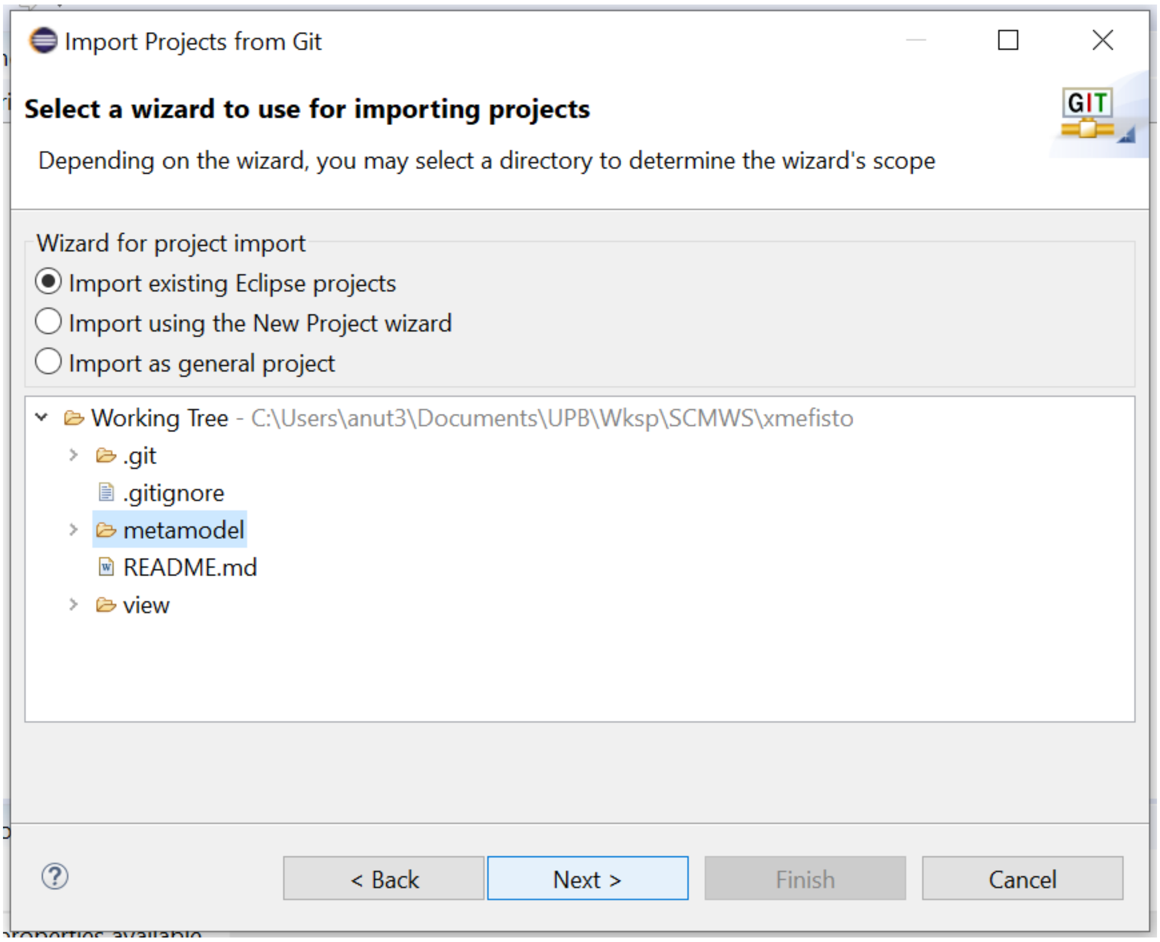
Guide to Create the Situational Context Model using the extended modeling workbench for Extended Method Engineering Framework for Situation-Specific Software Transformation Methods (xMEFiSTo).

- First, you have to the associated Sirius plugin in eclipse. Refer tutorial to learn more about Eclipse Sirius.

- Open Eclipse > open git perspective > clone the git repository https://git.cs.uni-paderborn.de/thottam/xmefisto.git > select a folder using browse button if you want to clone it to a specific folder.

- Switch to Modeling Perspective > Right-click on Model explorer and import projects from Git > Select 'Existing local repository' > Select only 'metamodel'.



Import Projects from Git

Select a wizard to use for importing projects

Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

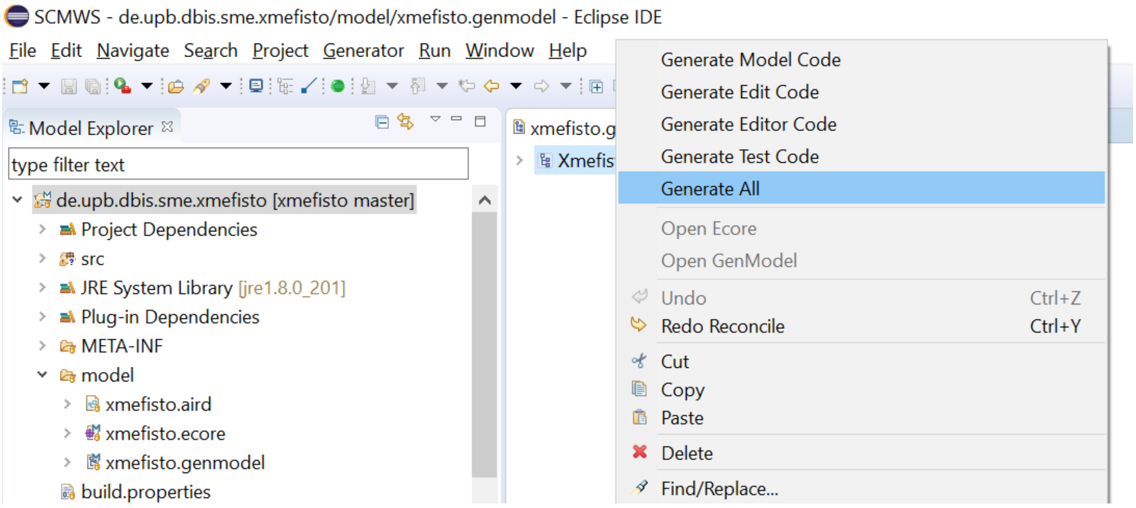
● Import existing Eclipse projects

○ Import using the New Project wizard

○ Import as general project

Working Tree - C:\Users\anut3\Documents\UPB\Wksp\SCMWS\xmefisto
- .git
- .gitignore
- metamodel
- README.md
- view

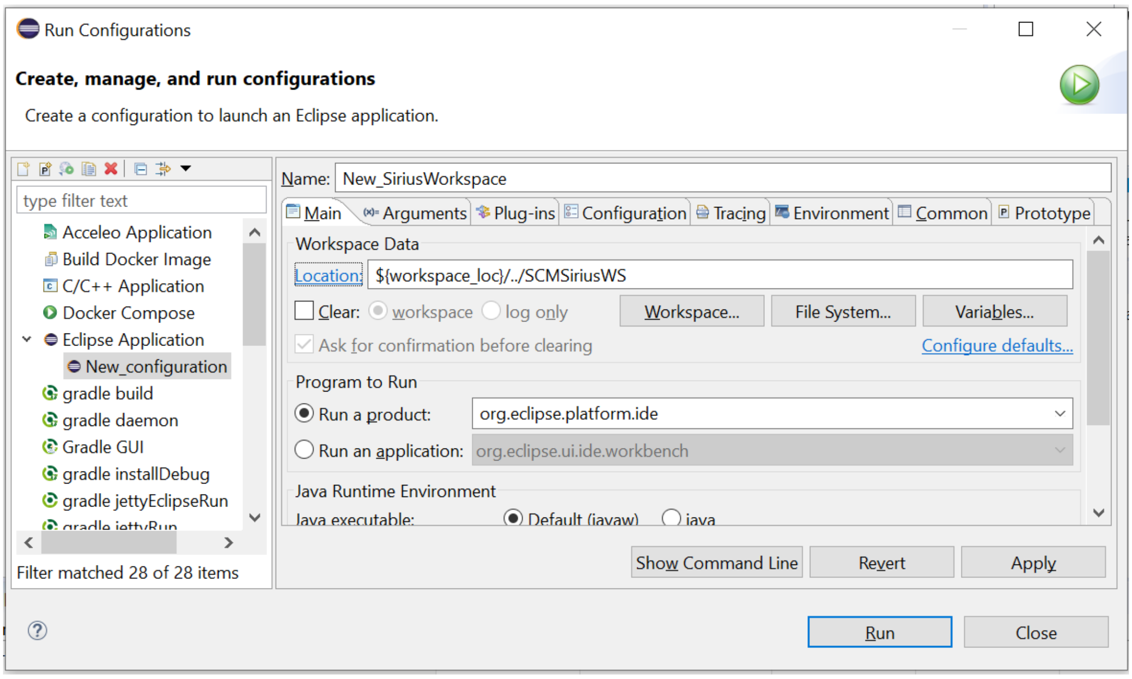
- Open the model 'xmefisto.genmodel' from the model explorer > right-click on 'Xmefisto' and click on 'Generate All'.



SCMWS - de.upb.dbis.sme.xmefisto/model/xmefisto.genmodel - Eclipse IDE

- de.upb.dbis.sme.xmefisto [xmefisto master]
  - Project Dependencies
  - src
  - JRE System Library [jre1.8.0_201]
  - Plug-in Dependencies
  - META-INF
  - model
    - xmefisto.aird
    - xmefisto.ecore
    - xmefisto.genmodel
  - build.properties

- Switch to Java perspective > go to run configurations and add new configuration under Eclipse Application > Click on Apply and then Run the configuration to open a new eclipse workspace.



Run Configurations

Create, manage, and run configurations

Create a configuration to launch an Eclipse application.

Name: New_SiriusWorkspace

Workspace Data

Location: ${workspace_loc}/../SCMSiriusWS

□ Clear: ○ workspace ○ log only

☑ Ask for confirmation before clearing

Program to Run

● Run a product: org.eclipse.platform.ide

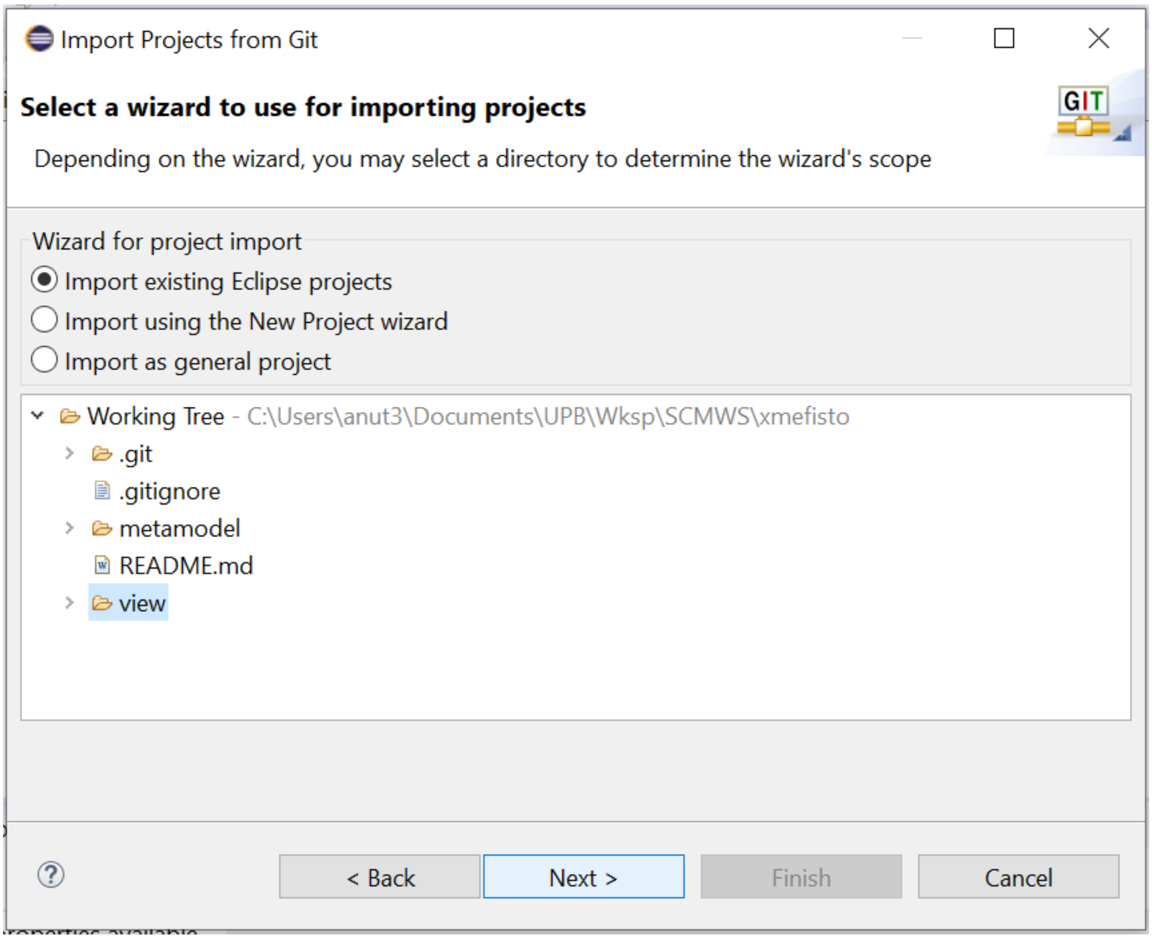
○ Run an application: org.eclipse.ui.ide.workbench

Java Runtime Environment

Java executable: ● Default (javaw) ○ java

- In the new eclipse instance, open Git perspective > Add an existing local Git repository > Browse and select the folder to which we cloned the git repository > click on Add.

- Switch to Sirius perspective > Right-click on model Explorer and import projects from Git > select Existing local repository > select 'xmefisto' which is the existing repository > select only 'view'.



Import Projects from Git

Select a wizard to use for importing projects

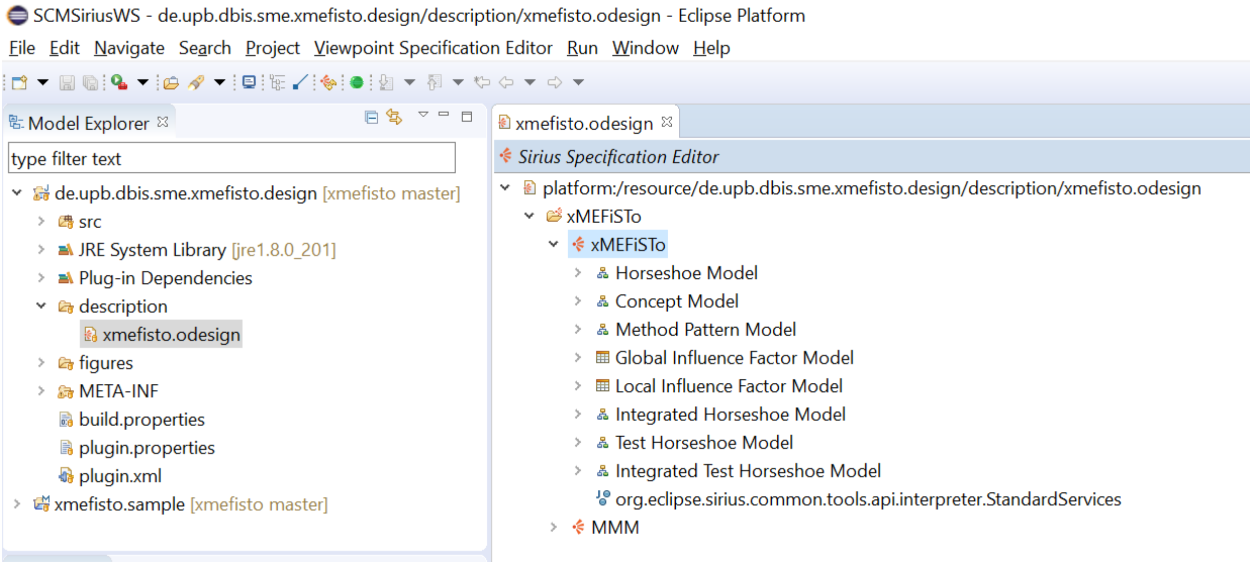
Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

● Import existing Eclipse projects
○ Import using the New Project wizard
○ Import as general project

```
⌄ Working Tree - C:\Users\anut3\Documents\UPB\Wksp\SCMWS\xmefisto
  › .git
    .gitignore
  › metamodel
    README.md
  › view
```

- The main definition for the tool to create the required models is the ‘xmefisto.odesign’ file.

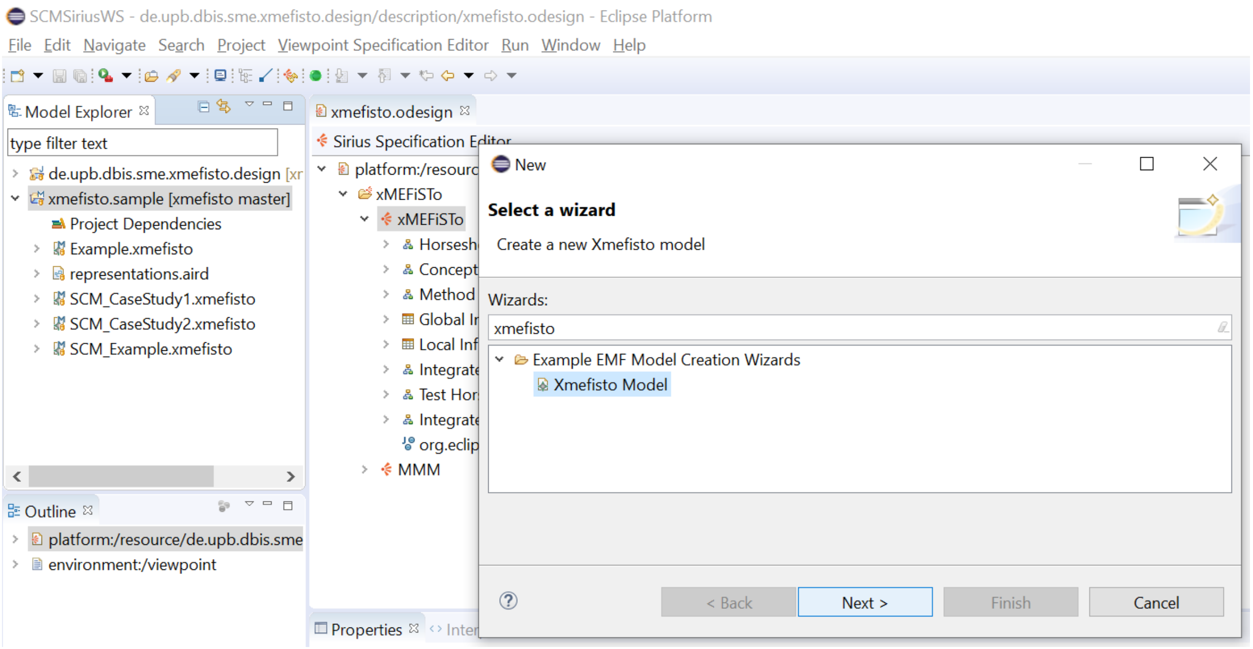


SCMSiriusWS - de.upb.dbis.sme.xmefisto.design/description/xmefisto.odesign - Eclipse Platform

```
Model Explorer
type filter text
⌄ de.upb.dbis.sme.xmefisto.design [xmefisto master]
  › src
  › JRE System Library [jre1.8.0_201]
  › Plug-in Dependencies
  ⌄ description
      xmefisto.odesign
  › figures
  › META-INF
    build.properties
    plugin.properties
    plugin.xml
› xmefisto.sample [xmefisto master]
```

```
xmefisto.odesign
Sirius Specification Editor
⌄ platform:/resource/de.upb.dbis.sme.xmefisto.design/description/xmefisto.odesign
  ⌄ xMEFiSTo
    ⌄ xMEFiSTo
      › Horseshoe Model
      › Concept Model
      › Method Pattern Model
      › Global Influence Factor Model
      › Local Influence Factor Model
      › Integrated Horseshoe Model
      › Test Horseshoe Model
      › Integrated Test Horseshoe Model
        org.eclipse.sirius.common.tools.api.interpreter.StandardServices
    › MMM
```

- Right-click on ‘xmefisto.sample’ project in the model explorer and click on new – other – Xmefisto Model.



SCMSiriusWS - de.upb.dbis.sme.xmefisto.design/description/xmefisto.odesign - Eclipse Platform

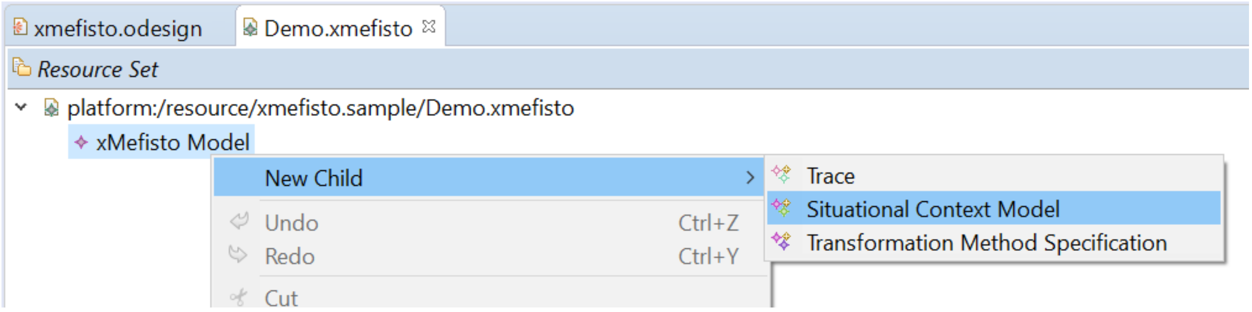
New

Select a wizard

Create a new Xmefisto model

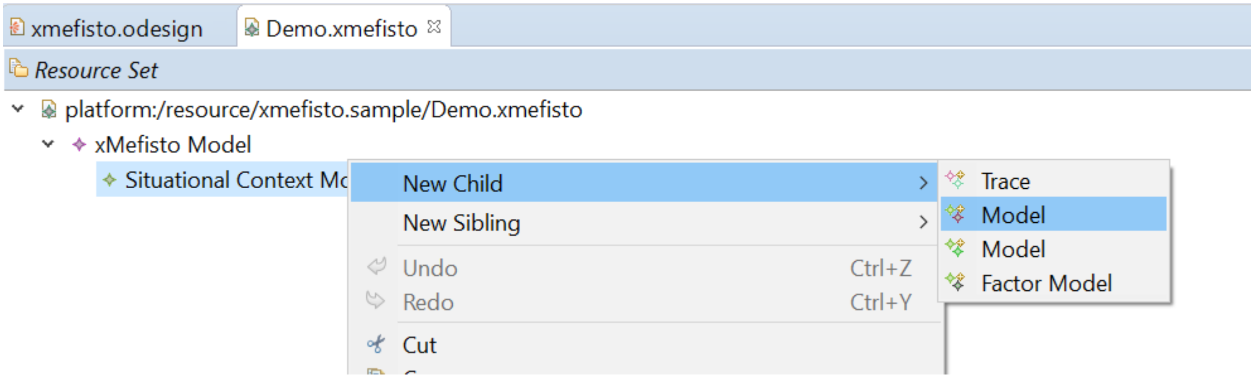
Wizards: xmefisto

- Right-click on the root element and create a new child ‘Situational Context Model’

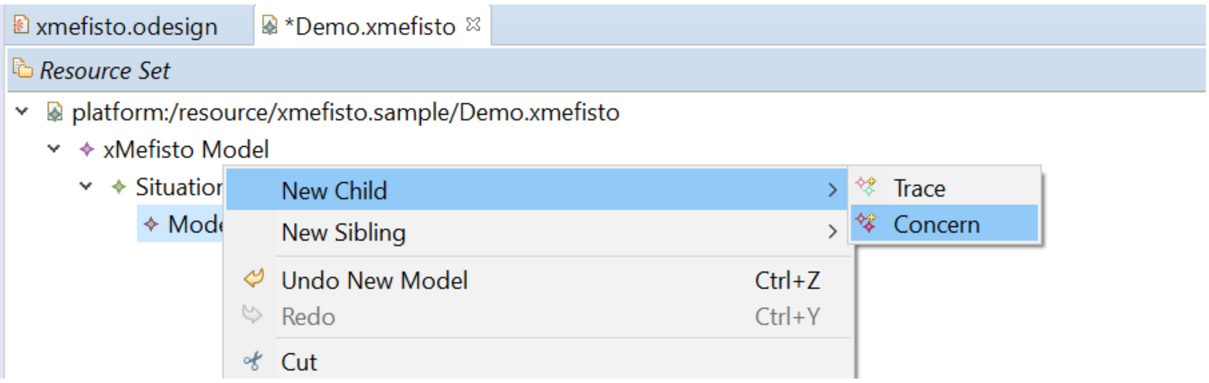


xmefisto.odesign
Demo.xmefisto
Resource Set
platform:/resource/xmefisto.sample/Demo.xmefisto
xMefisto Model
New Child
Undo Ctrl+Z
Redo Ctrl+Y
Cut
Trace
Situational Context Model
Transformation Method Specification

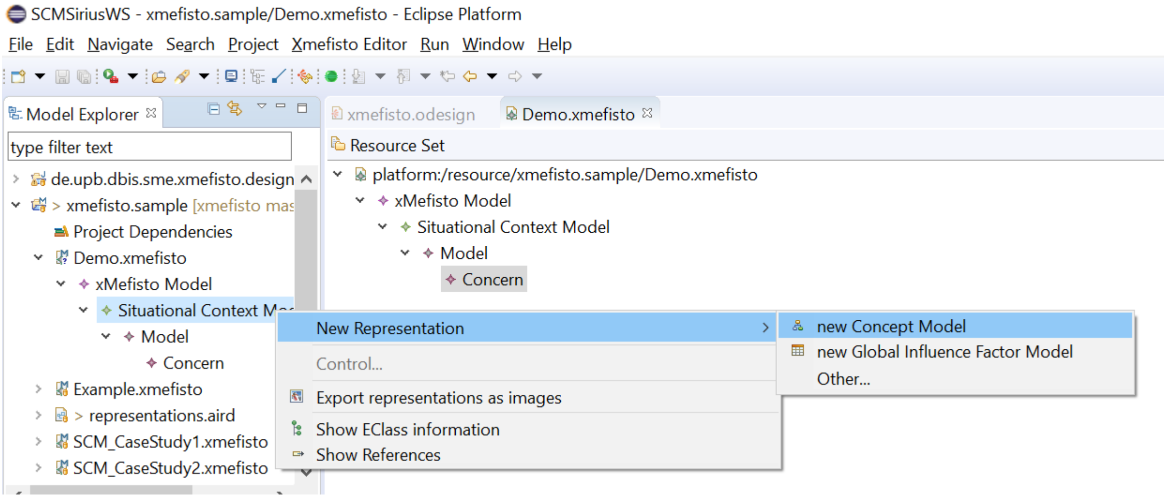
- Right-click on Situational Context Model and create a New Child ‘Model’



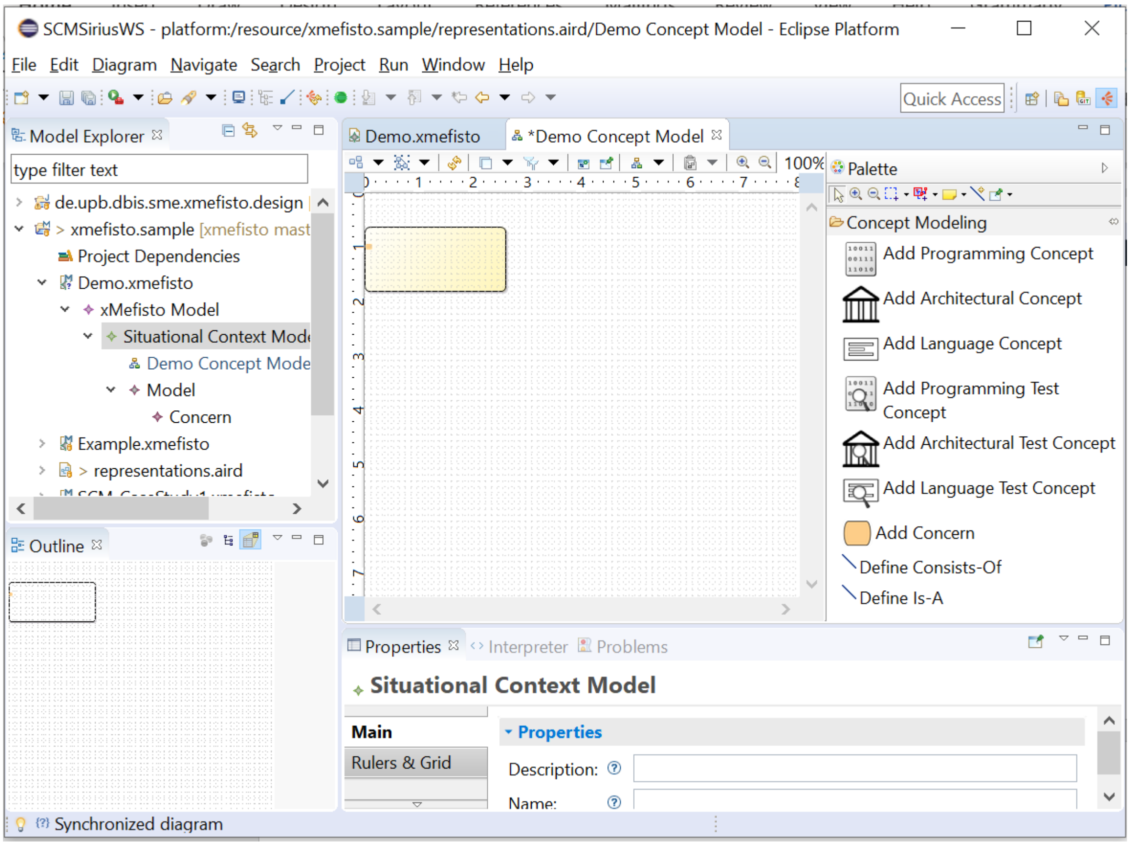
- Right-click on Model and create a New Child ‘Concern’



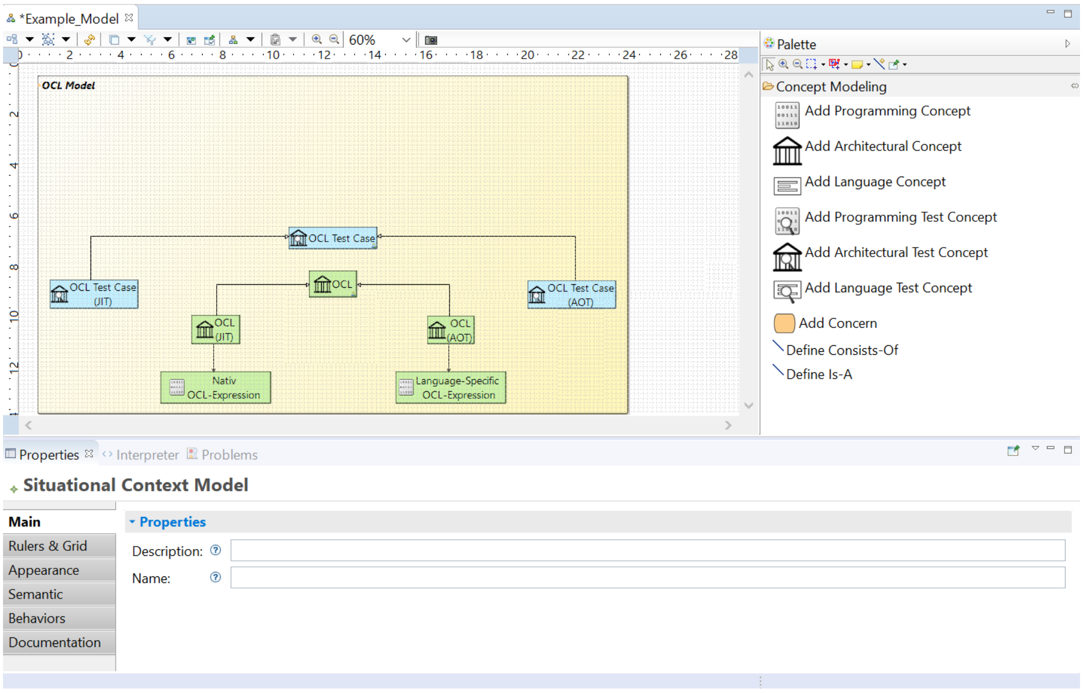
- Right-click on ‘Situational Context model’ in the model explorer and create a ‘new Concept Model’



- Now the modeling editor is opened with the concern that was created in it. You can see a ‘Palette’ to the right-hand side which helps you create the Concept Model.



- Using this, you can easily create the concept model as shown below:



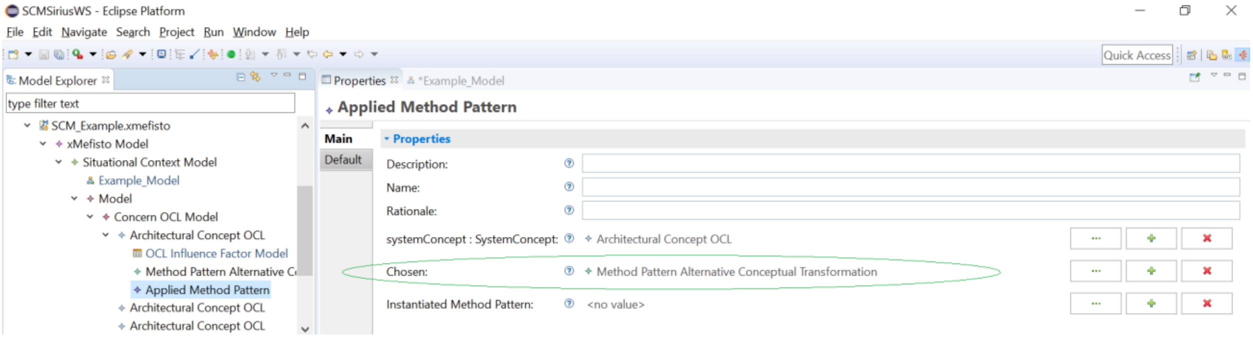
Situational Context Model

Properties

Description:

Name:

- To add the applied method pattern for system migration in the shared system concept, right-click on the shared system concept and create a new (tabular) Influence Factor Model and add the system method pattern that was applied. Now select the corresponding applied method pattern from the model explorer and select the chosen field as shown below:


SCMSiriusWS - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Model Explorer
type filter text
SCM_Example.xmefisto
xMefisto Model
Situational Context Model
Example_Model
Model
Concern OCL Model
Architectural Concept OCL
OCL Influence Factor Model
Method Pattern Alternative C
Applied Method Pattern
Architectural Concept OCL
Architectural Concept OCL
Properties
*Example_Model
Main
Default

Applied Method Pattern

Properties

Description:

Name:

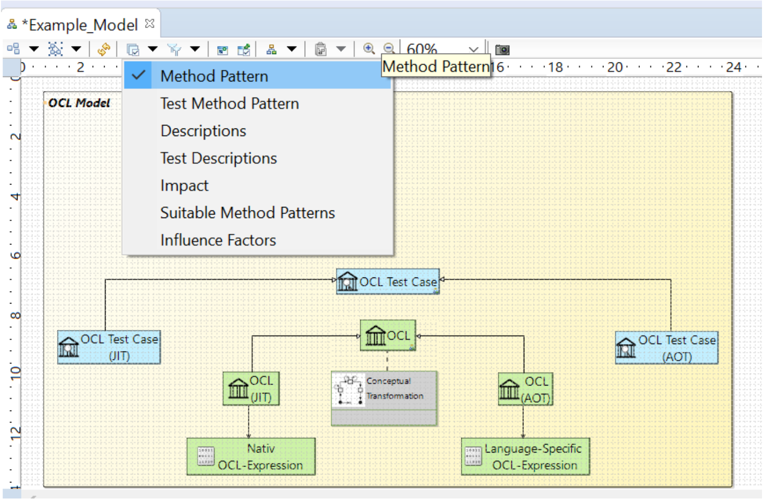
Rationale:

systemConcept : SystemConcept: Architectural Concept OCL

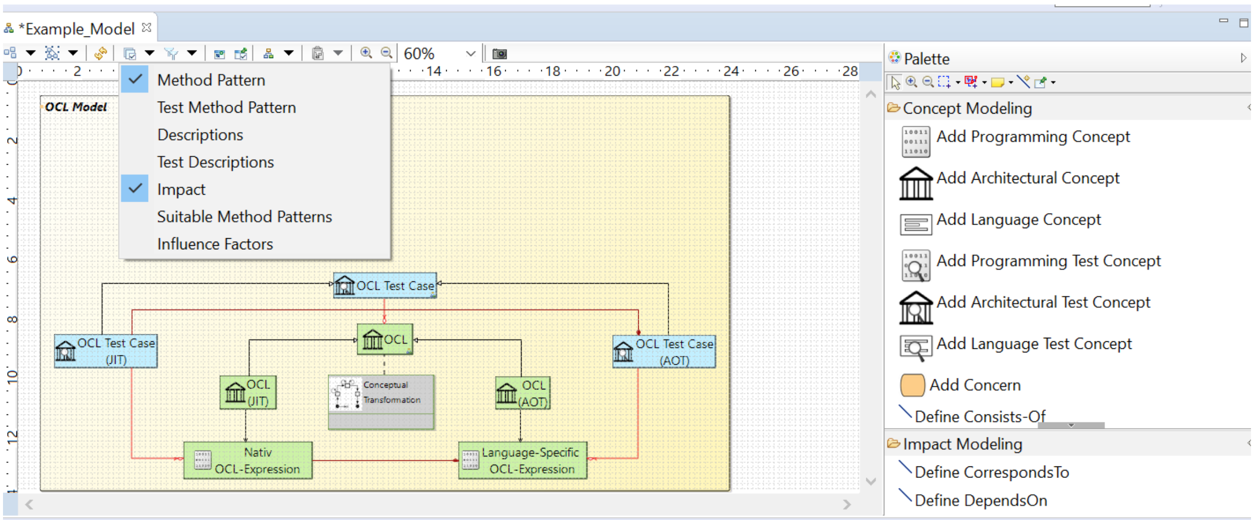
Chosen: Method Pattern Alternative Conceptual Transformation

Instantiated Method Pattern: <no value>

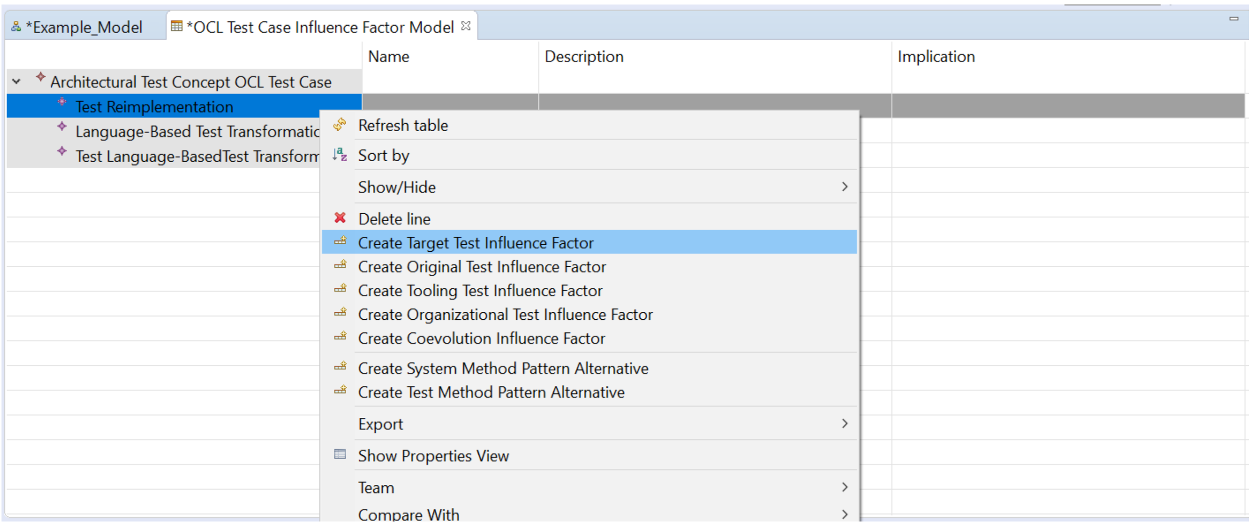
- Now select the ‘Method Pattern’ Layer.



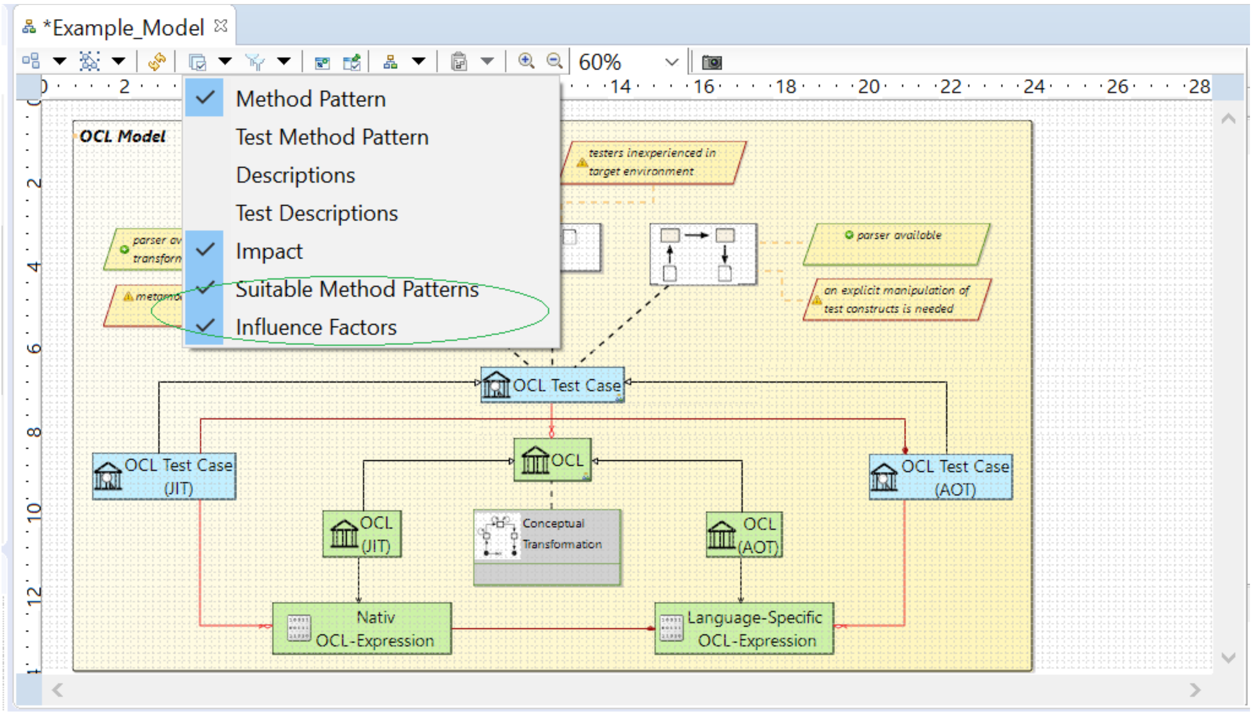
- Next step is to add the co-evolution dependencies and create the impact model. For that select the ‘Impact’ layer from the drop-down so that the corresponding relationships are visible in the palette. Using this you can define the impact model.



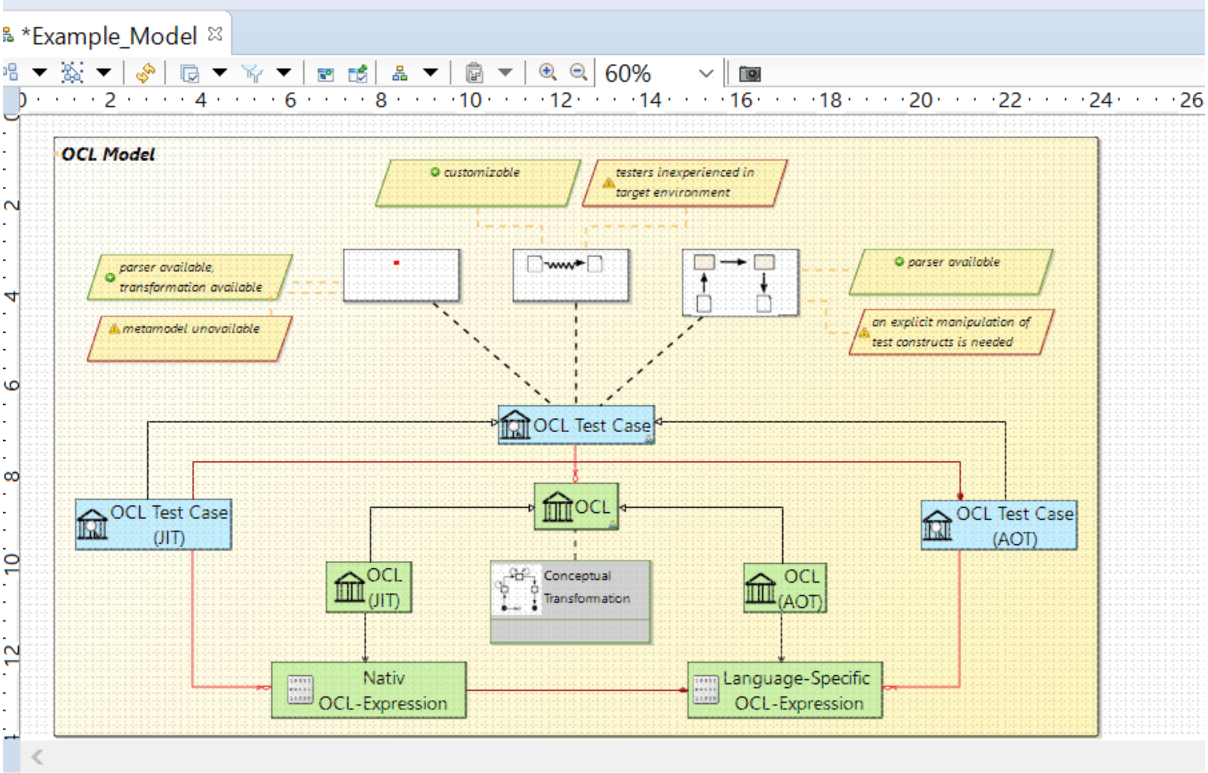
- Next step is to add the suitable test method patterns and extend this model. For that right-click on the shared test concept and create a new (tabular) Influence Factor Model. You can add all the suitable test method patterns and also create specific test influence factors in this model.


*Example_Model
*OCL Test Case Influence Factor Model
Name
Description
Implication
Architectural Test Concept OCL Test Case
Test Reimplementation
Language-Based Test Transformatic
Test Language-BasedTest Transform
Refresh table
Sort by
Show/Hide
Delete line
Create Target Test Influence Factor
Create Original Test Influence Factor
Create Tooling Test Influence Factor
Create Organizational Test Influence Factor
Create Coevolution Influence Factor
Create System Method Pattern Alternative
Create Test Method Pattern Alternative
Export
Show Properties View
Team
Compare With

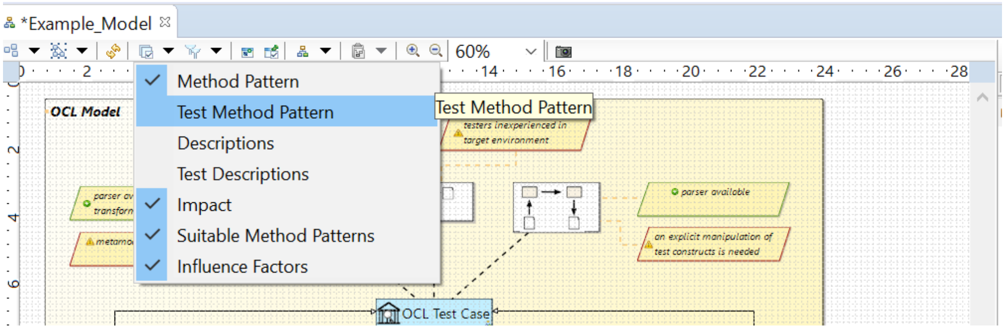
- Now select the ‘Suitable Method Patterns’ layer and the ‘Influence Factor’ layer from the drop-down. You can also edit the pros and cons of each suitable test method pattern in the graphical view of the Influence Factor Model.



- The Concept Model, the Impact Model, and the Influence Factor Model together constitute the Situational Context Model as shown below.


*Example_Model
60%
OCL Model
customizable
testers inexperienced in target environment
parser available, transformation available
metamodel unavailable
parser available
an explicit manipulation of test constructs is needed
OCL Test Case
OCL Test Case (JIT)
OCL
OCL Test Case (AOT)
OCL (JIT)
Conceptual Transformation
OCL (AOT)
Nativ OCL-Expression
Language-Specific OCL-Expression

- We have also added a provision to see the applied test method pattern for the shared test concept. For this select ‘Test Method Pattern’ layer from the drop-down.


*Example_Model
60%
Method Pattern
Test Method Pattern
Descriptions
Test Descriptions
Impact
Suitable Method Patterns
Influence Factors
Test Method Pattern
OCL Model
testers inexperienced in target environment
parser available
an explicit manipulation of test constructs is needed
OCL Test Case