INF9500 Project Report: An evidence-based decision of strategy for software product line testing

Martin Fagereng Johansen

December 13, 2010

Abstract

We should employ a strategy to test an industrial product line. Testing products individually is redundant for product lines since the products share a considerable amount of code. This document makes a decision that can be presented to industry about which product line testing strategy they should employ based on the best available evidence out there. Industry can use the evidence to decide on a product line testing strategy and get a reasonable estimate on how well it will perform.

Contents

1 Introduction

Take the following situation. We are asked for advice by a company which has a software product line, consisting of say 10 products, which has a considerable part of software in common. They are currently testing their system product by product without exploiting the commonality between the systems. This, they say, causes redundancy in their efforts which should be possible to avoid. They have tried to look into the literature but finds it confusing.

We have decided to employ an evidence-based decision process. Basing our decisions on the best possible evidence is a good strategy for ensuring that the approach is applicable in practice, and that we can get a number on the expected performance of the approach.

1.1 Problem formulation

Which strategy for developing a test suite for a software product line allows test engineers to best utilize the specification of similarities and differences (e.g. a variability model)? The technique must at least reduce the effort to develop a test suite below the effort required to develop one for each product by itself.

1.2 Clarification of problem formulation

It is a common case in software development to have to maintain a collection of applications which share a significant portion of code with each other. When one approaches such problems in a systematic way, it is called software product line engineering. The term *product lines* comes from the automobile industry where the need for mass customization of different models was first encountered.

Since product line share a large amount of code, it should be possible to improve testing by utilizing the specification of the similarities and differences between the products. Such a specification may be called a variability model.

A strategy is how a human or a computer should attack a problem systematically. A strategy for testing a software product line is a descriptions of which steps a test engineering should follow to reduce the effort on testing the product line significantly below the effort required to test each product individually. This strategy does not have to be formal, it just have to be executed by a test engineer, assuming he has the skills required to test individual products.

1.3 Motivation

This problem is a part of my PhD-project on improving the testing of software product lines. Before starting to researching a problem, it is typical for a PhD-student to read up on the current state of the art in order to not reinvent the wheel or try to solve problems already solved. Also, understanding which testing strategy is the best provides a good starting point for research, and also a good comparison to future results. Newly developed results should be better than the best know approach in some respects to be of interest.

2 Method

As a member of an EU project of which one goal is to develop a novel approach for product line testing, I have some insights into the field of software product line engineering. I knew that SEI at Carnegie Mellon has been a major advocate of software product line engineering since

the beginning. Thus, I decided to look up on their website for research related to my problem (section 2.1). In addition to this, a major commercial company within software product line engineering, BigLever, lists some case studies on their website as a part of promoting product line engineering (section 2.2).

Books are also a great source of quality material. Major publishers have high quality requirements, and review the publications in detail. Amazon is a good source for finding books (Section 2.3).

There are two major conferences in the field and one workshop. I ensured to take into account the proceedings of Software product line conferences (SPLC), Practical Product Lines (PPL) and Workshop on Software Product Line Testing (SPLiT) (section 2.5 and 2.4).

I then searched through some of the largest databases of publications online; including Science Direct, IEEE Xplore, IEEE computer society digital library, the ACM digital library, the ISI web of knowledge, Scopus and Google scholar. These were selected based on my experience in searching for literature and from suggestions by the INF9500 course (Section 2.6).

Finally, I found several other surveys. I looked through them to see if they had collected any interesting evidence or what they thought about the available evidence out there (Section 2.7).

For a complete list of considered works, see Appendix A.

2.1 SEI's Catalog of Software Product Lines

An important source of case studies was SEI's Catalog of Software Product Lines[1] and their hall of fame[2]. They maintain a list of case studies with a list of cited improvements in each case study. I read through the cited benefits carefully and noted the case studies with cited benefits of quality and with some discussion of testing.

There are 54 case studies in the catalog, and 114 written reports, of which most are publications, book chapters and technical reports. The hall of fame is the best subset of these cases, currently 18 case studies.

Out of the 54 case studies, 21 were found to be possibly relevant, of which six was chosen to be the most promising. These will be evaluated later in detail.

2.2 BigLever Customer Case Studies

BigLever is one of the few companies that offers a commercial SPL solution. They have a number of case studies listed on their website[3] and on another website they sponsor[4]. I looked through these case studies for any empirics on testing of product lines.

There are eight case studies on BigLever's site, and six case studies on the site which they sponsor. Except the ones already found, there were no new case studies of interest to this evaluation.

2.3 Amazon

I searched Amazon for books on software product lines which might contain case studies or experiments with evidence on testing. The query I used was "software product line". I selected the top ten books returned and looked through the table of contents of the books. There were

[1] `http://www.sei.cmu.edu/productlines/casestudies/catalog/`
[2] `http://splc.net/fame.html`
[3] `http://www.biglever.com/learn/reports.html`
[4] `http://www.softwareproductlines.com/successes/successes.html`

several case studies. Some of the books were unavailable to me, and thus they may contain interesting empirics that I did not have access to during my literature search.

From the books that I did get to look into and except for many repeats and case studies with irrelevant contents, I found the CaVE case study of 2009 (John et al. 2009).

Among the interesting books that were not available is Neto 2010 ”A Regression Testing Approach for Software Product Lines Architectures: Selecting an efficient and effective set of test cases”.

Also, I did not get to look into Kang 2009 ”Applied Software Product Line Engineering”, Gomaa 2004 ”Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures”, Bosch 2000 ”Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach” and Schmölzer 2008 ”Software Product Line Architecture for Enterprise Applications: Principles, Methodologies, and Practices for Model-based SPL Engineering”.

2.4 SPLiT Workshop Proceedings

I searched through the SPLiT proceedings of 2004-2008. This allowed me to identify the case studies, and evaluate if they contain valuable evidence.

One case study found was the Freescale Semiconductor Case Study of 2006. It is presented in Lew 2006 ”Test cost saving and challenges in the implementation of x6 and x8 parallel testing on freescale 16-bit HCS12 micro controller product family” [7].

This paper includes technical discussion which requires specialized knowledge into semiconductors and micro-electronics. Thus, I cannot manage to extract the relevant results from the paper due to a lack of understanding of these topics.

2.5 SPLC and PPL proceedings

These are the two major conferences in product line engineering. Their publications should have been indexed by the search facilities. Thus, looking at their proceedings directly is not so important.

2.6 Search facilities

I searched through several search facilities: IEEE Xplore, The IEEE Computer Society Digital Library, The ACM Digital Library, The ISI web of knowledge, SCOPUS, Science@Direct and Google Scholar.

The search was done in two parts. The first part is to find publications within the field of product line testing, the second search was to specifically find case studies in product line engineering which might contain an evaluation of testing.

2.6.1 Search for publications about SPL testing

Query The following query was used to search. Each search facility provided different syntax for writing searches, but the following is an understandable version of what I wanted to search for: I wanted to find all publications which has in its title: ”product lines” and ”testing”. There are variations over these words used; therefore, ”product lines” got the alternatives ”Product line” and ”Product family”. Testing got the alternatives ”Test”, ”Validation” or ”Verification”. In a syntax similar to what is offered on many of the search facilities, the query looks like this.

```
(
  title("Product lines") or
  title("Product line") or
  title("Product family")
) and (
  title("Testing") or
  title("Validation") or
  title("Verification")
)
```

Science@Direct [5] This search yielded two papers. The most relevant is [5], a journal article which studies the state of the art in product line testing.

IEEE Xplore [6] This search yielded 20 publications, of which 11 were relevant.

Interesting findings here are the Testo AG case study of 2007 and the Silva and Soares experiment 2009.

The Testo AG case study proved to be relevant and will be discussed later.

The Silva and Soares experiment of 2009, titled ”Analyzing structure-based techniques for test coverage on a J2ME software product line”, is presented in [14].

After a closer read, I can say that the paper does not deliver on its promise. It compares the coverage results of using eight different testing strategies for four products, but these products are tested product-by-product. What would have been interesting was if they compared this to an approach that utilizes the knowledge of the variability to not include tests for some of the products. It is only briefly mentioned that they removed some tests, but not how they did it. Hence, this paper does not include reliable empirical evidence possible to use in this report.

The IEEE Computer Society Digital Library [7] I chose the first 30 results of the 100 returned results.

Mallett et al. 2010 presents from interesting initial work, but they say that they will present empirics on the improvements on testing in future work. Also, Nascimento et al. 2008 contains some interesting work with potential in the future.

Denger and Kolb 2006 performs an experiment with analysis which seems to be directly relevant to the problem formulation, but after a closer read I discovered that it was not so relevant. In section 3.5, they say that they performed their experiment on two variants of the same component, which is essentially the same as viewing the component as a product and then testing two variants of it. What it interesting to this report is whether reuse of code in several products can be taken into account to ease testing, and not if common faults can be found when testing one complete product. Thus, this experiment is of little interesting to the problem formulation in this report.

The ACM Digital Library [8] Out of 30 results, eight were found to be relevant. Out of these, the Tevanlinna 2004 survey is interesting.

[5] On `http://www.sciencedirect.com/`, click ”advanced search” and then ”expert search”.

[6] On `http://ieeexplore.ieee.org/`, click on ”advanced search” and then on ”switch to command search”

[7] On `http://www.computer.org/portal/web/csdl` go to search-¿advanced search. Note that there is not a query search available.

[8] On `http://portal.acm.org/`, click on ”Advanced search”, ”search”, ”advanced search” (yes, again)

The ISI web of knowledge [9] Out of 26 results, the papers with one or more citation were chosen.

The new possibly interesting finding is the Reis et al. experiment of 2007, but this paper [10] showed to not contained any empirical evidence after close read.

SCOPUS [10] Out of 69 results, all papers with two or more citations were chosen. We did not find any new relevant papers.

Google Scholar [11] Google scholar gave many, many results, but there is no means of ordering the results. Thus, it is difficult to use Google Scholar for this purpose. It was used to more success for searching for case studies below.

2.6.2 Search for case studies

In this search, I search for the three synonymes for product lines in the title again. But now, we look for case study in the title or in the abstract. It is very common to include these terms in either of these places. In addition we see if there is a mention of testing in the abstract. I include the four key words for testing and search for it in the abstract. The reason is that testing is probably not the main focus of a case study, and if it is, it will surely be included in the abstract. The following is a generic form of the query used to search for case studies.

```
(
  title("Product lines") or
  title("Product line") or
  title("Product family")
) and (
  title("case study") or
  abstract("case study")
) and (
  abstract("Test") or
  abstract("Testing") or
  abstract("Validation") or
  abstract("Verification")
)
```

Science@Direct This yielded three papers, none of which were found to be interesting.

IEEE Xplore This search yielded six results, none of which were new and relevant to our problem.

The IEEE Computer Society Digital Library There were 8 results. The Sharma et al. experiment of 2008 seems to be an interesting experiment. It, [13], contains the result of a questionnaire, establishing the perceived benefit of product line engineering in three companies. It proved to be subjective, and does not give us any empirical evidence for the superiority of a testing strategy for product line engineering.

[9] On `http://www.isiknowledge.com/` click on "advanced search"

[10] On `http://www.scopus.com`, click on "advanced search".

[11] `http://scholar.google.com/`

The ACM Digital Library This search yielded three results, none of which were new and relevant to our problem.

The ISI web of knowledge This yielded five results, none of which were relevant to our problem formulation.

SCOPUS There were 24 results of which 11 were relevant or not found before. Here, the Cabral et al. 2010 experiment was found, and it will be considered in detail later.

Google Scholar This search gave five results, none of which were new and relevant to us.

2.7 Surveys

2.7.1 Engström's Survey from 2010

This survey is titled "Software Product Line Testing - A Systematic Mapping Study" [5] and surveys the current state of the art in software product line testing. She explains what a systematic mapping study is in this quote:

> [Systematic mapping] is an alternative to systematic reviews and could be used if the amount of empirical evidence is too little, or if the topic is too broad, for a systematic review to be feasible.

She concludes that there is a limited share of empirical studies, and that she could not find studies from "Philips, Nokia, Siemens, etc.", something I did not have big problems finding during my literature search.

> Product line testing is a large scale effort and evaluations are costly [73], which is one of the explanations behind the limited share of empirical studies.
>
> [...] empirical evaluations and experience as a minor group (17%).
>
> [...] extensive experience in PL engineering exist within companies (Philips, Nokia, Siemens, etc. [59]) but no studies on testing can be found [73].

Among the references of this survey, I found the Kolb 2006 survey. There were other interesting papers and books references that I did not get the hold of.

2.7.2 Tevanlinna's survey from 2004

This survey is titled "Product family testing: a survey" [16]. It also states that case studies with empirics cannot be found.

> Several companies, such as Philips, Nokia and Siemens, have extensive experience in product family engineering, but published case studies about testing product families cannot be found. Their results are usually in company-specific internal documents.

2.7.3 Kolb's survey from 2006

This survey is titled "Techniques and strategies for testing component-based software and product lines" [8]. It also has a claim about product line testing.

> [...] no significant reduction of the overall testing effort has been seen to date.

3 Analysis

The analysis contains two types of studies: experiments and case studies. The case studies are judged for internal validity only since they do not make a claim of the external validity of their own observed result. There will be an evaluation of the external validity when we take the claims of all the case studies together in the end. The experiment studies will of course be judged for both external and internal validity.

3.1 Market Maker Case Study of 2007

Source Sources are the book chapter by Verlage 2007 "market maker Software AG" [17].

Authors Both Martin Verlage and Thomas Kiesgen are employed at market maker software AG[12] *and* [13].

Martin Verlage holds a PhD in computer science and has been affiliated with academia in the past including University of Kaiserslautern and Fraunhofer Institute for Experimental Software Engineering.

I could not find any additional information on Thomas Kiesgen.

Main claims/results They claim to have reduced the time to market of their products by a factor of 2-4. They broke even for the investment required into product lines after five products. The effort of maintenance was reduced by 60%, and they report to have reduced the cost of quality measured by the products reliability in the field.

Evidence presented The 60% figure was found by developer analysis and according to the authors' sound estimates (page 186). They report that their products are successful in the market place, something that should indicate that they at least did not harm themselves beyond recovery by using software product lines.

Evaluation of the evidence Neither the developer analysis nor the sound estimates of the authors are elaborated, making it impossible to evaluate these claims further. Also, the authors have vested interests in reporting success of their company, but should not have the same vested interest in promoting software product line engineering.

The report that they collaborated with Fraunhofer Institute for Experimental Software Engineering, who developed the method PuLSE, which is used to build their product line.

They might also have improved their products just from the fact that they put a lot of development effort into it. This might have caused beneficial improvements to existing code through beneficial refactorings.

[12] `http://portal.acm.org/citation.cfm?id=1062551`

[13] `http://portal.acm.org/author_page.cfm?id=81100330488&coll=GUIDE&dl=GUIDE&trk=0&CFID=110697981&CFTOKEN=11418631`

The reduction in the time to market for new products combined with the reduced cost of effort is an indication that software product line engineering does at least improve this slightly, and at worst that it does not harm development.

Summary This case study gives a poor backing for the internal validity of their claims. Thus, the claims remain inconclusive and are thus unreliable. There are vested interests for the two authors in telling a good story about their company, and their academic partner have some interests in promoting their research. These are factors that lessen the reliability of the claims. At best, this case study shows that the company did not have significant new problems when adopting software product line engineering, and that they managed to test their product line after introducing this technique without significant problems. If this was not the case, they probably would not have written the report.

3.2 NRO Case Study 2001

Source Cohen 2001 "Control software toolkit: A software product line that controls satellites" [4], a chapter in [2], which is cited by 2000 other publications.

Authors Sholom Cohen is today employed by SEI. I could not find any information on Patrick Donohoe.

Content This case study presents the case of the initial development of the Control Channel Toolkit (CCT), consisting of 101 components with 111 variation points. This asset base is used to build a product line that is developed by NRO which commissioned the development to the Reytheon company. As far as I can tell, there is only one product developed using CCT, and that is something refereed to as Spacecraft C2. They report how they developed CCT and how they developed the first product from the base assets in CCT. They report the benefits of developing the one product from the base assets instead of developing it the normal way, which involves some cut and paste from earlier systems and some additional code. The main claim and results presented next is for the development of this one product from the base assets in CCT instead of developing it without the reusable assets, for which a typical systems is 500 kloc in size.

During domain analysis they identified that the commonality of requirements between products were between 49% and 89%.

Testing of this product line is described over two pages in section 10.3.4, page 469. They do a detailed presentation of their architecture and process in general, which is of good quality.

They report that the costs was higher than normal for the initial development.

Main claims/results CCT was delivered on time and within budget. Smaller development staff required (15 versus 100 for other similar systems.) "Overall costs cut by 50%.", "Overall schedule cut by 50%.", the lines of code was 76% lower than it would have been otherwise, "Quality: One-tenth the typical number of discrepancy (defect) reports for a system of this type. The problems identified were all local ones, with localized fixes, having no ripple effects, and no effect on the architecture."

Evidence presented They present their architecture and how they worked to achieve the results in some detail. This discussion includes a two page description of how they tested it. They state that they can back up their claims because they are measurable. They say to have attributed

the benefits to the product line approach, but does not describe how they attributed these benefits to this approach. They say that they interviewed the developers, but it is not stated how they interviewed them.

Evaluation of the evidence The really interesting thing in this case study would have been an evaluation of the validity of attributing the observed benefits of Table 10.3 (in their paper) to the use of the product line approach. Such an evaluation is not present, which leaves the validity of the claims hard to validate. They do give a good description of how they built the system and how they tested it, which show a certain level of skill in the developers. This does give some increase in the confidence of the claims.

If we did have an evaluation of the attributing of the results to the product line approach, then we could have seen whether some of the benefits are also due to the effort put into developing the reusable assets from the legacy code. But, this would apply to the development of one system. But, they do not present the development of more than one system in the case study. Thus, it is hard to say with confidence that the same observations will show from the second system. In my experience reuse beyond one is a lot harder than reuse beyond, for example, three. In the case of three, the reusable assets would have to be reused as is over three different products, something that is considerably more difficult than using them for only one product. For one product, one can even modify the assets to suite this one product. Doing this for the third product will likely break one of the other products that uses thet asset.

Summary The confidence in the internal validity of the results presented in this case study is low due to two primary things: (1) no reporting of how they manage to attribute the benefits to the product line approach and (2) only one product has been developed this far in the life of the product line. The strength shown in the architecture and testing process gives some increased confidence.

3.3 Dialect Solutions Case Study 2004

Source This case study is published in Staples and Hill 2004 ”Experiences Adopting Software Product Line Development Without a Product Line Architecture” [15]

Authors The authors are Mark Staples of National ICT Australia, a research institute, and Derrick Hill of Dialect Solutions, the company which product this case study is about. Mark Staples work is founded by ”Backing Australia’s Ability initiative”.

Content ”Dialect Solutions develops and supports a collection of Internet payment gateway infrastructure products.” ”There are half-a-dozen [5-7] products in the product set.” They have one for each customer. Previously to using product line engineering, they used Single-System Development with Reuse. At version 1.3 of their set of products they were faced with the alternative up upgrading all their existing products to 1.4, or to adopt the product line approach and upgrade all systems to 2.0. They chose to adopt the product line approach, and ended up spending the same amount of effort they might have spent on upgrading all the systems anyway. This was based on their previous experience with upgrading the products.

They explain how they implemented product line practices for their case, and they discuss problems they faced and how they ended up solving them. They finally reflect on their experience, and what was improved after adopting these practices.

Main claims/results ”[...] unit test costs within core assets can be ”shared” across many products. This has increased overall productivity.” ”PLD [Product Line Development] has helped Dialect Solutions improve development efficiency and product quality.”

Evidence presented

- The effort required to migrate to product line engineering was felt to be equal to the effort required to upgrade the existing system.
- They have about six products in production, one for each of their six customers.
- They present of how they adopted the product line approach and how they solved the various problem related to it.

Evaluation of the evidence The paper is well written, and they give a convincing case for why they had to adopt product line engineering. Their description of how they migrated to product line engineering is to my judgment realistic, and their discussion of problems they faced and how they solved them are also realistic.

They also have six products running in production. A running system has to be complete. Thus they are not speculating when they are reporting improved quality.

This adds to the confidence of their claims.

There are some things that reduces the confidence also. They do not give a number for the improvement. I think it is fair to assume they mean at least a 30% improvement, or else it would be hard to notice. Also, they do not state how they determine that the quality has gone up. Thus, they are in the risk of miss-attributing the improvement to product line development when it was in fact caused by other improvements.

A final thing is that one of the authors is from the company which products they are talking about. This gives him an incentive to talk about the positive aspects of their system, but not necessarily for defending the product line approach. Thus, this is not a significant problem.

Summary All in all, the reliability of the claims have some strong points supporting it, but also some points questioning it, giving a medium confidence level in their claims.

3.4 Philips Medical Systems Case Study 2007

Source This case study is from Schouten 2007 [12].

Authors The author is Gerard Schouten. Schouten seems to be an employee of Philips healthcare, which own the product line of which he writes about in the case study.

Content The case study presents the experiences with developing a product line of imaging equipment used to support medical diagnosis and interventions. The product line encompasses more than ten product groups as of writing the case study. They describe how they adopted product line engineering, and that there was an initial investment that paid of after a certain point, the break-even point compared to developing single products in isolation.

They say that since they produce products that are used within hospitals, they are subjected to high quality standards.

They say that ”The notion of living components, interfaces and data models has proven to be crucial to build, verify, validate and test software upon a basis of already existing and tested software.” A living component ”means that the component or interface is well tested and ready to be used.” Thus, they test their reusable components.

They give a presentation of how they developed their system that shows that they have done a thorough job at designing the system.

Main claims/results ”The platform components are built and maintained with about 1.6 times the number of people necessary for a single product group to develop the software itself at the same time.” ”Product defect density to 50% of original rate for reused functionality.” ”Two to four times effort reduction.”

Evidence presented They demand payment for their work on the platform component from the product groups. They only have to pay their share of the 1.6 number. Payment is demanded on a yearly basis, and depends on the part of the platform that is used by the product group.

Another piece of evidence is that their systems pass the quality standards for being allowed to be used in hospitals. Thus, they certainly have to be of a rather high quality.

Evaluation of the evidence The evidence presented in this case study is good. They have an actual running product line with ten products which run in hospitals. Hospitals have high quality standards, and thus they have to evaluate the quality of the products to certify it for use in a hospital. They also measure the reduction in effort by actually demanding less payment for their reduced effort. This is good evidence that they indeed had to spend less effort. Their discussion of the technical design of the system is also convincing.

If there is a thing that is not so clear, it is which part of the benefit can be attributed to the testing strategy. They claim that it was proven that it was crucial, but they do not present how they proved it. Thus, we cannot be certain about what part of the 50% that we should attribute to testing.

Another problem is that they do not explain how they measured the reduction in defects. But, a company that develops software for hospitals keeps a close eye on defects reported and have to pass quality certification. Thus, we should give them the benefit of the doubt on this point.

Summary There are several pieces of evidence that favors the claims in this paper. This includes the size of the systems developed, the fact that they have ten different production systems running in hospitals, that their defect density dropped by 50%, and they they demand less payment for their efforts. There are some weaknesses in how they found the 50% number and how they could attribute this to testing. Thus, and by being pessimistic, we can attribute 25% points of the reduction to testing reusable components with high confidence (4/5).

3.5 Siemens Medical Systems Case Study 2007

Source This case study is presented in Reuys et al. 2007 [11].

Authors Authors are Andreas Reuys, Klaus Pohl and Josef Weingärtner. Andreas Reuys, Klaus Pohl have published the papers which presents ScenTED, the test method used in the case study. Weingärtner is the Team Lead Product Care at Siemens Healthcare.

Content The paper presents the case study of the SIENET COSMOS product line at Siemens Healthcare. It is developed with around 100 developers. Relevant for this case study is the development of three applications which tests were used to evaluate the improvement upon introducing the ScenTED product line test methodology into Siemens. The test process is as follows.

1. Creation of activity diagrams representing the control flow of use cases.
2. Manual derivation of domain system test case scenarios.
3. Manual derivation of application system test cases.

This process, called ScenTED, is compared to how it was previously, when one had to develop unique test cases for individual products, even though they had significant similarities.

Main claims/results "[...] 36 scenarios were saved comparing to single system development. [...] an economization of 57% of for the considered part of the system."

Evidence presented "[...] 27 test case scenarios developed during domain engineering were reused in 63 scenarios in application engineering. This implies that 36 scenarios were saved comparing to single system development."

In addition to this, they explain the architecture of their product line and how they organized the development.

They do not say whether the systems are in production or not. They do say that there are high quality standards for systems that are used in hospitals.

Evaluation of the evidence The evidence presented in case study is a little weak. The fact that they could reuse 57% of the test cases does not imply that they used less effort. There is no indication of the time spent. The ScenTED method is presented as intended to be automatic, but in this case study they had to do the generation of test cases manually. This indicates that the time spent is higher than intended if the technique is automatized.

They also say that their technique does not scale up to many applications. This questions to external validity of the conclusion of increased efficiency.

In addition to this, two of the authors are the main creators of the ScenTED method, and they have vested interests in seeing their method succeed. This might warrant not giving them the benefit of the doubt on some of the questions unanswered.

Summary All in all, the evidence presented in this paper does not strongly support their claims. The fact that they did not present the effort measurements together with the statement that they had to work with the method sub-optimally, by doing things manually, taken together with the vested interests some of the authors might have, the confidence in the claims in this paper is low.

3.6 NUWC Case Study 2002

Source This case study is from Cohen et al. 2002, "Successful Product Line Development and Sustainment: A DoD Case Study" [3].

Authors The authors of this case study are Sholom Cohen and Albert Soule of SEI, and Ed Dunn of NUWC.

Content This case study describes experiences from transitioning to and developing using the product line approach for navy range system, systems that test and evaluate systems acquired for the US Army.

The range systems are created from a common collection of components called RangeWare. The systems are not completely built from these, but they do provide a significant part of new systems.

The case study presents some technicalities of how they structure their systems and explains a little how they test the product line. Even though the case study is more than 50 pages long, they do not go into a detailed level of how they structure things.

Main claims/results (Claim nr. 1) "Cost of building software for ranges is at least 50% lower using RangeWare. Development time has also been cut from years to months for several applications. Total personnel for projects may be cut by up to 75%, allowing NUWC to take additional assignments."

(Claim nr. 2) "Fortunately, projects generally benefit far more often from upgrades/fixes to modules done by other projects than they are inconvenienced by the need to upgrade/test to maintain currency with upgrades."

Evidence presented They have actually developed six subsystems over two years: one of 245 kloc, four of about 150 kloc and one of 150 kloc. The claims are taken from these experiences. They say that the product line is in its "teens", i.e. half-way to maturity. The systems developed are called minor subsystems. In the future, when the product line is mature, they would like to develop major systems.

They do not explain how they evaluated claim nr. 2, something that would have been very interested for the questions we want to answer in this report.

Evaluation of the evidence The measured reduction in cost in developing the systems are reliable. The costs do not lie in that the customers got their products at the expected quality but at a cheaper price, which is a very important metric of software development.

It is not possible to attribute a certain percentage of the improvements of testing. The case study does not attribute any part of the improvement to testing. The only improvement they claim are due to reusable component testing, is the claim nr. 2. But they do not present any evidence supporting this claim.

Summary The case study is an interesting one, and they did implement six sizable systems, which would make a very good basis of evaluating many things about software product line engineering. Unfortunately, they did not do a careful comparison between the situation before and after they adopted product line engineering, other than cost. Thus, their improvement claims are reliable on the level that they reduced costs, but it is hard to attribute this reduction to any part of product line engineering. Maybe the testing part reduced the efficiency gained at the design and implementation stage? Thus, we must conclude that the reliability of a claim to improvement to testing is low, even though the reliability of of the improvement with respect to product line engineering is medium or high.

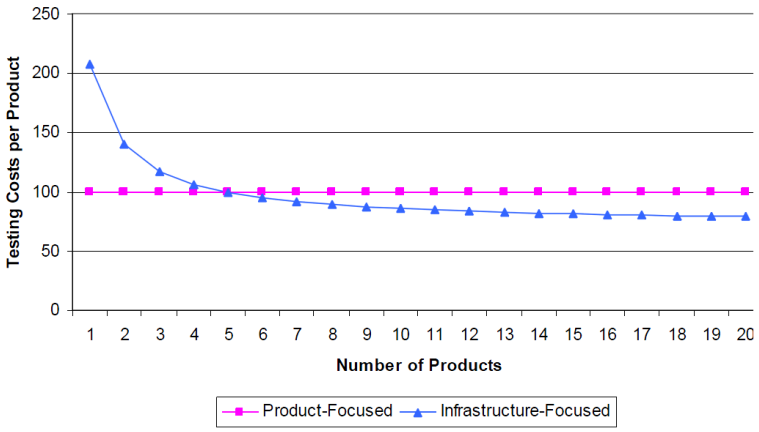

Figure 1: Comparison of testing costs per product [6]

3.7 Testo AG Case Study 2007

Source This case study is published in Ganesan et al. 2007 [6].

Authors The authors are Dharmalingam Ganesan, Ronny Kolb, Uwe Haury and Gerald Meier. The three first authors were employed at Fraunhofer, and the two last at Testo AG. The testing strategy evaluated is not invented at Fraunhofer. The product evaluated is developed at Testo AG, but there is no reason why Testo AG should uphold product line testing, as this is not their business doing. What is their business is the quality of their product, but this is not evaluated in the paper.

Content The paper presents a case study of a product line of portable measurement devices for industry and emission business. They study the products released to market as of writing the paper. The size of these products is not stated, but we can assume it is considerable. Based on these products, they evaluate parameters to an equation comparing the cost of individual product testing, and reusable component testing. These parameters are set based on studying the source code of the studied systems. Based on this, and on probability assignments to the values, they estimate the cost savings of one strategy compared to the other.

Main claims/results ”For the Testo product line of climate and flue gas measurement devices, the results showed that, using an infrastructure-focused test strategy [reusable component testing], on average a cost saving of 13% can be expected with 87% certainty.”

Figure 1 shows the cost of testing each product as it is added to the Testo product line.

Evidence presented Some evidence are the observations of the Testo products released to market as of writing the publication. This consists of estimating eleven parameters to the formula which estimates the cost of testing a product line.

Evaluation of the evidence The paper is well written, well structured and tidy. The estimates presented are probable based on my experience. The treatment of the data is serious and easy to understand. The assumptions and the focus of the analysis are stated sharply.

The weakest part of the study is the estimates of the relative cost of testing reusable components. They state that these are conservative assumptions, and thus cannot be called empirics. They are probably based on the authors previous experience, but since the reasons for choosing these values are not explicitly argued for, or since evidence for them are not presented, this weakens the conclusion of the paper.

Summary This paper is a good study, but the assignment of parameters based on conservative assumptions weakens the conclusion. Thus, a medium reliability is given to the paper's main claims.

3.8 Cabral et al. 2010 experiment

Source This experiment is published, along with a novel technique for SPL testing, in Cabral et al. 2010 [1].

Authors The authors are Isi Cabral, Myra Cohen and Gregg Rothermel of the University of Nebraska-Lincoln. They are all academic employees of this university, and the originators of one of the testing techniques evaluated.

Content The paper presents a novel technique for SPL testing called the FIG Basis Path Method. After the presentation of this technique, they describe a case study. They test two research questions in this paper: How the new method compares to existing methods for SPL testing, and how well the *grouped basis path method* performs for reducing the effort required to use the new testing approach on product lines with alternative features, a special case. They do their experiment on three product lines from academia, the Graph product line and version 5 and 6 of the Mobile Software Product Line, which are of size about 1400 loc and 2200 loc respectively. They seed faults to create faulty mutants of the product line for testing. Students of their research lab created tests for the product line, and these were executed on the correct and faulty versions of the product line to detect defects.

Main claims/results "[...] the FIG Basis Path method [their novel approach] is as effective as testing all products, but tests no more than 24% of the products in the SPL."

Evidence presented They present the number of test cases for each SPL testing method, including product-by-product and their basis path method, and how many defects these test cases found. They also list how many products have to be instantiated to execute the test cases. These numbers are reported for each of the three product lines. They also report that they managed to further reduce the number of product to test and test cases by employing a grouped version of their new approach.

Evaluation of the evidence This paper is well written and deliver what it promises, but the proper end result of an improved testing technique is that the effort spent to achieve the same level of fault detection is reduced. This paper present a reduction in the number of product that one must test, but it does not report on the effort spent. It is quite realistic that the effort spent creating the test suite with fewer test cases is higher in total than creating the test suite with more test cases. The paper does not even mention the comparative size or length of the test cases. Thus, the only difference we have is the difference in quantity.

Another problem is that 24% percent of the products of a product line is ordinarily still extremely high. For example, a product line with 100 features can be combined in say $2^{100/2}$ ways, taking into account constraints by dividing by two. 24% of this number still produces $24\% * 2^{50} \approx 2^{48}$ products.

Summary Given the focus on reduced effort in this report, the results from this experiment does not provide us with good evidence. The reduction is product to test and the reduction of test cases can still probably mean that the effort to create the test cases still are higher than the bigger set. Thus, the reliability in the results as an indication of reduction in testing effort is low.

4 Synthesis of results

Table 1 summarizes the evaluated publications, and the reliability of its result as evidence for choosing that particular testing strategy for product line testing and the reliability of the observed improvement of using it.

The most popular construction techniques for product lines were composition of reusable components. As for the testing techniques, reusable component testing was used by all, as the method of SPL testing, or just one part of it. Some projects used integration testing, one used protocol testing, and some used instantiation of system tests and acceptance tests which included variability.

The best evidence available are from the three publications rated medium or high in reliability. Therefore, they will be used as the basis of doing an evidence based decision. They are the Dialect Solutions case study of 2004 [15], the Philips case study of 2007 [12] and the Testo AG case study of 2007 [6]. Common among these are the technique of reusable component testing. They report an improvement, found to be due to testing using reusable component testing, to be approximately 20%, 25% and 13%. The improvement for Dialect Solutions is estimated to be over 30%, and since they explicitly mention reusable component testing as an important reason for the quality improvement, I have chosen to attribute 20% points of the improvement to that.

Thus we can give the following advice to a company that has a product line: Based on the best evidence available to us, [15, 12, 6], if you construct your products from a repository of reusable components, then testing these components in isolation will probably reduce the effort required on testing by between 13% and 25% compared to testing each product individually to achieve the same defect detection level.

Case study	SPL Technology	Testing Technique	Project complexity	Improvements	%	Reliability
Market Maker [17]	Reusable Components	Component testing	520 kLOC	Reduction of maintenance costs: approx. 60% and Reduced cost of quality (reliability in the field).	60%	low (2/5)
NRO [4]	Reusable Components	Reusable Component, integration and system testing	500 kLOC per product	one tenth the defects	90%	low (2/5)
Dialect Solutions [15]	Reusable components	component testing, product acceptence testing, protocol testing	six products running in production	”unit test costs within core assets can be ”shared” across many products. This has increased overall productivity.” ”PLD has helped [..] improve [..] product quality.	$> 30\%$	medium (3/5)
Philips [12]	Reusable components	component testing	ten product groups	Product defect density to 50% of original	25%	high (4/5)
Siemens [11]	Reusable components	ScenTED	100 developers	Reduction of cost of quality: approx. 57%.	57%	low (2/5)
NUWC [3]	reusable components	reusable component testing	22 components, 6-10 developers	$> 50\%$ reduction in costs	a part of 50%	low (2/5)
Testo AG [6]	reusable components	reusable component testing	10 products	13% cost savings, 87% certainly	13%	medium (3/5)
Cabral et al. experiment [1]	?	FIG Basis Path Method	1.4-2.2 kloc	testing 24% of the products yield the same defect detection rate	66%	low (2/5)

Table 1: Summary of case studies and their reliability verdict

Product	bit	SCS	platform
Eclipse IDE for Java	64-bit	CVS	MacOS
Eclipse IDE for Java EE	64-bit	Subversion	Linux
Eclipse IDE for C/C++	64-bit	Git	Windows
Eclipse for PHP	64-bit	CVS	Linux
Eclipse IDE for JavaScript Web	64-bit	Subversion	MacOS
Eclipse Modeling Tools	32-bit	Git	Windows
Eclipse IDE for Java and Report Developers	32-bit	CVS	Linux
Pulsar for Mobile Developers	32-bit	Subversion	Windows
Eclipse for RCP and RAP	32-bit	Git	Linux
Eclipse SOA Platform for Java and SOA	32-bit	CVS	MacOS

Table 2: Selection of products to test

5 Proposed study design

5.1 Study from scratch

Since I ended up with advising to use reusable component testing as the main product line testing strategy, I will present a study to measure the effect of using that technique compared to a product-by-product strategy.

Within software product line engineering, it is generally agreed, and it also is likely, that one must make three or more products to beat product-by-product development. The more products, the clearer the difference will be probably. One might run into the same problem if one has more than 100 products. The reason is that, although it is common to support the derivation of a large amount of products, it is not so common to actively support and maintain that many products as a catalog. Thus, the study should have about ten products, according to my experience.

The size of the products is not a big problem: both the products and the components it is composed of is to be tested as black-boxes. The components and the products should supply functionality which the participants of the study are knowledgeable of, and be functionality that is non-minor and useful in an industrial setting.

Many developers today use Eclipse, and are thus familiar with its functionality, it is also used in industrial software development. Thus, it is a good candidate as a system of study.

The main opposition to reusable component testing say that one cannot be sure that a component still functions when integrated with other components, and that whatever you test, it must be retested when the component is in a different context.

Thus, the ten eclipse products can be configured as follows. Vary the 32-bit and 64-bit versions, vary the platform, vary the source control system. This gives 180 different systems. Select 10 products such that each eclipse product is represented once, each bit is represented five times, the source control systems and platforms are represented 3, 3 and 4 times. An example of such a selection is shown in table 2

> $\{$Eclipse IDE for Java, Eclipse IDE for Java EE, Eclipse IDE for C/C++, Eclipse for PHP, Eclipse IDE for JavaScript Web, Eclipse Modeling Tools, Eclipse IDE for Java and Report Developers, Pulsar for Mobile Developers, Eclipse for RCP and RAP, Eclipse SOA Platform for Java and SOA$\}$ $\times$ $\{$64-bit, 32-bit$\}$ $\times$ $\{$CVS, Subversion, Git$\}$ $\times$ $\{$MacOS, Windows, Linux$\}$

Strategy	man days	faults found	faults in team-component
Product-by-product testing	12	$F_{p,a}$	$F_{p,t}$
Reusable component testing	3	$F_{t,a} = 0$	$F_{t,t}$

Table 3: Variables resulting from the study

Exercise the system on product level by importing a project from a source control system, do changes to the source and commit to see how the source control system handles the changes to the source tree. The things one is testing here is the source control abstraction layer, the team plugins.

My study would include 13 developers. Each product would get one developer, and the team plugin would get three, each working individually. The ten developers would work one day on developing tests for each product. The three developers would each spend one day testing their component separately.

To account for the difference in skill-level and tooling, the developers testing the products would write down their tests in text as a sequence of commands issued to the Eclipse GUI with the expected behavior. The guys testing the component would get a predefined interface with understandable methods that they would write pseudo code for in a text document. All this is to ensure that the participants do not spend time of tinkering with programming, but instead on testing the system. A researches would then sit down and execute all the tests by hand, locate the errors and attribute it to the correct part of the code.

The results can be interpreted as follows. If $F_{t,t}$ is higher than $F_{p,t}$, taking into account the difference on days of work spent, then testing the reusable components are better. But, if $F_{p,a} - F_{p,t}$ is higher than $F_{t,t}$, taking into account the differences in days of work, then it is inconclusive.

This would give preliminary evidence that one can use to argue for a larger study to estimate the exact benefit of reusable component testing. The estimated effort required for the study presented above is 30 days of work. To estimate the percentage improvement would require the experiment above to be done for several components. We can extend the product tests to two days and assign two persons to test five different components, resulting in a study of two months of work in total, but it should be possible to complete the work in less than a month in duration. Product line engineering is a large scale effort, and experimenting with it requires a lot of effort.

5.2 Study from industry

Another way to estimate the improvement is for companies to spend their development effort in estimating the improvement.

If the teams developing five industrial systems, such as Dialect Solutions, Philips Medical solutions and Testo AG, do a redundant and separate round of testing starting with the same systems, then these five reports, containing experiences from each of the two strategies would result in a large scale experiment that would have great potential for external validity. Such an effort would have to be coordinated from a research institute, and planned out in advance. The duration of such an experiment would be several years.

Hopefully, by gathering the evidence as done in this report, we have somewhat approximated such an effort by considering three of the best case studies which happened to use similar testing strategies.

References

[1] Isis Cabral, Myra B. Cohen, and Gregg Rothermel. Improving the testing and testability of software product lines. In Jan Bosch and Jaejoon Lee, editors, *SPLC*, volume 6287 of *Lecture Notes in Computer Science*, pages 241–255. Springer, 2010.

[2] Paul Clements and Linda Northrop. *Software product lines: practices and patterns*, volume 0201703327. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[3] S. Cohen, E. Dunn, A. Soule, G. Chastek, P. Donohoe, and J. D. McGregor. Successful Product Line Development and Sustainment: A DoD Case Study, Technical Assessments. Technical report, CMU/SEI-2002-TN-018. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 2002.

[4] Sholom Cohen and Patrick Donohoe. Control channel toolkit: A software product line that controls satellites. In *Software product lines: practices and patterns* [2], pages 443–483.

[5] Emelie Engström and Per Runeson. Software product line testing - a systematic mapping study. *Information and Software Technology*, In Press, Accepted Manuscript:–, 2010.

[6] Dharmalingam Ganesan, Jens Knodel, Ronny Kolb, Uwe Haury, and Gerald Meier. Comparing costs and benefits of different test strategies for a software product line: A study from testo ag. In *Proceedings of the 11th International Software Product Line Conference*, pages 74–83, Washington, DC, USA, 2007. IEEE Computer Society.

[7] Lew Boon Kian. Test cost saving and challenges in the implementation of times;6 and times;8 parallel testing on freescale 16-bit hcs12 microcontroller product family. In *Electronic Design, Test and Applications, 2006. DELTA 2006. Third IEEE International Workshop on*, page 7 pp., jan 2006.

[8] Ronny Kolb and Dirk Muthig. Making testing product lines more efficient by improving the testability of product line architectures. In *ROSATEA '06: Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis*, pages 22–27, New York, NY, USA, 2006. ACM.

[9] Frank J. van der Linden, Klaus Schmid, and Eelco Rommes. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[10] Sacha Reis, Andreas Metzger, and Klaus Pohl. Integration testing in software product line engineering: A model-based technique. In Matthew B. Dwyer and Antónia Lopes, editors, *FASE*, volume 4422 of *Lecture Notes in Computer Science*, pages 321–335. Springer, 2007.

[11] Andreas Reuys, Klaus Pohl, and Josef Weingärtner. Siemens medical solutions. In *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering* [9], pages 249–263.

[12] Gerard Schouten. Philips medical systems. In *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering* [9], pages 233–248.

[13] Devesh Sharma, Aybuke Aurum, and Barbara Paech. Business value through product line engineering - a case study. In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications*, pages 167–174, Washington, DC, USA, 2008. IEEE Computer Society.

[14] L. Silva and S. Soares. Analyzing structure-based techniques for test coverage on a j2me software product line. In *Test Workshop, 2009. LATW '09. 10th Latin American*, pages 1 –6, march 2009.

[15] Mark Staples and Derrick Hill. Experiences adopting software product line development without a product line architecture. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, APSEC '04, pages 176–183, Washington, DC, USA, 2004. IEEE Computer Society.

[16] Antti Tevanlinna, Juha Taina, and Raine Kauppinen. Product family testing: a survey. *SIGSOFT Softw. Eng. Notes*, 29(2):12–12, 2004.

[17] Martin Verlage and Thomas Kiesgen. market maker software ag. In *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering* [9], pages 167–189.

A Complete list of considered works

These are all the considered papers which this evidence-based decision is based on.

A.1 Publications considered in detail

1. Cabral 2010 "Improving the testing and testability of software product lines"
2. Cohen 2002 "Successful Product Line Development and Sustainment: A DoD Case Study"
3. Denger 2006 "Testing and inspecting reusable product line components: first empirical results" (symposium)
4. Engstrom 2010 "Software Product Line Testing - A Systematic Mapping Study" (journal)
5. Ganesan 2007 "Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG" (Conference)
6. Kolb 2006 "Techniques and strategies for testing component-based software and product lines" Experience report Open items (Book chapter)
7. Lew 2006 "Test cost saving and challenges in the implementation of x6 and x8 parallel testing on freescale 16-bit HCS12 microcontroller product family" (Workshop)
8. Reis 2007 "Integration testing in software product line engineering: A model-based technique" (conference)
9. Reuys 2007 (book chapter)
10. Schouten 2007 (book chapter)
11. Sharma 2008 "Business Value through Product Line Engineering - A Case Study"
12. Silva 2009 "Analyzing structure-based techniques for test coverage on a J2ME software product line" (Workshop)
13. Staples 2004 "Experiences Adopting Software Product Line Development Without a Product Line Architecture"
14. Tevanlinna 2004 "Product family testing: a survey" (journal)
15. van der Linden 2007 "Software Product Lines in Action" (book)
16. Verlage 2007 (book chapter)

A.2 Publications considered

1. Ali 2010 ”An approach for requirements based software product line testing” (Conference)
2. Ardis 2000 ”Software product lines: A case study”
3. Batory 2002 ”Achieving extensibility through product-lines and domain-specific languages: A case study”
4. Bell 2009 ”A High Volume Software Product Line” (SystemsForge Case Study 2009)
5. Bertolino 2003 ”Use Case-based Testing of Product Lines” (Conference)
6. Bertolino 2006 ”Product Line Use Cases: Scenario-Based Specification and Testing of Requirements” (book chapter)
7. Blundell 2004 ”Parameterized interfaces for open system verification of product lines” (Conference)
8. Breivold 2008 ”Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies”
9. Börger 2008 ”Coupling design and verification in software product lines” (conference)
10. Clements 2001 ”Cummins Inc.: Embracing the future” (book chapter)
11. Clements 2001 ”Software Product Lines : Practices and Patterns” (book)
12. Clements 2005 ”The U.S. Army’s Common Avionics Architecture System (CAAS) product line: A case study”
13. Cohen 2001 ”Control software toolkit: A software product line that controls satellites” (book chapter)
14. Cohen 2006 ”Coverage and adequacy in software product line testing” (workshop)
15. Dager 2000 ”Cummins’s experience in developing a software product line architecture for real-time embedded diesel engine controls”
16. Day 2008 ”A Case Study of Safety integrity level assessment and verification: Electronics division product line evaluation and analysis”
17. Deelstra 2005 ”Product derivation in software product families: a case study”
18. Denger 2006 ”Testing and inspecting reusable product line components: First empirical results” (Symposium)
19. Dinnus 2005 ”Experiences with software product line engineering” (book chapter)
20. DNV Case Study 2007 (in van der Linden 2007) (book chapter)
21. Doerr 2000 ”Freeing Product Line Architectures from Execution Dependencies”
22. Engenio Information Technologies, by William Hetrick, Joseph Moore, and Charles Krueger
23. Engström 2010 ”A qualitative survey of regression testing practices” (book chapter)
24. Engström 2010 ”Regression Test Selection and Product Line System Testing” (Conference)
25. Eriksson 2009 ”Managing requirements specifications for product lines-An approach and industry case study”
26. Gacek 2001 ”Successful Software Product Line Development in a Small Organization. A Case Study” (book chapter)
27. Ganesan 2005 ”Towards testing response time of instances of a web-based product line” Evaluation research Tool (Workshop)
28. Ganesan 2006 ”Discovering Organizational Aspects from the Source Code History Log during the Product Line Planning Phase–A Case Study”
29. Ganesan 2007 ”Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG”
30. Hanssen 2008 ”Process fusion: An industrial case study on agile software product line engineering”

31. Hetrick 2006 ”Incremental Return on Incremental Investment: Engenio’s Transition to Software Product Line Practice” (Engenio Case study 2006)

32. Jaring 2008 ”Modeling Variability and Testability Interaction in Software Product Line Engineering” (conference)

33. Kahsai 2008 ”Specification-Based Testing for Software Product Lines” (Conference)

34. Kakarontzas 2008 ”Product Line Variability with Elastic Components and Test-Driven Development” (conference)

35. Kang 2007 ”Towards a Formal Framework for Product Line Test Development” (Conference)

36. Kangtae 2008 ”A Case Study on SW Product Line Architecture Evaluation: Experience in the Consumer Electronics Domain”

37. Kangtae 2008 ”Building Software Product Line from the Legacy Systems ”Experience in the Digital Audio and Video Domain”” (conference)

38. Kishi 2005 ”Design verification for product line development” (conference)

39. Kishi 2006 ”Formal verification and software product lines” (journal)

40. Kofron 2009 ”Modes in component behavior specification via EBP and their application in product lines”

41. Kolb 2005 ”A Case Study in Refactoring a Legacy Component for Reuse in a Product Line”

42. Kolb 2006 ”Making testing product lines more efficient by improving the testability of product line architectures” (Workshop)

43. Krueger 2002 ”Data from Salion’s Software Product Line Initiative”

44. Krueger 2008 ”HomeAway’s Transition to Software Product Line Practice: Engineering and Business Results in 60 Days” (Conference)

45. Käkölä 2006 ”Software Product Lines: Research Issues in Engineering and Management” (book)

46. Lamancha 2009 ”Model-driven testing in software product lines” (Journal)

47. Li 2008 ”The W-Model for Testing Software Product Lines” (Symposium)

48. Linden 2007 ”Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering” (book)

49. Lisboa 2008 ”A Case Study in Software Product: Lines An Educational Experience”

50. Luo 2009 ”Feature Dependency Modeling for Software Product Line” (conference)

51. Magro 2008 ”A Software Product Line Definition for Validation Environments” (Conference)

52. Mallett 2010 ”Modelling Requirements to Support Testing of Product Lines” (Workshop)

53. Mannion 2004 ”Theorem proving for product line model verification” (conference)

54. McGregor 2004 ”Testing variability in a software product line” Evaluation research Method (Workshop)

55. McGregor 2009 ”Building reusable testing assets for a software product line” (tutorial)

56. Mellado 2008 ”Security Requirements Engineering Process for Software Product Lines: A Case Study” (conference)

57. Mellado 2008 B ”Towards security requirements management for software product lines: A security domain requirements engineering process” (journal)

58. Muccini 2003 ”Towards testing product line architectures” (Journal)

59. Nascimento 2008 ”A Case Study in Software Product Lines - The Case of the Mobile Game Domain” (conference)

60. Nebut 2004 ”A requirement-based approach to test product families”

61. Nebut 2006 ”System Testing of Product Lines: From Requirements to Test Cases” (book chapter)

62. Nokia Mobile Phones Case Study 2007 (in van der Linden 2007) (book chapter)

63. Nokia Networks Case Study 2007 (in van der Linden 2007) (book chapter)
64. Olimpiew 2005 ”Model-based testing for applications derived from software product lines” (workshop)
65. Perrouin 2010 ”Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines” (conference)
66. Philips Medical Systems, by Frank van der Linden
67. Pohl 2005 ”Software Product Line Engineering: Foundations, Principles and Techniques” (book)
68. Pohl 2005 A ”Documenting Variability in Test Artifacts” (book chapter)
69. Pohl 2005 B ”Domain Testing” (book chapter)
70. Pohl 2005 C ”Application Testing” (book chapter)
71. Pohl 2006 ”Software product line testing - Exploring principles and potential solutions” (journal)
72. Reis 2007 ”Integration testing in software product line engineering: A model-based technique”
73. Reuys 2005 ”Model-based system testing of software product families” (Conference)
74. Reuys 2006 ”The ScenTED Method for Testing Software Product Lines” (book chapter)
75. Reuys 2007 (book chapter)
76. Schmid 2005 ”A comprehensive product line scoping approach and its validation” (Conference)
77. Schouten 2007 (book chapter)
78. Sharp 2000 ”Reducing Avionics Software Cost Through Component Based Product Line Development” (Conference) (Boing Case Study 2000)
79. Silva 2009 ”Analyzing structure-based techniques for test coverage on a J2ME software product line” (workshop)
80. Smidth 2002 ”The Economic Impact of Product Line Adoption and Evolution”
81. Svendsen 2010 ”Developing a software product line for train control: A case study of CVL”
82. Toft 2000 ”A Cooperative Model for Cross-Divisional Product Development for a Software Product Line”
83. Uzuncaova 2007 ”A specification-based approach to testing software product lines” (symposium)
84. Uzuncaova 2008 ”Testing software product lines using incremental test generation” (Symposium)
85. Uzuncaova 2010 ”Incremental Test Generation for Software Product Lines” (journal)
86. van Ommering 2007 (book chapter)
87. van Ommering 2008 ”GTV – NXP’s Software Product Line for Mainstream TV” (NXP Case Study 2008)
88. Verlage 2007 (book chapter)
89. Vernazza 2000 ”Moving toward software product lines in a small software firm: a case study”
90. Weiss 1999 ”Software Product-Line Engineering: A Family-Based Software Development Process”

A.3 Potentially relevant publications

These publications was not possible to get hold of during the litterature considerations. The ones prefixes with a star seem especially relevant for the problem in this report.

1. Bosch 2000 ”Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach”
2. Cesare 2005 ”Development of Component-based Information Systems” (Book)
3. Gomaa 2004 ”Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures”
4. Gomaa 2004 Microwave Oven Software Product Line Case Study
5. Gomaa 2004 Electronic Commerce Software Product Line Case Study

6. Gomaa 2004 Factory Automation Software Product Line Case Study
7. Kang 2009 ”Applied Software Product Line Engineering”
8. Kang 2009 ”Formal Verification and Software Product Lines”
9. *Kang 2009 ”Efficient Scoping with CaVE: A Case Study”
10. Kang 2009 ”Model-Driven, Aspect-Oriented Product Line Engineering: An Industrial Case Study”
11. Li 2007 ”Reuse execution traces to reduce testing of product lines” Evaluation research Method (Workshop)
12. Li 2007 ”Automatic integration test generation from unit tests of eXVantage product family” Evaluation research Method (Workshop)
13. Mellado 2008 ”IDEAS09: Applying a Security Domain Requirements Engineering Process for Software Product Lines” (journal)
14. Madhavapeddi 2004 ”Testing Variabilities in Software Product Lines: A Case Study of the Inheritance Mechanism in the Arcade Game Product Line”
15. Neto 2010 ”A Regression Testing Approach for Software Product Lines Architectures: Selecting an efficient and effective set of test cases”
16. Schmölzer 2008 ”Software Product Line Architecture for Enterprise Applications: Principles, Methodologies, and Practices for Model-based SPL Engineering”