement_method : string
…

Enrich the model elements following recommendations

Our Approach



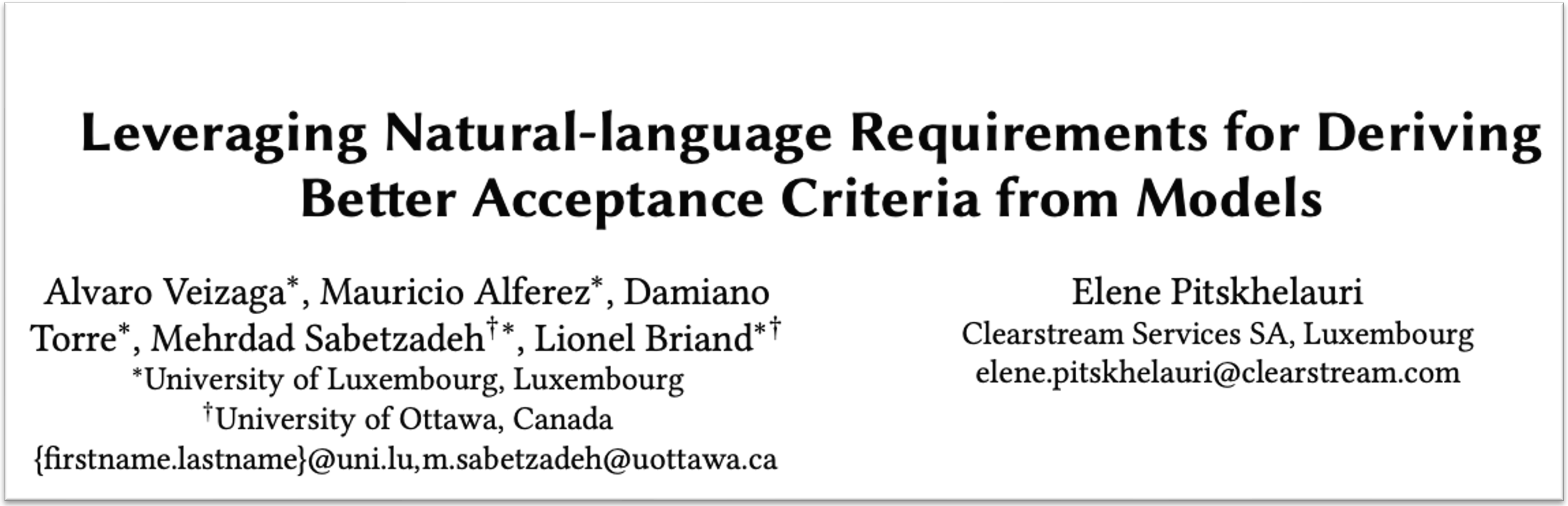
Generate Acceptance Criteria



@*Intent Create*
@*Requirement_Id: R1*
Scenario: Create an Order
Given an Order of type Subscription_Order does not
exist in OI of type Order_Issuer
When OI Create Order,
Then Order exists in OI
And the property settlement_method of Order is equal to FOP

More Information about Reconciliation Approach

- https://orbilu.uni.lu/handle/10993/43900



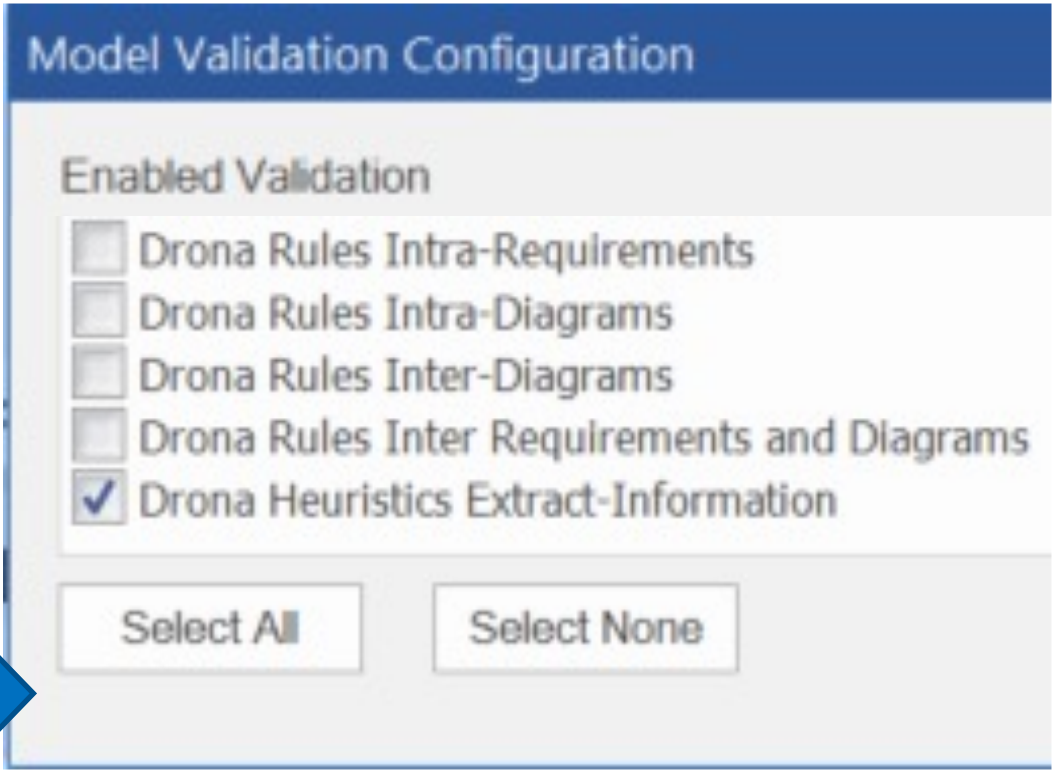
Leveraging Natural-language Requirements for Deriving Better Acceptance Criteria from Models

Alvaro Veizaga[*], Mauricio Alferez[*], Damiano Torre[*], Mehrdad Sabetzadeh[†*], Lionel Briand[*†]
[*]University of Luxembourg, Luxembourg
[†]University of Ottawa, Canada
{firstname.lastname}@uni.lu,m.sabetzadeh@uottawa.ca

Elene Pitskhelauri
Clearstream Services SA, Luxembourg
elene.pitskhelauri@clearstream.com

Practice 4: Reconciliation Support (1/2)

- **Goal:** Learn how to perform automatic requirements-to-model reconciliation and model verification
- **Tasks:**
  1. Open the “**MT103 9x Cash Deadline**” model
  2. Select one of the Qualisist validation rules


Model Validation Configuration

Enabled Validation

- [ ] Drona Rules Intra-Requirements
- [ ] Drona Rules Intra-Diagrams
- [ ] Drona Rules Inter-Diagrams
- [ ] Drona Rules Inter Requirements and Diagrams
- [✓] Drona Heuristics Extract-Information

Practice 5: Reconciliation Support (2/2)

Tasks:

3. Run the validation rules and understand the warning messages

   Shortcut: Ctrl + Alt + v

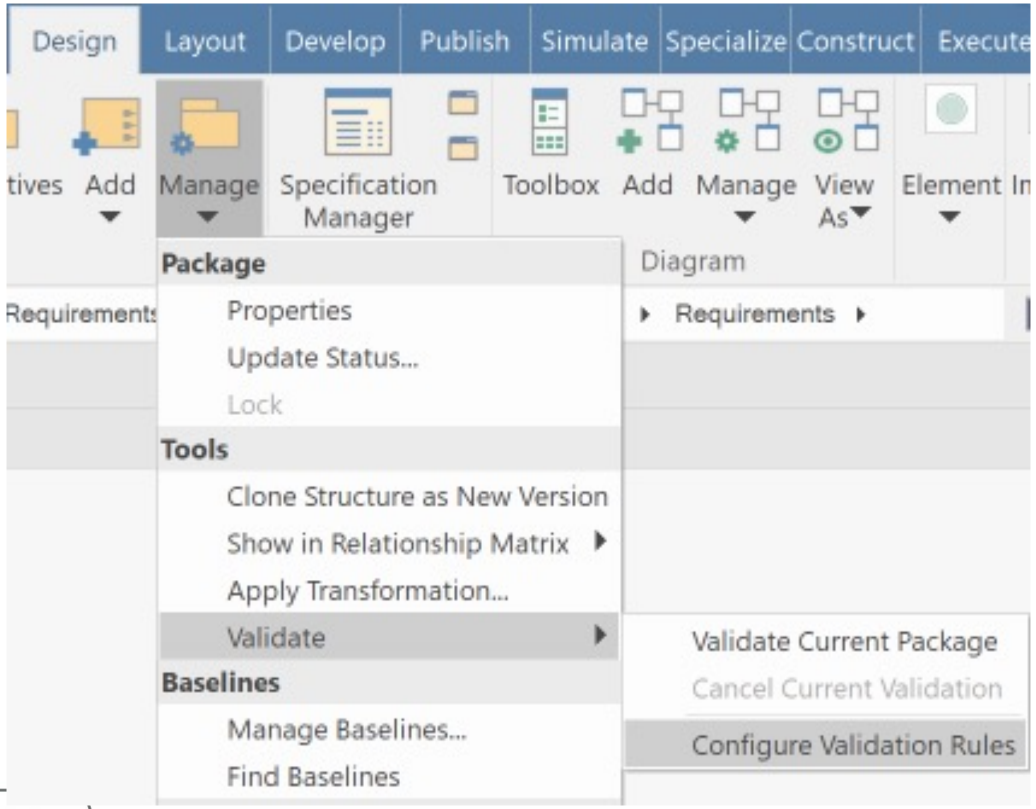
4. Fix the model/requirements related to the warnings

- **Questions:**

  1. How many warnings did you found?
  2. How did you fix the model/requirements?

Steps to Select the Validation Rules

1 Go to Design → Manage → Validate → Configure Validation Rules

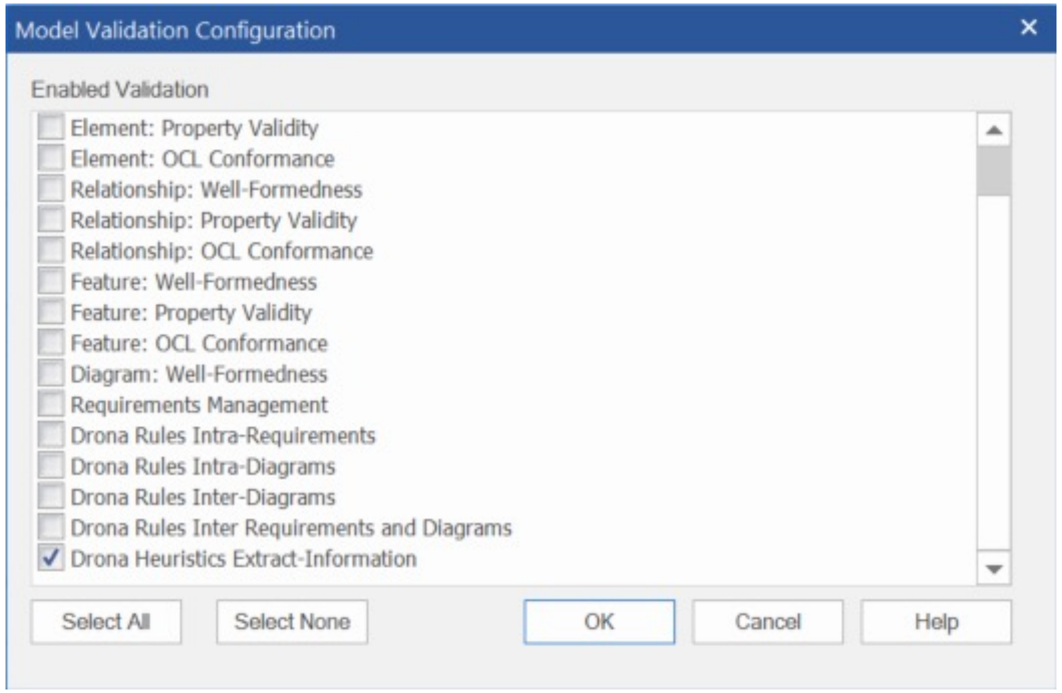


Package
- Properties
- Update Status...
- Lock

Tools
- Clone Structure as New Version
- Show in Relationship Matrix
- Apply Transformation...
- Validate
  - Validate Current Package
  - Cancel Current Validation
  - Configure Validation Rules

Baselines
- Manage Baselines...
- Find Baselines

2 Check that the Qualisist rule's categories



Model Validation Configuration

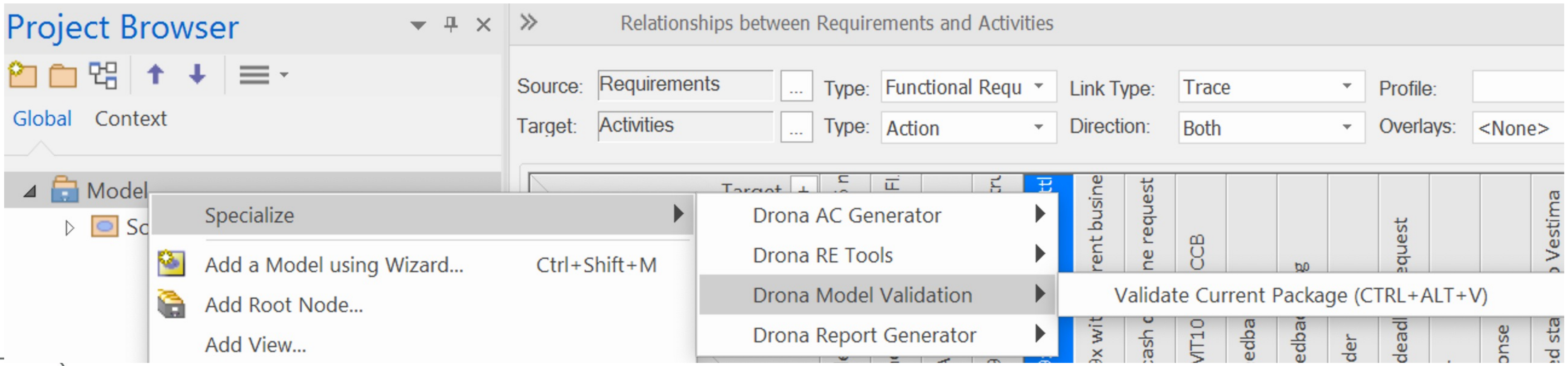
Enabled Validation

- [ ] Element: Property Validity
- [ ] Element: OCL Conformance
- [ ] Relationship: Well-Formedness
- [ ] Relationship: Property Validity
- [ ] Relationship: OCL Conformance
- [ ] Feature: Well-Formedness
- [ ] Feature: Property Validity
- [ ] Feature: OCL Conformance
- [ ] Diagram: Well-Formedness
- [ ] Requirements Management
- [ ] Drona Rules Intra-Requirements
- [ ] Drona Rules Intra-Diagrams
- [ ] Drona Rules Inter-Diagrams
- [ ] Drona Rules Inter Requirements and Diagrams
- [✓] Drona Heuristics Extract-Information

Steps to Validate the Model (or a Package)

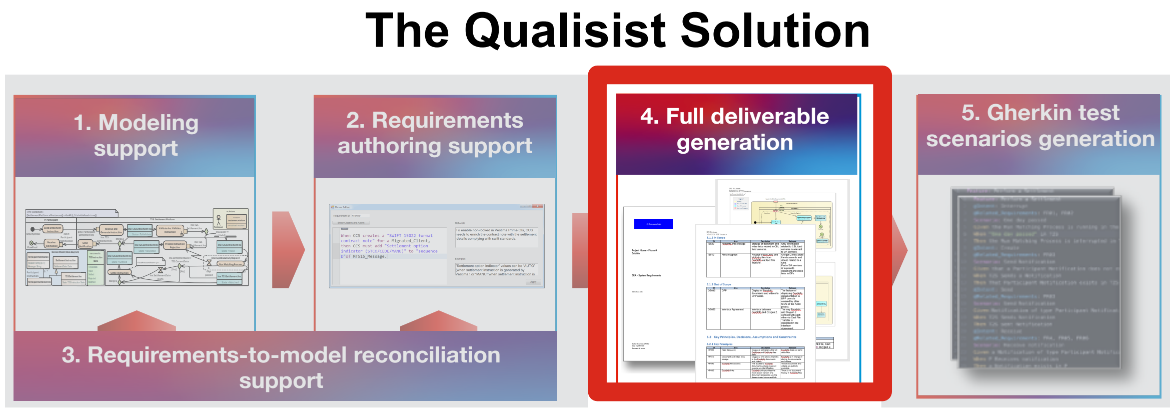
① Select the model or a package in the Project Browser

② **Hit Ctrl + Alt + V** , or select Specialize → Qualisist Model Validation → Validate Current Package



Agenda

0. Installation and Configuration
1. Modelling Support
2. Requirements authoring support
3. Requirements-to-Model reconciliation support
4. **Full deliverable generation**
5. Gherkin test Scenarios generation


The Qualisist Solution
1. Modeling support
2. Requirements authoring support
4. Full deliverable generation
5. Gherkin test scenarios generation
3. Requirements-to-model reconciliation support

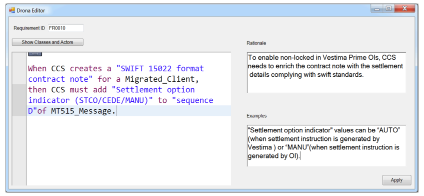
Deliverable Generator

Capture of all deliverable sections in Enterprise Architect ensuring

- ✓ automated generation of full deliverable
- ✓ consistent structure and formatting
- ✓ versioning management (Future Work)

Requirements
in Enterprise Architect

Text


Drona Editor

Requirement ID: FR0010

Show Classes and Actors

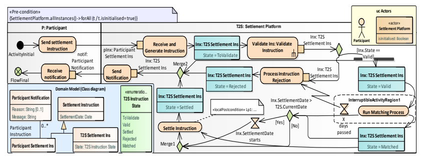
```
When CCS creates a "SWIFT 15022 format
contract note" for a Migrated_Client,
then CCS must add "Settlement option
indicator (STCO/CEDE/MANU)" to "sequence
D"of MT515_Message.
```

Rationale: To enable non-locked in Vestima Prime OIs, CCS needs to enrich the contract note with the settlement details complying with swift standards.

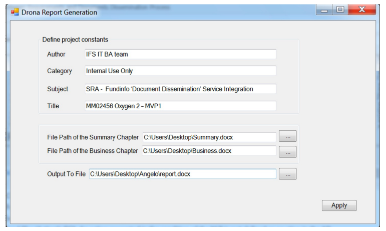
Examples: "Settlement option indicator" values can be “AUTO” (when settlement instruction is generated by Vestima) or “MANU”(when settlement instruction is generated by OI).

Apply

Model



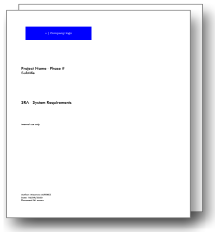
Deliverable
in MS Word


Drona Report Generation

Define project constants

Author	IFS IT BA team
Category	Internal Use Only
Subject	SRA - Fundinfo ‘Document Dissemination’ Service Integration
Title	MM02456 Oxygen 2 – MVP1
File Path of the Summary Chapter	C:\Users\Desktop\Summary.docx
File Path of the Business Chapter	C:\Users\Desktop\Business.docx
Output To File	C:\Users\Desktop\Angelo\report.docx

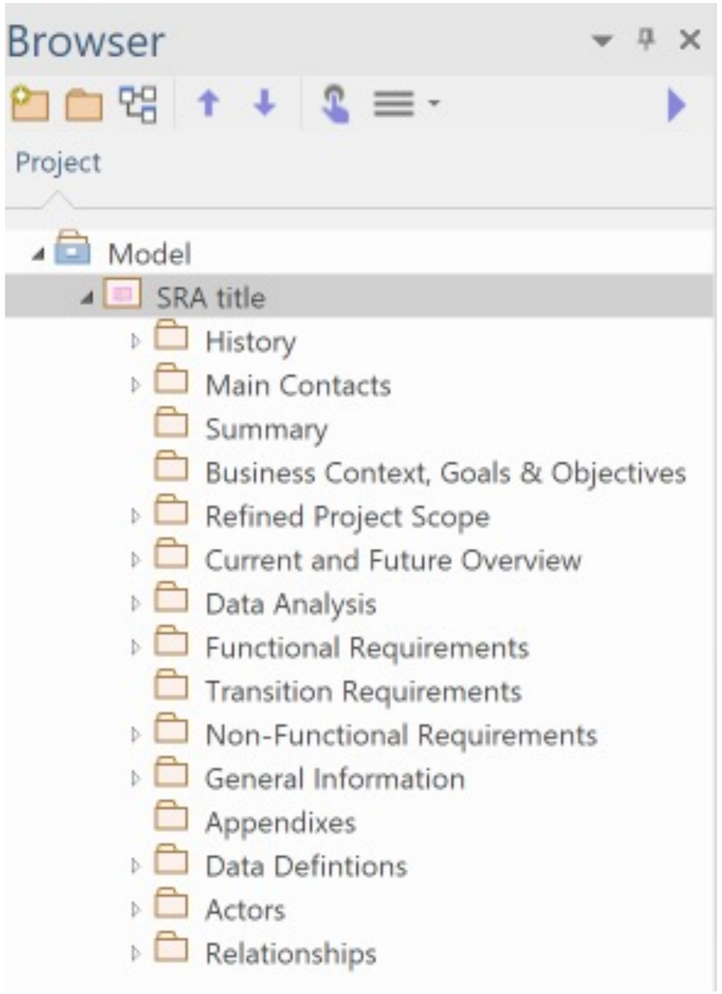
Apply



Read only

Modeling an SRA in Qualisist

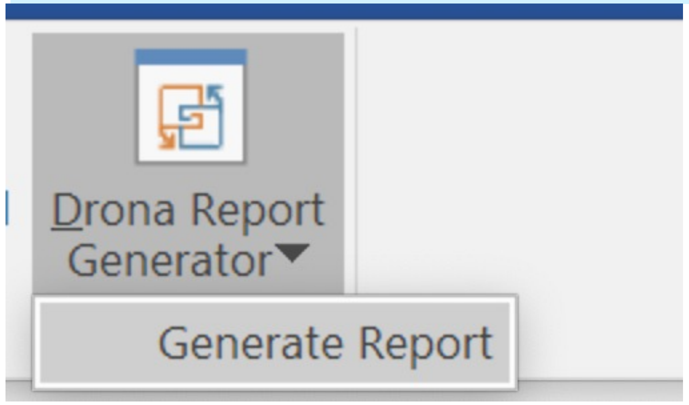
- All the sections are modelled as a Packages
  - Exceptions: Glossary, Acronyms
- Packages are composed of Sub-Packages and Elements
- The Qualisist SRA Model template generates the structure of an SRA
  - See the Modelling section of this training



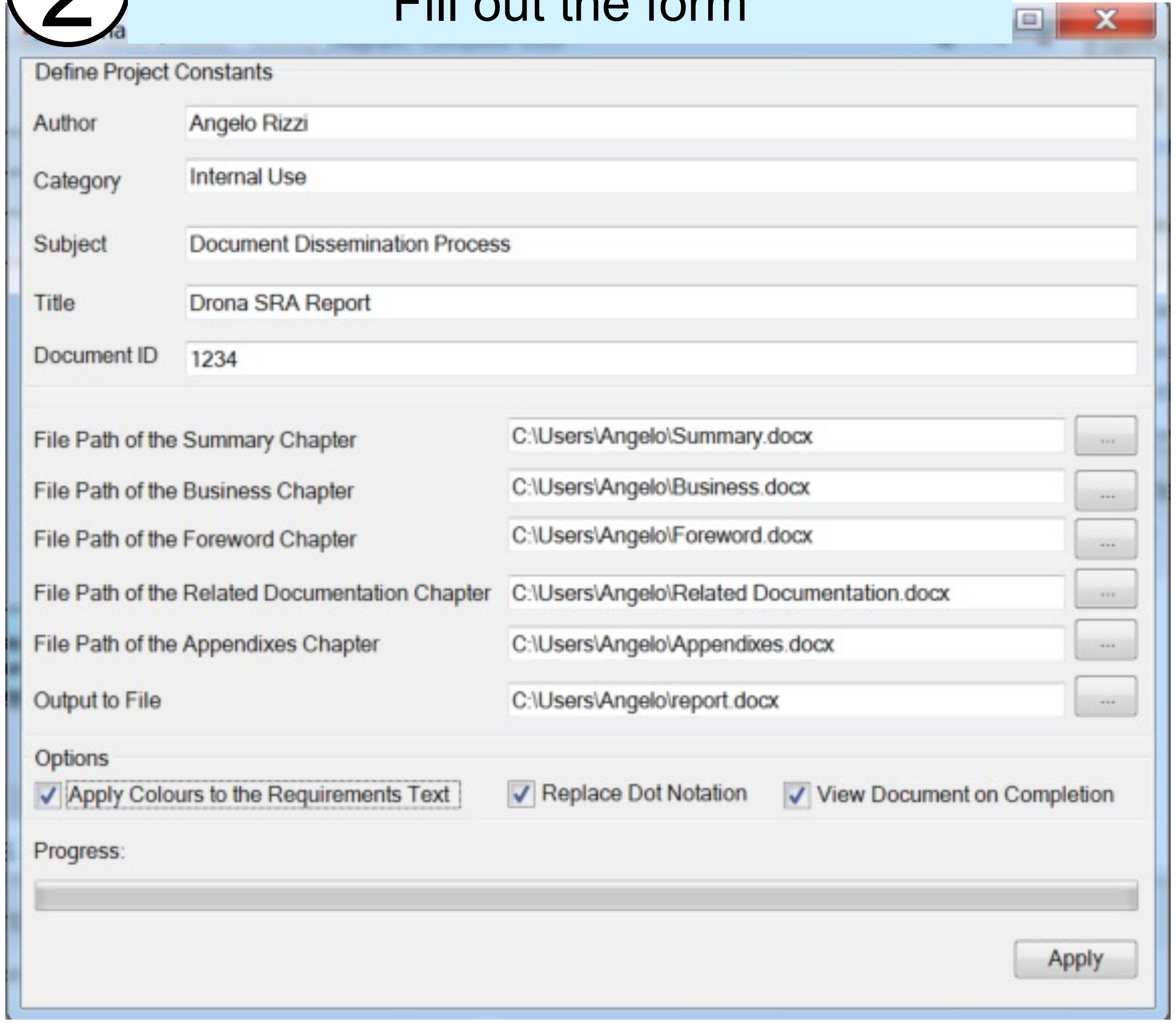
Model
- SRA title
  - History
  - Main Contacts
  - Summary
  - Business Context, Goals & Objectives
  - Refined Project Scope
  - Current and Future Overview
  - Data Analysis
  - Functional Requirements
  - Transition Requirements
  - Non-Functional Requirements
  - General Information
  - Appendixes
  - Data Defintions
  - Actors
  - Relationships

Steps to Fill Out the Deliverable Generator

1 Click on Specialize > Qualisist Report Generator > Generate Report



2 Fill out the form



Define Project Constants

Author	Angelo Rizzi
Category	Internal Use
Subject	Document Dissemination Process
Title	Drona SRA Report
Document ID	1234

File Path of the Summary Chapter	C:\Users\Angelo\Summary.docx
File Path of the Business Chapter	C:\Users\Angelo\Business.docx
File Path of the Foreword Chapter	C:\Users\Angelo\Foreword.docx
File Path of the Related Documentation Chapter	C:\Users\Angelo\Related Documentation.docx
File Path of the Appendixes Chapter	C:\Users\Angelo\Appendixes.docx
Output to File	C:\Users\Angelo\report.docx

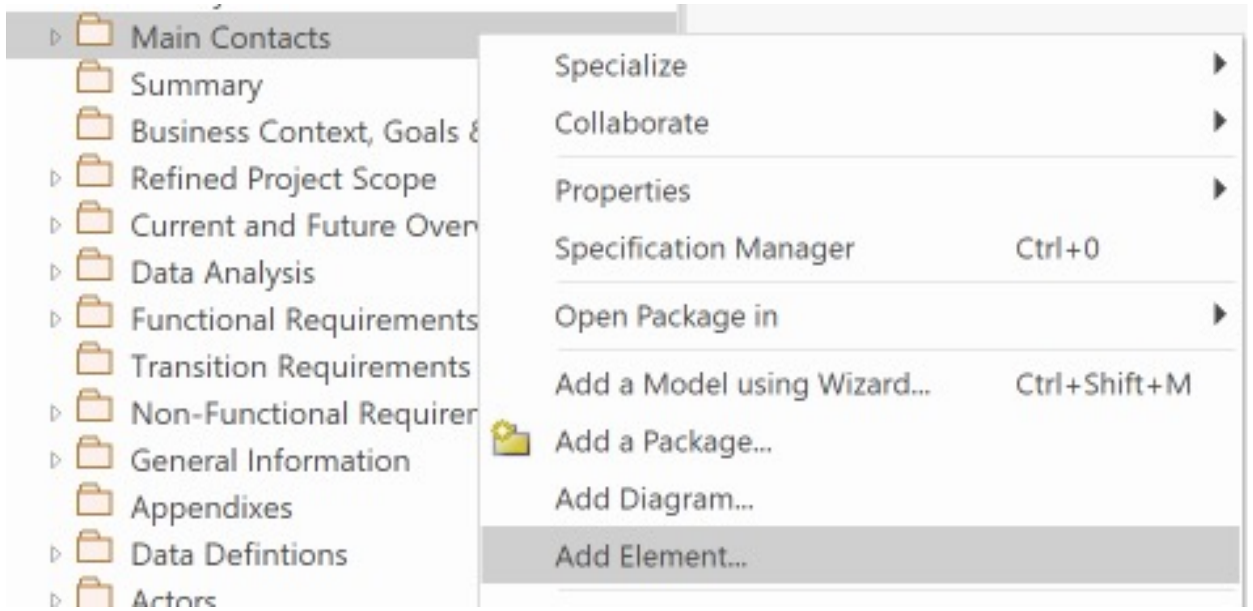
Options

✓ Apply Colours to the Requirements Text ✓ Replace Dot Notation ✓ View Document on Completion

Progress:

Steps to Create an Element in a Package

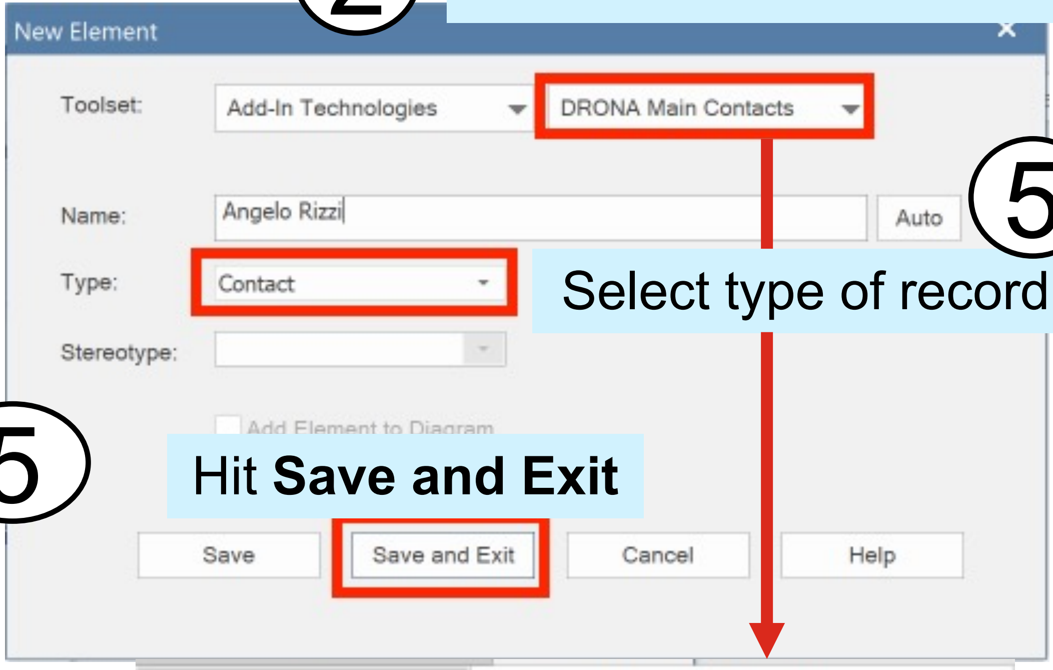
1 Right click on a package and click on **Add Element…**

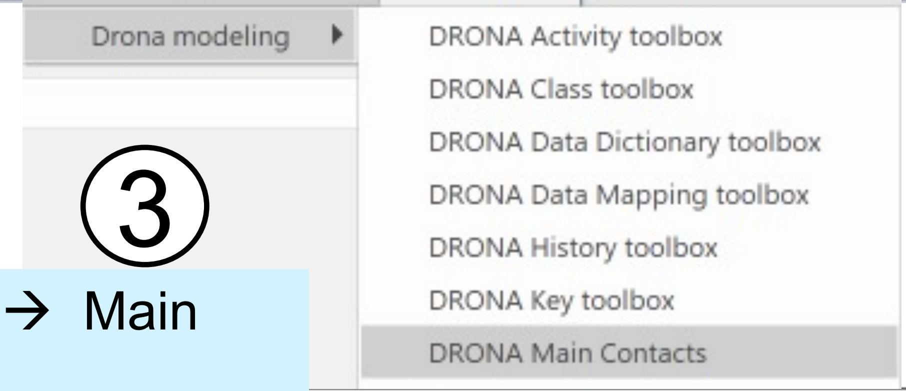


2 Click on **Toolset list**

5 Select type of record

5 Hit **Save and Exit**

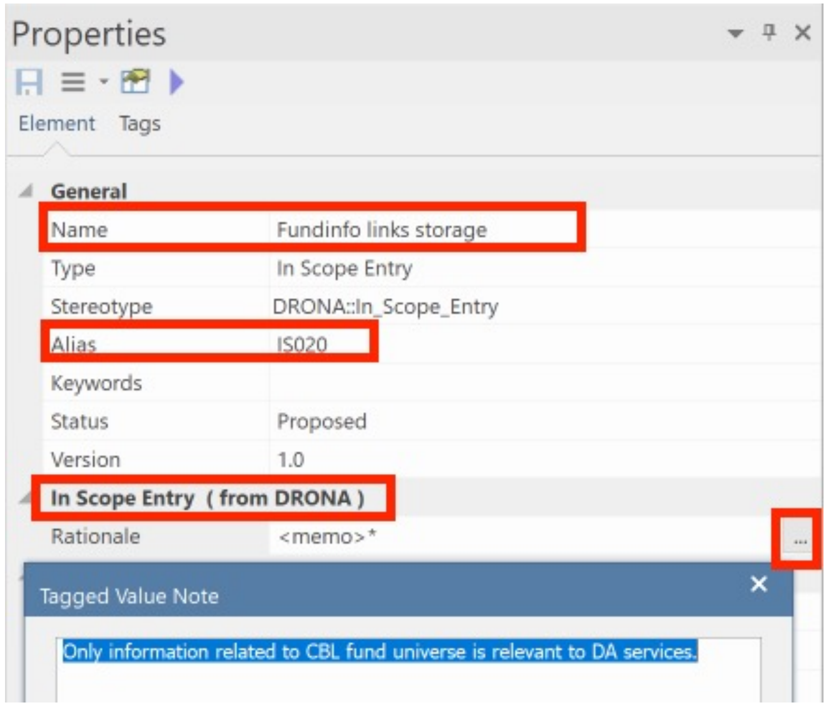




3 Select Qualisist Modelling → Main Contacts

Editing Elements

- Property values are edited in different views, e.g.,
- **Properties** view
  - Use the **General** section to write Alias and Name
  - Use the **Qualisist section** to write other except the Description.
- **Notes** view
  - Use it to write the description



General

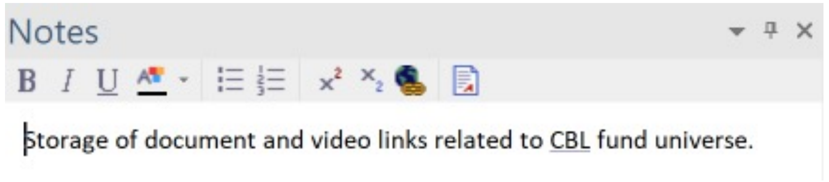
Name	Fundinfo links storage
Type	In Scope Entry
Stereotype	DRONA::In_Scope_Entry
Alias	IS020
Keywords	
Status	Proposed
Version	1.0

In Scope Entry (from DRONA)

Rationale	<memo>*

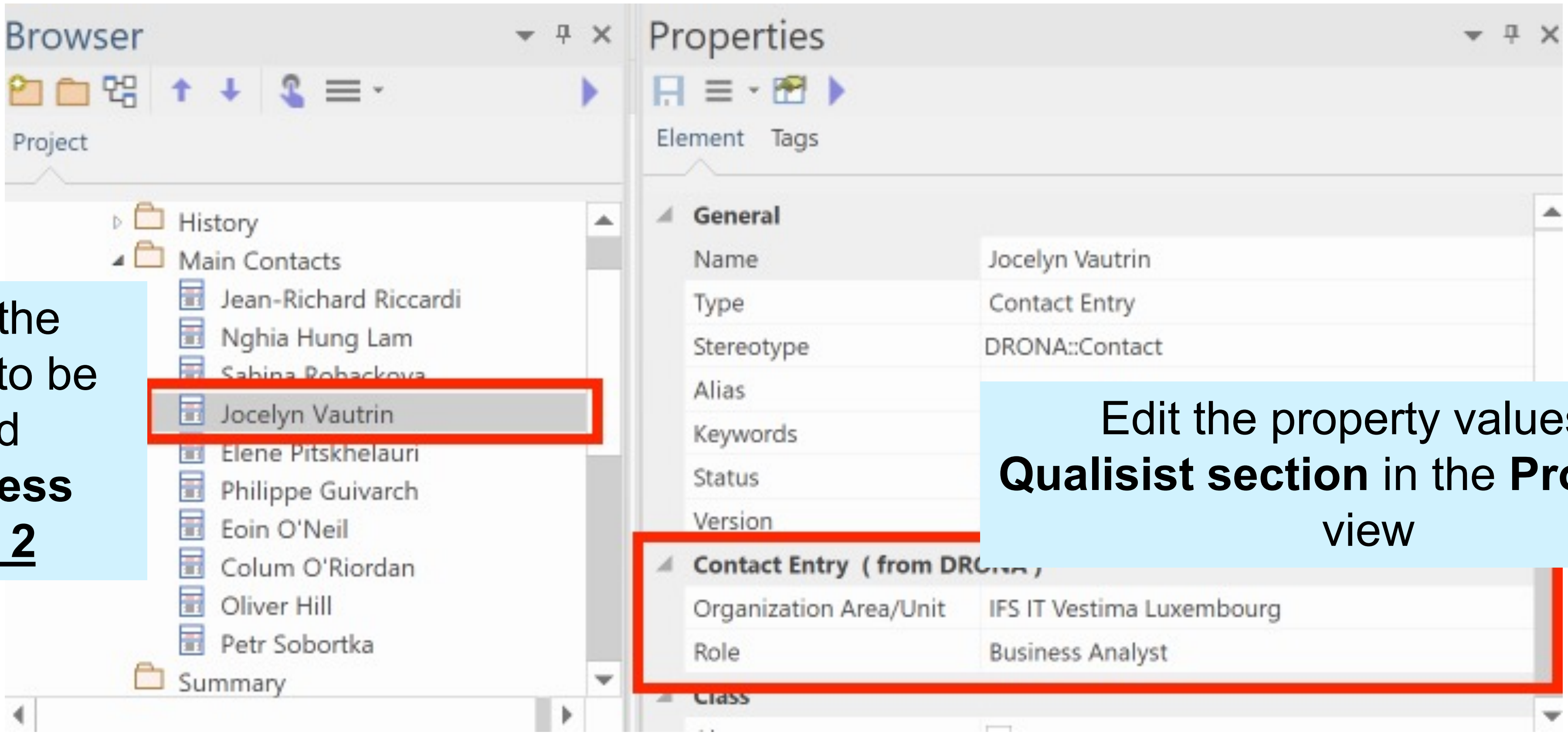
Tagged Value Note

Only information related to CBL fund universe is relevant to DA services.



Storage of document and video links related to CBL fund universe.

Steps to Edit Qualisist Element's Properties in the Property View



Browser

Project

- History
- Main Contacts
  - Jean-Richard Riccardi
  - Nghia Hung Lam
  - Sabina Rohackova
  - Jocelyn Vautrin
  - Elene Pitskhelauri
  - Philippe Guivarch
  - Eoin O'Neil
  - Colum O'Riordan
  - Oliver Hill
  - Petr Sobortka
- Summary

Properties

Element Tags

General	
Name	Jocelyn Vautrin
Type	Contact Entry
Stereotype	DRONA::Contact
Alias	
Keywords	
Status	
Version	
Contact Entry (from DRONA)	
Organization Area/Unit	IFS IT Vestima Luxembourg
Role	Business Analyst
Class	

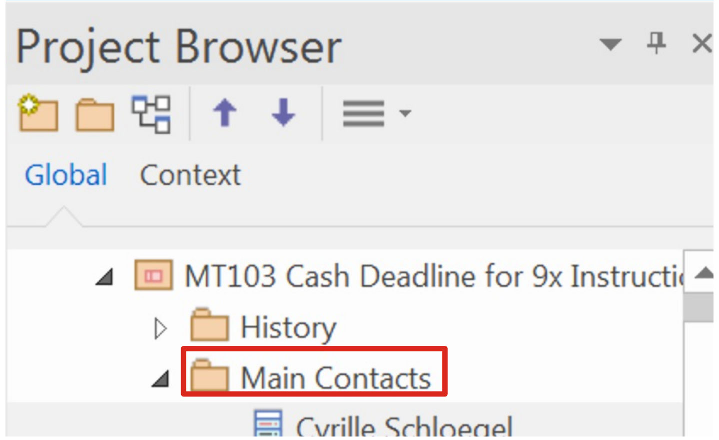
① Select the element to be edited **and press <u>Ctrl + 2</u>**

② Edit the property values in **Qualisist section** in the **Properties** view

Steps to Edit Qualisist Elements in the Qualisist View

securityandtrust.lu

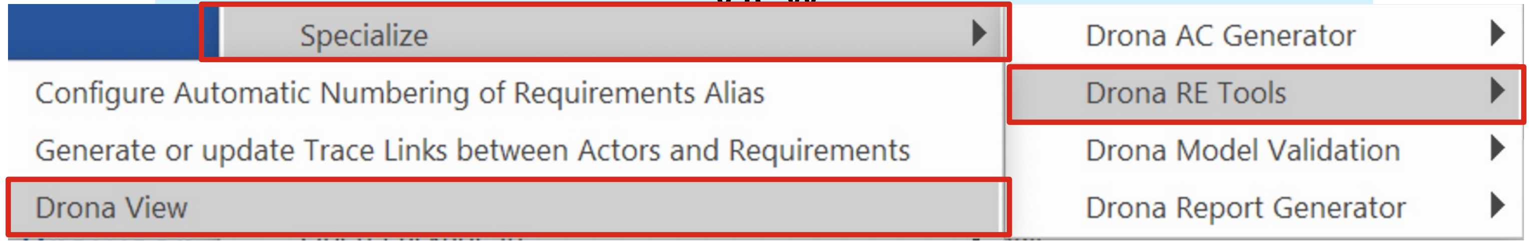
1 Right click on the package that contains the element(s)


Project Browser

Global Context

MT103 Cash Deadline for 9x Instructi
History
Main Contacts
Cyrille Schloegel

2 Go to Specialize > Qualisist RE Tools > Qualisist View

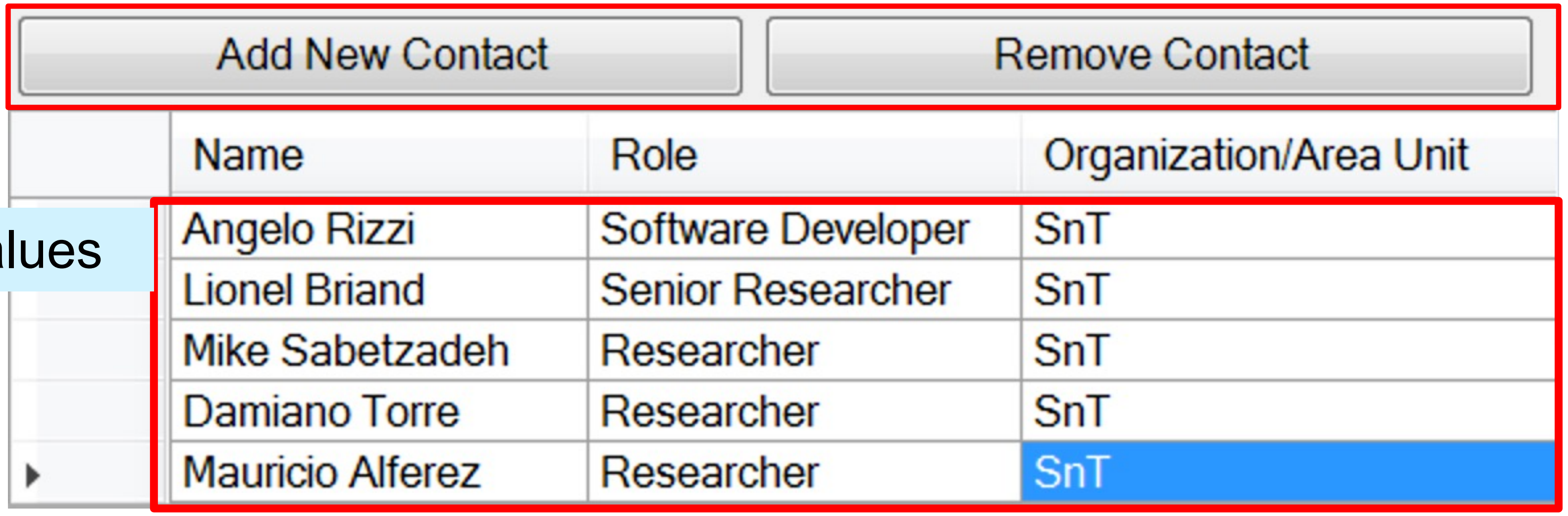

Specialize
Configure Automatic Numbering of Requirements Alias
Generate or update Trace Links between Actors and Requirements
Drona View
Drona AC Generator
Drona RE Tools
Drona Model Validation
Drona Report Generator

UNIVERSITÉ DU LUXEMBOURG

Qualisist View Example

Add/Remove buttons to create/remove a contact

Editable values



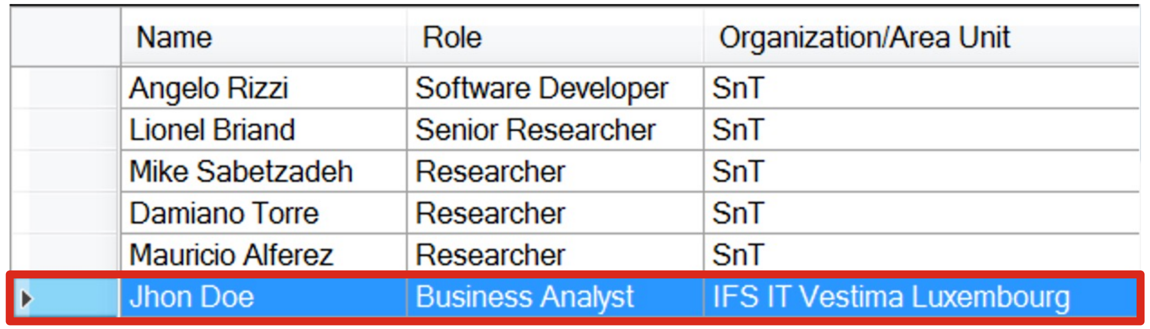
Name	Role	Organization/Area Unit
Angelo Rizzi	Software Developer	SnT
Lionel Briand	Senior Researcher	SnT
Mike Sabetzadeh	Researcher	SnT
Damiano Torre	Researcher	SnT
Mauricio Alferez	Researcher	SnT

Practice 6: Contacts

- **Goal:** Learn to create/edit records using the Qualisist View.
- **Task:**
  1. Create a new contact in the Main Contacts section

Name: Jhon Doe
Role: Business Analysts
Organization Area/Unit: IFS IT Vestima Luxembourg

- **Expected Result**



	Name	Role	Organization/Area Unit
	Angelo Rizzi	Software Developer	SnT
	Lionel Briand	Senior Researcher	SnT
	Mike Sabetzadeh	Researcher	SnT
	Damiano Torre	Researcher	SnT
	Mauricio Alferez	Researcher	SnT
▶	Jhon Doe	Business Analyst	IFS IT Vestima Luxembourg

List of Qualisist Elements

- History entries
- Contact entries
- In Scope/Out of Scope entries
- Key Principles/Key Decision, Assumptions, Constraints and dependencies entries
- Data mapping/data dictionary entries
- Functional requirement entries
- Transition requirement entries
- Non-functional requirement entries

Remember the Predefined Data Types in Qualisist

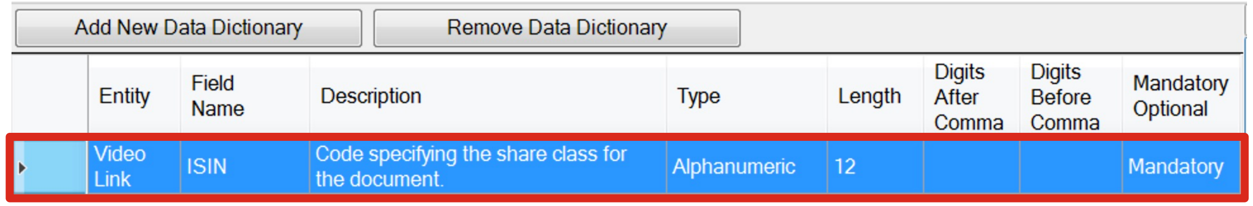
- According to Clearstream, Qualisist should support four types of data attributes:
  - **Boolean**: Contain the value either true or false
  - **Date**: Contain a timestamp
  - **Alphanumeric**: Contain either numbers and/or alphabetical characters
  - **Numeric**: Contain only numbers (either integers or decimals)

Practice 7: Data Dictionary

- **Goal:** Learn to create/edit new records in the Data Dictionary section using the Qualisist View.
- **Task:** Create a new data entry [illegible]

Entity: VideoLink
Field name: ISIN
Length: 12
Mandatory/Optional: Mandatory
Type: Alphanumeric
Description: Code specifying the share class for the document.

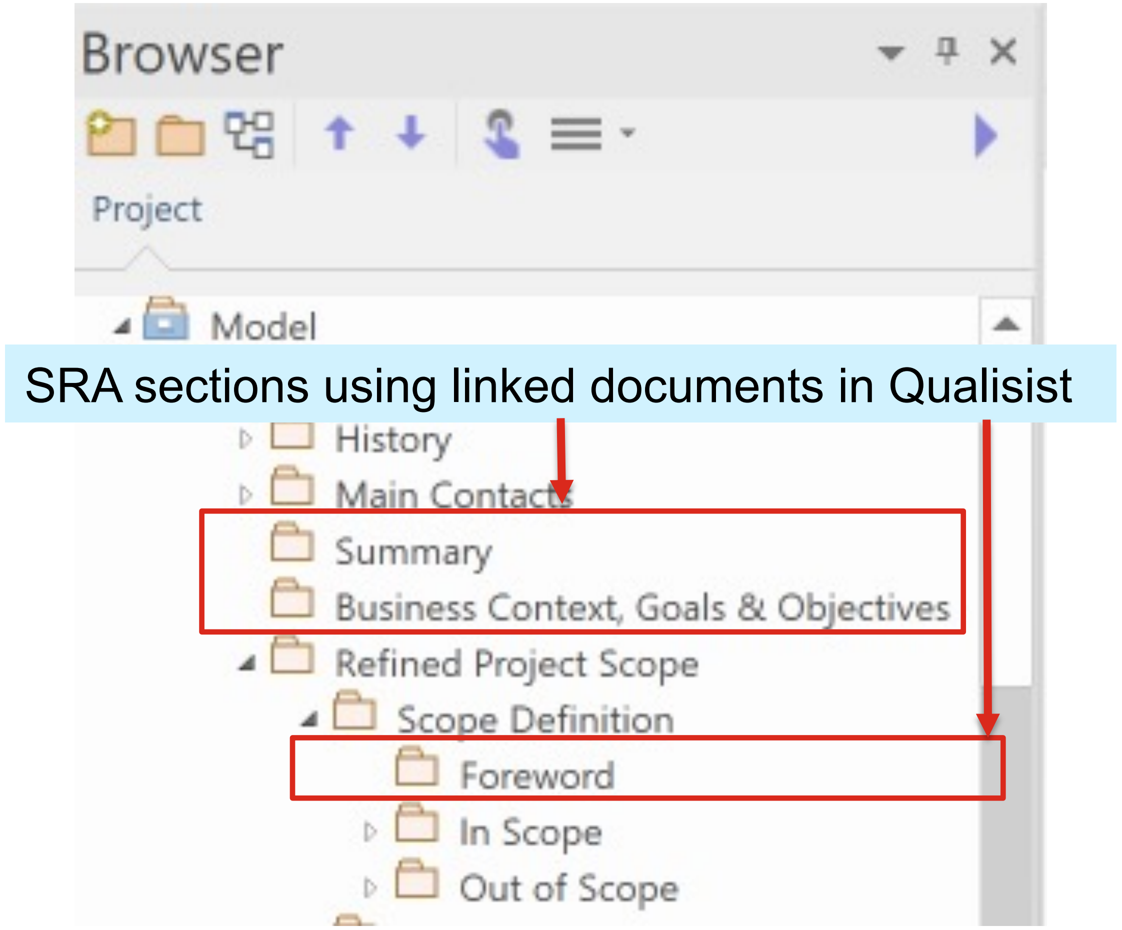
- **Expected Result**



	Entity	Field Name	Description	Type	Length	Digits After Comma	Digits Before Comma	Mandatory Optional
▸	Video Link	ISIN	Code specifying the share class for the document.	Alphanumeric	12			Mandatory

Linked Documents

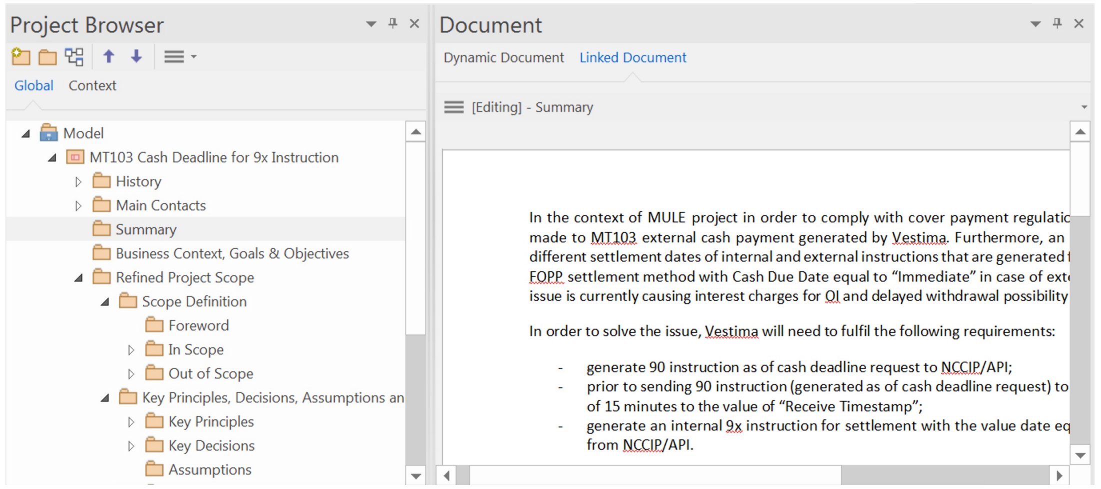
- Linked Documents are formatted documents associated with Elements and used to write structured text or extensive documentation



Practice 8: Link a Document to a Package

- **Goal:** Learn how to provide extensive documentation on an element using the Linked Documents tool.
- **Task:**
  1. Link a document to the package representing the section named “Summary”
  2. You don’t need to type the section title “Summary”

- **Expected Result**



In the context of MULE project in order to comply with cover payment regulatic
made to MT103 external cash payment generated by Vestima. Furthermore, an
different settlement dates of internal and external instructions that are generated
FOPP settlement method with Cash Due Date equal to “Immediate” in case of ext
issue is currently causing interest charges for OI and delayed withdrawal possibility

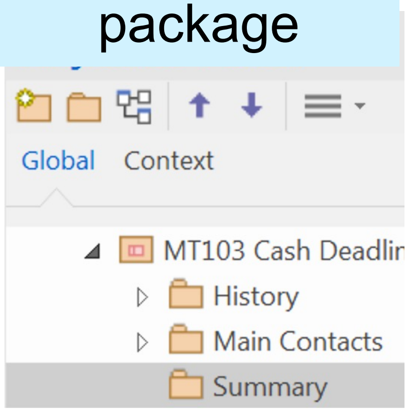
In order to solve the issue, Vestima will need to fulfil the following requirements:

- generate 90 instruction as of cash deadline request to NCCIP/API;
- prior to sending 90 instruction (generated as of cash deadline request) to of 15 minutes to the value of “Receive Timestamp”;
- generate an internal 9x instruction for settlement with the value date eq from NCCIP/API.

Steps to Import and Link a Document

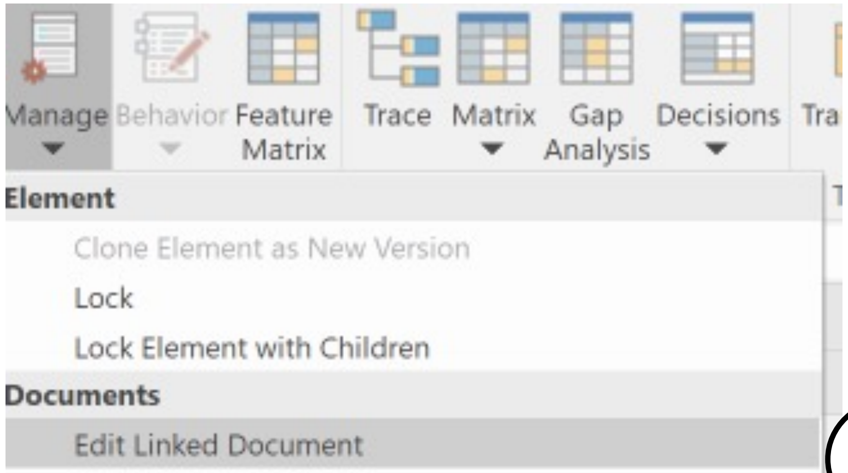
①

Select a package



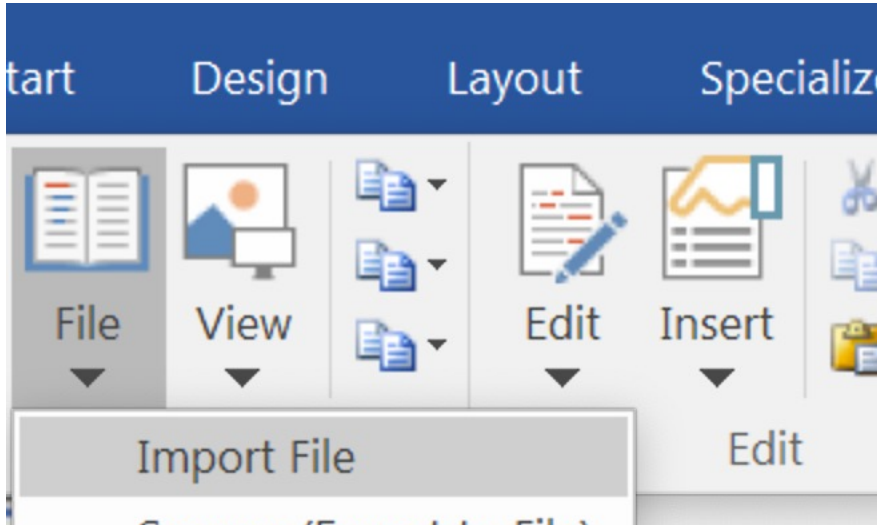
②

Select Design → Manage → Edit Linked Document



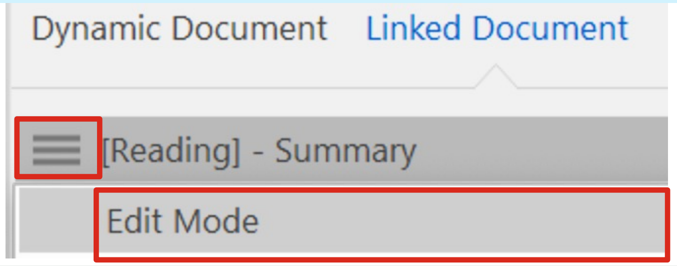
③

Go to Edit→ File → Import File



④

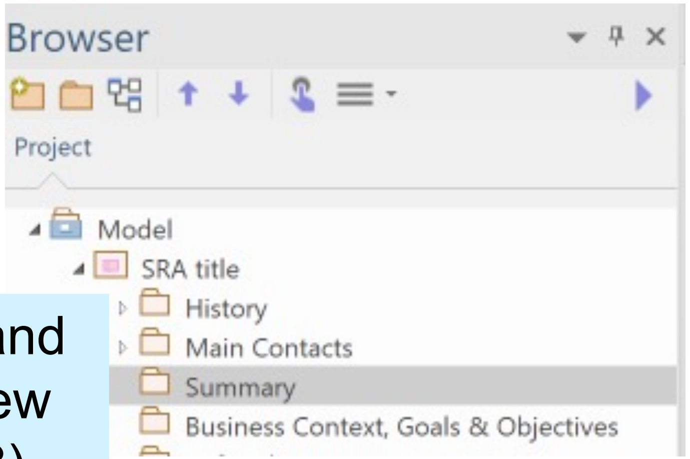
(Optional) Select **Edit Mode** if the *Import File* menu is disabled



Steps to Link Documents (1/2)

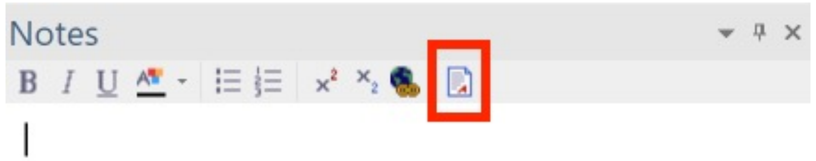
①

Select a package and open the Notes view (Shortcut **Ctrl + 3**)



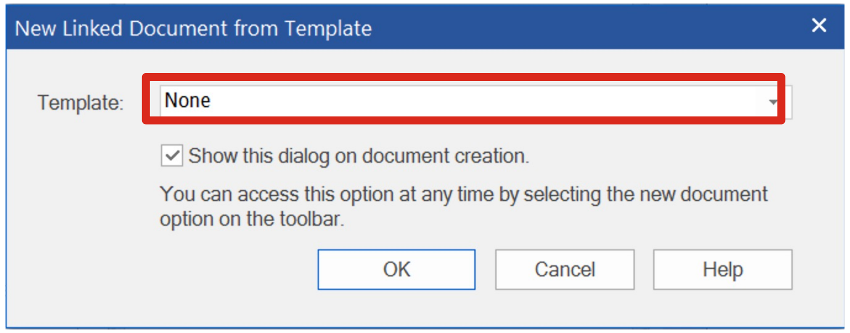
②

Click on the Link Document Icon



③

Select **None** and hit **OK**



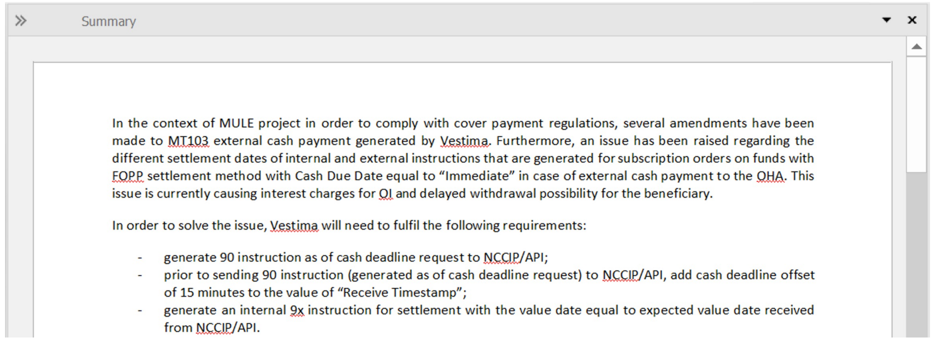
Template: None

✓ Show this dialog on document creation.

You can access this option at any time by selecting the new document option on the toolbar.

④

Add text, images, tables, etc., to the document)



In the context of MULE project in order to comply with cover payment regulations, several amendments have been made to MT103 external cash payment generated by Vestima. Furthermore, an issue has been raised regarding the different settlement dates of internal and external instructions that are generated for subscription orders on funds with FOPP settlement method with Cash Due Date equal to "Immediate" in case of external cash payment to the OHA. This issue is currently causing interest charges for OI and delayed withdrawal possibility for the beneficiary.

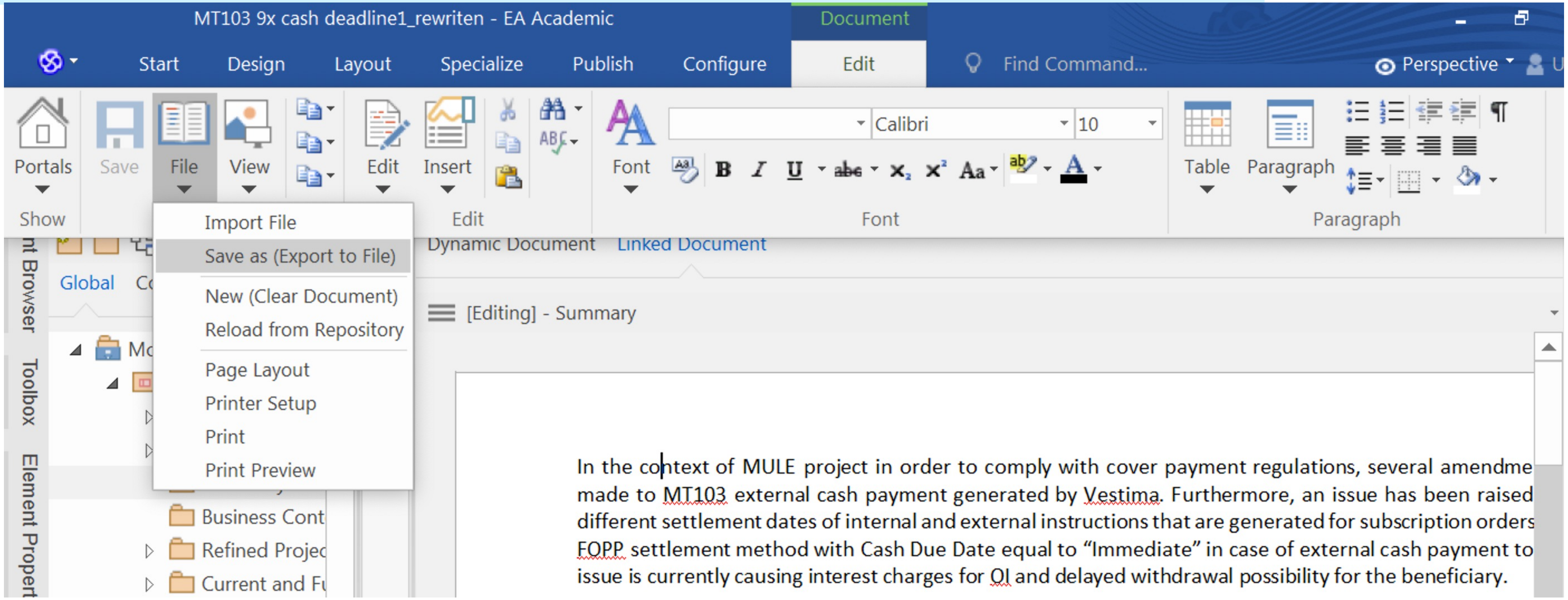
In order to solve the issue, Vestima will need to fulfil the following requirements:

- generate 90 instruction as of cash deadline request to NCCIP/API;
- prior to sending 90 instruction (generated as of cash deadline request) to NCCIP/API, add cash deadline offset of 15 minutes to the value of "Receive Timestamp";
- generate an internal 9x instruction for settlement with the value date equal to expected value date received from NCCIP/API.

Steps to Link Documents (2/2)

securityandtrust.lu

⑤ Go to Edit → File → Save as (Export to File) and follow the instructions



In the context of MULE project in order to comply with cover payment regulations, several amendme
made to MT103 external cash payment generated by Vestima. Furthermore, an issue has been raised
different settlement dates of internal and external instructions that are generated for subscription orders
FOPP settlement method with Cash Due Date equal to "Immediate" in case of external cash payment to
issue is currently causing interest charges for OI and delayed withdrawal possibility for the beneficiary.

UNIVERSITÉ DU LUXEMBOURG

Practice 9: Additional Information

- **Goal:** Learn to use standard EA tools to create glossary and abbreviations tables.
- **Task 1:**

  Create the following glossary entry

  Name: IU
  Definition: Internal User – Clearstream Banking administrator with permissions to act, under strict guidelines, on behalf of the OI or the OHA.

- **Task 2:**

  Create the following abbreviation entry

  Name: CBL
  Definition: Clearstream Banking Luxembourg

Practice 10: Generate a Deliverable Report

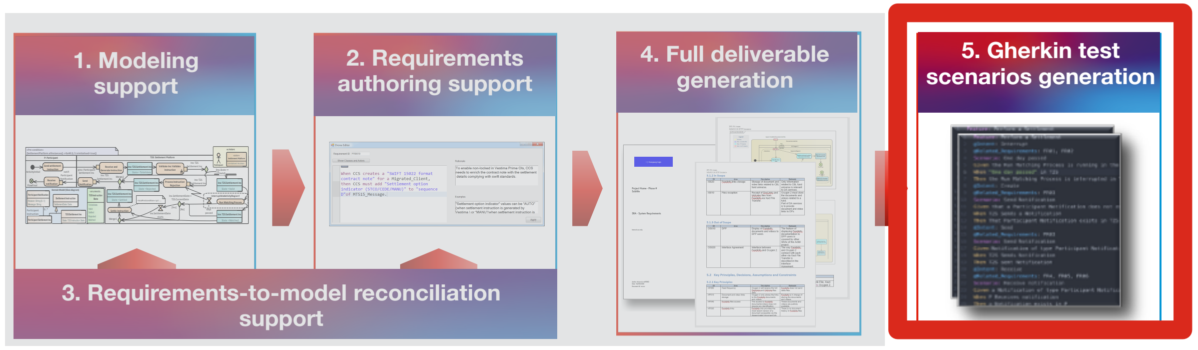
- **Goal:** Learn to configure and use the Qualisist report generator.

- **Tasks:**
  1. Fill out the Qualisist report generator form for the “**MT103 9x Cash Deadline**” project based on the SRA provided by Elene
  2. Generate the deliverable report

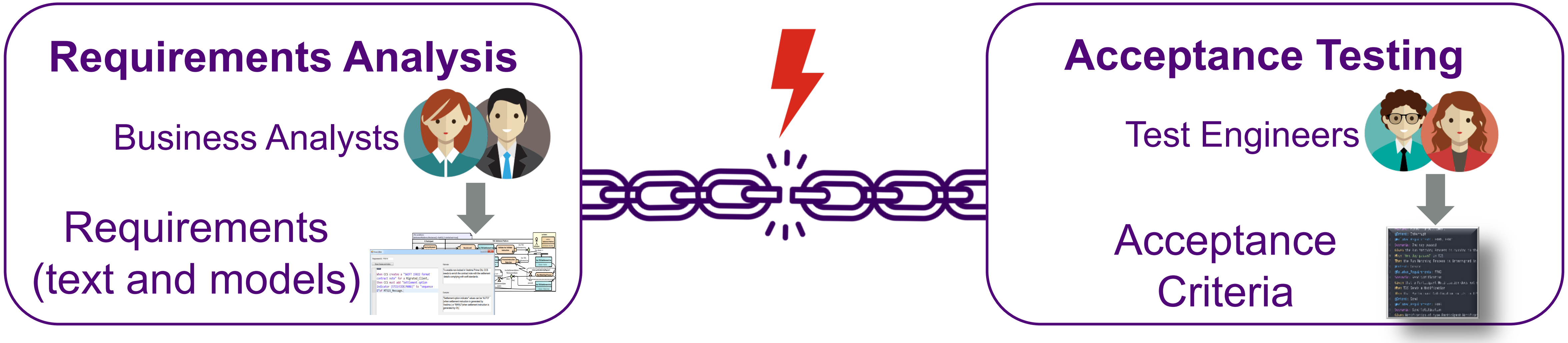
Agenda

0. Installation and Configuration
1. Modelling Support
2. Requirements authoring support
3. Requirements-to-Model reconciliation support
4. Full deliverable generation
5. **Gherkin test Scenarios generation**

The Qualisist Solution


1. Modeling support
2. Requirements authoring support
4. Full deliverable generation
5. Gherkin test scenarios generation
3. Requirements-to-model reconciliation support

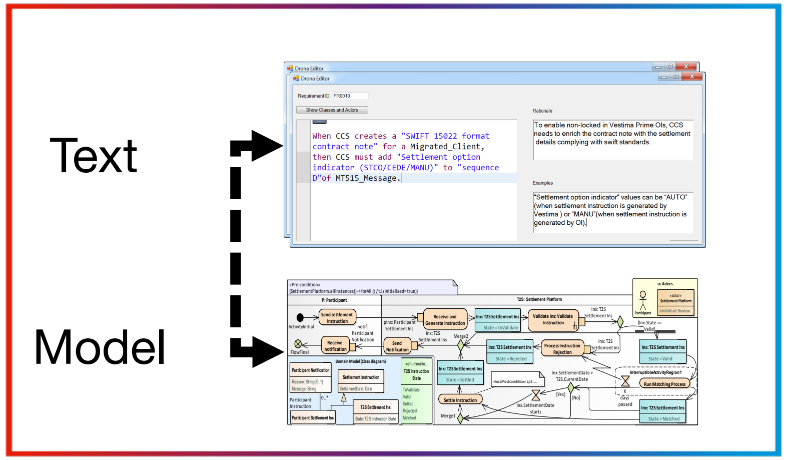
Disconnect between Requirements and Testing



Acceptance Criteria (AC) Generation

- Automated generation of Acceptance Criteria in the Gherkin language
- Generation of Acceptance Criteria based on the text and models
- Negligible execution time of Acceptance Criteria generation

Requirements

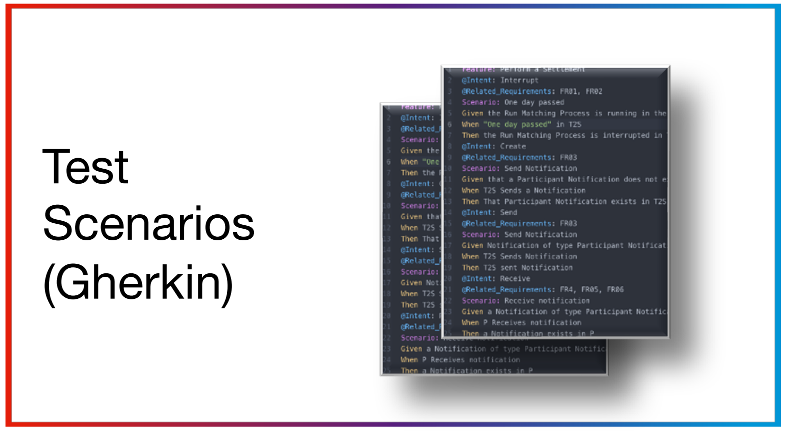


```
When CCS creates a "SWIFT 15022 format
contract note" for a Migrated_Client,
then CCS must add "Settlement option
indicator (STCO/CEDE/MANU)" to "sequence
D"of MT515_Message.
```

Rationale: To enable non-locked in Vestima Prime OIs, CCS needs to enrich the contract note with the settlement details complying with swift standards.

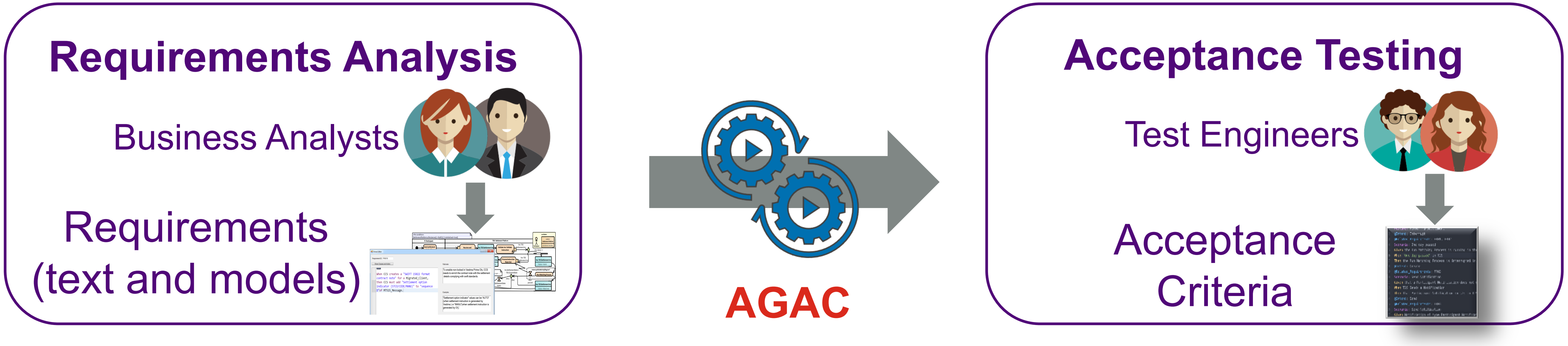
Examples: "Settlement option indicator" values can be "AUTO" (when settlement instruction is generated by Vestima) or "MANU"(when settlement instruction is generated by OI).

Acceptance Criteria



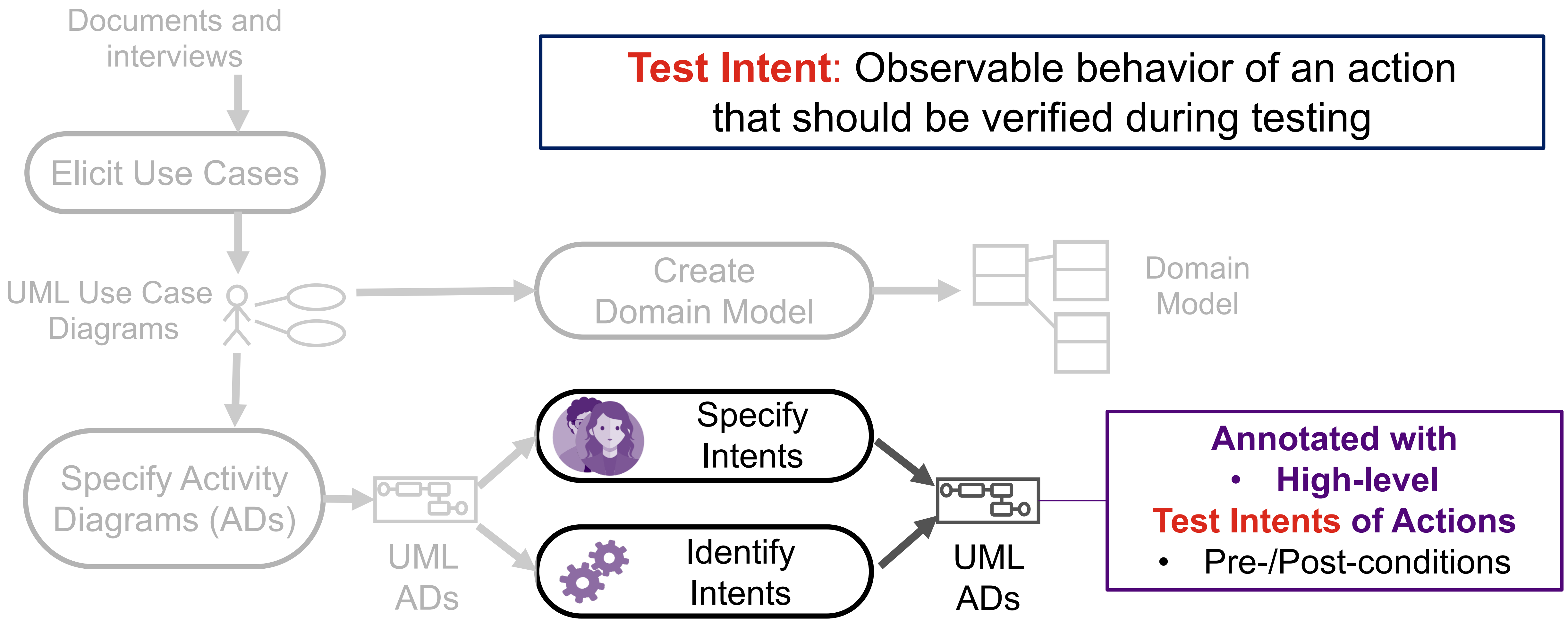
```
Feature: Perform a Settlement
@Intent: Interrupt
@Related_Requirements: FR01, FR02
Scenario: One day passed
Given the Run Matching Process is running in the
When "One day passed" in T2S
Then the Run Matching Process is interrupted in
@Intent: Create
@Related_Requirements: FR03
Scenario: Send Notification
Given that a Participant Notification does not e
When T2S Sends a Notification
Then That Participant Notification exists in T2S
@Intent: Send
@Related_Requirements: FR03
Scenario: Send Notification
Given Notification of type Participant Notificat
When T2S Sends Notification
Then T2S sent Notification
@Intent: Receive
@Related_Requirements: FR4, FR05, FR06
Scenario: Receive notification
Given a Notification of type Participant Notific
When P Receives notification
Then a Notification exists in P
```

AGAC: Automatic Generation of Acceptance Criteria



Add Test Intents

Test Intent: Observable behavior of an action that should be verified during testing

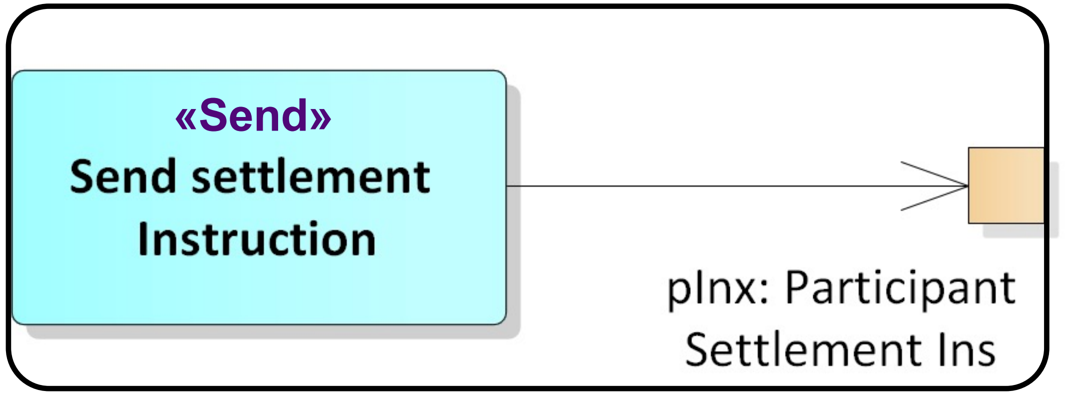


Annotated with
- **High-level Test Intents of Actions**
- Pre-/Post-conditions

11 Intent Types

Based on manual investigation of 841 AC from Clearstream

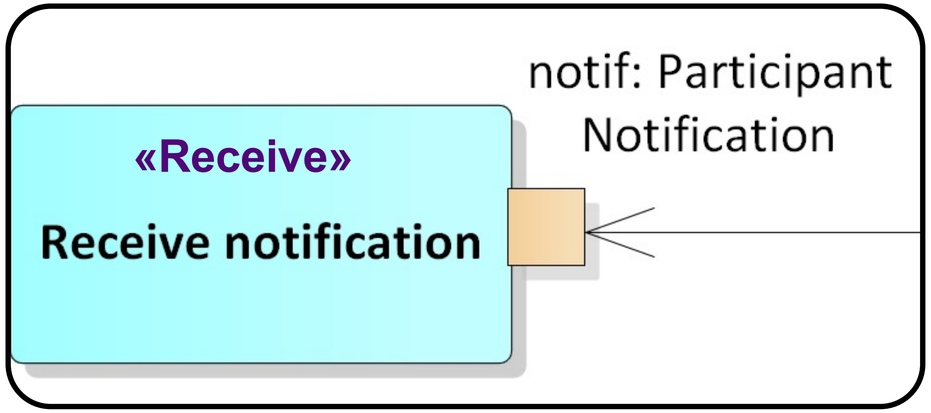
- «Create», «Read», «Update», «Send»
  - concern the object associated to the outgoing edge



11 Intent Types

Based on manual investigation of 841 AC from Clearstream

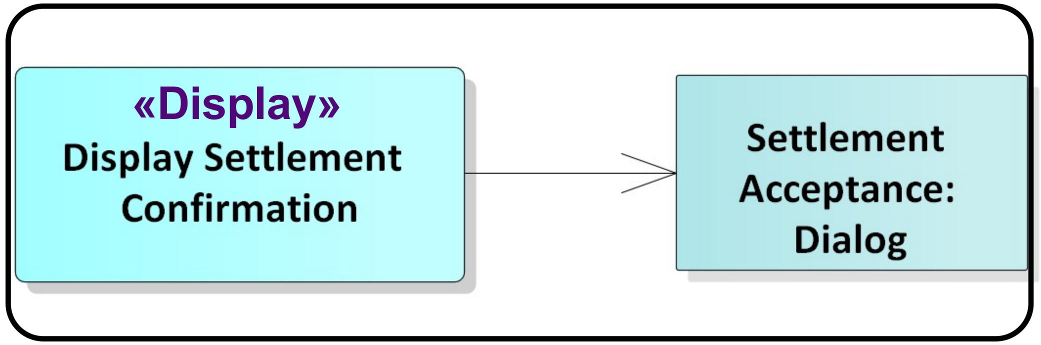
- «Create», «Read», «Update», «Send»
  - concern the object associated to the outgoing edge
- «Delete», «Receive», «Validate»
  - concern the object associated to the incoming edge



11 Intent Types

Based on manual investigation of 841 AC from Clearstream

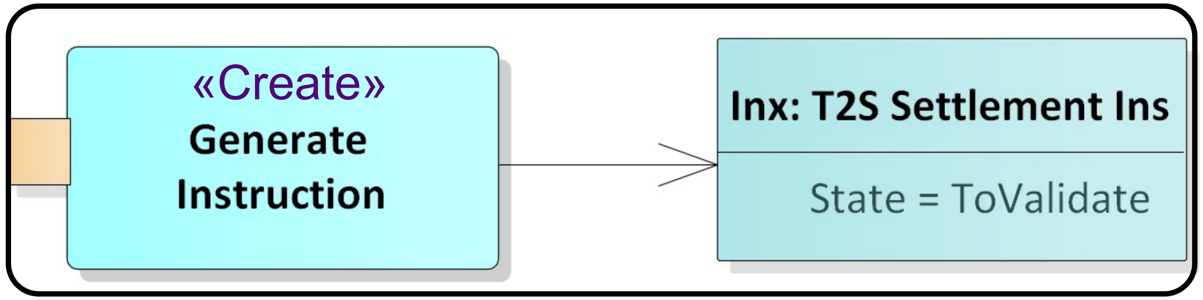
- «Create», «Read», «Update», «Send»
  - Concern the object associated to the outgoing edge
- «Delete», «Receive», «Validate»
  - Concern the object associated to the incoming edge
- «Display», «Not-Display», «Enable», «Disable»
  - Concern the user interface


«Display»
Display Settlement Confirmation
Settlement Acceptance: Dialog

Automated Identification of Intents

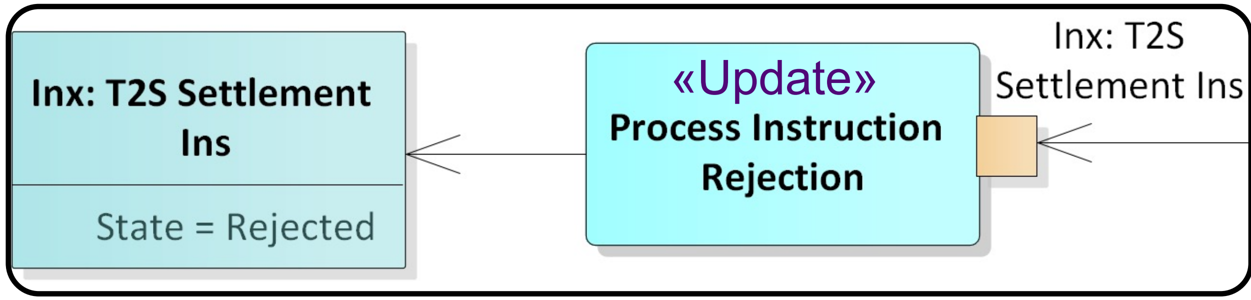
«Create»

The object on the output edge has an ID never observed before



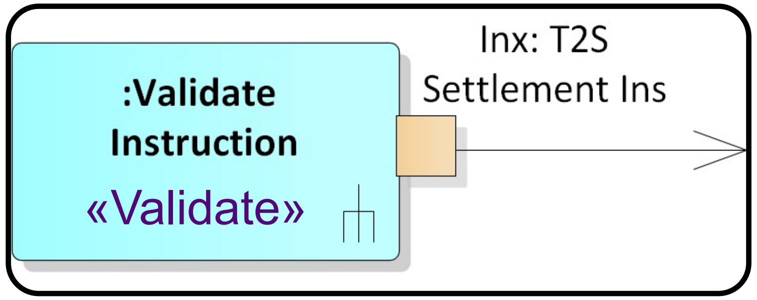
«Update»

Input and output connected to objects with the same ID

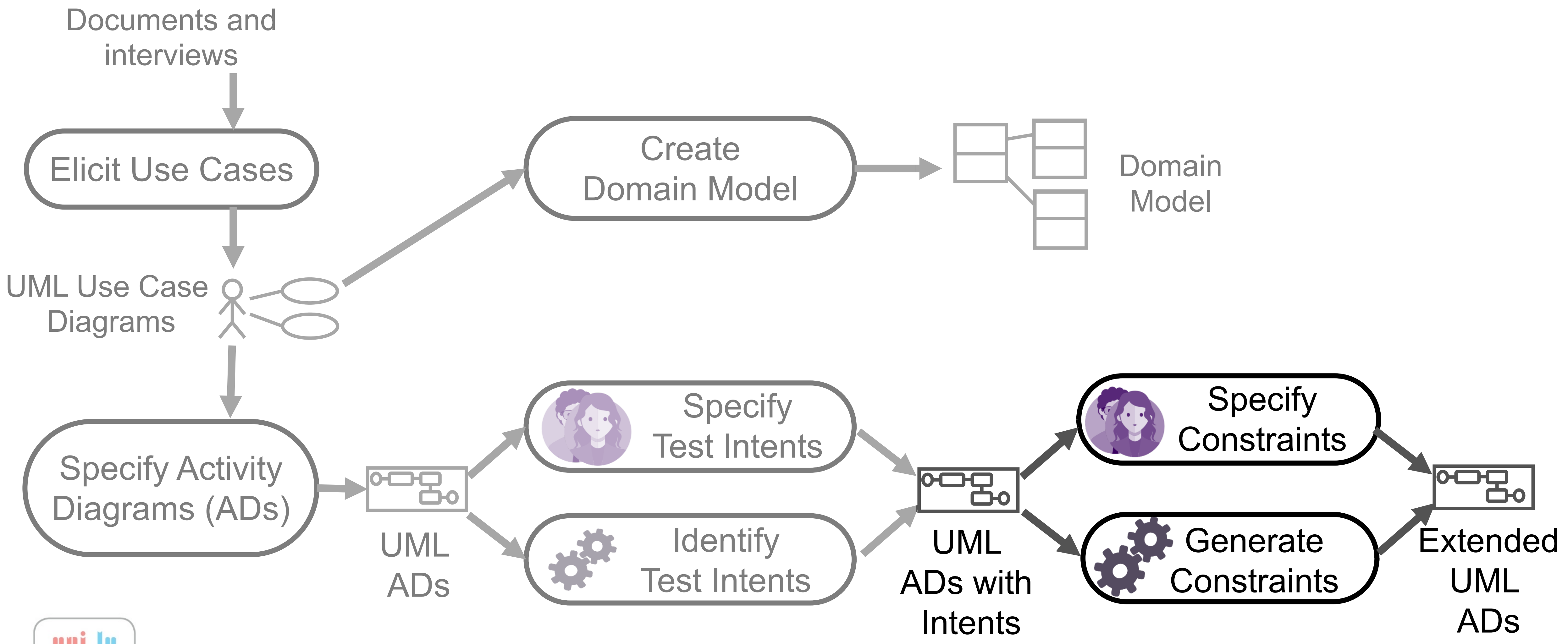


«*ANY*» (e.g., «Validate»)

The verb (or a synonym) in the action name matches the intent name

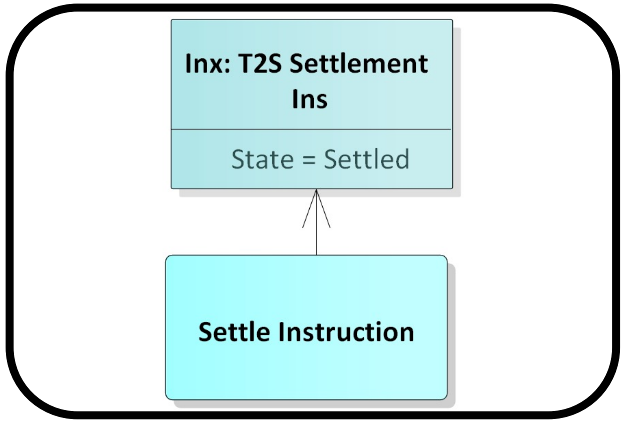


Qualisist Modeling Methodology



Automated Generation of Constraints based on Object–flow Analysis

The action produces an object

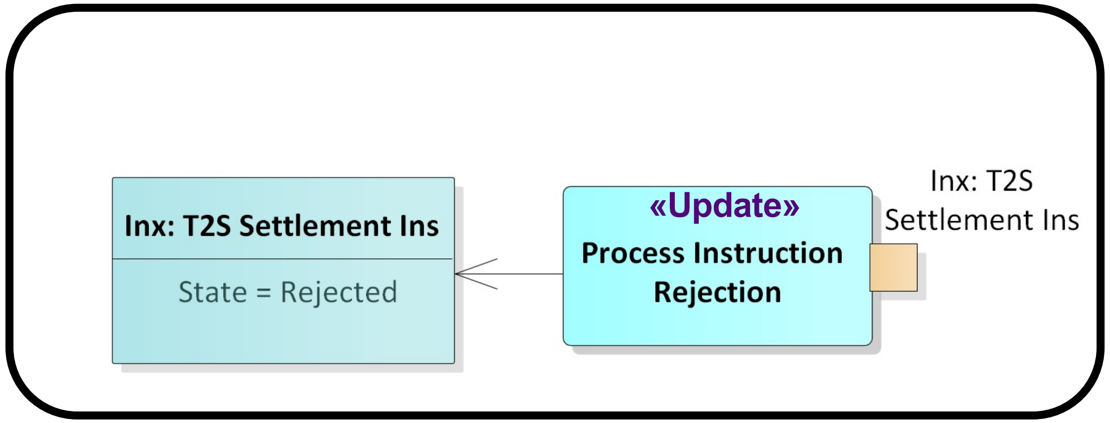


Postcondition: attribute values should match the model

the property "State" of Inx is equal to "Settled"

Automated Generation of Constraints based on Intents Analysis

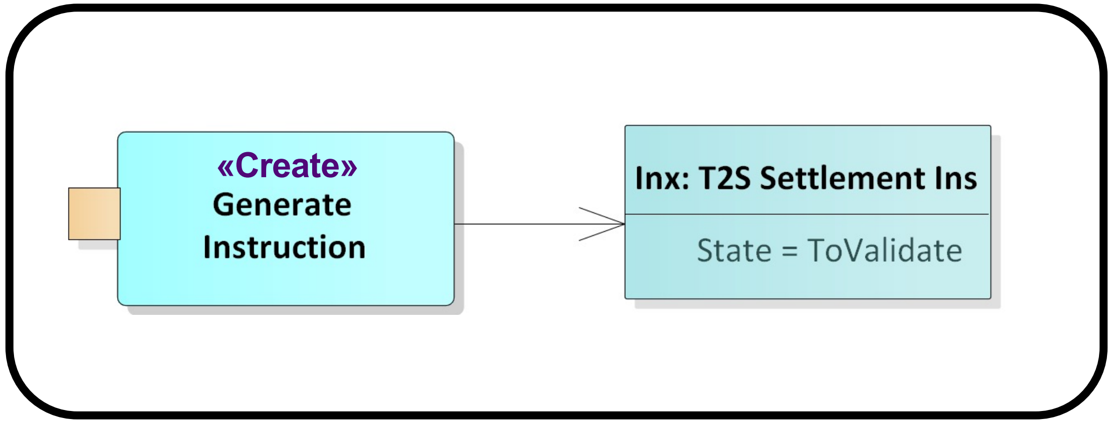
«Read»
«Update»
«Delete»
«Send»



Precondition:
the entity exists

Inx exists in T2S

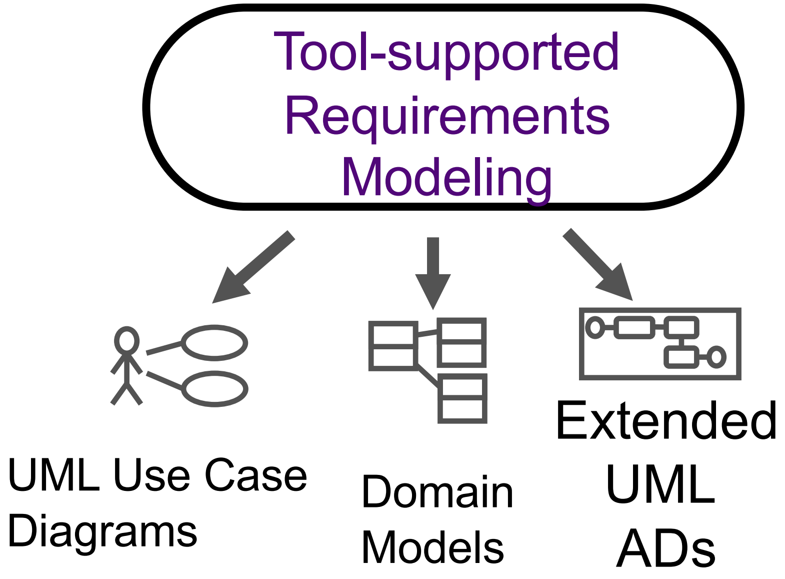
«Create»
«Receive»



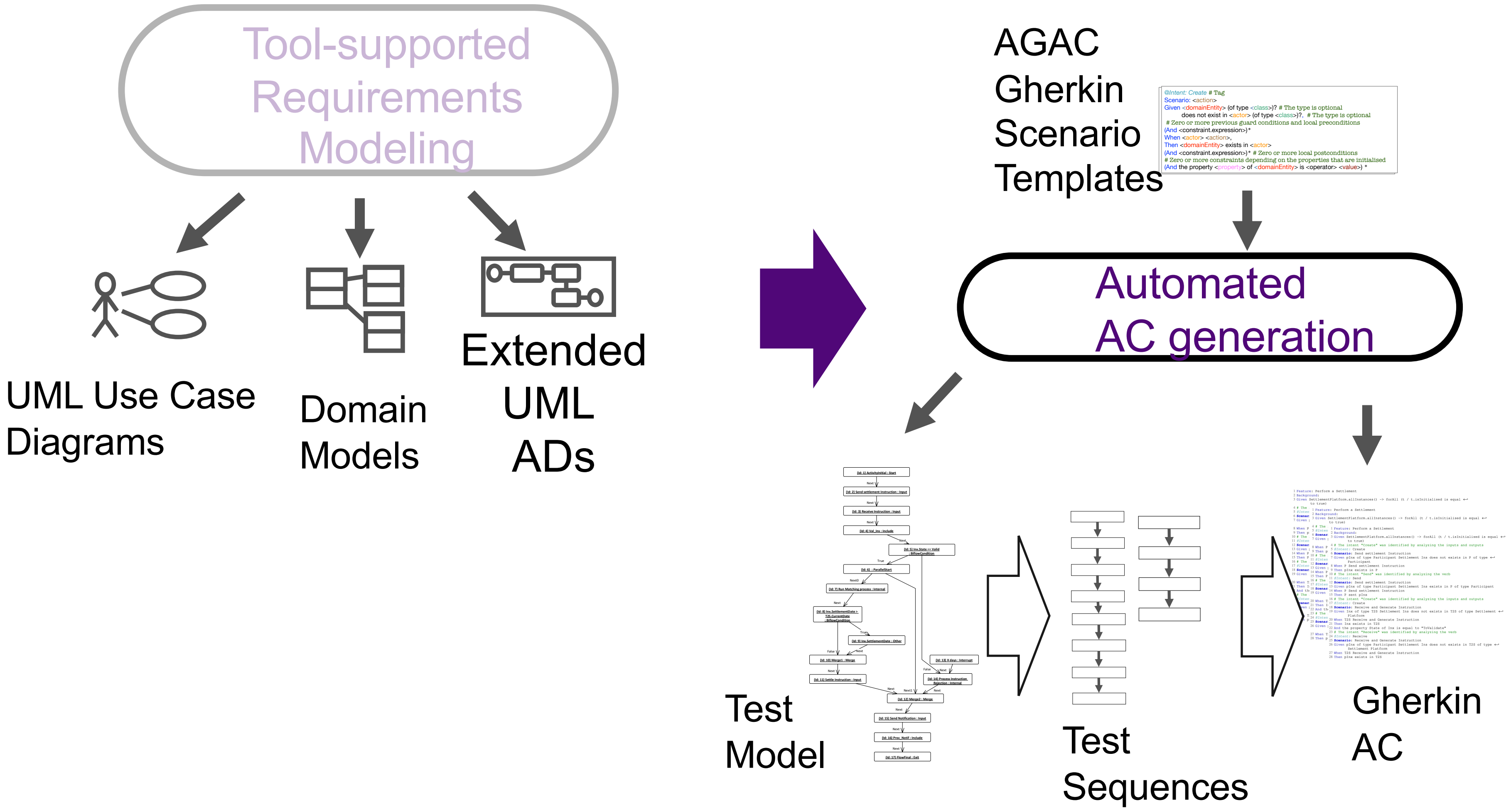
Precondition:
the entity does not exist

Inx does not exist in T2S

Automated AC generation



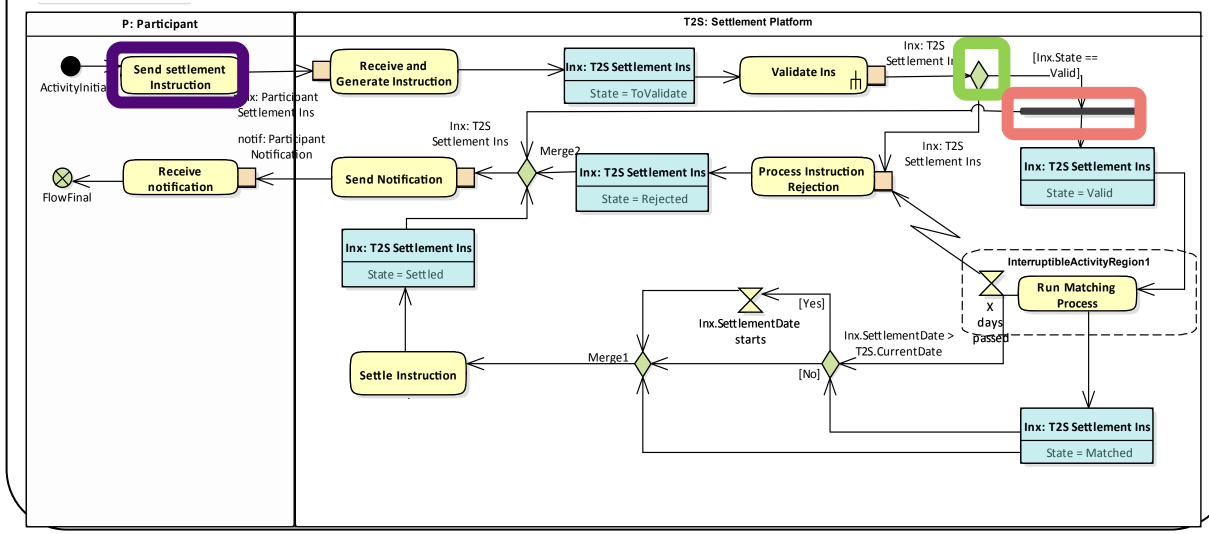
Automated AC generation



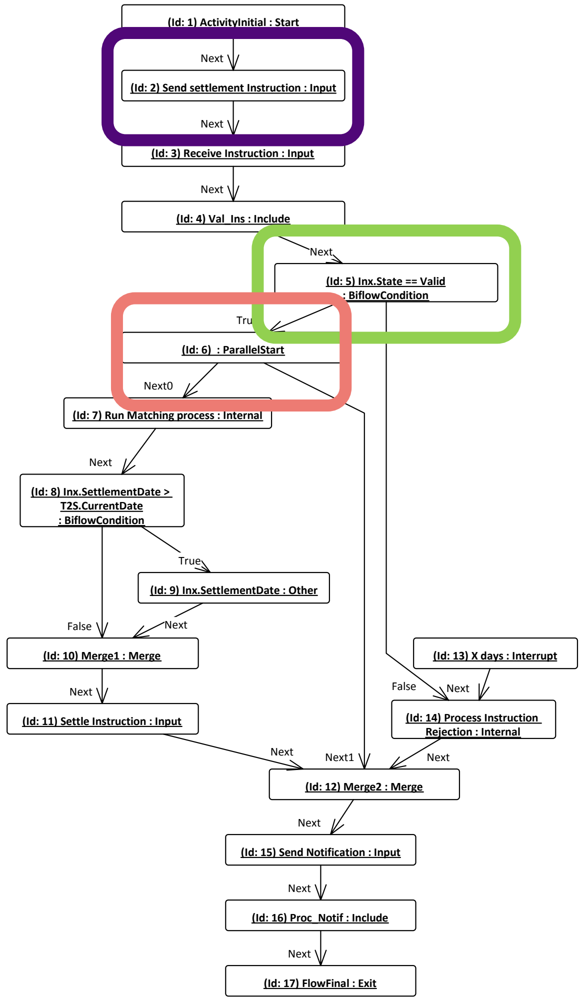
```
@Intent: Create # Tag
Scenario: <action>
Given <domainEntity> (of type <class>)? # The type is optional
        does not exist in <actor> (of type <class>)?,  # The type is optional
 # Zero or more previous guard conditions and local preconditions
(And <constraint.expression>)*
When <actor> <action>,
Then <domainEntity> exists in <actor>
(And <constraint.expression>)* # Zero or more local postconditions
# Zero or more constraints depending on the properties that are initialised
(And the property <property> of <domainEntity> is <operator> <value>) *
```

Test Model Generation

Activity Diagram



Test Model

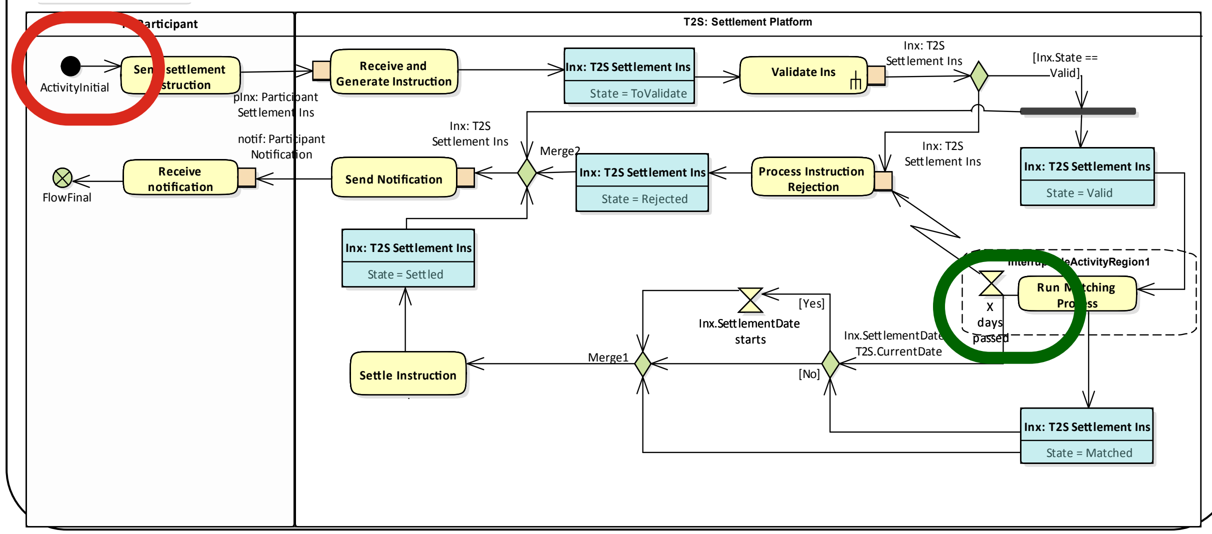


(Id: 1) ActivityInitial : Start
Next
(Id: 2) Send settlement Instruction : Input
Next
(Id: 3) Receive Instruction : Input
Next
(Id: 4) Val_Ins : Include
Next
(Id: 5) Inx.State == Valid : BiflowCondition
True
(Id: 6) : ParallelStart
Next0
(Id: 7) Run Matching process : Internal
Next
(Id: 8) Inx.SettlementDate > T2S.CurrentDate : BiflowCondition
True
(Id: 9) Inx.SettlementDate : Other
False
Next
(Id: 10) Merge1 : Merge
Next
(Id: 11) Settle Instruction : Input
(Id: 13) X days : Interrupt
False
Next
(Id: 14) Process Instruction Rejection : Internal
Next
Next1
Next
(Id: 12) Merge2 : Merge
Next
(Id: 15) Send Notification : Input
Next
(Id: 16) Proc_Notif : Include
Next
(Id: 17) FlowFinal : Exit

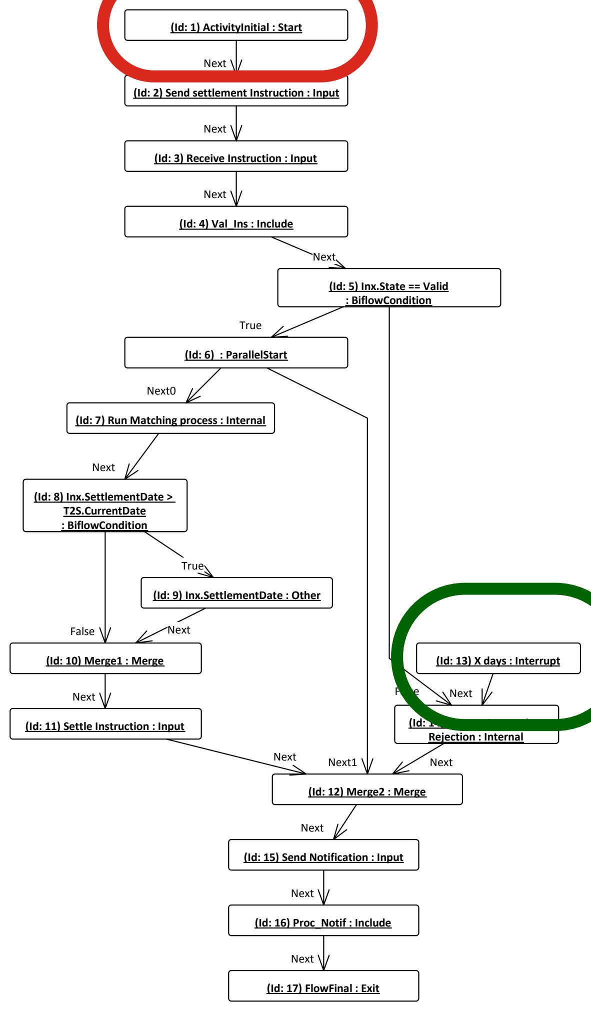
- Metamodel in the paper
- One node for each element in the Activity Diagram
- Multiple roots:
  - Initial activity nodes
  - Events (e.g., events in interruptible activity regions)

Test Model Generation

Activity Diagram

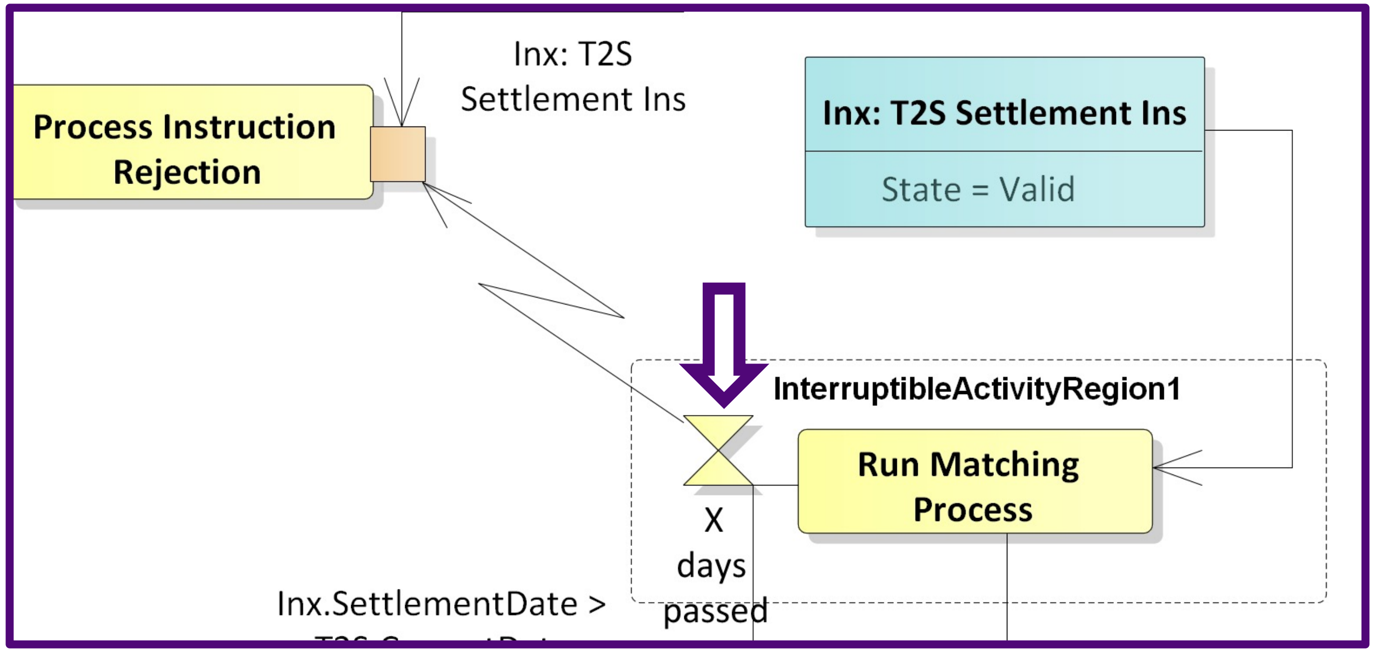


Test Model


(Id: 1) ActivityInitial : Start
Next
(Id: 2) Send settlement Instruction : Input
Next
(Id: 3) Receive Instruction : Input
Next
(Id: 4) Val_Ins : Include
Next
(Id: 5) Inx.State == Valid : BiflowCondition
True
(Id: 6) : ParallelStart
Next0
(Id: 7) Run Matching process : Internal
Next
(Id: 8) Inx.SettlementDate > T2S.CurrentDate : BiflowCondition
True
(Id: 9) Inx.SettlementDate : Other
False
Next
(Id: 10) Merge1 : Merge
Next
(Id: 11) Settle Instruction : Input
(Id: 13) X days : Interrupt
Next
Rejection : Internal
Next
Next1
Next
(Id: 12) Merge2 : Merge
Next
(Id: 15) Send Notification : Input
Next
(Id: 16) Proc_Notif : Include
Next
(Id: 17) FlowFinal : Exit

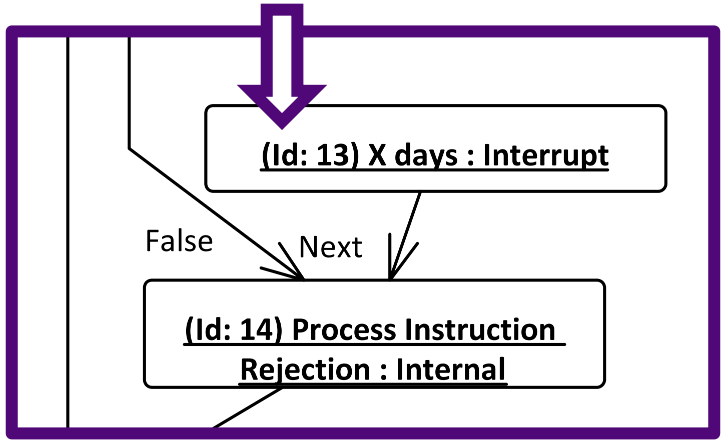
- Metamodel in the paper https://orbilu.uni.lu/handle/10993/39710
- One node for each element in the Activity Diagram
- Multiple roots:
  - Initial activity nodes
  - Events (e.g., events in interruptible activity regions)

Root derived from



A Fragment of an Activity Diagram

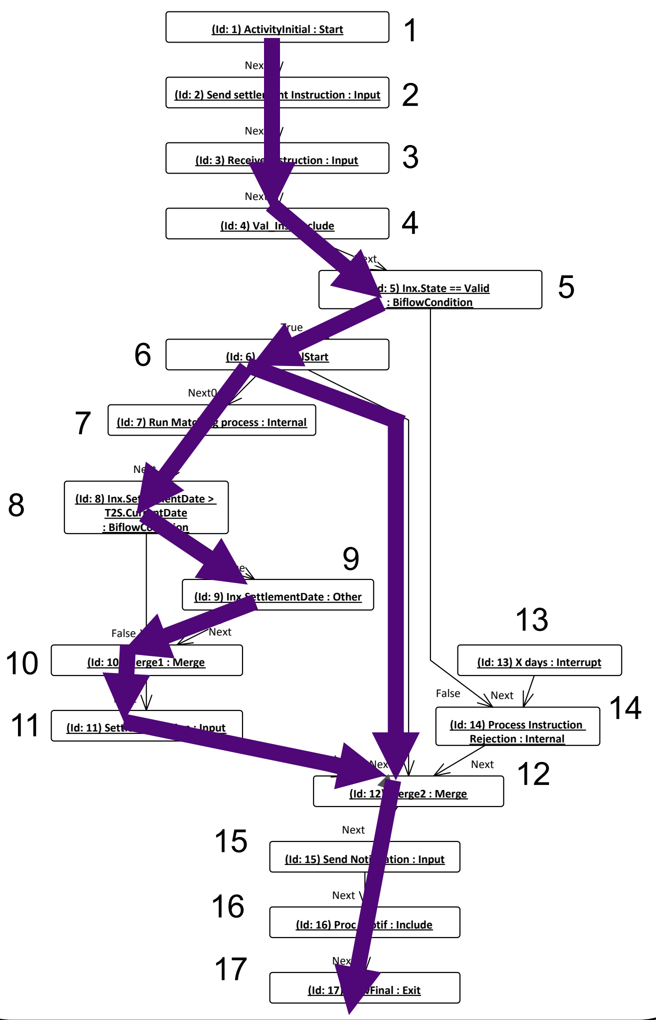
A Fragment of a Test Model


(Id: 13) X days : Interrupt
False
Next
(Id: 14) Process Instruction Rejection : Internal

Test Sequences Generation

- Test Sequence: sequence of nodes
- Each test sequence leads to an Acceptance Criterion

Test Model


(Id: 1) ActivityInitial : Start
1
Next
(Id: 2) Send settle nt Instruction : Input
2
Next
(Id: 3) Receive truction : Input
3
Next
(Id: 4) Val_In clude
4
Next
d: 5) Inx.State == Valid : BiflowCondition
5
True
(Id: 6) lStart
6
Next0
(Id: 7) Run Mat process : Internal
7
Next
(Id: 8) Inx.Se entDate > T2S.Cu ntDate : BiflowCo ion
8
(Id: 9) Inx SettlementDate : Other
9
False
Next
(Id: 10 erge1 : Merge
10
(Id: 11) Sett : Input
11
(Id: 13) X days : Interrupt
13
False
Next
(Id: 14) Process Instruction Rejection : Internal
14
Next
(Id: 12) erge2 : Merge
12
Next
(Id: 15) Send Noti ation : Input
15
Next
(Id: 16) Proc otif : Include
16
Next
(Id: 17) Final : Exit
17

Generated Test Sequences

P_1 = **[1,2,3,4,5,6,12,7,15,8,16,9,17,10,11,12,15,16,17]**

P_2 = [1,2,3,4,5,6,12,7,15,8,16,10,17,11,12,15,16,17]

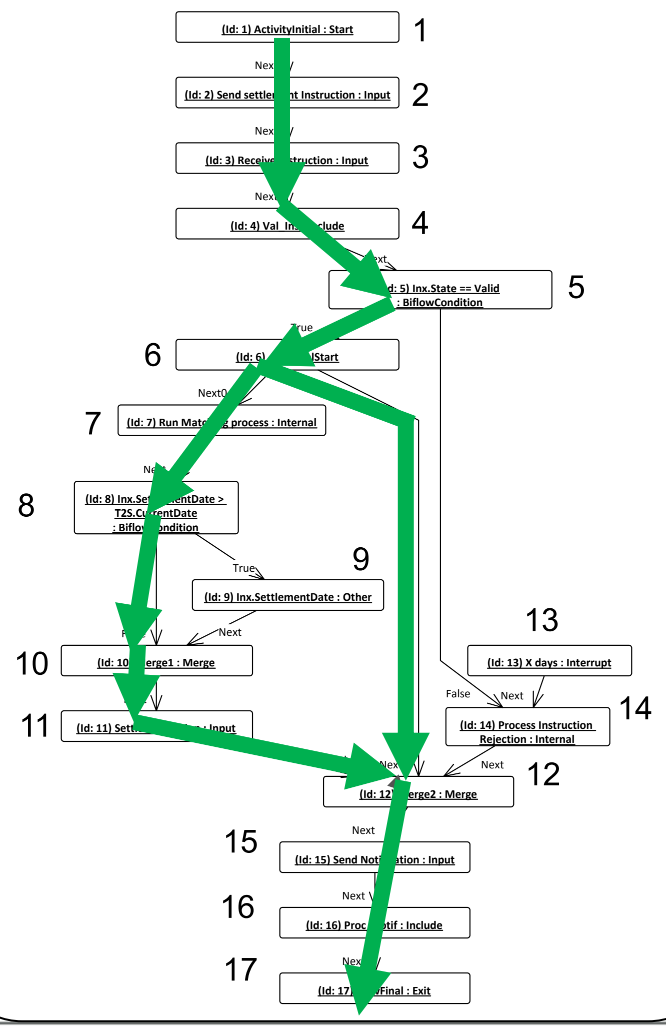
P_3= [1,2,3,4,5,14,12,15,16,17]

P_4= [13,14,15,16,17]

Test Sequences Generation

- Test Sequence: sequence of nodes
- Each test sequence leads to an Acceptance Criterion

Test Model



(Id: 1) ActivityInitial : Start
1
Nex
(Id: 2) Send settle nt Instruction : Input
2
Nex
(Id: 3) Receive struction : Input
3
Next
(Id: 4) Val_In clude
4
ext
d: 5) Inx.State == Valid : BiflowCondition
5
True
6
(Id: 6) lStart
Next0
7
(Id: 7) Run Mat process : Internal
Ne
8
(Id: 8) Inx.Se entDate > T2S.Cu rentDate : Biflo ondition
True
9
(Id: 9) Inx.SettlementDate : Other
Next
13
10
(Id: 10 erge1 : Merge
(Id: 13) X days : Interrupt
False
Next
14
11
(Id: 11) Sett : Input
(Id: 14) Process Instruction Rejection : Internal
Nex
Next
12
(Id: 12) erge2 : Merge
Next
15
(Id: 15) Send Noti ation : Input
Next
16
(Id: 16) Proc otif : Include
Nex
17
(Id: 17) Final : Exit

Generated Test Sequences

P_1 = [1,2,3,4,5,6,12,7,15,8,16,9,17,10,11,12,15,16,17]

P_2 = [**1,2,3,4,5,6,12,7,15,8,16,10,17,11,12,15,16,17**]

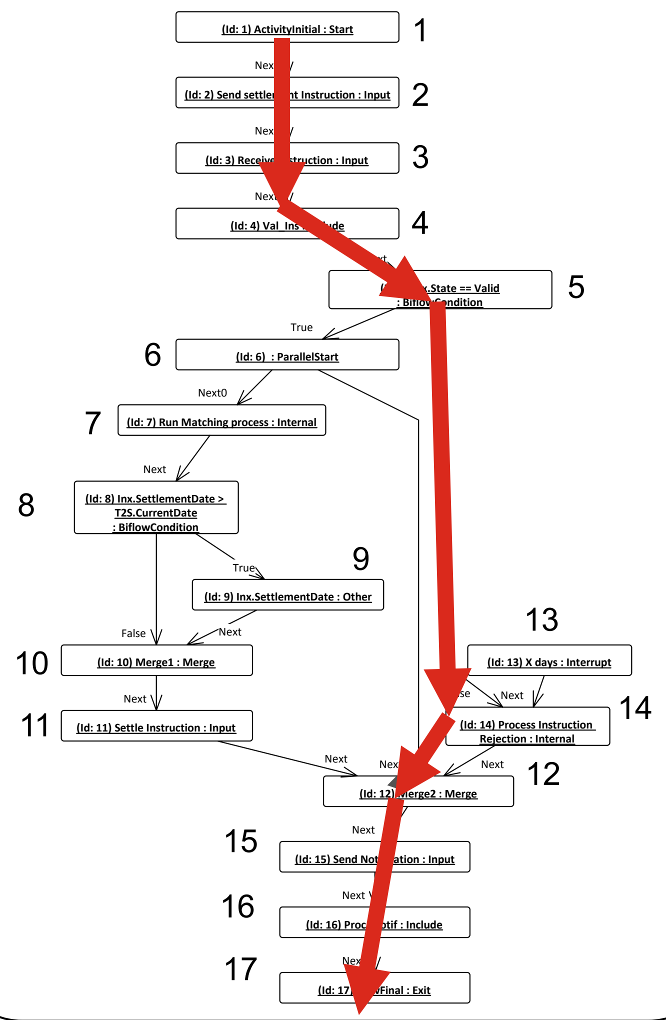
P_3= [1,2,3,4,5,14,12,15,16,17]

P_4= [13,14,15,16,17]

Test Sequences Generation

- Test Sequence: sequence of nodes
- Each test sequence leads to an Acceptance Criterion

Test Model


(Id: 1) ActivityInitial : Start
1
Next
(Id: 2) Send settle nt Instruction : Input
2
Next
(Id: 3) Receive truction : Input
3
Next
(Id: 4) Val_Ins ude
4
.State == Valid : Biflow Condition
5
True
(Id: 6) : ParallelStart
6
Next0
(Id: 7) Run Matching process : Internal
7
Next
(Id: 8) Inx.SettlementDate > T2S.CurrentDate : BiflowCondition
8
True
(Id: 9) Inx.SettlementDate : Other
9
False
Next
(Id: 10) Merge1 : Merge
10
Next
(Id: 11) Settle Instruction : Input
11
(Id: 13) X days : Interrupt
13
Next
(Id: 14) Process Instruction Rejection : Internal
14
Next
Next
(Id: 12) Merge2 : Merge
12
Next
(Id: 15) Send Not ation : Input
15
Next
(Id: 16) Proc otif : Include
16
Next
(Id: 17) Final : Exit
17

Generated Test Sequences

P_1 = [1,2,3,4,5,6,12,7,15,8,16,9,17,10,11,12,15,16,17]

P_2 = [1,2,3,4,5,6,12,7,15,8,16,10,17,11,12,15,16,17]

P_3= [1,2,3,4,5,14,12,15,16,17]

P_4= [13,14,15,16,17]

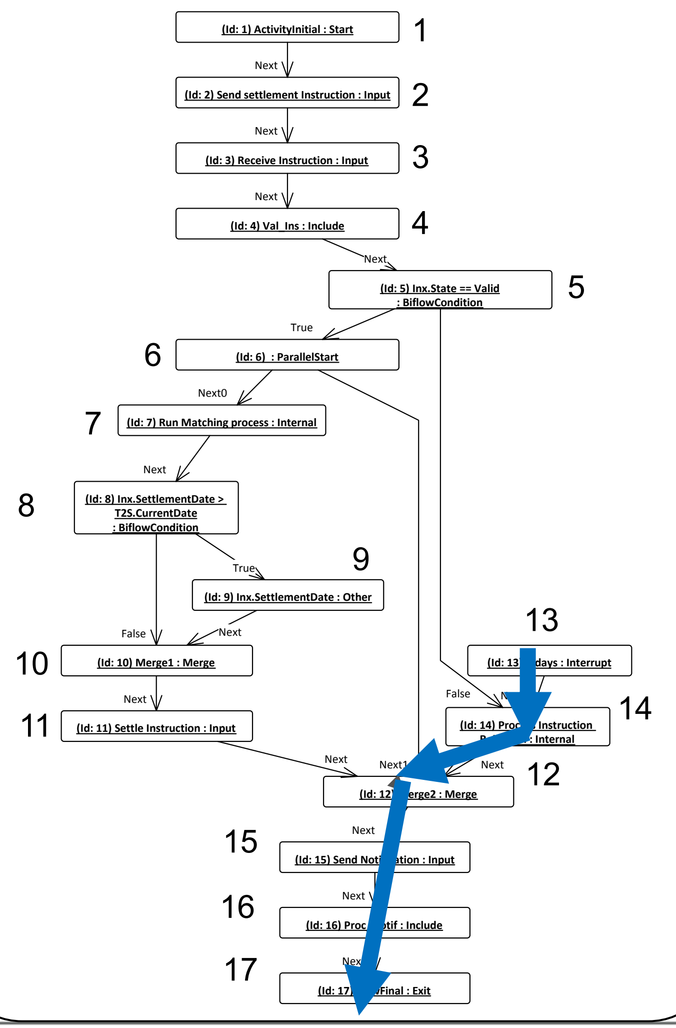
Test Sequences Generation

- Test Sequence: sequence of nodes
- Each test sequence leads to an Acceptance Criterion

Generated Test Sequences

P_1 = [1,2,3,4,5,6,12,7,15,8,16,9,17,10,11,12,15,16,17]
P_2 = [1,2,3,4,5,6,12,7,15,8,16,10,17,11,12,15,16,17]
P_3= [1,2,3,4,5,14,12,15,16,17]
P_4= [**13,14,15,16,17**]

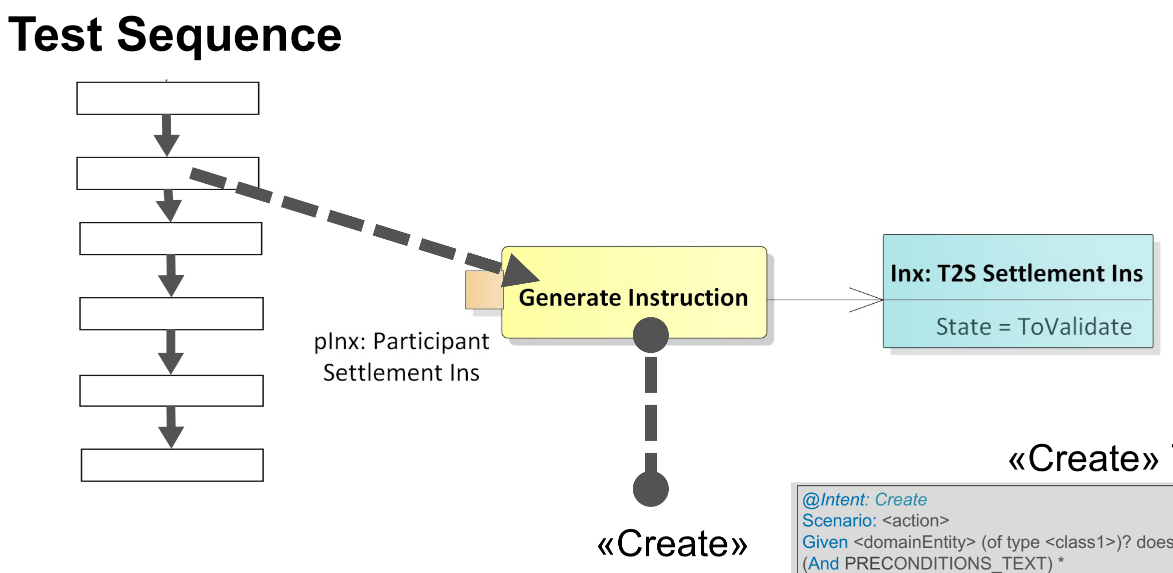
Test Model



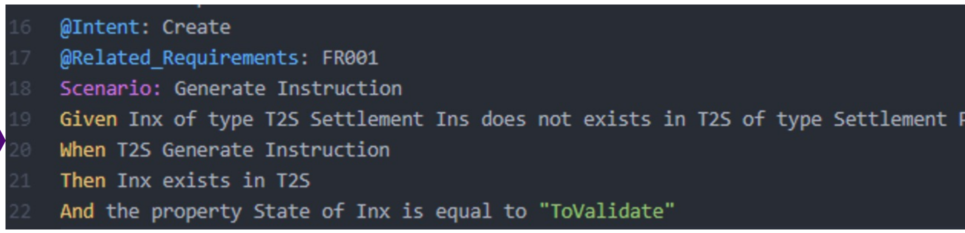
(Id: 1) ActivityInitial : Start
1
Next
(Id: 2) Send settlement Instruction : Input
2
Next
(Id: 3) Receive Instruction : Input
3
Next
(Id: 4) Val_Ins : Include
4
Next
(Id: 5) Inx.State == Valid : BiflowCondition
5
True
(Id: 6) : ParallelStart
6
Next0
(Id: 7) Run Matching process : Internal
7
Next
(Id: 8) Inx.SettlementDate > T2S.CurrentDate : BiflowCondition
8
True
(Id: 9) Inx.SettlementDate : Other
9
False
Next
(Id: 10) Merge1 : Merge
10
Next
(Id: 11) Settle Instruction : Input
11
13
(Id: 13) days : Interrupt
False
(Id: 14) Proc Instruction : Internal
14
Next
Next1
Next
(Id: 12) erge2 : Merge
12
Next
(Id: 15) Send Noti ation : Input
15
Next
(Id: 16) Proc otif : Include
16
Nex
(Id: 17) Final : Exit
17

Scenario Generation with Gherkin Templates

- 12 templates: one for each of the 11 intents + one for interrupts

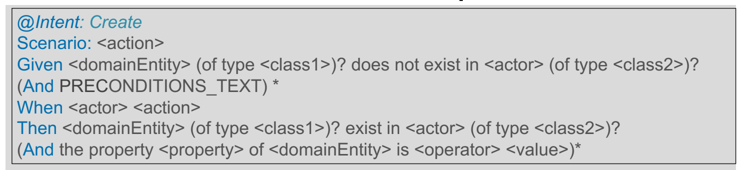


Gherkin Scenario



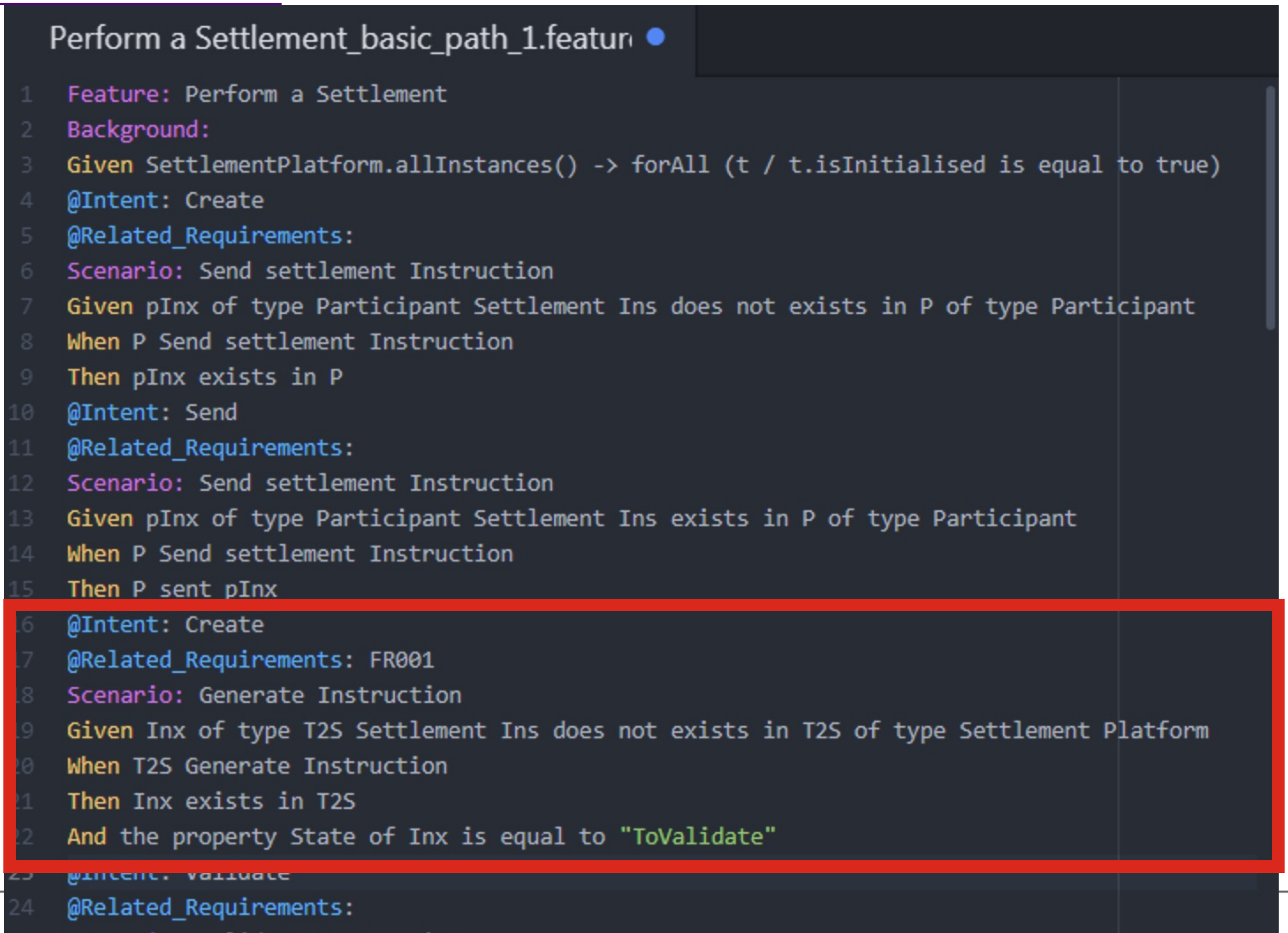
```
@Intent: Create
@Related_Requirements: FR001
Scenario: Generate Instruction
Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement F
When T2S Generate Instruction
Then Inx exists in T2S
And the property State of Inx is equal to "ToValidate"
```

«Create» Template



@Intent: Create
Scenario: <action>
Given <domainEntity> (of type <class1>)? does not exist in <actor> (of type <class2>)?
(And PRECONDITIONS_TEXT) *
When <actor> <action>
Then <domainEntity> (of type <class1>)? exist in <actor> (of type <class2>)?
(And the property <property> of <domainEntity> is <operator> <value>)*

Example of Generated Acceptance Criterion

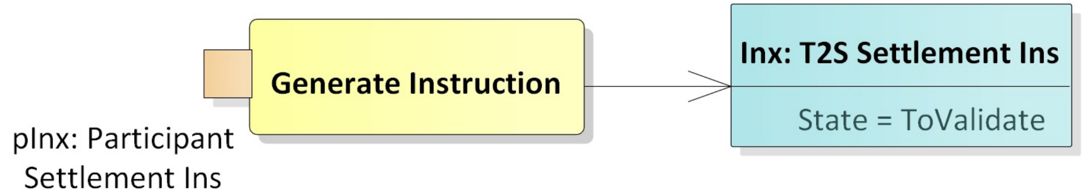


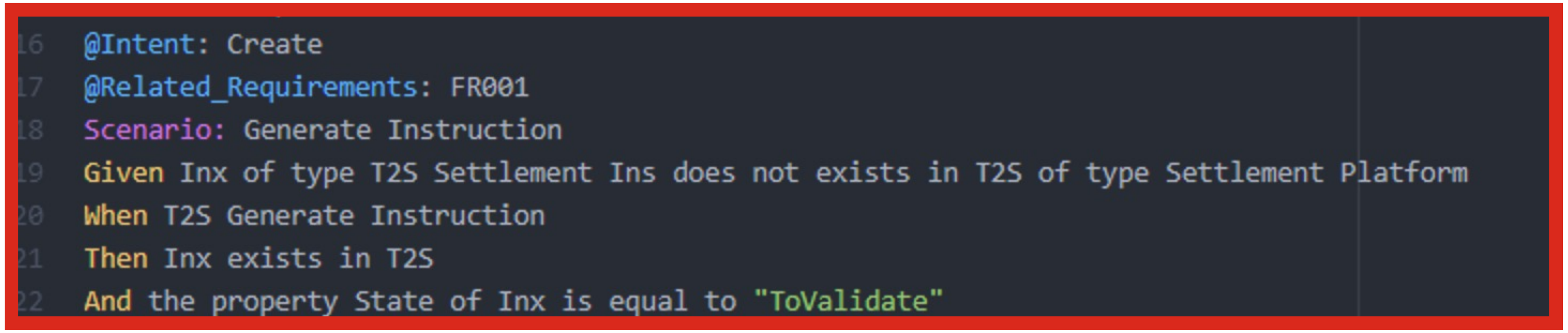
Perform a Settlement_basic_path_1.featur

```
Feature: Perform a Settlement
Background:
Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal to true)
@Intent: Create
@Related_Requirements:
Scenario: Send settlement Instruction
Given pInx of type Participant Settlement Ins does not exists in P of type Participant
When P Send settlement Instruction
Then pInx exists in P
@Intent: Send
@Related_Requirements:
Scenario: Send settlement Instruction
Given pInx of type Participant Settlement Ins exists in P of type Participant
When P Send settlement Instruction
Then P sent pInx
@Intent: Create
@Related_Requirements: FR001
Scenario: Generate Instruction
Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement Platform
When T2S Generate Instruction
Then Inx exists in T2S
And the property State of Inx is equal to "ToValidate"
@Intent: Validate
@Related_Requirements:
```

Template for «Create» Intent

```
@Intent: Create
@Related_Requirements: (<requirementID> (, <requirementID>)*)?
Scenario: <action>
Given <domainEntity> (of type <class1>)? does not exist in <actor> (of type <class2>)?
(And PRECONDITIONS_TEXT) *
When <actor> <action>
Then <domainEntity> (of type <class1>)? exist in <actor> (of type <class2>)?
(And GENERATED_POSTCONDITIONS_FOR_UPDATED_OBJECTS)*
```





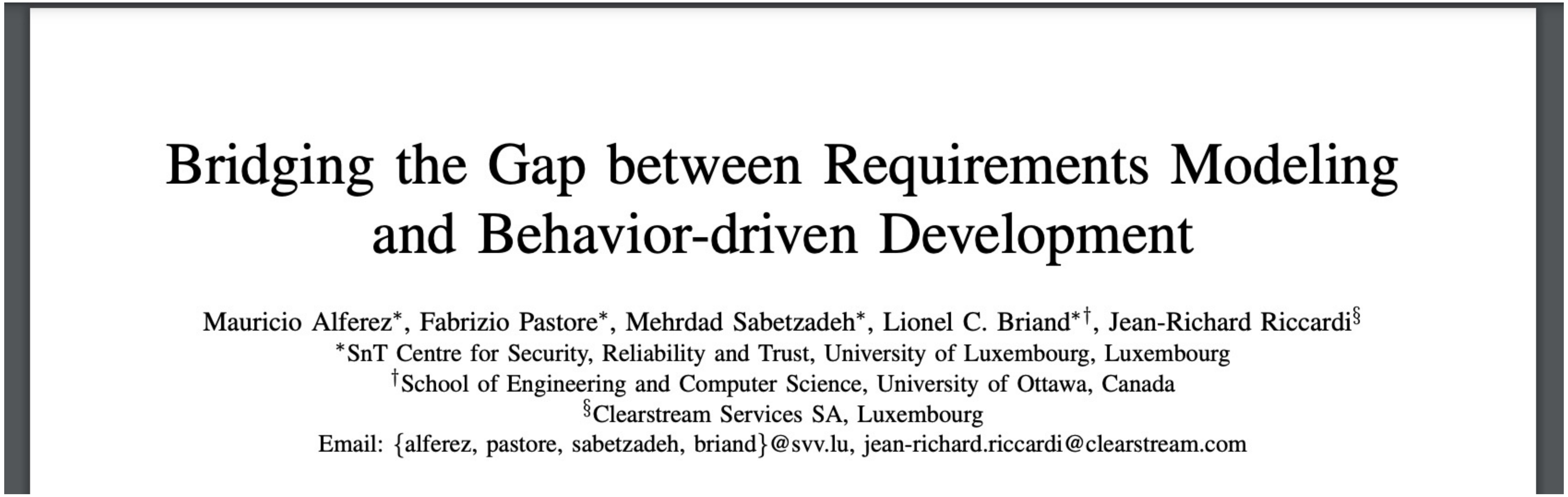
```
@Intent: Create
@Related_Requirements: FR001
Scenario: Generate Instruction
Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement Platform
When T2S Generate Instruction
Then Inx exists in T2S
And the property State of Inx is equal to "ToValidate"
```

AGAC Contributions

- Rely on Gherkin templates to produce Gherkin scenarios using information in the ADs
- Rely on automatically generated pre-/post- conditions to specify context and expected result for each Gherkin scenario
- Exercise relevant test paths (e.g., parallelism)

More Information about Qualisist Modeling Approach

https://orbilu.uni.lu/handle/10993/39710

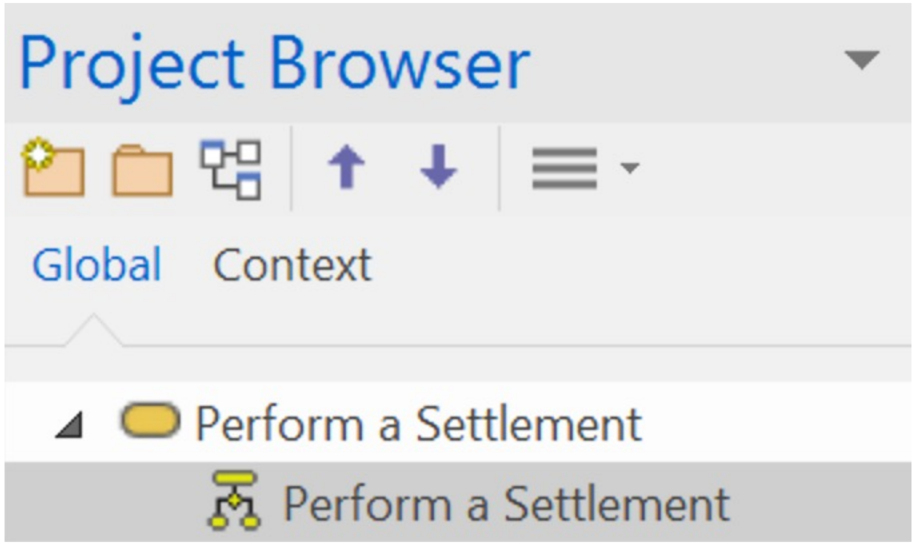


Bridging the Gap between Requirements Modeling and Behavior-driven Development

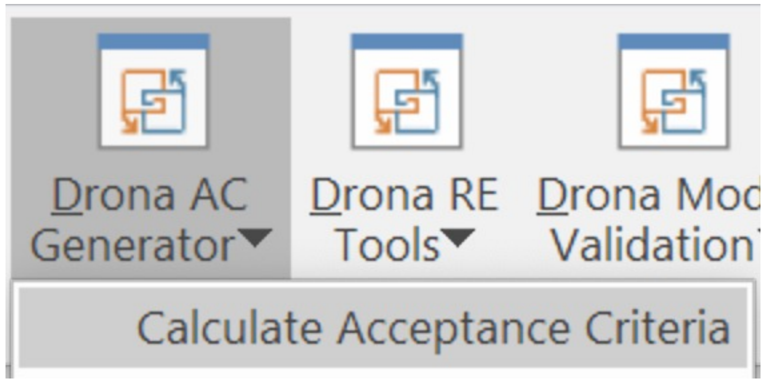
Mauricio Alferez*, Fabrizio Pastore*, Mehrdad Sabetzadeh*, Lionel C. Briand*†, Jean-Richard Riccardi§
*SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
†School of Engineering and Computer Science, University of Ottawa, Canada
§Clearstream Services SA, Luxembourg
Email: {alferez, pastore, sabetzadeh, briand}@svv.lu, jean-richard.riccardi@clearstream.com

Steps to Run the AC Generator

1 Select an Activity Diagram

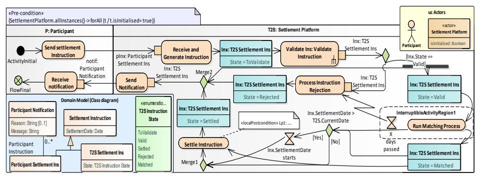


2 Select Specialize → Qualisist AC Generator → Calculate Acceptance Criteria

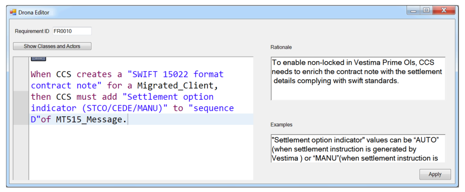


The Qualisist Solution

1. Modeling support



2. Requirements authoring support



Requirement ID FR0010

```
When CCS creates a "SWIFT 15022 format
contract note" for a Migrated_Client,
then CCS must add "Settlement option
indicator (STCO/CEDE/MANU)" to "sequence
D"of MT515_Message.
```

Rationale

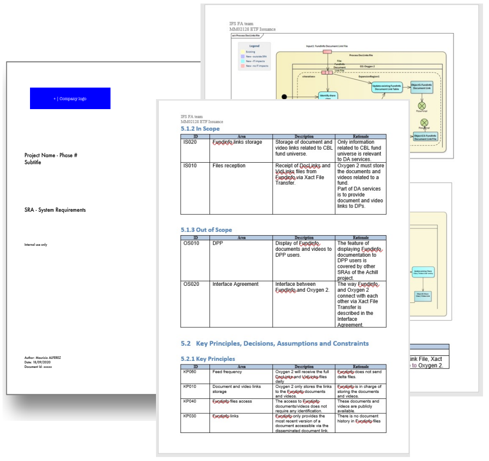
To enable non-locked in Vestima Prime OIs, CCS needs to enrich the contract note with the settlement details complying with swift standards.

Examples

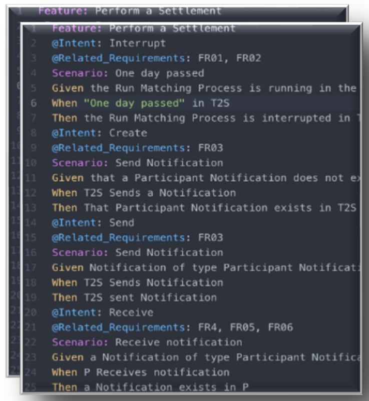
"Settlement option indicator" values can be "AUTO" (when settlement instruction is generated by Vestima) or "MANU"(when settlement instruction is

3. Requirements-to-model reconciliation support

4. Full deliverable generation



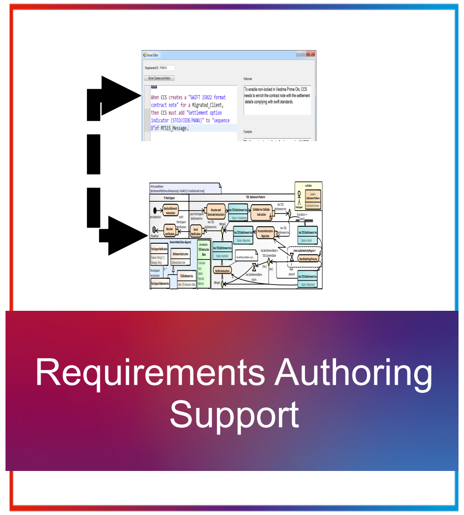
5. Gherkin test scenarios generation



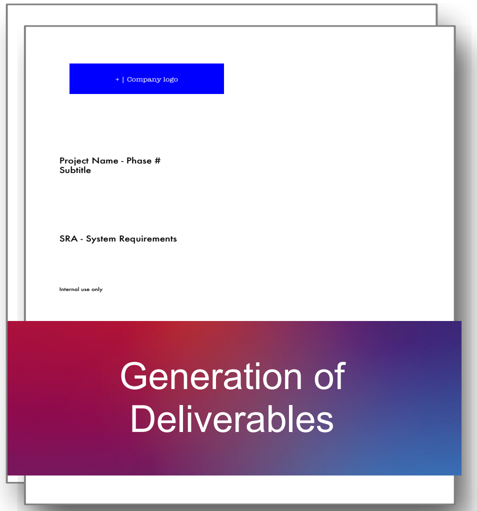
```
Feature: Perform a Settlement
@Intent: Interrupt
@Related_Requirements: FR01, FR02
Scenario: One day passed
Given the Run Matching Process is running in the
When "One day passed" in T2S
Then the Run Matching Process is interrupted in T
@Intent: Create
@Related_Requirements: FR03
Scenario: Send Notification
Given that a Participant Notification does not ex
When T2S Sends a Notification
Then That Participant Notification exists in T2S
@Intent: Send
@Related_Requirements: FR03
Scenario: Send Notification
Given Notification of type Participant Notificati
When T2S Sends Notification
Then T2S sent Notification
@Intent: Receive
@Related_Requirements: FR4, FR05, FR06
Scenario: Receive notification
Given a Notification of type Participant Notifica
When P Receives notification
Then a Notification exists in P
```

Accelerate Time to Market

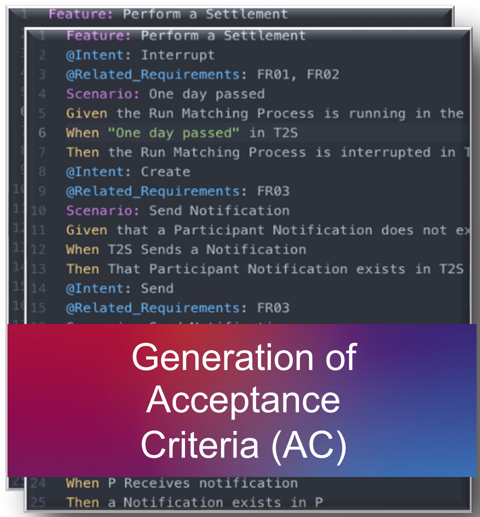
Qualisist requirements authoring support, deliverables generation and automation of Acceptance Criteria generation **will save significant time on your projects!**



Requirements Authoring Support



Generation of Deliverables



```
Feature: Perform a Settlement
@Intent: Interrupt
@Related_Requirements: FR01, FR02
Scenario: One day passed
Given the Run Matching Process is running in the
When "One day passed" in T2S
Then the Run Matching Process is interrupted in T
@Intent: Create
@Related_Requirements: FR03
Scenario: Send Notification
Given that a Participant Notification does not ex
When T2S Sends a Notification
Then That Participant Notification exists in T2S
@Intent: Send
@Related_Requirements: FR03
When P Receives notification
Then a Notification exists in P
```

Generation of Acceptance Criteria (AC)

Imagine what Qualisist can do for you

Contact Persons

Elene Pitskhelauri, IFS IT Luxembourg
Email: elene.pitskhelauri@clearstream.com
Thomas Henin, IFS IT Luxembourg
Email: thomas.henin@clearstream.com