A Risk-Driven Approach for Efficiently Testing Software Product Lines

Ronny Kolb

Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6, 67661 Kaiserslautern, Germany
+49 (0) 6301 707 – 195
Ronny.Kolb@iese.fraunhofer.de

During the last decade, software product lines have increasingly received attention as they promise to significantly improve the productivity of the software development process and so to reduce both the cost and time of developing and maintaining software systems. In order to achieve the promised improvements, however, a high level of quality of the components reused in the various product line members is required. Therefore, more than for traditional software development, testing as still the most prevalent approach for ensuring quality becomes a very crucial part of every product line effort. Nevertheless, research in the field of testing reusable components and software product lines is still in a very early phase and there is actually little guidance for product line organizations on how to systematically test their product lines yet. This paper discusses what the challenges in testing software product lines are and outlines an approach that enables to test product lines more effectively and efficiently.

Keywords: Product Lines, Testing, Test Planning, Test Case Design

1 Introduction

Software product lines are a rather new and very promising approach to software development that enables organizations developing a number of similar products in the same application domain not only to reduce development and maintenance costs but also to increase productivity and shorten time-to-market by facilitating reuse of software artifacts. According to [4], a software product line is "a set of software-intensive systems sharing a common, managed set of features that satisfy 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." Core assets are reusable software artifacts, such as components, architectures, requirements specifications, test cases or other models and documents, which are generic enough to support the development of the different products in a software product line.

The product line approach to software development exploits the commonalities among a set of products and controls the variabilities among those products. As depicted in Figure 1, the overall development life cycle is split into two concurrent phases in the product line approach: development for reuse, which is done in the domain engineering phase, and development with reuse, which is done in the

application engineering phase. During domain engineering, the product line infrastructure, or core asset base, consisting of a set of reusable assets from which the individual software products in the product line are built by product development teams during application engineering is developed.

Although the move towards software product lines offers many promises for improved productivity, it also highlights the need for more effective methods and techniques for ensuring the quality of software components that are intended for reuse. As quality problems in a component not only lead to an end product with low quality but also propagate this low quality into all the products in the product line depending on it, a high level of component quality is required in order to successfully develop product lines. Therefore, testing, as still the most prevalent approach for ensuring software quality, becomes a very crucial part of every product line effort. Not only have testing issues become more critical for software product lines, but also more complex due to the special properties of product lines such as genericity of software components. Rather than on effective methods and techniques for testing reusable components and product lines, however, most research has been focusing on the analysis, design, and implementation of software product lines and reusable components. In fact, only some work (e.g., [4], [5], [6], [7], and [8]) addresses the unique testing problems that arise in a product line context and adequate techniques, methods, models, and tools for testing software product lines are still missing. Hence, test techniques and strategies for traditional single system development are usually applied in practice even though they are mostly inadequate and ineffective for detecting errors. Compared to the reductions in development and maintenance efforts, consequently, there has been no major reduction of the overall testing effort for software product lines so far.

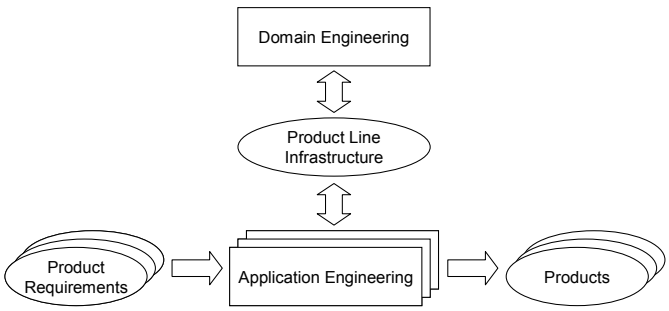


Fig. 1. Product Line Engineering

This paper discusses what the problems and challenges in testing software product lines are and outlines an approach that enables product line organizations to test product lines more effectively and efficiently compared to currently practiced approaches. The remainder of this paper is structured as follows. Section 2 discusses testing in the context of software product lines and summarizes the problems and challenges in testing product lines with currently available techniques and strategies. In Section 3, an approach that addresses the identified problems and challenges and that will support product line organizations in general and test managers and testers in particular in testing software product lines is outlined. Finally, Section 4 summarizes the paper and draws some conclusions.

2 Testing Product Lines

Testing is the process of executing a component or system on a set of test cases and comparing the actual results with the expected results [2]. It is still a very labor-intensive and thus very expensive and time-consuming activity of the software development process. In fact, testing can consume up to 50 percent of the overall software development costs. This is also because effective testing requires a large number of tests to be developed, executed, and examined. As resources and time for testing are usually limited in practice, testing is a tradeoff between budget, time, and quality in most cases.

Due to the split of the development process into development for reuse and development with reuse, the test process for software product lines differs from that of conventional systems, but is very similar to component-based software. In the context of product lines, the testing process primarily encompasses testing the generic components of the product line infrastructure that are used across various product line members, as well as testing the product-specific components and the members of the product line that have been derived from the product line infrastructure. As a consequence, the overall test process, as shown in Figure 2, is split into the product line infrastructure, or core asset, test process and the test processes for the various products in the software product line. Not only is the testing process for software product lines split into two distinct processes, but testing may also be distributed across different parts of a product line organization, that is infrastructure development team and product development teams.

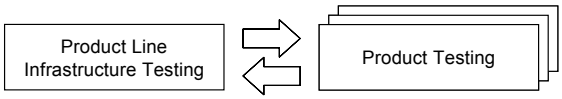


Fig. 2. Product Line Test Process

Because of the large-scale reuse of components and the distinction between development for reuse and development with reuse, testing product lines is in many aspects similar to testing component-based software systems [3][9]. In essence, therefore, the same techniques used for testing in the context of component-based software can be applied for testing software product lines. Because of the very nature of product lines, however, these techniques alone are not sufficient to efficiently and adequately test product lines.

The main problem in testing product lines is caused by the generic nature of the components in the product line infrastructure. The genericity of the product line components makes testing all their functionality problematic as it is often impossible, or at least economically infeasible, in practice to test all possible combinations which result from the large number of variation points and possible variants at each variation point. For example, a component with just 10 variation points and only 3 possible variants at each variation point results in 59,049 variations that need to be tested. Generally speaking, the number of combinations to be tested grows exponentially. Hence, test techniques that take into account the generic aspect of software components and that can handle variability are required. The techniques particularly have to address the following problems: which variants of a generic component

should be tested in which order and how many possible variants should be tested. The genericity of components also causes problems in testing configurations of a number of generic components. Other problems in testing product lines are caused by the potentially large number of products making use of generic product line components and because of interdependencies between product line components and product-specific components. Especially, it is very challenging to efficiently and effectively test all products making use of a component every time the component has been changed. From an economical point of view, another problem in testing product lines as currently practiced is that generally a lot of time and resources are wasted in testing the same thing over and over again. This is, among others, caused by the fact that the reuse of test cases and test results is more the exception than the rule although it holds the potential of reducing the cost and time for testing in a product line context.

Testing software product lines using traditional techniques and strategies is a very challenging process and more comprehensive techniques are required in order to test product lines in general and generic components in particular adequately and in a cost-effective manner. As pointed out by McGregor [6], testing product lines requires careful planning and a well-defined process. The major problems in testing product lines are the large number of variations, redundant work, the interplay between generic components and product-specific components, and regression testing. Although software product lines pose a number of challenges for testing, they also have some unique characteristics, such as an explicit software architecture and the amount of documentation typically available, that ease testing. In addition, it is generally not only possible to access the source code of the generic product line components when needed, but also to have access to the personnel and expertise used in the creation of these components.

3 Approach

As discussed in the previous section, the testing of software product lines using currently available strategies, methods, and techniques poses a number of unique challenges and problems for product line organizations. The potential benefits of the product line approach can even be lost if the testing process does not take into account the issues specific to software product lines. Therefore, an approach will be developed and validated that enables to test product lines more effectively and efficiently. The approach aims at reducing the costs, time, and effort required for testing product lines while at the same time improving test coverage and hence quality.

Focusing on test planning and test case design, the approach supports product line organizations in planning and preparation for testing in a product line context and provides techniques for derivation and management of reusable test cases for generic components and the products depending on it. The approach provides guidance in planning product line test activities and in defining an appropriate test strategy for a product line based on quality requirements, organizational characteristics, and context constraints such as available resources and given schedules. This also includes guidance in choosing suitable test techniques and in defining when and how to apply each technique. In order to optimally use the limited time and resources available for testing, the test effort is focused – based on identified and prioritized risks – on the

critical areas of a component or product in a product line. As shown in Figure 3, basic test strategies, test techniques, and test coverage criteria are used as an input for defining a product line test strategy. The basic strategies can be the basis for defining project-specific test strategies. They primarily describe how much is tested, what is tested, which techniques are used for testing, when is tested, and who performs the tests in a software product line.

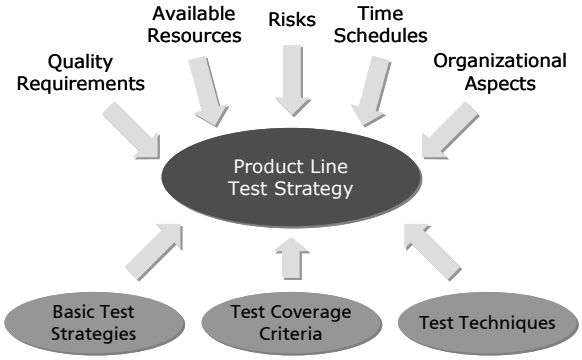


Fig. 3. Product Line Test Approach

As currently available test techniques do not take into account the specificities of software product lines and generic components, a new technique for efficiently designing effective test cases in the context of product lines will be provided. The technique particularly assists in defining and managing generic and thus reusable test cases, as well as in the instantiation of those test cases for particular components and products. To determine the extent to which a generic component, product-specific component, or product in a product line has been tested, specific coverage criteria for software product lines will be provided. The contributions of the test approach for product lines are summarized in Table 1.

Table 1. Contributions of the Product Line Test Approach

Contribution	Effect
Support in systematically defining a product line test strategy with respect to available resources, given time schedules, quality requirements, and taking into account risks	Product line organizations can define a test strategy and plan testing activities for a given context in a repeatable manner. The limited resources are used to the best possible degree for testing the product line and its members.
Basic product line test strategies	The basic strategies provide a good basis for defining project-specific test strategies.
Guidance in choosing suitable test techniques and in defining when and how to apply each technique	Even without much experience, product line organizations can choose the test technique most appropriate for the given context
Test coverage criteria specific for software product lines	Coverage criteria allow to determine the extent to which a subject has been tested. They help in planning testing and in deciding when to stop testing.

Contribution	Effect
Technique for the definition and instantiation of generic test cases	Generic test cases are reusable and help minimizing the effort for creating test cases as well as in avoiding redundant work.

4 Conclusions

This paper has discussed the problem of testing software product lines and outlined an approach for testing product lines more effectively and efficiently compared to currently practiced approaches. The approach focuses on the activities of test planning and test case design. It provides guidance in defining a test strategy for a given context and supports in the efficient definition of effective test cases for the generic product line components and the products built using them. Being risk-driven, the approach uses identified and prioritized risks for focusing test effort and so making the best use of limited resources. As automation is a good way to cut down time and cost, a tool will support the management and instantiation of generic test cases. The applicability as well as effectiveness and efficiency of the developed approach will be validated by means of case studies and a controlled experiment. The developed and validated product line test approach will be integrated into PuLSE[1], the customizable product line engineering methodology developed at Fraunhofer Institute for Experimental Software Engineering (IESE) [1].

References

[1] J. Bayer et al., "PuLSE: A Methodology to Develop Software Product Lines", In Proceedings of the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR'99), pp. 122–131, Los Angeles, CA, May 1999.

[2] B. Beizer, Software Testing Techniques. Second Edition, New York: Van Nostrad Reinhold, 1990.

[3] A. Bhor, "Component Testing Strategies." Technical Report UCI-ICS-02-06, University of California, Irvine, June 2001.

[4] P. Clements and L. M. Northrop, Software Product Lines: Practices and Patterns. Addison-Wesley, 2001.

[5] R. Kauppinen, "Testing Framework-Based Software Product Lines", Department of Computer Science, University of Helsinki, Technical Report, October 2002.

[6] J. D. McGregor, "Testing a Software Product Line." Technical Report, CMU/SEI-2001-TR-022, Software Engineering Institute, Carnegie Mellon University, December 2001.

[7] H. Muccini and A. van der Hoek, "Towards Testing Product Line Architectures", In ETAPS 2003 Workshop Test and Analysis of Component Based Systems (TaCOS), Warsaw, Poland, April 2003.

[8] C. Nebut, S. Pickin, Y. Le Traon, and J.-M. Jézéquel. "Reusable Test Requirements for UML-Modeled Product Lines", In Proceedings of the Workshop on Requirements Engineering for Product Lines (REPL'02), Essen, Germany, September 2002.

[9] E. J. Weyuker, "Testing Component-Based Software: A Cautionary Tale." IEEE Software 15, 5 (September/October 1998): 54–59.

[1] PuLSE™ is a registered trademark of the Fraunhofer IESE.