Handling Combinatorial Explosion in Software Testing

by

Mats Grindal

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2007
To the memory of Thomas Vesterlund, who encouraged and motivated me to pursue this project but never got to see it finished.
Abstract

In this thesis, the overall conclusion is that combination strategies, (i.e., test case selection methods that manage the combinatorial explosion of possible things to test), can improve the software testing in most organizations. The research underlying this thesis emphasizes relevance by working in close relationship with industry.

Input parameter models of test objects play a crucial role for combination strategies. These models consist of parameters with corresponding parameter values and represent the input space and possibly other properties, such as state, of the test object. Test case selection is then defined as the selection of combinations of parameter values from these models.

This research describes a complete test process, adapted to combination strategies. Guidelines and step-by-step descriptions of the activities in process are included in the presentation. In particular, selection of suitable combination strategies, input parameter modeling and handling of conflicts in the input parameter models are addressed. It is also shown that several of the steps in the test process can be automated.

The test process is validated through a set of experiments and case studies involving industrial testers as well as actual test problems as they occur in industry. In conjunction with the validation of the test process, aspects of applicability of the combination strategy test process (e.g., usability, scalability and performance) are studied. Identification and discussion of barriers for the introduction of the combination strategy test process in industrial projects are also included.

This research also presents a comprehensive survey of existing combination strategies, complete with classifications and descriptions of their different properties. Further, this thesis contains a survey of the testing maturity of twelve software-producing organizations. The data indicate low test maturity in most of the investigated organizations. Test managers are often aware of this but have trouble improving. Combination strategies are suitable improvement enablers, due to their low introduction costs.

Keywords: Combination Strategies, Software Testing, State-of-Practice, Equivalence Partitioning, Test Process.
Acknowledgements

A project of this magnitude is not possible to complete without the support and encouragement from many different people. I am forever grateful for all the help I have received during this time.

A number of people have been of crucial importance for the completion of this project and as such they all deserve my special gratitude.

First and foremost, I would like to thank my supervisor Prof. Sten F. Andler, and my co-supervisors Prof. Jeff Offutt, Dr. Jonas Mellin, and Prof. Mariam Kamkar for their excellent coaching, feedback and support.

Towards the end of this thesis project, Prof. Yu Lei, Prof. Per Runeson, Prof Kristian Sandahl, Prof. Benkt Wangler, and Ruth Morrison Svensson all played significant roles in providing important feedback.

Then I am also very much indebted to my co-authors Prof. Jeff Offutt, Dr. Jonas Mellin, Prof. Sten F. Andler, Birgitta Lindström, and Åsa G. Dahlstedt. Several anonymous reviewers of our papers have also contributed significantly. Thank you very much for all the valuable feedback.

Past and present colleagues at University of Skövde, and in particular the members of the DRTS group: Alexander, AnnMarie, Bengt, Birgitta, Gunnar, Joakim, Jonas, Jörgen, Marcus, Robert, Ronnie, Sanny, and Sten, have been great sources of inspiration and have provided constructive advice. Thanks also Anna, Camilla, and Maria at University of Skövde and Lillemor and Britt-Inger at Linköping University for helping me out with all the administrative tasks.

Past and present colleagues at Enea: Anders, Bogdan, Gunnel, Johan M., Johan P., Joakim, Krister, Maria, Niclas, Per, Sthure, Thomas T. and Thomas V. have contributed with enthusiasm and lots of patience. Special thanks to Marie for helping out with the cover.

Last but not least, I couldn’t have done this without the support of my family and my close friends: Lars Johan, Birgitta, Asa and Claes, Joakim, Arne and Timo. I love you all.

It is impossible to list everyone that deserves my appreciation so to all those persons who have contributed knowingly or unknowingly to the completion of this project I bow my head.

The financial sponsors of this project are Enea, the Knowledge Foundation, the University of Skövde, and the Information Fusion Programme at the University of Skövde, directly sponsored by Atlas Copco Tools, Enea, Ericsson, the Knowledge Foundation and the University of Skövde.
Contents

I Introduction 1

1 Background 3

2 Theoretical Framework 7
  2.1 What is testing 7
  2.2 Challenges of Testing Software Systems 9
  2.3 State-of-Practice in Testing Software Systems 10

3 Combination Strategies 13

4 Problem - Can Combination Strategies Help? 19
  4.1 Test Process 21
  4.2 Combination Strategy Selection 21
  4.3 Input Parameter Modeling 22
  4.4 Handling Conflicts in the IPM 22
  4.5 Validation 23

5 Research Methodology 25
  5.1 Introduction to Research Methodology 25
  5.2 Research Methodologies Applied 26

6 Method - A Combination Strategy Test Process 33
  6.1 A Combination Strategy Test Process 33
  6.2 Combination Strategy Selection 36
  6.3 A Method for Input Parameter Modeling 38
  6.4 Handling Conflicts in the IPM 41
  6.5 Validation of the Combination Strategy Test Process 42
  6.6 Overview of Articles 43
7 Results

7.1 Combination Strategy Selection ........................................... 47
7.2 Input Parameter Modeling .................................................. 51
7.3 Conflict Handling in the IPM .............................................. 52
7.4 Applying the Combination Strategy Test Process ................. 55
  7.4.1 Handling Large and Complex Test Problems .................. 55
  7.4.2 Alternative Test Case Generation Methods .................. 57
  7.4.3 Validation of Thesis Goals ......................................... 58
7.5 Related Work ................................................................. 59
7.6 Discussion ........................................................................ 62
  7.6.1 Formulation and Evaluation of Research Goals .............. 63
  7.6.2 Software Process Improvements ................................. 63
  7.6.3 Properties of Methods ............................................... 66

8 Conclusions ........................................................................ 69

8.1 Summary ........................................................................... 69
8.2 Contributions ................................................................. 71
8.3 Future Work ................................................................. 72

References ............................................................................. 77
Part I

Introduction
Testing consumes significant amounts of resources in development projects [Mye79, Bei90, GOM06]. Hence, it is of general interest to assess the effectiveness and efficiency of current test methods and compare these with new or refinements of existing test methods to find possible ways of improving the testing activity [Mye78, BS87, Rei97, WRBM97, SCSK02].

Research on testing has been conducted for at least thirty years [Het76]. Despite numerous advances in the state-of-art, it is still possible to find examples where only limited testing knowledge is used in practice [NMR+04, GSM04, GOM06]. This research is an attempt to bridge the gap between the state-of-art and the state-of-practice.

Combinatorial explosion is a frequently occurring problem in testing. One instance of combinatorial explosion in testing is when systems under test have several parameters, each with so many possible values that testing every possible combination of parameter values is infeasible. Another instance of combinatorial explosion in testing may occur for configurable systems. When systems under test have many configuration parameters, each with several possible values, testing each configuration is infeasible. Examples of configuration parameters are versions of a specific software or hardware module, different types of software or hardware modules, and number of logical or physical entities included in a computer system.

Combination strategies is a family of methods, which targets combinatorial explosion. The fundamental property of combination strategies is their
ability to select a subset of all possible combinations such that a coverage criterion is satisfied. In their general form combination strategies can be applied to any testing problem, which can be described in terms of parameters with values. For instance, consider testing a date field (YYMMDD). This testing problem can be described by the three parameters “YY”, “MM”, and “DD”, where each of the three parameters has a set of associated values. Any combination of values of the three parameters represent a possible test case.

The seminal work on combination strategies applied to testing was performed in the mid-eighties [Man85]. The early research investigated how combination strategies could be used to identify system configurations that should be used during testing [Man85, WP96]. In more recent research, combination strategies are also used for test case selection [CDKP94, LT98, CGMC03].

Despite twenty years of research on combination strategies, little is known about the feasibility of combination strategies in industrial settings. Hence, the goal of this research is to investigate if combination strategies are feasible alternatives to test case selection methods used in practice.

Within the context of using combination strategies in industrial settings several issues are to a great extent unexplored. These issues are (i) how to integrate combination strategies within existing test processes, (ii) given a specific test situation, how to select an appropriate combination strategy, (iii) how to represent the system under test as a set of parameters with corresponding values, and (iv) how to manage conflicts among the values in the input space of the system under test.

Combination strategies need to be integrated into existing test processes (issue (i)) since these are used to as a general means to provide structure to planning, monitoring and controlling the performed activities within testing. When new techniques, such as combination strategies, are considered, it is vital that these fit within established ways of working.

The need to select an appropriate combination strategy for a specific test problem (issue (ii)) arises from the fact that there exist more than 15 combination strategies with different properties [GOA05, GLOA06].

Combination strategies require the system under test to be represented as an input parameter model (IPM). As an example, consider testing function int index(element, vector) which returns the index of the element in the vector. An IPM of the index function consists of two parameters, one representing different elements and the other representing different vectors.
An alternative IPM could use one parameter to represent possible results, for instance, element found first in vector, element found last in vector, element not found at all, etc. A second parameter could be used to represent the size of the vector, for instance, zero elements, one element, and several elements. This example illustrates that the tester has several alternatives when deciding on a suitable IPM. Therefore, it is vital to provide support to the tester during input parameter modeling (issue (iii)).

In some cases, the IPM contains values of two or more parameters that cannot be combined. For instance, in the second alternative of the index example, it is impossible to return an element in the first position in an empty list. Hence, there is a need to handle such conflicts, which illustrates issue (iv).

Within this thesis all of these issues (i)-(iv) are addressed. A test process custom-designed for combination strategies is formulated and partly explored. Further, methods are developed for selection of a suitable combination strategy, formulation of IPMs, and handling of conflicts in the input space of the test object. To validate the process, with its methods, experiments in real industrial settings have been conducted.

The combination strategy selection method is based on an assessment of the project priorities and the importance of the test object. The project priorities allow the tester to determine which properties of combination strategies are important. The importance of the system under test gives advice to the tester about suitable levels for the important properties. Analysis and experimentation, within the scope of this research, have resulted in descriptions, and in some cases quantification, of properties of several existing combination strategies [GOA05, GLOA06].

An eight-step input parameter modeling method is formulated within this research project. The method allows requirements on the test object to be expressed in any way, ranging from informal to formal. These requirements are translated step-by-step into a semi-formal model, which satisfies the requirements imposed by combination strategies. The input parameter modeling method has been evaluated in an experiment involving a number of professional testers. Results from this experiment indicate that the input parameter modeling method can be successfully employed after less than an hour of teaching the method. The results also confirm the observation by Grochtmann and Grimm [GG93] that input parameter modeling is a creative process that can never be fully automated.
Cohen, Dalal, Fredman, and Patton [CDFP97] show examples of conflicts in the input space of the test object, that is, when the combination of two or more IPM parameter values is infeasible. New methods for conflict handling are proposed within this research. These and existing methods have been evaluated with respect to the size of the final conflict-free test suite in an experiment. The main conclusion from this is that conflicts handled in the test generation step result in smaller test suites than if the conflicts are handled in the input parameter modeling step [GOM07].

This thesis contains two parts. The first part is organized as follows. Chapter 2 provides a general background on testing, describing the challenges of testing a software based system. Chapter 3 gives a thorough description of combination strategies. Chapter 4 describes and motivates the research problem. The problem section also explains how the problem is divided into a set of goals for the research. The goals are described in separate sections. In chapter 5, the research methods used within this research project are discussed. For each goal section in chapter 4 there is a corresponding methods section in chapter 6. Chapter 7 contains the results achieved and chapter 8 concludes the first part of this thesis with a summary, some conclusions, and directions for future work.

The second part of this thesis contains six papers [GOM06, GOA05, GLOA06, GOM07, GO07, GDOM06]. Section 6.6 describes the framework for this research and how these papers relate to each other and to this framework.
Chapter 2

Theoretical Framework

This chapter contains all necessary background on testing for this thesis. There are three parts to this background. The first part (section 2.1) provides a general overview on testing and defines a number of central concepts. The second part (section 2.2) describes the main challenges of testing when applied in industrial projects. The third part (section 2.3) contains an overview of the state-of-practice in a number of software producing organizations.

2.1 What is testing

Testing is the activity in which test cases are identified, prepared and executed. A test case contains, at least, some input and expected results. The software unit (e.g., module, program or system) under test is called the test object. During the execution of a test case, the test object is stimulated by the input of the test case and reacts to this by producing actual results. The expected results of the test case is compared with the actual results produced by the test object and a test result, that is, either pass or fail, is produced.

A failure [AAC+94] is a deviation of the delivered service from fulfilling the system function. A failure is normally detected by a difference between expected and actual result. An error is the part of the system state that is liable to lead to a subsequent failure [AAC+94]. Finally, a fault in the general sense is the adjudged or hypothesized cause of an error [AAC+94].
Identification of test cases is the task of deciding what to test. Methods that aid the tester in test case identification are called *test case selection methods*. A set of test cases is a *test suite*. Test suites may be joined to form larger test suites.

Figure 2.1 shows a generic test process [BS 98b]. Step 1 of any test process is to plan the forthcoming activities. The planning includes, at least, identifying the tasks to be performed, estimating the amount of resources needed to perform the tasks, and making financial and time budgets. Step 2 is to make any preparations needed for the upcoming test execution. Important tasks during the preparation step are to select and document the test cases. In step 3, the test cases are executed and test results are collected. These results are then analyzed in step 4 in order to determine whether or not more testing is needed. If more testing is needed feedback loops allow for returning to any of the previous steps depending on the amount of work needed. Also, feedback loops from step 3 allow for correction and re-execution of failed test cases.

During testing it is important to use test cases with both valid and invalid values. *Valid* values are values within the normal operating ranges of the test object and correspondingly *invalid* values are values outside the normal operating ranges. Testing using invalid values in the test cases is called *negative testing*. Negative testing is used to test error or exception handling mechanisms.
2.2 Challenges of Testing Software Systems

The software testing organization as well as the individual testers face a number of challenges in producing good-quality testing. Some of these are quality of requirements, test planning, efficiency of testing, and test case selection.

The quality of the requirements has a large impact on both the quality of the testing and the quality of the final product. Incorrect requirements may both lead to faults in the implementation of the product and to incorrect test cases. Studies show that incorrect requirements may be the source of as much as 51% of the faults [VL00]. It can safely be stated that the infrastructure, that is, tools, processes, and organization, for handling requirements will impact the testing. Further it is important for testers to be able to influence the quality of the requirements [Dah05].

Related to incorrect requirements are requirement changes that are allowed to occur late in the software development project. Changes may occur, either triggered by new needs or by the realization that the current set of requirements contains faults. The later a change is allowed to occur the higher demands on the development and test processes to be able to handle the change. For the tester, an ideal situation, for handling changes, would be if all test cases affected by the changes could be both identified and changed automatically.

Another area of challenge for a testing organization is planning, in particular, estimation of resource consumption. It is known in advance that failures will be detected but it is very difficult to predict the amount of failures. The correction times, the severity, and when and where the failures and their corresponding faults will be found are other aspects of faults that are difficult to predict. Test planning is a large and important area for research.

Increasing the efficiency of the testing is a major challenge for many test organizations. The automation of suitable testing tasks has interested software organizations for several years [KP99].

Test case selection is in itself a difficult problem. Many test objects have large input spaces. Testing all possible inputs is generally infeasible. Instead the input space needs to be sampled. Often, there are many aspects of a test object that need to be considered in the sampling process.

The focus of this research is on handling the large input space but also touches on handling late requirement changes and automation.
2.3 State-of-Practice in Testing Software Systems

Reports from the 1970s and 1980s show that testing in industry consumes a large amount of resources, sometimes more than 50% [Boe, Bro75, Deu87, You75]. Corresponding figures for the years 2004-2005 are reported in a study of twelve software producing organizations. The reported test time consumption ranges from 10% up to 65% with a mean of 35% (paper I [GOM06]). It is safe to say that testing has been and still is a big cost for software developing organizations.

The study also revealed that despite the incentives to cut costs in testing and the large body of knowledge of testing generated during the last 30 years, the test maturity of software producing organizations can be surprisingly low. Even organizations developing safety-critical applications exhibit a great variance in applying structured test case selection methods and in their collection and use of metrics.

Lack of usage of structured test case selection methods is seen in organizations developing any type of software products, from web-based systems to safety-critical applications. A similar observation is made by Ng et al. [NMR+04].

Further, the results of the study indicate that the testers’ knowledge is at least fair. The testers are also allowed to start working early in the projects so the reasons for the relative immaturity have to be sought elsewhere. The test managers expressed concern over the situation, which shows an awareness of the problem.

The study identifies three major obstacles for improving test maturity. First, there may be a lack of understanding among upper management of the contributions of testing. From their perspective, as long as their products generate profit, they are good enough and improvement is not prioritized. Second, the relative immaturity of testing does not only manifest itself in lack of structured usage of test case selection methods. There is also a lack of structured usage of metrics. With few or even no collected metrics it is very difficult to assess the current situation and to estimate the potential gains of an improvement. Without the ability to express the potential benefits in economical terms it is hard to get an approval for a change. Third, a change in the way of working or even the introduction of a tool almost always requires an initial investment which should payoff later. Most development projects are conducted under high time-pressure. It is therefore difficult to find a project in which one is willing to take the initial investment costs to make future projects more effective and efficient.
The most important conclusion from this study is that organizations need to focus more on structured metrics programs to be able to establish the necessary foundation for change, not only in testing. A second conclusion is that improvements should be made in many small steps rather than one large step, which is also appealing from a risk perspective.
Theoretical Framework
Chapter 3

Combination Strategies

This chapter introduces the necessary background and concepts relating to combination strategies.

Most test objects have too many possible test cases to be exhaustively tested. For instance, consider the input space of the test problem. It can be described by the parameters of the system under test. Usually, the number of parameters and the possible values of each parameter result in too many combinations for testing to be feasible. Consider for instance testing the addition functionality of a pocket calculator. Restricting the input space to only positive integers still yields a large number of possible test cases, \((1+1, 1+2, 1+3, \ldots, 1+N, 2+1, 2+2, \ldots, N+N)\), where \(N\) is the largest integer that the calculator can represent. This is one example of combinatorial explosion in testing.

Combinatorial explosion can be handled by combination strategies. *Combination strategies* is a class of test case selection methods that use combinatorial strategies to select test suites that have tractable size. Combination strategies are used in experimentation in other disciplines, for instance physics and medicine, to identify combinations of values of the controlled variables. Mandl [Man85] is the seminal work on application of combination strategies in testing. Since then, a large number of combination strategies have been proposed for testing. Many of these are surveyed in paper II [GOA05].
Figure 3.1 shows an overview of a classification scheme for combination strategies. *Non-deterministic* combination strategies rely to some degree on randomness. Hence, these combination strategies may produce different solutions to the same problem at different times. The *deterministic* combination strategies always produce the same solution for a given problem. The *instant* combination strategies produce all combinations in an atomic step while the *iterative* combination strategies build the solution one combination at a time.

A common property of combination strategies is that they require an input parameter model. The *input parameter model* (IPM) is a representation of properties of the test object, for instance input space, state, and functionality.

The IPM contains a set of parameters, *IPM parameters*. The IPM parameters are given unique names, for instance A, B, C, etc. Each IPM parameter has a set of associated values, *IPM parameter values*. These values are also given unique names, for example, 1, 2, 3, 4, and so on. Figure 3.2 shows a small example of an IPM with 4 IPM parameters, containing 3, 4, 2, and 4 number of values respectively.

<table>
<thead>
<tr>
<th>Parameter A</th>
<th>1, 2, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter B</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>Parameter C</td>
<td>1, 2</td>
</tr>
<tr>
<td>Parameter D</td>
<td>1, 2, 3, 4</td>
</tr>
</tbody>
</table>

Figure 3.2: Simple input parameter model.

The IPM can be used to represent the input space of the test object. A simple approach is to assign one IPM parameter to each parameter of the
test object. The domains of each test object parameter are then partitioned into groups of values using for instance equivalence partitioning [Mye79] or boundary value analysis [Mye79]. Each partition of a test object parameter is then represented by a separate value of the corresponding IPM parameter. Almost all software components have some input parameters, which could be partitioned into suitable partitions and used directly. This is also the approach taken in many of the papers on combination strategies [KKS98].

However, Yin, Lebne-Dengel, and Malaiya [YLDM97] point out that in choosing a set of IPM parameters the problem space should be divided into sub-domains that conceptually can be seen as consisting of orthogonal dimensions. These dimensions do not necessarily map one-to-one onto the actual input parameters of the system under test. Along the same lines of thinking, Cohen, Dalal, Parelus, and Patton [CDPP96] state that in choosing the IPM parameters, one should model the system’s functionality, not its interface. Dunietz, Ehrlich, Szablak, Mallows, and Iannino [DES+97] show that the same test problem may result in several different IPMs depending on how the input space is represented and partitioned.

The following example illustrates that the same test problem may result in more than one IPM. Consider testing the function \texttt{step(start, end, step)}, which prints a sequence of integers starting with start, in steps of step up to end.

\begin{verbatim}
A (start): 1: negative, 2: zero, 3: positive, 4: non-integer
B (end): 1: negative, 2: zero, 3: positive, 4: non-integer
C (step): 1: negative, 2: zero, 3: one, 4: > 1
\end{verbatim}

Figure 3.3: IPM, alternative (i), for the \texttt{step} function.

There are two alternative examples of IPMs for the \texttt{step} function: alternative (i) depicted in Figure 3.3 and alternative (ii) depicted in Figure 3.4. In alternative (i), the parameters of the function are mapped one-to-one onto IPM parameters. This results in 64 (4 × 4 × 4) possible test cases. In alternative (ii), properties of the printed sequence are represented as parameters in the IPM. This IPM results in 18 (3 × 3 × 2) possible test cases.

Based on the IPM, combination strategies generate abstract test cases. Abstract test cases are combinations of IPM parameter values consisting of one value from each IPM parameter. Abstract test cases may be translated, in a post-processing step, into actual inputs of the test object.
An abstract test suite is a set of abstract test cases. Abstract test suites can be described as sets of $N$-tuples, where $N$ is the number of IPM parameters in the IPM.

Coverage is a key factor when deciding which combination strategy to use. Different combination strategies support different levels of coverage with respect to the IPM. The level of coverage affects the size of the test suite and the ability to detect certain types of faults.

1-wise (also known as each-used) coverage is the simplest coverage criterion. 1-wise coverage requires that every value of every IPM parameter is included in at least one test case in the test suite. Table 3.1 shows an abstract test suite with three test cases, which satisfy 1-wise coverage with respect to IPM (ii) depicted in figure 3.4.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>TC3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.1: 1-wise coverage of IPM alternative (ii).

2-wise (also known as pair-wise) coverage requires that every possible pair of values of any two IPM parameters is included in some test case. Note that the same test case can often cover more than one unique pair of values. Table 3.2 shows an abstract test suite with nine test cases, which satisfy 2-wise coverage with respect to IPM (ii) depicted in Figure 3.4.

A natural extension of 2-wise coverage is $t$-wise coverage, which requires every possible combination of values of $t$ IPM parameters to be included in some test case in the test suite.

The most thorough coverage criterion, $N$-wise coverage, requires a test suite to contain every possible combination of the IPM parameter values in the IPM. The resulting test suite is often too large to be practical. $N$-wise coverage of IPM (ii), depicted in Figure 3.4, requires all possible combinations of the values of the three IPM parameters ($3 \times 3 \times 2 = 18$ combinations).
### Table 3.2: 2-wise coverage of IPM alternative \( (ii) \).

<table>
<thead>
<tr>
<th>Test Case</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>TC3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>TC4</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TC5</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>TC6</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>TC7</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>TC8</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>TC9</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Base choice coverage [AO94] is an alternative coverage criterion partly based on semantic information. One base value is selected from each parameter. The base value may, for instance, be selected based on the most frequently used value of each parameter. The combination of base values is called the base test case. Base choice coverage requires every value of each IPM parameter to be included in a test case in which the rest of the values are base values. Further, the test suite must also contain the base test case.

Recent years show a growing interest from academia and industry alike in applying combination strategies to testing. From an academic perspective the interest has manifested itself in an increased production of research focusing on combination strategies. One sign of the increased interest from industry is the growing number of combination strategy tools. The website [1](http://www.pairwise.org) contains a collection of about 20 commercial and free combination strategy tools.

Although test cases selected by combination strategies often focus on the input space of the test object, any property of test objects that can be expressed in IPMs can be tested. Combination strategies have this wide applicability in common with other test case selection methods such as Equivalence Partitioning [Mye79] and Boundary Value Analysis [Mye79]. In contrast, state testing [Bei90], focuses primarily on testing the functionality of the test object. Another, related, difference is that test cases resulting from state testing often contain sequences of relatively simple inputs whereas combination strategies usually select test cases with a single but more complex input.
A common property of all the above mentioned test case selection methods, including combination strategies is that they are based on models of the test object. In all these cases these models are generated from information of the specifications and rely, at least to some extent, on the ingenuity of the tester. This means that different testers may come up with different models and hence different test cases. As soon as specifications are available, modeling with its subsequent test case selection can be initiated.

Two other classes of model-based test case selection methods are control-flow based [Bei90] and data-flow based [Bei90] test case selection. In both these cases, models (graphs) are derived from the source code. This has two important implications for testing. First, modeling (and hence testing) cannot be started before the source code is finished. Second, the models can be generated automatically.

From a performance perspective it is still unclear how combination strategies relate to other test case selection methods, such as the above mentioned. There is still room for much more research on this topic.
Chapter 4

Problem - Can Combination Strategies Help?

The overall aim of this research is to investigate *if combination strategies are feasible alternatives to test case selection methods used in practical testing*.

The motivation behind this aim is that combination strategies are appealing since they offer potential solutions to several challenges in testing.

First and foremost, combination strategies can handle combinatorial explosion, which was illustrated in section 3. As described in section 2.2, many test problems have very large input spaces. It is often the case that large input spaces are a result of combinatorial explosion. Hence, combination strategies target the challenge of large input spaces.

Other possible advantages with combination strategies relate to the problems identified by the test managers in the state-of-practice investigation (see section 2.3). The application of combination strategies should be time-efficient and hence not jeopardize test projects under time pressure. Further, combination strategies naturally provide a way of assessing the test quality through the use of coverage with respect to the IPM as discussed in section 3.

A potential gain with combination strategies is the possibility of postponing some of the costly test activities until it is certain that these activities should be executed. The same IPM can be input to several combination strategies supporting different coverage levels. If the combination strategies are automated the cost of generating several abstract test suites is small. Each test suite can be evaluated, for instance with respect to size, and the most suitable abstract test suite is then selected. Identification of expected
results for a test case is often costly since, in the general case, it has to be
done manually. With combination strategies it is possible to delay this step
until it is decided which test cases should be used.

Finally, in some circumstances, combination strategies can provide a ba-
sis for automation, in part, of the test generation step. In cases where the
IPM parameters map one-to-one onto the parameters of the test object it
is possible to automatically transform abstract test cases generated by the
combination strategies to real test case inputs. A consequence of this is that
changes in the requirements of the test object can be handled efficiently by
changing the IPM and generating a new test suite.

On a number of occasions, combination strategies have been applied to
test problems in industry settings. In most cases combination strategies
have been used to select test cases for functional testing [DJK+99, BY98,
Hul00, DHS02]. Although the examples of using combination strategies for
functional testing dominate, there are several examples of the applicability
of combination strategies in other areas of testing. Combination strategies
can also be used to select system configurations that should be used dur-
ding testing [WP96, YCP04]. Robustness testing is another area in which
combination strategies have been used [KKS98].

Despite these experience reports, there is little documented knowledge
about the actual effects of applying combination strategies to test problems.
Further, these reports do not contain much substantial information about
requirements and guidelines for the use of combination strategies in industry
settings. Hence, it is not possible to judge from previous knowledge if combi-
nation strategies are feasible alternatives to the test case selection methods
used in practical testing, which is the aim of this research.

To reach this aim five goals have been identified. The following list
summarizes these goals and sections 4.1 - 4.5 discuss them in more detail.

\textit{G1} A test process tailored for combination strategies should be defined.
\textit{G2} Means for comparing and selecting suitable combination strategies for
test problems should be explicitly described.
\textit{G3} A method for input parameter modeling should be formulated.
\textit{G4} A method for handling conflicts in the input space should be identified.
\textit{G5} The combination strategy test process should be validated and com-
pared to currently used methods in real industrial settings.
4.1 Test Process

The first goal, \(G1\), states that a test process tailored for combination strategies should be defined.

Most organizations describe the set of activities that are followed in order to build software in development processes. The purposes with a process approach are to ensure that all activities are performed in the right order and to allow for more predictable quality in the performed activities. The use of a process also makes the work less dependent on key persons. Testing is part of the development process so the testing activities can be part of a development process. In practice, it is often more convenient to define an own test process, which should be possible to plug into an arbitrary development process.

The generic test process, depicted in Figure 2.1, contains the necessary activities in a testing project. In sequential development processes, based on for instance the Waterfall- or the V-models, the test process can be applied directly. In iterative development processes, a slightly adjusted version of the test process is used. First, test planning is conducted for the entire project, then the four steps of the test process are repeated for each iteration. Finally, an extra test execution activity together with a test stop decision is used on the final product.

A test process tailored for combination strategies should include these activities and support both sequential and iterative project models. Hence, a preferable approach is to refine the generic test process. Further, it should define the necessary activities to enable use of combination strategies, for instance combination strategy selection.

4.2 Combination Strategy Selection

The second goal, \(G2\) expresses the need to explicitly describe means for comparing and selecting a suitable combination strategy for a specific test problem.

Our survey of combination strategies presented in paper II [GOA05] identifies more than 15 combination strategies. Moreover, the results from the study presented in paper III [GLOA06] indicate that the combined usage of several combination strategies may be beneficial, further increasing the choices available to the tester. The sheer number of combination strategies
Problem - Can Combination Strategies Help?

...to choose from makes it difficult to select a suitable combination strategy for a given test problem.

Which combination strategy to apply may depend on several properties, for instance, associated coverage metric, size of generated test suite, and types of faults that the combination strategy targets. Hence, it is necessary to provide the tester with means to compare combination strategies. These means include descriptions of the properties of existing combination strategies as well as descriptions of methods for exploring these properties of future combination strategies.

4.3 Input Parameter Modeling

The applied combination strategy is a factor that obviously has a large impact on the contents of the final test suite. The IPM with its contents is another factor that influences the contents of the final test suite greatly. Hence, the third goal, \((G3)\), which calls for the definition of an input parameter modeling method.

Recall from section 3 that the IPM is a representation of the input space of the test object and that the same test problem may result in several different IPMs. At first glance, creating an IPM for a test problem seems an easy task. However, with different modeling alternatives available to the tester the task becomes less obvious.

It is an open question how the effectiveness and efficiency of the testing is affected by the choice of IPM parameters and their values. Hence, it is important to investigate different alternatives to input parameter modeling.

At some point, the contents of the IPM depend on the experience and creativity of the tester. To decrease the alternatives and guide the tester towards an IPM of acceptable quality there is a need for an input parameter modeling method or a set of guidelines for this task.

4.4 Handling Conflicts in the IPM

The fourth goal, \((G4)\), calls for the identification of a method for handling conflicts in the input space of the test object.

Cohen, Dalal, Fredman, and Patton [CDFP97] show examples in which a specific value of one of the IPM parameters is in conflict with one or more values of another IPM parameter. In other words, some [sub-]combination of
IPM parameter values is not feasible. A test case that contains an infeasible sub-combination cannot be executed. Hence, there must be a mechanism to handle infeasible sub-combinations. Infeasible sub-combinations should not be confused with negative testing (see section 2.1).

An important principle of conflict handling is that the coverage criterion must be preserved. That is, if a test suite satisfies 1-wise coverage with conflicts, it must still satisfy 1-wise coverage after the conflicts are removed. An important goal is to minimize the growth of the number of test cases in the test suite.

4.5 Validation

The fifth, goal $(G5)$ requires the combination strategy process to be validated in a real setting.

This is a key goal of this research, that is, to determine if combination strategies are feasible alternatives to the test case selection methods used in practical testing. Only the use of the combination strategy process in a real setting can demonstrate if the theory works in practice. Further, it is of great interest to assess the performance of combination strategies under realistic conditions.
Problem - Can Combination Strategies Help?
Within the scope of this research six empirical studies have been conducted. Empirical studies can be conducted in different ways, for instance depending on the goal of the study. This chapter presents a brief introduction to research methodology, in section 5.1. Given this introduction, section 5.2 describes the six studies from a research methodology point of view and highlights important methodological issues in each study.

5.1 Introduction to Research Methodology

The researcher can choose between three types of research strategies when designing an empirical study according to Robson [Rob93]:

- A survey is a collection of information in standardized form from groups of people.
- A case study is the development of detailed, intensive knowledge about a single case or of a small number of related cases.
- An experiment measures the effect of manipulating one variable on another variable.

An important difference between a case study and an experiment is the level of control, which is lower in a case study than in an experiment [WRH+00].
Which research strategy to choose for a given study depends on several factors, such as the purpose of the evaluation, the available resources, the desired level of control, and the ease of replication [WRH+00].

Another aspect of empirical studies is whether to employ a quantitative or a qualitative approach [WRH+00]. Studies with qualitative approaches are aimed at discovering and describing causes of phenomena based on descriptions given by the subjects of the study [WRH+00]. Qualitative studies are often used when the information cannot be quantified in a sufficiently meaningful way. In contrast, studies with quantitative approaches seek to quantify relationships or behaviors of the studied objects, often in the form of controlled experiments [WRH+00].

Both surveys and case studies can be conducted using either qualitative or quantitative approaches. Experiments are, in general, conducted with quantitative approaches [WRH+00].

The three main means for data collection are through observation, by interviews and questionnaires, and by unobtrusive measures [Rob93]. Two classes of measurements exist. Objective measures contain no judgement [WRH+00]. In contrast, subjective measures require the person measuring to contribute with some kind of judgement [WRH+00].

The validity of the results of a study is an important aspect of the research methodology. Cook and Campbell [CC79] identify four different types of validity. Conclusion validity concerns on what grounds conclusions are made, for instance the knowledge of the respondents and the statistical methods used. Construct validity concerns whether or not what is believed to be measured is actually what is being measured. Internal validity concerns matters that may affect the causality of an independent variable, without the knowledge of the researcher. External validity concerns the generalization of the findings to other contexts and environments. The representativity of the studied sample, with respect to the goal population, has a large impact on the external validity since it determines how well the results can be generalized [Rob93].

5.2 Research Methodologies Applied

This research project contains five studies S1-S5. Study S5 has three major parts S5a, S5b and S5c. These five studies have resulted in six papers (I-VI), which can be found in the second part of this thesis. Papers I-IV document
5.2 Research Methodologies Applied

studies S1-S4, one-to-one. Paper V documents parts of study S5c and paper VI documents studies S5a, S5b and the rest of study S5c.

Table 5.1 categorizes the five studies with respect to the employed research strategy and whether the chosen approach is quantitative or qualitative. The different parts of study S5 have different research methodologies and are thus indicated separately in the table.

<table>
<thead>
<tr>
<th>Study</th>
<th>Research Strategy</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>survey</td>
<td>qualitative</td>
</tr>
<tr>
<td>S2</td>
<td>survey</td>
<td>qualitative</td>
</tr>
<tr>
<td>S3</td>
<td>experiment</td>
<td>quantitative</td>
</tr>
<tr>
<td>S4</td>
<td>experiment</td>
<td>quantitative</td>
</tr>
<tr>
<td>S5a</td>
<td>case study</td>
<td>qualitative</td>
</tr>
<tr>
<td>S5b</td>
<td>case study</td>
<td>qualitative</td>
</tr>
<tr>
<td>S5c</td>
<td>experiment</td>
<td>quantitative</td>
</tr>
</tbody>
</table>

Table 5.1: Overview of research methodologies used in the studies.

The following paragraphs discuss key methodological issues with respect to each of the conducted studies. Further details can be found in the corresponding papers.

Study S1

Prior, informal observations of aspects of testing maturity aspects in several organizations indicate generally low testing maturity. If formal studies would confirm these informal observations, that would motivate further research on how this situation could be improved. With an explicit focus on combination strategies in this research project, a natural consequence was to focus on test case selection methods in the state-of-practice investigation.

Hence, the aim of study S1 was to describe the state-of-practice with respect to testing maturity, focusing on the use of test case selection methods. Low usage of test case selection methods, would increase the relevance of further studies of combination strategies.

Testing maturity can be defined in many ways and may consist of several components, some of which are quantifiable and others not. The first key methodological issue for study S1 is whether to use a qualitative or a quantitative approach.
The governing factor when selecting a qualitative approach was that the most important questions in the study are more qualitative than quantitative.

Although study S1 is classified as a survey, it also has some properties in common with case studies, for instance the difficulty of generalizing the results. The second key methodological issue for study S1 was the selection of subjects, that is organizations to study. The total population of organizations producing and testing software is not known, which means that judging the representativity of a sample is impossible. The effect of this is that regardless of which and how many subjects are studied, results are not generalizable [Yin94]. Based on the decision to use a qualitative approach, that is to identify and explore potential cause-effect relationships, it was deemed worthwhile to have a heterogenous set of study objects. This is the motivation behind the decision to deliberately select the subjects instead of sampling them.

A third key methodological issue for study S1 was how to collect the data. Using a self-completed questionnaire would in many respects have been adequate but to avoid problems with interpretation and terminology fully structured interviews were used.

Qualitative studies use different techniques, such as analysis, triangulation, and explanation-building to analyze the data [Rob93]. In several of these techniques sorting, re-sorting, and playing with the data are important tools for qualitative analysis [Rob93]. Further, qualitative data may be converted into numbers and statistical analysis can be applied to these data as long as this is done overtly [Rob93].

In study S1, much of the initial aim was reached by just examining the raw data, that is most organizations did not use test case selection methods in a structured way. Since much of the aim with the study was already reached, only limited analysis of the data was conducted. From a qualitative analysis perspective, this study may be considered to be prematurely terminated. However, reaching the original study aims, that is identification of several organizations where test case selection methods are not used in a structured way, motivated the decision to terminate the study.

Study S2
The objective of study S2 is to devise a comprehensive compilation of the previous work on combination strategies. The main challenges from a methodological perspective are how to find all relevant sources and when to stop.
The approach adopted for these questions was to query the major article databases and follow all relevant references recursively until all relevant articles have been found. Peer review was used to validate the completeness of the survey.

**Study S3**

The aim of study S3 was to investigate and compare the performance of a number of test case selection methods. An experimental approach suited this purpose best. The Goal Question Metric (GQM) approach [BR88] was used to identify suitable metrics and to describe the experiment in a structured manner.

A general methodological challenge of experiments with test case selection methods is that there needs to be test objects with known faults. It is impossible to judge the representativity of a set of test objects and equally impossible to judge the representativity of a set of faults. This makes generalization difficult, which is a threat to external validity. The approach taken in this experiment is to look not only at the number of faults found but also at the types of faults found. Through subsumption [RW85] it was then possible to make some general claims from the results.

**Study S4**

Just like study S3, study S4 is conducted as a controlled experiment. Again the aim is to investigate and compare a number of alternative methods for handling conflicts in IPMs. The design of the experiment allowed for the investigated methods to be compared with a reference method. This allowed for a hypothesis testing approach [WRH+00] in the analysis of the results.

The controlled variables of this experiment are different properties of IPMs. As described in chapter 3, IPMs contain representations of test problems and are used as input to combination strategies. The conflict handling methods do not require the IPMs to include semantic information, that is the meanings of the IPMs. This allowed IPMs with desired properties to be created artificially, that is without basing them on actual test problems. From a methodological perspective, this experiment can be seen as a simulation experiment [Rob93] since the test problems are simulated. However, it should be noted that although the IPMs in this experiment are artificial, it is perfectly feasible to identify actual test problems which will result in IPMs that are exactly the same as those in the experiment. Further, the studied
conflict handling methods cannot discriminate between an artificially created IPM and one that is based on an existing test problem. The reason is that the conflict resolution algorithms do not utilize semantic information.

Also in this study representativity was an issue. As has already been stated, it is possible to identify test problems that will yield exactly those IPMs used in the experiment. However, it is impossible to know the general distributions of properties of test problems, which means that the experimental results alone are not sufficient for general claims. To strengthen the results, the experiment was complemented with studies of the algorithms of the investigated methods. Specifically, the relation between the size of the IPM and the size of the generated test suite was studied. These studies confirm the experimental results and point in the direction of the results being general.

Study S5
The aim of study S5 was to validate the combination strategy test process in a real setting. Validation of a suggested solution in a real setting presents several challenges. First, there may be specific requirements on the object of study limiting the number of potential candidates. In the context of this research, an ideal study object requires a well specified test problem, existing test cases derived by some structured test case selection method, documented found faults, and time available for generation and execution of an alternative test suite by combination strategies. Second, if a suitable study object is found, it may involve a great economic risk for a company to allow the researcher to experiment with a previously untried solution or concept within its own production. Third, if the researcher is allowed into the company, there are often restrictions on how the study can be performed and on how much information can be published. These problems manifested themselves in this research project. The ideal study object was not available, with the result that the validation study was divided into three different studies, S5a, S5b and S5c.

Studies S5a and S5b were both conducted as case studies and study S5c was conducted as an experiment. Both case studies (S5a and S5b) were single case designs [Yin94]. This approach is motivated, in part, by the decision to perform a feasibility study. It is also motivated, in part, by the difficulty of finding study objects for untried solutions. From an industry perspective, a single case feasibility study has the properties of a pilot study, which, if successful, is a strong case for further larger studies.
As described in section 5.1, the results of a case study cannot be generalized based on statistical analysis. Instead, analytical generalization has to be used [Yin94]. An important part of analytical generalization is replication of the study under similar conditions. Hence, a detailed description of the context and conditions of the study is required. Both studies S5a and S5b were performed in commercial companies under confidentiality agreements. As far as possible, the descriptions of these studies reveal the details but to some extent the completeness of the descriptions is limited. The two most severe limitations are details of the actual faults found in study S5a and details of the already existing test cases in study S5b.

To enhance the reliability of the two case studies, detailed protocols were used during the studies [Yin94]. Due to the confidentiality agreement, these protocols cannot be released. However, a general description of the major steps is included in the appendix of the technical report [GDOM06]. Further, parts of study S5a are currently being repeated by the company that owns the test problem as part of their normal testing.

The aim of study S5c was a proof-of-concept with respect to IPM. The object of investigation is a set of guidelines for transforming a test problem into an IPM. Previous to this study, these guidelines had not been validated under realistic conditions, which is why this study focuses mainly on the feasibility of the guidelines. Within this context, the guidelines are considered feasible if they can be used to produce IPMs of sufficient quality within reasonable time frames.

This study was conducted as an experiment in which subjects (testers) were asked to follow the guidelines and the resulting IPMs were evaluated with respect to a number of desired properties.

A research methodological issue in study S5c was how to choose response variables. Input parameter modeling may contain an element of creativity from the tester. Further, some desired properties of an IPM, such as completeness, can to some extent be subjective. To decrease the potential risk of bias, response variables were selected to support objective measures. Where subjective measures could not be avoided they were formulated as Boolean conditions to magnify the differences in the values as much as possible.

A small pilot study was executed as part of study S5c. This pilot study provided valuable feedback for the execution of the main experiment.

Due to the decision to perform a feasibility study, further studies will be necessary to investigate the actual performance of the input parameter modeling guidelines.
Chapter 6

Method - A Combination Strategy Test Process

A major part of this work is focused on refining the generic test process, depicted in Figure 2.1, to support the practical use of combination strategies. This includes the development of techniques, methods, and advice for the combination specific parts of the test process and a validation of the complete process.

This chapter provides a top-down description of the test process tailored for combination strategies in section 6.1. It is followed by detailed descriptions of some of the specific activities in sections 6.2 - 6.4.

In addition to the descriptions of these activities, this research also includes a series of experiments and case studies to investigate and evaluate these methods and techniques. In particular, a proof-of-concept experiment has been conducted. A description of this experiment is presented in section 6.5.

Further details of these methods, techniques and advice can be found in the six papers that form the basis of this thesis. Section 6.6 provides an overview of how the papers are related to each other and to the process of using combination strategies in testing.

6.1 A Combination Strategy Test Process

Figure 6.1 shows a test process specifically designed for the use of combination strategies [Gri04]. This test process is an adaptation of the generic test process described in section 2.1.
The planning step of the generic test process is omitted in the combination strategy test process. The reason is that the planning step is general in the sense that the planning decisions made govern the rest of the testing activities, for instance which test case selection methods to use. One implication of this is that instructions on how to use a specific test case selection method do not need to take planning into account. Actually, it is beneficial to keep planning and test case selection independent. Planning may be performed in a large variety of ways and the desired test case selection method should not impose any unnecessary restrictions on the planning.

Apart from the absence of a planning activity, the main difference between the combination strategy test process and the generic test process is in the test preparation activity of the generic test process. In the combination strategy test process, this activity has been refined to satisfy the requirements from combination strategies. Steps (1)-(5) in the combination strategy test process are all specific to combination strategies.

Step (1) is to select a combination strategy to use. This step is covered in more detail in section 6.2. Step (2) is to construct an IPM. This step is presented in more detail in section 6.3. There is a bidirectional dependence between these steps, that is, the results of one step may affect the other step. For instance, if the combination strategy base choice is selected, one value for each IPM parameter in the IPM should be marked as the base choice. In a similar fashion, if the result of input parameter modeling is two or more IPMs, it may be favorable to use different combination strategies for the different IPMs. Hence, the combination strategy test process should support multiple iterations between the two steps choice of combination strategies.
Step (3) is the generation of abstract test cases. In this step, the selected combination strategies are applied to the created IPM. The result of this step is an abstract test suite. Most combination strategies can be expressed as algorithms. Hence, this step is possible to automate, which makes this step inexpensive to perform.

In practice, the selection of test cases is often influenced by the time available for test case execution. In step (4) the abstract test suite is evaluated. The evaluation may, for instance, focus on the size of the test suite and indirectly consider the testing time.

If the abstract test suite is too large the tester may return to steps one and two to try to reduce the size of the test suite. The advantage with this approach is that the costly parts of test case development, that is, identification of expected results and documentation of test cases are postponed until it is certain that the test cases will actually be used.

In step (5), “test case generation”, the abstract test cases are transformed into executable test cases. This step consists of at least three tasks. The first task is the identification of actual test case inputs to the test object. The abstract test cases are converted into real test case inputs through some mapping function that is established during the input parameter modeling. The second task is to identify the expected result for the specific input and the third task is to document the test case in a suitable way. If the intention is to automate test execution this part involves writing test programs. For manual test execution, test case instructions should be documented.

All three of the test generation tasks are difficult to automate. Identification of actual test case inputs can be automated but it requires that the function mapping IPM parameter values to actual inputs be formalized. Automation of the identification of the expected results may possibly be the most difficult task to automate. The reason is that the specification needs to be semantically exact and machine readable, that is, expressed in some formal language. Finally, automatic documentation of test cases requires code generation, which also requires semantically exact specifications. Unless these requirements are satisfied, the test generation step is likely to be relatively expensive due to much manual intervention.

Step (6) of the combination strategy test process is test case execution. As the name implies, the test cases are executed and the results recorded. There are no differences in this step compared to the corresponding step
in the generic test process. The final step (7) is the test stop decision. Again, it is a copy of the corresponding test stop decision step in the generic test process. Steps (6) and (7) are included in the combination strategy test process to indicate the opportunity for combination strategy re-selection and input parameter re-modeling should the test results be unsatisfactory.

### 6.2 Combination Strategy Selection

Recall from section 4.2 that combination strategy selection for a specific testing problem is not trivial. There are many combination strategies with different properties. Combination strategies may also be combined, increasing the options for the tester [GLOA06]. The used combination strategies have significant impacts on both the effectiveness and the efficiency of the whole test activity.

1) Determine the project priority (time, quality, cost)
2) Determine the test object priority
3) Select suitable combination strategies

**Figure 6.2: Three-step combination strategy selection method.**

This research advocates a three-step method, shown in Figure 6.2, for the selection of combination strategies. In the first step, the overall project priorities are determined. A well-known and often used model of the project priorities involves the three properties: time, quality, and cost [Arc92]. The fundamental idea is that a project cannot focus on all three of these properties at the same time. Instead one or two of the properties should take precedence over the rest. As a basis for selection of combination strategies for a test object, this research suggests a total ordering among these three properties. This ordering forms the basis for which properties of combination strategies that should be considered during the selection process. Further, a description of the relations between properties of combination strategies and the three project properties is used. Table 6.1 exemplifies such relations. These relations were identified through reasoning. The column static analysis, denotes the properties that can be assessed statically. Strong relations indicate which properties that should be considered for each of the three project properties.

The objective of the second step is to determine the importance of the specific test object. Any type of importance metric may be used. This
Combination Strategy Selection

<table>
<thead>
<tr>
<th>Combination Strategy Properties</th>
<th>Static Analysis Possible</th>
<th>Project Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Supported coverage criteria</td>
<td>X</td>
<td>Strong</td>
</tr>
<tr>
<td>2) Size of generated test suite</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>3) Algorithmic time complexity</td>
<td>X</td>
<td>Strong</td>
</tr>
<tr>
<td>4) Types of targeted faults</td>
<td>X</td>
<td>Strong</td>
</tr>
<tr>
<td>5) Number of found faults</td>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>6) Tool support</td>
<td>X</td>
<td>Strong</td>
</tr>
<tr>
<td>7) Conflict handling support</td>
<td>X</td>
<td>Strong</td>
</tr>
<tr>
<td>8) Predefined test case support</td>
<td>X</td>
<td>Strong</td>
</tr>
</tbody>
</table>

Table 6.1: Combination strategy properties and project priorities.

information is used to determine the level of the considered properties from the first step.

In the third and final step, one or more combination strategies are selected based on the project priority and the importance of the test object. In particular, when the main priority of the project is quality it may be desirable to use more than one combination strategy with properties that complement each other.

Two of the eight combination strategy properties, “size of generated test suite” (2) and “number of faults” (5) differ between test problems and can thus only be evaluated dynamically. The remaining six properties can be assessed statically.

As shown in section 3, combination strategies can be classified into the two broad categories deterministic and non-deterministic. In the former case, it is possible to calculate exactly the size of the test suite based on the number of IPM parameters and IPM parameter values. In the latter case, randomness plays a role in the combination strategy algorithms. Hence, the exact size of the test suite cannot be stated in advance for all combination strategies. In these cases, the approximate size of the test suite may be estimated or the actual test case generation may be performed.

The number of faults found, depends not only on the used combination strategy. The number and types of existing faults in the test object also has a large impact on the number of faults found. This makes it very difficult to predict, how many faults a combination strategy will actually find.
To assess the performance of combination strategies with respect to some of these properties an experiment was staged. In the experiment, five test problems were used. IPMs were prepared from the specifications of each of these test objects. Different combination strategies were then applied to the IPMs to generate test suites. Implementations of the five test problems, seeded with faults, were then tested using the generated test cases and the results were recorded, analyzed and presented. The properties used in the comparison are number of test cases in the generated test suite, the number and types of the faults found and the supported coverage criteria. The details of the experiment can be found in paper III [GLOA06].

The data presented in this thesis represent a snap-shot of knowledge about combination strategies. It is likely that new combination strategies will emerge and that further properties will be deemed interesting. Hence, there is a need to extend combination strategy property data. Several of the properties, such as, “support of specific coverage criteria” (1), “types of targeted faults” (4), “tool support” (6), “conflict handling support” (7), and “support for predefined test cases” (8), should be possible to assess by reviewing the available documentation and the combination strategy algorithm. The “algorithmic time complexity” (3) requires a careful analysis of the algorithm.

Assessment of the “test suite size” (2) and “number of found faults” (5) requires the investigated combination strategy to be executed in a defined context. The combination strategy test process in Figure 6.1 supports efficient evaluation of the test suite size through automated generation of abstract test cases and the feedback loop to the combination strategy selection. The number of faults found is difficult to assess since it depends on the number, types and distribution of faults in the test object, which is impossible to know without actually running the test cases.

### 6.3 A Method for Input Parameter Modeling

Section 4.3 identifies input parameter modeling as a challenge for the tester. A representation of the test object in an IPM is a pre-requisite for the use of combination strategies.

Figure 6.3 shows an overview of the eight steps of the proposed input parameter modeling method [GO07].
In step (1), the tester decides whether to use a structural or functional approach to input parameter modeling. In the structural approach, each physical parameter in the interface is modeled as a separate IPM parameter. The strength of the structural approach is that it is usually easy to identify the IPM parameters and to translate the abstract test cases into real test case inputs. The weakness of the structural approach is that the model may not reflect the functionality of the test object. To illustrate this, consider testing a function: \texttt{boolean find\_element(list, element)}, which returns true if the integer \texttt{element} is found in the \texttt{list} of integers. Figure 6.4 shows an IPM generated with a structural approach. In this example there is no guarantee that the IPM will result in test cases that exercise the two situations when the element is found and when it is not found.

![Figure 6.4: Structural based IPM for the find\_element function.](image)

The basis for the functional approach is the observation by Cohen et al. [CDPP96], that in choosing the IPM parameters, one should base the model on the system’s functionality not on its interface. This means that the IPM parameters do not necessarily map one-to-one onto the physical parameter in the interface. The state of the test object may also be represented by IPM parameters. Higher test quality is the strength of the
functional approach. A downside is the difficulty to identify the IPM parameters. Figure 6.5 shows an IPM for the find_element function, generated with a functional approach.

| A (length of list): | 1: empty, 2: one element, 3: several elements |
| B (sorting):       | 1: unsorted, 2: sorted, 3: all elements equal |
| C (match):         | 1: yes, 2: no |

Figure 6.5: Functional based IPM for the find_element function.

Step (2) and (3) of the IPM parameters and IPM parameter values are identified. An IPM parameter represents some characteristic of the test object. Some examples are the physical parameters in the interface, properties of the physical parameters, parts of the functionality, and parts of the state of the test object. IPM parameters may be identified from any source of information about the test object, for instance, the interface descriptions or the functional specifications. The experience of the tester may also play a role during the IPM parameter identification. The value domains of each IPM parameter are then partitioned into non-empty groups of values. The underlying idea of this step is the same as in equivalence partitioning [Mye79], that is, all elements in a partition are expected to result in the same behavior of the test object. The set of partitions for an IPM parameter should be complete and disjoint.

Step (4) allows the tester to statically evaluate the IPM, for instance with respect to completeness. When the tester is satisfied with the IPM in step four, additional information is added to the IPM in steps (5), (6), and (7). Step (5) allows the tester to document constraints in how values of different IPM parameters may be combined. This topic is further explained in section 6.4. In step (6), a translation table is established. The translation table contains mappings between IPM parameter values and values of the actual test object interface. This information is used when automatic generation of actual test cases is desired. Step (7) allows the tester to include already existing test cases into the test suite.

Finally, in step (8) the tester decides whether to use one or several IPMs to represent the test object. If a test object consists of several independent parts or aspects it may make sense to create several smaller models rather than one large model. More than one model also allows for varying coverage of the different models.
6.4 Handling Conflicts in the IPM

As was mentioned in section 4.4, Cohen et al. [CDFP97] observed that sometimes one value of an IPM parameter cannot be combined with one or more values of another IPM parameter. Ignoring such conflicts may lead to the selection of test cases that are impossible to execute. Hence, a general mechanism for handling conflicts is needed. The purpose of conflict handling is to give the tester the power to exclude impossible sub-combinations, which is not the same as eliminating negative test cases, as was described in section 4.4.

Two methods to handle conflicts have previously been proposed. In the sub-models method the IPM is rewritten into several conflict-free sub-IPMs [CDFP97, WP96, DHS02]. The avoid method is to avoid selecting infeasible combinations, used for example in the AETG system [CDFP97].

In paper IV [GOM07] two new methods are suggested, the abstract parameter method and the replace method.

The abstract parameter method allows the IPM parameters involved in a conflict to be replaced by an abstract IPM parameter. The abstract IPM parameter contains valid combinations of the replaced IPM parameters, such that the desired coverage is maintained. The replace method makes it possible to handle conflicts after the abstract test cases have been selected. The idea is to replace an abstract test case containing a conflict with two or more conflict-free test cases such that the coverage is maintained.

The previously proposed sub-models method, as well as the new abstract parameter and replace methods, may all result in unnecessary large test suites due to over-representation of some IPM parameter values. This can be compensated for by post analysis and reduction of the abstract test suite. A simple reduction algorithm is to process test cases one-by-one and discard any test case that does not increase the coverage.

With this reduction algorithm it is possible to form seven distinct methods for conflict handling. These seven methods are evaluated and compared in an experiment described in paper IV [GOM07]. In the experiment, the number and types of conflicts, as well as the size of the IPM and the used coverage criterion are varied. All in all, 2568 test suites with a total of 634,263 test cases were generated. The conflict handling methods are evaluated on the number of test cases in the final conflict-free test suites.

In addition to the size of the test suite, other properties of the conflict handling methods could also influence the tester’s choice. Four of these
properties are possibilities to automate, time consumption, ease of use, and applicability of the conflict handling methods. Because these properties heavily depend on potential tools used, experimental evaluation using one specific tool is essentially meaningless. Instead these properties are discussed from an analytical perspective in paper IV [GOM07].

6.5 Validation of the Combination Strategy Test Process

Sections 6.1 to 6.4 describe the key parts of the combination strategy test methodology. The main aim of this research is to determine if combination strategies are feasible alternatives to test case selection methods used in practical testing. Hence, an important part of this project is to validate the use of combination strategies in a practical setting and to compare the results with the results of using alternative methods.

Section 5 describes the challenges of using real settings to evaluate a theory. In this research project, these challenges caused the validation study to be split into three separated but related studies with partially different focuses [GDOM06]. These are: (a) application of the combination strategy test process on a large test problem, (b) a performance comparison between combination strategies and the currently employed test case selection methods, and (c) evaluation of the input parameter modeling step of the combination strategy test process.

All three studies are staged in existing industrial settings. Most notably, this includes using real testers and solving test problems as they occur in real life.

In study (a) combination strategies were used to test a large and complex test problem involving parts of the functionality of an embedded system. The combination strategy test process described in section 6.1 was used as a basis for the experiment. Selection of combination strategies and input parameter modeling were performed by the author of this thesis. The purpose of this study was to investigate if the combination strategy test process works for real, large-size problems.

Study (b) also involved an embedded system. In this study a more limited part of the functionality was used. The purpose of this study was to compare the results of applying the combination strategy process with results of currently used testing practices.
6.6 Overview of Articles

Study (c) focused on input parameter modeling. An experiment was staged with professional software testers, in which they were asked to apply the input parameter modeling method to a specification. The resulting IPMs were collected and analyzed to determine the feasibility of the input parameter modeling method. The motivation for the third study is that it is important that input parameter modeling can be conducted by non-experts.

6.6 Overview of Articles

The following six papers are included in this thesis and figure 6.6 on page 45 shows how the papers relate to each other and to the combination strategy test process.

This paper describes state-of-practice in twelve software testing organizations. It provides motivation for this research.

This paper surveys the previous research on combination strategies. In particular it contains descriptions of a large number of combination strategies and their static properties, for example, support for coverage criteria and algorithmic complexity.

This paper compares dynamic properties, for example, found faults and size of test suite, of a number of combination strategies in a controlled experiment.
IV Managing Conflicts when Using Combination Strategies to Test Software - M. Grindal, J. Offutt, J. Mellin - in Proceedings of the 18th Australian Conference on Software Engineering (ASWEC2007), Melbourne, Australia, 10-13 April 2007 [GOM07]
This paper describes and evaluates a number of methods to handle conflicts in the input space of the test object. The evaluation is conducted through a controlled experiment.

This paper describes a method for input parameter modeling and contains results of an experiment in which the method is used by a number of non-experts to construct input parameter models from a specification.

This paper contains three related studies, which together form a proof-of-concept of the combination strategy test process and indirectly indicate that combination strategies are feasible alternatives to currently used test case selection methods.

Papers V and VI have some overlap since they report details of the same study (S5c in section 5.2. Paper V contains details on the method and summaries of the experiment and the results of the experiment, whereas paper VI is focused more on describing the details of the experiment and the results thereof.
Figure 6.6: Papers included in this research.
As was described in section 5.2, this research is based on five studies. This chapter contains the main results from the studies. The results are presented in the same order as in chapter 6, with the exception of the first section on the combination strategy test process, which has not been explored in isolation. After the results have been presented, this research is contrasted with related work in section 7.5. Finally, the methods and results are discussed in section 7.6.

### 7.1 Combination Strategy Selection

Section 6 presents a test process custom designed for combination strategies. The first activity in this process focuses on selection of one or more combination strategies to use when faced with a testing problem. Further, section 6.2 lists a number of important properties to consider. As a result of this research a number of combination strategies have been investigated with respect to these properties. Detailed data from these investigations are contained in papers II [GOA05] and III [GLOA06].

Tables 7.1 and 7.2 show detailed data on these properties for six combination strategies. These are Each Choice (EC) [AO94], Base Choice (BC) [AO94], Orthogonal Arrays (OA) [Man85, WP96], In-Parameter-Order (IPO) [LT98], Automatic Efficient Test Generator (AETG) [BJE94, CDKP94], and All Combinations (AC). The AETG combination strategy supports several
levels of coverage, which explains the two AETG entries in the table. In the formulae, \( t \) is the level of coverage, \( N \) denotes the number of IPM parameters, \( V_i \) is the number of values of the \( i \)th IPM parameter and \( V_{\text{max}} = \max_{i=1}^{N} V_i \). For example, \( V_{\text{max}} = 3 \) for the IPM given in Figure 6.4 on page 39 since parameter A contains three elements.

\[
\begin{align*}
\text{Comb. Strategy} & & \text{Coverage Criteria} & & \text{Test Suite Size} & & \text{Algorithm Complexity} \\
\text{EC} & & 1\text{-wise} & & V_{\text{max}} & & O(V_{\text{max}}) \\
\text{BC} & & \text{base choice} & & 1 + \sum_{i=1}^{N} (V_i - 1) & & O(N \times V_{\text{max}}) \\
\text{OA} & & 2\text{-wise} & & \sim V_{\text{max}}^{2} & & O(1) \\
\text{IPO} & & 2\text{-wise} & & \sim V_{\text{max}}^{3} & & O(V_{\text{max}}^{2}N^{2}\log(N)) \\
\text{AETG} & & 2\text{-wise} & & \sim V_{\text{max}}^{t} & & O(V_{\text{max}}^{2}N^{2}\log(N)) \\
\text{AC} & & N\text{-wise} & & \prod_{i=1}^{N} V_i & & O((V_{\text{max}})^{N}) \\
\end{align*}
\]

Table 7.1: Properties of some combination strategies (I).

\[
\begin{align*}
\text{Comb. Strategy} & & \text{fault type} & & \# \text{ faults} & & \text{tool support} & & \text{conflict handling} & & \text{predefined test cases} \\
\text{EC} & & 1\text{-factor} & & 107 & & \text{Yes} & & \text{Yes} & & \text{Yes} \\
\text{BC} & & 1\text{-factor} & & 119 & & \text{Yes} & & \text{No} & & \text{No} \\
\text{OA} & & 2\text{-factor} & & 117 & & \text{No} & & \text{No} & & \text{No} \\
\text{IPO} & & 2\text{-factor} & & - & & \text{Yes} & & \text{Yes} & & \text{Yes} \\
\text{AETG (2-wise)} & & 2\text{-factor} & & 117 & & \text{Yes} & & \text{Yes} & & \text{Yes} \\
\text{AETG (t-wise)} & & \text{t-factor} & & - & & \text{No} & & \text{No} & & \text{No} \\
\text{AC} & & N\text{-factor} & & 120 & & \text{Yes} & & \text{Yes} & & - \\
\end{align*}
\]

Table 7.2: Properties of some combination strategies (II).

The supported coverage criteria is a fundamental property of combination strategies. The underlying algorithms of each combination strategy are defined to support one or more specified coverage criteria, which is illustrated in Table 7.1. The relations between different coverage criteria can
be expressed in a subsumption hierarchy. Rapps and Weyuker [RW85] define subsumption: coverage criterion X subsumes coverage criterion Y if and only if 100% X coverage implies 100% Y coverage. (Rapps and Weyukers’ original paper used the term inclusion instead of subsumption). Figure 7.1 shows a subsumption hierarchy for coverage criteria supported by combination strategies. The left, boxed-in, column of coverage criteria do not rely on semantic information (i.e., information about what the model represents). Detailed information about the coverage criteria and the subsumption hierarchy can be found in paper II [GOA05]. It explains why t-wise coverage does not subsume t-wise valid coverage.

![Figure 7.1: Subsumption for combination strategy coverage criteria.](image)

The size of the generated test suite is heavily dependent on the specific coverage criterion and the algorithm used to satisfy that coverage criterion. As mentioned in section 6.2, some algorithms are deterministic while others rely to some degree on randomness. For the deterministic combination
strategies, for example, EC, BC, and AC, it is possible to precisely calculate the size of the test suite based on the number of IPM parameters and IPM parameter values.

The other three combination strategies in Table 7.1, that is, OA, IPO, and AETG all contain an element of randomness. These combination strategies may generate test suites with varying contents and sizes from the same IPM at different times. Hence, the exact size of the test suite cannot be stated in advance. However, in most cases the variation in test suite size seems to be small which makes it possible to state an approximate size. For more details on test suite sizes see paper III [GLOA06]).

The algorithms of some of the more complex combination strategies, for instance, IPO [LT98] and AETG [CDPP96] are time consuming. The algorithmic complexity is a static property and can thus be assessed analytically. OA does not rely on an algorithm in the same sense as the others. Based on the assumption that precalculated Orthogonal Latin Squares [Man85] can be used, the algorithmic complexity of OA is \( O(1) \). For \( t \)-wise, AETG \( \Omega \) denotes the floor for the algorithmic complexity, which is given since the exact value depends on the value of \( t \).

The types of targeted faults depend on the coverage level and the amount of semantic information available in the IPM. Dalal and Mallows [DM98] suggest a model for software faults in which faults are classified according to how many IPM parameters (factors) need distinct values to cause the fault to result in a failure. A \( t \)-factor fault is triggered whenever the values of some \( t \) IPM parameters are involved in triggering a failure. Each possible fault can be specified by giving the combination(s) of values relevant for triggering that fault.

Recall from section 6.2 that it is impossible to predict how many faults a combination strategy will find. Within the scope of this research an experiment was conducted, which involved five test objects with a total of 128 faults. The numbers reported in Table 7.2 refer to this experiment. Although no general conclusions can be drawn based on these figures due to representativeness issues it is surprising that the more simple combination strategies detect as many faults as they do. The main reason for this behavior is that many of the faults in the study were simple in the sense that specific values of only one or two parameters are needed to trigger the fault. A more detailed explanation of this behavior can be found in paper III [GLOA06]).

The tool support described in Table 7.2 reflects the current situation and is subject to change over time. Hence, tool support for AETG \( t \)-wise is likely...
to emerge if there is a need for it. As previously stated, the OA combination strategy does not rely on an algorithm in the strict sense. Hence tool support for OA is less likely to appear.

The conflict handling property describes the possibility of adding conflict handling support to the combination strategy algorithm. In the case of BC, its associated coverage criterion makes conflict handling unclear. For the OA combination strategy the same reason as in the previous cases, that is, lack of algorithm description, limits the possibility of adding conflict handling support.

Support for predefined test cases means that the combination strategy takes a set of test cases, that is supplied in advance by the tester, and selects further test cases such that the coverage criterion is satisfied. This feature permits legacy test cases to be used. It also allows a combination strategy to be combined with other test case selection methods in an efficient way.

7.2 Input Parameter Modeling

Input parameter modeling plays an important role when using combination strategies. The constructed IPM affects both the effectiveness and the efficiency of the test process [DHS02]. As there is an element of creativity involved in the creation of IPMs it is vital to provide support to the tester in the modeling task. An input parameter modeling method is presented in section 6.3. In the context of this research, this input parameter modeling method has been investigated in an experiment. This section summarizes the results from this experiment. More details on the experiment and its results can be found in paper VI [GDOM06].

In the experiment, nineteen professional testers were given a one hour tutorial on combination strategies with focus on the input parameter modeling method. After the tutorial, the subjects of the experiment were asked to independently create IPMs from a specification of a priority queue. The subjects’ solutions were then graded with respect to number of desired properties. The results of the input parameter modeling experiment is shown in Table 7.3.

Fifteen of the nineteen subjects managed to create an IPM from which test cases can be generated. Three of the four subjects whose IPMs could not be used for test case generation reported no experience in applying other structured test case selection methods, such as equivalence partitioning and
Table 7.3: Number of solutions (of 19) that satisfy different properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Overall</td>
<td>-</td>
</tr>
<tr>
<td>A1. Model can be used for test case generation</td>
<td>15</td>
</tr>
<tr>
<td>A2. Semantic information indicated in IPM</td>
<td>3</td>
</tr>
<tr>
<td>A3. Conflicts indicated</td>
<td>5</td>
</tr>
<tr>
<td>B. IPM Parameters</td>
<td>-</td>
</tr>
<tr>
<td>B1. Concept correctly used</td>
<td>14</td>
</tr>
<tr>
<td>B2. All aspects of test object covered</td>
<td>6</td>
</tr>
<tr>
<td>B3. No overlap of IPM parameters</td>
<td>17</td>
</tr>
<tr>
<td>C. IPM Parameter Values</td>
<td>-</td>
</tr>
<tr>
<td>C1. Concept correctly used</td>
<td>16</td>
</tr>
<tr>
<td>C2. No overlap of IPM Parameter Values</td>
<td>18</td>
</tr>
<tr>
<td>C3. Valid and invalid IPM Parameter Values included</td>
<td>17</td>
</tr>
</tbody>
</table>

boundary value analysis.

Except for the three properties “semantic information indicated” (A2), “conflicts indicated” (A3), and “all aspects covered” (B2), the number of solutions satisfying the other properties vary around 15.

The low results in inclusion of semantic and conflict information in the subjects’ solutions can be explained by the fact that the tutorial did not stress how test cases are generated by combination strategies. This is not a problem since adding this information afterwards is inexpensive and does not affect the contents of the IPM. Covering all aspects is difficult, since input parameter modeling relies to some extent on experience and creativity. Further training, checklists, and reviews are different methods to remedy this problem.

The results from this experiment show that it is possible to tutor persons with proper testing background to perform input parameter modeling, according to the suggested input parameter modeling method, such that the resulting models are useful.

7.3 Conflict Handling in the IPM

In section 6.4 seven methods for handling conflicts in the input space of the test object are mentioned. The point in the test process when these methods
7.3 Conflict Handling in the IPM

should be applied varies with the method. Further, some of the methods also affect the input parameter modeling process. Hence it is important to decide from the start which conflict handling method to use.

The abstract parameter and sub-models methods are both applied during input parameter modeling. The avoid method is an extension of the combination strategy algorithm that prohibits conflicting combinations to be selected. The replace method removes conflicts from the final test suite. The abstract parameter, sub-models, and replace methods can be combined with a reduction mechanism. This reduction mechanism filters test suites and removes any test case not increasing the coverage.

To determine the efficiency of these conflict handling methods, they were compared in an experiment. In the experiment, the number and types of conflicts, as well as the size of the IPM and the coverage criterion used, are varied. The response variable in the experiment is the number of test cases in the final conflict-free test suite. Details of this experiment can be found in paper IV [GOM07].

Table 7.4 shows the sum of sizes of the test suites in two experiments for the seven investigated methods. In the small experiment, one or two conflicting pairs were used and test suites for 1-wise and 2-wise coverage were generated. In the large experiment, conflicts with up to ten conflicting pairs were used and coverage was extended to also include 3-wise coverage. In the large experiment the abstract parameter method was omitted due to large execution times of the experiment. Long execution times are indirectly visible in the small experiment where the abstract parameter method generated about four times as many test cases as the rest of the methods. Further, analysis of the results from 2-wise coverage indicates that the abstract parameter will perform even worse for 3-wise coverage than in the case of 2-wise coverage.

The abstract parameter and sub-models methods are both applied during input parameter modeling. The avoid method is an extension to the combination strategy algorithm that prohibits conflicting combinations to be selected. The replace method removes conflicts from the final test suite. Reduction filters a test suite and removes any test case not increasing the coverage.

Statistical analysis of all the results shows that the abstract parameter and sub-models methods (even with reductions) perform worse than the replace and avoid methods. The avoid method seems to be the best but this cannot be verified with statistical significance.
In addition to the size of the test suite, other properties of the conflict handling methods may also influence the tester’s choice of method. Four such properties are: possibilities to automate, time consumption, ease of use, and applicability of the conflict handling methods.

The avoid and the replace methods are possible to fully automate, although the avoid method requires a different implementation for each combination strategy since it needs to be built into the actual algorithm of the combination strategy. The abstract parameter and sub-model methods are more difficult to automate since they require transformations of the IPM.

When test suites are generated, with the exception of the abstract parameter method, time consumption is dominated by test case generation, not conflict handling.

To operate the combination strategies, the avoid and replace methods only require a command to be invoked. The abstract parameter and sub-model methods require the IPM to be adjusted, a step that probably requires human intervention.

The abstract parameter, sub-models, and replace methods are all completely general with respect to the combination strategy used. The avoid method has to be integrated into the combination strategy, which may not be possible for some combination strategies. For instance, OA is one combination strategy that cannot be used with the avoid method. More importantly, if third-party combination strategy tools are used, it may not be possible to integrate the avoid method into the tool since the source code may not be available.

The overall conclusion of the experimental results and the following discussion is that the avoid or reduced replace methods are superior to the
Applying the Combination Strategy Test Process

7.4 Applying the Combination Strategy Test Process

The previous section (7.3) describes the results of validating the input parameter modeling method (study S5c in section 5.2), which is part of the study to demonstrate the practical use of the combination strategy process. The results of the two remaining parts of this study (applying combination strategies to a large testing problem, study S5a in section 5.2, and comparing the combination strategy generated test cases with test cases derived by other means, study S5b in section 5.2) are presented in this section. Further details on all three studies can be found in paper VI [GDOM06].

In the concluding subsection 7.4.3 validation of the goals of this thesis is discussed.

7.4.1 Handling Large and Complex Test Problems

The complexity study (study S5a) was conducted in a case study at a company with a complex testing problem. This testing problem consists of a programmable event handler based on Boolean expressions. 100 different Boolean input signals are available to the user. These can be toggled using the unary \textit{NOT} operator, combined arbitrarily using any of the six standard binary Boolean operators \textit{AND}, \textit{OR}, \textit{NAND}, \textit{NOR}, \textit{XOR}, and \textit{XNOR} or the two ternary \textit{flip-flop} operators.

In addition, two ternary and one four-input \textit{counter functions} can also be used in the expressions. The Boolean input signals are continuous over time so there also exist five unary \textit{timing operators}, for instance, delay all events, delay up flanks, delay down flanks and so on. The output from an expression can be connected to some physical output, for instance a sound alarm or a lamp.
The event handler, viewed as a test problem, has an enormous input space. For instance, it is possible to vary the number of Boolean input signals and the types of operators included in an expression. Further, it is also possible to vary the bit vectors and their timing on the input signals. The vast number of possibilities makes it impossible, in practice, to test everything. Combination strategies is therefore an interesting alternative for this test problem.

The final solution is based on four different IPMs with different foci. These are (1) Boolean input signals, (2) Boolean operators including flip-flops, (3) counter functions, and (4) timing operators. Table 7.5 shows an overview of the number of test cases required to satisfy different levels of coverage for the different IPMs. Several of the IPMs have so few parameters that \( N \)-wise (full) coverage can be reached within the limits of the table. Higher coverage of these IPMs are thus not applicable. These situations are indicated with \( N \) in the table. Already for low coverage levels, the input signal and timer IPMs yield so many test cases that test case generation for higher levels of coverage is meaningless. This is represented by (-) in the table.

<table>
<thead>
<tr>
<th>IPM</th>
<th>contents</th>
<th>no.</th>
<th>1-wise</th>
<th>2-wise</th>
<th>3-wise</th>
<th>4-wise</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPM1</td>
<td>input signals</td>
<td>-</td>
<td>101</td>
<td>816</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>IPM2</td>
<td>unary</td>
<td>1</td>
<td>2</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td></td>
<td>binary</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td></td>
<td>ternary</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>( N )</td>
</tr>
<tr>
<td>IPM3</td>
<td>small counters</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>( N )</td>
</tr>
<tr>
<td></td>
<td>large counters</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>IPM4</td>
<td>timers</td>
<td>5</td>
<td>4</td>
<td>23</td>
<td>86</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.5: Test cases for varying coverage of the event handler IPMs.

In the case of the input signals, a naive solution would be to test each signal in isolation, thus requiring \( 2 \times 100 \) test cases. An alternative solution is to use OR operators to create expressions with input signals. An expression can then be tested using an MCDC approach [CM94] by setting one signal at a time true. This approach also allows for varying coverage by considering the inclusion of individual signals, pairs of signals, triplets of signals etc, in the expressions. The results in the table are derived using this approach.

In the table, the values in bold indicate the level of coverage used during execution. All in all, 1352 test cases were executed. These test cases found
four major faults. Additional experiences from this experiment are that (i) constructing the IPMs took less than a day, (ii) once the IPMs were constructed, changes in the specification were easy to handle, and (iii) several alternative abstract test suites with different levels of coverage was cheap to generate. This proved beneficial since it allowed a quantified discussion with the owner of the test problem concerning test quality vs. cost and risk.

### 7.4.2 Alternative Test Case Generation Methods

The study comparing combination strategies with currently used test case selection methods (study S5b) was also conducted in the form of an industrial case study.

The test problem in this experiment is a low-level resource manager. It administers different types of communications, such as point-to-point links or package based services between processes, which may be distributed across a set of computation nodes. Within the resource manager a `set-up()` function was selected for this study. A structural modeling approach was used to create an IPM covering the input space defined by the 11 actual parameters of the `set-up` function. A twelfth parameter was added to handle the device state. The specification made it easy to identify both valid and invalid values for each of the 12 IPM parameters. The large number of invalid values in the IPM created an interest in using coverage criteria based on semantic information (see Figure 7.1 on page 49). Table 7.6 classifies the IPM parameter values into (N)ormal, (B)ase choice, and (I)invalid values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value # 1</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>B</td>
</tr>
<tr>
<td>Value # 2</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>Value # 3</td>
<td>I</td>
<td>I</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>-</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>-</td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>Value # 4</td>
<td>B</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Value # 5</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Value # 6</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.6: IPM value classification for the low level resource handler.

Table 7.7 shows the total number of test cases and the number of valid test cases for varying levels of coverage. A valid test case contains only
normal or base choice values. The large number of invalid values in the IPM is the reason for the low number of valid test cases.

<table>
<thead>
<tr>
<th>Type of Test Cases</th>
<th>1-wise</th>
<th>2-wise</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>6</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>Valid</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 7.7: Test cases for varying coverage of the resource handler IPM.

The set-up function was originally tested by a requirement based approach. Given explicitly stated requirements on the functionality, test cases were designed to verify these. This resulted in five test cases. Four of these are valid and the fifth test case is invalid, that is tests error handling.

Experiences from comparing the existing test cases with the test cases generated in the combination strategy approach are: (i) the existing test cases over-emphasize valid test cases, (ii) the structural approach used to generate the IPM results in test cases missed by the requirement approach, and (iii) the requirements based approach makes coverage assessments more difficult since it is not obvious how to test a requirement.

7.4.3 Validation of Thesis Goals

The aim of this research is to investigate if combination strategies are feasible alternatives to test case selection methods used in practical testing. To achieve this aim, a test process custom-designed for combination strategies has been refined from a generic test process. A large part of this research project has focused on defining those activities in the test process that are combination strategy specific. The formulation of the test process with its associated activities is a necessary step towards reaching the aim. This is also reflected in goals G1-G4 on page 20.

The work in formulating the test process with its activities is based on a combination of literature studies and experimental evaluations of alternative solutions. Experience in practical testing has also played a role in this work.

The final step of this research project, reflected in goal G5, is a series of validation experiments. The joint objective of these experiments is to show that the test process is tractable and that it performs at least as well as currently used test case selection methods. A common feature of all these experiments is that they are all conducted in actual industrial settings. The key results of these experiments are that (i) testers can learn to apply
7.5 Related Work

In the attempt to bridge the gap between academia and industry with respect to use of combination strategies, this thesis combines three themes: selection of combination strategy, input parameter modeling and handling conflicts in the IPM. Selection of combination strategies has previously been partly explored through evaluation of selected properties. However, there is no previous work on how to use this information to select a suitable combination strategy for a given test problem. Input parameter modeling has also previously been partly targeted but not with an explicit focus on combination strategies. Conflict handling has been recognized as an important
aspect of combination strategy usage but no explicit exploration of alternative conflict handling methods exists.

Tables 7.8 and 7.9 give overviews of the most important related work with respect to these themes. Much of the previous work has been centered around suggesting new combination strategies, which is why this has been included as a fourth theme in the tables. More details on these related works can be found in paper II [GOA05].

<table>
<thead>
<tr>
<th>Work</th>
<th>CS Evaluate</th>
<th>IPM</th>
<th>Confl.</th>
<th>CS Suggest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohen et al. [CDKP94]</td>
<td>No</td>
<td>No</td>
<td>Partly</td>
<td>Yes</td>
</tr>
<tr>
<td>Williams &amp; Probert [WP96]</td>
<td>No</td>
<td>No</td>
<td>Partly</td>
<td>Yes</td>
</tr>
<tr>
<td>Mandl [Man85]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ammann &amp; Offutt [AO94]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Lei &amp; Tai [LT98]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Williams [Wil00]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cohen et al. [CGMC03]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Shiba et al. [STK04]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Lei &amp; Tai [LT01]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 7.8: Classification of related work on combination strategies.

Most of the previous work on combination strategies has emphasized new combination strategies and compared these with previously suggested combination strategies. In the early papers, there is usually no or little comparison, which is indicated with “No” in column “CS Evaluate” in tables 7.8 and 7.9. Examples of this are the papers by Cohen, Dalal, Kajla, and Patton [CDKP94], Williams and Probert [WP96], and Mandl [Man85].

In later papers, in addition to describing new combination strategies, more emphasis is put on comparing different combination strategies. Ammann and Offutt [AO94], Lei and Tai [LT98], Williams [Wil00], Cohen, Gibbons, Mugridge, and Colburn [CGMC03], and Shiba, Tsuchiya, and Kikuno [STK04] are all examples of this. In addition, Lei and Tai [LT01] provide further information on the performance of different combination strategies without suggesting any new combination strategies. These comparisons are the basis for combination strategy selection.

Similar to Lei and Tai [LT01], this work also compares the performance of different combination strategies without suggesting any new, but in contrast
to their work, this project suggests how to use the obtained performance information to select an appropriate combination strategy for a given test problem.

The Category Partition Method (CPM) by Ostrand and Balcer [OB88] is likely to be the seminal work with respect to input parameter modeling. It is a step-by-step method to transform a specification into abstract test cases (test frames in their terminology). In a sense, the Category Partition Method can be viewed as a combination strategy with the ability to handle conflicts since the method requires all valid test frames to be included. However, selecting all valid combinations may be infeasible for larger test problems. The input parameter modeling method suggested in this work is in many ways an extension of the CPM. The main difference is in the generality. Where the CPM is custom-designed for a specific combination strategy and a certain way of handling conflicts, the method suggested in this work is completely general with respect to combination strategies and conflict handling methods.

Grochtmann and Grimm [GG93] suggest Classification Trees as a solution to the input parameter modeling problem. Classification Trees subdivides the input space based on properties of the objects in the input space. Classification Trees contains a built-in ability to handle conflicts but does not give any advice on test case selection. The input parameter modeling suggested for Classification Trees has aspects in common with the input parameter modeling method suggested in this work, for instance, the explicit goal to cover all aspects of the test object and for each aspect to cover the entire input domain. The main difference is again that this work explicitly targets combination strategies.

### Table 7.9: Classification of related work on input parameter modeling

<table>
<thead>
<tr>
<th>Work</th>
<th>CS Evaluate</th>
<th>IPM</th>
<th>Confl.</th>
<th>CS Suggest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ostrand &amp; Balcer [OB88]</td>
<td>No</td>
<td>Yes</td>
<td>Partly</td>
<td>Partly</td>
</tr>
<tr>
<td>Grochtmann &amp; Grimm [GG93]</td>
<td>No</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Mayrhauser et al. [vMSOM96]</td>
<td>No</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Chen et al. [CTPT05]</td>
<td>No</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Piwowarski et al. [POC93]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Sleuth, which was developed by von Mayrhauser, Shumway, Ocken, and Mraz [vMSOM96], is an alternative solution to the input parameter modeling problem. It has similarities with the Classification Tree approach in the sense that both methods result in hierarchical models of the test object domains. Sleuth contains a built-in, rule-based, ability to handle conflicts. Rules can be specified to describe different types of constraints, such as inheritance rules and parameter binding. Combination strategies support different levels of coverage of an IPM. Sleuth takes another approach. The tool always generates all possible test cases that satisfy the constraints, but with a mechanism to turn certain constraints on and off, the size of the final test suite can be manipulated.

Chen, Tang, Poon, and Tse [CTPT05] propose an input parameter modeling method starting from UML activity diagrams. Like Classification Trees, this method has a built-in ability to handle conflicts but give no advice on test case selection. The UML input parameter modeling method has the same similarities and differences with this input parameter modeling method as Classification Trees.

Piwowarski, Ohba, and Caruso [POC93] showed the applicability of refining the IPM by monitoring the code coverage achieved by the generated test suite. This is a completely different approach compared to the one employed in this work, since combination strategies generally are black-box methods and thus do not rely on access to the source code.

As far as the author knows, there is no previous work specifically targeting conflict handling in IPMs even though some briefly touch upon the subject. Distantly related to this area is constraint satisfaction programming [Tsa93].

Only fragmental information exists on the performance of combination strategies in real industrial settings, which is why this aspect is not included in this comparison.

7.6 Discussion

This section contains a discussion of some aspects of the methods and experiments conducted in this research. These aspects fall into the three categories (i) formulation and evaluation of research goals, (ii) software process improvement, and (iii) properties of the proposed methods.
7.6 Discussion

7.6.1 Formulation and Evaluation of Research Goals

Recall from section 4 on page 19 that the aim of this research project was to investigate if combination strategies are feasible alternatives to test case selection methods used in practical testing. The following issues were considered during the formulation of this aim and the derived research questions.

- There should be a focus on practical aspects of combination strategies.
- Goals should be measurable.
- Goals should be challenging.
- A negative result with respect to the performance of combination strategies should not jeopardize the research project.

The use of the words “feasible alternatives” in the aim was a direct result of these considerations. One implication of these words is that parts of the research had to be staged in real industrial settings. Another implication is that it is not assumed in the formulation of the goals that combination strategies will lead to success. In other words, if the performance of combination strategies had turned out to be distinctly worse than contemporarily used test case selection methods, this result would not invalidate the research.

The diversity of the state-of-practice in testing, see section 2.3, presented a challenge in the evaluation of the goals. What would be an improvement in one organization does not necessarily improve another. To avoid making claims that are not general, words like “feasible” and “tractable” have been used in the analysis and evaluation of goals. Although imprecise, the use of these words is deliberate. These words allow the readers to form own opinions based on what they consider reasonable. These words also allow for future investigations based on defined performance goals.

7.6.2 Software Process Improvements

The low test maturity reported in paper I [GOM06] motivates a discussion around introducing combination strategies in industry testing projects. Input parameter modeling being to some extent a creative and intellectual activity also deserves further discussion. Finally, aspects of reuse with respect to combination strategies and their artifacts is an interesting area to discuss.
Implementation of Combination Strategies

This research indicates that many organizations could benefit from using combination strategies in their test processes. However, the relatively low testing maturity of several software producing organizations, reported in paper I [GOM06], suggests that other test process improvements may be at least as important as improving the test case selection mechanism. For instance, several test managers in the test maturity investigation point in the direction of improving their metrics collection mechanisms. The underlying reason is that without the means for objectively assessing the current situation it is very difficult to state a valid case for any improvement.

Test Process Improvement (TPI) [KP99] is, in industry, a widely known and recognized test process improvement method. Among other things it contains a dependency matrix, which gives advice to the user on the order of which improvements should be undertaken. It suggests two preconditions for implementation of structured test case selection methods. These are training of testers and test managers and infrastructure for testware management, for example, configuration management tools and structured storage of test cases and test specifications.

In the light of these observations, it can be suspected that there exist organizations that are not ready for combination strategies. This is probably true for full-scale implementations of combination strategies, but small pilot studies are cheap and do not require more than a few hours training, hence the conclusion presented in section 7.4.3, that combination strategies are likely to be interesting to most software testing organizations. A further advantage with combination strategies, observed in the event handler case study (section 7.4.1), is the ability to handle changed requirements. During the course of the study the requirements on the event handler changed, making the original IPMs obsolete. The amount of additional work to handle these changes was very small since it was enough to change the IPMs and then automatically regenerate the test cases.

Input Parameter Modeling

As noted previously, input parameter modeling is likely to always contain an element of creativity. Hence, it is very difficult to develop an exact method for this task. The subjects of the input parameter modeling experiment, described in paper VI [GDOM06], created several different solutions to the input parameter modeling task. This can be seen as an indication of the creativity element. The results are also likely to be explained, in part, by
7.6 Discussion

the fact that the tutorial was the first contact with input parameter modeling and that the tutorial only contained theory. It is likely that the quality of the achieved solutions will improve with practice.

Another aspect of input parameter modeling, also worth mentioning, is the opportunity to exploit constraint handling for other purposes rather than to exclude apparently infeasible sub-combinations. One can imagine that some sub-combinations required for full coverage is deemed so unlikely in operation that a decision is taken to exclude these in the generation step. An alternative way of handling this situation would be to include these unlikely sub-combinations but refrain from translation of these into actual test cases.

Reuse

Reuse of tools and artifacts within an organization is a potential cost and time saver. This research has not explicitly targeted reuse. However, the ability to use pre-selected test cases in the input to some combination strategies can be seen as a kind of support for reuse of test cases. This is referred to by von Mayrhauser et al. [vMSOM96] as test case reuse. A further development of this thought would be to include an ninth row in Table 6.1 on page 37, with a strong relationship between test case reuse and time.

Reuse of tools is another topic relevant to this project. As described earlier, several implementations of combination strategies exist. A limitation in the reuse of tools is that the avoid conflict handling method requires access to the source code, which is usually not the case with commercial tools.

Other parts of the combination strategy test process may also be automated. One example is the translation from abstract inputs into actual test case inputs. This activity is likely to require customization to the specific test object, which is why general tools are less likely. Instead, it is probable that such tools will be developed in-house. For instance, this is the case in the experiment reported in paper III [GLOA06]. In-house developed tools simplify reuse but often require more time and effort to develop and maintain than commercial tools.

Finally, it is worth discussing reuse with respect to IPMs, which is called domain reuse by von Mayrhauser et al. [vMSOM96]. As described previously, IPMs are representations of the input space and possibly other properties of a system under test. Within the combination strategy test process reuse of IPMs is exploited in the feedback loop from the evaluation of abstract test suites in step 4. In a larger context it is also possible to reuse IPMs from one
project to another. For instance, an IPM developed for a new function in one project, may be used with a less thorough coverage criterion for regression testing purposes in a following project.

7.6.3 Properties of Methods

The introduction of new processes, methods, and tools draws attention to usage aspects of the products to be launched. Among other things usability, scalability and performance are important aspects to consider.

Usability

The British standard BS7925-1 [BS 98a] defines usability testing as testing the ease with which users can learn and use a product. This definition focuses the attention both on learning and using a product.

This research has resulted in a refined test process with several activities that need to be taught. In particular, choosing combination strategies and developing IPMs for a given test problem are activities that probably works best if taught. One important reason for this is that these activities both rely to some extent on the knowledge and experience of the individual tester. For most test problems there are likely to be several possible solutions for both problems, with similar performance. Another contributing reason for the need for training is the lack of detail presented in the existing research (including this). A research document is generally not the right place for detailed usage instructions, which would be required when properly using these products. The conclusion is that for a roll-out of combination strategies to succeed in industry there is a need for the development of training material.

This research has only partly focused on the ease of use of the suggested methods and tools. Among the methods and processes, there is certainly more to do to increase usability. At this stage, apart from the combination strategy test process, the methods and guidelines should be seen as research prototypes rather than finished products.

As far as the available tools (http://www.pairwise.org/) are concerned it is impossible to make any general claims about the ease of use. There is a big diversity among the tools and there is also much development currently taking place.
7.6 Discussion

Scalability
The major benefit with combination strategies is to handle large test problems. The combination strategy test process contains three mechanisms for the tester to control the size of the final test suite. First, the choice of combination strategy will have a great impact on the test suite size. Second, the tester can to a large extent influence the size of the IPM, which will also impact the final test suite size. Third, the process itself contains a checkpoint where the size of a suggested test suite can be checked and the previous process steps revisited. Together, these three mechanisms provide an opportunity for the tester to iteratively find an acceptable compromise between test quality and time consumption. This means that from a scalability point of view, the combination strategy test process scales well with the size of test problem.

Although the test process itself scales well, the combination strategy algorithms for higher coverage (2-wise and higher) have scalability issues. Among the studied combination strategies, IPO [LT98] has the lowest algorithmic time complexity for 2-wise coverage ($O(V_{max}^3N^2\log(N))$ where $N$ is the number of parameters in the IPM and $V_{max}$ is the number of values of the parameter with the most values). With this time complexity, at some point, the size of the IPM will have a major impact on the test suite generation time. However, as shown in section 7.4.1, even a prototype implementation in Perl can handle large IPMs (100 parameters, each with two values). It is our belief that a majority of test problems in industry can already be managed in a satisfactory manner with existing algorithms and tools.

Performance
Performance of combination strategies can be assessed in several ways. The size of the generated test suite is obviously an important performance indicator since it directly affects the efficiency of the testing. Equally important are the types and number of faults found, which together with the achieved coverage are measures of the effectiveness of the process. Coverage and types of faults found for a given combination strategy are relatively simple to assess. In both cases this enables comparisons between different combination strategies but since both coverage and types of faults are defined based on properties of combination strategies it is difficult to compare these metrics with similar metrics from other test case selection methods. The number of faults found makes comparisons easier but due to representativity issues of
the existing faults no general conclusions can be drawn from these results. In summary, comparing the performance of different test case selection methods is difficult.
Chapter 8

Conclusions

This chapter provides a summary of this thesis and describes its contributions to this area of research. This chapter also outlines some directions of future work.

8.1 Summary

The aim of this thesis is to investigate if combination strategies are feasible alternatives to test case selection methods used in practical testing. Combination strategies are test case selection methods designed to handle combinatorial explosion. Combinatorial explosion can occur in testing when a test object can be described by a number of parameters, each with many possible values. Every complete combination of parameter values is a potential test case. Due to its generality, combinatorial explosion is a common problem in practical testing. The effect of combinatorial explosion is that it is infeasible to test every possible combination of parameter values. Combination strategies handle the combinatorial explosion by identification of a tractable subset of all combinations. This selection is based on coverage, which enables objective evaluation of the selected combinations.

To investigate the tractability of combination strategies in practical testing, this research formulates a test process custom-designed for combination strategies. The combination strategy test process is a refinement of a generic
test process containing the four activities (i) planning, (ii) specification, (iii) execution, and (iv) test stop decision. Most of the tailoring of the generic test process is focused on the specification activity. Within the scope of test specification, the custom-designed test process contains activities for (a) selection of suitable combination strategies, (b) input parameter modeling, and (c) handling conflicting values in the input space of the test object. This research suggests guidelines or methods for each of these three activities.

The custom-designed test process, including its guidelines and methods, is the result of a series of studies and experiments. Existing combination strategies are surveyed in paper II [GOA05] and a subset of these are compared in an experiment presented in paper III [GLOA06]. These two papers form the basis for the activity of selecting a suitable combination strategy for a given test problem.

A method for input parameter modeling is presented and partly validated in paper V [GO07]. Input parameter modeling is the process of converting requirements on the test object into a model suitable for combination strategies.

Paper IV [GOM07] describes an experiment which experimentally evaluates seven alternative ways of handling conflicts in the IPM. The results of the experiment show that the two best ways of handling conflicts are (i) to avoid the selection of combinations that contain conflicts and (ii) to post-process the set of selected combinations. In this post processing step, combinations containing conflicts are replaced by alternative conflict-free combinations such that the original coverage is maintained.

Finally, this research project contains a proof-of-concept study, which validates the combination strategy test process in three industrial settings. This study, documented in paper VI [GDOM06], shows that combination strategies can be used to identify tractable test suites for large and complex test problems. Further, the study shows that combination strategies can be more effective than other, currently used test case selection methods. Finally, the study indicates that testers with proper testing background can be trained to use combination strategies without significant effort.

The overall conclusion from this research project is that combination strategies are feasible alternatives to test case selection methods used in practical testing.
8.2 Contributions

Several aspects of using combination strategies in software testing were relatively unexplored prior to this research. In the light of the related work, presented in section 7.5, the following paragraphs describe the most important contributions of this work.

An important contribution of this work are the guidelines for selection of suitable combination strategies for a given test problem. These are based on previous explorations of the properties of various combination strategies. As described earlier, none of the previous work has elaborated on how to select combination strategies given the special circumstances of specific development projects.

Another equally important contribution of this research is the formulation of an input parameter modeling method custom-designed for combination strategies. To the author’s knowledge, no other input parameter modeling method specific to combination strategies exists. Previously suggested methods for identification of parameters and parameter values are either completely general or specific to other methods.

A third contribution, related to the two previous, is the advice on how to effectively and efficiently handle conflicts in the IPM. No previous evaluation of conflict handling methods exists. The relationship between these three contributions is through the suggested combination strategy test process, which is in itself a contribution.

The three above mentioned contributions are combined with previous work into a combination strategy test process, which is a fourth contribution of this research project.

The state-of-practice investigation included in this research is another type of contribution. The results, which for instance show surprisingly low usage of structured test case selection methods, are important as such. The most important conclusion from the state-of-practice study is that organizations need to focus more on structured metrics programs to be able to establish the necessary foundation for change, not only in testing.

Further, the results from this study confirm results from several previously performed studies [GSM04, NMR+04, RAH03]. Confirming previously achieved results is in itself a valuable contribution.

Just like the state-of-practice study, the case studies conducted within this research project have resulted in several contributions. First, the results themselves, that is that combination strategies have been shown to be
feasible in industrial contexts. Second, the descriptions of these case studies enable future replication of these studies. From an analytical generalization perspective, this is an important contribution since the representativity problem makes it difficult to base theory-building on statistical generalization. Third, combination strategies are no longer unexplored territory in industry. The successful results from applying combination strategies in industrial projects may be used as an argument for staging more extensive case studies, which previously may have been considered too risky.

The knowledge transfer occurring naturally during research at industry sites also deserves to be mentioned explicitly as a contribution of this research project. For instance, one of the organizations in which a case study was conducted now use combination strategies as part of their testing strategy. Knowledge transfer to industry has also occurred in other ways, for instance by conducting seminars and training sessions.

8.3 Future Work

The old truism “the more you know, the more you realize you don’t know” is certainly true in this case. It is easy to identify a large number of potential research directions. In the paragraphs to follow, a number of interesting questions and research directions are described. These are grouped into four main themes (i) combination strategies, (ii) input parameter modeling (iii) automation, and (iv) general research on test case selection methods.

Combination Strategies

This work shows that combination strategies can be applied to a large number of test problems in industry. Although this research contains some comparisons with other test case selection methods, little is still known about the actual performance of combination strategies applied to different testing problems in real industrial settings. Further studies of the efficiency and effectiveness of combination strategies are desirable to determine in which cases combination strategies can make a difference. This is especially true in the light of the validation study, of this research project, being divided into three different parts.

The explicit industrial perspective of this research project has resulted in a shift in focus towards the less complex (and cheaper) coverage criteria, such as 1-wise, 2-wise, and base choice. In many industrial settings, time
to market is more important than quality or cost. This drives organizations towards finding fast solutions to their problems, which means that the less complex coverage criteria are of special interest. Moving up in the subsumption hierarchy (see fig 7.1) will certainly increase the test quality but at the same time increase the cost for testing. At some point, it is likely that there will be diminishing returns, that is the increase in test time is higher than the increase in quality. The study in paper III, can be used as an example of this. Of the 120 detected faults, 95 are guaranteed to be discovered by around 180 test cases if 2-wise testing is used. Moving to 3-wise coverage, would detect all 120 faults but would at least require 1022 test cases. (The minimum number of test cases required for 3-wise coverage is the product of the sizes of the three largest parameters). Further research on t-wise coverage, where t is larger than two is certainly needed.

There is also room for more combination strategies. In particular, combination strategies that support variable-strength coverage, that is, different levels of coverage for different IPM parameters and IPM parameter values within the same IPM. A related area of interest is combination strategies that build their selection strategies more on semantic information. In both cases, these properties can be used to improve the quality of test cases, for example by more closely mimicking actual usage.

Combination strategies may help in other areas of testing, for instance in testing real-time systems. One important property of a real-time system is timeliness. Timeliness is the ability of the system to deliver a result at or within a specific time. In timeliness testing different execution orders play a crucial part [Nil06]. It is possible that an IPM can be used to express different execution order patterns. With a similar approach, IPMs may be used to express orderings between feature invocations and hence be useful for feature interaction testing.

**Input Parameter Modeling**

The IPM plays a crucial role in determining both efficiency and effectiveness of the testing. This research presents a general method for input parameter modeling. This method is based, to a large extent, on the current knowledge of the behavior of combination strategies. As this knowledge is likely to increase with an increased use of combination strategies, it is also likely that there is room for new and refined methods for input parameter modeling.
Conclusions

Automation
One of the key features of combination strategies is that semi-formal models of the test object input, that is, IPMs are created. Automatic generation of the abstract test cases is already implemented. One of the strengths of combination strategies is that it should be possible to increase the level of automation.

First, for some applications it is possible to automatically translate abstract test cases into actual test case inputs. An example of this was used in the experiment presented in paper III [GLOA06]. Translation into actual test cases requires a mapping of information from IPM parameter values to values in the input space of the test object. What this mapping looks like and when it is applicable are largely unexplored.

A second topic for potential automation is the test case results. For this to work it is necessary to include behavioral information in the IPM. In the general case this may be too difficult or expensive but in some cases it may be possible. For instance, it may be possible to add high-level behavioral information in terms of normal or abnormal termination. Such a scheme can be useful for robustness testing.

A third topic for potential automation is to amend the combination strategy test process with a third evaluation point with corresponding feedback loops. The idea is to divide step 5 - test case generation into two steps, 5a and 5b. In step 5a, abstract test cases are transformed into real test case inputs. These are then automatically executed while code coverage is monitored. If coverage is sufficient, then step 5b is invoked, where expected results are added to the test cases and the ordinary progress through the process continues. Otherwise, feedback loops return the tester back to the first two steps of the process in order to increase the amount of test cases and thus raise the code coverage.

Research on Test Case Selection Methods
All of these described directions for future work on combination strategies will at one point or another require evaluation of suggested enhancements and solutions. Both the research community and representatives from industry want, as far as possible, results from such evaluations to be generalizable. Hence, there is a great need for studies with respect to representativity. This need spans from general areas such as types of test objects and distribution of types of faults to combination strategy specific areas such as types of conflicts in IPMs.
Generalization of results may not be realistic in every situation. In cases where generalization is difficult to reach, much is gained if the research is conducted and documented in such a way that experiments can be repeated with controlled changes. This enables researchers to develop confidence in the comparison of results. A repository of described test problems to be used for research on test case selection methods would greatly help in this area.

All in all, it can be concluded that the research on applying combination strategies to testing has only just begun.
References


References


References


References

Applic


No 582 Vanja Josifovski: Design, Implementation and...


No 872

No 823

No 828

No 833

No 852

No 867

No 876

No 893

No 909

No 920

No 929

No 933

No 937

No 938

No 945

No 946

No 947

No 952

No 954

No 955

No 986

No 987

No 998

No 999

No 1004

No 1005

No 1006

No 1008

No 1009

No 1010

No 1011

No 1012
Linköping Studies in Information Science


