Information and Software Technology 56 (2014) 1183–1199

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

On strategies for testing software product lines: A systematic literature review

Ivan do Carmo Machado [a,*], John D. McGregor [b], Yguaratã Cerqueira Cavalcanti [c], Eduardo Santana de Almeida [a]

[a] Computer Science Department, Federal University of Bahia, UFBA, Salvador, Brazil
[b] School of Computing, Clemson University, Clemson, SC, USA
[c] Federal Data Processing Service, SERPRO, Florianópolis, Brazil

ARTICLE INFO

Article history:
Received 4 December 2012
Received in revised form 5 March 2014
Accepted 2 April 2014
Available online 13 April 2014

Keywords:
Software product lines
Software testing
Software quality
Systematic literature review

ABSTRACT

Context: Testing plays an important role in the quality assurance process for software product line engineering. There are many opportunities for economies of scope and scale in the testing activities, but techniques that can take advantage of these opportunities are still needed.
Objective: The objective of this study is to identify testing strategies that have the potential to achieve these economies, and to provide a synthesis of available research on SPL testing strategies, to be applied towards reaching higher defect detection rates and reduced quality assurance effort.
Method: We performed a literature review of two hundred seventy-six studies published from the year 1998 up to the 1st semester of 2013. We used several filters to focus the review on the most relevant studies and we give detailed analyses of the core set of studies.
Results: The analysis of the reported strategies comprised two fundamental aspects for software product line testing: the selection of products for testing, and the actual test of products. Our findings indicate that the literature offers a large number of techniques to cope with such aspects. However, there is a lack of reports on realistic industrial experiences, which limits the inferences that can be drawn.
Conclusion: This study showed a number of leveraged strategies that can support both the selection of products, and the actual testing of products. Future research should also benefit from the problems and advantages identified in this study.

Contents

* Corresponding author. Tel.: +55 71 9183 9735.
E-mail addresses: ivanmachado@dcc.ufba.br (I.d.C. Machado), johnmc@cs.clemson.edu (J.D. McGregor), ycc@cin.ufpe.br (Y.C. Cavalcanti), esa@dcc.ufba.br (E.S. de Almeida).

http://dx.doi.org/10.1016/j.infsof.2014.04.002
0950-5849/

1. Introduction

Software product line (SPL) engineering has proved to be an efficient and effective strategy to achieve economies of scale and scope, by exploiting commonalities and managing variation among products [1]. It is based on the idea that assets, built on the basis of a common design, can be configured and composed in different ways, enabling the creation of a diversity of products, in a shortened building period. Such a software development paradigm leads companies to achieve remarkable results such as substantial cost savings, reduction of time to market, and large productivity gains [2].

The systematic variability management is a fundamental characteristic of a SPL. This practice is what mainly distinguishes this from other software development strategies such as single system development. Variation points identify places in design and code where individual variants can be inserted. When assets designed with variation points are composed, with specific variants selected for use at each variation point, a large number of products can be built leading to what is often referred to as a combinatorial explosion in the number of variant products [3]. Therefore, managing variations at different levels of abstraction and across all generic development assets might be a cumbersome and complex task [4].

In a SPL, features are basic building blocks for specifying products. They can be defined as the distinctive characteristics of a system [5]. Feature modeling has become the de facto standard variability model, as it provides a compact representation of all products of a SPL in terms of their features [6]. Feature models represent the products in a SPL by means of dependencies among features and interrelationships among variation points [7].

Guaranteeing that every feature in a SPL will work as expected, and ensuring that combinations of features will work in each product is often problematic because of the high costs involved. Exhaustive testing is seldom feasible in any development process. This is particularly infeasible in SPL due to the variability in features, due to the many input variables.

A scoping review, carried out as background to the present review [8], revealed two independent but complementary interests a SPL testing strategy should handle, as follows:

- Firstly, it is necessary to check the feature interaction coverage, i.e., when checking the properties or configurations of a SPL, every feature combination have to be consistent with the specification and must not violate the stated constraints.
- Secondly, it is necessary to check the set of correctness properties of each product. Given that a built software artifact can be used by a range of products, an uncovered defect may be propagated to the many products that include it.

The *first interest* considers testing generation as a systematic selection of a representative set of product instances, comprising a subset of all possible configurations in a SPL. The main idea is to reduce the testing space. Fig. 1 illustrates the first interest. It shows that test cases refer to product configurations, i.e., a test case is responsible for testing whether an instance of the feature model (or whatever represents the variability in a SPL) is valid or not.

There are two main inputs to consider for this *interest*: **a set of product requirements** and the **quality of the variability model under test**. The role of requirements is to establish what functionalities a product instance should encompass. Regarding the quality of the variability model, through the analysis of its consistency, we could ensure that the produced models are complete and correct, in the sense that there are no conflicting restrictions between features, and that all the important distinctions and differences in a domain are covered by the model.

The *second interest* focuses on performing testing on end-product functionalities. Such an interest deals with the systematic reuse of test assets and results, as a means to reduce the overall effort, and avoid retesting of already tested features, while being effective at revealing faults. Test assets are designed to test the functionalities of features that will compose the products.

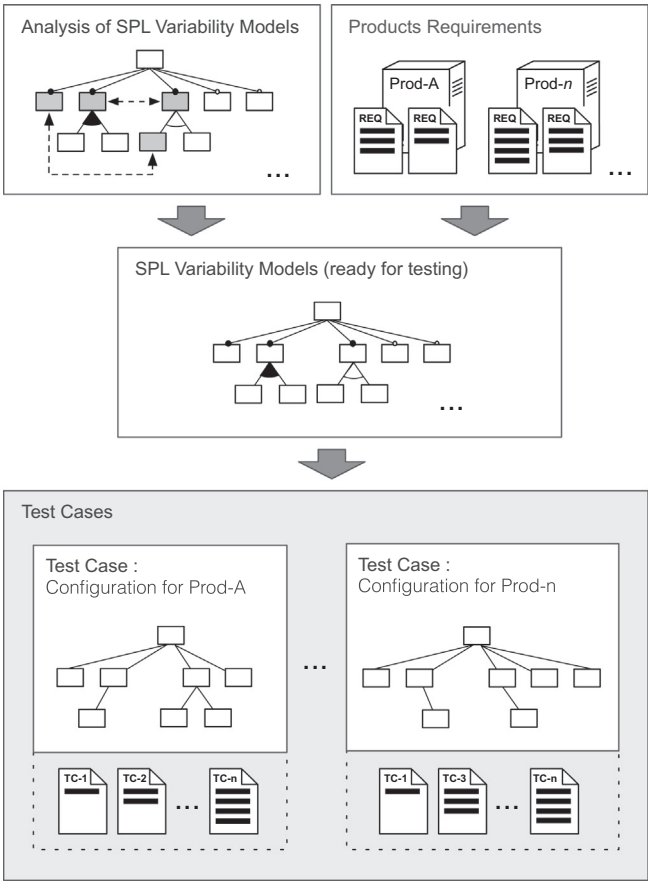

Fig. 1. SPL Testing interest: selection of product instances to test.