Software Product Line Testing

Sajeta, Neha
Research Scholar in
Maharishi Markendeshwar University, Mullana(Ambala)

ABSTRACT

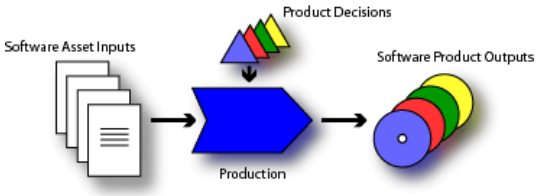
The software product line approach to the development of software intensive systems has been used by organizations to improve quality, increase productivity and reduce cycle time. These gains require different approaches to a number of the practices in the development organization including testing. The objective is to analyze the existing approaches to testing in software product lines. A suitably organized and executed test process can contribute to the success of a product line organization. Testing is used to identify defects during construction and to assure that completed products possess the qualities specified for the products.

Keywords
Software Product line, Testing, SPL Architecture Testing.

1 Introduction:

A software product line (SPL) 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 .
Testing has two main functions: (1) helping to identify faults that lead to failures so they can be repaired and (2) determining whether the software under test can perform as specified by its requirements. In certain domains and styles of development, testing has been performed to estimate the reliability of software. Different types of testing, such as unit, integration, and system testing, are carried out during the development process. Regardless of the type of testing, each task involved is organized around three basic activities:
Analysis: The material to be tested is examined using specific strategies to identify appropriate test cases. Analysis

Test-related activities are organized into a test process that is designed to take advantage of the economies of scope and scale that are present in a product line organization. These activities are sequenced and scheduled so that a test activity occurs immediately following the construction activity whose output the test is intended to validate. Test-related activities that can be used to form the test process for a product line organization are described. Product line organizations face unique challenges in testing.



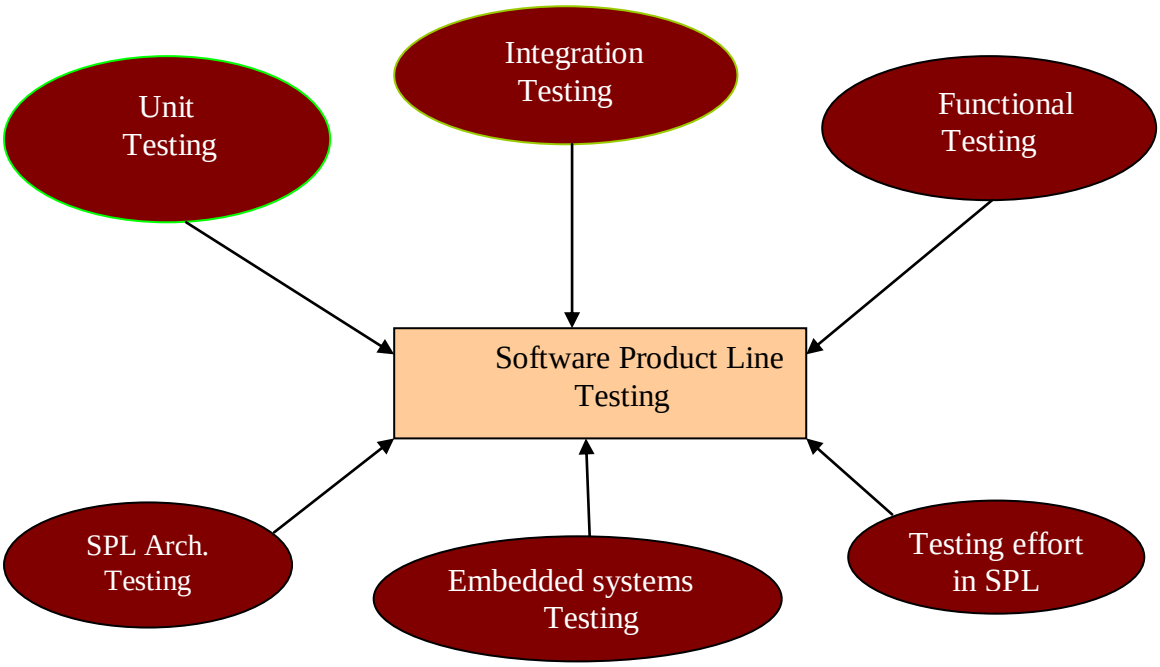
Basic software product line

techniques that involve structured artifacts such as architecture description languages and programming languages can be automated to reduce the test resources needed for a project.
Construction: The artifacts needed to execute the tests specified in the test plan are built. These artifacts usually include test drivers, test data sets, and the software that implements the actual tests.
Execution and Evaluation: The tests are conducted, and the results are analyzed. The software is judged to have passed or failed each test. This information guides decisions about what the next step will be in the development process.

"The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component"[1]

2 SOFTWARE PRODUCT LINE TESTING

2.1 Unit Testing (UT)
2.2 Integration Testing (IT)
2.3 Functional Testing (FT)
2.4 SPL Architecture Testing (AT)
2.5 Embedded Systems Testing (ET)
2.6. Testing Effort in SPL (TE)



2.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application called units , are individually and independently scrutinized for proper operation. The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use. The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit.

2.2 Integration Testing

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. You can do integration testing in a variety of ways but the following are two common strategies:

The top-down approach to integration testing requires the highest-level modules be test and integrated first.
The bottom-up approach requires the lowest-level units be tested and integrated first.

2.3 Functional Testing

Functional testing is a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered .Functional testing differs from system testing in that functional testing "verif[ies] a program by checking it against ... design document(s) or specification(s)", while system testing "validate[s] a program by checking it against the published user or system requirements". Functional testing typically involves five steps
The identification of functions that the software is expected to perform
The creation of input data based on the function's specifications
The determination of output based on the function's specifications

The execution of the test case The comparison of actual and expected outputs.

2.4 SPL Archieture Testing

Product line architecture is a single specification capturing the overall architecture of a series of closely related products. Its structure consists of a set of mandatory elements and a set of variation points. Whereas mandatory elements are part of the architecture of every product in the product line architecture, variation points precisely define the dimensions along which the architectures of individual products differ from each other. New testing techniques are needed to be able to test product line architectures. A first option could be to build new, ad hoc techniques from scratch. In particular, we believe unit testing, integration testing and functional testing are architectural testing techniques form the basis for our approach. Finally, a SA-based regression testing approach is proposed, based on an adaptation of traditional code-based selective regression testing techniques.

2.5 Embedded System Testing:

Embedded systems are in every intelligent device that is infiltrating our daily lives: the cell phone in our pocket and all the wireless infrastructure behind it; the Palm Pilot on our desk. The works found for testing in embedded systems environments or industries Kim et al[2] present a tool that supports the development of SPL for embedded systems of control, using FORM (Feature-Oriented Reuse Method) and using simulation for testing. Kishi et al. [3] apply formal verification techniques (model checking) to verify the design of embedded systems in SPL. They represent the variability in UML models and present a tool that supports this approach. Pesonen et al.[4]use aspects to implement specialisations at the core assets level in embedded systems for smoke testing the devices.

are generally specific to a particular domain and have been tested in artificial

.

2.6 Testing Effort

Effort estimation consists in predict how many hours of work and how many workers are needed to develop a project. In software development Test Ajila [5] make a study of the changes in the product line architecture of a large telecommunications equipment supplier. They conclude that code size is not a good effort refers to the expenses for (still to come) tests. There is a relation with test costs and failure costs (direct, indirect, costs for fault correction predictor of testing effort at either product or product line levels and that testing effort does not seem to depend on the product's target market

.

3 Conclusion

This paper has presented an analysis of the current state of the art in software product lines testing .In general, SPL in Software Engineering is a young discipline, but a very promising one, proving that most of the results and benefits obtained from SPL can be extrapolated to other methodologies or development paradigms. In the case of testing, Bertolino (Bertolino, 2007) has pointed out a transversal challenge to the development of testing techniques and their reuse from emerging paradigms, as product lines may well be.

4 References

[1] Institute of Electrical and Electronics Engineers. 'IEEEStandard Glossary of Software Engineering Terminology' IEEE Std. 610.121990. New York, NY: IEEE, 1990.

[2] Kim, K., Kim, H., Ahn, M., Seo, M., Chang, Y., and Kang, K, 'Asadal: a tool system for co-development of software and test environment based on product line engineering'(2006).

[3] Kishi, T. and Noda, N.'Formal verification and software product lines. Communications of the ACM'(2006).

[4] Pesonen, J., Katara, M., and Mikkonen, T. 'Production-testing of embedded systems with aspects'(2006).

[5] Ajila, S. and Dumitrescu, R. 'Experimental use of code delta, code churn, and rate of change to understand software product line evolution' The Journal of Systems and Software(2007).