Decision support for test management and scope selection in a software product line context

Engström, Emelie; Runeson, Per

Published in:
2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops

DOI:
10.1109/ICSTW.2011.80

2011

Decision Support for Test Management and Scope Selection in a Software Product Line Context

Emelie Engström* and Per Runeson*
*Software Engineering Research Group
Dept. of Computer Science, Lund University, Sweden
(emelie.engstrom, per.runeson)@cs.lth.se

***Abstract*—In large software organizations with a product line development approach, system test planning and scope selection is a complex tasks for which tool support is needed. Due to repeated testing: across different testing levels, over time (test for regression) as well as of different variants, the risk of double testing is large as well as the risk of overlooking important tests, hidden by the huge amount of possible tests. This paper discusses the need and challenges of providing decision support for test planning and test selection in a product line context, and highlights possible paths towards a pragmatic implementation of context-specific decision support of various levels of automation. With existing regression testing approaches it is possible to provide automated decision support in a few specific cases, while test management in general may be supported through visualization of test execution coverage, the testing space and the delta between the sufficiently tested system and the system under test. A better understanding of the real world context and how to map research results to the same is needed.**

I. Introduction

Software product line engineering is a means for organizations to customize large numbers of software products from a common base, instead of developing one-off solutions to each customer or end product. In large software organizations with a product line development approach, a selective testing of product variants is necessary in order to keep pace with the decreased development time for new products.

The number of testing combinations in such variability intensive context are extensive. Testing is repeated across three dimensions, see Figure 1: 1) the traditional testing at different levels (unit, integration, system etc.) 2) regression testing as the system evolves over time, and 3) testing over the variation in space. This entails a high risk of costly redundancy in the testing and also the reverse: that important aspects of the differences between the tested artifacts are overlooked. One of the major challenges in testing a software product line (SPL) is the balancing of testing efforts across these three dimensions [1]. Several questions appear in this context:

- What is the cost vs. the benefit of extensive feature testing? At an early stage, it is not always known how a specific feature is going to be used: in which product variants and customizations.
- How can the testing of different product variants be optimized?
- How much and what do we have to test on the latest version of the software? In the case of an agile development process, the software is updated frequently. It is not feasible to test everything on the latest version.

Trade-offs need to be done both on a management level with respect to the overall distribution of test efforts, and in each situation where a test scope is to be selected. Today there is a lack of support in making those decisions. Decision support may be provided with different levels of automation and should increase the efficiency of the testing as well as the transparency of the decisions. SPL testing literature offers a number of proposals for improvements regarding those issues which could be incorporated in a decision support system [1]. However, only a few of them are are evaluated in real projects and there is little knowledge on their practical implementation. In order to provide decision support, the real context need to be described and the proposed solutions need to be put in relation to both general and specific context descriptions.

This paper discusses the challenges in providing decision support in a variability intensive context from the perspective of its practical implementation. Research on regression testing and the potential use of such techniques for test scope selection in a product line context is discussed as well as the need for less automated support in terms of visualization.

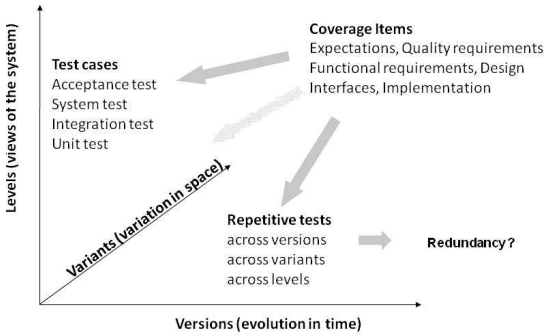

Figure 1. Testing is repeated across three dimensions: testing at different levels, testing of regressions, testing of multiple variants

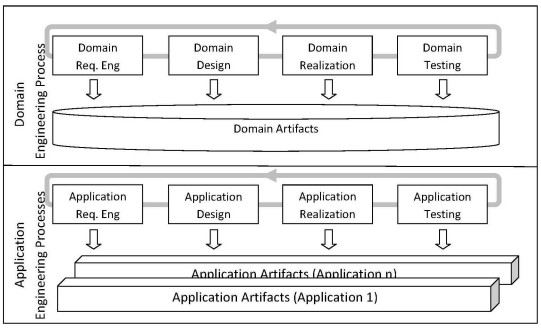

Figure 2. Product line processes modeled by Pohl et al. [2]

II. Decision support in a complex environment

SPL engineering offers a systematic approach for handling variability and for parallel development of several product variants derived from a common base, see Figure 2. Consequently, many proposed SPL testing strategies are based on the existence of models, describing the system and the variability space. According to a recent systematic review [1], the major focus of SPL testing research is on model-based test case design and systematic test case derivation. The review also summarizes the research on test management in software product line engineering, which contains of several proposals, possibly useful in some contexts, and a few evaluations in small and controlled environments.

However in industry there is a need for test planning support even in less idealistic cases of product line like development, where SPL concepts are not strictly followed. Variants may be added in a less controlled manner than suggested by research literature (e.g. by Pohl et al [2]) and clearly specified variability models may not exist. In such contexts, model-based techniques are hard to apply. Furthermore, test management is a part of a large management system, including release planning, requirements management, configuration management, development management, test management, and more, which makes it difficult to enroll major changes from any of those single perspectives. A decision support system must support the complex context, as well as its own incremental evolution within this context.

There is a need for pragmatic techniques guiding not only derivation of test cases, but also planning and selection of test cases, i.e. the test scope. While model-based test case design spans the testing space, the actual selection of the test scope needs to be supported by a tool. This must account for constraints in the environment, for example, cost and lead times for testing, varying importance of product variants, incremental evolution, distribution of the development organization, and even psychological factors. In a large organization, a test manager also needs to deal with lack of trust between different parts of an organization and people's propensity to perform work sloppy, whose results are not directly visible.

A typical scenario is the lack of holistic strategies. Different testing units develop their own testing strategies, for handling the huge amount of possible tests, independently of each other, e.g. low level testers may test only one configuration despite the requirement to deliver support for several variants, while at the same level, another unit distribute a test suite randomly over the possible variants. At product level a common practice is to fully test the most important (or first) variant, leaving the subsequent product variants more or less untested. A decision support system should predict the indirect consequences of the direct choices.

III. Automated decision support

A tool for decision support seeks to automate the management parts of the testing chain, i.e. planning and analysis, in contrast to other test automation activities with a focus on the mechanic parts, i.e. generation and execution of test cases. There may be a difference in goals between the two. While the goal of automation of mechanization often is to replace humans performing repetitive and simple tasks, e.g. executing test scripts, the goal of automating the management in this case is to accomplish tasks beyond our capabilities, e.g. collecting and synthesizing huge amount of information (originating in the huge amount of possible tests and complex relationships between internal factors such as variation points at different levels and external factors such as the development process, the development organization and market demands) and further predicting the effects of different decisions, and select the best one. A tool may give support at many different levels. A high level of automation refers to systems with minimal user interaction, e.g. system collects and analyses data, provides a decision suggestion and in some cases even acts. A lower level of automation refers to support through visualization of relevant information based on synthesis of collected data, leaving the decision and action to the user.

IV. High level decision support

A common proposal for testing in a variability intensive context is to apply regression testing approaches not only to regressions but also to variants [1]. Regression testing (RT) literature offers a number of techniques, all on a rather high automation level; e.g. providing decision proposals regarding which test cases to rerun, based on information about changes and dependencies. RT techniques could roughly be divided into three categories [3]:

1) *Test selection techniques* guide test suite selection based on changes (delta) by focusing testing on changes and possible side effects.
2) *Prioritization techniques* order test cases in a test suite based on some criteria e.g. risk, coverage or test execution history.

3) *Reduction techniques* optimize a test suite with respect to redundancy.

Selection techniques may be useful for the specific selection situation, e.g. analyzing the delta between the software under test and previously tested software, suggesting a test suite covering the differences, and parts that may be affected by the differences. Reduction techniques may support the more general test planning, optimizing the total test pool with respect to coverage of the testing space and minimizing redundancy. Prioritization techniques could be used for both purposes, e.g. prioritize the total test pool with respect to risk and use the prioritization as a basis for test scope selection in the specific case or for the more general test planning. Different criteria for prioritization may be useful in different situations depending on the testing goals.

However, most of these techniques are very context dependent [4], and in many cases only applicable for small parts of the system and at the lowest level of test. While there may exist techniques useful in a very specific selection situation, there is none applicable to a high level, supporting the general test planning and scoping. Many test situations tend to be too complex to benefit from a regression test selection technique alone, even though it still could offer useful guidance in a semi-automated decision support system [5]. It is difficult for a practitioner to identify situations where those techniques are useful and select a suitable technique. In order to enable guidance for practitioners, existing knowledge need to be reported in a more systematic way. Important context variables need to be classified and accordingly mapped to important techniques [4][6]. A decision support system combines knowledge and technologies in a way that serves the specific purpose and has to adapt to and evolve within the complex and changing context where the purpose is to be served.

V. Low level decision support

A more pragmatic way of supporting test scope selection with a lower level of automation (i.e. more human interaction) is through visualization. A tool may collect and synthesize information from different sources of information (e.g. configuration management system or test management system), report it in an effective way and leave decisions and actions to the user. A key to efficient management is to handle and report relevant information with a proper level of details. An example visualization support tool from another domain is the feature visualization tool by Wnuk et al [7] shown in Figure 3. This tool helps management see how decisions on the project scope impact on project as a whole.

For the specific test selection situation, relevant information is, for example, test execution coverage, and the delta between the system under test and the aggregation of previously passed tests. The suitable level of detail depends on the manager's scope of responsibility. Coverage should not be restricted to measure structural coverage but also comprise high level goals, requirements, and quality attributes. The units of measurement is the coverage items (CI:s) which are defined by the test design strategy. For the general test planning task, the execution coverage information should be visualized together with the variability space at each level of testing, and highlight the important variants across testing levels. Examples of questions for which this kind of information would offer support are:

- To what extent has this functionality been tested previously on lower levels, in previous versions, and other variants?
- What is the difference between the software currently under test and the software previously tested?
- How important is this test for the future? How can the testing be optimized with respect to possible customizations?

Redundancy may be defined in different ways (which is illustrated with an example in Figure 4):

1) It may refer to the execution of test cases; a test is redundant if the same test case is executed twice on the same software, even if it could not possibly have another verdict than last time. This is a common definition used in optimizing regression test selection techniques [4]
2) It may also refer to the design of test cases; a test case is redundant if it covers coverage items (including both the covered part of the artifact and the purpose of a test), covered by other test cases, which are often the starting point for regression test reduction,
3) or a combination; a test is redundant if the same CI is executed by tests several times.

The latter definition includes more perspectives than the other two and would be of more value for decision support. Identifying redundancy based only on the execution of the same test cases will not capture the major part of double testing, since it will miss double testing across organizational units (with different testing strategies) or double testing due

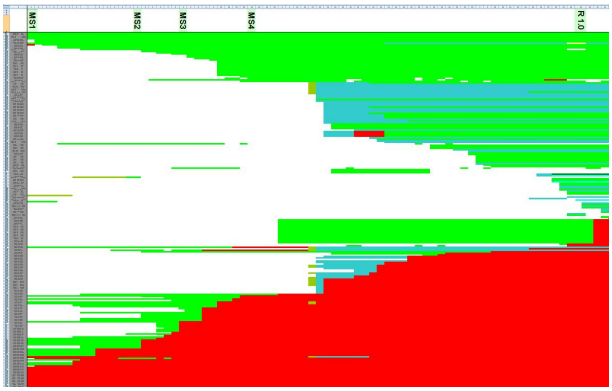

Figure 3. Visualization of which features (y-axis) are introduced and removed over the duration of the project (x-axis).

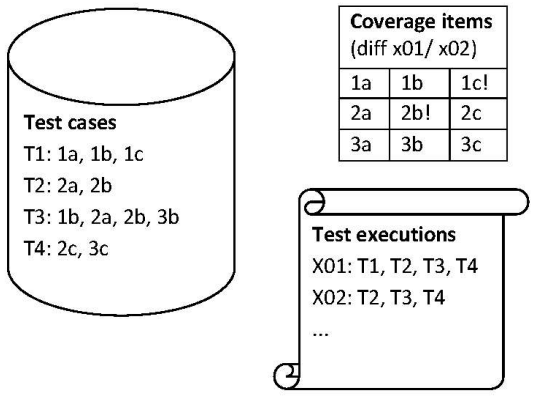

Figure 4. Example case illustrating the different perspectives on redundancy. Test cases are stored in a test pool. Each test case is designed to cover one or more coverage items. Coverage items affected by differences between the artifacts tested in x01 and x02 are marked with an exclamation mark. The execution of T4 in x02 is redundant with respect to the re-execution of non-affected test cases while T2 is redundant in both x01 and x02 with respect to re-execution of coverage items. If the two perspectives are combined T2 is redundant in x01, T4 in x02 and either of T2 or T3 in x02.

to non-systematic reuse of test cases (e.g. copy, paste and adapt). On the other hand if test coverage is based on test case design only, the link between test case execution and the CI:s are missing and it is not possible to identify which tests are redundant or where the testing gaps are but only a measure of the extent of test coverage.

VI. Conclusion

System test planning and scope selection in a variability intensive context is a complex task for which tool support is needed. Tool support should improve the decisions (i.e. increase efficiency of testing) and increase the transparency of the decisions. SPL testing literature proposes a number of strategies for improving test efficiency, but lacks real life evaluations. Research on regression testing, which address similar problems, offers some empirically evaluated approaches. With existing regression testing approaches it is possible to provide automated decision support in a few specific cases. In order to support the application of proposed approaches there is a need for a better understanding of the real world context and a clearer reporting of how and when to apply the approaches. Thus we encourage the community to explore the real world context, model general and typical cases of variability intensive testing, evaluate improvement proposals in industry and report research result in relation to real cases. This would support development and incremental evolution of decision support systems within the organizations.

A first step towards automated decision support is the automated collection and synthesis of data. Through visualization of relevant information at a proper level of detail, test management in general may be supported. Information to be visualized is:

- the aggregated test execution coverage
- the testing space defined in coverage items (CI:s) comprising three dimensions of test variation:
  - Varying test levels
  - Evolution in time
  - Variation in space
- the delta between the sufficiently tested system, derived from the execution coverage of the testing space, and the system (or subsystem) under test.

The core elements for analyzing test coverage and redundancy are the CI:s, the test cases and the links between the two. The definition and granularity of CI:s depend on the context and may include structural units as well as quality goals and functional requirements. By separating the definition of coverage items from the decision algorithm in a test selection strategy the generalization of decision algorithms between contexts is enabled. In order to enable analysis and reuse of test results across levels, versions and variants, the relation between different CI:s within the organization need to be identified.

References

[1] E. Engström and P. Runeson, "Software product line testing - a systematic mapping study," *Information and Software Technology*, vol. 53, no. 1, pp. 2 – 13, 2011.

[2] K. Pohl, G. Böckle, and F. J. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[3] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, pp. n/a–n/a, 2010. [Online]. Available: http://dx.doi.org/10.1002/stvr.430

[4] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

[5] E. Engström, P. Runeson, and A. Ljung, "Improving regression testing transparency and efficiency with history based prioritization - an industrial case study," in *Proceedings of the 4th International Conference on Software Testing Verification and Validation*. IEEE Computer Society, 2011.

[6] P. Runeson, M. Skoglund, and E. Engstrom, "Test benchmarks – what is the question?" in *Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International Conference on*, 2008, pp. 368 –371.

[7] K. Wnuk, B. Regnell, and L. Karlsson, "What happened to our features? visualization and understanding of scope change dynamics in a large-scale industrial setting," in *Proceedings 17th IEEE International Requirements Engineering Conference*. IEEE Computer Society, 2009.