securityandtrust.lu

Product line Use case Model Configurator (PUMConf)
User Manual v1.5

Software Verification and Validation Laboratory
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg

January 30, 2019

UNIVERSITÉ DU LUXEMBOURG

Contents

1 Introduction

PUMConf is a tool, integrated with an industrial requirements management tool, i.e., IBM DOORS, for supporting use case-driven configuration approach. It interactively receives configuration decisions from the analysts to generate Product Specific (PS) use case and domain models. PUMConf employs the modeling methodology presented at the International Conference on Model Driven Engineering Languages and Systems (MODELS 2015) [**?**].

The configuration approach that PUMConf implement is published in the Journal on Software and System Modeling (SoSyM 2016) [**?**].

This document details PUMConf features and provides its installation instructions. The remainder of the document is organized as follows. Section 2 presents the steps required to install and run PUMConf. Section 3 describes the features implemented as a DOORS plugin.

2 PUMConf Installation

2.1 System Requirements

PUMConf is implemented in two java components: *checker* and *PUM*. The component *checker*, checking artifact consistency, uses the Java Runtime Environment (JRE) 1.7_0.55 while the component *PUM* which generates PS models uses the Java Runtime Environment (JRE) 1.8.0_65.

PUMConf is compatible with Windows 7 and higher versions with:

- Java Runtime Environment (JRE) 1.8.0_65 and 1.7_0.55. Please verify the installed java versions in your computer (see Figure 1).

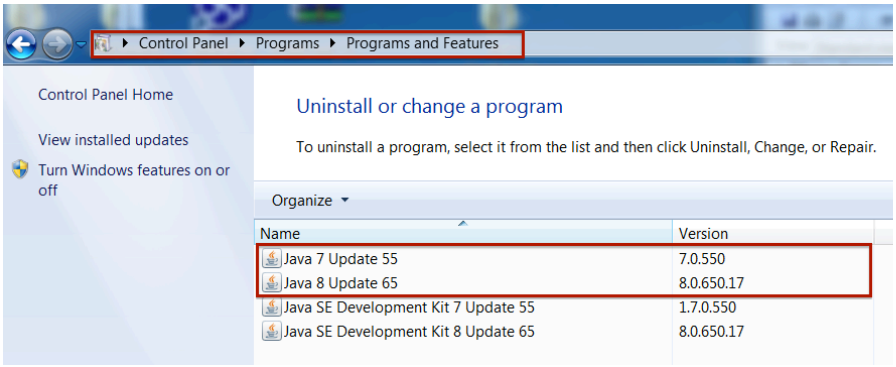

Figure 1: Required Java Versions

If you don't have these two java versions installed, please follow the following links to download them.

For JRE 1.7_0.55:
`http://www.oracle.com/technetwork/java/java-archive-downloads-javase7-521261.html`

For JRE 1.8.0_65:
`http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`

- Telelogic DOORS 8.3 for managing use case specifications. Please verify DOORS version in your computer (see Figure 2).

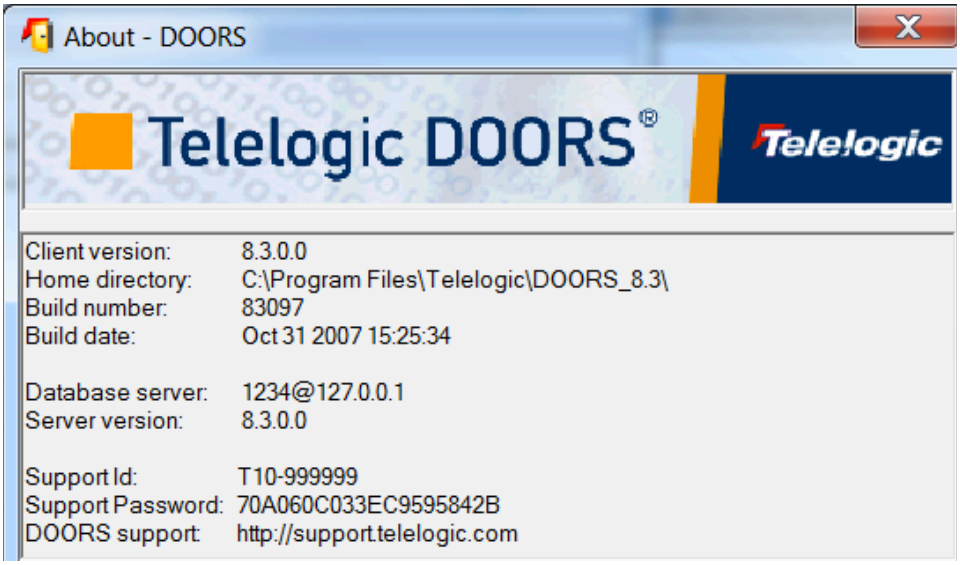

Figure 2: Required DOORS Version

- IBM Rational Rhapsody 8.0.3 for domain modeling. Please verify Rhapsody version in your computer (see Figure 3).

Figure 3: Required Rhapsody Version

- GATE Framework 7.1 release for processing use case specifications. Please verify GATE Framework version in your computer (see Figure 4).

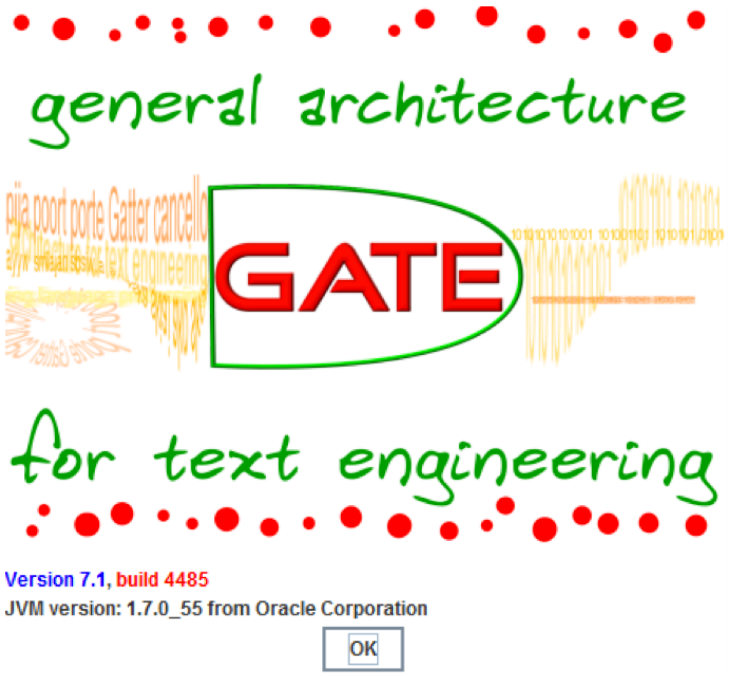

Figure 4: Required GATE Version

- R version 3.5.2 (2018-12-20) for Prioritizing test cases. Please install R in your computer (see Figure 5).

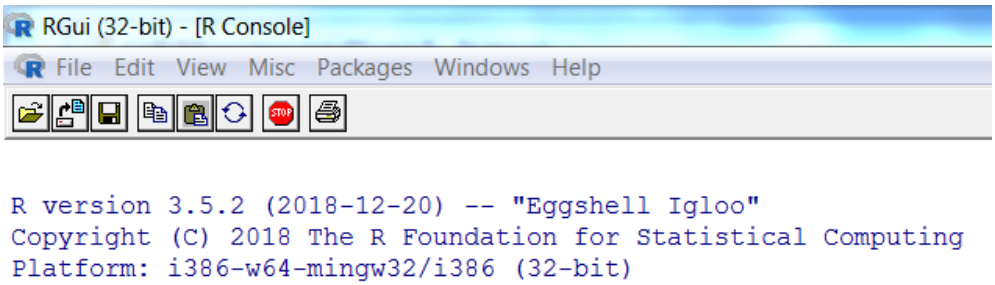

Figure 5: Required R version

If you don't have R installed, please follow the following link to download it.
`https://cran.r-project.org/bin/windows/base/`

Please make sure to install *ROCR* package in R. The function *install.packages()* in R is used to install ROCR package (see figure 6).

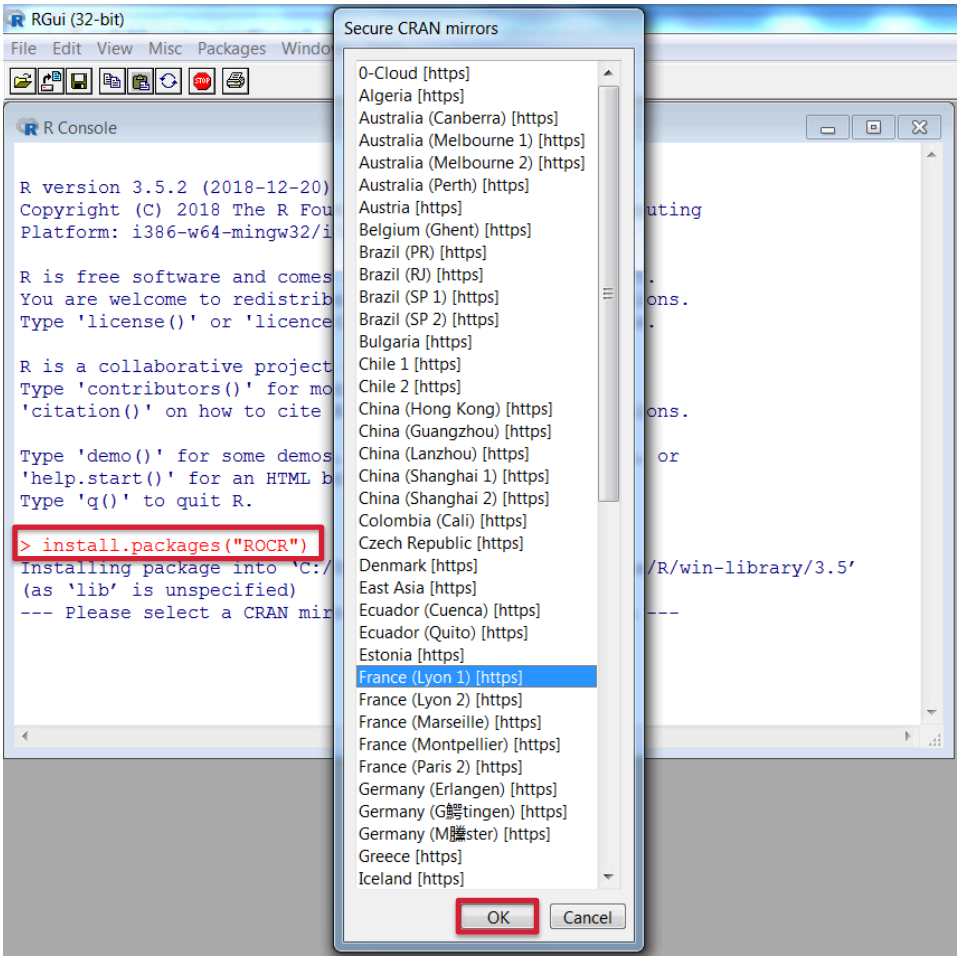

Figure 6: Installing ROCR Package in R

2.2 Installation Steps

2.2.1 Copying Folder PUM under C Directory

First, please download and unzip *PUMConf_Distribution* folder. Then, copy the folder *PUM* under your C directory as shown in Figure 7.

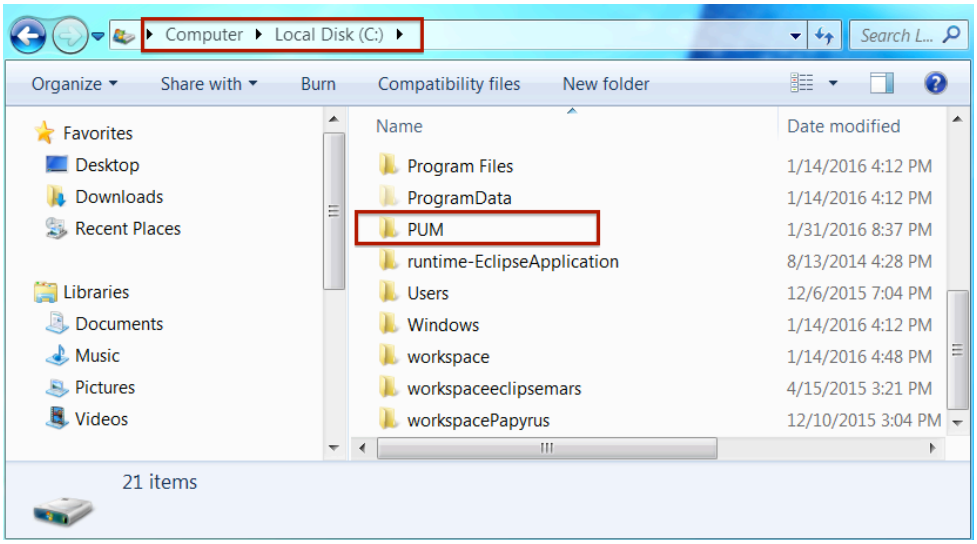

Figure 7: Folder PUM under C Directory

2.2.2 Modifying Config File

The second step consists of modifying the config file to establish connection between PUMConf and DOORS. Open the file *config.properties* located under `C:\PUM\jars`. Please update the file *config.properties* (see Figure 8). You should modify these fields:

- doorsInstallPath. For example: `C\:\\ProgramFiles\\Telelogic\\DOORS_8.3\\bin\\`
- username. Just put your DOORS username here
- password. Just put your DOORS password here
- dbPort.
- dbHost.
- modulePath. The module path is the destination path where PUMConf will save all the generated use case specifications. For example: `/STO/PSSpec/PS1`
- RScriptCommand. The RScriptCommand is used when prioritizing system test cases. For example: `C:/ProgramFiles/R/R-3.5.2/bin/Rscript`

```
#DOORS installation path for example C\:\\Program Files\\IBM\\Rational\\DOORS\\9.5\\bin\\
doorsInstallPath= [Please replace with DOORS installation path]
#DOORS password for example 987654321
password=[Please replace with DOORS password]
#DOORS server port please don't change it if your are not sure
dbPort=36677
#DOORS server host for example 127.0.0.1
dbHost=[Please replace with DOORS server host]
#DOORS user name for example Admin
username=[Please replace with DOORS user name]
#DOORS path for the generated use case specifications for example /STO/STOPSModel/test/
modulePath =[Please replace it with the desired path for generating use case specifications]
#DOORS module name for the generated use case specifications for example PsUseCases
ProductSpecificUCName =[Please replace it with PS module name]
#DOORS module description for the generated use case specifications for example PsUseCases
Description =[Please replace it with PS module name]
#Dxl script name Please don't change it
dxlFileName=test.dxl
#Dxl script name Please don't change it
dxlFileNameForRegeneration=incrementale.dxl
#Template name please don't change it
templateName=TCS-Template
#Rscript file path for example C:/Program Files/R/R-3.5.2/bin/Rscript
RScriptCommand = [Please replace it with Rscript file path]
```

Figure 8: DOORS Property File

2.2.3 Modifying Creation order File

The last step consists of modifying the *creationOrder* file. this file is used for the classification of system test cases.
Please open the file *creationOrder* located under `C:\PUM\jars`.
Please update the file *creationOrder* with the order of creation of products from the old to the recent one (see Figure 9).

```
P5==>5
P4==>4
P3==>3
P2==>2
P1==>1
```

Figure 9: Order of Creation of Products in a Product Line

2.2.4 Integrating PUMConf Plugin into DOORS

This step is installing the DOORS plugin. It has multiple intermediate steps:

1. Open the folder *DOORSPlugin* located under *PUMConf_Distribution*. Then copy the folder *ProductLine* into *Addins Folder*.
   The *Addins Folder* is located under `"%DOORSInstallFolder%"\lib\dxl\addins`.
   The keyword `"%DOORSInstallFolder%"` indicates the installation location of DOORS (for example: `C:\ProgramFiles\Telelogic\DOORS_8.3\`) (see Figure 10).

2. Copy the file *ProductLine.idx*, located in the folder *DOORSPlugin*, under `"%DOORSInstallFolder%"\lib\dxl\addins` (see Figure 11).

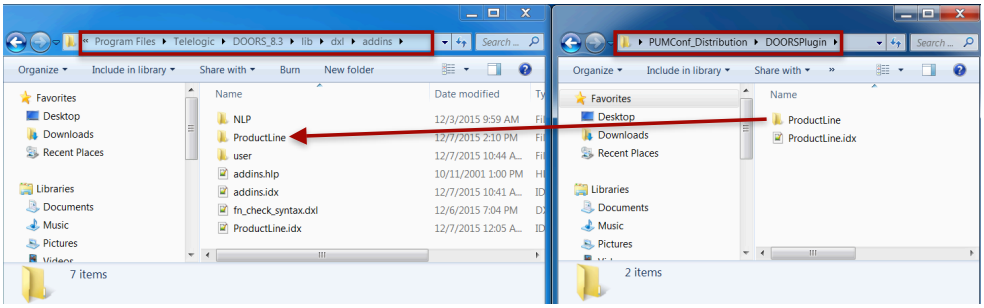

Figure 10: Window to Copy the Folder ProductLine

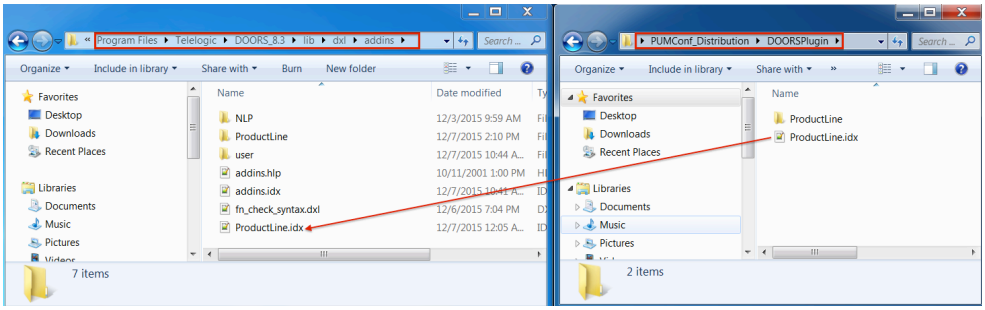

Figure 11: Window to Copy the File ProductLine.idx

3. Open a formal module in DOORS (restart DOORS). If you have followed all the steps so far, a new ProductLine menu should appear in the top menu bar (see Figure 12).

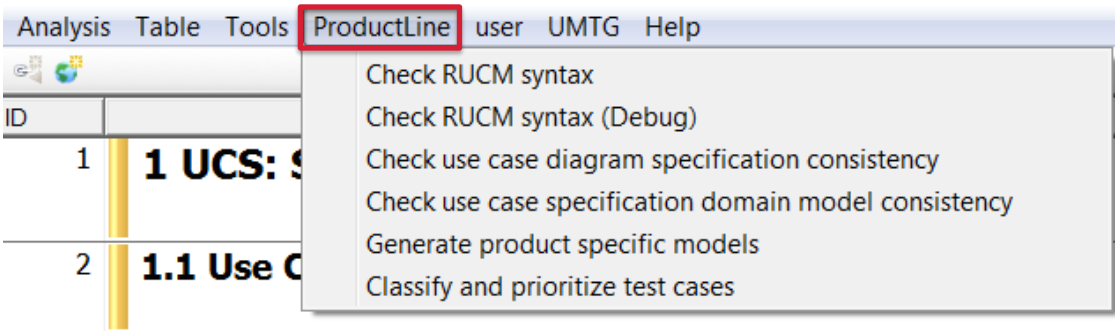

Figure 12: ProductLine Menu for DOORS

4. Import the module *TCSTemplate.dma*, located under `PUMConf_Distribution\ExampleModels\PLSpecifications`, into DOORS (see Figure 13). This template is required to generate PS use case specifications.

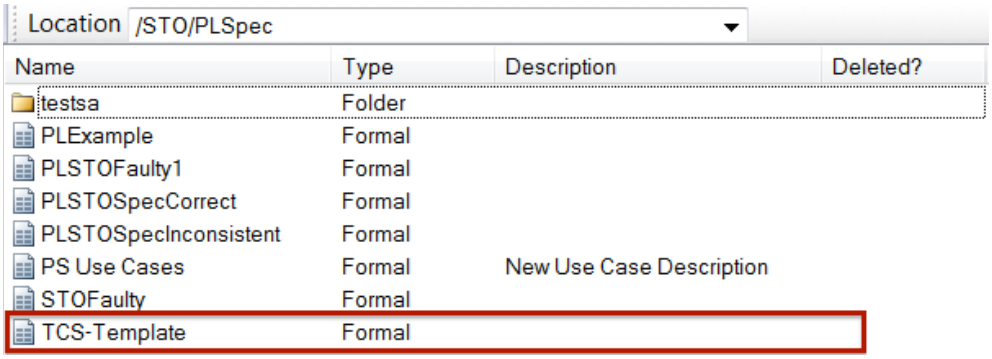

Figure 13: Window to Import Template Into DOORS

3 PUMConf Features

PUMConf plugin activates a new menu in DOORS interface named ProductLine. The menu ProductLine provides four features:

- Checking RUCM syntax
- Checking RUCM syntax (Debug)
- Checking use case diagram specification - consistency
- Generating product specific use case and domain models
- Classifying and prioritizing system test cases in a product family

These functions are described next.

3.1 Checking RUCM Syntax

The feature *Check RUCM syntax* relies on Natural Language Processing (NLP). It automatically reports inconsistencies between Product Line (PL) use case specifications, specified in the current DOORS module, and the PL RUCM template. In case of inconsistencies, PUMConf lists the sentences where the inconsistencies occur with their *object_id* in DOORS. The analyst can click on the link of the *object_id*. This link points directly on the inconsistency in DOORS thus the the analyst can easily update the use case specifications until they conform to the extended RUCM template. Below you can find the screenshots illustrating this functionality.

Figure 14 presents the menu item in DOORS that activates the functionality of checking RUCM syntax.

Figure 15 points the command window that starts GATE Framework used for checking RUCM syntax. This window is automatically closed when PUMConf lists the inconsistencies (Figure 16).

Figure 16 lists one example inconsistency between the use case specifications and the RUCM template. This inconsistency is about the wrong use of the *EXTEND* keyword with a variation point in use case specifications.

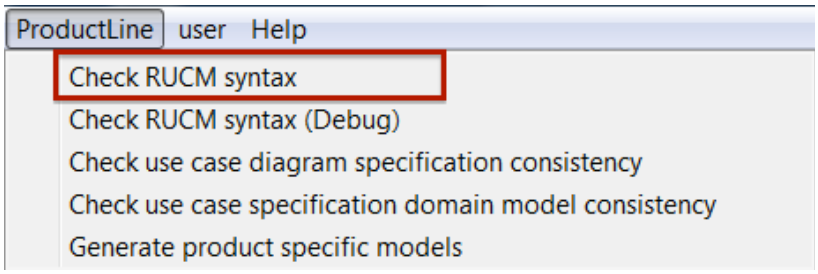

Figure 14: Menu Item in DOORS for Checking Consistency

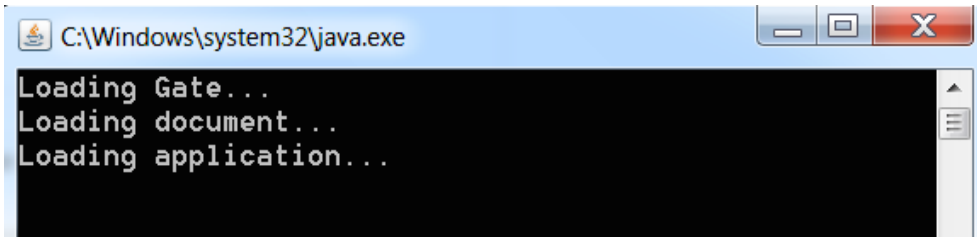

Figure 15: Command Window to Automatically Start GATE

3.2 Checking RUCM Syntax (Debug)

This feature provides analysts detailed information (e.g., invalid variation points, invalid alternative flows, and invalid if condition) about the use cases which do not conform to the PL RUCM template. This information is given in terms of annotations in use cases provided by GATE workbench.

Figure 17 depicts the menu item in DOORS that activates the functionality of checking RUCM syntax with debug mode.

Figure 18 points the command window that starts GATE to process use case specifications.

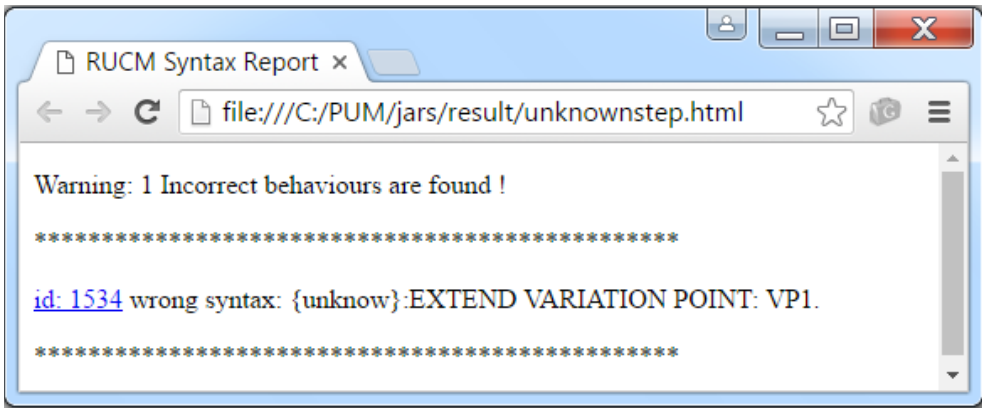

Figure 16: Window for the List of Inconsistencies

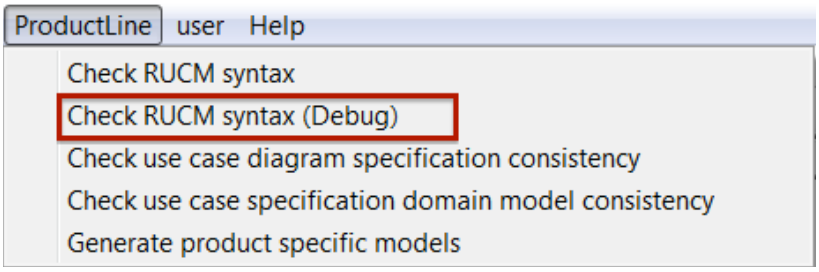

Figure 17: Menu Item in DOORS for Checking RUCM Syntax (Debug)

Figure 19 indicates the window with the annotated use cases in GATE.

3.3 Checking Use Case Diagram - Specification Consistency

The consistency of PL use case diagram and specifications is checked using this feature, and inconsistencies are reported automatically if there are any. For instance, a variation point in the use case diagram might be missing in the corresponding use case specification.

Figure 20 depicts the menu item in DOORS that activates the functionality of checking use case diagram specification consistency.

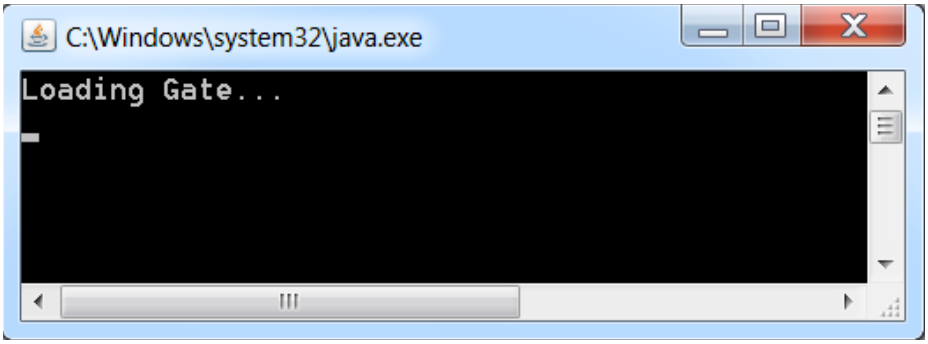

Figure 18: Command Window to Automatically Start GATE

The first step is to import the PL use case diagram which is in UML format as shown in Figure21.

Once the use case diagram is imported, PUMConf processes the use case specifications and the use case diagram to automatically report inconsistencies. Figure 22 presents the window displayed after checking consistency.

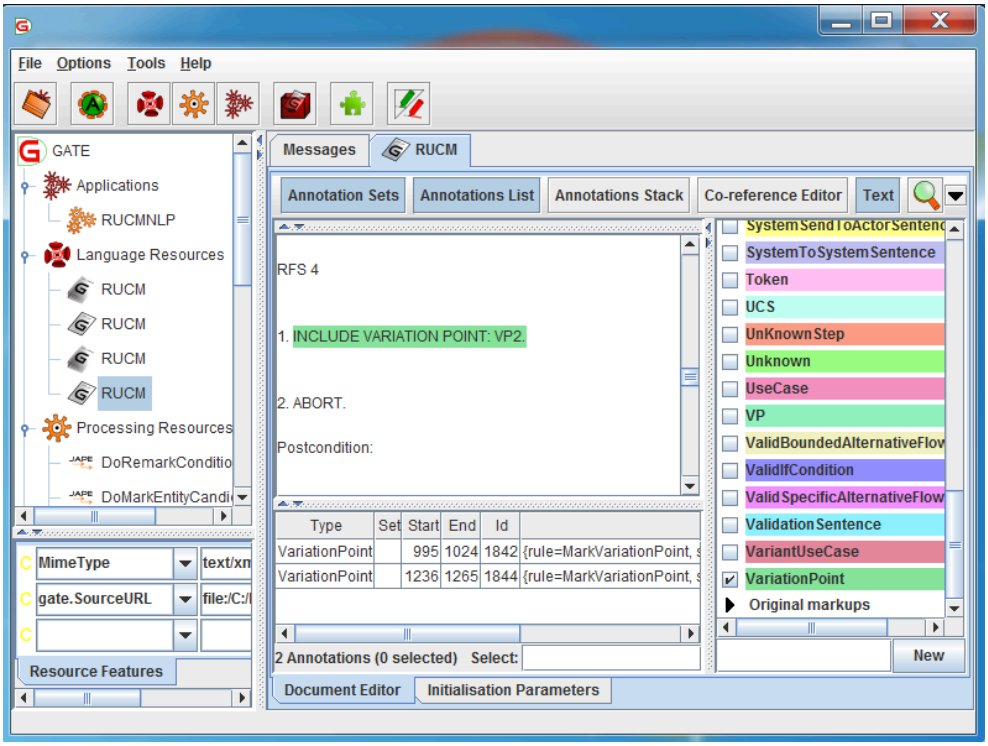

Figure 19: GATE Window for Annotated Use Cases

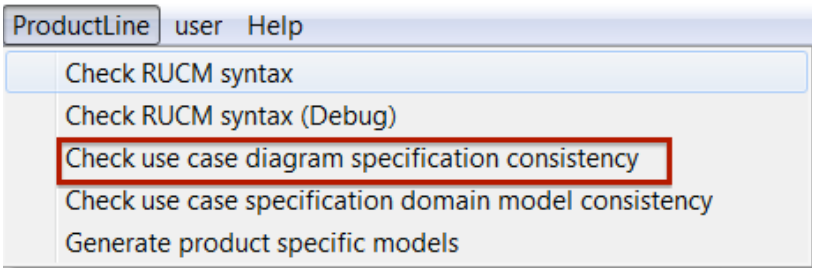

Figure 20: Menu Item in DOORS for Checking Use Case Diagram Specification Consistency

By clicking to the OK button in Figure 22, the inconsistency report is automatically displayed as shown in Figure 23. The inconsistency described in Figure 23 consist of a variation point present in the use case diagram while missing in the corresponding specification.

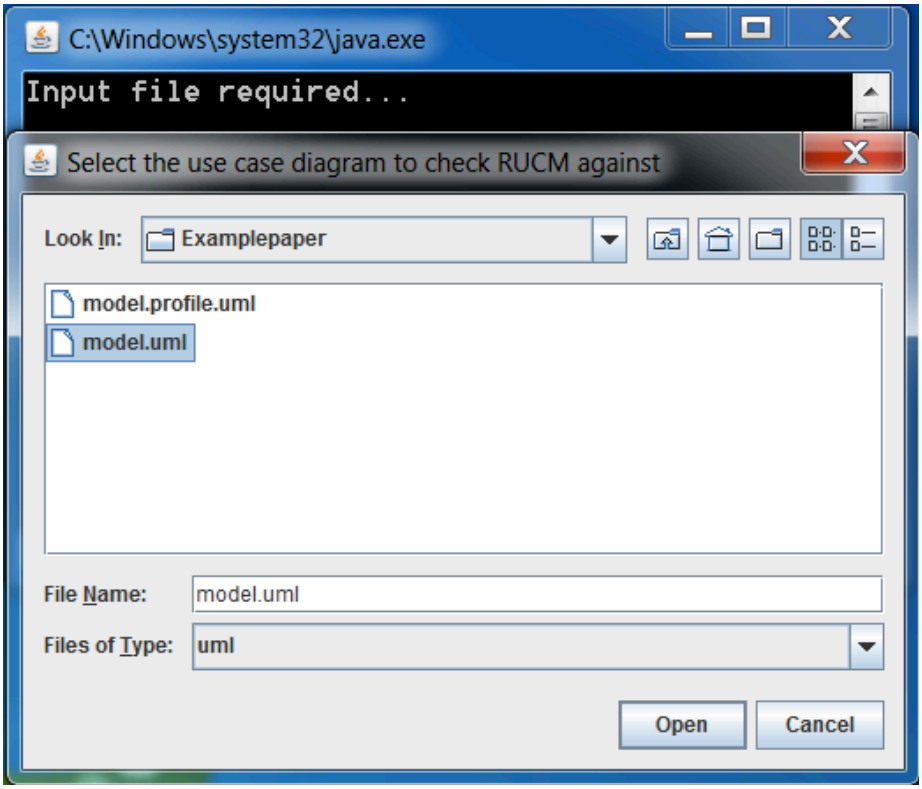

Figure 21: Window to Import UML File for Use Case Diagram

3.4 Generating Product Specific Use Case and Domain Models

This feature automatically configures PS use case and domain models based on the configuration decisions made by the user. Figure 24 presents the menu item in DOORS that activates the functionality of generating PS use case and domain models.

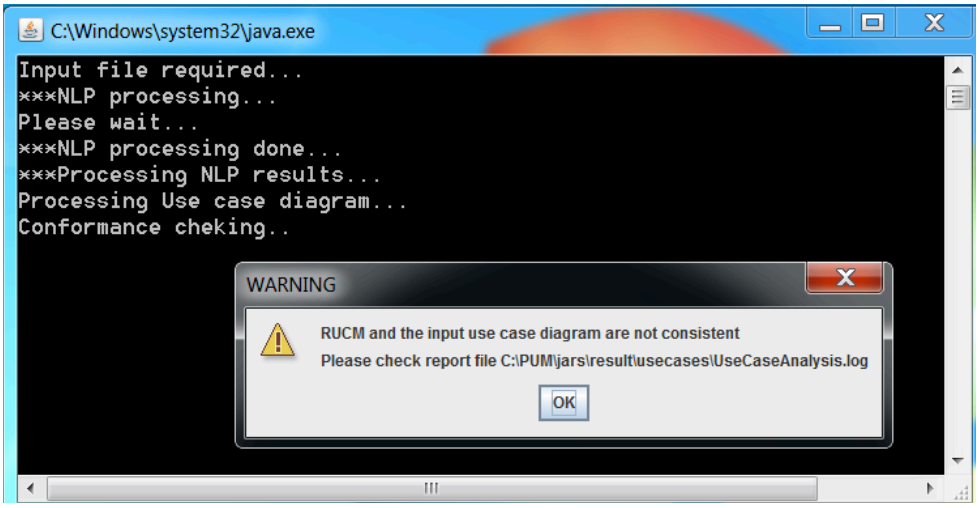

Figure 22: Consistency Checking Result

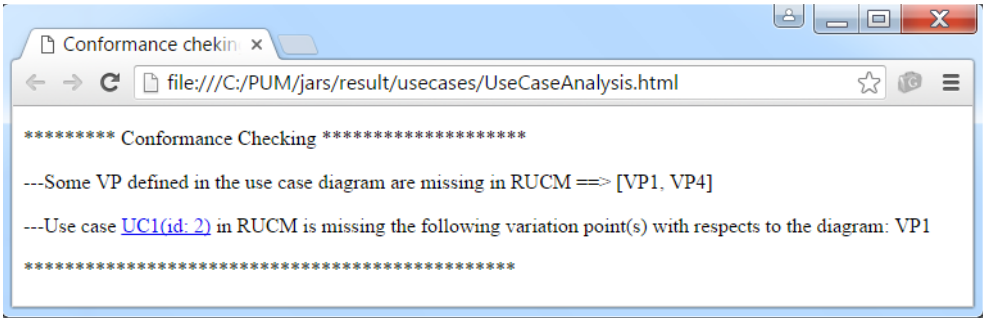

Figure 23: Inconsistency Report

First the analyst is asked to import the PL use case diagram (see Figure 25).

PUMConf first displays all the variation points, their including use cases and their types (i.e., optional or mandatory) (see Figure 26).

The partial order of decisions to be made is automatically identified from the dependencies among variation points and variant use cases. By clicking on Start Configuration in Figure 26, the list of variation points to start with is displayed (see Figure 27).

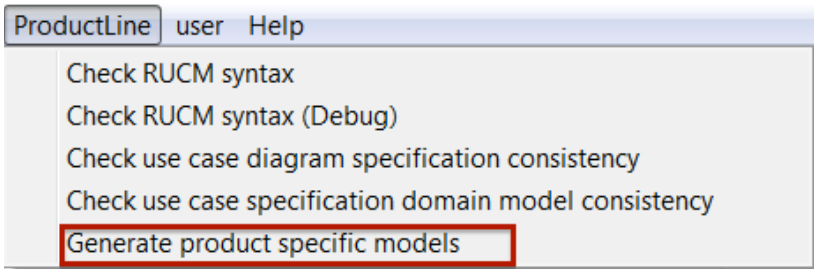

Figure 24: Menu Item in DOORS for Configuring PS Use Case and Domain Models

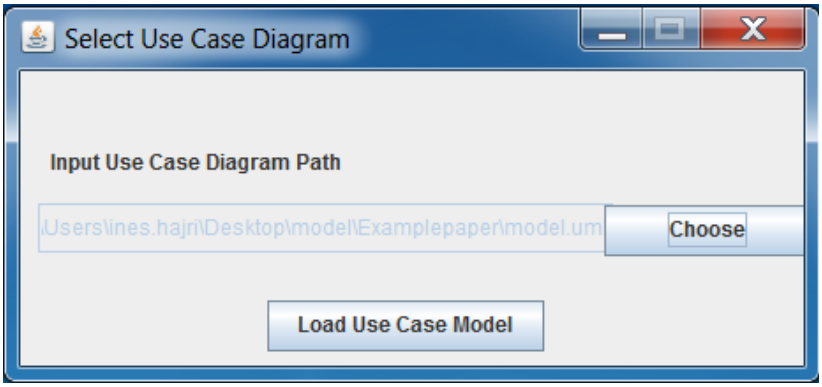

Figure 25: Window to Import PL Use Case Diagram

The analyst makes a decision for each variation point in the list as shown in Figure 28.

PUMConf checks the consistency of the decision with prior decisions. If there is any contradicting decision, the analyst is asked to update the current and/or previous decisions causing the contradiction. An example of contradiction is shown in Figure 29.

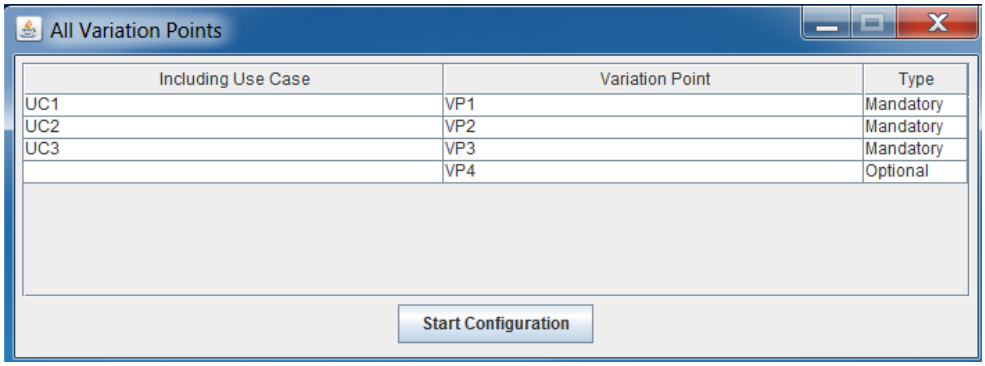

Figure 26: List of Variation Points

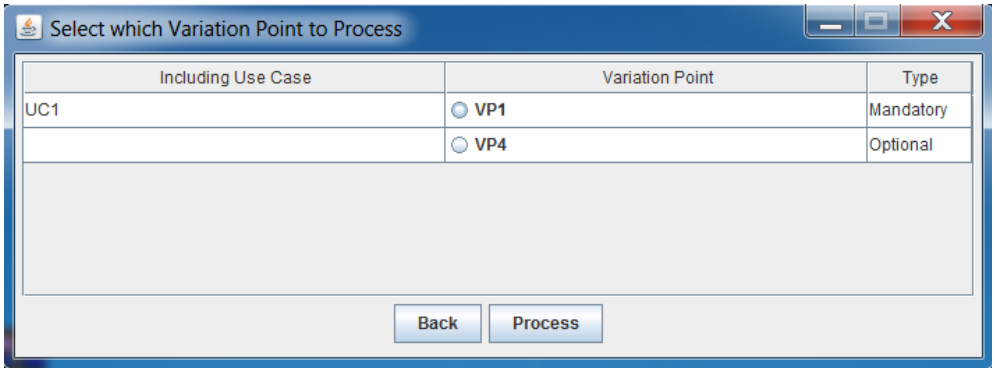

Figure 27: List of Variation Points to Start With

A decision may cause further decisions to be made for some other variation points, i.e., included by the variant use cases selected in the decision. The tool automatically updates the list of variation points to be decided as shown in Figure 30.

When all variation points are treated, the analyst can proceed to the generation of PS use case diagram. The PS diagram is generated in UML format. The user may import the generated model in any EMF-based modeling environment (eclipse.org/modeling/emf/) that supports UML. An example of such a modeling environment is Papyrus (eclipse.org/papyrus/).

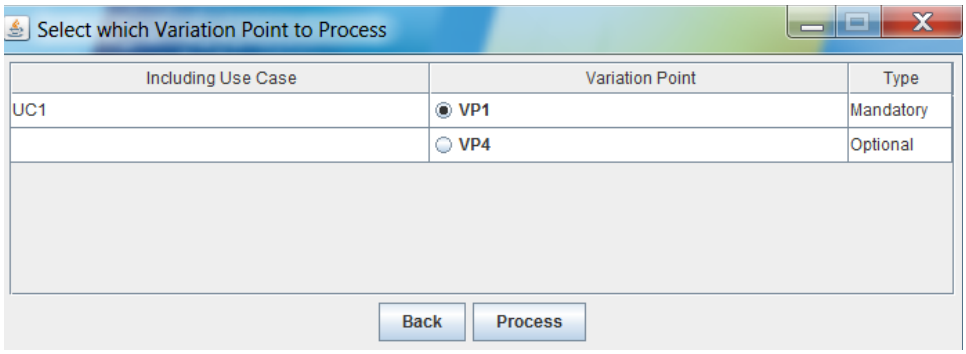

Figure 28: Window to Select Variation Points

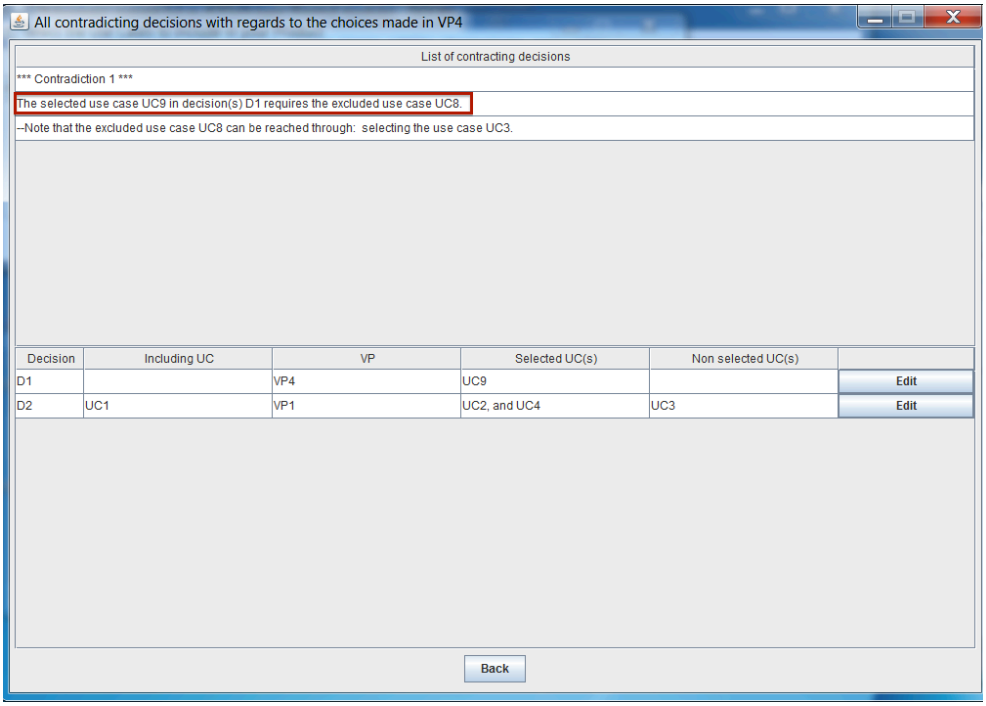

Figure 29: Example of Contradicting Decisions

After all decisions for the PL diagram are made, the analyst proceeds with the PL use case specifications (see Figure 31). The analyst makes a decision for each optional step (see Figure32) and optional alternative flow if there are any in the specification. The tool asks then the order of the variant order steps (see Figure 33).

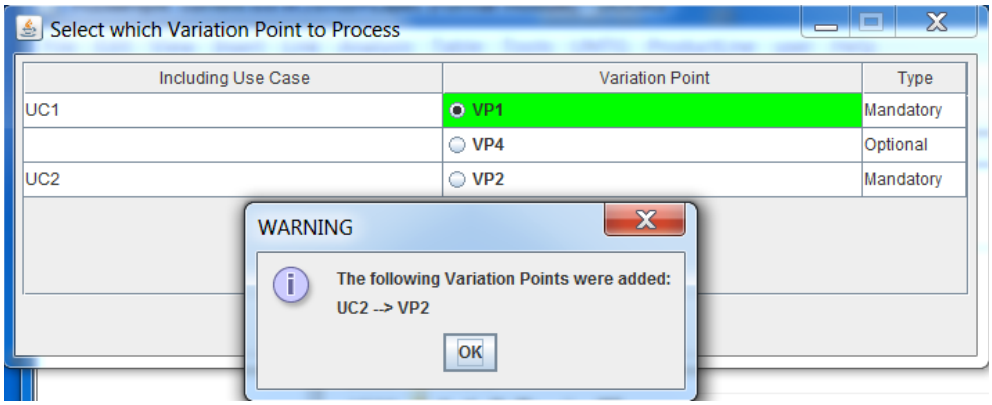

Figure 30: List of Variation Points Updated

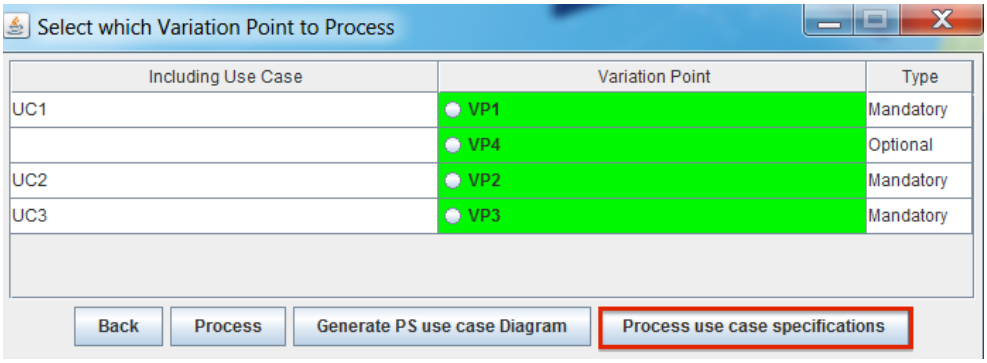

Figure 31: Window to Process Use Case Specifications

After making all decisions for the use case specifications, the analyst can generate the PS use case specifications. The PS use case specifications are generated as a DOORS module (See Figure 34).

Lastly the analyst proceeds with decisions for the domain model (see Figure 35). The first step is to load the domain model in XMI format (see Figure 36).

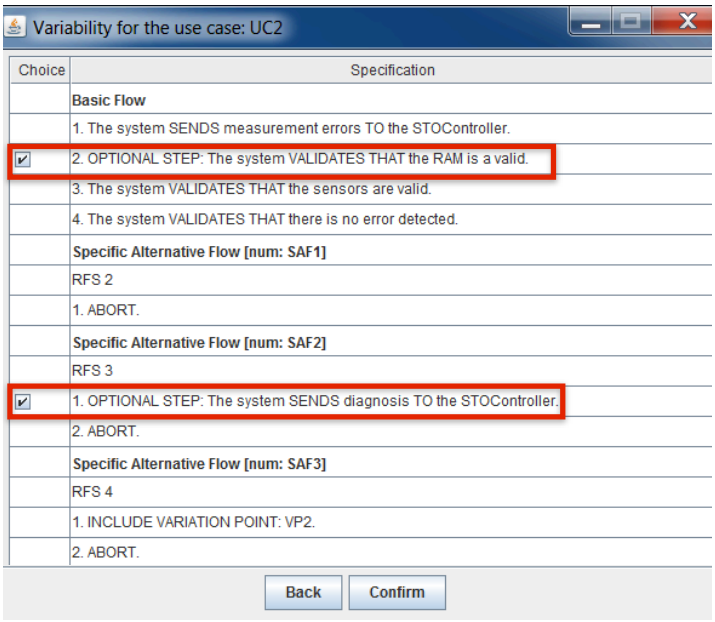

Figure 32: Window to Make Decisions for Optional Steps

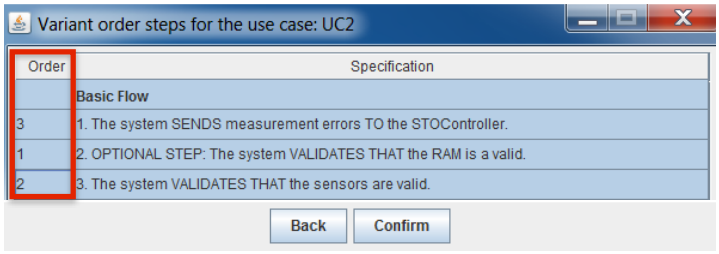

Figure 33: Window to Make Decisions for Variant Order Steps

The user is asked to input her decisions for optional entities (see Figure 37), then for variant entities (see Figure 38).

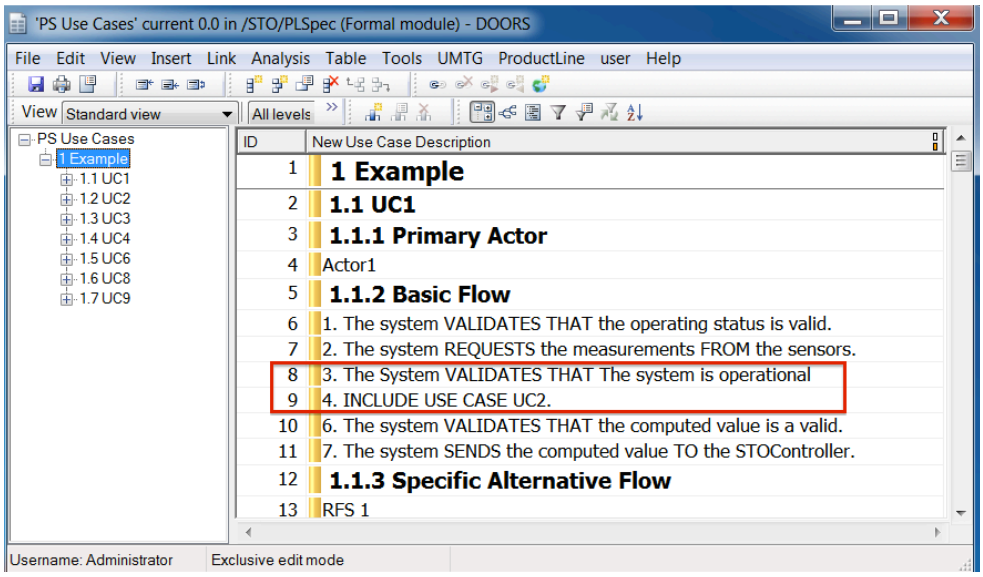

Figure 34: Generated PS Use Case Specifications

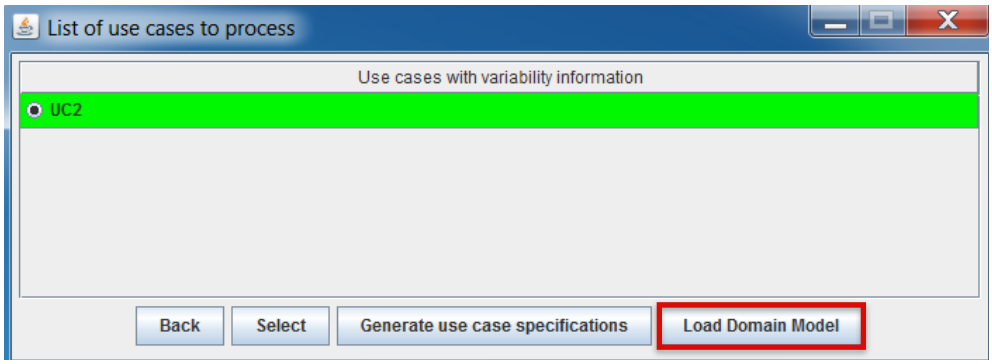

Figure 35: Window to Process the Domain Model

After all decisions for the domain model are made, the analyst can generate the domain model in XMI format. The generated XMI model can be imported in a given modeling environment such as Rhapsody.

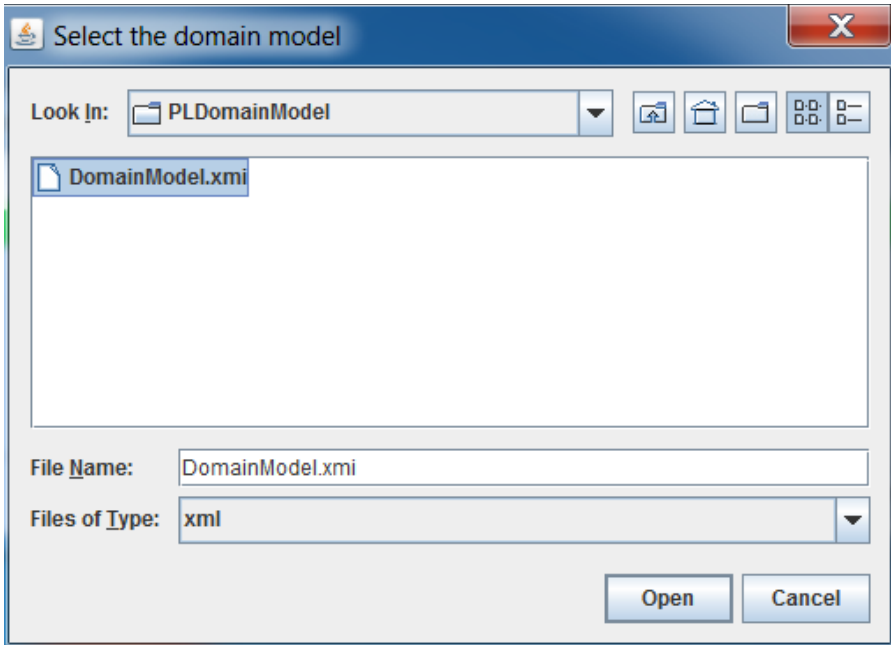

Figure 36: Window to Load the Domain Model in XMI Format

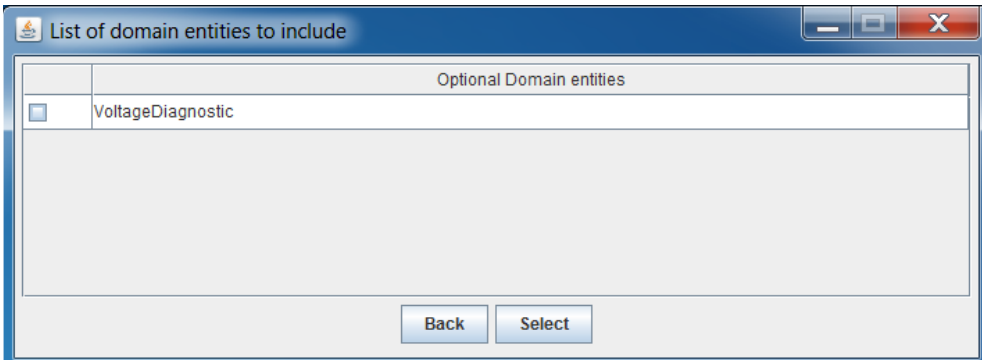

Figure 37: Window to Make Decisions for Optional Entities

3.5 Identifying the Change Impact for Evolving Configuration Decisions in PL Use Case Diagrams

Changes on configuration decisions for the PL use case diagram have an impact on prior decisions as well as on subsequent decisions to be made. Therefore, our impact analysis approach checks (i) if the change causes any contradiction with prior decisions, (ii) if there is any prior decision which becomes invalid, (iii) if any additional variation point should be considered for decision-making, and (iv) if any subsequent decision should be restricted to ensure the consistency of decisions.

Next, we provide an example illustrating the impact of decision changes on prior and subsequent decisions. Fig 39 shows the PL use case diagram for this example.

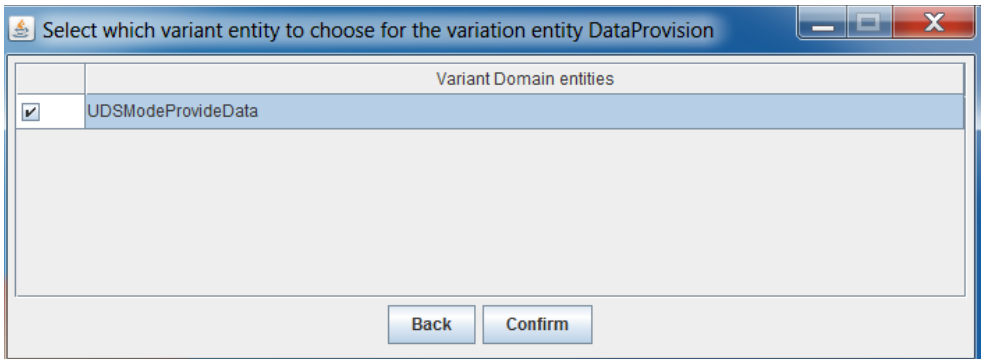

Figure 38: Window to Make Decisions for Variant Entities

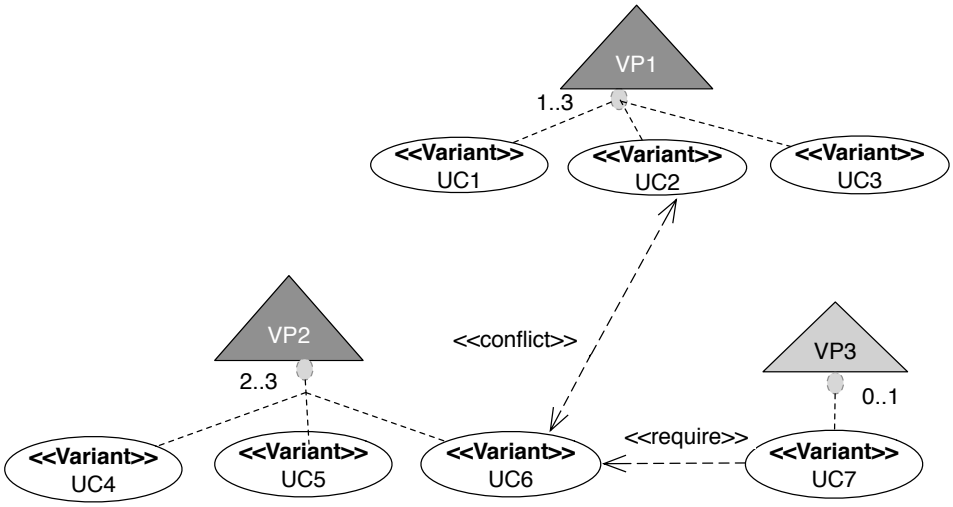

Figure 39: PL Use Case Diagram for the Example Model

Please start PUMConf and import the uml file for the example model saved under `ExampleModels\CIA\Example` (see Figure 40).

We first start by making the decision *d1* for the variation point *VP1*. We select the use cases *UC1* and *UC3* (see Figure 41).

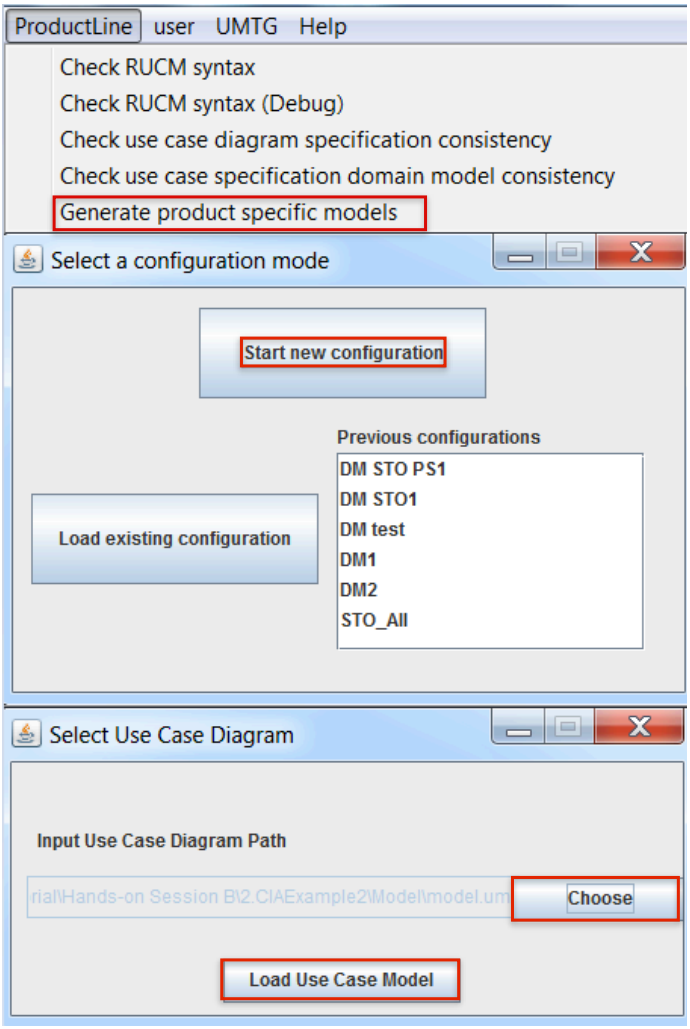

Figure 40: Load PL Use Case Diagram for the Example Model

Then, we make the decision *d2* for the variation point *VP3* by selecting the use case *UC7* (see Figure 42).

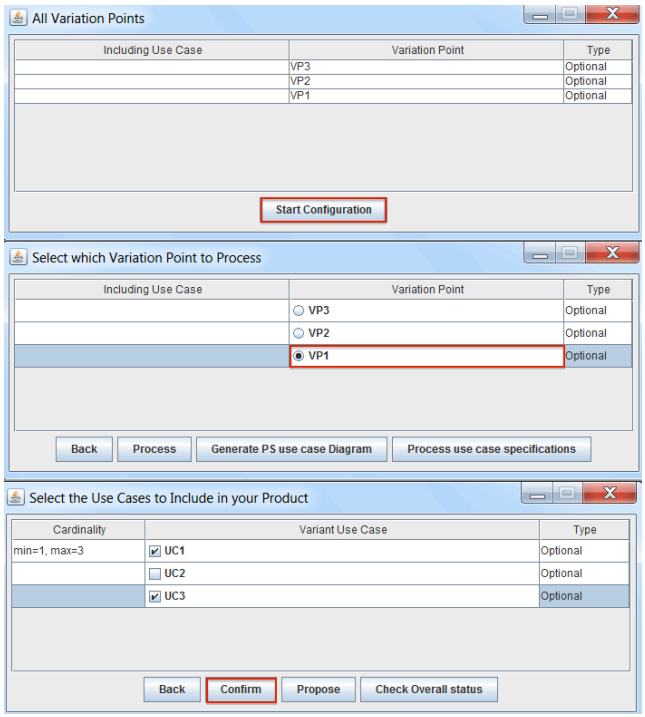

Figure 41: Making Decision for VP1

Finally, we decide to change the decision *d1* for *VP1* by selecting *UC2* instead of *UC3*. Before confirming our decision, we propose the decision change to determine the change impact on other diagram decisions (see Figure 43).

Once we propose the change, PUMConf automatically identifies the impact of the diagram decision changes on prior and subsequent diagram decisions. Our tool provides an impact report documenting the impacted decisions with the reason of the impact. Figure 44 shows the impact report for the example change of *d1*.

Due to the change of the decision *d1*, we have two other impacted decisions. The subsequent decision for *VP2* and the prior decision for *VP3*. *UC6* in *VP2* is impacted and colored in *yellow* meaning that in *UC6* we have two contradicting restrictions (explained in the legend of the figure). After the change, the selected use case *UC2* conflicts with *UC6* which is at the same time required by the previously selected *UC7* in *VP3*. Thus, the prior decision *VP3* is also impacted and *UC7* is colored in *pink* meaning that due to the changed decision, *UC7* should be unselected in order to avoid contradicting decisions.

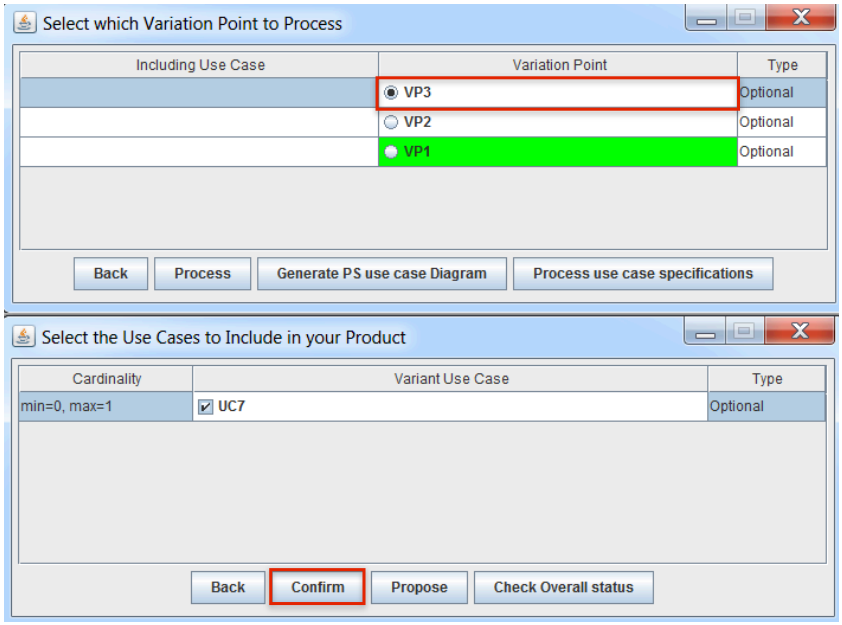

Figure 42: Making Decision for VP3

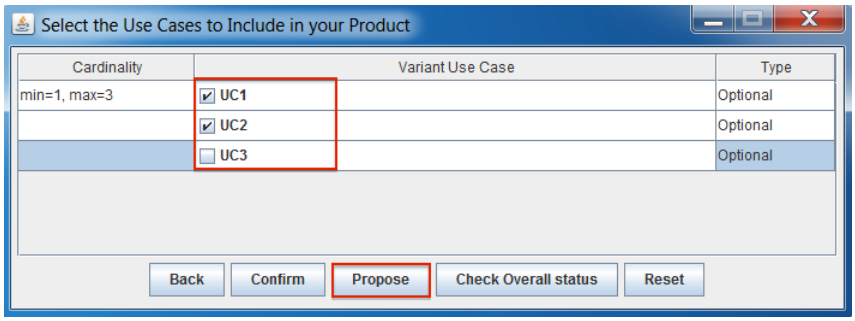

Figure 43: Changing the Decision for VP1

3.6 Incrementally Reconfiguring Product Specific Use Case Models for Evolving Configuration Decisions

The configuration decisions are captured in a decision model. Once all the decisions are made and the PS models are configured, the analyst is asked to save his decisions. (see Figure 45)

Please click on *Save Decision Model*. The PS use case models are successfully saved (see Figure 46). The saved configuration should appear in the *previous configurations* list (see Figure 47).

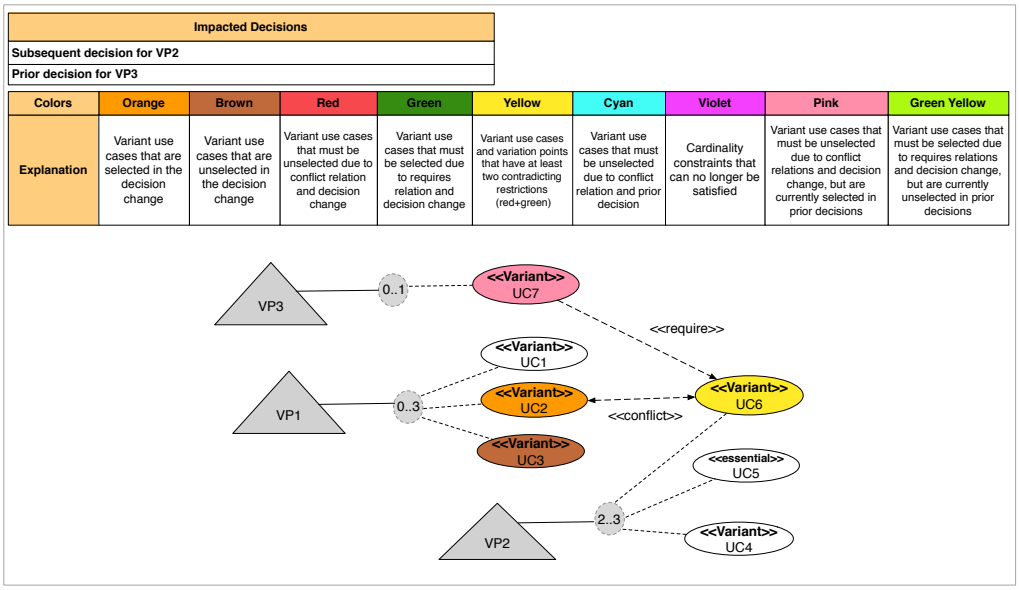

Figure 44: PUMConf's User Interface for Displaying the Impact of the Change Decision *d1* on Other Diagram Decisions

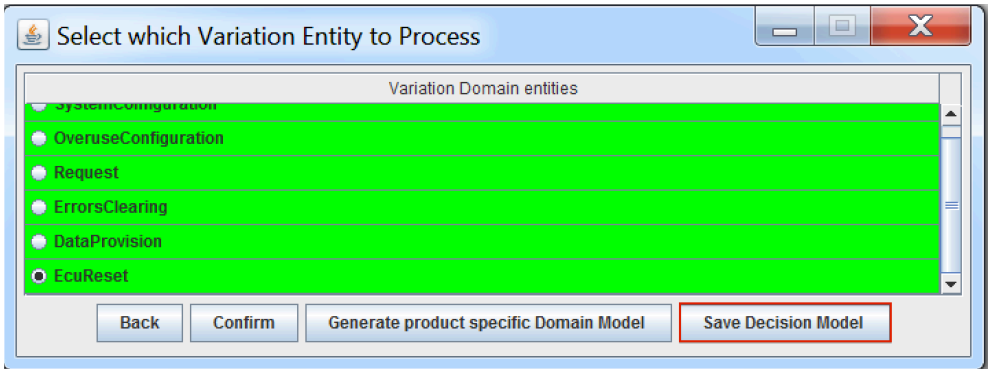

Figure 45: Window to Save the Decision Model

Configuration decisions may evolve in the PL use case models. PS use case models are incrementally reconfigured by focusing only on the changed decisions. The first step is to load the PS models that need to be reconfigured. (see Figure 47)

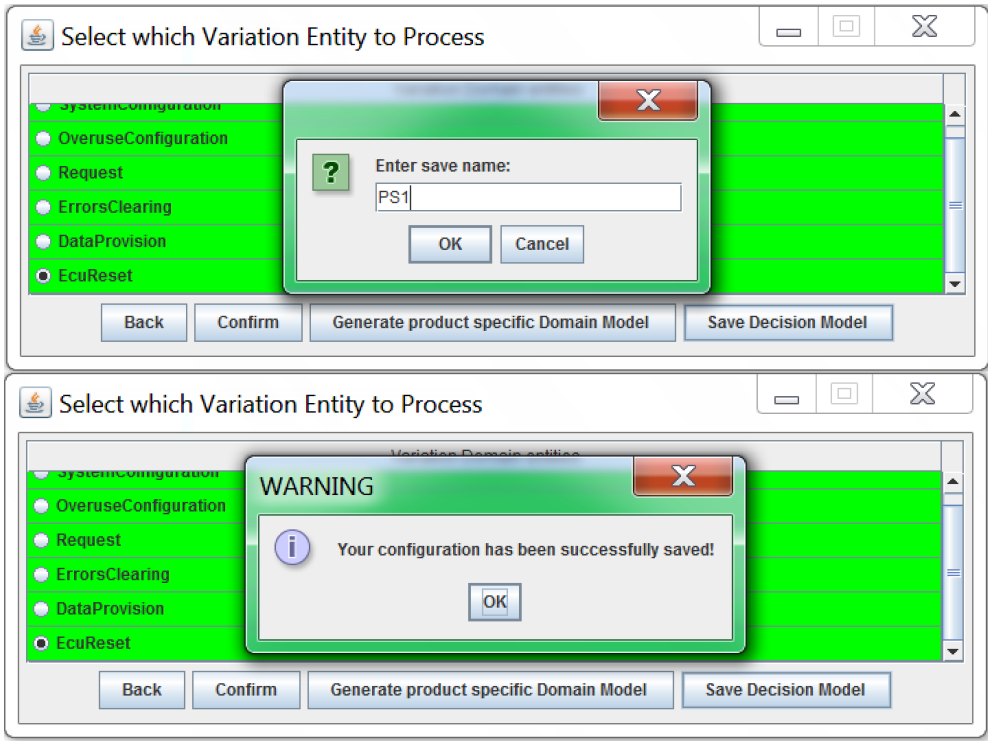

Figure 46: Window for Succefully Saving the Decision Model

Whenever we load an existing configuration, all decisions already made for a given PS model are loaded (see Figure 48). In case that a PS use case diagram decisions evolve, the analyst is asked to update his decision.

Please upload the Project *Models* in IBM DOORS. The project is saved under `PUMConf_Distribution\ExampleModelsforReconfiguration`. This project contains the PL use case specifications example, the generated use case specifications *PS1* and the links between them.

Please load *PS1* from the *previous configurations* list, then update a diagram decision by unselecting the use use case *UC4* in the variation point *VP1*. (see Figure 48)

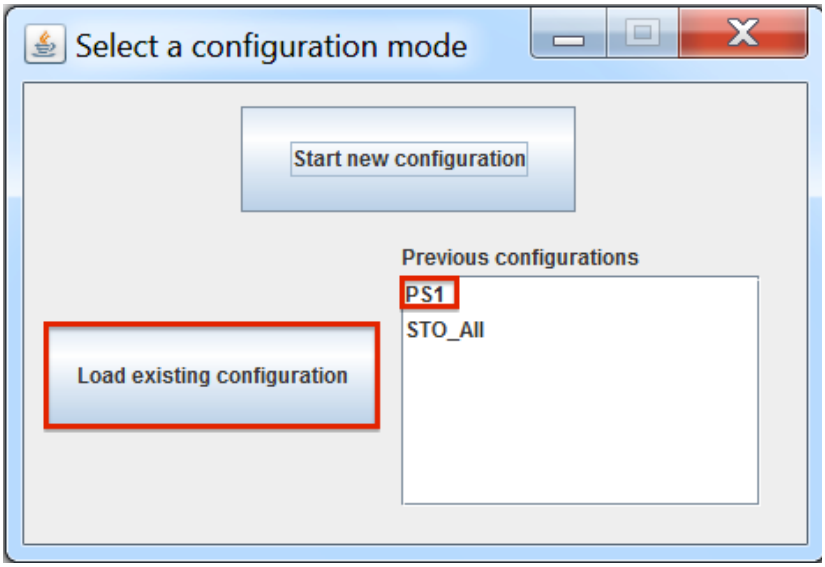

Figure 47: Window to load the Saved Configuration

In addition to the evolution of use case diagram decisions, the decisions for the use case specifications may evolve too. The analyst is asked to update a specification decision. Please click on *processes the use case specification* and select the use case *UC2* in order to update the use case specification decision. (see Figure 49)

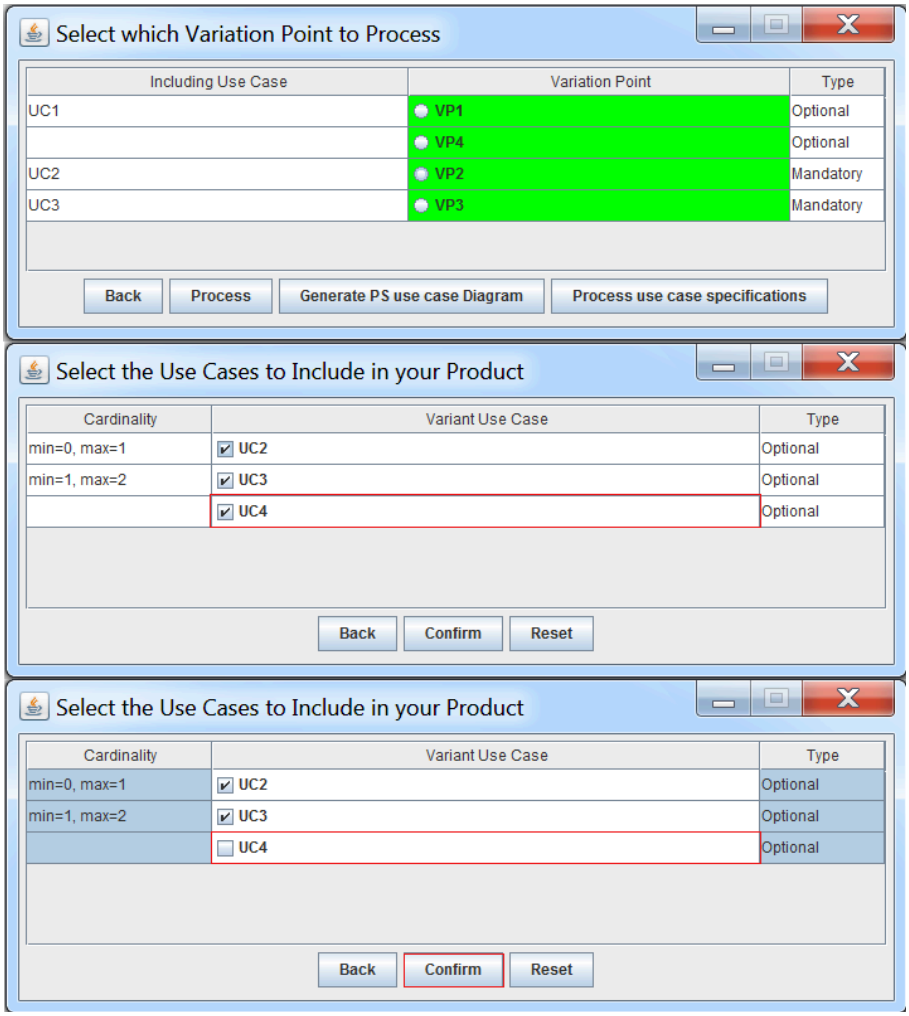

Figure 48: Window to Update a Use Case Diagram Decision

During the initial configuration of *PS1* we selected the two optional steps in the basic flow of *UC2* (see Figure 50).

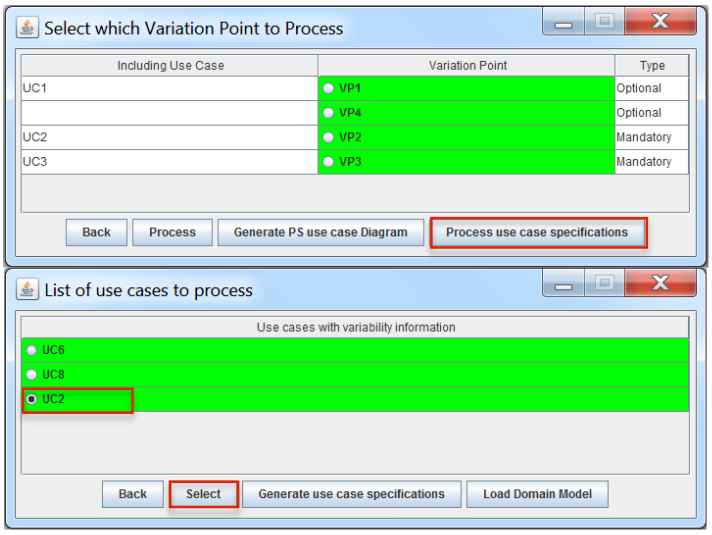

Figure 49: Window to Process a Use Case Specifications

Please update the decision for *UC2* by unselecting the first optional step (see Figure 51).

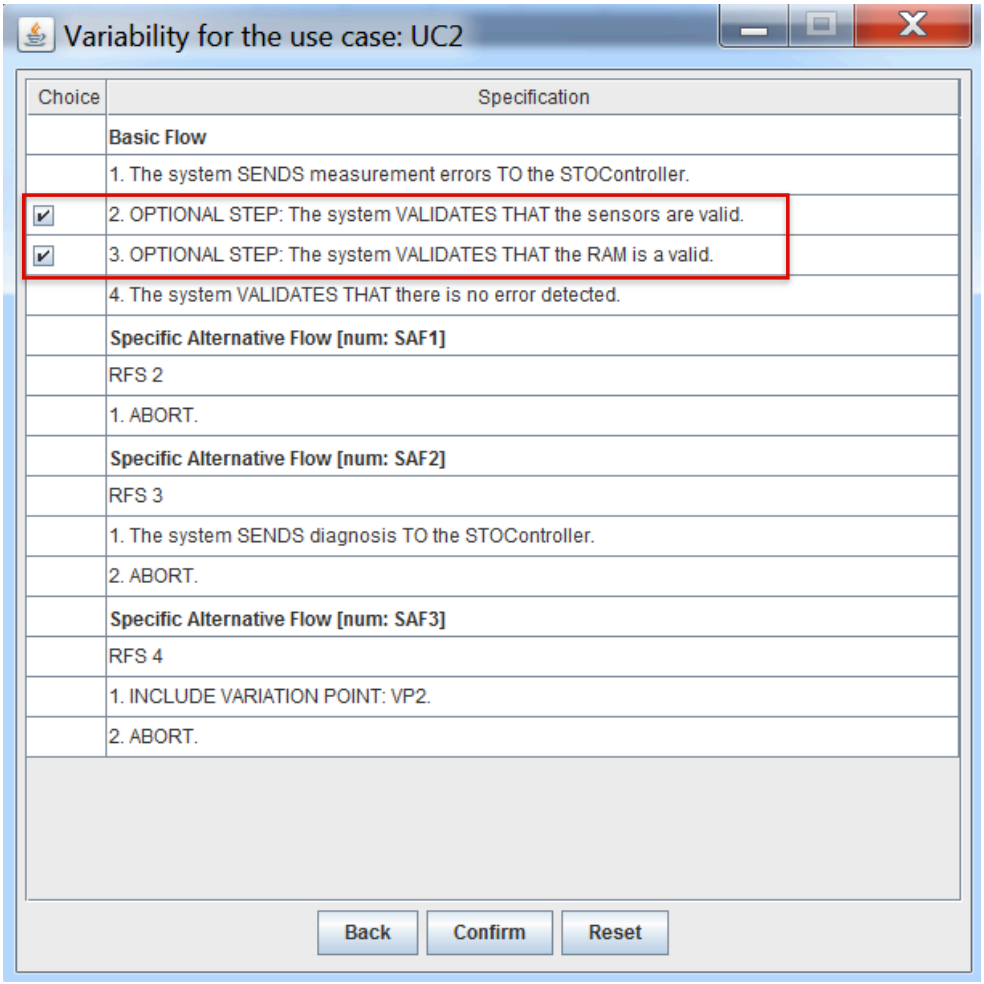

Figure 50: Window for the Initial Configuration Decision

Similarly, in case of the evolution of the domain model decision, the analyst is asked to update his decision (see Figure 52). Please click on *Load Domain Model*, then unsselect the optional entity *VoltageDiagnostic* and the variant domain entity *UDSModeOveruseConfiguration* which are selected initially.

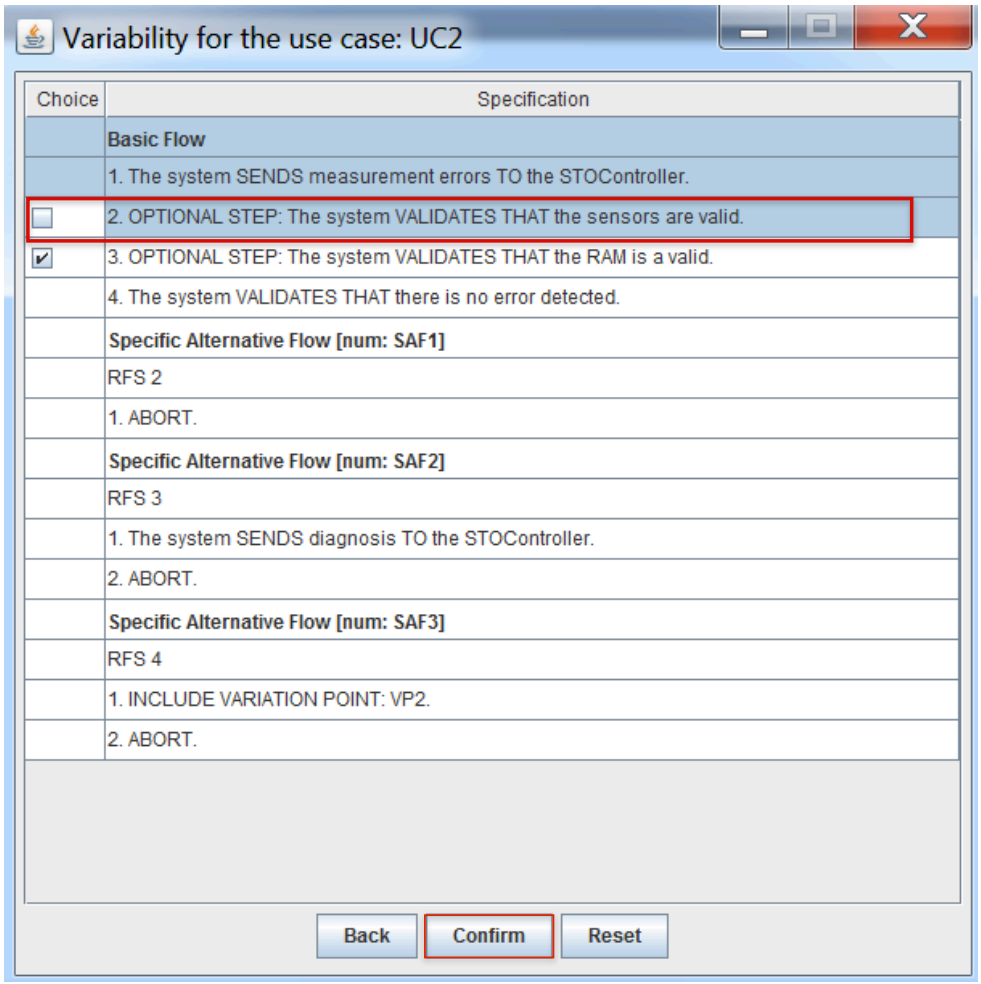

Figure 51: Window to Update a Use Case Specification Decision

Once all the required changes are considered, the analyst can save the new configuration. The tool offers two possibilities (1) overwriting the existing configuration by entering the same configuration name or (2) saving a new configuration.

Please click on *Save Decision Model*, enter the same configuration name and save the new decisions (see Figure 53).

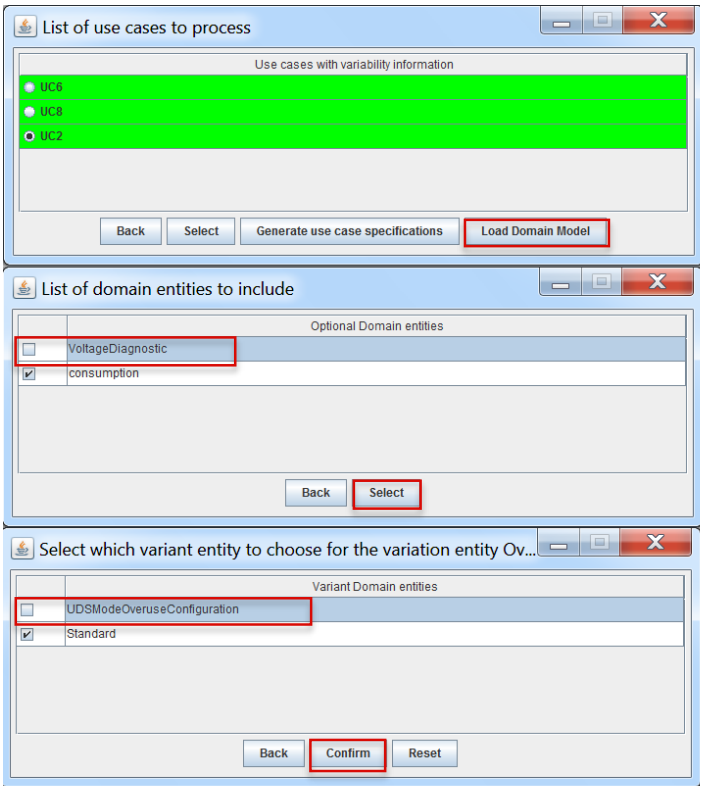

Figure 52: Window to Update a Domain Model Decision

Before saving the new configuration, the tool displays an impact report with all the changes in the use case diagram, specifications and the domain model (see Figure 54).

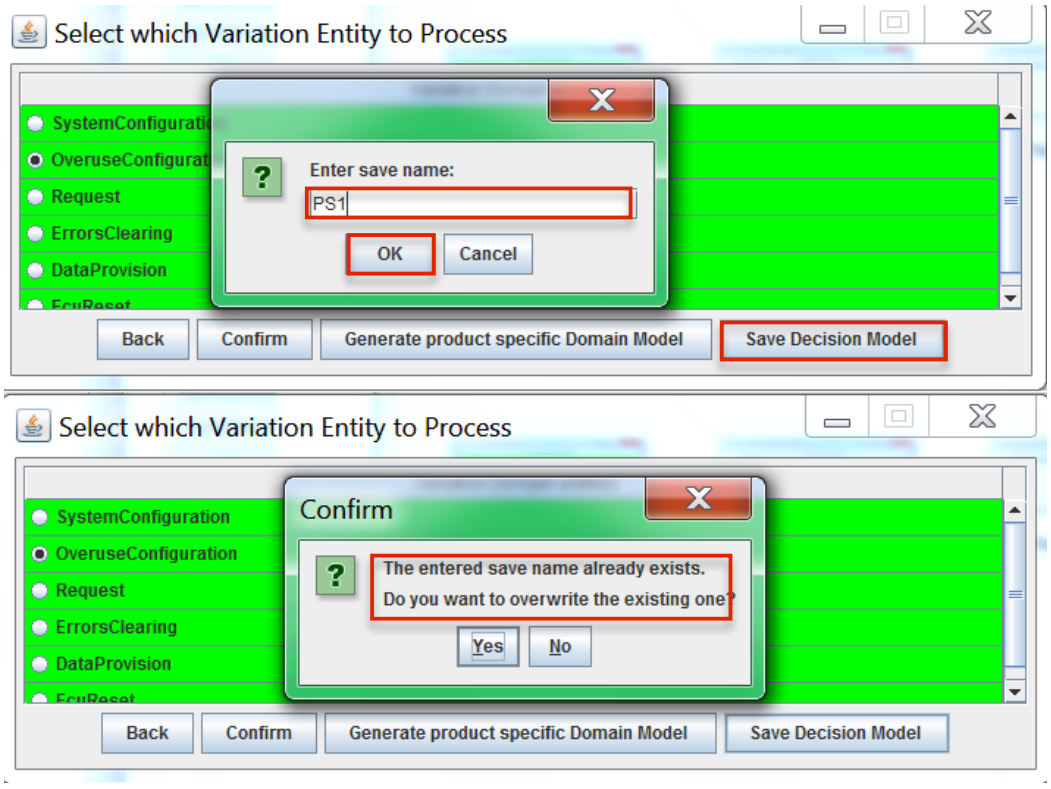

Figure 53: Window to Save the Updated Configuration

The analyst consult the change report and decide whether to apply the reported changes or not (see Figure 55)

- Use case diagram

ID		Old status	New status	Variant	Old status	New status
Including use case	Variation point					
UC1	VP1	☑	☑	UC4	☑	☐

- Domain model (optional entities)

Optional entity	Old status	New status
VoltageDiagnostic	☑	☐

- Use Case Specifications

Use Case	Old Status	New Status	Flow	Old status	New status	Old Steps	New Steps
UC2	☑	☑	**Basic Flow**	☑	☑	1. The system SENDS measurement errors TO the STOController.	1. The system SENDS measurement errors TO the STOController.
						2. OPTIONAL STEP: The system VALIDATES THAT the sensors are valid.	2. OPTIONAL STEP: The system VALIDATES THAT the RAM is a valid.
						3. OPTIONAL STEP: The system VALIDATES THAT the RAM is a valid.	3. The system VALIDATES THAT there is no error detected.
						4. The system VALIDATES THAT there is no error detected.	
			Specific Alternative Flow [num: SAF1]	☑	☐	RFS 2	
						1. ABORT.	
			Specific Alternative Flow [num: SAF2]	☑	☑	RFS 3	RFS 2
						1. The system SENDS diagnosis TO the STOController.	1. The system SENDS diagnosis TO the STOController.
						2. ABORT.	2. ABORT.
			Specific Alternative Flow [num: SAF3]	☑	☑	RFS 4	RFS 3
						1. INCLUDE VARIATION POINT: VP2.	1. INCLUDE VARIATION POINT: VP2.
						2. ABORT.	2. ABORT.

Figure 54: Window for the Change Impact Report

After all the changes are calculated, the PS use case models are incrementally reconfigured. Only the parts affected by the changed decisions are automatically regenerated. (see Figure 56).

In Figure 56, the PS use case specification is incrementally reconfigured,i.e., the unselected optional step in *UC2* is removed as well as the alternative flow referring to this step (2) the manually assigned trace link in the PS use case specifications is not deleted and not affected by the decisions change .

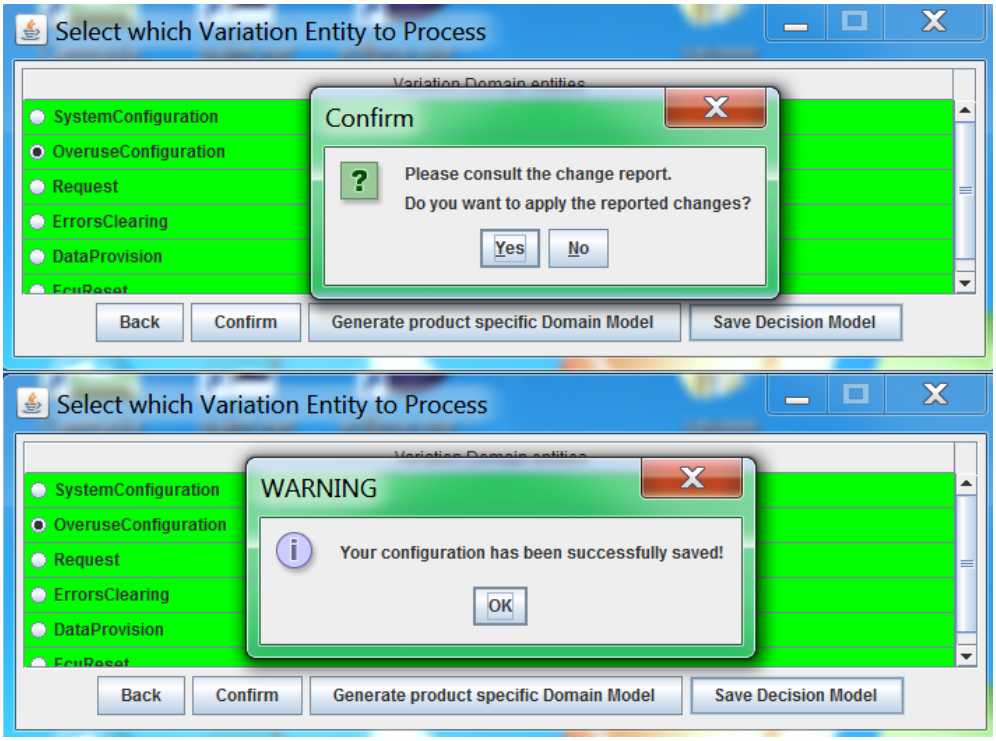

Figure 55: Window to Save the Reported Changes

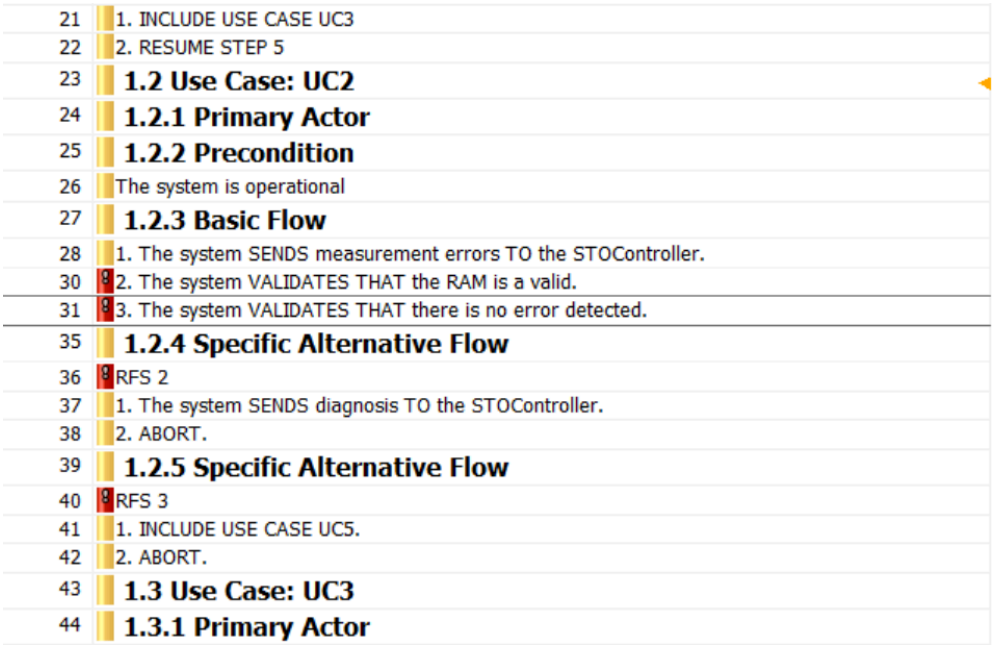

Figure 56: Window for the Incrementally Generated Specifications

3.7 Classifying and prioritizing System Test Cases in a Product Family

3.7.1 Classifying System Test Cases in a Product Family

The system test cases of previous products are classified for the new product in a product line by using the decision changes and the trace links between the system test cases and the PS use case specifications. Please click on *Classify and prioritize test cases* (see Figure 57).

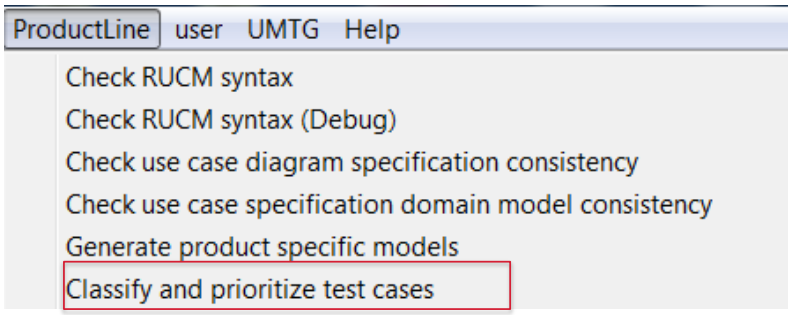

Figure 57: Menu Item in DOORS for Classifying and Prioritizing Test Cases

The list of products to be tested appears. Please select a product to be tested, for example *STOPS2*, and click on *Select the new product* (see Figure 58).

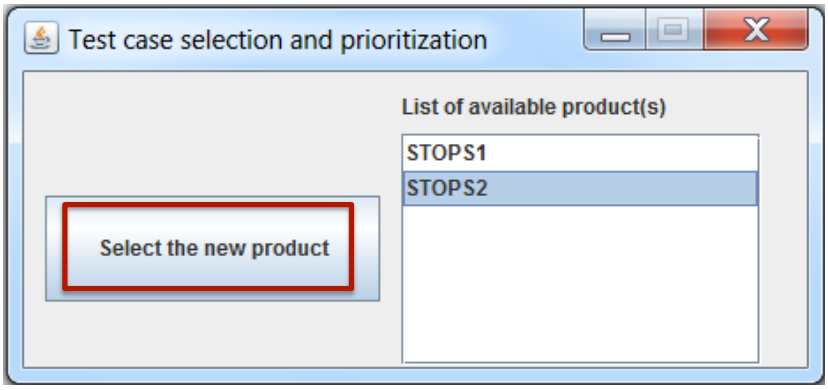

Figure 58: Window for Selecting New Product

The list of existing products in the product line appears. Please select a product from which we classify test cases for the product to be tested, for example *STOPS1*, and click on *Classify* (see Figure 59).

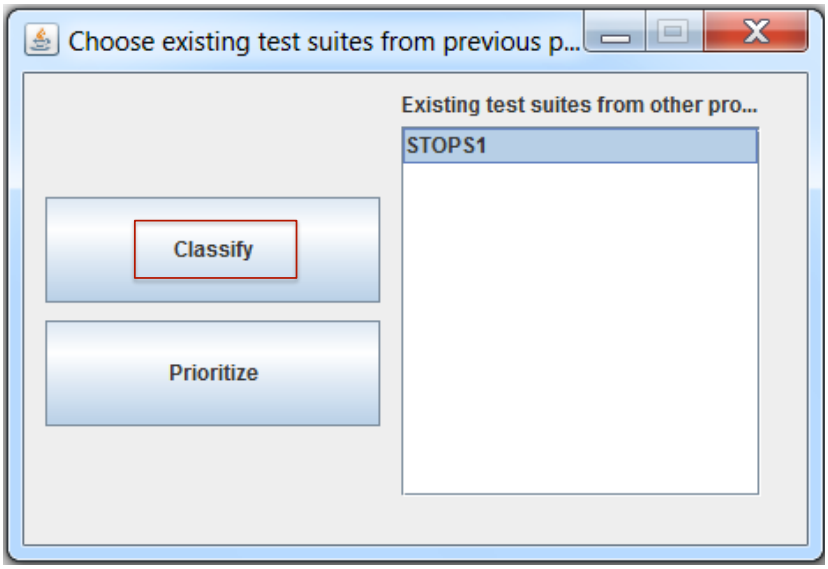

Figure 59: Window for Classifying Test Cases

Figure 60 shows the output of our test case classification. In this figure, we report a summary of the total number of *reusable*, *retestable*, and *obsolete* test cases. We also report the result of test case classification and the list of new test case scenarios in an Excel format, in addition to the guidance for building new test cases.

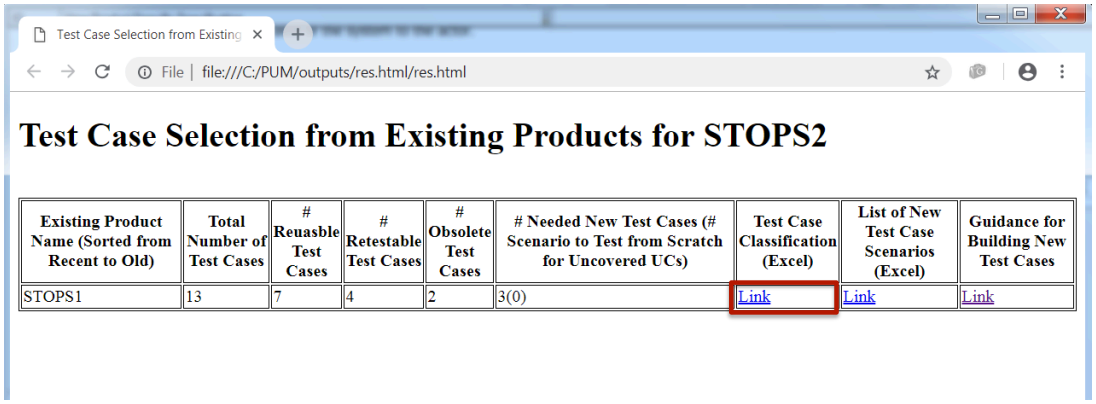

Figure 60: Output of Test Case Classification

Please click on *Link* in *Test Case Classification (Excel)* to consult the result of test case classification (see Figure 61).

A	B	C	D
Use Cases	Use Case Scenarios	Test Case IDs	Test Classification
Recognize gesture	Basic Flow	TCS16	Retestable (graph)
	SAF1	TCS33	Retestable (graph)
	SAF2	TCS74	Retestable (graph)
	SAF3	TCS178	Retestable (graph)
Provide system user data via standard mode	Basic Flow	TCS422	Obsolete (graph)
Provide system user data	Basic Flow	TCS421	Obsolete (graph)
Provide system operating status	Basic Flow	TCS412	Reusable (graph)
	SAF1	TCS413	Reusable (graph)
Identify System Operating Status	Basic Flow	TCS322	Reusable (graph)
	SAF1	TCS346	Reusable (graph)
	SAF2	TCS347	Reusable (graph)
	SAF3	TCS348	Reusable (graph)
	SAF4	TCS1499	Reusable (graph)

Figure 61: Classified Test Suite in Excel Format

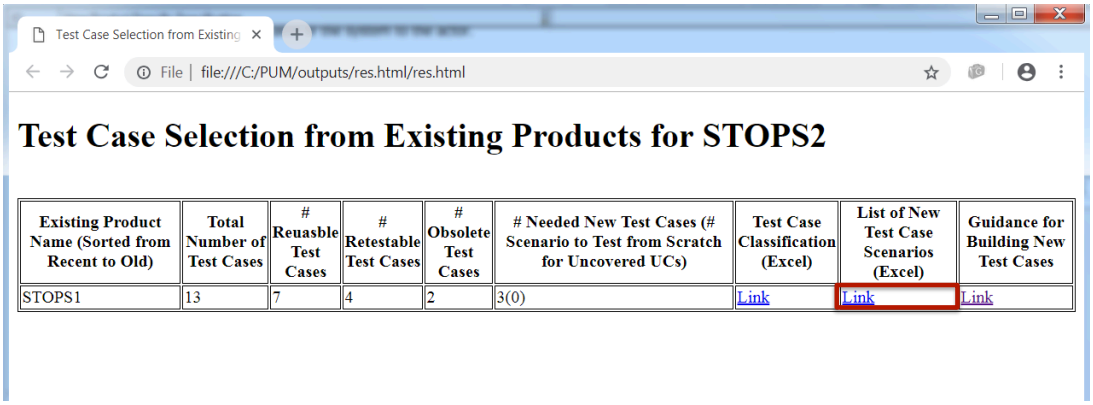

Figure 62: Link for New Scenarios

Please click on *Link* in *List of New Test Case Scenarios (Excel)* to consult the list of new scenarios derived from scenarios covered by obsolete and retestable test cases (see Figure 62).

A	B	C	D
Use Cases	Use Case Scenarios	Test Case IDs	Test Classification
Recognize gesture	Basic Flow--Derived From SAF3 covered by "TCS178" in STOPS1	TCS178	New
Provide system user data via standard mode	Basic Flow--Derived From Basic Flow covered by "TCS422" in STOPS1	TCS422	New
Provide system user data	Basic Flow--Derived From Basic Flow covered by "TCS421" in STOPS1	TCS421	New

Figure 63: List of New Scenarios in Excel Format

Figure 63 shows the list of new scenarios in Excel format.

Please click on *Link* in *Guidance for Building New Test Cases* to consult the guidance for creating new test scenarios for the new product (see Figure 64).

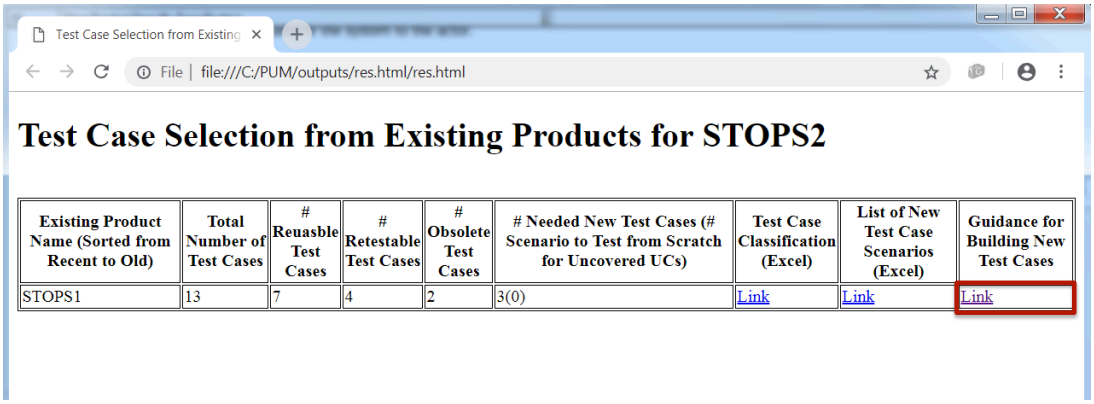

Figure 64: Link for Guidance to Create New Scenarios

Please select, for example, the *Basic Flow* of the use case *Recognize gesture* and click on *Link* in *Link for Guidance* to consult the guidance for creating new scenario (see Figure 65).

List of New Test Cases for STOPS2 from STOPS1

Use Case	Scenario	From Test Case with ID	Link for Guidance
Recognize gesture	Basic Flow	TCS178	Link
Provide system user data via standard mode	Basic Flow	TCS422	Link
Provide system user data	Basic Flow	TCS421	Link

Figure 65: Link for Guidance

We provide a set of suggestions for adding, removing and updating test case steps corresponding to added, removed and updated use case steps in the old and new scenarios. Figure 66 shows an example of the guidance we provide.

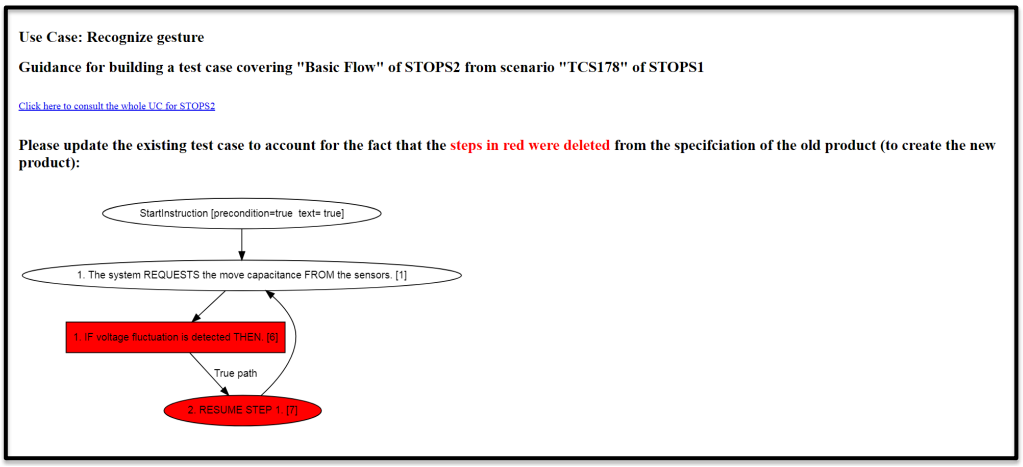

Figure 66: Window for Providing Guidance

3.7.2 Prioritizing System Test Cases in a Product Family

PUMConf employs logistic regression to prioritize test cases. System test cases are automatically prioritized based on multiple risk factors such as fault proneness of requirements and requirements volatility in the product line.

First, select a product to be tested, for example STOPS2, and click on *Select the new product*. Then, select the test suite from which we prioritize test cases, for example STOPS1, and click on *Prioritize* (see Figure 67).

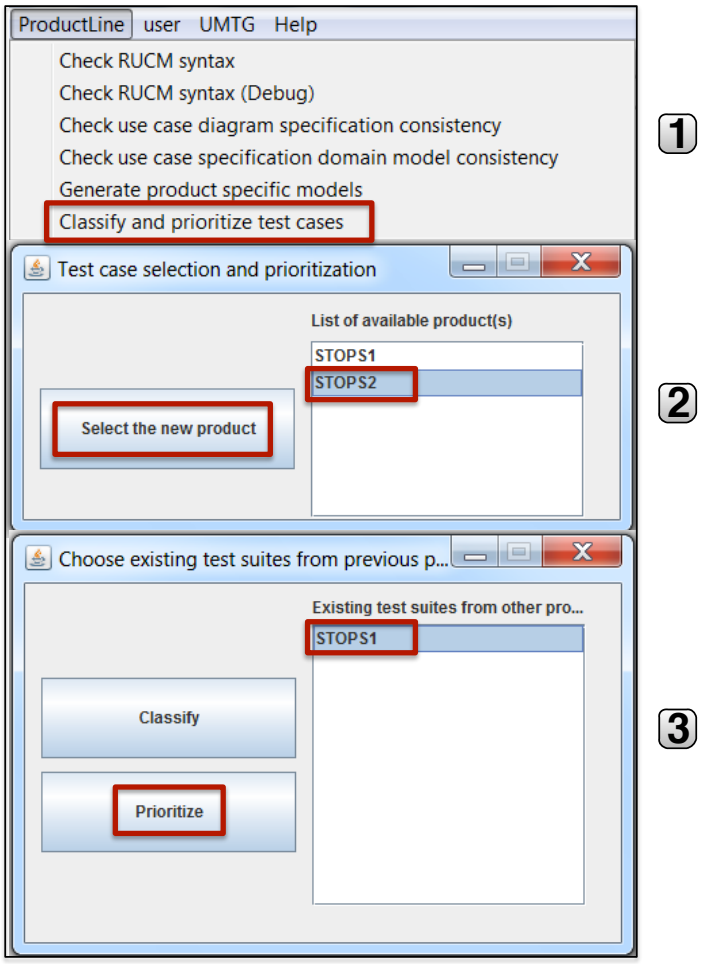

Figure 67: Windows for Prioritizing Test Cases

The test suite for the product to test is sorted using the probability of failure calculated by the regression model. The test cases are sorted in descending order of probability and presented to engineers (see Figure 68).

A	B	C	D	E	F
Project	**Version ID**	**Use Case**	**Senario ID**	**Test Case IDs**	**Prioritization score**
STOPS1	-	Recognize gesture	SAF2	TCS74	0.99999541
STOPS1	-	Provide system operating status	Basic Flow	TCS412	0.99999462
STOPS1	-	Recognize gesture	Basic Flow	TCS16	0.99999151
STOPS1	-	Identify System Operating Status	SAF1	TCS346	0.10431635
STOPS1	-	Identify System Operating Status	SAF4	TCS1499	0.10431635
STOPS1	-	Identify System Operating Status	SAF3	TCS348	0.10431635
STOPS1	-	Identify System Operating Status	SAF2	TCS347	0.08906964
STOPS1	-	Identify System Operating Status	Basic Flow	TCS322	0.08906964
STOPS1	-	Recognize gesture	SAF1	TCS33	0.06100513
STOPS1	-	Recognize gesture	SAF3	TCS178	0.01666668
STOPS1	-	Provide system operating status	SAF1	TCS413	0.01287042

Figure 68: Prioritized Test Suite