Project References

Project Acronym	XIVT
Project Title	eXcellence In Variant Testing
Project Number	17039
Project Start Date	November 1, 2018 — Project Duration 36 months
Project Manager	Gunnar Widforss, Bombardier Transportation, Sweden
Website	https://www.xivt.org/

Document References

Work package	WP3: Testing of Configurable Products		
Deliverable	D3.2.b: Tool for assessment of test suite quality		
Deliverable type	Software (SW)		
Dissemination level	Public	Date & Version	Feb 1, 2021 V1.1
Mapped tasks	T3.2 Test case instantiation and distribution of testing efforts amongst variants		

1. Executive Summary

This deliverable includes the description of the tool “TSQA” for assessing the quality of test suites, which was developed and extended in the XIVT project in the scope of WP3. Author of the initial version of the tool is Dr. H. Lackner at Fraunhofer FOKUS, revisions have been done by BA T. Sikatzki and Prof. Dr. H. Schlingloff, Fraunhofer FOKUS.

2. Access Information

XIVT project has its repository on Gitlab at: https://gitlab.com/xivt

The tools are accessible at https://gitlab.com/xivt/itea
with username: ITEA3XIVT
and password: 20222018XIVT

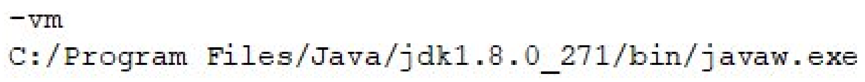
3. Tool Description
QATS – The **Q**uality **A**ssessor for **T**est **S**uites

1. Overview of the functionality

QATS uses mutation operators on the domain model to mimic development faults in product models to assess the fault detection capability of a test suite. By using mutating operators on the domain model and the test suite configurations QATS returns a set of product model mutants and generates product mutants based on these models. At last, the tests are executed on the created product mutants and a mutation score will be generated based on the number of killed domain model mutants.

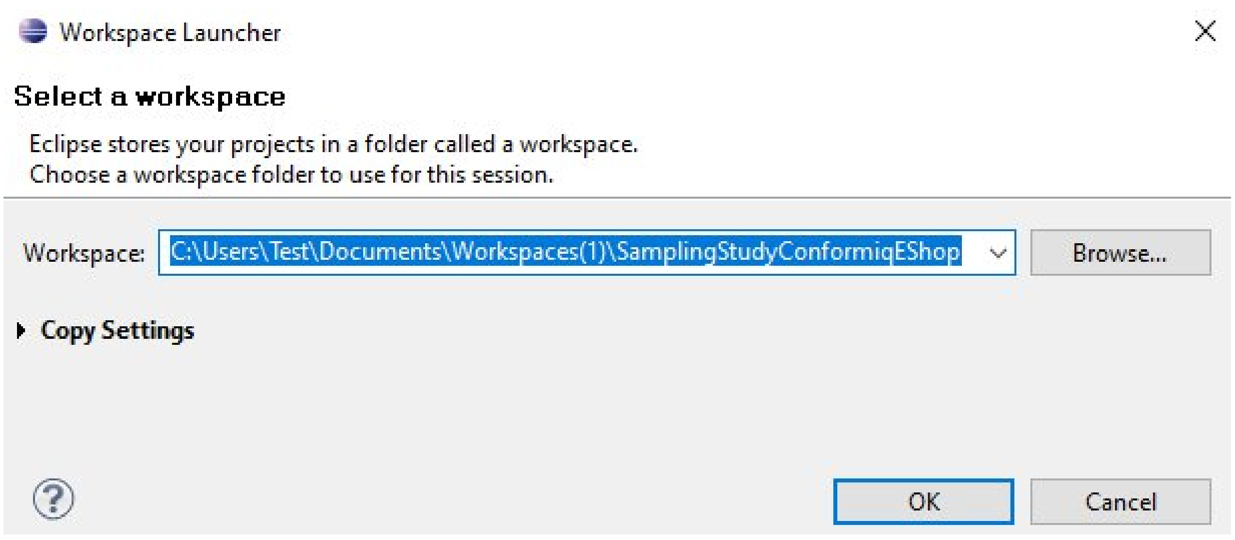
2. System Requirements and Installation

The Eclipse IDE distribution that is provided can only be used with a Windows system at the moment. It is required to have Java 8 installed before you run Eclipse. Also make sure that, in the “eclipse.ini” which is located in the Eclipse directory, the correct path of your “javaw.exe” is set. See the picture below



```
-vm
C:/Program Files/Java/jdk1.8.0_271/bin/javaw.exe
```

For the first execution you can use one of the “SamplingStudy” directories as your workspace in Eclipse. Each of the directories in the workspace contains different variants of the given product line.

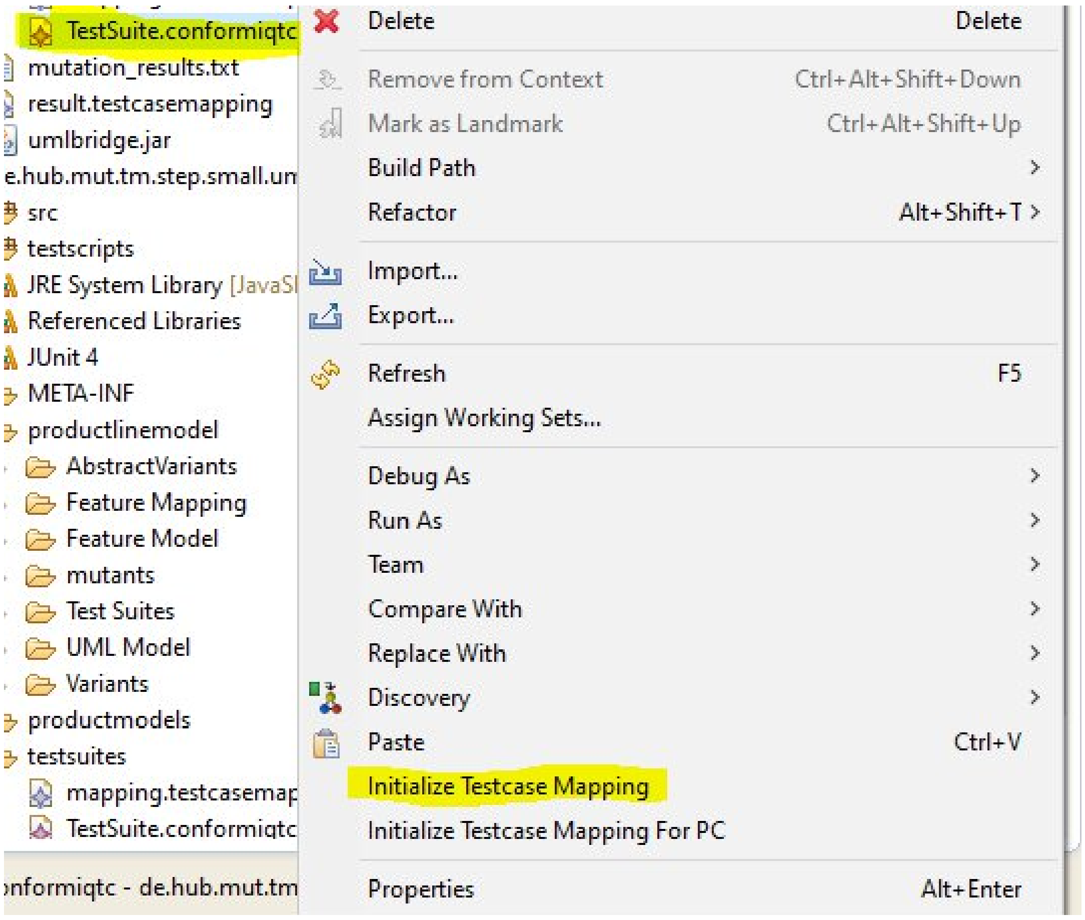

Workspace Launcher
Select a workspace
Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.
Workspace: C:\Users\Test\Documents\Workspaces(1)\SamplingStudyConformiqEShop
Browse...
Copy Settings
OK
Cancel

3. Basic Tool Usage, Input and Output

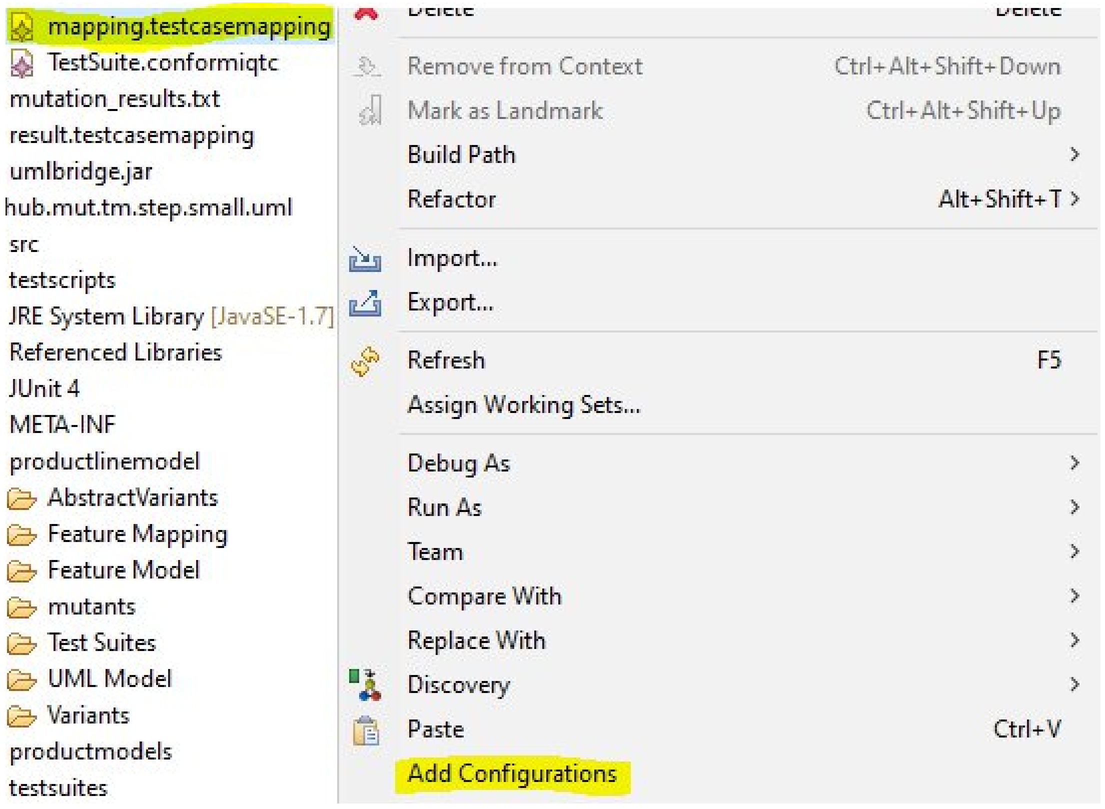
3.1 Input

As input you need a UML model of the product line as well as the correlated feature model and feature mapping. You also need to provide the test suite under test, the feature configurations for each product variant and the UML model for the product variants.

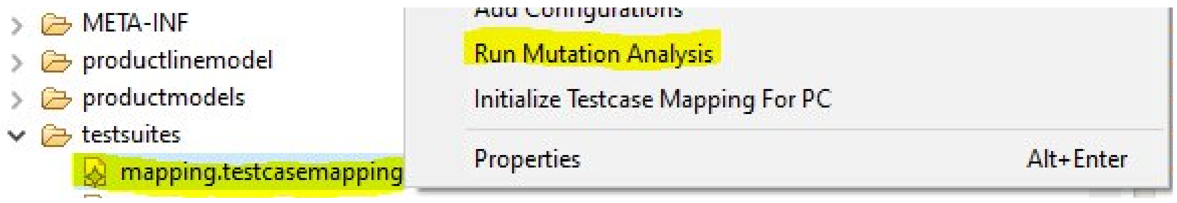
First you will want to initialize the test case mapping by right-clicking on the testsuite.xml you provide. Our example test suite was created by conformiq therefore the example test suites are called TestSuite.conformiq and can be found in the “testsuites” directory.

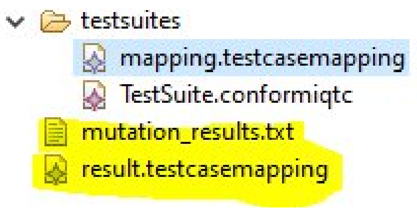


As the next step you need to add the variants to the test case mapping. Right click on the test case mapping and select “Add Configurations” then choose the directory that includes your variants.



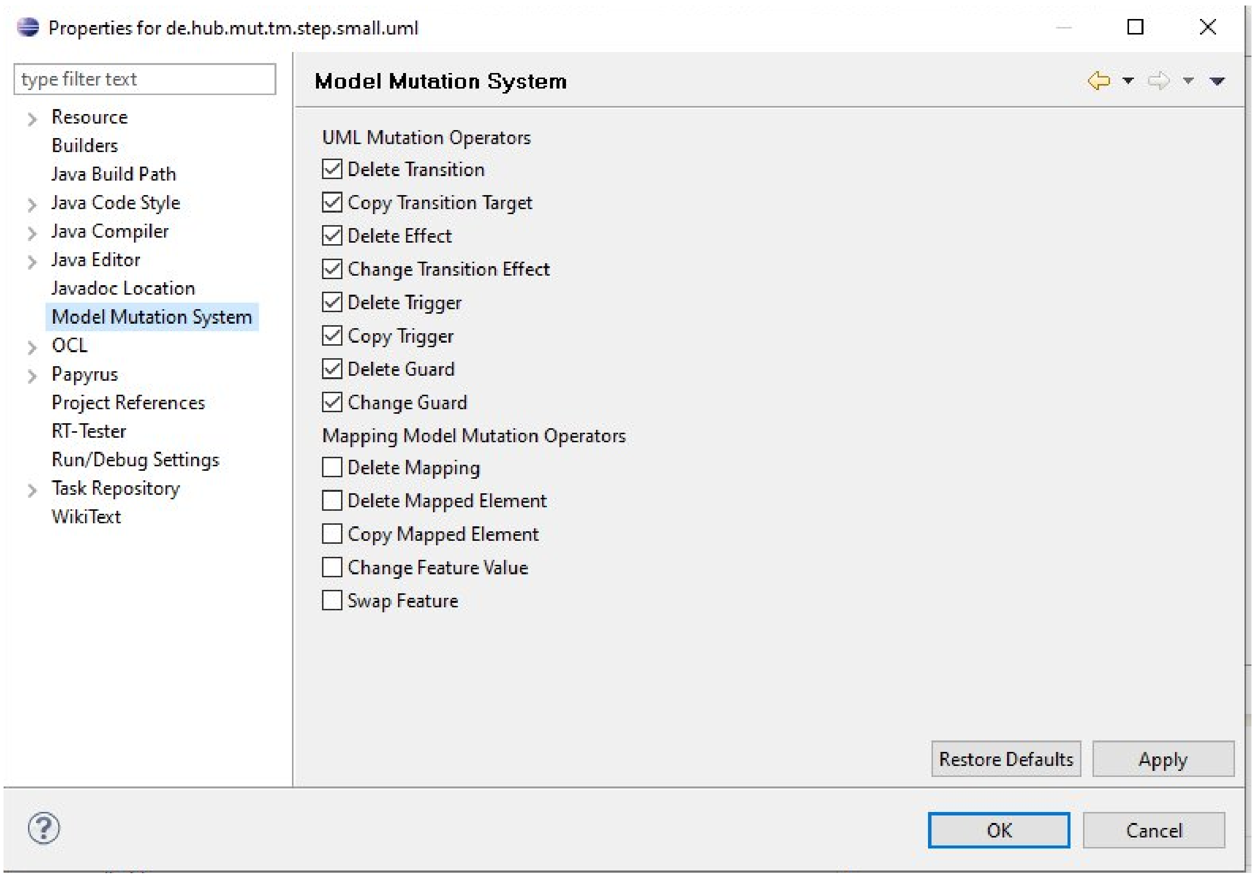
This will set up the test case mapping so you can run the mutation analysis on it. This will create the mutants and a “testcasemapping.result”. In some cases the test suite will then be executed automatically in other cases you will need to run the mutation analysis directly on the “testcasemapping.result” file. This will generate the “mutation_results.txt” that includes the mutation score.





4. User Manual

For the mutation we are using multiple mutation operators. We provide mutation operators which operate on the UML model, as well as operators which alter the feature mappings. By right-clicking on the project you can select “properties” and select the operators that should be used for the given project.



Model Mutation System

UML Mutation Operators
- [✓] Delete Transition
- [✓] Copy Transition Target
- [✓] Delete Effect
- [✓] Change Transition Effect
- [✓] Delete Trigger
- [✓] Copy Trigger
- [✓] Delete Guard
- [✓] Change Guard

Mapping Model Mutation Operators
- [ ] Delete Mapping
- [ ] Delete Mapped Element
- [ ] Copy Mapped Element
- [ ] Change Feature Value
- [ ] Swap Feature

5. Mutation Operators

5.1. UML Operators

Delete Transition – Deletes a transition in the UML model

Copy Transition Target – Changes the target of a transition by copying the target state of another transition in the Model

Delete Effect – Deletes a random effect of a transition

Change Transition Effect – not yet implemented

Delete Trigger – Deletes a random trigger from a transition

Copy Trigger – Adds another trigger from the same region to a random transition

Delete Guard – Deletes a guard from a random transition

Change Guard – Changes the guard of a random transition (flip booleans, invert arithmetic operators...)

5.2. Feature Mapping Operators

Delete Mapping – Deletes a mapping

Delete Mapped Element – delete the UML element of a mapping

Copy Mapped Element – Similar to Copy Trigger. Adds a UML element from the subsequent mapping to a mapping.

Change Feature Value – Flips the feature value of a feature in a mapping

Swap Feature – Swaps mapped behavior between two mappings

6. Examples

We already have pre-generated results for three systems. An Alarm System, a Ticket Machine and an ecommerce Shop. These also come with example test suites so you can get an idea of how QATS works.