Generic Adaptable Test Cases for Software Product Line Testing

Suriya Priya R Asaithambi*
School Of Computing; National University of Singapore
suria@nus.edu.sg

Stan Jarzabek
Associate Professor
School Of Computing; National University of Singapore
stan@comp.nus.edu.sg

Summary

This research study is about constructing **"generic adaptable test cases"** to counter test case libraries explosion problem. Our work focuses on effort reduction via systematic reuse of generic test assets by taking advantage of common aspects and predicted variability in test cases. We envision that the proposed approach to organizing test case libraries will be particularly useful in the context of Software Product Line Testing (SPLT). By exploring strategies for generic test cases, I hope to address problems of domain-level testing. Our work will investigate existing testing (SPLT) practices in variability management context by conducting empirical studies. We plan to synthesize principles for "generic test case" design, identify gaps between required and exiting techniques, and finally propose new approach for *generic adaptive test case construction*.

Categories and Subject Descriptors D.2.13 [**Reusable Software**]: Domain Engineering, Reusable Libraries, Software Product Line Testing

Keywords Generic adaptable test cases; software product line testing

1. Motication

"A **software product line (SPL)** is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [SEI Definition (McGregor 2001)]. SPL is a family of systems designed to take advantage of their *common aspects and predicted variability*.

The essence of Software Product Line (SPL) (Pohl, Böckle et al. 2005) approach is to systematically analyze system variants and build so-called SPL core assets from which system variants can be developed and maintained in cost-effective way. Test cases form an important part of SPL core asset. Further, testing of core assets is considered critical because a fault within certain functionality can spread over thousands of products which reuse this functionality. Thus, it is important to prioritize and thoroughly test SPL core assets by taking advantage of reusability.

In (Myers 2011) single system development, testing consumes between 35% and 50% of the development costs. In SPL context, testing core assets is a challenge. SPL testing (SPLT) is executed at two levels namely: domain and application testing. ***Domain testing*** is responsible for the validation and verification of reusable components (SPL core assets). Properties validated at the core asset level would also apply to system variants, eliminating the need to re-test them for individual system variants. The challenge is to test parameterized software components without instantiation. Thus domain testing has many open challenges. ***Application testing*** will reuse the test assets created from domain testing heavily; it focuses on testing individual products with all needed variability bounded to appropriate product variant choices.

Our work focuses on test effort reduction through the systematic reuse of generic test assets by taking advantage of product line common aspects and predicted variability. As generic test cases reflect properties of SPL core assets, by exploring strategies for generic test cases, I hope to address problems of domain-level testing.

2. Problem Description

In single system engineering, test artifacts [Ref: IEEE 1998] are deliverables from the testing process. ***Test artifacts*** can be classified as non-executable artifacts (such as test plans, test model, test strategies and test reports) and executable artifacts (such as test cases, test data sets and test scripts). A ***test case*** typically validates whether certain aspect of system specification is correctly realized in the implementation. ***Result*** is a verdict (pass or fail) documented inside test result report. In SPL domain testing context, test plan enumerates testing activities, elicits plan for effort/resource consumption and importantly selects which common and variable test cases are to be executed based on accumulated variants information. The test summary report includes additional information such as variant description, its relation to test cases and a classification (domain or application defect). Thus test plan and test summary reports are less impacted by variability management in comparison to test cases.

Derivation of test case for core assets product families is difficult owing to presence of variability. Each variation

Copyright is held by the author/owner(s).
SPLASH'12, October 19–26, 2012, Tuscon, Arizona, USA.
ACM 978-1-4503-1563-0/12/10.

point presents multiple behaviours to be tested. Industry projects can easily incorporate thousands of variable features and configuration parameters. With continuous evolution of projects, features gets added, modified and removed over time – maintenance of such test case libraries is a research problem worth investigation. In domain testing the key challenge is to test unbound variant points. In application testing the challenge is testing correct binding of variant points against selected product.

For example, if SPL contains 16 feature variants, then it is theoretically possible to derive $2^{16} = 65536$ variant combinations. Thus even a small number of feature variants can results in combinatorial explosion of variants**.** ***Combinatorial explosion of test case libraries*** is caused by the need to test individual variants. This need can be reduced if we could exploit the fact that test cases for different product variants are similar, in the same way as respective products are similar.

The example shown above is a simple acceptance test written in selenium tool for testing a particular scenario

```
public class OpenHomePage {
 . . .
 WebDriver driver = new FirefoxDriver();
 driver.get("http://www.mycompany.com/home");
 WebElement query =  driver.findElement(By.name("q"));
 . . .
}
```

Example: Acceptance test for firefox browser

inside Firefox browser. Similar test are used to ensure compatibility for other browsers. Maintaining this test case as multiple copies is complicated. SPLT artifacts comprises of representation such as natural language, programming language and scripting language. Thus managing variants among SPL Test artifacts using a language neutral mechanism is a key success factor.

Generic adaptive test case design attempts to directly exploit the fact that test cases for system variants form groups of similar test cases. Our design promotes parameterization techniques for building generic, reusable and modifiable generic test cases. For example, replace the driver variant point above with appropriate frame ***<<Browser>>.*** The mechanism complements and extends mechanisms supported by SPL's programming language.

From literature survey on SPLT, limitations of current approaches are: (1) Existing approaches are more focused on non-executable test artifacts. (2)There is no formal classification of test artifacts in terms of nature of variability. (3)Very few specific techniques are available for executable test artifacts.

2.1 Motivating research questions (RQs)

RQ1: How to save time and increase productivity using generic test artifacts? In SPL Variation points are often source of faults. Testing all variants of SPL core assets a priori is usually impossible for all but the simple cases. Formulating effective "generic test cases" that would minimize efforts and increase productivity is essential.

RQ2: How do we assemble generic test artifacts, that tests commonalities and preserve variation in domain testing? The study (Engström and Runeson 2011) highlights need for new techniques addressing variability preservation and commonality testing. The study also reveals a trend in increase of test automation recently.

RQ3: How is generic test artifacts managed at different levels and phases? Our research will focus on generic test case construction parallel to core asset creation and before application engineering.

The intent of our research work is to propose a new approach that avoids combinatorial explosion via use of generic adaptive (domain) test cases that preserves variability. Related Work

2.2 Summary of current generic test cases research:

- *Kolb and Muthig (Kolb and Muthig 2006) discuss the importance and complexity of testing a SPL and component based systems. They promote the need for generic test cases.*
- *McGregor (McGregor 2001) creates generic test cases from the use-case scenarios. The variability combination is resolved using orthogonal arrays technique.*
- *For legacy systems, (Geppert, Li et al. 2004) obtained a family of generic test cases by generalizing(using decision tree) existing (or new) test cases driven by the parameters of variation of the commonality analysis.*
- *In model driven SPLT (Reuys, Kamsties et al. 2005), state chart describes a generic test case with variant point as Boolean expression.*
- *CAFÉ project (Bayer, Flege et al. 1999) presents a method called ScenTeD (Scenario based Test Derivation for product family testing) that addresses generic test cases with respect to system and integration testing. It supports the derivation of generic test case from requirements and architecture information at the domain engineering level.*

Thus current research lacks language neutral techniques to address generic test case. The understandings and challenges are well established. But contibutions are either ideas or partial implementations targeted on specific modeling language or notations. Research (Engström and Runeson 2011) shows that empirical evaluations are sparse in the context of industrial projects.

3. Proposed Work

Our proposed technique works in two steps. The first step is to analyze and craft appropriate generic test case artifacts. Additional inputs inferring the type of core asset and nature of variant points are provided to help choosing suitable mechanism(s). The outcome will be a 'generic test case specification' constructed preserving the variations present. The second step is to derive product specific test cases from the generic test artifacts with appropriate variation points being bound.

Our approach primarily contributes to the first step of constructing generic adaptive test cases as shown in figure 1. The *novelty* of our approach is that generative technique we propose being language/application/domain independent. Our approach can manage variations, propagating changes across all the artifacts. Singular mechanisms have their own merits and limitations and handles only one type of variability. Such mechanisms are simple, cost-effective and work well only for small feature sets. In reality, test artifacts are complex to be dealt using single technique. There are customizations at file level, domain level and method level. Thus mixed techniques are more appropriate.

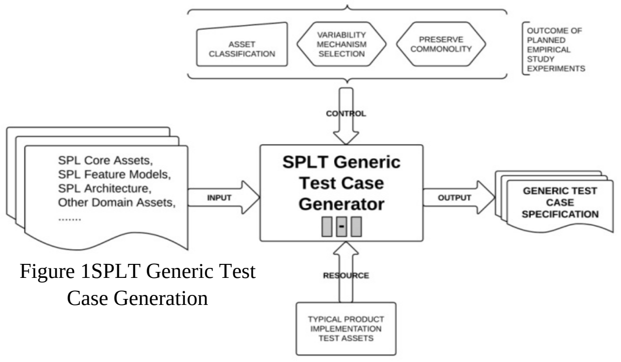
ASSET CLASSIFICATION
VARIABILITY MECHANISM SELECTION
PRESERVE COMMONOLITY
OUTCOME OF PLANNED EMPIRICAL STUDY EXPERIMENTS
CONTROL
SPL Core Assets, SPL Feature Models, SPL Architecture, Other Domain Assets, ...
INPUT
SPLT Generic Test Case Generator
OUTPUT
GENERIC TEST CASE SPECIFICATION
RESOURCE
TYPICAL PRODUCT IMPLEMENTATION TEST ASSETS

Figure 1SPLT Generic Test Case Generation

3.1 Generic Test Cases under Study

Test cases approaches could be black-box or white-box and implementation could be manual or automated. We select two kinds of test case specifications and provide relevant techniques for generic test case. (1) Unit Testing: conducted by developer on individual code components, usually automated white box testing. (3)Acceptance Test conducted by business user on final product, usually manual and black box testing.

4. Methodology

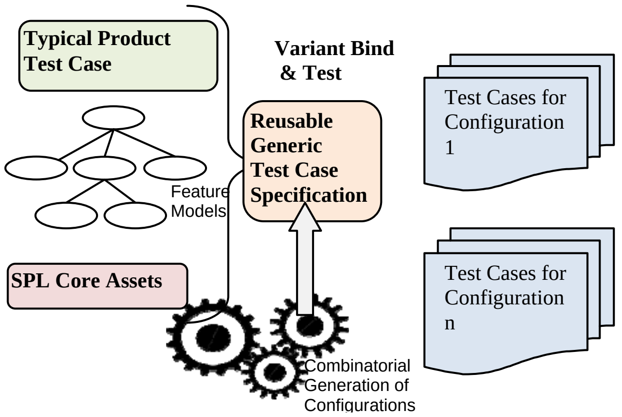
Typical Product Test Case
Variant Bind & Test
Reusable Generic Test Case Specification
Test Cases for Configuration 1
Feature Models
SPL Core Assets
Test Cases for Configuration n
Combinatorial Generation of Configurations

Figure 2: Overview of our contribution

Literature survey involved intensive review of journals, proceedings, projects, and Internet resources related to the SPLT literature. The main research idea in product lines testing is to reuse test case and related artifacts throughout the entire product lines instead of testing every application as an independent software product. It is therefore, important to create proper testing artifacts in SPL as core assets using reuse principles.

Our research study is planned to be conducted in three phases. *Phase one:* Conduct empirical studies identified sets of test artifacts. Observe different types of variability occurring in domain testing, document the variant point representation and draw conclusion regarding generic test artifacts possiblity. *Phase two:* Classify and propose possible variability management techniques for different test artifacts. Propose a systematic method that identifies test case clones, understands the nature of variability and treats with mixed-strategy based reuse approach. The generic adaptive test cases will be built and maintained using generative reuse technique. *Phase three:* Evaluate the approach in qualitative and quantitative ways, by conducting controlled experiment. Discuss its strengths and weaknesses.

The following are needful activities: (1) Define research question (2) Detailed Literature Study: to find out what is already known before trying to answer research goal. (3) Create Theoretical Model to conceptualize the problem stated in research question. (4) Identify possible open source projects. Perform initial empirical studies and identify research gaps. Depending on gaps, one research method will be selected **[RQ2]**. (5) The researcher injects practices directly into pilot project and observes before and after reactions. Data is collected with research instrument, for example test artifacts review **[RQ3]**. (6) Conclusions can be drawn statistically or analytically. Consideration will be given to reliability, validity and threats to validity (internal, external) **[RQ1]**.

References

[1] Bayer, J., O. Flege, et al. (1999). PuLSE: a methodology to develop software product lines. Proceedings of the 1999 symposium on Software reusability. New York, NY, USA.

[2] Engström, E. and P. Runeson (2011). "Software product line testing – A systematic mapping study." Information and Software Technology.

[3] Geppert, B., J. Li, et al. (2004). Towards Generating Acceptance Tests for Product Lines. Software Reuse: Methods, Techniques, and Tools. J. Bosch and C. Krueger, Springer Berlin, Heidelberg.

[4] Kolb, R. and D. Muthig (2006). Making testing product lines more efficient by improving the testability of product line architectures. Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis. New York, NY, USA, ACM.

[5] McGregor, J. (2001 & 2010). "Testing a Software Product Line" Software Engineering Institute.

[6] Glenford J. Myers, Corey Sandler, Tom Badgett (2011). "The Art of Software Testing." John Wiley andSons.

[7] Pohl, K., G. Böckle, et al. (2005). Software Product Line Engineering: Foundations, Principles, and Techniques, Birkhäuser.

[8] Reuys, A., E. Kamsties, et al. (2005). Model-Based System Testing of Software Product Families. Advanced Information Systems Engineering. O. Pastor and J. Falcãoe Cunha, Springer.