Features and Behaviours Mapping In Model-based Testing in Software Product Line

Raduni Sulaiman[1,2], DNA. Jawawi[1] and Shahliza Abdul Halim[1]
[1] School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, 81300 Skudai, Johor, Malaysia
[2] Faculty of Computer Science and Information System, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Johor, Malaysia

E-mail: raduni2@live.utm.my

Abstract. Testing in Software Product Line (SPL) is very hard task due to the high degree of variability that existed in products. Nowadays, many testing approaches have concern on reusability of technique. Feature Model (FM) is used to represent variability and commonality of products in SPL. The components of FM that represented as symbols caused the needs of mapping with other models to represent their semantics. In this paper, there are concise definitions that relates with mapping approaches between FM and behaviour model. Model definitions presented in our algorithms is used for automation mapping process based on traceability link created. The advanced query mechanism proposed to automate the process of mapping between models. Based on the experimental result, it shows that our proposed algorithm can help tester in automate searches for accurate mapping of features and requirements.

1. Introduction

Software Product Line (SPL) is one of the subdivisions in software engineering that is mainly focused on managing variability and commonality in group of products [1]. It is based on the concept of the same products that still differ based on specific features or functionalities. High degree of variability leads to many possible products. In industry, new products are continually developed. For example, in automotive industry for software testing field, almost all car brand have their own product configurations, but it is difficult to differentiate each configuration. Development of products has caused configuration testing to be infeasible due to limited time and high cost. SPL domain face challenges in managing large number of products and requirements in terms of commonality and variability [2].

In order to handle this issue, many techniques have been introduced including model-based testing(MBT). This technique is proven to be one of the efficient ways to manage variability and commonality. The usage of MBT can help to handle SPL products to be more understandable since formal model descriptions are used [3]. This technique can also be used starting at early phase of software development and offers automated test case generation with better coverage and can achieve a higher fault detection. Our research strives to manage requirement and enhance quality improvement in terms of product mapping between features and behaviours. We address this problem by considering variability and commonality represented in Feature Model (FM) and behaviour model (statechart) that can be used to generate test cases to prove our test suite generation algorithm.

In our previous work, test case generation by using MBT test model with consideration of coverage criteria has been presented [4]. The usage of FM as part of our work since FM is functional and capable to manage variability. It is worth noting that the representation of FM in the form of tree does not have association with requirements or architecture model. The importance of FM can be realized if there is no integration with other existing models [5]. In this paper, we continue this issue with some new improvement in terms of automation of mapping technique.

2. Related Works

The mechanism of variability in test model has been adapted in MBT by using activity diagram [6], [7] statechart [8]–[10], sequence diagram and Markov chain [11], [12]. Some studies have reused multiple model to represent requirements based on their proposed work. K. Czarnecki et al.[6] have map FM with any kind of models. They represent the template-based approach to map the features. However, there is still required interaction of tester to validate each products. Thus, it caused this approach inappropriate to handle large size of products. S. Weißleder et al.[5] has represent two approaches which are top-down and bottom-up that used to link features with statechart. H.Lackner et al.[13] have proposed an approach that based on the propositional formula and UML model transformations. They have represent the mapping of feature with transition of statechart. The propositional formula provided consists of Boolean flag that specify the model element to be mapped with features. In contrast to our work, they focus on different test design to map feature with other model. Limitation of these approaches involves the needs of testing all individual products and required manual human effort to validate all features. This caused the complexity of model and incompatible for large number of products.

This motivated us to used FM and to represent statechart as test model that used to map commonality and variability with behaviour requirements. Our aim is to have an automated mapping of features and states. Traceability link based on advance query mechanism is proposed to automate the process. This mapping approach is capable to handle the large number of products since we have considered automation.

3. The proposed traceability generator

In order to map features with states, the generator is proposed in order automate the trace process. As per illustrated in Figure 1, the proposed traceability generator mainly consist of the advance query mechanism. Here, the advance query comprise three functionalities which are validation, searching rule and execution rule. This query takes input from the traceability link created. This query aim to remove, unrelated states that are not able to present the variants of products. As a result, only states that are related with at least one product configuration are considered in test model. The restrictions of mapping process are, the single feature can be mapped with one or many states. Secondly, if feature is mapped with states, it is automatically set as true (accepted) for test model. Details of the traceability relationship of FM_STATE are describe in Section 4.

4. Traceability Relationship of FM_STATE Mapping Approach

Our aim of this work is to automate the process of MBT for SPL with minimal involvement of human effort in the process of mapping between models. The process of automation is designed in multiple search process of states and product configurations. However, due to the different naming conventions, some of the features are not included for mapping. Two rules are used to create relationship between product configurations and test model. Thus, one of the way is by using traceability in the mapping process. This relationship will specify the pair of artefacts comprising the source and target. In our perspective, the target is the statechart whereas the source is product configurations *vc*. In order to have the traceability relationship between *vc* and *s*, we consider two types of requirements which are one feature can represent one state and one feature can represent many states. We refer to the forward requirement traceability since we only consider one way trace instead of backward traceability [15]. The mapping between product configurations as $s_{vc} = \Delta(f \cap s)$ where S = total states; s_{vc} = states that have link with product configurations.

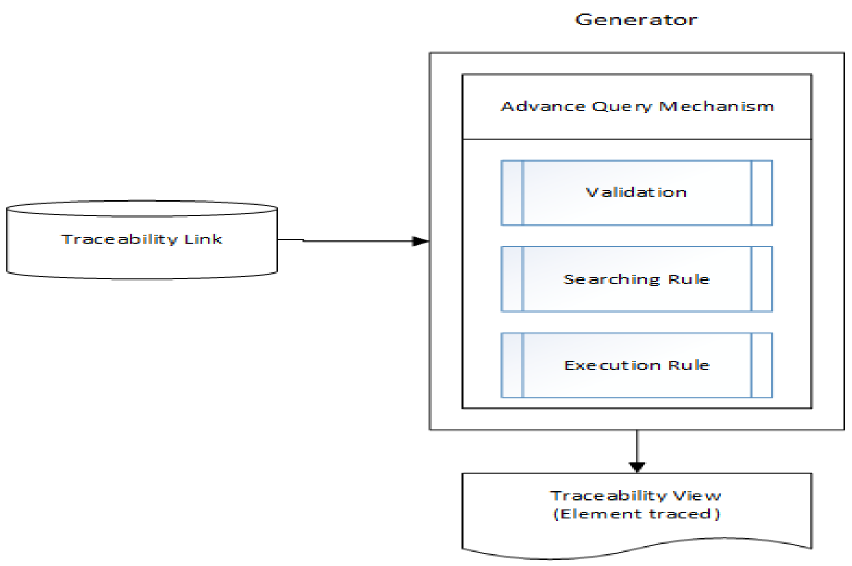

Figure 1: The proposed traceability generator

This automation approach will provide the list of states that considered variability and commonality with FM. It consists of the bindings between features, requirements and states that help to derive test model. Figure 2 consists of the automated process of mapping to generate new statechart test model.

```
Input: $fd, s_s, t_s$
Output: $m_{fm\_state}$
Upload files $fd, s_s, t_s \rightarrow m_{fm\_state}$

1.  Upload files $fd, s_s, t_s \rightarrow m_{fm\_state}$
2.  Display data of $fd, s_s, t_s$ ($fdcontroller()$, $umlcontroller()$)
3.  Upload dictionary $dic \rightarrow m_{fm\_state}(t_s)$
4.      Insert in dataDetail()
5.  dataDetail $\rightarrow m_{fm\_state}$
6.  Function dataDetail ($fd, s_s, t_s$)
7.      dataList = 0;
8.      For ($fd$.dataList = 0)
9.          $fd \in s_s$    {Retrieve all reachable features and states}
10.         If (fd $\neq$ 0) then
11.             getMatchList();
12.         EndIf
13.     End for
14. Function getMatchList ($fd, s_s, t_s$)
15. $n \leftarrow$ dataList
16. $m \leftarrow$ dataListStates
17. For each $n, m$ do
18.     For each $n \in m$ do
19.         $m_{fm\_state}$ = n ∪ m
20.     End for
21. End for
```

Figure 2: Advanced query for traceability mapping algorithm

We implemented the multi-search algorithm in advance query conduct mapping. The output of this query is to provide the list of states that are required in generation of test model. This algorithm also will automate the termination of transitions that have unrelated states. Our algorithm starts with upload of .xml files into the FM_STATE approach. This function consists of the <arraylist> dataDetail that auto matched name between fd and ss. These two parameters then will call functions *fdcontroller*() and *umlcontroller*(). *fdController*() contains list of *fd* data that is taken from .xml file of SPLOT whereas *umlController* is used to listed states and transitions taken from .xml file. Furthermore, there are also another input that we called as *dic* file that is uploaded by engineer to create another tracing process of *fd* and s_s since we have noticed that some of the *fd* is not fully traced in the second step. In this process, we start with function *getMatchList*() that will check the product that is still not map with any state. This function contains all values of parameters after matching and will be call by code function. Here, double search query is implemented to match the products. Then, the traceability link will help to automate searches of products another one time to create new checking. It help to produce a new valid model that is ready for test case generation. New *tm* created consists of the three main parts which are $tm=\{fm,sc,s_{vc}\}$.

5. Results and Discussion

For evaluation purpose, we applied our mapping algorithm based on three case studies with different size of products. As per Table 1, case study details used is listed. For evaluation purpose, we relied on three metrics of information retrieval [15] based on precision, recall and F-measures. Fig. 3 depicted the summary of precision, recall and F-measure against mapping results. For alarm system case study, it return 1 for precision, vc_1(0.90), vc_2(0.75) and vc$_3$(1) for recall. F-measure are vc_1(0.94), vc_2(0.85), and vc_3(1). When go for the mobile phone case study, recall value is decreased to vc_1 (0.80), vc_2(0.85) and vc_3 (0.78). For precision also decreased to vc_1(0.84), vc_2(0.82) and vc_3(0.89). The highest F-measure for mobile phone are vc_2 and vc_3 with 0.83. For online shop, the precision values are vc_1(0.87) followed by vc_2(0.82) and vc_3(0.80). For recall, the highest is vc_1(0.82) followed by vc_2(0.80) and vc_3(0.73) and the F-measure are vc_1(0.84), vc_2(0.80) and vc_3(0.76).

As per results, precision values also tend to decrease if large size of products is involved. It is considered as low which means that the mapping process does not include all related product configurations. As summary, alarm system covered the best average value for precision and recall. It related with the size of case study since it is the smallest among others. However, to check the validity of our tool with size of large case study, the results of precision and recall are better since the average values starts from 0.76. Average values of precision and recall that represented as F-measure shows that the alarm system can covered all configurations followed by mobile phone and online shop.

Table 1: FM_STATE mapping results

Case Study	*vc*	Relevant S_{vc}	Actual S_{vc}	Missing S_{vc}	Selected S_{vc}	Precision	Recall	F-Measure
Alarm System	vc$_1$	11	10	1	10	1	0.90	0.94
	vc$_2$	8	6	2	6	1	0.75	0.85
	vc$_3$	8	8	0	8	1	1	1
Mobile Phone	vc$_1$	20	16	4	19	0.84	0.80	0.81
	vc$_2$	27	23	4	28	0.82	0.85	0.83
	vc$_3$	32	25	7	28	0.89	0.78	0.83
Online Shop	vc$_1$	69	57	12	65	0.87	0.82	0.84
	vc$_2$	60	48	12	58	0.82	0.80	0.80
	vc$_3$	68	50	18	62	0.80	0.73	0.76

The mapping algorithm automatically picks as random the states and products based on naming convention setup and dictionary of traceability link created. This process still require one time effort by engineer to set up the process. This tool practice have benefits and drawbacks concerning reusability and efficiency. These issue relate to the nature of MBT that required more effort for development but can work more efficiently in terms of reusability of model-based that express variability. There are also efficiency issue related with engineer knowledge regarding model, variants and behaviour based on product specification. However, this practice have advantages in terms of certifying the complete test model used for test generation with high computational complexity.

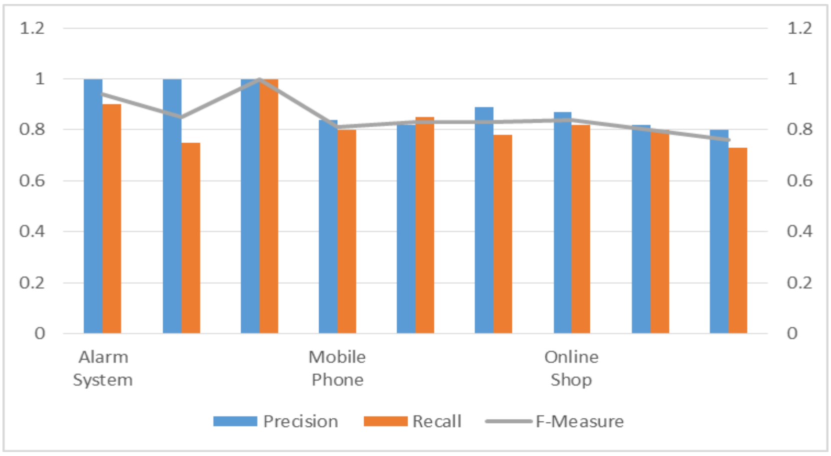

Figure 3. Traceability accuracy based on precision, recall and F-measure

We list two benefits of our mapping process which are (1) for each potential SPL, variants are validated including features constraint. (2) automation process of mapping that can minimize the cost of human and development based on traceability link svc. The consistency and completeness of test model is decided by the domain experts. Our mapping method is based on forward traceability that will relate software artefacts. We allow definition of artefacts between product configurations and states in terms of dictionary form. We also offered to define the mapping link manually by engineer. There are also advanced query mechanism that will instantly trace the mapping process between artefacts and trace links. There are filtration processes of model artefacts and definition dictionary that help to detail out the mapping process.

6. Conclusion and Future Works

This paper presents automated mapping process of software artefacts based on the traceability link created. To achieve the objective of this work, one algorithm has proposed which based on the advance query mechanism to trace software artefacts. Experiments is performed with three different sizes of case study to check the effectiveness of the approach. We include coverage criteria element in product configurations resulting in consideration of constraints in FM. Result reveal that the traceability link created supports mapping and fulfils the automation process. Furthermore, it was observed that our mapping algorithm can accurately detect the products and states. Finally, results showed that the large product configuration and statechart can give effect towards tracing results. Some artefacts are possibly missing from the mapping process since the size of product has increased. As a future improvement of our method, we aim to enhance our approach as automated MBT test case generator based on coverage criteria. We intend to further expand this approach to create MBT test case generator based on test model created based on current mapping result.

Acknowledgments
This research is fully funded by Ministry of Higher Education Malaysia (MOHE) for FRGS Grant Vot No. 5F117 and Universiti Teknologi Malaysia for UTM-TDR Grant Vot No. 06G23. We would like to express our profound gratitude to lab members of Software Engineering Department, Universiti Teknologi Malaysia (UTM) for their continuous support and constructive feedback towards the completion of this paper.

References

[1] X. P. G. L. A. Devroey, "Search-based Similarity-driven Behavioural SPL Testing," in *Proceedings of the ACM International Conference on Computing Frontiers. ACM, 2016*, 2016, pp. 89–96.

[2] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon, "Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines," in *Proceedings of the 37th International Conference on Software Engineering*, 2015, pp. 517–528.

[3] H. Samih *et al.*, "An Approach to Derive Usage Models Variants for Model-based Testing," in *The 26th IFIP International Conference on Testing Software and Systems*, 2016, pp. 80–96.

[4] R. A. Sulaiman, D. N. A. Jawawi, and S. A. Halim, "Coverage-based Approach for Model-based Testing in Software Product Line," *Int. J. Eng. Technol.*, vol. 7, pp. 63–68, 2018.

[5] S. Weißleder and H. Lackner, "Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines," *Electron. Proc. Theor. Comput. Sci.*, vol. 111, no. Mbt, pp. 82–94, 2013.

[6] K. Czarnecki and M. Antkiewicz, "Mapping Features to Models : A Template Approach Based on Superimposed Variants Background : Feature Modeling," *Proc. 4th Int. Conf. Gener. Program. Compon. Eng. GPCE 2005*, pp. 422–437, 2005.

[7] A. Reuys, E. Kamsties, K. Pohl, and S. Reis, "Model-Based System Testing of Software Product Families," *Int. Conf. Adv. Inf. Syst. Eng.*, pp. 519–534, 2010.

[8] S. Oster, "Feature Model-based Software Product Line Testing," Technische Universität, 2012.

[9] H. Cichos, S. Oster, M. Lochau, and A. Schuerr, "Model-Based Coverage-Driven Test Suite Generation for Software Product Lines," *Model Driven Eng. Lang. Syst.*, vol. 6981, pp. 425–439, 2011.

[10] M. Lochau, S. Oster, U. Goltz, and A. Schürr, "Model-based pairwise testing for feature interaction coverage in software product line engineering," *Softw. Qual. J.*, vol. 20, no. 3–4, pp. 567–604, 2012.

[11] C. S. Gebizli and H. Sozer, "Model-Based Software Product Line Testing by Coupling Feature Models with Hierarchical Markov Chain Usage Models," *Proc. - 2016 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS-C 2016*, pp. 278–283, 2016.

[12] X.Devroey, "Behavioural model-based testing of software product lines," University of Namur, 2017.

[13] H. Lackner, M. Thomas, F. Wartenberg, and S. Weißleder, "Model-based test design of product lines: Raising test design to the product line level," *Proc. - IEEE 7th Int. Conf. Softw. Testing, Verif. Validation, ICST 2014*, pp. 51–60, 2014.

[14] M. Varshosaz, G. Schneider, and W. Mostowski, "Modeling and Model-Based Testing of Software Product Lines," Halmstad University Press, 2019.

[15] N. Anquetil, R. Mitschke, A. Moreira, A. Rummler, and J. Royer, "A Model-Driven Traceability Framework for Software Product Lines," *Softw. Syst. Model.*, pp. 427–451, 2012.