Automated Product Line Methodologies to Support Model-Based Testing

Shuai Wang, Shaukat Ali and Arnaud Gotlieb

Certus Software V&V Center, Simula Research Laboratory, Norway

{shuai, arnaud, shaukat}@simula.no

Abstract. Testing products in a cost-efficient way remains an attractive topic for Model-Based Testing (MBT) of product lines in both academia and industry, which can be addressed by employing systematic and automated approaches based on models (such as feature models and UML models). Cost-effective testing products can be divided into three main problems, i.e., test selection, test generation, and test minimization. Driven by the needs of our industrial problems for testing Video Conferencing Systems (VCSs) product line developed by Cisco, Norway, this paper presents Product Line Model-based Testing Methodologies (PL-MTM) to tackle the above-mentioned three problems for cost-effective testing a product in product line, which includes: 1) an systematic and automated test selection methodology; 2) an automated test minimization approach; and 3) an automated and systematic test generation methodology.

1 Introduction

Product Line Engineering (PLE) is well known to systematically capture and manage commonalities and variabilities of a product line, which usually includes a number of products [1]. By employing PLE, cost-effective testing products in a product line can be classified into three main problems; 1) *Test Selection*: Automatically and systemically select a sub test suite including a set of relevant test cases for a product from the entire suite available for a product line; 2) *Test Minimization*: Minimizing the test suite obtained by the selection to eliminate redundant test cases for reducing the cost of testing (e.g., execution time) while preserving high effectiveness (e.g., fault detection capability); and 3) *Test Generation*: Automatically and systemically generate test cases when new functionalities are introduced to the product line by the product.

Driven by the needs of our industrial problem of testing Video Conferencing Systems (VCSs) product line developed by Cisco, Norway, we proposed Product Line Model-based Testing Methodologies (PL-MTM) for cost-effective testing of a product, which mainly includes three activities to tackle the above three problems respectively. For the *Test Selection* problem, we proposed an automated and systematic methodology using Feature Model (FM) and Component Family Model (CFM) to select a set of relevant test cases for a product from the entire test suite developed for the product line [2, 3]. For the *Test Minimization* problem, we proposed an automated approach by defining five cost-effectiveness measures based on the testing requirements and discussion with test engineers in Cisco and formulating them as a fitness function. The proposed fitness function was integrated into various search algorithms (e.g., Genetic Algorithms (GA)) for evaluating their performance in terms of finding optimal solutions [4-6]. For the *Test Generation* Problem, we proposed a systematic modeling methodology based on FM, CFM, standard UML class diagrams, UML state machines, aspect class diagrams, and aspect state machines to automatically generate executable test case [7]. This paper presents PL-MTM in detail.

To our knowledge, few of the existing works studied test selection, test minimization and test generation together using FM. In our context, we employed FM and CFM in product lines for these testing challenges by: 1) applying FM and CFM for automated selection of test cases; 2) minimizing the obtained test suite based on various cost/effectiveness measures using search algorithms; and 3) applying FM and CFM to automatically select and configure behavioral models for model-based test case generation (More related work can be consulted in [2-7]).

2 Product Line Model-based Testing Methodologies (PL-MTM)

In this section, we present our PL-MTM in terms of test selection, test generation and test minimization, respectively.

2.1 Test Selection

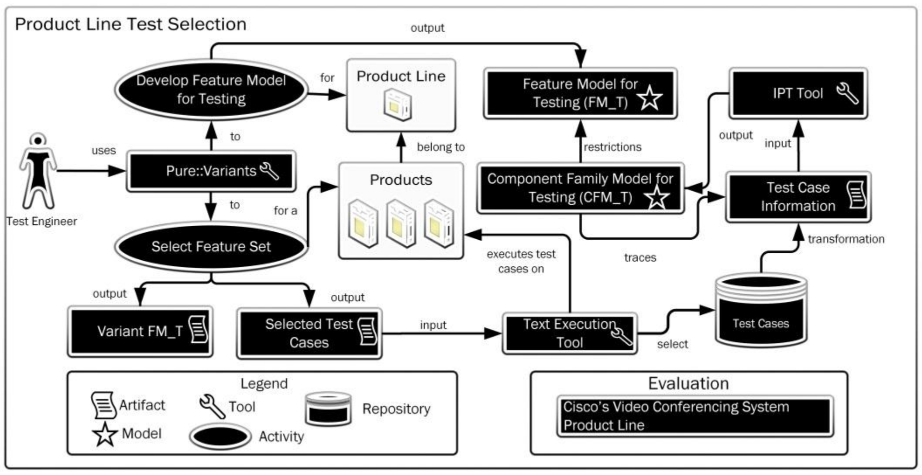

Fig. 1. Overview of test selection methodology

Fig. 1 illustrates an overview of our test selection methodology, which aims at obtaining a set of relevant test cases systematically and automatically. First, based on the expertise and discussions with test engineers in Cisco, we developed a Feature Model for Testing (FM_T) to capture the commonalities and variabilities of the product line using a commercial tool namely Pure::Variants[1]. Second, a Component Family Model for Testing (CFM_T) was automatically developed to capture the overall test case structure in the *Test Cases* repository using a tool we developed called Import Plugin and Transformation (IPT) [2]. By linking CFM_T and FM_T with restrictions (also built automatically by IPT), test engineers only requires to select a set of features for a product through Pure::Variants and a set of relevant of test cases will be obtained automatically. Finally, the obtained test suite can be put into a test execution tool (developed by Cisco) for testing the product.

We evaluated our test selection methodology using the product line of Video Conferencing Systems developed by Cisco called Saturn and performed test case selection for its four products. The results showed the effort such as selection can be reduced significantly (on average 87.5%) as compared with the current manual process [2].

2.2 Test Minimization

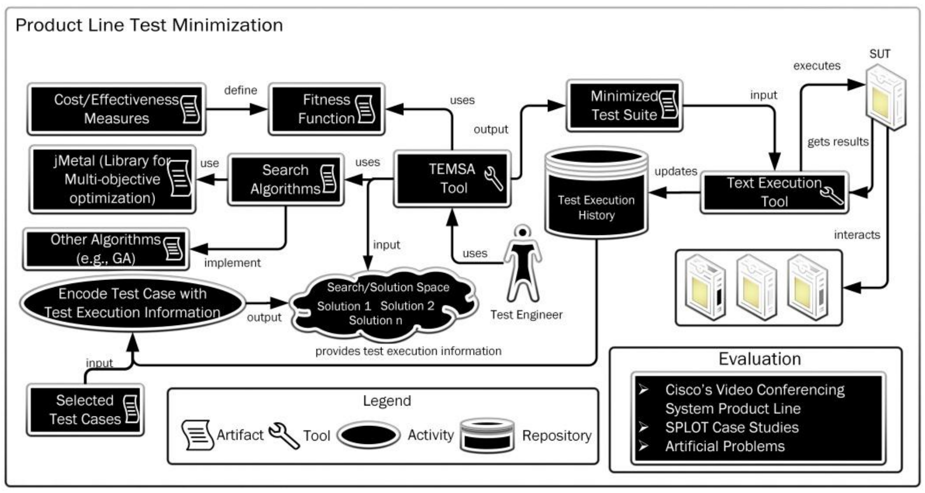

Fig. 2. Overview of test minimization approach

Through more investigation, we observed there were still redundant test cases existing in the selected test suite, which required minimization for reducing the cost of testing (e.g., execution time). But such minimization needs to preserve high effectiveness as compared with the original test suite, which can be considered as a multi-objective optimization problem and solved by search algorithms [4]. Based on that, we proposed an automated approach by defining five cost/effectiveness measures (e.g., fault detection capability) and formulating them as a fitness function as shown in Fig. 2 [4, 5]. The proposed fitness function was investigated into various search algorithms for guiding the search, which were implemented based on jMetal (a java library for multi-objective optimization search algorithms[2]) and by ourselves (e.g., (1+1) Evolutionary Algorithm). Using the test cases obtained by the selection methodology (Section 2.1) and test execution information from the repository *Test Execution History* (e.g., fault

[1]http://www.pure-systems.com/Home.142.0.html [2]http://jmetal.sourceforge.net/

detection capability), a solution/search space can be encoded, which includes a large number of potential solutions, and our goal is to find the optimal solution from such space using search algorithms included by our tool called TEst Minimization using Search Algorithms (TEMSA), which was developed using Java Sencha ExtJS (a JavaScript Framework for Rich Web Apps[3]). By TEMSA, the minimized test suite is obtained and further given as an input to the test execution tool for testing the System Under Testing (SUT). Finally, the test execution information for the test cases will be returned back to the repository *Test Execution History* and update the related test data (e.g., fault detection capability). Notice that the SUT system may interact with other systems during the test case execution.

The proposed cost/effectiveness measures and fitness function were evaluated in conjunction with various search algorithms using the Cisco industrial case study, five case studies from SPLOT[4] and 500 artificial problem of varying complexity [4-6]. The results showed that (1+1) EA combined with Path-oriented Random Testing (PRT) weight strategy achieved the best performance (on average 47% test minimization with 84% feature pairwise coverage and 91% fault detection capability) [6].

2.3 Test Generation

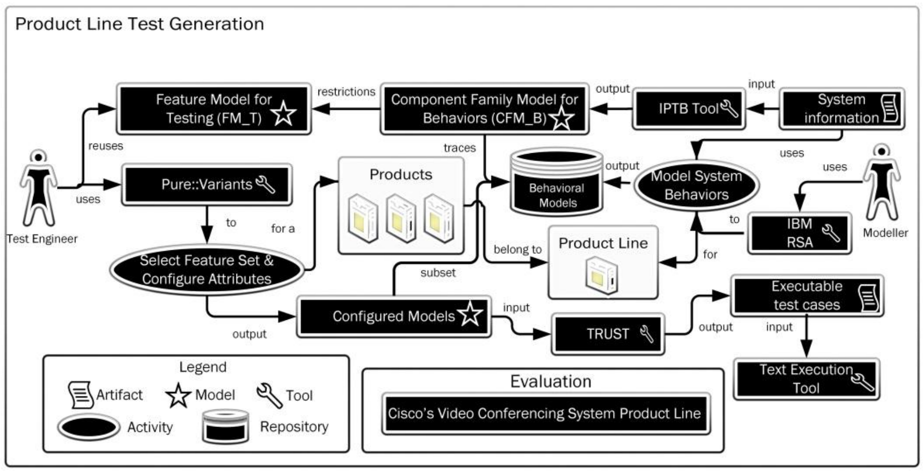

Fig. 3. Overview of test generation methodology

It is well known that major effort to apply MBT in practice is to develop models for SUT systems in order to generate test cases [7]. In our previous work [8], four types of variability for a product line were captured using standard UML class diagrams, UML state machine diagrams, aspect class diagrams and aspect state machines, which were developed by a commercial tool called IBM Rational Software Architect (RSA). All these models are stored in a repository *Behavioral Models* and used for test case generation after configuring them for each new product (Fig. 3). However, using such models for generation, test engineers are required to be familiar with concepts of all the behavioral models (e.g., UML class diagram). To ease the adoption of MBT in

[3] http://www.sencha.com/products/extjs [4] http://www.splot-research.org/

practice, we proposed a test generation methodology using FM and CFM to shield test engineers from all the above modeling expertise as shown in Fig. 3 [7].

More specifically, we first reused the same FM_T (Section 2.1) to capture the commonalities and variabilities of a product line. Second, a Component Family Model for Behaviors (CFM_B) is developed to associate the behavioral models in the repository from system information (e.g., API information and system states). Notice CFM_B can be built automatically using a tool we developed namely Import Plugin and Transformation for Behaviors (IPTB). By linking CFM_B and FM_T via restrictions, test engineers are only required to perform selection and configure related parameters in FM_T and all relevant behavioral models will be selected and configured automatically. Finally, the configured models will be given as in input to a tool called TRansformation-based tool for Uml-baSed Testing (TRUST) for generating executable test cases used to test the SUT system through the test execution tool [7].

The test generation methodology was also applied to the Saturn product line and its four products. The results showed the complexity of configuration can be reduced significantly (on average 66.7% modeling effort is reduced as compared with [8]) [7].

References

1. Benavides, D., Segura, S., and Cortés, A. R.: Automated analysis of feature models 20 years later. A literature review. Information Systems (35), pp. 615–636. 2010.
2. Wang, S., Gotlieb, A., Ali, S., and Liaaen, M.: Automated Selection of Test Cases using Feature Model: An Industrial Case Study. In Proceedings of the ACM International Conference of Model-Driven Engineering Languages and Systems (MODELS), pp. 237-253. 2013.
3. Wang, S., Gotlieb, A., Liaaen, M., and Briand, L.C.: Automatic Selection of Test Execution Plans from a Video Conference System Product Line. In Proceedings of the MODELS Workshop VARiability for You (VARY' 12), pp. 32-37, 2012.
4. Wang, S., Ali, S., and Gotlieb, A.: Automated Search-Based Test Suite Minimization in Product Lines: An Empirical Study. Technical Report (2012-28), Simula Research Laboratory, 2013. (https://simula.no/publications/TR2012-28).
5. Wang, S., Ali, S., and Gotlieb, A.: Minimizing Test Suites in Software Product Lines Using Weighted-based Genetic Algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1493-1500. 2013.
6. Wang, S., Ali, S., and Gotlieb, A.: Empirically Evaluating Three Weight Strategies with Search Algorithms for Product Line Test Minimization. Technical Report (2013-01), Simula Research Laboratory, 2013. (https://simula.no/publications/TR2013-01).
7. Wang, S., Ali, S., Yue, T., and Liaaen, M.: Using Feature Model to Support Model-Based Testing of Product Lines: An Industrial Case Study. In Proceedings of the International Conference of Quality Software (QSIC), pp. 75-84. 2013.
8. Ali, S., Yue, T., Briand, L.C., and Walawege, S: A product line modeling and configuration methodology to support model-based testing: an industrial case study. In Proceedings of the ACM International Conference Model Driven Engineering Languages and Systems (MODELS), pp. 726-742. 2012.