ITEA 3 Call 4: Smart Engineering

D3.3.1 Tool for variant selection and test case instantiation

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	WP 3: Testing of Configurable Products	
Task	T3.2: Test case instantiation and distribution of testing efforts amongst variants	
Version	V 1.0	Oct 31st, 2020
Deliverable Type	SW (Software)	
Dissemination Level	Public	

Summary

This document provides a user guide for the tool for variant selection and test case instantiation.

Table of Contents

1. Getting started

1.1 Deliverable

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

The following D3.3 tool is accessible at https://gitlab.com/xivt/itea with username: ITEA3XIVT & password: 20222018XIVT

Varsel tool is delivered as 'Varsel.exe' file. This allows an easy start of the tool. To open Varsel, simply double-click on the .exe.

When Varsel is started, a workbench window is displayed as shown in Figure1.

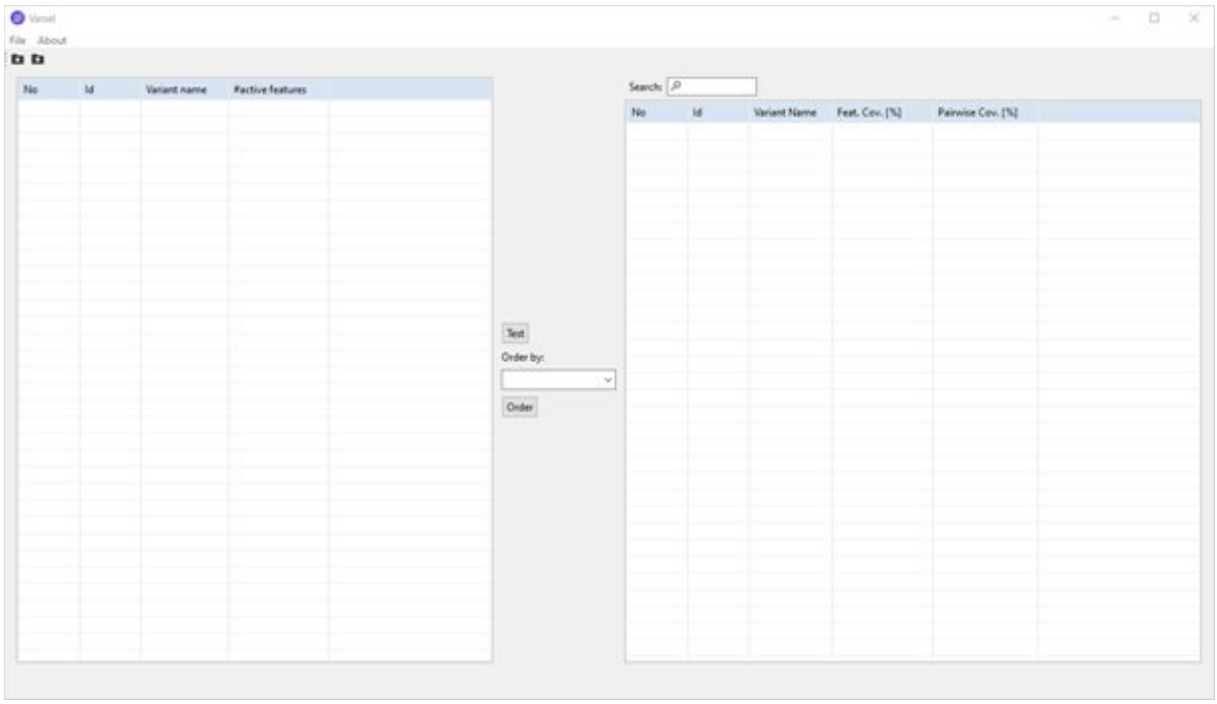

Figure 1: Varsel workbench

1.2 Load file and test case instantiation

In order to load (import) a file containing variants one can click on the menu entry 'File' or on the corresponding icon (see Figure 2). A file dialog opens where the path to the input file can be chosen. Only files of the type *.uml or *.testona can be selected. In case a testona file is selected, the user needs to make sure, that the file contains variants. Then the user can decide whether classes or classifications shall be used as features. Testona allows test case instantiation for the chosen variants. The variants will then be loaded to the left hand side table within the workbench.

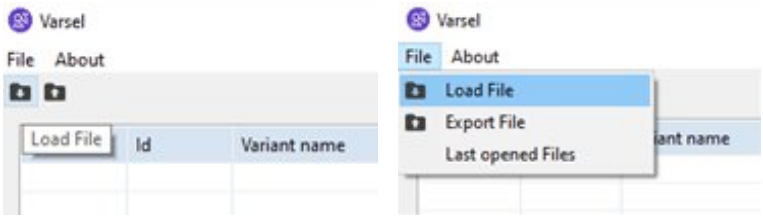

Figure 2: Import / load file

2. Concept

2.1 Select Algorithm

In the middle of the workbench window, below the text 'Order by' the algorithm that shall be performed can be selected. One can choose the algorithm by selecting one of the algorithms within the drop down menu shown in Figure 3.

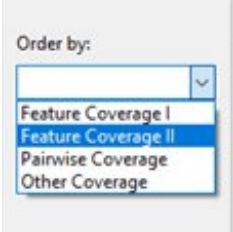

Figure 3: Select algorithm

2.2 Apply Algorithm

Directly below the drop down menu to select the algorithm, one can click on the button *'Order'* and the chosen algorithm will be performed. This may take a while, depending on the number of variants and features coming from the loaded file. The progress on calculating the algorithm is shown on the bottom right corner by the progress bar (see Figure 4).

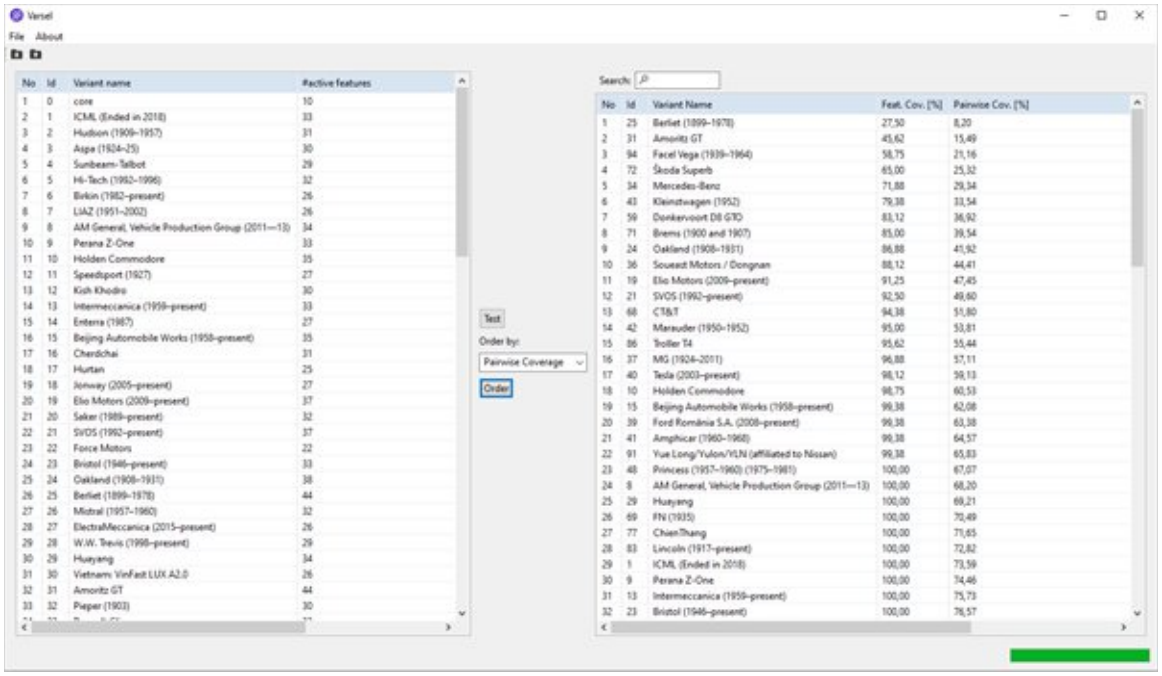

Figure 4: Workbench showing performed pairwise algorithm and status bar

3. Algorithms

This chapter contains a brief description of algorithms used in variant selection. The description is followed by an example using the list of variants in Table 1. An active feature is represented by a 1 and an inactive feature by 0.

Variant number	Features	Number of active features (1)
1	1 0 0 0 1 1 0 0	3
2	0 1 0 0 0 0 0 0	1
3	0 0 1 1 0 0 0 1	3
4	1 1 1 1 0 0 0 1	5
5	0 0 0 0 0 1 0 0	1
6	0 0 0 0 0 0 0 0	0
7	1 0 1 0 1 0 1 1	5
8	1 1 0 0 1 0 0 0	3
9	0 0 0 0 0 0 1 1	2
10	1 0 1 0 0 0 1 1	4

Table 1: Example list of variants

3.1 Feature Coverage I

This algorithm orders a list of variants to achieve maximum feature coverage per step.

Given is a core model and delta models together containing a full list of features that can appear in a product. Furthermore given is a list of Variants V_1 … V_n. Each variant is represented by its instantiated features.

To order this list by maximizing feature coverage, the following steps need to be performed.

1. Get a list of all possible features from the core and delta models, duplicate it to obtain a list of not yet covered features.

2. Go through the variants and choose the variant covering the most not yet covered features. If there is more than one, again, choose any.

3. Move that variant to the ordered variant list.

4. Compute the percentage of already covered features for the ordered variant list and show it next to the variant.

5. Remove the features covered by this variant from the list of not yet covered features.

6. Go to step 2.

7. If all features are covered: Just add the remaining features to the list (or prioritize in some way)

Example:

Given the list of unordered variants in Table 1, the variants shall be ordered according to Feature Coverage I algorithm. Please note, that the example is only ordering the variants, the rest of the algorithm (e.g. moving variants from one list to another etc. is not explained in detail here.

- *Step 1:* Choose variant with the most active features:

  Variant 7: 1 0 1 0 1 0 1 1 62,5%

  Note: Instead of variant 7, variant 4 could also be the possible first variant.

- *Step 2:* Choose next variant, covering most yet uncovered features:

  Variant 4: 1 1 1 1 0 0 0 1 87,5%

- *Step 3:* Repeat step 2, until 100% coverage is reached.

  Variant 1: 1 0 0 0 1 1 0 0 100,0%

  Note: Instead of variant 1, variant 5 could also be the third possible variant, adding up to 100% percent coverage.

- *Step 4:* 100% coverage is reached, therefore all the remaining variants will just be added to the ordered list.

  Ordered list: variant 7, variant 4, variant 1, variant 2, variant 3, variant 5, variant 6, variant 8, variant 9, variant 10

3.2 Feature Coverage II

This algorithm orders a list of variants to achieve maximum feature coverage per step. Given is a core model and delta models together containing a full list of features that can appear in a product. Furthermore given is a list of variants V_1 … V_n. Each variant is represented by its instantiated features.

To order this list by maximizing feature coverage, the following steps need to be performed.

1. Get a list of all possible features from the core and delta models, duplicate it to obtain a list of not yet covered features.

2. Go through the variants and choose the variant covering the scarcest feature(s). If there is more than one, choose the variant also covering the most active features.

3. Move that variant to the ordered variant list.

4. Compute the percentage of already covered features for the ordered variant list and show it next to the variant.

5. Remove the features covered by this variant from the occurrence array.

6. Go to step 2.

7. If all features are covered: Just add the remaining features to the list (or prioritise in some way)

Example:

Given the list of unordered variants in Table 1, the variants shall be ordered according to Feature Coverage II algorithm. Please note, that the example is only ordering the variants, the rest of the algorithm (e.g. moving variants from one list to another etc. is not explained in detail here.

- *Step 1:* Choose variant with the rarest features. Therefore, the occurrence array is generated. If more than one exists, choose the one covering the most active features:

  Occurrence array: 5 3 4 2 3 2 3 5

  Variant 4: 1 1 1 1 0 0 0 1 62,5%

- *Step 2:* According to already chosen variant, the occurrence array is updated: Every feature that is covered now, will be set to 0 in the occurrence array. Then, chose the next variant covering now rarest features. If more than one exists, choose the one covering the most active features:

  Occurrence array: 0 0 0 0 3 2 3 0

Variant 1: 1 0 0 0 1 1 0 0 87,5%

- *Step 3:* Repeat step 2, until 100% coverage is reached.

  Occurrence array: 0 0 0 0 0 0 3 0

  Variant 7: 1 0 1 0 1 0 1 1 100,0%

- *Step 4:* 100% coverage is reached, therefore all the remaining variants will just be added to the ordered list.

  Ordered list: variant 4, variant 1, variant 7, variant 2, variant 3, variant 5, variant 6, variant 8, variant 9, variant 10

3.3 Pairwise Coverage

This algorithm orders a list of variants in order to achieve maximum pairwise coverage per step. Given is a core model and delta models together containing a full list of features that can appear in a product. Furthermore given is a list of Variants V_1 … V_n. Each Variant is represented by its instantiated features.

1. Get a list of all possible pairs from the core and delta models, duplicate it to obtain a list of not yet covered pairs.

2. Go through the variants and choose the variant covering the most not yet covered pairs. If there is more than one, choose the one that additionally covers the most not already covered features. If there is more than one again, choose any.

3. Move that variant to the ordered variant list.

4. Compute the percentage of already covered pairs for the ordered variant list and show it next to the variant.

5. Remove the pairs covered by this variant from the list of not yet covered pairs.

6. Go to 2.

7. If all pairs are covered: Just add the remaining pairs to the list (or prioritise in some way).

Example:

Given the list of unordered variants in Table 1, the variants shall be ordered according to Pairwise Coverage algorithm. Please note, that the example is only ordering the variants, the rest of the algorithm (e.g. moving variants from one list to another etc. is not explained in detail here.

List of all possible pairs: [1,2] [1,3] [1,4] [1,5] [1,6] [1,7] [1,8] [2,3] [2,4] [2,8] [3,4] [3,5] [3,7] [3,8] [4,8] [5,6] [5,7] [5,8] [7,8]

- *Step 1:* Choose variant covering the most pairs of active features:

  Variant 4: [1,2] [1,3] [1,4] [1,8] [2,3] [2,4] [2,8] [3,4] [3,8] [4,8] 52,63%

  Note: Instead of variant 4, variant 7 could also be the possible first variant.

- *Step 2:* Remove covered pairs from list of uncovered pairs. Then choose next variant, that covers most yet uncovered pairs:

  Uncovered Pairs: [1,5] [1,6] [1,7] [3,5] [3,7] [5,6] [5,7] [5,8] [7,8]

  Variant 7: [1,5] [1,7] [3,5] [3,7] [5,7] [5,8] [7,8] 89,47%

- *Step 3:* Repeat step 2, until 100% coverage is reached.

  Variant 1: [1,6] [5,6] 100,00%

- *Step 4:* 100% coverage is reached, therefore all the remaining variants will just be added to the ordered list.

  Ordered list: variant 4, variant 7, variant 1, variant 2, variant 3, variant 5, variant 6, variant 8, variant 9, variant 10

Additional info: The example list of variants shows an easy example to calculate the three different algorithms. Therefore, the ordered list do not seem to be a lot different. As soon as the list of variants is longer or the amount of features is higher, the ordered lists can vary more.

4. Export

The output of the tool (ordered list according to the chosen algorithm) is not only shown in the Tool on the right hand side but can be exported as .json.

Either click on the Export icon or select 'File' □ 'Export File'. The saving dialog opens and the location for the export file can be selected. If the chosen filename already exits, the file will be overwritten. A warning will appear and needs to be confirmed with yes or no.

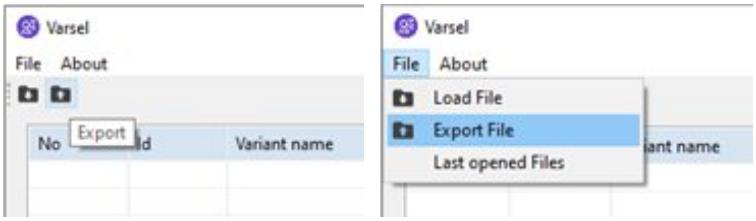

Figure 5: Export file