1-1-2019

Evaluating Software Testing Techniques: A Systematic Mapping Study

Mitchell Mayeda
University of Denver

Evaluating Software Testing Techniques: A Systematic Mapping Study

Abstract

Software testing techniques are crucial for detecting faults in software and reducing the risk of using it. As such, it is important that we have a good understanding of how to evaluate these techniques for their efficiency, scalability, applicability, and effectiveness at finding faults. This thesis enhances our understanding of testing technique evaluations by providing an overview of the state of the art in research. To accomplish this we utilize a systematic mapping study; structuring the field and identifying research gaps and publication trends. We then present a small case study demonstrating how our mapping study can be used to assist researchers in evaluating their own software testing techniques. We find that a majority of evaluations are empirical evaluations in the form of case studies and experiments, most of these evaluations are of low quality based on proper methodology guidelines, and that relatively few papers in the field discuss how testing techniques should be evaluated.

Document Type

Thesis

Degree Name

M.S.

Department

Computer Science

First Advisor

Anneliese Andrews, Ph.D.

Second Advisor

Scott Leutenegger, Ph.D.

Third Advisor

Michael Keables, Ph.D.

Keywords

Evaluation, Gaps, Quality, Software testing, Systematic mapping study, Techniques

Subject Categories

Computer Sciences | Software Engineering

Evaluating Software Testing Techniques: A Systematic Mapping Study

---

A Thesis

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Mitchell Mayeda

June 2019

Advisor: Anneliese Andrews

Author: Mitchell Mayeda
Title: Evaluating Software Testing Techniques: A Systematic Mapping Study
Advisor: Anneliese Andrews
Degree Date: June 2019

Abstract

Software testing techniques are crucial for detecting faults in software and reducing the risk of using it. As such, it is important that we have a good understanding of how to evaluate these techniques for their efficiency, scalability, applicability, and effectiveness at finding faults. This thesis enhances our understanding of testing technique evaluations by providing an overview of the state of the art in research. To accomplish this we utilize a systematic mapping study; structuring the field and identifying research gaps and publication trends. We then present a small case study demonstrating how our mapping study can be used to assist researchers in evaluating their own software testing techniques. We find that a majority of evaluations are empirical evaluations in the form of case studies and experiments, most of these evaluations are of low quality based on proper methodology guidelines, and that relatively few papers in the field discuss how testing techniques should be evaluated.

Acknowledgements

I am extremely grateful to my thesis advisor, Dr. Anneliese Andrews, whose remarkable guidance, encouragement, and expertise made this research possible. I cannot thank her enough for her time and effort in supporting this endeavor. I would also like to thank the Computer Science faculty at the University of Denver for their exemplary work as educators. I am blessed to have been a student of such passionate and knowledgeable teachers. I would like to thank my examining committee members, Dr. Scott Leutenegger and Dr. Michael Keables, for agreeing to serve on my oral defense committee. I would also like to thank my friends and family for their support, and give a special thanks to Dr. Leutenegger and Meredith Corley for going well out of their way in encouraging me to succeed. Last but not least, I would like to thank my parents for their unbounded love and support. This accomplishment would not have been possible without them.

Contents

List of Figures

List of Tables

1 Introduction

Software testing is a vital process for detecting faults in software and reducing the risk of using it. With a rapidly expanding software industry and a heavy reliance on increasingly prevalent software, there is a serious demand for employing software testing techniques that are efficient, scalable, applicable, and effective at finding faults. Utilizing such testing techniques to reduce the risk of using software can help avoid catastrophes that jeopardize safety or cost companies millions of dollars, such as when Intel spent $475 million replacing processors due to inaccurate floating point number divisions [1]. Given the importance of applying high-quality software testing techniques, understanding how they should be evaluated is also crucial. What is the current state of the art in research evaluating software testing techniques and where are there gaps in research? As a researcher looking to evaluate a particular technique, how should I do so?

A systematic mapping study is a methodology that is useful for providing an overview of a research area by classifying papers in it and counting the number of them belonging to each category in the classification. For example, one can classify papers in a field by their publication year with each year being a category in the classification. Counting the number of papers belonging to each category (in this case the number of papers published each year) can give us an idea of activity level in the field over time. Similarly, classifying papers based on their content gives us a sense of what content is commonly researched and where there are research gaps. Such classifications can also provide higher level insight regarding the current state of the art. As an example from this thesis, classifying papers by the method they utilized for evaluating software testing techniques gives a very general sense of which methods are commonly used for evaluations. Considering this classification with others such as the testing technique type or dimension of evaluation allows us to answer more interesting questions about the state of the art: What evaluation method is most commonly used when evaluating the efficiency of mutation testing techniques? What is the distribution of evaluation methods when evaluating the effectiveness of white box testing techniques? Additionally, classifications can be used to construct a mapping from categories to sets of papers belonging to them; allowing researchers to very easily locate papers in the field belonging to categories they are interested in. Here, we utilize a systematic mapping study in the field of research evaluating software testing techniques to achieve our main goals of (1) summarizing recent publication trends and (2) identifying research gaps and the state of the art when it comes to evaluating software testing techniques. We hope by structuring the field that we can provide guidance to other researchers who are unsure of how to evaluate their particular testing technique and point them to specific papers that have evaluated similar techniques. We also hope that we can provide direction for future work and initiate improvements in areas where evaluations are of lower relative quality. Our systematic mapping process follows guidelines proposed by Petersen et al. [255] and is discussed in more detail in section 2.

1.1 Background

Other relevant papers have addressed the state of software testing technique evaluations. Juristo et al.[168] examined 25 years of empirical studies evaluating techniques in order to compile empirical results and assess the maturity level of knowledge for different testing technique families. More specifically, they collected major contributions by testing technique family and summarized significant implications of their empirical results. They additionally assessed the maturity of knowledge on relative testing technique effectiveness based on the extent that laboratory study, formal analysis, laboratory replication, and field study had been performed. Our study is similar in that it also compiles and examines empirical studies evaluating testing techniques. However, our study systematically gathers a larger set of papers in the field and categorizes them according to different classification schemes better suited for our research goals. This approach provides assistance for answering a broader range of finer-grain questions regarding testing technique evaluations by pointing researchers to sets of actual papers belonging to more specific categories they are interested in.

[137] extended the work of Juristo et al. [168] by performing a more recent examination of testing technique experiments with similar goals. The extension is similar to our research in that it utilizes a systematic mapping study to develop an understanding of the state of testing technique evaluations. Our research goals are somewhat different in that we place a particular emphasis on assisting researchers in determining how to evaluate software testing techniques in specific contexts and do not only consider experiments. For this reason this thesis provides a great deal of distinct information due to major differences in scope and classification schemes. In terms of scope, it includes other common evaluation methods such as case studies and does not exclude a large number of papers that report smaller experiments. It also includes papers providing guidelines or proposals regarding how testing techniques should be evaluated. In terms of classification schemes, we utilize 6 distinct schemes and some additional secondary categorizations of these schemes. Due to these deviations this thesis is able to answer different research questions that align more with our desire to help researchers in evaluating their testing technique.

[137] mentions 3 other papers, [96], [97], and [287], that are systematic literature reviews of regression testing technique evaluations. Our study does not include regres-

sion testing selection or prioritization techniques since we are mainly interested in the evaluation of fault-detecting software testing techniques. [164] also references a mutation testing survey. While we are interested in the state of mutation testing evaluations, the mutation testing survey is not sufficient for answering our research questions about the overall state of testing technique evaluations.

Finally, a paper by Briand [56] reports on the common threats to the validity of empirical studies evaluating the cost effectiveness of software testing techniques. This critical analysis of the field raises awareness of common threats and how they can be reduced. Our mapping study does not investigate deeply enough to confirm threats to validity that are common to certain evaluation types, but it may similarly provide some insight on the quality of current evaluations based on guidelines for proper evaluation methodology. Our study additionally brings awareness to other papers in the field that provide guidelines or propose enhancements when it comes to evaluating software testing techniques.

1.2 Thesis Layout

The next section of this thesis gives an overview of the systematic mapping process and a detailed explanation of each step in the process as it relates to our particular mapping study. Section 3 presents the study classification schemes used to classify papers into categories for this study. Section 4 presents the results of the data mapping. Section 5 provides a discussion of the results. Section 6 demonstrates the use of the resulting map with a case study. Finally, section 7 considers threats to the validity of our findings followed by a conclusion and future work in section 8.

2 Research Method

An overview of the systematic mapping process is illustrated in Fig. 2.1. Each step of the process is described in more detail in the following subsections. At a high level, we define research questions from our research goals, systematically gather a set of papers that are ideally representative of the field of interest, and then map the papers into defined categories in order to structure the field and answer our research questions.

2.1 Definition of Research Questions

We begin by deriving research questions from the main goals of this study. As stated in section 1, we would like to structure the field of research evaluating software testing techniques and develop an understanding of what is state of the art by identifying and analyzing papers in the field. The following questions are derived from the goals.

1. *RQ1:* What are the publication trends in research evaluating software testing techniques?

   a) **RQ1.1:** What is the annual number of publications in the field?

   b) **RQ1.2:** What are the main publication venues that publish papers in the field?

   c) **RQ1.3:** What is the distribution of papers in terms of academic or industrial affiliation?

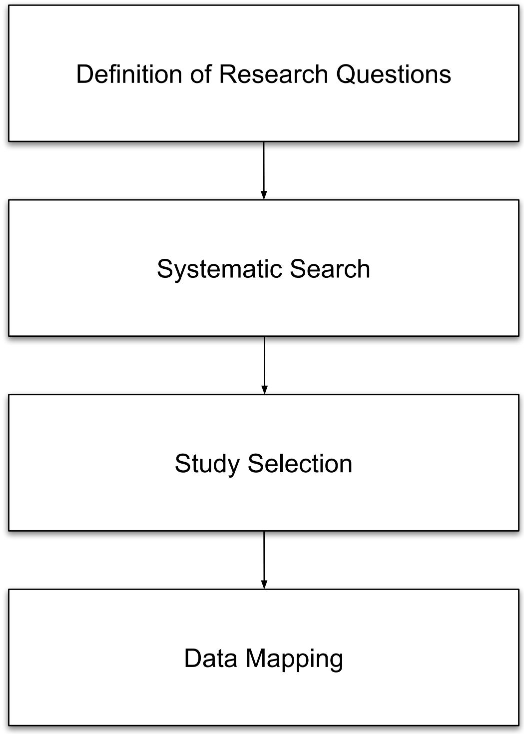

Figure 2.1: Overview of the systematic mapping process

2. *RQ2:* What is the current state of the art when it comes to evaluating software testing techniques for their effectiveness, efficiency, applicability, and scalability and where are there research gaps?

   a) **RQ2.1:** What methods have been used or proposed for evaluating software testing techniques?

   b) **RQ2.2:** What is the distribution of methods used for evaluating software testing techniques?

   c) **RQ2.3:** What is the distribution of dimensions being evaluated?

   d) **RQ2.4** What is the distribution of evaluations of white-box vs black-box testing techniques?

e) **RQ2.5:** What can we say about the relative quality of evaluations made for each evaluation method?

f) **RQ2.6:** What is the distribution of papers in terms of contribution type?

g) **RQ2.7:** What is the distribution of effectiveness evaluations utilizing mutation analysis?

2.2 Systematic Search

The next step of the mapping study process is to gather a set of papers that are potentially relevant to the field of interest. We do so by systematically defining a search string, identifying important scientific databases, and then applying the search string to the identified databases to retrieve papers.

Similar to the systematic literature review performed by Nair et al. [236], our search string was derived by first splitting up the phenomena under investigation into major terms. For each major term, keywords synonymous with the term were added to it using the OR operator. The added keywords were heavily influenced by our research questions and research goal scope. Next we joined the populated major terms together with the AND operator. The resulting search string was iteratively refined by assessing its ability to generate relevant papers from small subsets of papers in the databases and modifying keywords accordingly. Doing so we arrived at the following search string:

(evaluate OR validate OR assess)
AND
(effectiveness OR efficiency OR applicability OR scalability)
AND
("software testing" OR "software verification" OR "black-box testing" OR "white-box testing")
AND
(techniques)

For scientific databases we selected some of the most common online sources:

1. ACM
2. IEEE Xplore
3. Springer
4. Wiley

Due to our fairly broad scope and interest in the current state of the art and research gaps, we limited our search to only include papers published within the last 11 years [2007 - 2017]. We also excluded books from our search results since we are interested in scholarly peer-reviewed work that is more likely to be of higher quality. Only one paper was excluded due to being written in a language other than English (the language the researchers carrying out the mapping study could read). We applied our search string to each of the online databases with these filters to obtain 7,426 potentially relevant papers.

2.3 Study Selection

The study selection process entails removing all of the irrelevant studies from the large number of search results. Figure 2.2 illustrates our study selection process along with the number of papers remaining after applying each step in the process.

We began by applying title and abstract exclusion. Title and abstract exclusion refers to excluding papers that are deemed irrelevant based on the content of their title and abstract. We will refer to the criteria used to assess a paper's relevance in this step as the content criteria. Our content criteria is heavily influenced by the research goals and their scope. A paper was deemed relevant if it (1) proposed a method or guidelines for evaluating a failure-detecting software testing technique's effectiveness, efficiency, applicability, or scalability or (2) utilized a method for evaluating a failure-detecting software testing technique's effectiveness, efficiency, applicability, or scalability. Note that for now we are only interested in failure-detecting techniques, so software testing techniques that do not detect failures such as test case prioritization and fault localization are not considered. This criteria included papers evaluating a developed tool, given that the tool implemented some failure-detecting software testing technique. If it could be determined that a paper did neither (1) or (2) based on its title and abstract, it was considered irrelevant and excluded from the rest of the systematic mapping process. For some papers it was unclear whether or not they satisfied the content criteria solely from their abstract and title. A text skimming was applied to such papers until the researchers could confidently assert that the paper was relevant or irrelevant.

There were many duplicates within some databases that were removed from the set of potentially relevant search results while applying title and abstract exclusion. Afterwards, 11 more duplicates cross-indexed between databases were removed.

To reduce the threat of missing relevant papers, we applied backwards snowballing [161] to a small subset of the relevant papers by looking through their references to identify potentially relevant papers not found by our initial search. The subset of papers that snowballing was applied to were selected as researchers evaluated papers in the title and abstract exclusion step. We found that many of the papers generated via backwards snowballing had already been identified as relevant papers in our initial search. Nonetheless, applying the study selection process described above to papers generated by backwards snowballing resulted in 7 more relevant studies. In all, 335 primary studies were identified in the study selection process.

2.4 Data Mapping

The final step of the systematic mapping study process involves mapping each of the relevant papers into categories based on well-defined classification schemes. The classification schemes are defined in detail along with how they were constructed in section 3. Each relevant paper was skimmed to the extent necessary for the researcher to categorize the paper according to each classification scheme.

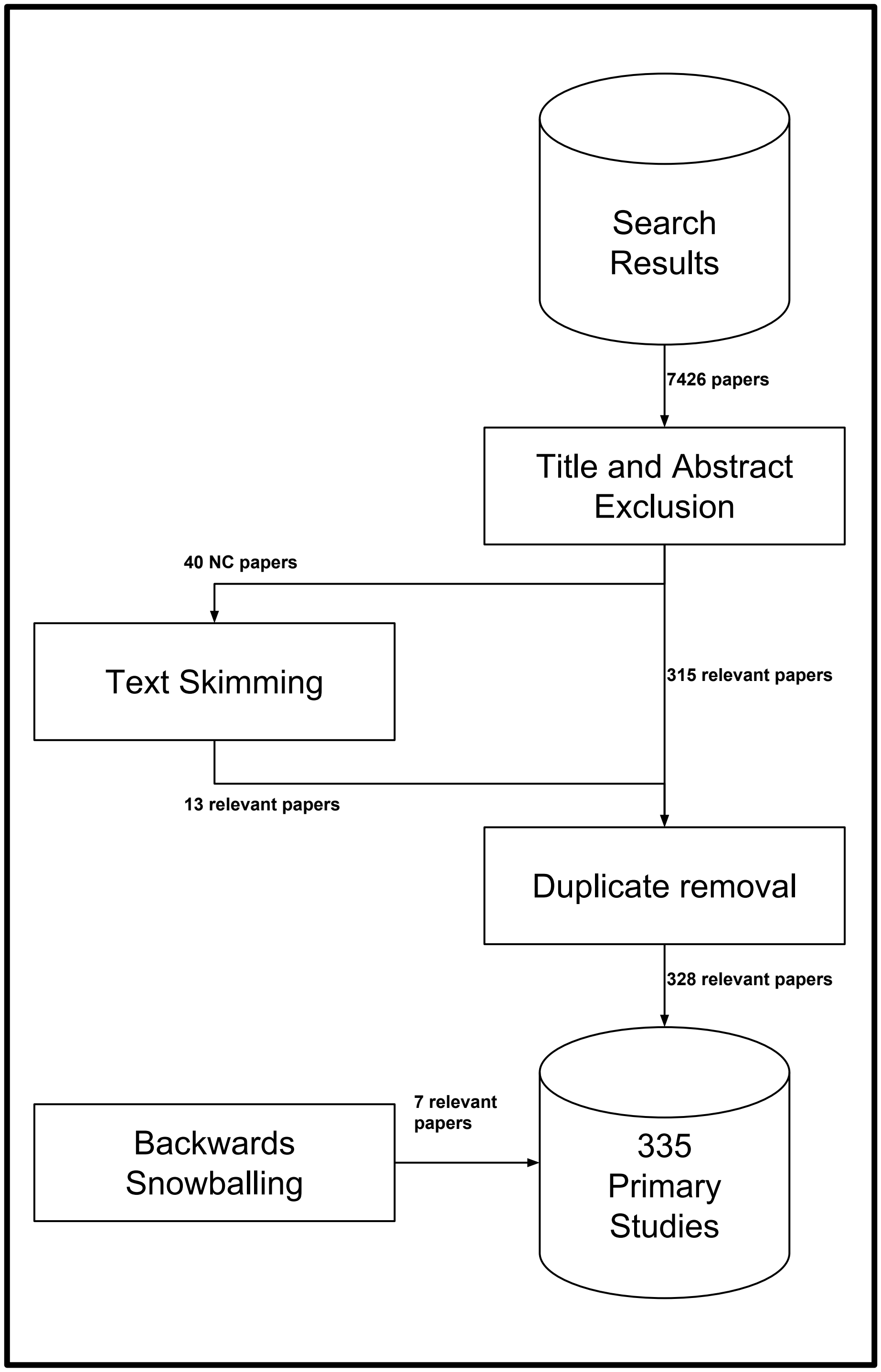

Figure 2.2: Overview of the study selection process including the number of papers resulting from each step.

3 Classification Schemes

In this section we provide the classification schemes used for the data mapping and discuss how they were constructed. The data facets that schemes were developed for were mostly derived from our research questions. For example, to answer research question 1.1, "What is the annual number of publications in the field?", papers were categorized based on the year in which they were published. Data facets were also derived with our goal of assisting researchers looking to evaluate a testing technique in mind.

The publication year, publication venue, and affiliation of the authors were extracted to answer research questions related to general publication trends. The *evaluation method*, *evaluation dimension*, *testing technique type*, *contribution type*, and *usage of mutation analysis* were extracted to answer more context-specific research questions. The classification schemes for these facets are discussed in more detail in the following subsections.

3.1 Evaluation Method

The evaluation method scheme categorizes papers based on the method they use for evaluating a software testing technique. Due to a lack of existing knowledge about the types of methods used, we systematically determined evaluation method categories using *Keywording* as suggested by [255]. This consisted of reading the abstracts of a subset of the collected relevant papers and generating keywords for the evaluation methods. After reading a fairly large number of abstracts, the generated keywords were clustered to form categories for methods of evaluating software testing techniques. In our case there were few unique keywords, most of which referred to fairly well-defined methods in research. Thus we relied on existing definitions to classify the four major categories we developed for this data facet:

1. *Experiment*: A paper was classified in the experiment category if it utilized an experiment to evaluate a software testing technique. This determination relied heavily on Wohlin's definition of experiments as an empirical investigation in which "different treatments are applied to or by different subjects, while keeping other variables constant, and measuring the effects on outcome variables" [322]. We considered quasi-experiments to be a type of experiment when making our determination.

2. *Case Study*: A paper was classified in the case study category if it utilized a case study to evaluate a software testing technique. A case study was considered to be "an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified" [270]. As opposed to an experiment, case studies exhibit much less control; usually due to their examination of the phenomenon in a much larger, real-world context.

3. *Example*: A paper was classified in the example category if it utilized an example to evaluate a software testing technique. We define an example as a demonstration of a single technique in a small and constructed context.

4. *Analytic*: A paper was classified in the analytic category if it utilized a direct evaluation of a technique based on its clear or provable properties.

Some papers utilized multiple methods for evaluating software testing techniques, so it was possible for a single paper to be placed in multiple categories. On the other hand, a small number of papers discussed guidelines or enhancements when evaluating techniques without actually utilizing an evaluation method. For example, a paper discussing experiment subject selection is a relevant paper since it provides insight on evaluating the effectiveness of a fault-detecting software testing technique, but it does not utilize a method for evaluating software testing techniques.

3.2 Evaluation Dimension

The evaluation dimension scheme categorizes papers based on the dimension for which they evaluate software testing techniques. Categories for this schema were derived directly from our research scope:

1. *Effectiveness*: A paper was classified in the effectiveness category if it evaluated the ability of a software testing technique to detect failures, kill mutants, or achieve some degree of coverage.

2. *Efficiency*: A paper was classified in the efficiency category if it evaluated the performance of a software testing technique in terms of speed, memory usage, or work done.

3. *Scalability*: A paper was classified in the scalability category if it evaluated how a technique performed in larger domains.

4. *Applicability*: A paper was classified in the applicability category if it evaluated the ability of the technique to be applied or generalized to other contexts.

As with the last classification scheme, it was possible for papers to be placed into multiple categories or to not fit any of the categories.

3.3 Testing Technique Type

This data facet refers to the type of testing technique a paper used in its evaluation. The categories for this scheme were directly derived from research question 2.4, which seeks to determine the distribution of white-box and black-box testing technique evaluations. Thus we categorized papers based on whether their evaluation was of a white-box or black-box testing technique:

1. *White-box:* At least one of the software testing techniques evaluated is a white-box testing technique. We classify a technique as a white-box technique using a definition from Amman and Offut [18], which states that a white-box technique derives "tests from the source code internals of the software, specifically including branches, individual conditions, and statements".

2. *Black-box:* At least one of the software testing techniques evaluated is a black-box testing technique. We again relied on a definition from Amman and Offut for determining whether or not a technique was black-box; considering a black-box technique as one that derived "tests from external descriptions of the software, including specifications, requirements, and design" [18]. Evaluations of gray-box testing techniques that did not require access to the source code of the software under test, but utilized partial knowledge of its internal structure were included in this category.

For this schema, papers could be classified as belonging to both categories if both a white-box and a black-box testing technique were evaluated. Papers were also classified as belonging to both categories if the technique type of the technique being evaluated was ambiguous and the technique was potentially applicable in both black-box and white-box contexts. Thus all papers utilizing a technique evaluation were classified as at least white-box or black-box.

3.4 Contribution Type

This scheme classifies papers based on the type of contribution they make in the field. We were particularly interested in the separation of papers utilizing methods as opposed to proposing new methods or guidelines for evaluating software testing techniques. Thus we defined the following categories:

1. *Guideline:* A paper was classified as a guideline paper if it provided guidelines for evaluating a software testing technique, proposed a method for evaluating software testing techniques, or proposed an enhancement for a method of evaluating a software testing technique. Thus papers primarily discussing mutation analysis methods or enhancements to them were considered proposal papers due to the ability of these methods to evaluate other testing techniques.

2. *Usage:* A paper was classified as a usage paper if it utilized some method for evaluating a software testing technique for its effectiveness, efficiency, scalability, or applicability.

Papers that met both criteria were classified in both categories. Due to our study selection criteria, every paper was classified in at least one of the contribution type categories.

3.5 Use of Mutation Analysis

Mutation analysis is a popular technique for evaluating the fault-detection capabilities of test suites. Unfortunately the technique is also computationally expensive; consisting of the generation of a usually large set of mutants and the execution of a large number of tests (potentially the entire suite) for each mutant in the set. This has led to the development of a wide range of cost reduction strategies for making mutation testing and analysis more feasible. Additionally, a wide range of mutation operators exist for different contexts and for seeding different types of faults. Which cost reduction technique should be used when evaluating a particular test suite? Which mutation operators should be used? For a mapping study of testing technique evaluations, identifying mutation analysis papers to assist researchers in answering such questions is an important goal. Thus the mutation analysis schema below categorizes papers based on whether or not they utilize mutation analysis to evaluate the effectiveness of software testing techniques:

1. *Mutation:* A paper was classified as a mutation paper if it utilized mutation analysis and evaluated the effectiveness, efficiency, scalability, or applicability of one or more software testing techniques.

2. *Not Mutation:* A paper was classified in this category if it evaluated the effectiveness, efficiency, scalability, or applicability of one or more software testing techniques and did not use mutation analysis.

As a result of this classification schema, all usage papers were categorized as either mutation or not mutation papers. Additionally no papers with only the guideline contribution type were included in this categorization since guideline-only papers did not evaluate the effectiveness, efficiency, scalability, or applicability of a testing technique.

3.6 Evaluation Quality

To answer RQ2.5, additional data was extracted from the two most common evaluation methods: case studies and experiments. For each of these methods, we relied on proper methodology guidelines to derive data facets that would help us assess the current state of evaluations in the field in terms of quality.

Guidelines for case study methodology in the field of software engineering are discussed by Runeson in [270]. Summarized from this work, some characteristics of an exemplary case study are the definition of research questions from a significant topic or theoretical basis, examination of multiple perspectives while investigating the topic, provision of a logical link between evidence and conclusions made, and a discussion of threats to the validity of the study. From these guidelines, the following categories were created for papers utilizing case studies to evaluate software testing techniques:

1. *Research Questions:* A paper was classified in this category if it clearly defined research questions to be addressed by the study.

2. *Triangulation:* This category assessed the case study's consideration of multiple perspectives. A paper utilizing a case study was classified as a triangulation paper if it collected data from multiple sources or used multiple types of data collection.

3. *Threats to Validity:* A paper was classified in this category if it seriously discussed threats to the validity of the study. A discussion was considered "serious" if it presented multiple threats and was at least a paragraph in length.

It should be noted that an evaluation framework for *empirical methods* in software testing was recently developed by [312]. This framework is much more detailed and focused, but due to its newness in the field it was not feasible to derive categories from it for this mapping study.

Guidelines for controlled experiment methodology in the field of software engineering are used to similarly develop categories for experiment papers. We rely on [322] for these guidelines. Some important characteristics of exemplary experiments include a clearly stated hypothesis with hypothesis testing, some justification for object/subject selection, descriptive statistics, and a discussion of threats to the validity of the experiment. From these guidelines, the following categories were created for papers utilizing controlled experiments to evaluate software testing techniques:

1. *Hypothesis Testing:* A paper was classified in the *Hypothesis Testing* category if it clearly stated a hypothesis and performed hypothesis testing to accept or reject this hypothesis.

2. *Descriptive Statistics:* A paper was classified in the *Descriptive Statistics* category if it utilized descriptive statistics when quantitatively analyzing results.

3. *Context Justification:* This category assessed the appropriateness of objects and subjects selected in controlled experiments. To meet the *Context Justification* criteria, a paper's objects or subjects needed to be fairly representative of the research question context, a common benchmark, or at least justified to a degree by some discussion in the paper. Thus papers presenting objects/subjects without justification for their selection or a clear connection to research goals were not included in this category.

4. *Threats to Validity:* A paper was classified in this category if it seriously discussed threats to the validity of the experiment. A discussion was considered "serious" if it presented multiple threats and was at least a paragraph in length.

4 Evaluating Software Testing Techniques: A Map of the Field

We present a map of the field of research evaluating software testing techniques. 335 relevant papers were systematically collected and mapped according to the classification schemes defined above; providing a large-scale overview of publication trends, research gaps, and the state of the art when it comes to evaluating software testing techniques.

4.1 Publication Trends

We begin by presenting the distribution of publications based on the extracted general publication information (publication year, publication venue, and author affiliation).

4.1.1 Annual Activity Level

Figure 4.1 illustrates the level of activity in the field over the last 11 years. The annual number of relevant papers increased significantly from 2009-2011 before fluctuating over the 7 remaining years of the mapping. As shown by the black line of best fit, the annual number of published papers in the field has grown a good amount overall. This suggests an increased interest in research evaluating software testing techniques.

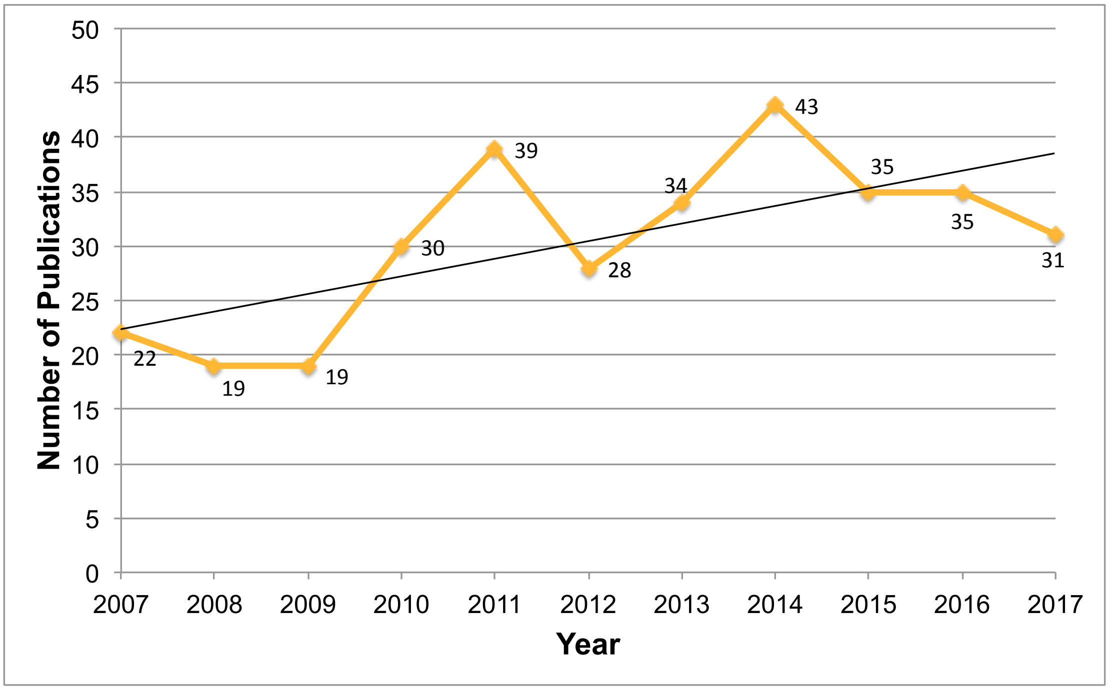

Figure 4.1: Annual number of publications.

4.1.2 Main Publication Venues

Not surprisingly given our fairly broad research scope, the relevant papers collected spanned 120 unique publication venues. While many of these venues only published one relevant paper, there were some venues responsible for publishing a significant number of contributions in the field. Table 4.1 lists the venues that published the most relevant papers along with how many they published. By a significant margin, the journal of Software Testing, Verification and Reliability was the most active publication venue with 33 relevant papers published over the last 11 years. The International Symposium on Software Testing and Analysis was the next largest contributor with 24 relevant papers. Six other venues listed in Table 4.1 had 10-20 relevant publications. The remaining venues had less than 10 relevant publications, with 79 venues having only 1 relevant publication.

Publication Venue	#	%
Software Testing, Verification and Reliability	33	9.85
International Symposium on Software Testing and Analysis	24	7.16
International Conference on Automated Software Engineering	17	5.07
International Conference on Software Engineering	15	4.48
International Conference on Software Testing	15	4.48
International Symposium on Foundations of Software Engineering	14	4.18
Empirical Software Engineering	13	3.88
International Conference on Software Testing, Verification and Validation	13	3.88

Table 4.1: Main publication venues

4.1.3 Industry vs Academia

Figure 4.2 shows the relative contributions of industry and academia based on author affiliation. Similar to most fields of research, a large majority of contributions are made by academia. 291 papers (about 87%) had exclusively authors affiliated with academic institutions. 30 papers (about 9%) had both authors affiliated with academic institutions and authors affiliated with industry. Only 14 papers (about 4%) had exclusively authors affiliated with industry.

4.2 Context-Specific Mappings

Next we present the results of the mapping based on the remaining classification schemes: evaluation method, evaluation dimension, testing technique type, contribution type, use of mutation analysis, and evaluation quality.

4.2.1 Evaluation Method

We developed 4 major categories for methods of evaluation: experiments, case studies, examples, and analytic evaluations. Figure 4.3 shows the number of papers

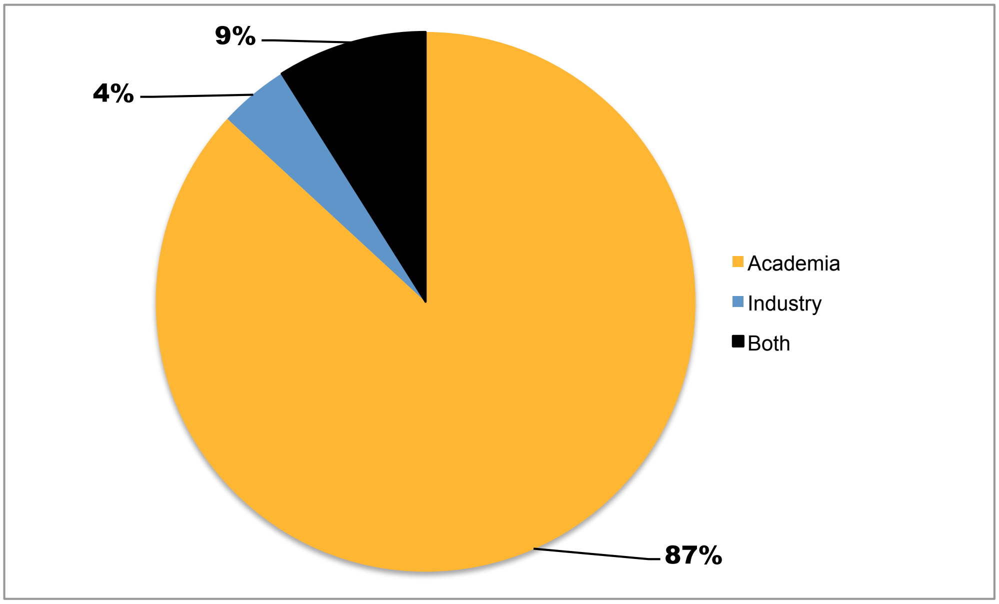

Figure 4.2: Percentage of contributions from industry and academia.

that utilized each evaluation method. Percentages shown are of the total number of evaluation instances as opposed to the total number of primary studies. As mentioned earlier, case studies and controlled experiments were by far the most common methods. Experiments in particular were utilized very frequently for evaluating software testing techniques. Of the 320 instances of testing technique evaluations, 214 of them (%66.88) were controlled experiments. 73 of them (%23.13) were case studies. Only 18 were analytic evaluations and only 15 fell into the example category. From this one data facet it seems that performing controlled experiments is the state of the art when it comes to evaluating software testing techniques. Exploring the relation between evaluation methods and other data facets provides more insight on how the state of the art changes with the dimension and type of testing technique evaluated.

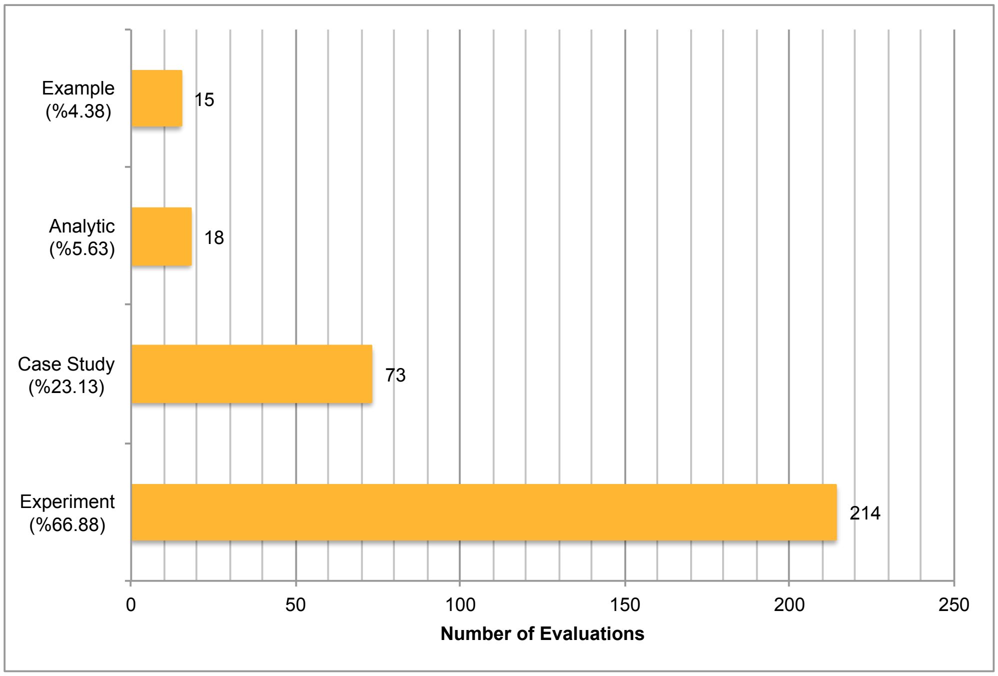

Figure 4.3: Distribution of primary study evaluations by method.

4.2.2 Evaluation Dimension

We also categorized papers based on the dimension they evaluated (effectiveness, efficiency, applicability, and scalability). Figure 4.4 shows the number of evaluations performed for each dimension. Percentages shown are of the total number of dimension evaluations. Note that there are more dimension counts than the number of relevant papers collected since some papers evaluated more than one dimension of a software testing technique.

Of the 425 total dimension evaluations, more than half of them (%55.06) evaluated effectiveness; suggesting that researchers are the most interested in evaluating techniques based on their ability to detect failures, kill mutants, or achieve some degree of coverage. This makes sense given the main purpose of testing techniques to reduce the risk of using software by detecting failures.

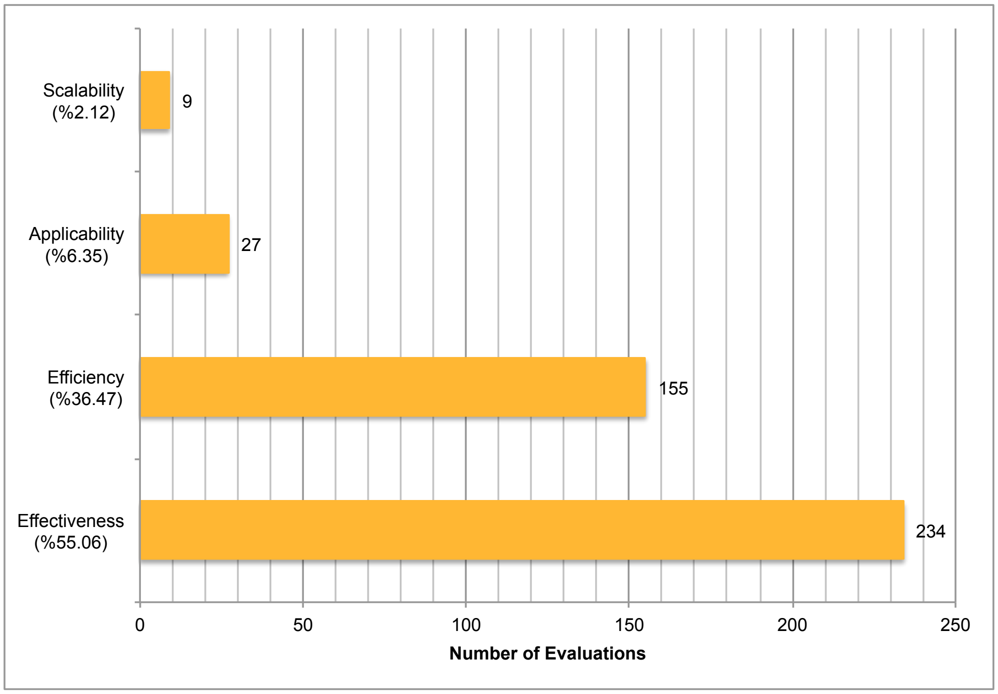

Figure 4.4: Number of evaluations by dimension.

Another large portion of the total dimension evaluations (%36.47) assessed the efficiency of a technique. The remaining %8.47 is split between applicability and scalability evaluations at %6.35 and %2.12 respectively.

4.2.3 Testing Technique Type

Figure 4.5 illustrates the distribution of testing technique types that were evaluated. We see that research evaluating software testing techniques is quite evenly split between white-box and black-box testing techniques. About (%46.71) of papers with evaluations are focused on white-box testing techniques, %49.01 are focused on black-box testing techniques, and the remaining %4.28 evaluated both of these testing technique types. While there are a good portion of papers dealing with the evaluation of black-box testing techniques, we found that a large chunk of these evaluations are of the same few techniques. Upon further investigation, about 30% of the black-box evaluations were of random testing or combinatorial testing techniques.

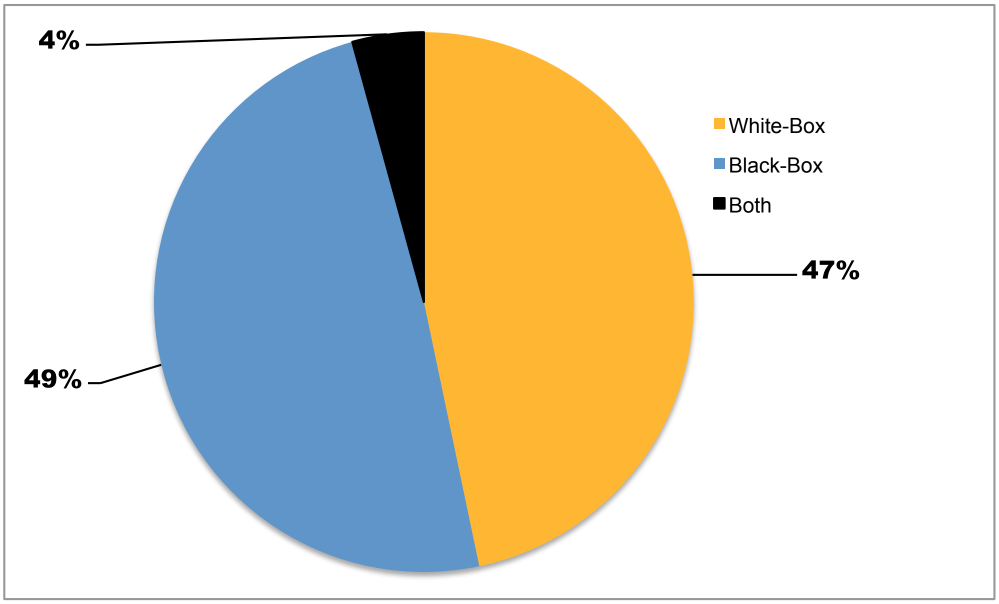

Figure 4.5: Percentage of white box and black box evaluations

4.2.4 Contribution Type

This scheme classified papers based on whether they evaluated a software testing technique or proposed some method or insights regarding how software testing techniques should be evaluated. As figure 4.6 illustrates, the majority of papers were usage papers that utilized some method for evaluating software testing techniques. On the other hand relatively very few papers discussed how techniques should be evaluated or proposed a new methodology for doing so.

Despite the lower number of proposal papers, we believe this type of contribution to the field is important for evolving and enhancing our ability to assess software testing techniques. As such, a secondary classification was performed on these papers to develop an understanding of the types of insights and proposals that exist for evaluating software testing techniques and to be able to point researchers towards higher level guidelines in areas they are interested in. Our hope is that bringing awareness to these papers will allow researchers to make higher quality evaluations of testing techniques as well as motivate more research of this contribution type. The following section describes the secondary classification and presents the results.

Guideline Paper Classification

Due to a lack of existing knowledge regarding the types of guideline papers we would find, *keywording* [255] was again used to develop categories for the types of guideline papers after examining each one in more detail. Doing so resulted in the following classification schema:

1. *Program Artifact:* Program artifact papers provide guidelines or insight for program artifacts under test when empirically evaluating software testing techniques. These include papers discussing the importance of considering fault types in effectiveness evaluations, advocating for common benchmark artifacts, and the state of the art in software fault injection.

2. *Evaluation Metric:* Evaluation metric papers provide guidelines or insight for choosing a metric when empirically evaluating software testing techniques. These include empirical correlations between evaluation metrics and fault-detecting ability, analytic effectiveness bounds, and proposals of novel criteria for evaluating test suite quality.

3. *Human Subject Selection:* A guideline paper was placed in this category if it provided insight with regards to the selection of human subjects for an empirical evaluation. Only one paper was placed in this category for exploring the impact of subject experience on study results.

4. *Methodology:* Methodology papers presented empirical study methodology guidelines not already addressed by the artifact or human subject selection categories. Examples of papers mapped to this category include the proposal of a unified framework and an outline of proper methodology when conducting empirical evaluations in software testing.

5. *Mutation Analysis (code):* Code mutation analysis papers presented an innovation or guideline to mutation analysis of test suites at the source code level. In some form they provided suggestions for how mutation analysis should be performed. Most of these papers discuss efficiency improvements as this is a well-known limitation of mutation analysis techniques. We separate mutation testing at the code level from mutation testing at the model level due to the large number of mutation analysis guideline papers and significant differences in guidelines between the two.

6. *Mutation Analysis (model):* Model mutation analysis papers presented an innovation or guideline to mutation analysis of test suites at the model level.

A full text skimming was applied to each of the proposal papers as they were categorized using the above schema. Table 4.2 presents the results of the secondary categorization; mapping each category to a set of proposal papers belonging to it. Furthermore, a short summary is provided with each of the proposal papers to make it easier for researchers to locate papers relevant to their interests.

Table 4.2: Guideline papers by category

Guideline Category	**Papers**
Program Artifact	[239] **Assessing Dependability with Software Fault Injection: A Survey** Presents an overview of the state of the art in software fault injection and insight on which approaches to apply in different contexts.

Table 4.2: (continued)

Guideline Category	Papers
	[76] **BegBunch Benchmarking for C Bug Detection Tools** Presents two benchmark programs in the C language with the hopes of providing a "common ground" for empirical comparisons of different fault-detecting techniques. [240] **On the improvement of a fault classification scheme with implications for white-box testing** Presents improvements for a fault classification scheme with the notion that testing techniques are better at finding certain types of faults than others. This paper is included in the artifact selection category since considering the nature of faults in artifacts used in empirical studies may enhance our understanding of the effectiveness of software testing techniques. [77] **On the number and nature of faults found by random testing** An evaluation of the nature of faults that are discovered by random testing. Also provides a fault classification scheme and evidence that the nature of faults should also be considered when comparing testing techniques.
Evaluation Metric	[71] **An Upper Bound on Software Testing Effectiveness** Provides an analytic upper bound on the effectiveness of software testing techniques that rely on failure patterns. [338] **Assertions Are Strongly Correlated with Test Suite Effectiveness** Empirically evaluates the relationship between the fault-detection ability of a test suite and its assertions. [126] **Comparing Non-adequate Test Suites using Coverage Criteria** An empirical evaluation in an attempt to answer which criteria should be used to evaluate test suites, particularly when test suites are non-adequate. [88] **Evaluating Test Suite Effectiveness and Assessing Student Code via Constraint Logic Programming** Suggests the evaluation of test suites by comparing their effectiveness with a suite automatically generated by Constraint Logic Programming

Table 4.2: (continued)

Guideline Category	Papers
	[328] **Information gain of black-box testing** Introduces a novel coverage criteria for assessing black-box tests based on information gain from test cases. [81] **On Use of Coverage Metrics in Assessing Effectiveness of Combinatorial Test Designs** Investigates the use of certain coverage metrics when evaluating combinatorial testing strategies. Due to somewhat variable coverage across contexts for a given strategy, suggests some measure of variability should be included when assessing the effectiveness of strategies using these metrics. [104] **PBCOV: a property-based coverage criterion** Proposes a new property-based criterion for assessing the adequacy of test suites. [184] **State Coverage: A Structural Test Adequacy Criterion for Behavior Checking** Proposes state coverage, a new structural criterion for assessing the adequacy of test suites. [293] **Structural testing criteria for message-passing parallel programs** Introduces a novel structural testing criteria specifically for message-passing parallel programs. Additionally presents a tool that implements the new criteria along with results from applying it. [122] **The Risks of Coverage-Directed Test Case Generation** An empirical evaluation of structural coverage criteria. Among other things, concludes that traditional structural coverage criteria by itself may be a poor indicator of a test suite's fault-detection capabilities and that Observable MC/DC may be a promising alternative. [297] **Selecting V&V Technology Combinations: How to Pick a Winner?** Proposes a systematic method for evaluating verification and validation technique combinations. [174] **Towards a deeper understanding of test coverage** Suggests coverage criteria should be calculated at different testing levels instead of for the test suite as a whole.

Table 4.2: (continued)

Guideline Category	Papers
	[143] **Web Application Fault ClassificationAn Exploratory Study** Introduces a web application fault classification schema based on the exploration of two large, real-world web systems.
Human Subject	[82] **The Impact of Students Skills and Experiences on Empirical Results: A Controlled Experiment with Undergraduate and Graduate Students** A controlled experiment investigating how the experience of human subjects in empirical studies evaluating effectiveness and efficiency can impact results.
Methodology	[48] **Towards a Semantic Knowledge Base on Threats to Validity and Control Actions in Controlled Experiments** Proposes a knowledge base of threats to validity to assist researchers in mitigating threats when planning experiments. [283] **The role of replications in Empirical Software Engineering** Identifies types of empirical study replications, discusses the purpose of each type, and gives guidelines for providing sufficient information about reported empirical studies to better enable study replication. [56] **A Critical Analysis of Empirical Research in Software Testing** Provides a critical analysis of empirical research in software testing and discusses common threats that arise when determining cost-effectiveness of a technique via empirical research. [64] **Towards Reporting Guidelines for Experimental Replications: A Proposal** Suggests publishing guidelines for experiment replications in order to "increase the value of experimental replications". [102] **Empirical Evaluation of Software Testing Techniques in an Open Source Fashion** Presents and advocates for a unified framework for testing technique evaluations to ease study replication and improve reproducibility of results.

Table 4.2: (continued)

Guideline Category	Papers
	[312] **A Methodological Framework for Evaluating Software Testing Techniques and Tools** Defines a general methodological evaluation framework for case studies in software testing.
Mutation Analysis (c)	[150] **A Generic Approach to Run Mutation Analysis** Introduces a generic approach for mutation analysis that is not restricted to particular execution environments. [144] **An approach for experimentally evaluating effectiveness and efficiency of coverage criteria for software testing:** Provides guidelines and a demonstration of how to evaluate the effectiveness and efficiency of coverage criteria utilizing mutation analysis. [169] **Do Redundant Mutants Affect the Effectiveness and Efficiency of Mutation Analysis?** Empirically demonstrates efficiency and effectiveness improvement gains from removing redundant mutants in mutation analysis. [127] **Efficient mutation testing of multithreaded code** "Introduces a general framework for efficient exploration that can reduce the time for mutation testing of multithreaded code" [286] **Extended Firm Mutation Testing: A Cost Reduction Technique for Mutation Testing** Discussion of various mutation cost reduction techniques and a proposal for a new execution based cost reduction technique. [334] **Faster Mutation Testing Inspired by Test Prioritization and Reduction** Proposes a mutation testing cost reduction technique that prioritizes tests to more quickly determine which mutants were killed. [138] **Measuring Effectiveness of Mutant Sets** Empirical investigation and guidelines regarding how mutant sets should be evaluated. [310] **Mutants Generation For Testing Lustre Programs** Presents a mutation generator for Lustre programs that employs mutation cost reduction techniques.

Table 4.2: (continued)

Guideline Category	Papers
	[199] **Mutation Testing in Practice using Ruby** Presents mutation operators for Ruby and guidelines for mutation testing based on experience from an industrial Ruby project.
	[247] **Mutation Testing Strategies using Mutant Classification** Proposes mutant classification strategies to assist in isolating equivalent mutants along with an experimental evaluation of the technique.
	[146] **Mutation Testing Techniques: A Comparative Study** An empirical comparison of four mutation testing techniques (operators at class level, operators at method level, all operators, and random sampling)
	[170] **The Major Mutation Framework: Efficient and Scalable Mutation Analysis for Java** Introduces a JUnit mutation analysis and fault seeding framework with claims of scalability and efficiency.
	[237] **The Use of Mutation in Testing Experiments and its Sensitivity to External Threats** Brings to light important external threats to consider when utilizing mutation testing in experiments. These threats may be caused by test suite size, selected mutation operators, and programming languages.
	[83] **Using Evolutionary Computation to Improve Mutation Testing** Introduces a mutation testing cost reduction technique that utilizes a genetic algorithm to produce a reduced set of mutants.
	[246] **An Empirical Evaluation of the First and Second Order Mutation Testing Strategies** Provides an evaluation of the cost and effectiveness of different mutation testing strategies.
	[257] **Decreasing the cost of mutation testing with second-order mutants** Proposes a cost reduction technique for mutation testing/analysis that combines mutants from an original set to obtain a new set of mutants. Additionally performs an empirical evaluation of a test suite created from these combined mutants.

Table 4.2: (continued)

Guideline Category	Papers
	[230] **Efficient JavaScript Mutation Testing** Proposes mutation operators specific to web applications and a mutation cost reduction technique.
	[171] **Efficient Mutation Analysis by Propagating and Partitioning Infected Execution States** Significant efficiency gains in mutation analysis using state infection conditions. The approach is also implemented and empirically evaluated on open source programs.
	[182] **Evaluating Mutation Testing Alternatives: A Collateral Experiment** Proposes second order mutation strategies and provides experimental results suggesting the strategies lead to significant cost reductions without considerably reducing test effectiveness.
	[66] **Exploring hybrid approach for mutant reduction in software testing** Introduces a hybrid mutation testing cost reduction technique.
	[342] **JDAMA: Java database application mutation analyser** Introduces a mutation analyzer useful for evaluating testing techniques applied to java database applications.
	[147] **Mutation Operators for Simulink Models** Proposes a set of mutation operators for Simulink models and provides a procedure for mutation testing of Simulink Models.
	[178] **Mutation Operators for the Atlas Transformation Language** Presents mutation operators for the Atlas Transformation Language and evaluates their effectiveness in an empirical study.
	[225] **Parallel mutation testing** Suggests enhancing the efficiency of mutation testing by utilizing parallel execution.
	[226] **Reducing mutation costs through uncovered mutants** Presents a mutation cost reduction technique that leverages the analysis of covered mutants to reduce the number of executions required.

Table 4.2: (continued)

Guideline Category	Papers
	[128] **Selective Mutation Testing for Concurrent Code** "Explores selective mutation techniques for concurrent mutation operators" and provides an empirical study evaluating these techniques.
	[344] **Speeding-Up Mutation Testing via Data Compression and State Infection** Speeds up mutation testing by filtering out executions using state infection information and grouping mutants with Formal Concept Analysis.
	[212] **Statistical Investigation on Class Mutation Operators** Provides statistical information regarding the number of mutants generated, the distribution of mutants generated, and the effectiveness of applying class mutation operators to 866 open source classes.
	[139] **Topsy-Turvy: A Smarter and Faster Parallelization of Mutation Analysis** Presents a new parallelization technique for mutation analysis.
	[172] **Using Conditional Mutation to Increase the Efficiency of Mutation Analysis** Introduces a new efficiency optimization when performing mutation analysis called conditional mutation.
	[211] **X-MuT: A Tool for the Generation of XSLT Mutants** Introduces mutation operators for the XLST language along with their implementation in a tool and an evaluation of its effectiveness.
Mutation Analysis (m)	[86] **A Variability Perspective of Mutation Analysis** Introduces method for modeling mutation operators as a feature diagram for better and faster mutation analysis.
	[87] **Featured Model-based Mutation Analysis** Proposes an optimization for model-based mutation analysis using a modeling framework. Performance evaluations of the proposed technique are carried out and compared to other optimizations.

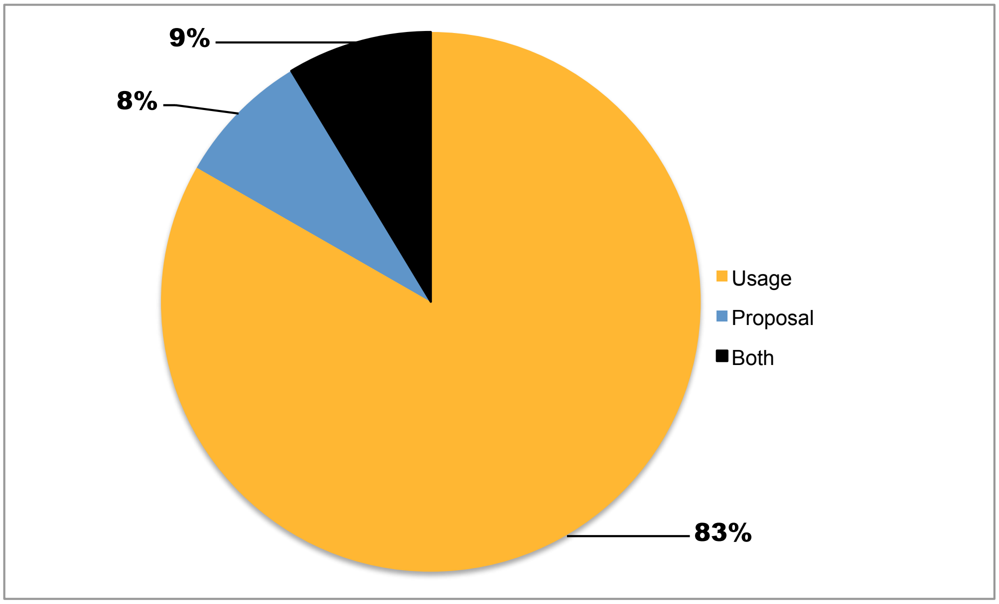

Figure 4.6: Contribution type distribution

4.2.5 Use of Mutation Analysis

Figure 4.7 illustrates the portion of evaluation papers classified as mutation papers. Of all 217 papers evaluating the effectiveness of a testing technique using a case study or experiment, a large portion of them (28%) utilized mutation analysis. Furthermore, mutation analysis seems to be becoming more popular over time. Figure 4.8 shows the proportion of effectiveness evaluations that utilize mutation analysis each year. One of the main limitations of mutation testing and analysis has been its high computational cost. It makes sense that mutation analysis has become more popular as more cost reduction strategies are developed and refined.

4.2.6 Evaluation Quality

Tables 4.3 and 4.4 present the results of extracting evaluation quality data facets from experiments and case studies respectively.

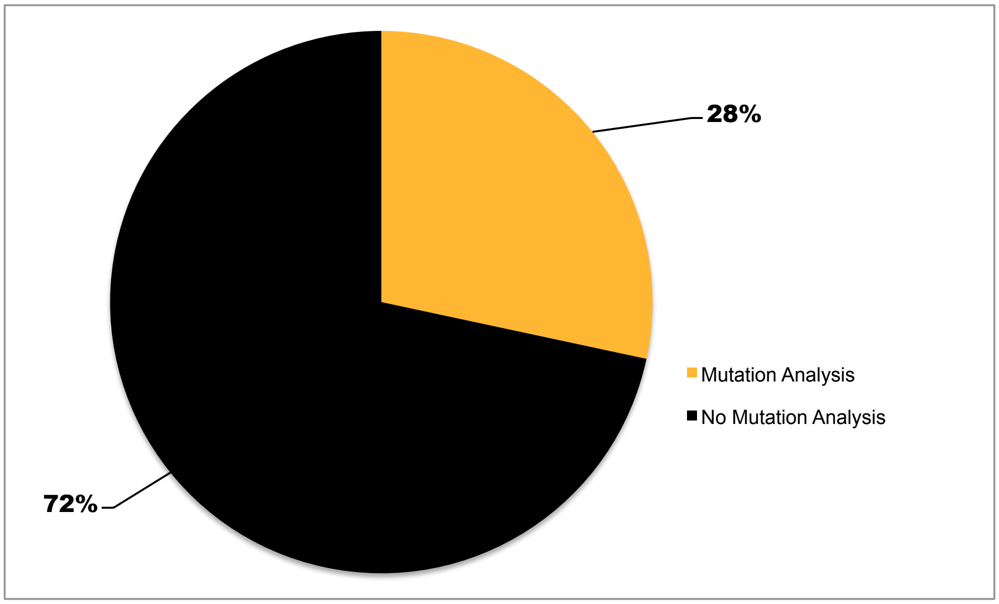

Figure 4.7: Mutation Analysis Distribution

Experiment Evaluation Quality		
Category	**# of Experiments**	**%**
Hypothesis Testing	39	18.22
Context Justification	98	45.79
Descriptive Statistics	160	74.77
Threats to Validity	100	46.73

Table 4.3: The number and percent of experiments that satisfy each of the experiment evaluation quality criteria

Very few experiments (%18) formally stated a hypothesis and performed hypothesis testing. On the other hand, a majority of experiments utilized descriptive statistics. We see that close to half of experiments meet the justified context criteria and provide a serious discussion of threats to validity. A smaller percentage of case studies provided threats to validity. 57% of case studies implemented some form of data triangulation while few (27%) clearly stated research objectives.

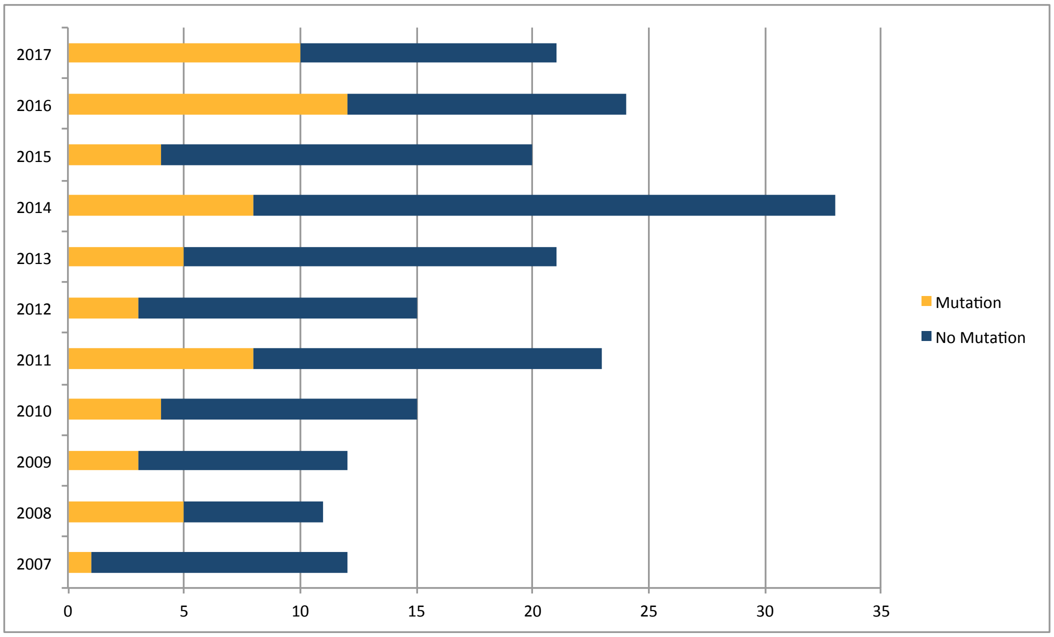

Figure 4.8: Distribution of mutation analysis over time

Case Study Evaluation Quality		
Category	**# of Case Studies**	**%**
Research Questions	20	27.40
Triangulation	42	57.53
Threats to Validity	27	36.99

Table 4.4: Number and percent of case studies that satisfy each of the case study evaluation quality criteria

4.2.7 Distribution of Evaluation Methods Over Time

Figure 4.9 shows the distribution of evaluation methods over time. Experiments were the most common method of evaluating testing techniques every year. The number of case study and experiment evaluations grew considerably from 2007 to 2014; growing by %366.67 and %236.36 respectively. The number of experiments and case studies remained fairly high in the last 3 years of the study. Both the number of examples and analytic evaluations remained low throughout the study with minor variation.

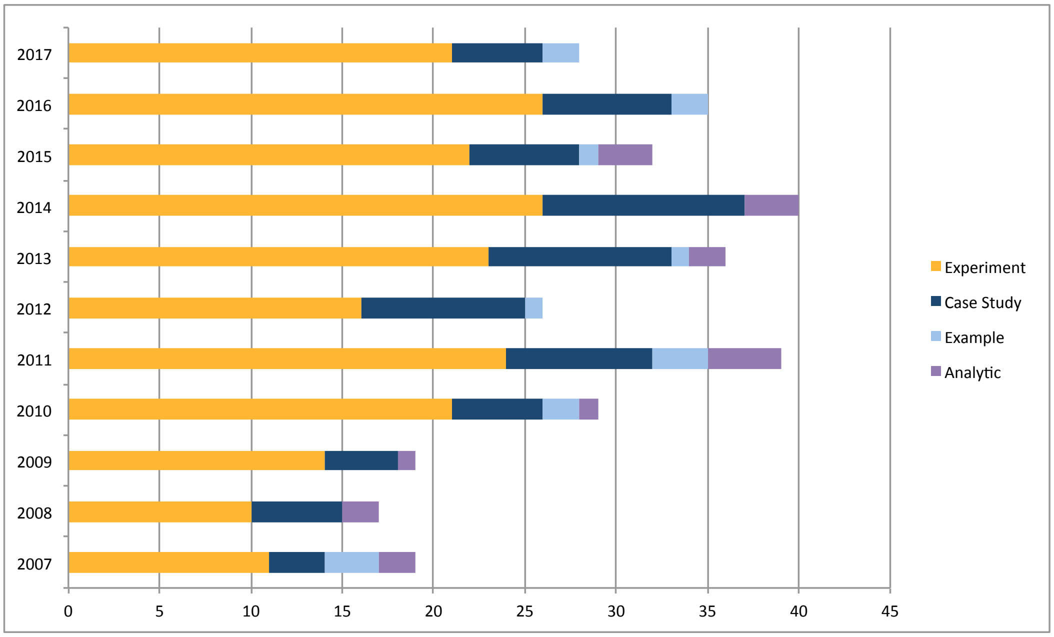

Figure 4.9: Distribution of evaluation methods over time

4.2.8 Relation of Evaluation Method and Dimension

Table 4.5 gives the number of relevant papers by evaluation method and evaluation dimension. Figure 4.10 illustrates their distribution. Note that the total number of papers is greater than 335 since a paper could utilize multiple evaluation methods or evaluate multiple dimensions. Given that experiments were the most common evaluation method and effectiveness was the most common evaluation dimension, it is not surprising that experiments evaluating the effectiveness of a technique are the most common here. Experiments evaluating the effectiveness and efficiency of testing techniques make up over half of the total testing technique evaluations. We see that relatively very few experiments evaluated the scalability or applicability of testing techniques. A large number of case studies also evaluate the effectiveness and efficiency of software testing techniques. Despite the much lower number of applicability evaluations in general (%6.5 of all evaluations), %13.46 of case studies evaluated applicability. Furthermore, %50 of

	Experiment	Case Study	Example	Analytic
Effectiveness	161	57	8	8
Efficiency	123	30	0	7
Scalability	5	3	0	1
Applicability	2	14	8	4

Table 4.5: Distribution of papers by evaluation method and evaluation dimension.

applicability evaluations were case studies compared to %7.14 that were experiments. Very few scalability evaluations are performed in general, but case studies and experiments make up %88.89 of them. Examples were evenly used to assess the effectiveness and applicability of techniques. No examples were used to investigate efficiency or scalability. Examples also make up a large amount of applicability evaluations (28.57%). We see that analytic evaluations assessed effectiveness and efficiency the most, but only assess scalability once.

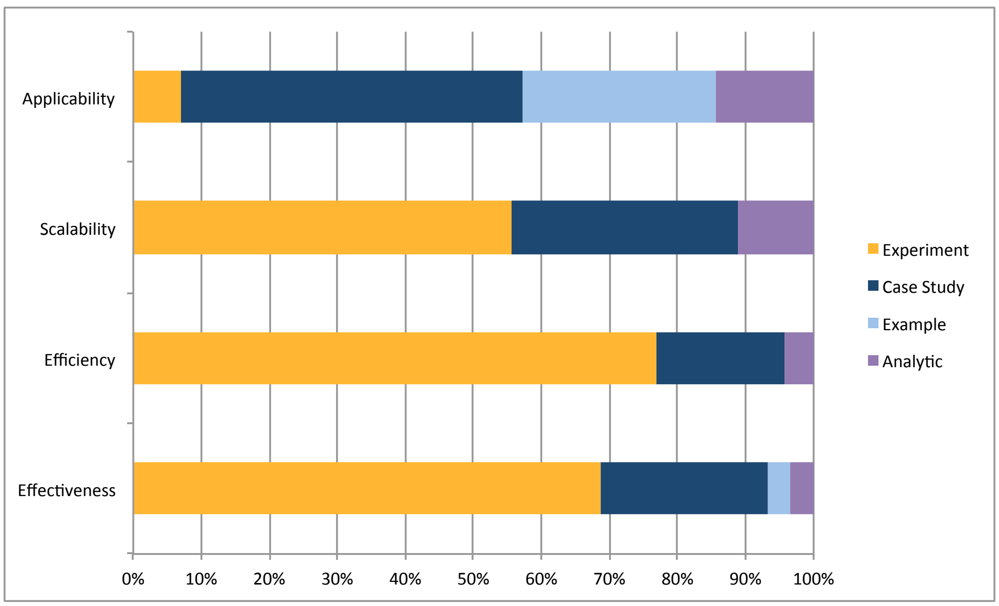

Figure 4.10: Distribution of evaluations by method and dimension

4.2.9 Relation of Mutation Analysis, Evaluation Method, and Technique Type

Figures 4.11 and 4.12 show the distribution of effectiveness papers utilizing mutation analysis in experiments and case studies. The distribution is surprisingly similar for experiments and case studies, differing only by about one percent of papers.

A somewhat greater difference can be observed when comparing the distributions of mutation analysis papers by testing technique type. Figures 4.13 and 4.14 illustrate this difference. (33%) of black-box effectiveness evaluations utilized mutation analysis. On the other hand, mutation analysis was surprisingly a bit less popular in white-box effectiveness evaluations; being used in about (25%) of these papers.

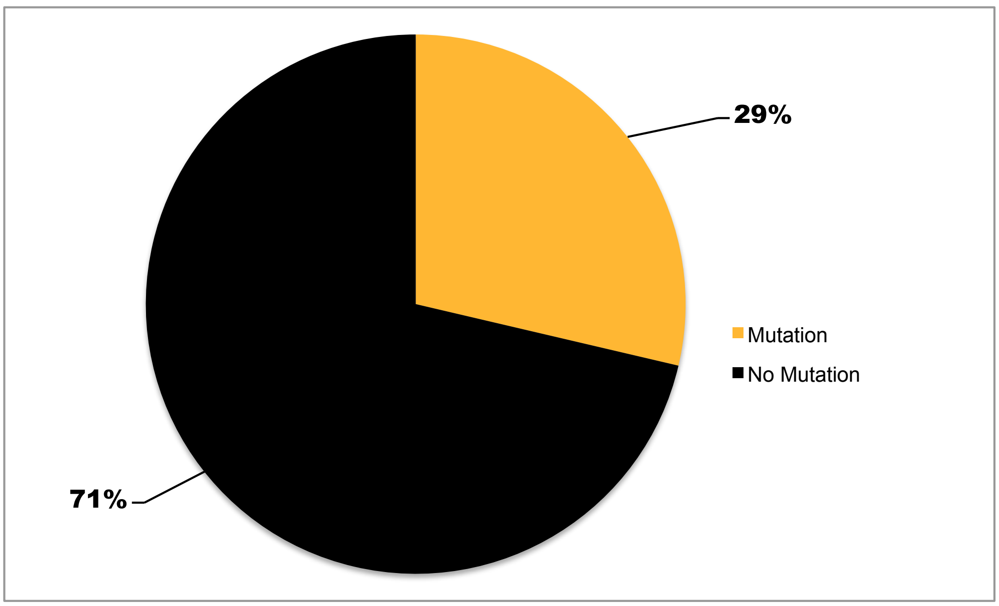

Figure 4.11: Distribution of mutation analysis experiment papers

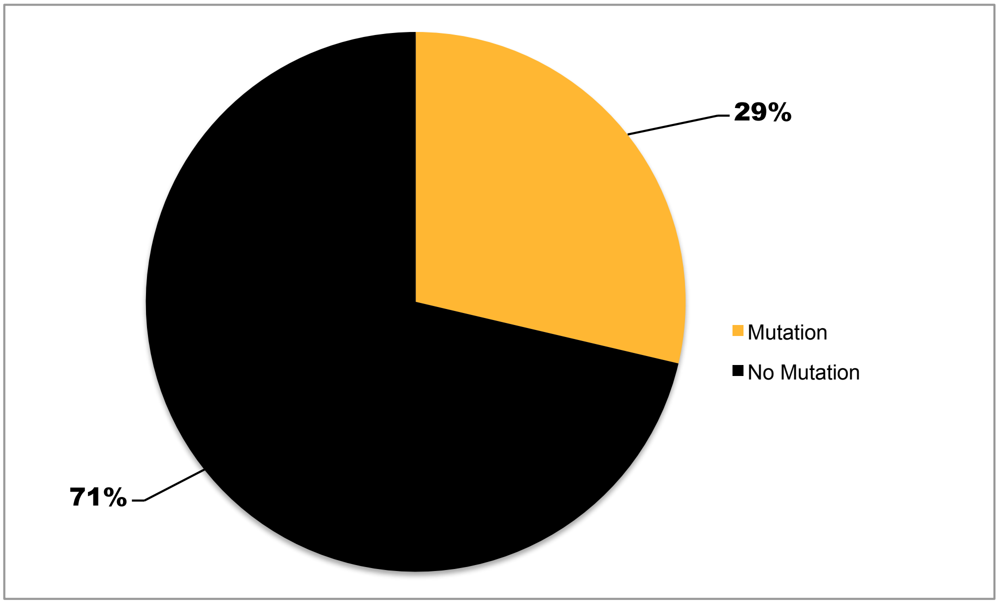

Figure 4.12: Distribution of mutation analysis case study papers

4.2.10 Relation of Author Affiliation, Evaluation Method, and Evaluation Dimension

Figure 4.15 shows the relation between author affiliation, evaluation method, and dimension of evaluation. We see that industry has the most involvement with experiments assessing effectiveness and efficiency and with case studies assessing effectiveness, efficiency, and applicability. Industry has little affiliation with other evaluation methods or dimensions of evaluation.

4.2.11 Relation of Technique Type, Evaluation Method, and Evaluation Dimension

Figure 4.16 shows the relation between technique type, evaluation method, and evaluation dimension. We see that for most combinations of technique type and evaluation dimension, experiments are the most common method of evaluation followed by case studies. Of notable exception are applicability evaluations of both white box and

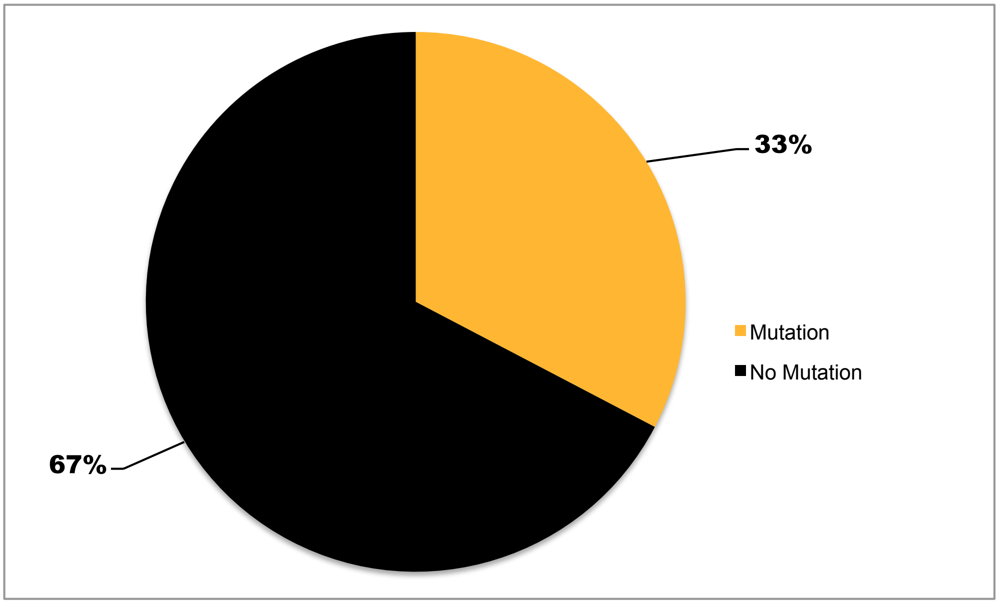

Figure 4.13: Distribution of mutation analysis black-box papers

black box testing techniques. In these applicability evaluations, case studies become the most common evaluation method, making up %52.63 of black-box evaluations and %44.44 of all white-box evaluations. %69 of all case studies evaluating applicability were evaluations of black box testing techniques.

More interesting are the differences between some of the evaluation method distributions with the same evaluation dimension. For instance, white-box scalability evaluations found in this study exclusively use experiments while about %50 of black-box scalability evaluations consist of case studies and analytic evaluations. Analytic evaluations also made up a greater amount of white-box applicability evaluations than they did black-box applicability evaluations. We find that across the board case study evaluations are a good amount more common when evaluating black-box testing techniques.

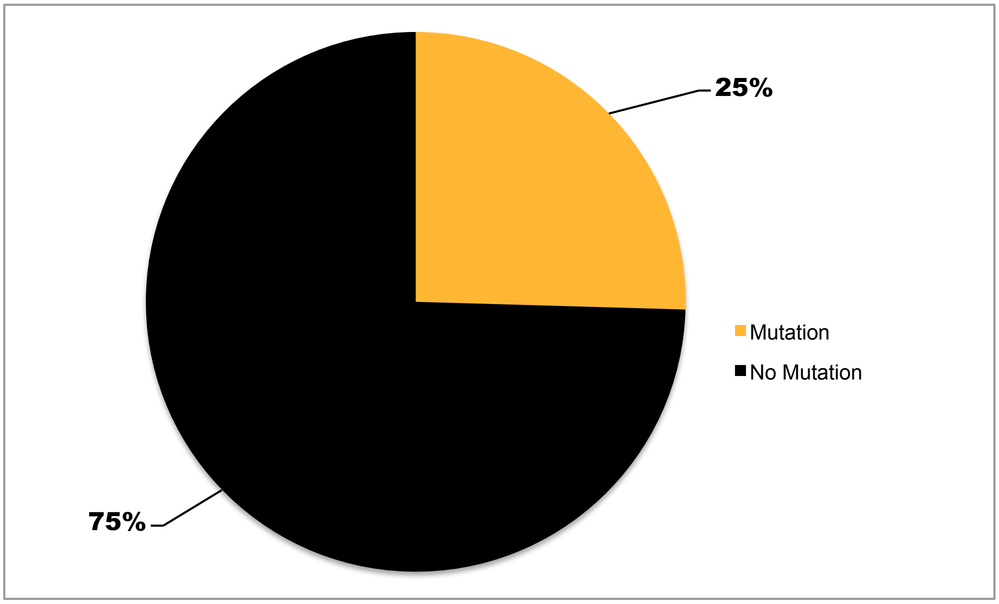

Figure 4.14: Distribution of mutation analysis white-box papers

4.3 Papers By Category

Probably the largest contribution of this thesis is a map from our classifications to sets of specific papers belonging to them. We hope such a map will allow researchers to easily locate papers evaluating software testing techniques with certain characteristics. In particular, researchers looking to evaluate a particular technique can develop an understanding of how they should do so by utilizing the map to find the state of the art for similar technique evaluations.

Each combination of technique type, evaluation dimension, evaluation method, and mutation affiliation is mapped to a set of papers along with the set's cardinality in Table 4.6. Due to the large number of papers, each paper is presented using its citation number. Due to the large number of category combinations (64), the table utilizes a unique context identifier as a key assigned to each subset of evaluation method combinations. A complementary decision tree (Figure 4.17) is provided for quickly obtaining a context identifier based on paper characteristics, and thus for quickly finding a table entry of interest since context identifiers are sorted alphabetically. The internal nodes of the tree represent classification schemes, with branches to children representing each classification in the scheme. The leaves of the tree are the context identifiers for entries in Table 4.6. Thus context identifiers are obtained from the tree by following a path from its root to a leaf based on classification categories of interest. A more in depth demonstration utilizing the tree and table is presented in a case study in Section 6.

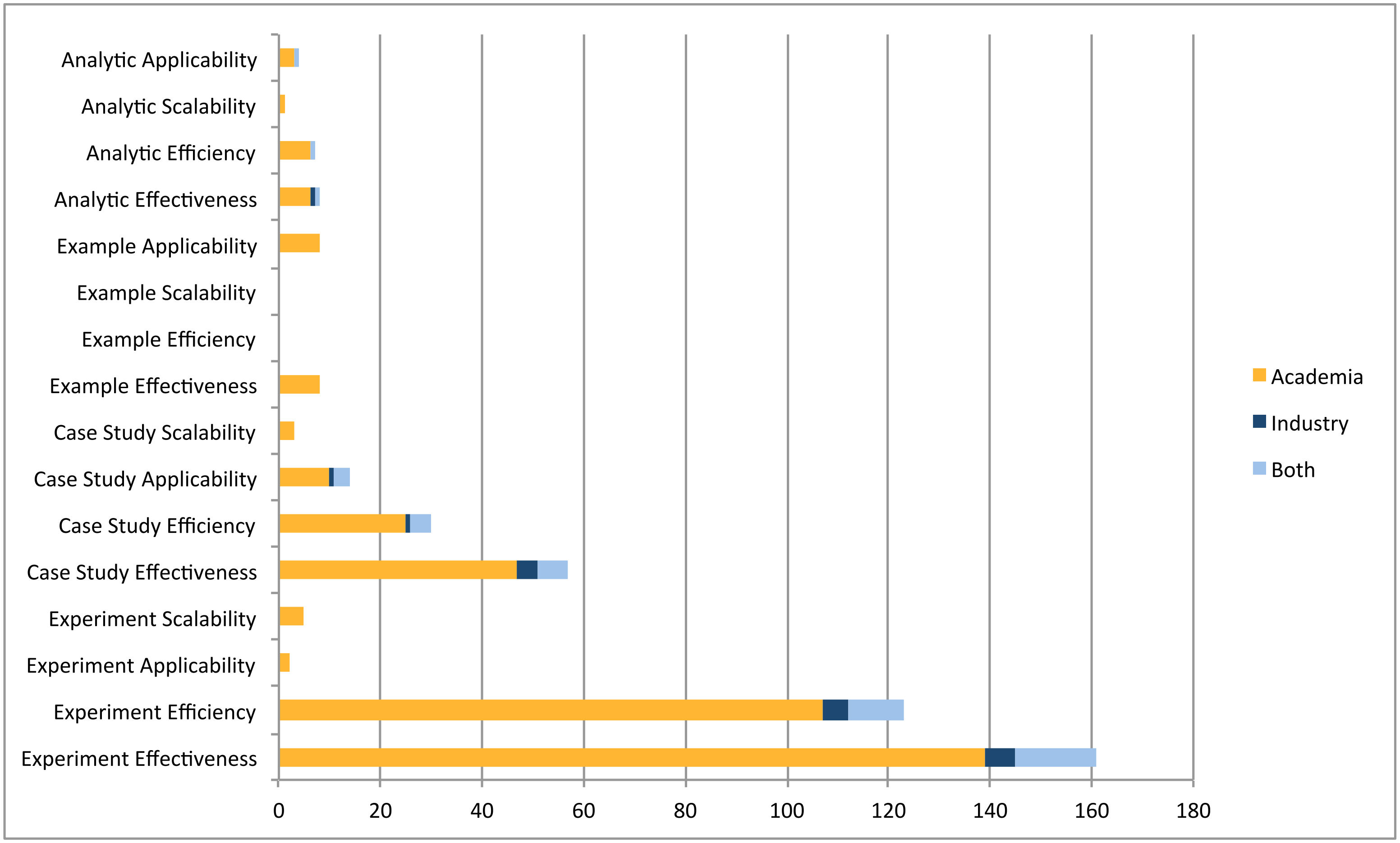

Figure 4.15: Distribution of evaluations by author affiliation, method, and dimension.

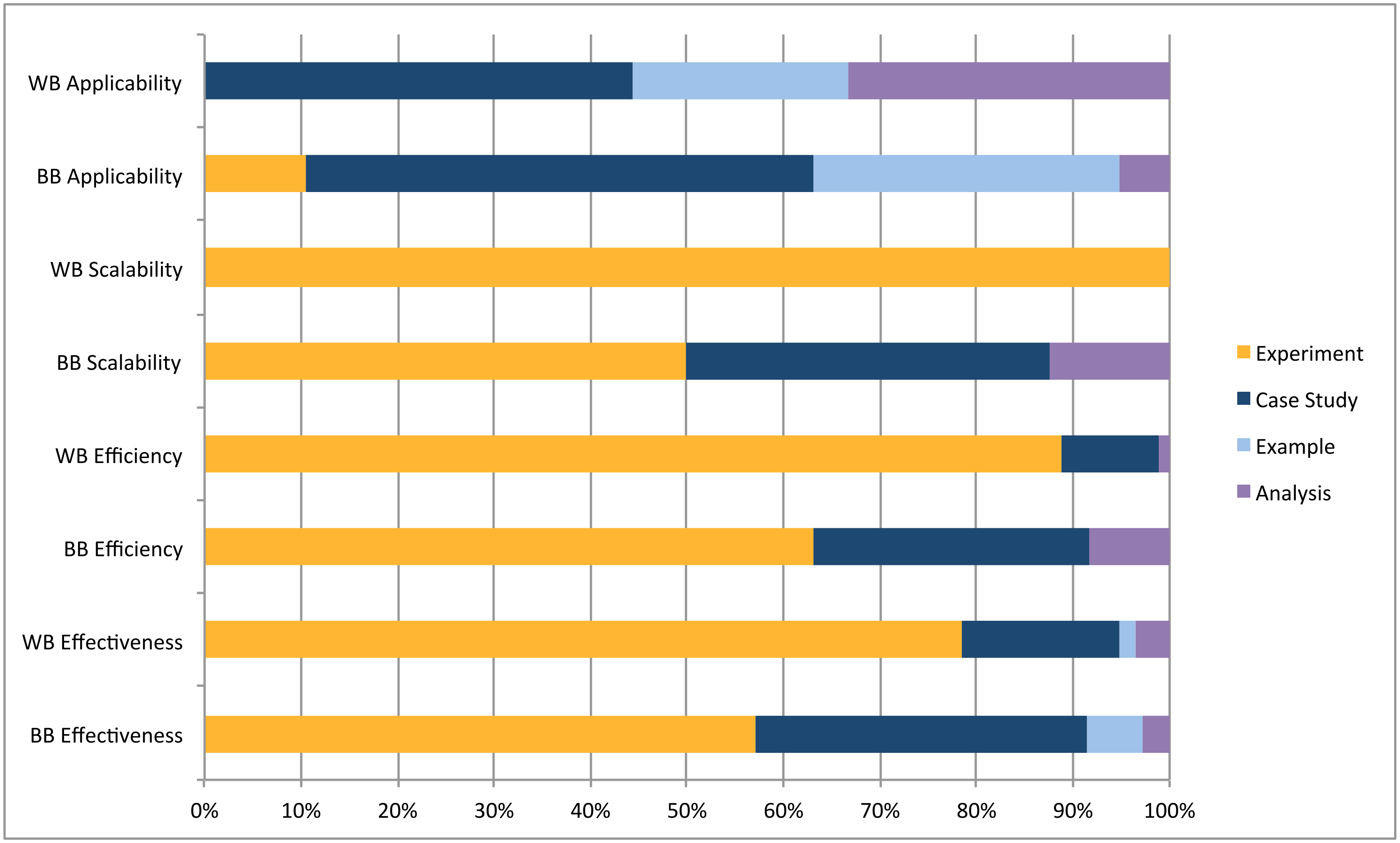

Figure 4.16: Relation of technique type, evaluation method, and evaluation dimension.

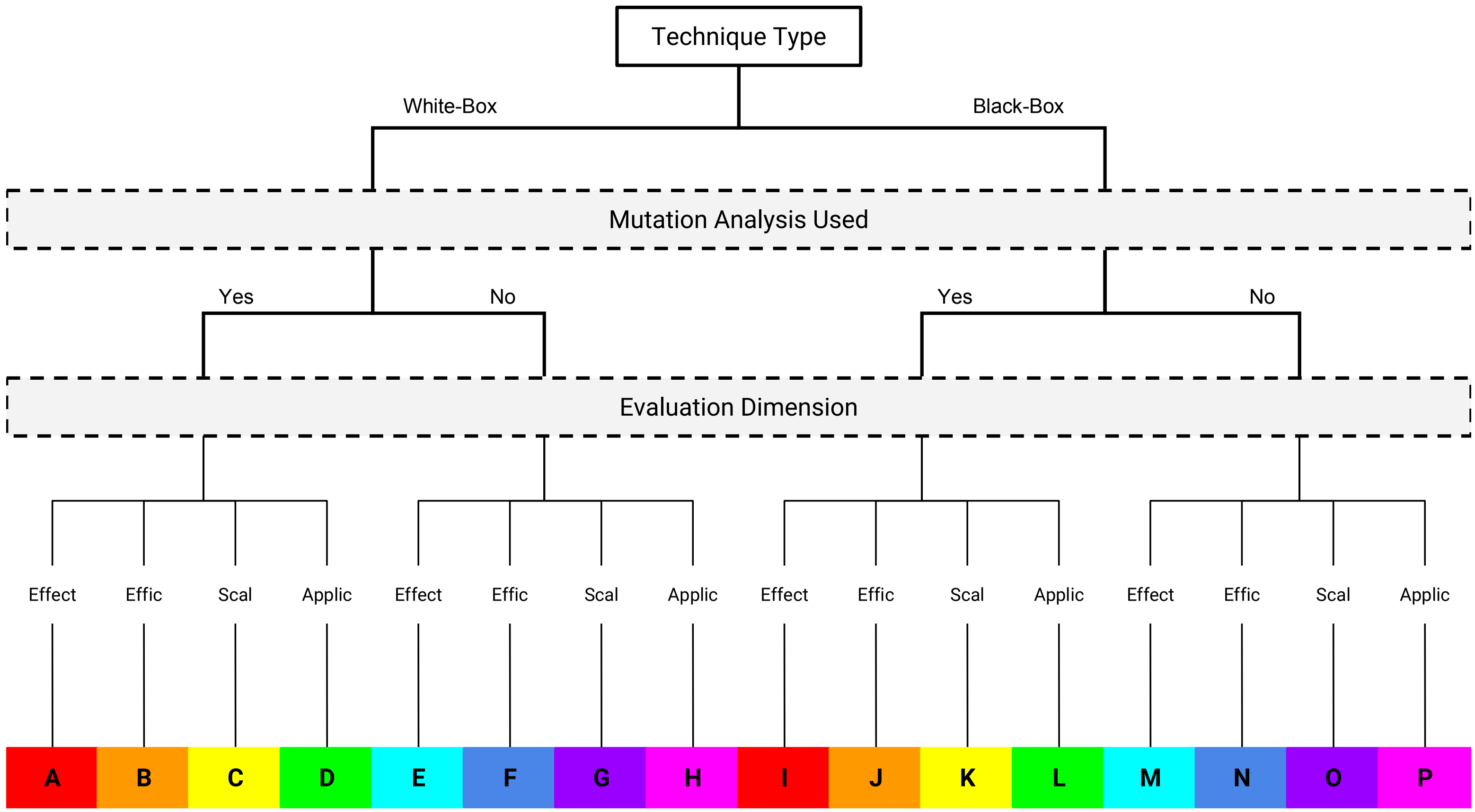

Figure 4.17: Decision tree for quickly locating entries in Table 4.6

Table 4.6: Papers belonging to each category combination

Context ID	**Evaluation Method**	**Count**	**Papers**
A	Experiment	30	[80], [269], [292], [193], [182], [146], [72], [254], [116], [311], [258], [290], [128], [249], [334], [200], [273], [340], [317], [36], [144], [248], [162], [100], [190], [63], [109], [115], [234], [309]
	Case Study	6	[230], [199], [213], [330], [256], [247]
	Analytic	0	
	Example	0	
B	Experiment	24	[80], [92], [225], [292], [212], [344], [182], [66], [146], [254], [171], [170], [311], [258], [290], [172], [334], [273], [36], [144], [248], [181], [63], [234]
	Case Study	4	[230], [139], [330], [100]
	Analytic	0	
	Example	0	
C	Experiment	0	
	Case Study	2	[199], [330]
	Analytic	1	[342]
	Example	0	
D	Experiment	0	
	Case Study	0	
	Analytic	0	
	Example	0	

Table 4.6: (continued)

Context ID	**Evaluation Method**	**Count**	**Papers**
E	Experiment	71	[101], [25], [214], [316], [7], [132], [152], [257], [90], [252], [131], [24], [142], [42], [19], [108], [319], [167], [259], [130], [166], [84], [121], [336], [217], [285], [4], [76], [135], [231], [346], [16], [145], [321], [9], [245], [300], [340], [298], [68], [31], [232], [113], [20], [103], [198], [141], [159], [218], [148], [160], [3], [53], [21], [67], [215], [136], [133], [111], [95], [155], [165], [154], [120], [47], [46], [343], [110], [246], [194], [12]
	Case Study	15	[180], [202], [11], [30], [88], [335], [223], [124], [278], [112], [52], [265], [253], [308], [41]
	Analytic	2	[304], [125]
	Example	5	[25], [289], [60], [23], [134]
F	Experiment	54	[127], [226], [214], [316], [106], [257], [90], [252], [345], [319], [259], [105], [130], [166], [186], [34], [84], [304], [171], [170], [280], [76], [135], [231], [266], [78], [296], [16], [321], [203], [298], [68], [31], [232], [250], [189], [20], [284], [160], [14], [67], [136], [307], [13], [185], [111], [188], [155], [165], [47], [271], [110], [246], [12]
	Case Study	6	[202], [139], [124], [253], [35], [41]
	Analytic	0	

Table 4.6: (continued)

Context ID	**Evaluation Method**	**Count**	**Papers**
	Example	1	[149]
G	Experiment	0	
	Case Study	2	[52], [268]
	Analytic	1	[73]
	Example	3	[173], [125], [117]
H	Experiment	1	[141]
	Case Study	0	
	Analytic	0	
	Example	0	
I	Experiment	22	[207], [208], [129], [114], [55], [210], [324], [206], [314], [98], [176], [140], [311], [26], [126], [281], [93], [282], [190], [234], [318], [123]
	Case Study	13	[233], [175], [178], [147], [59], [256], [10], [294], [227], [201], [6], [39], [260]
	Analytic	1	[291]
	Example	1	[26]
J	Experiment	10	[208], [92], [55], [70], [98], [311], [26], [87], [156], [234]
	Case Study	7	[233], [175], [10], [294], [227], [6], [187]
	Analytic	0	
	Example	1	[26]
K	Experiment	0	
	Case Study	0	

Table 4.6: (continued)

Context ID	**Evaluation Method**	**Count**	**Papers**
	Analytic	1	[211]
	Example	0	
L	Experiment	0	
	Case Study	0	
	Analytic	0	
	Example	0	
M	Experiment	46	[101], [221], [333], [197], [196], [275], [77], [204], [331], [38], [81], [74], [15], [167], [209], [43], [327], [251], [191], [192], [303], [329], [341], [61], [274], [17], [222], [57], [305], [22], [218], [160], [216], [53], [21], [37], [263], [337], [315], [119], [179], [235], [49], [244], [69], [302]
	Case Study	25	[32], [295], [205], [157], [44], [325], [323], [220], [228], [40], [89], [17], [33], [85], [75], [163], [242], [243], [267], [299], [320], [29], [107], [41], [183]
	Analytic	5	[224], [306], [151], [301], [195]
	Example	3	[197], [28], [60]
N	Experiment	39	[333], [197], [65], [62], [275], [332], [58], [94], [2], [204], [79], [74], [241], [8], [209], [153], [327], [91], [251], [191], [326], [341], [277], [222], [57], [22], [160], [216], [219], [37], [263], [262], [5], [264], [179], [235], [49], [69], [302]

Table 4.6: (continued)

Context ID	**Evaluation Method**	**Count**	**Papers**
	Case Study	15	[279], [50], [44], [325], [323], [40], [89], [17], [163], [118], [158], [267], [347], [107], [41]
	Analytic	0	
	Example	5	[197], [58], [74], [45], [347]
O	Experiment	2	[99], [27]
	Case Study	10	[288], [50], [157], [51], [276], [261], [313], [158], [27], [320]
	Analytic	5	[229], [238], [177], [339], [306]
	Example	1	[272]
P	Experiment	4	[333], [79], [329], [5]
	Case Study	3	[279], [313], [242]
	Analytic	0	
	Example	1	[8]

5 Discussion

Our map of the field reveals that interest in research evaluating software testing techniques has grown significantly since 2007. Despite the broad scope of the field, we see that this interest does manifest itself in a few publication venues with a much higher relative concentration of relevant papers. Contributions in the field come almost entirely from academia with only a small percentage of papers written by authors affiliated with industry. Even though industrial contributions are relatively few, the distribution of evaluation methods and dimensions are somewhat different in this set of papers. A large portion of case studies examining the applicability of testing techniques from authors in industry suggests that industry can provide a valuable niche in that area.

Our study also reveals there is a good amount of research evaluating both white-box and black-box testing techniques, with about half of evaluations being of each technique type. We found that black-box technique evaluations focused largely on combinatorial and random testing techniques; leaving a relative shortage of research evaluating other black-box testing techniques. For the most part, the distribution of evaluation methods and evaluation dimensions in black-box evaluations is similar to that of white-box evaluations. That said, black-box evaluations more often utilize case studies and analytic evaluations when assessing techniques.

In general, evaluations of software testing techniques are overwhelmingly empirical studies in the form of experiments and case studies with a large focus on evaluating effectiveness and efficiency. On the other hand, there are gaps in research evaluating scalability and applicability. Based on the distribution of the dimensions of these evaluations, we can provide insight on what is the state of the art when it comes to evaluating software testing techniques:

1. For researchers looking to evaluate the effectiveness of their testing technique experiments were by far the most common methodology for doing so. Despite being the most common method of evaluation, a majority of experiments looking at the effectiveness of techniques neglected to provide a hypothesis with hypothesis testing. Less than 20% did so. Only about half of experiments met the justified context criteria or provided a serious discussion of threats to validity. About 75% of experiments utilized descriptive statistics. The second most common method for evaluating effectiveness was case studies. These were often used when research goals had to do with evaluating the technique in an industrial context unsuitable for the level of control required for an experiment. The case studies did a poor job of meeting the case study quality criteria described in section 3.5. About %57 utilize data triangulation, %20 define research questions, and %42 provide a serious discussion of threats to validity. Only a few papers used examples or analytic methods to demonstrate the effectiveness of their technique. In short, experiments should be used for evaluating the effectiveness of testing techniques when possible and experiments are so far relatively weak according to proper experiment methodology laid out by [322].

2. The state of the art is fairly similar when it comes to evaluating the efficiency of testing techniques. Experiments were again by far the most common methodology for doing so. Many of these experiments also neglected to provide hypothesis testing or discuss threats to their validity; something that can be improved upon in this field. Case studies were the second most common method used and were of similar quality to those evaluating effectiveness. A few analytic evaluations and no examples were used to assess efficiency.

3. For researchers looking to evaluate the applicability of their testing technique, case studies were the most used by a significant margin. These case studies did a better job of utilizing data triangulation and clearly defining research questions. Still, only %35 provided a serious discussion of threats to validity. Examples were the next most common method used for assessing applicability. These assessments tended to be simple demonstrations of how a technique could be applied in different contexts as opposed to a more rigorous empirical evaluation. Despite being the most common evaluation method, experiments evaluated the applicability of testing techniques the least. In short, case studies should be used in most cases to assess the applicability of testing techniques, with examples being used for simpler demonstrations of applicability.

4. Finally, for researchers looking to evaluate the scalability of their testing techniques, case studies and experiments were the most common methods for doing so. Even though only 9 scalability evaluations were collected in this mapping study, almost all of them utilized case studies or experiments. As mentioned earlier, Scalability was a dimension in which the distribution of evaluation methods changed drastically with testing technique type. We see that the scalability of white-box techniques is only evaluated using experiments while the scalability of black-box techniques largely utilizes case studies. Thus, researchers looking to follow the state of the art when evaluating the scalability of their testing technique should consider the testing technique type when deciding between experiments and case studies.

In terms of contribution type, most of the collected papers performed an evaluation of some software testing technique. There were relatively very few papers actually discussing how techniques should be evaluated or proposing a new methodology for doing so. That said, a few important papers with the latter contribution type were presented in section 4.2.4. These papers suggest that convergence in empirical study methodology and more careful analysis and characterization of objects to which treatments are applied will significantly improve reproducibility and the efficacy of claims made in evaluating software testing techniques.

6 Case Study

To demonstrate how the results of this mapping study can be used by researchers looking to evaluate a particular testing technique, we present a small case study based on the case of our peers who are interested in evaluating the effectiveness of a novel black-box testing technique. We first introduce the case in more detail. Then we step through various sections of the results; discussing how each section helps us develop an understanding of how the novel black-box testing technique developed by our peers should be evaluated.

6.1 The Case

One of the motivating examples for this mapping study came from our peers who developed a novel black-box testing technique. As with many researchers who have developed a novel testing technique, a greater understanding of how to evaluate their particular technique was desired. How have other papers evaluated similar testing techniques? Are there any best practices or guidelines to be aware of? Furthermore, the case of our peers presented a particular challenge when evaluating a testing technique empirically. With the source code embedded in the system under test, modifying it between test executions for a large number of test cases was simply infeasible. This made a popular approach like mutation analysis very difficult to apply at the code level. How have other researchers evaluated techniques where this is the case?

6.2 Intuition from Aggregate Information

To begin, we might want to develop some higher level intuition regarding how similar techniques are evaluated in the field. Aggregate statistics and their visualizations presented in the earlier parts of section 4 can help us quickly identify common characteristics of evaluations performed for similar testing technique types and dimensions.

Looking at the evaluation method distribution for the effectiveness dimension in Figure 4.10, we see that over 90% of all effectiveness evaluations were made up of experiments and case studies. Given such a large majority (and in our case the difficulty of performing some analytic evaluation), Figure 4.10 gives us a clear indication that our evaluation should most probably be some empirical evaluation in the form of an experiment or case study. Figure 4.16 gives us similar information, but considers the testing technique type as well. This figure shows that case studies were somewhat more popular in black-box effectiveness evaluations than they were in white-box evaluations. While experiments were certainly the most common method for evaluating the effectiveness of black-box techniques, many papers also utilized case studies. Thus we would likely choose our evaluation method by reading actual papers evaluating the effectiveness of black-box techniques (see section 6.3 below) and by considering whether or not a high level of experimental control is possible.

Another area we might be interested in is how often mutation analysis is utilized in effectiveness evaluations of similar techniques. Figures 4.11-4.14 show us that the proportion of evaluations utilizing mutation analysis remains fairly consistent regardless of evaluation method or testing technique type. For black-box evaluations in particular, Figure 4.13 shows that about one-third utilize mutation analysis. Being such a popular technique, we keep it in mind when considering how to evaluate our technique.

6.3 Locating Related Papers

While aggregate information can give us a quick intuition when it comes to evaluation methods and the use of mutation analysis, it fails to provide a more in-depth understanding of the state of the art in similar testing technique evaluations. We may have many finer-grain questions about how to evaluate our technique or just want to examine papers evaluating similar testing techniques for guidance or inspiration. In our case, we are especially interested in how black-box effectiveness evaluations using mutation analysis are performed when access to the source code is limited. This is where a further understanding of the state of the art is necessary and can be obtained from reading papers performing similar technique evaluations.

Figure 4.17 and Table 4.6 help us to easily locate these papers. As mentioned earlier, Table 4.6 maps each combination of technique type, evaluation dimension, evaluation method, and mutation analysis affiliation to a set of papers along with the set's cardinality. Due to the large number of combinations and table size, Figure 4.17 has been provided as a complementary tool for quickly finding the table row we are interested in. To use the tool, we start at the root node labeled "Technique Type" and work our way down the tree by choosing the category we are interested in at each internal node of the tree. For this case study we are interested in learning about black-box technique evaluations, so we take the right path, labeled "Black-Box", to the *Mutation Analysis Used* internal node. Because we are interested in finding papers that utilize mutation analysis, we then take the left branch to the *Evaluation Dimension* internal node. Finally, our interest in effectiveness evaluations leads us to take the leftmost branch labeled "Effect" and arrive at the leaf node, I. This leaf node represents an identifier for the row in Table 4.6 containing papers evaluating the effectiveness of black-box testing techniques using mutation analysis.

Given our identifier I, we quickly locate the row labeled I in Table 4.6 (note identifiers are in alphabetical order and color coded) to find papers we are interested in. Table 4.6 shows there are 22 experiment and 13 case study papers. We are particularly interested in the evaluations where access to the code may be limited between test executions, so we skim through the set of 35 papers to find such evaluations. This reveals 3 empirical evaluations, [6], [59], and [10], that we can use to learn how other researchers evaluated the effectiveness of their technique under similar conditions. We see that each of the 3 evaluations are able to apply some form of mutation analysis without altering the source code between test executions by utilizing model-level mutants for various models. In particular, [6] reveals a model-based mutation testing tool for UML models and additionally presents a case study demonstrating how model-based mutation testing can be applied to an industrial measurement device using the tool. By referring to [6], we see how we might model our own SUT in UML and utilize model-based mutation analysis to evaluate the effectiveness of our technique.

6.4 Guidelines

After reading through related evaluations, we also may want to consult papers providing higher-level guidelines pertaining to our evaluation. To do so, we refer to Table 4.2 which lists all of the higher level guideline and proposal papers collected in this mapping study by various categories. Looking through these papers and their summaries, we very quickly gather some valuable insights which will help us plan the evaluation of our testing technique. [77] tells us that different techniques may be better suited for finding different types of faults and that the nature of faults found should be considered in testing technique evaluations. [82] suggests that we should consider the experience level of human subjects and should probably apply random selection. [48] and [54] both do an excellent job of warning us about common threats to the validity of empirical research results. [283] and [64] provide reporting guidelines that will help others replicate our study. Finally, a range of papers in the table present applicable mutation cost-reduction techniques we may want to consider.

7 Threats to Validity

The main threats to the validity of this study are common to most mapping studies. While systematic, our methods of gathering a set of papers representative of the field under investigation and our methods of mapping them are not immune to these issues.

A major validity concern in systematic mapping studies is that the set of gathered papers fails to include relevant papers in the field. There are a few reasons why this is a threat to the validity of our particular study:

1. **Limited Search Space:** Relevant papers were only searched for in online databases. Furthermore, our search was only applied to four of the most common online databases. It is possible relevant papers not published online or published in a different online database were missed.

2. **Language Barrier:** Only papers written in English were considered in this study. One paper from the initial search was excluded on this basis. It is possible this paper was relevant.

3. **Search String:** The search string chosen obviously has a large impact on the ability of a search to return relevant papers. It is possible the search string used in this study resulted in relevant papers not being returned from online sources. We attempted to mitigate this threat by systematically deriving our search string from our research goal as suggested by [236] and by applying iterative refinements to our search string based on search results (discussed in section 2.2).

4. **Misleading Titles and Abstracts:** Some relevant papers may have been excluded in title and abstract exclusion due to titles and abstracts not accurately reflecting the content of papers.

Another major validity concern in systematic mapping studies is that gathered relevant papers are misclassified. This is a concern in our study due to the possibility of author error and poorly written abstracts. The threat is reduced by the fact that full text skimmings were applied to relevant papers to adequately perform some classifications.

8 Conclusion and Future Work

With the growing demand for high quality testing techniques it is important that we evaluate them effectively. An understanding of how we currently evaluate techniques and where our evaluations are lacking can give researchers a better idea of how they should evaluate their techniques as well as initiate research to improve technique evaluations. This paper provides such an understanding by mapping out the field in a systematic mapping study; illustrating the current state of the art and identifying research gaps. Based on the state of the art we have presented guidelines for how a researcher should evaluate their particular testing technique and have generated a mapping from categories to sets of papers belonging to them; allowing researchers to easily locate papers in the field that they are interested in.

The study also answers nine specific research questions declared in the introduction:

1. **RQ1.1:** The number of papers published annually increased greatly from 2009-2011 and has remained about at that level. Since 2011, on average about 35 relevant papers were published per year.

2. **RQ1.2:** Software Testing, Verification and Reliability and the International Symposium on Software Testing and Analysis are the two main publication venues, with 33 and 24 relevant contributions respectively. Other major publication venues include the International Conference on Automated Software Engineering, the International Conference on Software Engineering, the International Conference on

Software Testing, the International Symposium on Foundations of Software Engineering, Empirical Software Engineering, and the International Conference on Software Testing, Verification and Validation.

3. **RQ1.3:** A large majority of contributions (%87) are from academia based on author affiliation. Only %13 have authors affiliated with industry.

4. **RQ2.1:** Experiments, case studies, analytic evaluations, and examples are the main methods used for evaluating software testing techniques.

5. **RQ2.2:** Empirical evaluations in the form of experiments make up a very large majority of evaluation methods. Of these, experiments are used quite a bit more. Analytic evaluations and examples are seldom used.

6. **RQ2.3:** Over half of evaluations are of the effectiveness of software testing techniques. %36 evaluate efficiency. A very small remaining proportion of papers evaluate the applicability and scalability of techniques.

7. **RQ2.4** %47 of evaluations were of white-box techniques, %49 of evaluations were of black-box techniques, and %4 of evaluations were of both white-box and black-box techniques.

8. **RQ2.5:** Based on proper experiment and case study methodologies proposed by [312] and [322] respectively, evaluations are of relatively low quality.

9. **RQ2.6:** Most of the papers utilized a method to evaluate a software testing technique. Relatively few papers discussed how testing techniques should be evaluated or proposed a method for doing so.

10. **RQ2.7:** Almost %30 of effectiveness evaluations utilized mutation analysis. This percentage is fairly consistent across white-box and black-box testing technique evaluations.

More generally our work concludes that there is a need for research focused on how testing techniques should be evaluated. Very few papers were classified as proposal papers even though a large number of papers utilized evaluations for techniques. Furthermore, most of the empirical evaluations made were of fairly low quality according to proper methodology guidelines. While it is good that many researchers evaluate their techniques, it seems clear the field is lacking more serious testing technique evaluations that are influenced by findings from guideline research. Maturing in this area may greatly enhance our assessment capabilities and as a result further our understanding of the effectiveness, efficiency, scalability, and applicability of software testing techniques.

Bibliography

[1] 2018. URL: https://www.ibeta.com/historys-most-expensive-software-bugs/.

[2] P. Accioly, P. Borba, and R. Bonifcio. "Comparing Two Black-Box Testing Strategies for Software Product Lines". In: *2012 Sixth Brazilian Symposium on Software Components, Architectures and Reuse*. Sept. 2012, pp. 1–10. DOI: 10.1109/SBCARS.2012.17.

[3] Wasif Afzal et al. "An experiment on the effectiveness and efficiency of exploratory testing". en. In: *Empirical Software Engineering* 20.3 (June 2015), pp. 844–878. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-014-9301-4. URL: http://link.springer.com/article/10.1007/s10664-014-9301-4 (visited on 07/08/2018).

[4] Khushboo Agarwal and Gursaran Srivastava. "Towards Software Test Data Generation Using Discrete Quantum Particle Swarm Optimization". In: *Proceedings of the 3rd India Software Engineering Conference*. ISEC '10. New York, NY, USA: ACM, 2010, pp. 65–68. ISBN: 978-1-60558-922-0. DOI: 10.1145/1730874.1730888. URL: http://doi.acm.org/10.1145/1730874.1730888 (visited on 07/07/2018).

[5] Bernhard K. Aichernig, Dejan Nikovi, and Stefan Tiran. "Scalable Incremental Test-case Generation from Large Behavior Models". en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Cham, July 2015, pp. 1–18. ISBN:

978-3-319-21214-2 978-3-319-21215-9. DOI: 10.1007/978-3-319-21215-9_1. URL: http://link.springer.com/chapter/10.1007/978-3-319-21215-9_1 (visited on 07/09/2018).

[6] Bernhard K. Aichernig et al. "Model-Based Mutation Testing of an Industrial Measurement Device". en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Cham, July 2014, pp. 1–19. ISBN: 978-3-319-09098-6 978-3-319-09099-3. DOI: 10.1007/978-3-319-09099-3_1. URL: http://link.springer.com/chapter/10.1007/978-3-319-09099-3_1 (visited on 07/09/2018).

[7] Roger T. Alexander, Jeff Offutt, and Andreas Stefik. "Testing coupling relationships in object-oriented programs". en. In: *Software Testing, Verification and Reliability* 20.4 (2010), pp. 291–327. ISSN: 1099-1689. DOI: 10.1002/stvr.417. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.417 (visited on 07/07/2018).

[8] S. Ali et al. "Generating Test Data from OCL Constraints with Search Techniques". In: *IEEE Transactions on Software Engineering* 39.10 (Oct. 2013), pp. 1376–1402. ISSN: 0098-5589. DOI: 10.1109/TSE.2013.17.

[9] Mohammad Amin Alipour et al. "Generating Focused Random Tests Using Directed Swarm Testing". In: *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ISSTA 2016. New York, NY, USA: ACM, 2016, pp. 70–81. ISBN: 978-1-4503-4390-9. DOI: 10.1145/2931037.2931056. URL: http://doi.acm.org/10.1145/2931037.2931056 (visited on 07/08/2018).

[10] Harith Aljumaily, Dolores Cuadra, and Paloma Martnez. "Applying black-box testing to UML/OCL database models". en. In: *Software Quality Journal* 22.2 (June 2014), pp. 153–184. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-012-9192-9. URL: http://link.springer.com/article/10.1007/s11219-012-9192-9 (visited on 07/08/2018).

[11] Tristan Allwood, Cristian Cadar, and Susan Eisenbach. "High Coverage Testing of Haskell Programs". In: *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ISSTA '11. New York, NY, USA: ACM, 2011, pp. 375–385. ISBN: 978-1-4503-0562-4. DOI: 10.1145/2001420.2001465. URL: http://doi.acm.org/10.1145/2001420.2001465 (visited on 07/07/2018).

[12] N. Alshahwan and M. Harman. "Automated web application testing using search based software engineering". In: *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. Nov. 2011, pp. 3–12. DOI: 10.1109/ASE.2011.6100082.

[13] Mohammad Alshraideh, Leonardo Bottaci, and Basel A. Mahafzah. "Using program data-state scarcity to guide automatic test data generation". en. In: *Software Quality Journal* 18.1 (Mar. 2010), pp. 109–144. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-009-9083-x. URL: http://link.springer.com/article/10.1007/s11219-009-9083-x (visited on 07/09/2018).

[14] Mohammad Alshraideh, Basel A. Mahafzah, and Saleh Al-Sharaeh. "A multiple-population genetic algorithm for branch coverage test data generation". en. In: *Software Quality Journal* 19.3 (Sept. 2011), pp. 489–513. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-010-9117-4. URL: http://link.springer.com/article/10.1007/s11219-010-9117-4 (visited on 07/08/2018).

[15] D. Amalfitano, A. R. Fasolino, and P. Tramontana. "Rich Internet Application Testing Using Execution Trace Data". In: *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*. Apr. 2010, pp. 274–283. DOI: 10.1109/ICSTW.2010.34.

[16] Domenico Amalfitano et al. "AGRippin: A Novel Search Based Testing Technique for Android Applications". In: *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*. DeMobile 2015. New York, NY,

USA: ACM, 2015, pp. 5–12. ISBN: 978-1-4503-3815-8. DOI: 10.1145/2804345.2804348. URL: http://doi.acm.org/10.1145/2804345.2804348 (visited on 07/07/2018).

[17] Domenico Amalfitano et al. "Using GUI Ripping for Automated Testing of Android Applications". In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ASE 2012. New York, NY, USA: ACM, 2012, pp. 258–261. ISBN: 978-1-4503-1204-2. DOI: 10.1145/2351676.2351717. URL: http://doi.acm.org/10.1145/2351676.2351717 (visited on 07/08/2018).

[18] Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. 1st ed. New York, NY, USA: Cambridge University Press, 2008.

[19] A. Andalib and S. M. Babamir. "A new approach for test case generation by discrete particle swarm optimization algorithm". In: *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*. May 2014, pp. 1180–1185. DOI: 10.1109/IranianCEE.2014.6999714.

[20] James H. Andrews, Felix C. H. Li, and Tim Menzies. "Nighthawk: A Two-level Genetic-random Unit Test Data Generator". In: *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. ASE '07. New York, NY, USA: ACM, 2007, pp. 144–153. ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321654. URL: http://doi.acm.org/10.1145/1321631.1321654 (visited on 07/08/2018).

[21] Cecilia Apa et al. "Effectiveness for detecting faults within and outside the scope of testing techniques: an independent replication". en. In: *Empirical Software Engineering* 19.2 (Apr. 2014), pp. 378–417. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-013-9267-7. URL: http://link.springer.com/article/10.1007/s10664-013-9267-7 (visited on 07/08/2018).

[22] Sven Apel et al. "Strategies for Product-line Verification: Case Studies and Experiments". In: *Proceedings of the 2013 International Conference on Software Engineering*. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 482–491. ISBN: 978-1-4673-3076-3. URL: http://dl.acm.org/citation.cfm?id=2486788.2486852 (visited on 07/08/2018).

[23] Bellanov S. Apilli. "Fault-based Combinatorial Testing of Web Services". In: *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*. OOPSLA '09. New York, NY, USA: ACM, 2009, pp. 731–732. ISBN: 978-1-60558-768-4. DOI: 10.1145/1639950.1639987. URL: http://doi.acm.org/10.1145/1639950.1639987 (visited on 07/08/2018).

[24] A. Arcuri. "RESTful API Automated Test Case Generation". In: *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. July 2017, pp. 9–20. DOI: 10.1109/QRS.2017.11.

[25] Andrea Arcuri. "It really does matter how you normalize the branch distance in search-based software testing". en. In: *Software Testing, Verification and Reliability* 23.2 (2011), pp. 119–147. ISSN: 1099-1689. DOI: 10.1002/stvr.457. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.457 (visited on 07/05/2018).

[26] Andrea Arcuri and Lionel Briand. "Adaptive Random Testing: An Illusion of Effectiveness?" In: *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ISSTA '11. New York, NY, USA: ACM, 2011, pp. 265–275. ISBN: 978-1-4503-0562-4. DOI: 10.1145/2001420.2001452. URL: http://doi.acm.org/10.1145/2001420.2001452 (visited on 07/07/2018).

[27] Andrea Arcuri, Muhammad Zohaib Iqbal, and Lionel Briand. "Black-Box System Testing of Real-Time Embedded Systems Using Random and Search-Based

Testing". en. In: *Testing Software and Systems*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Nov. 2010, pp. 95–110. ISBN: 978-3-642-16572-6 978-3-642-16573-3. DOI: 10.1007/978-3-642-16573-3_8. URL: http://link.springer.com/chapter/10.1007/978-3-642-16573-3_8 (visited on 07/09/2018).

[28] Andrea Arcuri, Muhammad Zohaib Iqbal, and Lionel Briand. "Formal Analysis of the Effectiveness and Predictability of Random Testing". In: *Proceedings of the 19th International Symposium on Software Testing and Analysis*. ISSTA '10. New York, NY, USA: ACM, 2010, pp. 219–230. ISBN: 978-1-60558-823-0. DOI: 10.1145/1831708.1831736. URL: http://doi.acm.org/10.1145/1831708.1831736 (visited on 07/07/2018).

[29] Stephan Arlt et al. "Parameterized GUI Tests". en. In: *Testing Software and Systems*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Nov. 2012, pp. 247–262. ISBN: 978-3-642-34690-3 978-3-642-34691-0. DOI: 10.1007/978-3-642-34691-0_18. URL: http://link.springer.com/chapter/10.1007/978-3-642-34691-0_18 (visited on 07/09/2018).

[30] Shay Artzi et al. "Finding Bugs in Dynamic Web Applications". In: *Proceedings of the 2008 International Symposium on Software Testing and Analysis*. ISSTA '08. New York, NY, USA: ACM, 2008, pp. 261–272. ISBN: 978-1-60558-050-0. DOI: 10.1145/1390630.1390662. URL: http://doi.acm.org/10.1145/1390630.1390662 (visited on 07/07/2018).

[31] Thanassis Avgerinos et al. "Enhancing Symbolic Execution with Veritesting". In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 1083–1094. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568293. URL: http://doi.acm.org/10.1145/2568225.2568293 (visited on 07/08/2018).

[32] Roy Awedikian and Bernard Yannou. “A practical model-based statistical approach for generating functional test cases: application in the automotive industry”. en. In: *Software Testing, Verification and Reliability* 24.2 (2012), pp. 85–123. ISSN: 1099-1689. DOI: 10.1002/stvr.1479. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1479 (visited on 07/05/2018).

[33] Tanzirul Azim and Iulian Neamtiu. “Targeted and Depth-first Exploration for Systematic Testing of Android Apps”. In: *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*. OOPSLA ’13. New York, NY, USA: ACM, 2013, pp. 641–660. ISBN: 978-1-4503-2374-1. DOI: 10.1145/2509136.2509549. URL: http://doi.acm.org/10.1145/2509136.2509549 (visited on 07/08/2018).

[34] A. Baars et al. “Symbolic search-based testing”. In: *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. Nov. 2011, pp. 53–62. DOI: 10.1109/ASE.2011.6100119.

[35] Faezeh Sadat Babamir et al. “Application of Genetic Algorithm in Automatic Software Testing”. en. In: *Networked Digital Technologies*. Communications in Computer and Information Science. Springer, Berlin, Heidelberg, July 2010, pp. 545–552. ISBN: 978-3-642-14305-2 978-3-642-14306-9. DOI: 10.1007/978-3-642-14306-9_54. URL: http://link.springer.com/chapter/10.1007/978-3-642-14306-9_54 (visited on 07/09/2018).

[36] Luke Bajada, Mark Micallef, and Christian Colombo. “Using Control Flow Analysis to Improve the Effectiveness of Incremental Mutation Testing”. In: *Proceedings of the 14th International Workshop on Principles of Software Evolution*. IWPSE 2015. New York, NY, USA: ACM, 2015, pp. 73–78. ISBN: 978-1-4503-3816-5. DOI: 10.1145/2804360.2804369. URL: http://doi.acm.org/10.1145/2804360.2804369 (visited on 07/08/2018).

[37] Juliana M. Balera and Valdivino A. de Santiago Jnior. “An algorithm for combinatorial interaction testing: definitions and rigorous evaluations”. en. In: *Journal of Software Engineering Research and Development* 5.1 (Dec. 2017), p. 10. ISSN: 2195-1721. DOI: 10.1186/s40411-017-0043-z. URL: http://link.springer.com/article/10.1186/s40411-017-0043-z (visited on 07/09/2018).

[38] W. A. Ballance, S. Vilkomir, and W. Jenkins. “Effectiveness of Pair-Wise Testing for Software with Boolean Inputs”. In: *Verification and Validation 2012 IEEE Fifth International Conference on Software Testing*. Apr. 2012, pp. 580–586. DOI: 10.1109/ICST.2012.144.

[39] Gagandeep Batra and Jyotsna Sengupta. “An Efficient Metamorphic Testing Technique Using Genetic Algorithm”. en. In: *Information Intelligence, Systems, Technology and Management*. Communications in Computer and Information Science. Springer, Berlin, Heidelberg, Mar. 2011, pp. 180–188. ISBN: 978-3-642-19422-1 978-3-642-19423-8. DOI: 10.1007/978-3-642-19423-8_19. URL: http://link.springer.com/chapter/10.1007/978-3-642-19423-8_19 (visited on 07/09/2018).

[40] Sebastian Bauersfeld et al. “Evaluating the TESTAR Tool in an Industrial Case Study”. In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM ’14. New York, NY, USA: ACM, 2014, 4:1–4:9. ISBN: 978-1-4503-2774-9. DOI: 10.1145/2652524.2652588. URL: http://doi.acm.org/10.1145/2652524.2652588 (visited on 07/08/2018).

[41] S. Benli et al. “A Comparative Evaluation of Unit Testing Techniques on a Mobile Platform”. In: *2012 Ninth International Conference on Information Technology - New Generations*. Apr. 2012, pp. 263–268. DOI: 10.1109/ITNG.2012.45.

[42] L. Bentes et al. "JFORTES: Java Formal Unit TESt Generation". In: *2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC)*. Nov. 2016, pp. 16–23. DOI: 10.1109/SBESC.2016.012.

[43] C. Bertolini et al. "An Empirical Evaluation of Automated Black Box Testing Techniques for Crashing GUIs". In: *2009 International Conference on Software Testing Verification and Validation*. Apr. 2009, pp. 21–30. DOI: 10.1109/ICST.2009.27.

[44] S. M. B. Bhargavi et al. "Conventional testing and combinatorial testing: A comparative analysis". In: *2016 International Conference on Inventive Computation Technologies (ICICT)*. Vol. 1. Aug. 2016, pp. 1–5. DOI: 10.1109/INVENTIVE.2016.7823200.

[45] A. Bhat and S. M. K. Quadri. "Equivalence class partitioning and boundary value analysis - A review". In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. Mar. 2015, pp. 1557–1562.

[46] Neelesh Bhattacharya et al. "Divide-by-Zero Exception Raising via Branch Coverage". en. In: *Search Based Software Engineering*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2011, pp. 204–218. ISBN: 978-3-642-23715-7 978-3-642-23716-4. DOI: 10.1007/978-3-642-23716-4_19. URL: http://link.springer.com/chapter/10.1007/978-3-642-23716-4_19 (visited on 07/09/2018).

[47] Atieh Monemi Bidgoli et al. "Using Swarm Intelligence to Generate Test Data for Covering Prime Paths". en. In: *Fundamentals of Software Engineering*. Lecture Notes in Computer Science. Springer, Cham, Apr. 2017, pp. 132–147. ISBN: 978-3-319-68971-5 978-3-319-68972-2. DOI: 10.1007/978-3-319-68972-2_9. URL: http://link.springer.com/chapter/10.1007/978-3-319-68972-2_9 (visited on 07/09/2018).

[48] Stefan Biffl et al. "Towards a Semantic Knowledge Base on Threats to Validity and Control Actions in Controlled Experiments". In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '14. New York, NY, USA: ACM, 2014, 49:1–49:4. ISBN: 978-1-4503-2774-9. DOI: 10.1145/2652524.2652568. URL: http://doi.acm.org/10.1145/2652524.2652568 (visited on 07/08/2018).

[49] Roderick Bloem et al. "Case Study: Automatic Test Case Generation for a Secure Cache Implementation". en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Cham, July 2015, pp. 58–75. ISBN: 978-3-319-21214-2 978-3-319-21215-9. DOI: 10.1007/978-3-319-21215-9_4. URL: http://link.springer.com/chapter/10.1007/978-3-319-21215-9_4 (visited on 07/10/2018).

[50] E. Borjesson. "Industrial Applicability of Visual GUI Testing for System and Acceptance Test Automation". In: *Verification and Validation 2012 IEEE Fifth International Conference on Software Testing*. Apr. 2012, pp. 475–478. DOI: 10.1109/ICST.2012.129.

[51] E. Borjesson and R. Feldt. "Automated System Testing Using Visual GUI Testing Tools: A Comparative Study in Industry". In: *Verification and Validation 2012 IEEE Fifth International Conference on Software Testing*. Apr. 2012, pp. 350–359. DOI: 10.1109/ICST.2012.115.

[52] Ella Bounimova, Patrice Godefroid, and David Molnar. "Billions and Billions of Constraints: Whitebox Fuzz Testing in Production". In: *Proceedings of the 2013 International Conference on Software Engineering*. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 122–131. ISBN: 978-1-4673-3076-3. URL: http://dl.acm.org/citation.cfm?id=2486788.2486805 (visited on 07/08/2018).

[53] Pietro Braione et al. “Software testing with code-based test generators: data and lessons learned from a case study with an industrial software component”. en. In: *Software Quality Journal* 22.2 (June 2014), pp. 311–333. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-013-9207-1. URL: http://link.springer.com/article/10.1007/s11219-013-9207-1 (visited on 07/08/2018).

[54] L. C. Briand. “A Critical Analysis of Empirical Research in Software Testing”. In: *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. Sept. 2007, pp. 1–8. DOI: 10.1109/ESEM.2007.40.

[55] Lionel Briand, Y. Labiche, and Q. Lin. “Improving the coverage criteria of UML state machines using data flow analysis”. en. In: *Software Testing, Verification and Reliability* 20.3 (2009), pp. 177–207. ISSN: 1099-1689. DOI: 10.1002/stvr.410. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.410 (visited on 07/07/2018).

[56] Lionel C. Briand. “A Critical Analysis of Empirical Research in Software Testing”. In: *International Symposium on Empirical Software Engineering and Measurement* (2007).

[57] Penelope A. Brooks and Atif M. Memon. “Automated Gui Testing Guided by Usage Profiles”. In: *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. ASE ’07. New York, NY, USA: ACM, 2007, pp. 333–342. ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321681. URL: http://doi.acm.org/10.1145/1321631.1321681 (visited on 07/08/2018).

[58] Rene C. Bryce and Charles J. Colbourn. “A density-based greedy algorithm for higher strength covering arrays”. en. In: *Software Testing, Verification and Reliability* 19.1 (2008), pp. 37–53. ISSN: 1099-1689. DOI: 10.1002/stvr.393.

URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.393 (visited on 07/07/2018).

[59] M. Bures and B. S. Ahmed. "On the Effectiveness of Combinatorial Interaction Testing: A Case Study". In: *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. July 2017, pp. 69–76. DOI: 10.1109/QRS-C.2017.20.

[60] Marcel Bhme and Soumya Paul. "On the Efficiency of Automated Testing". In: *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2014. New York, NY, USA: ACM, 2014, pp. 632–642. ISBN: 978-1-4503-3056-5. DOI: 10.1145/2635868.2635923. URL: http://doi.acm.org/10.1145/2635868.2635923 (visited on 07/08/2018).

[61] Andrea Calvagna, Andrea Fornaia, and Emiliano Tramontana. "Random Versus Combinatorial Effectiveness in Software Conformance Testing: A Case Study". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. New York, NY, USA: ACM, 2015, pp. 1797–1802. ISBN: 978-1-4503-3196-8. DOI: 10.1145/2695664.2695905. URL: http://doi.acm.org/10.1145/2695664.2695905 (visited on 07/08/2018).

[62] Andrea Calvagna and Angelo Gargantini. "T-wise combinatorial interaction test suites construction based on coverage inheritance". en. In: *Software Testing, Verification and Reliability* 22.7 (2011), pp. 507–526. ISSN: 1099-1689. DOI: 10.1002/stvr.466. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.466 (visited on 07/06/2018).

[63] Jos Campos et al. "An Empirical Evaluation of Evolutionary Algorithms for Test Suite Generation". en. In: *Search Based Software Engineering*. Lecture Notes in Computer Science. Springer, Cham, Sept. 2017, pp. 33–48. ISBN: 978-3-319-66298-5 978-3-319-66299-2. DOI: 10.1007/978-3-319-66299-2_3. URL: http:

//link.springer.com/chapter/10.1007/978-3-319-66299-2_3 (visited on 07/10/2018).

[64] Jeffrey C Carver. "Towards Reporting Guidelines for Experimental Replications: A Proposal". en. In: *RESER* (2010), p. 4.

[65] Richard H. Carver and Yu Lei. "Distributed reachability testing of concurrent programs". en. In: *Concurrency and Computation: Practice and Experience* 22.18 (2010), pp. 2445–2466. ISSN: 1532-0634. DOI: 10.1002/cpe.1573. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1573 (visited on 07/06/2018).

[66] T. Carwalo and S. Jaswal. "Exploring hybrid approach for mutant reduction in software testing". In: *2015 International Conference on Communication, Information Computing Technology (ICCICT)*. Jan. 2015, pp. 1–4. DOI: 10.1109/ICCICT.2015.7045699.

[67] Priyanka Chawla, Inderveer Chana, and Ajay Rana. "A novel strategy for automatic test data generation using soft computing technique". en. In: *Frontiers of Computer Science* 9.3 (June 2015), pp. 346–363. ISSN: 2095-2228, 2095-2236. DOI: 10.1007/s11704-014-3496-9. URL: http://link.springer.com/article/10.1007/s11704-014-3496-9 (visited on 07/08/2018).

[68] Ning Chen and Sunghun Kim. "Puzzle-based Automatic Testing: Bringing Humans into the Loop by Solving Puzzles". In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ASE 2012. New York, NY, USA: ACM, 2012, pp. 140–149. ISBN: 978-1-4503-1204-2. DOI: 10.1145/2351676.2351697. URL: http://doi.acm.org/10.1145/2351676.2351697 (visited on 07/08/2018).

[69] T. Y. Chen, F. C. Kuo, and H. Liu. "Distribution Metric Driven Adaptive Random Testing". In: *Seventh International Conference on Quality Software (QSIC 2007)*. Oct. 2007, pp. 274–279. DOI: 10.1109/QSIC.2007.4385507.

[70] T. Y. Chen et al. "Code Coverage of Adaptive Random Testing". In: *IEEE Transactions on Reliability* 62.1 (Mar. 2013), pp. 226–237. ISSN: 0018-9529. DOI: 10.1109/TR.2013.2240898.

[71] Tsong Yueh Chen and Robert Merkel. "An Upper Bound on Software Testing Effectiveness". In: *ACM Trans. Softw. Eng. Methodol.* 17.3 (June 2008), 16:1–16:27. ISSN: 1049-331X. DOI: 10.1145/1363102.1363107. URL: http://doi.acm.org/10.1145/1363102.1363107 (visited on 07/07/2018).

[72] E. H. Choi, T. Fujiwara, and O. Mizuno. "Weighting for Combinatorial Testing by Bayesian Inference". In: *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Mar. 2017, pp. 389–391. DOI: 10.1109/ICSTW.2017.73.

[73] R. Chopra and S. Madan. "Reusing black box test paths for white box testing of websites". In: *2013 3rd IEEE International Advance Computing Conference (IACC)*. Feb. 2013, pp. 1345–1350. DOI: 10.1109/IAdCC.2013.6514424.

[74] C. Chow, T. Y. Chen, and T. H. Tse. "The ART of Divide and Conquer: An Innovative Approach to Improving the Efficiency of Adaptive Random Testing". In: *2013 13th International Conference on Quality Software*. July 2013, pp. 268–275. DOI: 10.1109/QSIC.2013.19.

[75] Kuan-Chun Chuang, Chi-Sheng Shih, and Shih-Hao Hung. "User Behavior Augmented Software Testing for User-centered GUI". In: *Proceedings of the 2011 ACM Symposium on Research in Applied Computation*. RACS '11. New York, NY, USA: ACM, 2011, pp. 200–208. ISBN: 978-1-4503-1087-1. DOI: 10.1145/

2103380.2103421. URL: http://doi.acm.org/10.1145/2103380.2103421 (visited on 07/08/2018).

[76] Cristina Cifuentes et al. "BegBunch: Benchmarking for C Bug Detection Tools". In: *Proceedings of the 2Nd International Workshop on Defects in Large Software Systems: Held in Conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2009)*. DEFECTS '09. New York, NY, USA: ACM, 2009, pp. 16–20. ISBN: 978-1-60558-654-0. DOI: 10.1145/1555860.1555866. URL: http://doi.acm.org/10.1145/1555860.1555866 (visited on 07/07/2018).

[77] I. Ciupa et al. "On the number and nature of faults found by random testing". en. In: *Software Testing, Verification and Reliability* 21.1 (2009), pp. 3–28. ISSN: 1099-1689. DOI: 10.1002/stvr.415. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.415 (visited on 07/06/2018).

[78] Katherine E. Coons, Sebastian Burckhardt, and Madanlal Musuvathi. "GAMBIT: Effective Unit Testing for Concurrency Libraries". In: *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. PPoPP '10. New York, NY, USA: ACM, 2010, pp. 15–24. ISBN: 978-1-60558-877-3. DOI: 10.1145/1693453.1693458. URL: http://doi.acm.org/10.1145/1693453.1693458 (visited on 07/07/2018).

[79] L. Cordeiro and B. Fischer. "Verifying multi-threaded software using smt-based context-bounded model checking". In: *2011 33rd International Conference on Software Engineering (ICSE)*. May 2011, pp. 331–340. DOI: 10.1145/1985793.1985839.

[80] Lajos Cseppent and Zoltn Micskei. "Evaluating code-based test input generator tools". en. In: *Software Testing, Verification and Reliability* 27.6 (2017), e1627.

ISSN: 1099-1689. DOI: 10.1002/stvr.1627. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1627 (visited on 07/05/2018).

[81] J. Czerwonka. "On Use of Coverage Metrics in Assessing Effectiveness of Combinatorial Test Designs". In: *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2013, pp. 257–266. DOI: 10.1109/ICSTW.2013.76.

[82] Marian Daun et al. "The Impact of Students' Skills and Experiences on Empirical Results: A Controlled Experiment with Undergraduate and Graduate Students". In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. EASE '15. New York, NY, USA: ACM, 2015, 29:1–29:6. ISBN: 978-1-4503-3350-4. DOI: 10.1145/2745802.2745829. URL: http://doi.acm.org/10.1145/2745802.2745829 (visited on 07/07/2018).

[83] Pedro Delgado-Prez, Inmaculada Medina-Bulo, and Mercedes G. Merayo. "Using Evolutionary Computation to Improve Mutation Testing". en. In: *Advances in Computational Intelligence*. Lecture Notes in Computer Science. Springer, Cham, June 2017, pp. 381–391. ISBN: 978-3-319-59146-9 978-3-319-59147-6. DOI: 10.1007/978-3-319-59147-6_33. URL: http://link.springer.com/chapter/10.1007/978-3-319-59147-6_33 (visited on 07/10/2018).

[84] Mingjie Deng, Rong Chen, and Zhenjun Du. "Automatic test data generation model by combining dataflow analysis with genetic algorithm". In: *2009 Joint Conferences on Pervasive Computing (JCPC)*. Dec. 2009, pp. 429–434. DOI: 10.1109/JCPC.2009.5420148.

[85] Xavier Devroey, Gilles Perrouin, and Pierre-Yves Schobbens. "Abstract Test Case Generation for Behavioural Testing of Software Product Lines". In: *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2*. SPLC '14. New

York, NY, USA: ACM, 2014, pp. 86–93. ISBN: 978-1-4503-2739-8. DOI: 10.1145/2647908.2655971. URL: http://doi.acm.org/10.1145/2647908.2655971 (visited on 07/08/2018).

[86] Xavier Devroey et al. "A Variability Perspective of Mutation Analysis". In: *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2014. New York, NY, USA: ACM, 2014, pp. 841–844. ISBN: 978-1-4503-3056-5. DOI: 10.1145/2635868.2666610. URL: http://doi.acm.org/10.1145/2635868.2666610 (visited on 07/08/2018).

[87] Xavier Devroey et al. "Featured Model-based Mutation Analysis". In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. New York, NY, USA: ACM, 2016, pp. 655–666. ISBN: 978-1-4503-3900-1. DOI: 10.1145/2884781.2884821. URL: http://doi.acm.org/10.1145/2884781.2884821 (visited on 07/08/2018).

[88] Kyle Dewey et al. "Evaluating Test Suite Effectiveness and Assessing Student Code via Constraint Logic Programming". In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '17. New York, NY, USA: ACM, 2017, pp. 317–322. ISBN: 978-1-4503-4704-4. DOI: 10.1145/3059009.3059051. URL: http://doi.acm.org/10.1145/3059009.3059051 (visited on 07/07/2018).

[89] Rahul Dixit, Christof Lutteroth, and Gerald Weber. "FormTester: Effective Integration of Model-based and Manually Specified Test Cases". In: *Proceedings of the 37th International Conference on Software Engineering - Volume 2*. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 745–748. URL: http://dl.acm.org/citation.cfm?id=2819009.2819154 (visited on 07/08/2018).

[90] TheAnh Do, A. C. M. Fong, and R. Pears. "Scalable automated test generation using coverage guidance and random search". In: *2012 7th International*

Workshop on Automation of Software Test (AST). June 2012, pp. 71–75. DOI: 10.1109/IWAST.2012.6228993.

[91] Thi Bich Ngoc Do et al. "Constructing Test Cases for N-wise Testing from Tree-based Test Models". In: *Proceedings of the Fourth Symposium on Information and Communication Technology*. SoICT '13. New York, NY, USA: ACM, 2013, pp. 275–284. ISBN: 978-1-4503-2454-0. DOI: 10.1145/2542050.2542074. URL: http://doi.acm.org/10.1145/2542050.2542074 (visited on 07/07/2018).

[92] Alos Dreyfus et al. "A random testing approach using pushdown automata". en. In: *Software Testing, Verification and Reliability* 24.8 (2014), pp. 656–683. ISSN: 1099-1689. DOI: 10.1002/stvr.1526. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1526 (visited on 07/06/2018).

[93] Michael Ellims, Darrel Ince, and Marian Petre. "The Effectiveness of T-Way Test Data Generation". en. In: *Computer Safety, Reliability, and Security*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2008, pp. 16–29. ISBN: 978-3-540-87697-7 978-3-540-87698-4. DOI: 10.1007/978-3-540-87698-4_5. URL: http://link.springer.com/chapter/10.1007/978-3-540-87698-4_5 (visited on 07/09/2018).

[94] Andre Takeshi Endo and Adenilso Simao. "Event tree algorithms to generate test sequences for composite Web services". en. In: *Software Testing, Verification and Reliability* 0.0 (2017), e1637. ISSN: 1099-1689. DOI: 10.1002/stvr.1637. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1637 (visited on 07/07/2018).

[95] Christian Engel and Reiner Hhnle. "Generating Unit Tests from Formal Proofs". en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Feb. 2007, pp. 169–188. ISBN: 978-3-540-73769-8 978-3-540-73770-4.

DOI: 10.1007/978-3-540-73770-4_10. URL: http://link.springer.com/chapter/10.1007/978-3-540-73770-4_10 (visited on 07/09/2018).

[96] Emelie Engström, Per Runeson, and Mats Skoglund. "A Systematic Review on Regression Test Selection Techniques". In: *Inf. Softw. Technol.* 52.1 (Jan. 2010), pp. 14–30. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2009.07.001. URL: http://dx.doi.org.du.idm.oclc.org/10.1016/j.infsof.2009.07.001.

[97] Emelie Engström, Mats Skoglund, and Per Runeson. "Empirical Evaluations of Regression Test Selection Techniques: A Systematic Review". In: *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '08. Kaiserslautern, Germany: ACM, 2008, pp. 22–31. ISBN: 978-1-59593-971-5. DOI: 10.1145/1414004.1414011. URL: http://doi.acm.org.du.idm.oclc.org/10.1145/1414004.1414011.

[98] E. P. Enoiu et al. "A Controlled Experiment in Testing of Safety-Critical Embedded Software". In: *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. Apr. 2016, pp. 1–11. DOI: 10.1109/ICST.2016.15.

[99] Eduard P. Enoiu et al. "Automated test generation using model checking: an industrial evaluation". en. In: *International Journal on Software Tools for Technology Transfer* 18.3 (June 2016), pp. 335–353. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-014-0355-9. URL: http://link.springer.com/article/10.1007/s10009-014-0355-9 (visited on 07/09/2018).

[100] Eduard P. Enoiu et al. "Mutation-Based Test Generation for PLC Embedded Software Using Model Checking". en. In: *Testing Software and Systems*. Lecture Notes in Computer Science. Springer, Cham, Oct. 2016, pp. 155–171. ISBN: 978-3-319-47442-7 978-3-319-47443-4. DOI: 10.1007/978-3-319-47443-4_10. URL:

http://link.springer.com/chapter/10.1007/978-3-319-47443-4_10 (visited on 07/10/2018).

[101] Sheikh Umar Farooq, S. M. K. Quadri, and Nesar Ahmad. "A replicated empirical study to evaluate software testing methods". en. In: *Journal of Software: Evolution and Process* 29.9 (2017), e1883. ISSN: 2047-7481. DOI: 10.1002/smr.1883. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1883 (visited on 07/05/2018).

[102] Sheikh Umar Farooq and SMK Quadri. "Empirical Evaluation of Software Testing Techniques in an Open Source Fashion". In: *Proceedings of the 2Nd International Workshop on Conducting Empirical Studies in Industry*. CESI 2014. New York, NY, USA: ACM, 2014, pp. 21–24. ISBN: 978-1-4503-2843-2. DOI: 10.1145/2593690.2593693. URL: http://doi.acm.org/10.1145/2593690.2593693 (visited on 07/07/2018).

[103] Azadeh Farzan et al. "Con2Colic Testing". In: *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 37–47. ISBN: 978-1-4503-2237-9. DOI: 10.1145/2491411.2491453. URL: http://doi.acm.org/10.1145/2491411.2491453 (visited on 07/08/2018).

[104] Kassem Fawaz et al. "PBCOV: a property-based coverage criterion". en. In: *Software Quality Journal* 23.1 (Mar. 2015), pp. 171–202. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-014-9237-3. URL: http://link.springer.com/article/10.1007/s11219-014-9237-3 (visited on 07/09/2018).

[105] R. Feldt and S. Poulding. "Broadening the Search in Search-Based Software Testing: It Need Not Be Evolutionary". In: *2015 IEEE/ACM 8th International Workshop on Search-Based Software Testing*. May 2015, pp. 1–7. DOI: 10.1109/SBST.2015.8.

[106] Javier Ferrer, Francisco Chicano, and Enrique Alba. “Evolutionary algorithms for the multi-objective test data generation problem”. en. In: *Software: Practice and Experience* 42.11 (2011), pp. 1331–1362. ISSN: 1097-024X. DOI: 10.1002/spe.1135. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/spe.1135 (visited on 07/07/2018).

[107] Lars Frantzen et al. “On-The-Fly Model-Based Testing of Web Services with Jambition”. en. In: *Web Services and Formal Methods*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2008, pp. 143–157. ISBN: 978-3-642-01363-8 978-3-642-01364-5. DOI: 10.1007/978-3-642-01364-5_9. URL: http://link.springer.com/chapter/10.1007/978-3-642-01364-5_9 (visited on 07/10/2018).

[108] G. Fraser and A. Arcuri. “The Seed is Strong: Seeding Strategies in Search-Based Software Testing”. In: *Verification and Validation 2012 IEEE Fifth International Conference on Software Testing*. Apr. 2012, pp. 121–130. DOI: 10.1109/ICST.2012.92.

[109] G. Fraser and A. Zeller. “Mutation-Driven Generation of Unit Tests and Oracles”. In: *IEEE Transactions on Software Engineering* 38.2 (Mar. 2012), pp. 278–292. ISSN: 0098-5589. DOI: 10.1109/TSE.2011.93.

[110] Gordon Fraser and Andrea Arcuri. “A Large-Scale Evaluation of Automated Unit Test Generation Using EvoSuite”. In: *ACM Trans. Softw. Eng. Methodol.* 24.2 (Dec. 2014), 8:1–8:42. ISSN: 1049-331X. DOI: 10.1145/2685612. URL: http://doi.acm.org/10.1145/2685612 (visited on 10/10/2018).

[111] Gordon Fraser and Andrea Arcuri. “Achieving scalable mutation-based generation of whole test suites”. en. In: *Empirical Software Engineering* 20.3 (June 2015), pp. 783–812. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-013-

9299-z. URL: http://link.springer.com/article/10.1007/s10664-013-9299-z (visited on 07/09/2018).

[112] Gordon Fraser and Andrea Arcuri. "Sound Empirical Evidence in Software Testing". In: *Proceedings of the 34th International Conference on Software Engineering*. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 178–188. ISBN: 978-1-4673-1067-3. URL: http://dl.acm.org/citation.cfm?id=2337223.2337245 (visited on 07/08/2018).

[113] Gordon Fraser, Andrea Arcuri, and Phil McMinn. "Test Suite Generation with Memetic Algorithms". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. New York, NY, USA: ACM, 2013, pp. 1437–1444. ISBN: 978-1-4503-1963-8. DOI: 10.1145/2463372.2463548. URL: http://doi.acm.org/10.1145/2463372.2463548 (visited on 07/08/2018).

[114] Gordon Fraser and Neil Walkinshaw. "Assessing and generating test sets in terms of behavioural adequacy". en. In: *Software Testing, Verification and Reliability* 25.8 (2015), pp. 749–780. ISSN: 1099-1689. DOI: 10.1002/stvr.1575. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1575 (visited on 07/05/2018).

[115] Gordon Fraser and Andreas Zeller. "Generating Parameterized Unit Tests". In: *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ISSTA '11. New York, NY, USA: ACM, 2011, pp. 364–374. ISBN: 978-1-4503-0562-4. DOI: 10.1145/2001420.2001464. URL: http://doi.acm.org/10.1145/2001420.2001464 (visited on 07/12/2018).

[116] Gordon Fraser et al. "Does Automated White-box Test Generation Really Help Software Testers?" In: *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. ISSTA 2013. New York, NY, USA: ACM, 2013,

pp. 291–301. ISBN: 978-1-4503-2159-4. DOI: 10.1145/2483760.2483774. URL: http://doi.acm.org/10.1145/2483760.2483774 (visited on 07/07/2018).

[117] Stefan J. Galler and Bernhard K. Aichernig. "Survey on test data generation tools". en. In: *International Journal on Software Tools for Technology Transfer* 16.6 (Nov. 2014), pp. 727–751. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-013-0272-3. URL: http://link.springer.com/article/10.1007/s10009-013-0272-3 (visited on 07/08/2018).

[118] Ruizhi Gao et al. "An Empirical Study of Requirements-based Test Generation on an Automobile Control System". In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. SAC '14. New York, NY, USA: ACM, 2014, pp. 1094–1099. ISBN: 978-1-4503-2469-4. DOI: 10.1145/2554850.2554934. URL: http://doi.acm.org/10.1145/2554850.2554934 (visited on 07/08/2018).

[119] Angelo Gargantini. "Using Model Checking to Generate Fault Detecting Tests". en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Feb. 2007, pp. 189–206. ISBN: 978-3-540-73769-8 978-3-540-73770-4. DOI: 10.1007/978-3-540-73770-4_11. URL: http://link.springer.com/chapter/10.1007/978-3-540-73770-4_11 (visited on 07/09/2018).

[120] Angelo Gargantini and Paolo Vavassori. "Efficient Combinatorial Test Generation Based on Multivalued Decision Diagrams". en. In: *Hardware and Software: Verification and Testing*. Lecture Notes in Computer Science. Springer, Cham, Nov. 2014, pp. 220–235. ISBN: 978-3-319-13337-9 978-3-319-13338-6. DOI: 10.1007/978-3-319-13338-6_17. URL: http://link.springer.com/chapter/10.1007/978-3-319-13338-6_17 (visited on 07/09/2018).

[121] G. Gay. "The Fitness Function for the Job: Search-Based Generation of Test Suites That Detect Real Faults". In: *2017 IEEE International Conference on*

Software Testing, Verification and Validation (ICST). Mar. 2017, pp. 345–355. DOI: 10.1109/ICST.2017.38.

[122] G. Gay et al. "The Risks of Coverage-Directed Test Case Generation". In: *IEEE Transactions on Software Engineering* 41.8 (Aug. 2015), pp. 803–819. ISSN: 0098-5589. DOI: 10.1109/TSE.2015.2421011.

[123] L. S. Ghandehari et al. "An Empirical Comparison of Combinatorial and Random Testing". In: *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2014, pp. 68–77. DOI: 10.1109/ICSTW.2014.8.

[124] Indradeep Ghosh et al. "JST: An Automatic Test Generation Tool for Industrial Java Applications with Strings". In: *Proceedings of the 2013 International Conference on Software Engineering*. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 992–1001. ISBN: 978-1-4673-3076-3. URL: http://dl.acm.org/citation.cfm?id=2486788.2486925 (visited on 07/08/2018).

[125] Aggelos Giantsios, Nikolaos Papaspyrou, and Konstantinos Sagonas. "Concolic Testing for Functional Languages". In: *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming*. PPDP '15. New York, NY, USA: ACM, 2015, pp. 137–148. ISBN: 978-1-4503-3516-4. DOI: 10.1145/2790449.2790519. URL: http://doi.acm.org/10.1145/2790449.2790519 (visited on 07/08/2018).

[126] Milos Gligoric et al. "Comparing Non-adequate Test Suites Using Coverage Criteria". In: *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. ISSTA 2013. New York, NY, USA: ACM, 2013, pp. 302–313. ISBN: 978-1-4503-2159-4. DOI: 10.1145/2483760.2483769. URL: http://doi.acm.org/10.1145/2483760.2483769 (visited on 07/08/2018).

[127] Milos Gligoric et al. "Efficient mutation testing of multithreaded code". en. In: *Software Testing, Verification and Reliability* 23.5 (2012), pp. 375–403. ISSN: 1099-1689. DOI: 10.1002/stvr.1469. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1469 (visited on 07/06/2018).

[128] Milos Gligoric et al. "Selective Mutation Testing for Concurrent Code". In: *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. ISSTA 2013. New York, NY, USA: ACM, 2013, pp. 224–234. ISBN: 978-1-4503-2159-4. DOI: 10.1145/2483760.2483773. URL: http://doi.acm.org/10.1145/2483760.2483773 (visited on 07/07/2018).

[129] Kyungmin Go et al. "Pairwise testing for systems with data derived from real-valued variable inputs". en. In: *Software: Practice and Experience* 46.3 (2014), pp. 381–403. ISSN: 1097-024X. DOI: 10.1002/spe.2295. (Visited on 07/05/2018).

[130] S. Godboley, A. Dutta, and D. P. Mohapatra. "Java-HCT: An approach to increase MC/DC using hybrid concolic testing for Java programs". In: *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*. Sept. 2016, pp. 1709–1713.

[131] S. Godboley et al. "Green-JEXJ: A new tool to measure energy consumption of improved concolic testing". In: *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. Oct. 2015, pp. 36–41. DOI: 10.1109/ICGCIoT.2015.7380424.

[132] Sangharatna Godboley et al. "An improved distributed concolic testing approach". en. In: *Software: Practice and Experience* 47.2 (2016), pp. 311–342. ISSN: 1097-024X. DOI: 10.1002/spe.2405. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2405 (visited on 07/07/2018).

[133] Sangharatna Godboley et al. "Making a concolic tester achieve increased MC/DC". en. In: *Innovations in Systems and Software Engineering* 12.4 (Dec. 2016), pp. 319–

332. ISSN: 1614-5046, 1614-5054. DOI: 10.1007/s11334-016-0284-8. URL: http://link.springer.com/article/10.1007/s11334-016-0284-8 (visited on 07/09/2018).

[134] Patrice Godefroid. "Higher-order Test Generation". In: *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '11. New York, NY, USA: ACM, 2011, pp. 258–269. ISBN: 978-1-4503-0663-8. DOI: 10.1145/1993498.1993529. URL: http://doi.acm.org/10.1145/1993498.1993529 (visited on 07/08/2018).

[135] Patrice Godefroid, Adam Kiezun, and Michael Y. Levin. "Grammar-based Whitebox Fuzzing". In: *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '08. New York, NY, USA: ACM, 2008, pp. 206–215. ISBN: 978-1-59593-860-2. DOI: 10.1145/1375581.1375607. URL: http://doi.acm.org/10.1145/1375581.1375607 (visited on 07/07/2018).

[136] Dunwei Gong and Yan Zhang. "Generating test data for both path coverage and fault detection using genetic algorithms". en. In: *Frontiers of Computer Science* 7.6 (Dec. 2013), pp. 822–837. ISSN: 2095-2228, 2095-2236. DOI: 10.1007/s11704-013-3024-3. URL: http://link.springer.com/article/10.1007/s11704-013-3024-3 (visited on 07/09/2018).

[137] Jorge E. González, Natalia Juristo, and Sira Vegas. "A Systematic Mapping Study on Testing Technique Experiments: Has the Situation Changed Since 2000?" In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '14. Torino, Italy: ACM, 2014, 3:1–3:4. ISBN: 978-1-4503-2774-9. DOI: 10.1145/2652524.2652569. URL: http://doi.acm.org.du.idm.oclc.org/10.1145/2652524.2652569.

[138] R. Gopinath et al. "Measuring Effectiveness of Mutant Sets". In: *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Apr. 2016, pp. 132–141. DOI: 10.1109/ICSTW.2016.45.

[139] Rahul Gopinath, Carlos Jensen, and Alex Groce. "Topsy-Turvy: A Smarter and Faster Parallelization of Mutation Analysis". In: *Proceedings of the 38th International Conference on Software Engineering Companion*. ICSE '16. New York, NY, USA: ACM, 2016, pp. 740–743. ISBN: 978-1-4503-4205-6. DOI: 10.1145/2884781.2892655. URL: http://doi.acm.org/10.1145/2884781.2892655 (visited on 07/07/2018).

[140] M. F. Granda et al. "Effectiveness Assessment of an Early Testing Technique Using Model-Level Mutants". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. EASE'17. New York, NY, USA: ACM, 2017, pp. 98–107. ISBN: 978-1-4503-4804-1. DOI: 10.1145/3084226.3084257. URL: http://doi.acm.org/10.1145/3084226.3084257 (visited on 07/07/2018).

[141] Shengjian Guo et al. "Assertion Guided Symbolic Execution of Multithreaded Programs". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pp. 854–865. ISBN: 978-1-4503-3675-8. DOI: 10.1145/2786805.2786841. URL: http://doi.acm.org/10.1145/2786805.2786841 (visited on 07/08/2018).

[142] T. Guo et al. "GramFuzz: Fuzzing testing of web browsers based on grammar analysis and structural mutation". In: *2013 Second International Conference on Informatics Applications (ICIA)*. Sept. 2013, pp. 212–215. DOI: 10.1109/ICoIA.2013.6650258.

[143] Yuepu Guo and Sreedevi Sampath. "Web Application Fault Classification - an Exploratory Study". In: *Proceedings of the Second ACM-IEEE International*

Symposium on Empirical Software Engineering and Measurement. ESEM '08. New York, NY, USA: ACM, 2008, pp. 303–305. ISBN: 978-1-59593-971-5. DOI: 10.1145/1414004.1414060. URL: http://doi.acm.org/10.1145/1414004.1414060 (visited on 07/07/2018).

[144] Atul Gupta and Pankaj Jalote. "An approach for experimentally evaluating effectiveness and efficiency of coverage criteria for software testing". en. In: *International Journal on Software Tools for Technology Transfer* 10.2 (Mar. 2008), pp. 145–160. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-007-0059-5. URL: http://link.springer.com/article/10.1007/s10009-007-0059-5 (visited on 07/08/2018).

[145] William G. J. Halfond and Alessandro Orso. "Improving Test Case Generation for Web Applications Using Automated Interface Discovery". In: *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. ESEC-FSE '07. New York, NY, USA: ACM, 2007, pp. 145–154. ISBN: 978-1-59593-811-4. DOI: 10.1145/1287624.1287646. URL: http://doi.acm.org/10.1145/1287624.1287646 (visited on 07/07/2018).

[146] S. Hamimoune and B. Falah. "Mutation testing techniques: A comparative study". In: *2016 International Conference on Engineering MIS (ICEMIS)*. Sept. 2016, pp. 1–9. DOI: 10.1109/ICEMIS.2016.7745368.

[147] L. T. M. Hanh and N. T. Binh. "Mutation Operators for Simulink Models". In: *2012 Fourth International Conference on Knowledge and Systems Engineering*. Aug. 2012, pp. 54–59. DOI: 10.1109/KSE.2012.22.

[148] Dan Hao et al. "Test-Data Generation Guided by Static Defect Detection". en. In: *Journal of Computer Science and Technology* 24.2 (Mar. 2009), pp. 284–293. ISSN: 1000-9000, 1860-4749. DOI: 10.1007/s11390-009-9224-5. URL: http:

//link.springer.com/article/10.1007/s11390-009-9224-5 (visited on 07/08/2018).

[149] Mark Harman and Phil McMinn. "A Theoretical & Empirical Analysis of Evolutionary Testing and Hill Climbing for Structural Test Data Generation". In: *Proceedings of the 2007 International Symposium on Software Testing and Analysis*. ISSTA '07. New York, NY, USA: ACM, 2007, pp. 73–83. ISBN: 978-1-59593-734-6. DOI: 10.1145/1273463.1273475. URL: http://doi.acm.org/10.1145/1273463.1273475 (visited on 07/08/2018).

[150] Siamak Haschemi and Stephan Weileder. "A Generic Approach to Run Mutation Analysis". en. In: *Testing Practice and Research Techniques*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2010, pp. 155–164. ISBN: 978-3-642-15584-0 978-3-642-15585-7. DOI: 10.1007/978-3-642-15585-7_15. URL: http://link.springer.com/chapter/10.1007/978-3-642-15585-7_15 (visited on 07/10/2018).

[151] M. U. Hayat and N. Qadeer. "Intra Component GUI Test Case Generation Technique". In: *2007 International Conference on Information and Emerging Technologies*. July 2007, pp. 1–5. DOI: 10.1109/ICIET.2007.4381328.

[152] Jane Huffman Hayes, Inies R. Chemannoor, and E. Ashlee Holbrook. "Improved code defect detection with fault links". en. In: *Software Testing, Verification and Reliability* 21.4 (2010), pp. 299–325. ISSN: 1099-1689. DOI: 10.1002/stvr.426. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.426 (visited on 07/07/2018).

[153] H. Hu et al. "A Parallel Implementation Strategy of Adaptive Testing". In: *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*. July 2010, pp. 214–219. DOI: 10.1109/COMPSACW.2010.44.

[154] Yan Hu and He Jiang. “Effective Test Case Generation via Concolic Execution”. en. In: *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*. Lecture Notes in Electrical Engineering. Springer, Berlin, Heidelberg, 2013, pp. 157–164. ISBN: 978-3-642-34530-2 978-3-642-34531-9. DOI: 10.1007/978-3-642-34531-9_17. URL: http://link.springer.com/chapter/10.1007/978-3-642-34531-9_17 (visited on 07/09/2018).

[155] Jiatong Huo et al. “Genetic Programming for Multi-objective Test Data Generation in Search Based Software Testing”. en. In: *AI 2017: Advances in Artificial Intelligence*. Lecture Notes in Computer Science. Springer, Cham, Aug. 2017, pp. 169–181. ISBN: 978-3-319-63003-8 978-3-319-63004-5. DOI: 10.1007/978-3-319-63004-5_14. URL: http://link.springer.com/chapter/10.1007/978-3-319-63004-5_14 (visited on 07/09/2018).

[156] Felix Hbner, Wen-ling Huang, and Jan Peleska. “Experimental evaluation of a novel equivalence class partition testing strategy”. en. In: *Software & Systems Modeling* (Mar. 2017), pp. 1–21. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-017-0595-8. URL: http://link.springer.com/article/10.1007/s10270-017-0595-8 (visited on 07/09/2018).

[157] S. Iftikhar et al. “An automated model based testing approach for platform games”. In: *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Sept. 2015, pp. 426–435. DOI: 10.1109/MODELS.2015.7338274.

[158] Muhammad Zohaib Iqbal, Andrea Arcuri, and Lionel Briand. “Environment modeling and simulation for automated testing of soft real-time embedded software”. en. In: *Software & Systems Modeling* 14.1 (Feb. 2015), pp. 483–524. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-013-0328-6. URL: http:

//link.springer.com/article/10.1007/s10270-013-0328-6 (visited on 07/08/2018).

[159] Mainul Islam and Christoph Csallner. "Generating Test Cases for Programs That Are Coded Against Interfaces and Annotations". In: *ACM Trans. Softw. Eng. Methodol.* 23.3 (June 2014), 21:1–21:38. ISSN: 1049-331X. DOI: 10.1145/2544135. URL: http://doi.acm.org/10.1145/2544135 (visited on 07/08/2018).

[160] Juha Itkonen and Mika V. Mntyl. "Are test cases needed? Replicated comparison between exploratory and test-case-based software testing". en. In: *Empirical Software Engineering* 19.2 (Apr. 2014), pp. 303–342. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-013-9266-8. URL: http://link.springer.com/article/10.1007/s10664-013-9266-8 (visited on 07/08/2018).

[161] Samireh Jalali and Claes Wohlin. "Systematic Literature Studies: Database Searches vs. Backward Snowballing". In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '12. Lund, Sweden: ACM, 2012, pp. 29–38. ISBN: 978-1-4503-1056-7. DOI: 10.1145/2372251.2372257. URL: http://doi.acm.org.du.idm.oclc.org/10.1145/2372251.2372257.

[162] Konrad Jamrozik et al. "Generating Test Suites with Augmented Dynamic Symbolic Execution". en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, June 2013, pp. 152–167. ISBN: 978-3-642-38915-3 978-3-642-38916-0. DOI: 10.1007/978-3-642-38916-0_9. URL: http://link.springer.com/chapter/10.1007/978-3-642-38916-0_9 (visited on 07/09/2018).

[163] Ajay Kumar Jha, Sunghee Lee, and Woo Jin Lee. "Modeling and Test Case Generation of Inter-component Communication in Android". In: *Proceedings of the Second ACM International Conference on Mobile Software Engineering and*

Systems. MOBILESoft '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 113–116. ISBN: 978-1-4799-1934-5. URL: http://dl.acm.org/citation.cfm?id=2825041.2825061 (visited on 07/08/2018).

[164] Y. Jia and M. Harman. "An Analysis and Survey of the Development of Mutation Testing". In: *IEEE Transactions on Software Engineering* 37.5 (Sept. 2011), pp. 649–678. ISSN: 0098-5589. DOI: 10.1109/TSE.2010.62.

[165] Ya-Hui Jia et al. "Generating Software Test Data by Particle Swarm Optimization". en. In: *Simulated Evolution and Learning*. Lecture Notes in Computer Science. Springer, Cham, Dec. 2014, pp. 37–47. ISBN: 978-3-319-13562-5 978-3-319-13563-2. DOI: 10.1007/978-3-319-13563-2_4. URL: http://link.springer.com/chapter/10.1007/978-3-319-13563-2_4 (visited on 07/09/2018).

[166] R. Jin, S. Jiang, and H. Zhang. "Generation of test data based on genetic algorithms and program dependence analysis". In: *2011 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*. Mar. 2011, pp. 116–121. DOI: 10.1109/CYBER.2011.6011775.

[167] N. Juristo et al. "Comparing the Effectiveness of Equivalence Partitioning, Branch Testing and Code Reading by Stepwise Abstraction Applied by Subjects". In: *Verification and Validation 2012 IEEE Fifth International Conference on Software Testing*. Apr. 2012, pp. 330–339. DOI: 10.1109/ICST.2012.113.

[168] Natalia Juristo, Ana M. Morena, and Sira Vegas. "Reviewing 25 Years of Testing Technique Experiments". In: *Empirical Software Engineering* (2004).

[169] R. Just, G. M. Kapfhammer, and F. Schweiggert. "Do Redundant Mutants Affect the Effectiveness and Efficiency of Mutation Analysis?" In: *Verification and Validation 2012 IEEE Fifth International Conference on Software Testing*. Apr. 2012, pp. 720–725. DOI: 10.1109/ICST.2012.162.

[170] Ren Just. "The Major Mutation Framework: Efficient and Scalable Mutation Analysis for Java". In: *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. ISSTA 2014. New York, NY, USA: ACM, 2014, pp. 433–436. ISBN: 978-1-4503-2645-2. DOI: 10.1145/2610384.2628053. URL: http://doi.acm.org/10.1145/2610384.2628053 (visited on 07/07/2018).

[171] Ren Just, Michael D. Ernst, and Gordon Fraser. "Efficient Mutation Analysis by Propagating and Partitioning Infected Execution States". In: *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. ISSTA 2014. New York, NY, USA: ACM, 2014, pp. 315–326. ISBN: 978-1-4503-2645-2. DOI: 10.1145/2610384.2610388. URL: http://doi.acm.org/10.1145/2610384.2610388 (visited on 07/07/2018).

[172] Ren Just, Gregory M. Kapfhammer, and Franz Schweiggert. "Using Conditional Mutation to Increase the Efficiency of Mutation Analysis". In: *Proceedings of the 6th International Workshop on Automation of Software Test*. AST '11. New York, NY, USA: ACM, 2011, pp. 50–56. ISBN: 978-1-4503-0592-1. DOI: 10.1145/1982595.1982606. URL: http://doi.acm.org/10.1145/1982595.1982606 (visited on 07/07/2018).

[173] R. Kannavara et al. "Challenges and opportunities with concolic testing". In: *2015 National Aerospace and Electronics Conference (NAECON)*. June 2015, pp. 374–378. DOI: 10.1109/NAECON.2015.7443099.

[174] Teemu Kanstrn. "Towards a deeper understanding of test coverage". en. In: *Journal of Software Maintenance and Evolution: Research and Practice* 20.1 (2007), pp. 59–76. ISSN: 1532-0618. DOI: 10.1002/smr.362. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/smr.362 (visited on 07/07/2018).

[175] S. K. Khalsa and Y. Labiche. "An Extension of Category Partition Testing for Highly Constrained Systems". In: *2016 IEEE 17th International Symposium on*

High Assurance Systems Engineering (HASE). Jan. 2016, pp. 47–54. DOI: 10.1109/HASE.2016.45.

[176] Sunint Kaur Khalsa, Yvan Labiche, and Johanna Nicoletta. "The Power of Single and Error Annotations in Category Partition Testing: An Experimental Evaluation". In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. EASE '16. New York, NY, USA: ACM, 2016, 28:1–28:10. ISBN: 978-1-4503-3691-8. DOI: 10.1145/2915970.2915999. URL: http://doi.acm.org/10.1145/2915970.2915999 (visited on 07/07/2018).

[177] M. A. Khan and M. Sadiq. "Analysis of black box software testing techniques: A case study". In: *The 2011 International Conference and Workshop on Current Trends in Information Technology (CTIT 11)*. Oct. 2011, pp. 1–5. DOI: 10.1109/CTIT.2011.6107931.

[178] Y. Khan and J. Hassine. "Mutation Operators for the Atlas Transformation Language". In: *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2013, pp. 43–52. DOI: 10.1109/ICSTW.2013.13.

[179] Sabira Khatun et al. "PS2Way: An Efficient Pairwise Search Approach for Test Data Generation". en. In: *Software Engineering and Computer Systems*. Communications in Computer and Information Science. Springer, Berlin, Heidelberg, June 2011, pp. 99–108. ISBN: 978-3-642-22202-3 978-3-642-22203-0. DOI: 10.1007/978-3-642-22203-0_9. URL: http://link.springer.com/chapter/10.1007/978-3-642-22203-0_9 (visited on 07/09/2018).

[180] M. Kim, Y. Kim, and H. Kim. "A Comparative Study of Software Model Checkers as Unit Testing Tools: An Industrial Case Study". In: *IEEE Transactions on Software Engineering* 37.2 (Mar. 2011), pp. 146–160. ISSN: 0098-5589. DOI: 10.1109/TSE.2010.68.

[181] Moonzoo Kim, Yunho Kim, and Yunja Choi. "Concolic testing of the multi-sector read operation for flash storage platform software". en. In: *Formal Aspects of Computing* 24.3 (May 2012), pp. 355–374. ISSN: 0934-5043, 1433-299X. DOI: 10.1007/s00165-011-0200-9. URL: http://link.springer.com/article/10.1007/s00165-011-0200-9 (visited on 07/08/2018).

[182] M. Kintis, M. Papadakis, and N. Malevris. "Evaluating Mutation Testing Alternatives: A Collateral Experiment". In: *2010 Asia Pacific Software Engineering Conference*. Nov. 2010, pp. 300–309. DOI: 10.1109/APSEC.2010.42.

[183] M. Kls et al. "A Large-Scale Technology Evaluation Study: Effects of Model-based Analysis and Testing". In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. May 2015, pp. 119–128. DOI: 10.1109/ICSE.2015.141.

[184] Ken Koster and David Kao. "State Coverage: A Structural Test Adequacy Criterion for Behavior Checking". In: *The 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers*. ESEC-FSE companion '07. New York, NY, USA: ACM, 2007, pp. 541–544. ISBN: 978-1-59593-812-1. DOI: 10.1145/1295014.1295036. URL: http://doi.acm.org/10.1145/1295014.1295036 (visited on 07/08/2018).

[185] Saparya Krishnamoorthy, Michael S. Hsiao, and Loganathan Lingappan. "Strategies for scalable symbolic execution-driven test generation for programs". en. In: *Science China Information Sciences* 54.9 (Sept. 2011), p. 1797. ISSN: 1674-733X, 1869-1919. DOI: 10.1007/s11432-011-4368-7. URL: http://link.springer.com/article/10.1007/s11432-011-4368-7 (visited on 07/09/2018).

[186] D. Kroening et al. “Effective verification of low-level software with nested interrupts”. In: *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2015, pp. 229–234. DOI: 10.7873/DATE.2015.0360.

[187] Felix Kurth, Sibylle Schupp, and Stephan Weileder. “Generating Test Data from a UML Activity Using the AMPL Interface for Constraint Solvers”. en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Cham, July 2014, pp. 169–186. ISBN: 978-3-319-09098-6 978-3-319-09099-3. DOI: 10.1007/978-3-319-09099-3_14. URL: http://link.springer.com/chapter/10.1007/978-3-319-09099-3_14 (visited on 07/09/2018).

[188] Kari Khknen, Olli Saarikivi, and Keijo Heljanko. “Unfolding based automated testing of multithreaded programs”. en. In: *Automated Software Engineering* 22.4 (Dec. 2015), pp. 475–515. ISSN: 0928-8910, 1573-7535. DOI: 10.1007/s10515-014-0150-6. URL: http://link.springer.com/article/10.1007/s10515-014-0150-6 (visited on 07/09/2018).

[189] Kari Khknen, Olli Saarikivi, and Keijo Heljanko. “Using Unfoldings in Automated Testing of Multithreaded Programs”. In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ASE 2012. New York, NY, USA: ACM, 2012, pp. 150–159. ISBN: 978-1-4503-1204-2. DOI: 10.1145/2351676.2351698. URL: http://doi.acm.org/10.1145/2351676.2351698 (visited on 07/08/2018).

[190] Kari Khknen et al. “Experimental Comparison of Concolic and Random Testing for Java Card Applets”. en. In: *Model Checking Software*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2010, pp. 22–39. ISBN: 978-3-642-16163-6 978-3-642-16164-3. DOI: 10.1007/978-3-642-16164-3_3. URL: http://link.springer.com/chapter/10.1007/978-3-642-16164-3_3 (visited on 07/10/2018).

[191] Zhifeng Lai, S. C. Cheung, and W. K. Chan. "Detecting Atomic-set Serializability Violations in Multithreaded Programs Through Active Randomized Testing". In: *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*. ICSE '10. New York, NY, USA: ACM, 2010, pp. 235–244. ISBN: 978-1-60558-719-6. DOI: 10.1145/1806799.1806836. URL: http://doi.acm.org/10.1145/1806799.1806836 (visited on 07/07/2018).

[192] Zhifeng Lai, S. C. Cheung, and W. K. Chan. "Inter-context Control-flow and Data-flow Test Adequacy Criteria for nesC Applications". In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, 2008, pp. 94–104. ISBN: 978-1-59593-995-1. DOI: 10.1145/1453101.1453115. URL: http://doi.acm.org/10.1145/1453101.1453115 (visited on 07/07/2018).

[193] Abdesselam Lakehal and Ioannis Parissis. "Structural coverage criteria for LUSTRE/SCADE programs". en. In: *Software Testing, Verification and Reliability* 19.2 (2008), pp. 133–154. ISSN: 1099-1689. DOI: 10.1002/stvr.394. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.394 (visited on 07/07/2018).

[194] S. Y. Lee et al. "An Improved Technique of Fitness Evaluation for Evolutionary Testing". In: *2011 IEEE 35th Annual Computer Software and Applications Conference Workshops*. July 2011, pp. 190–193. DOI: 10.1109/COMPSACW.2011.41.

[195] Raluca Lefticaru and Florentin Ipate. "An Improved Test Generation Approach from Extended Finite State Machines Using Genetic Algorithms". en. In: *Software Engineering and Formal Methods*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Oct. 2012, pp. 293–307. ISBN: 978-3-642-33825-0 978-3-642-33826-7. DOI: 10.1007/978-3-642-33826-7_20. URL: http:

//link.springer.com/chapter/10.1007/978-3-642-33826-7_20 (visited on 07/09/2018).

[196] Yu Lei et al. "A combinatorial testing strategy for concurrent programs". en. In: *Software Testing, Verification and Reliability* 17.4 (2007), pp. 207–225. ISSN: 1099-1689. DOI: 10.1002/stvr.369. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.369 (visited on 07/05/2018).

[197] Yu Lei et al. "IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing". en. In: *Software Testing, Verification and Reliability* 18.3 (2007), pp. 125–148. ISSN: 1099-1689. DOI: 10.1002/stvr.381. (Visited on 07/05/2018).

[198] Guodong Li et al. "GKLEE: Concolic Verification and Test Generation for GPUs". In: *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. PPoPP '12. New York, NY, USA: ACM, 2012, pp. 215–224. ISBN: 978-1-4503-1160-1. DOI: 10.1145/2145816.2145844. URL: http://doi.acm.org/10.1145/2145816.2145844 (visited on 07/08/2018).

[199] N. Li et al. "Mutation testing in practice using Ruby". In: *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Apr. 2015, pp. 1–6. DOI: 10.1109/ICSTW.2015.7107453.

[200] Nuo Li et al. "Reggae: Automated Test Generation for Programs Using Complex Regular Expressions". In: *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. ASE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 515–519. ISBN: 978-0-7695-3891-4. DOI: 10.1109/ASE.2009.67. URL: https://doi.org/10.1109/ASE.2009.67 (visited on 07/08/2018).

[201] Ping Li et al. "A practical approach to testing GUI systems". en. In: *Empirical Software Engineering* 12.4 (Aug. 2007), pp. 331–357. ISSN: 1382-3256, 1573-7616.

DOI: 10.1007/s10664-006-9031-3. URL: http://link.springer.com/article/10.1007/s10664-006-9031-3 (visited on 07/09/2018).

[202] K. Liaskos and M. Roper. "Automatic Test-Data Generation: An Immunological Approach". In: *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*. Sept. 2007, pp. 77–81. DOI: 10.1109/TAIC.PART.2007.24.

[203] Mengxiang Lin et al. "Enhancing Constraint Based Test Generation by Local Search". In: *Proceedings of the 6th International Conference on Software and Computer Applications*. ICSCA '17. New York, NY, USA: ACM, 2017, pp. 154–158. ISBN: 978-1-4503-4857-7. DOI: 10.1145/3056662.3056672. URL: http://doi.acm.org/10.1145/3056662.3056672 (visited on 07/08/2018).

[204] Y. Lin et al. "A Divergence-Oriented Approach to Adaptive Random Testing of Java Programs". In: *2009 IEEE/ACM International Conference on Automated Software Engineering*. Nov. 2009, pp. 221–232. DOI: 10.1109/ASE.2009.13.

[205] H. Liu and T. Y. Chen. "An Innovative Approach to Randomising Quasi-random Sequences and Its Application into Software Testing". In: *2009 Ninth International Conference on Quality Software*. Aug. 2009, pp. 59–64. DOI: 10.1109/QSIC.2009.16.

[206] H. Liu and T. Y. Chen. "Randomized Quasi-Random Testing". In: *IEEE Transactions on Computers* 65.6 (June 2016), pp. 1896–1909. ISSN: 0018-9340. DOI: 10.1109/TC.2015.2455981.

[207] Huai Liu, Fei-Ching Kuo, and Tsong Yueh Chen. "Comparison of adaptive random testing and random testing under various testing and debugging scenarios". en. In: *Software: Practice and Experience* 42.8 (2011), pp. 1055–1074. ISSN: 1097-024X. DOI: 10.1002/spe.1113. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/spe.1113 (visited on 07/05/2018).

[208] Huai Liu et al. “Adaptive random testing through test profiles”. en. In: *Software: Practice and Experience* 41.10 (2011), pp. 1131–1154. ISSN: 1097-024X. DOI: 10.1002/spe.1067. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/spe.1067 (visited on 07/05/2018).

[209] K. Liu, T. Wo, and L. Cui. “A Fine-Grained Fault Detection Technique Based on the Virtual Machine Monitor”. In: *2013 International Conference on Cloud Computing and Big Data*. Dec. 2013, pp. 275–282. DOI: 10.1109/CLOUDCOM-ASIA.2013.18.

[210] Y. Liu and H. Zhu. “An Experimental Evaluation of the Reliability of Adaptive Random Testing Methods”. In: *2008 Second International Conference on Secure System Integration and Reliability Improvement*. July 2008, pp. 24–31. DOI: 10.1109/SSIRI.2008.18.

[211] F. Lonetti and E. Marchetti. “X-MuT: A Tool for the Generation of XSLT Mutants”. In: *2010 Seventh International Conference on the Quality of Information and Communications Technology*. Sept. 2010, pp. 280–285. DOI: 10.1109/QUATIC.2010.52.

[212] Yu-Seung Ma, Yong-Rae Kwon, and Sang-Woon Kim. “Statistical Investigation on Class Mutation Operators”. en. In: *ETRI Journal* 31.2 (2009), pp. 140–150. ISSN: 2233-7326. DOI: 10.4218/etrij.09.0108.0356. URL: http://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.09.0108.0356 (visited on 07/07/2018).

[213] Yuta Maezawa et al. “Validating Ajax Applications Using a Delay-based Mutation Technique”. In: *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. ASE ’14. New York, NY, USA: ACM, 2014, pp. 491–502. ISBN: 978-1-4503-3013-8. DOI: 10.1145/2642937.

2642996. URL: http://doi.acm.org/10.1145/2642937.2642996 (visited on 07/07/2018).

[214] Jan Malburg and Gordon Fraser. "Search-based testing using constraint-based mutation". en. In: *Software Testing, Verification and Reliability* 24.6 (2013), pp. 472–495. ISSN: 1099-1689. DOI: 10.1002/stvr.1508. (Visited on 07/06/2018).

[215] Chengying Mao. "Harmony search-based test data generation for branch coverage in software structural testing". en. In: *Neural Computing and Applications* 25.1 (July 2014), pp. 199–216. ISSN: 0941-0643, 1433-3058. DOI: 10.1007/s00521-013-1474-z. URL: http://link.springer.com/article/10.1007/s00521-013-1474-z (visited on 07/09/2018).

[216] Chengying Mao, Tsong Yueh Chen, and Fei-Ching Kuo. "Out of sight, out of mind: a distance-aware forgetting strategy for adaptive random testing". en. In: *Science China Information Sciences* 60.9 (Sept. 2017), p. 092106. ISSN: 1674-733X, 1869-1919. DOI: 10.1007/s11432-016-0087-0. URL: http://link.springer.com/article/10.1007/s11432-016-0087-0 (visited on 07/08/2018).

[217] Ke Mao, Mark Harman, and Yue Jia. "Sapienz: Multi-objective Automated Testing for Android Applications". In: *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ISSTA 2016. New York, NY, USA: ACM, 2016, pp. 94–105. ISBN: 978-1-4503-4390-9. DOI: 10.1145/2931037.2931054. URL: http://doi.acm.org/10.1145/2931037.2931054 (visited on 07/07/2018).

[218] Alessandro Marchetto, Filippo Ricca, and Paolo Tonella. "A case study-based comparison of web testing techniques applied to AJAX web applications". en. In: *International Journal on Software Tools for Technology Transfer* 10.6 (Dec. 2008), pp. 477–492. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-008-

0086-x. URL: http://link.springer.com/article/10.1007/s10009-008-0086-x (visited on 07/08/2018).

[219] Alessandro Marchetto and Paolo Tonella. "Using search-based algorithms for Ajax event sequence generation during testing". en. In: *Empirical Software Engineering* 16.1 (Feb. 2011), pp. 103–140. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-010-9149-1. URL: http://link.springer.com/article/10.1007/s10664-010-9149-1 (visited on 07/08/2018).

[220] L. Mariani et al. "AutoBlackTest: a tool for automatic black-box testing". In: *2011 33rd International Conference on Software Engineering (ICSE)*. May 2011, pp. 1013–1015. DOI: 10.1145/1985793.1985979.

[221] Leonardo Mariani et al. "Automatic testing of GUI-based applications". en. In: *Software Testing, Verification and Reliability* 24.5 (2014), pp. 341–366. ISSN: 1099-1689. DOI: 10.1002/stvr.1538. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1538 (visited on 07/05/2018).

[222] Dusica Marijan et al. "Practical Pairwise Testing for Software Product Lines". In: *Proceedings of the 17th International Software Product Line Conference*. SPLC '13. New York, NY, USA: ACM, 2013, pp. 227–235. ISBN: 978-1-4503-1968-3. DOI: 10.1145/2491627.2491646. URL: http://doi.acm.org/10.1145/2491627.2491646 (visited on 07/08/2018).

[223] Paul D. Marinescu and George Candea. "Efficient Testing of Recovery Code Using Fault Injection". In: *ACM Trans. Comput. Syst.* 29.4 (Dec. 2011), 11:1–11:38. ISSN: 0734-2071. DOI: 10.1145/2063509.2063511. URL: http://doi.acm.org/10.1145/2063509.2063511 (visited on 07/08/2018).

[224] A. Mateen and K. Abbas. "Optimization of model based functional test case generation for android applications". In: *2017 IEEE International Conference*

on Power, Control, Signals and Instrumentation Engineering (ICPCSI). Sept. 2017, pp. 90–95. DOI: 10.1109/ICPCSI.2017.8391869.

[225] Pedro Reales Mateo and Macario Polo Usaola. "Parallel mutation testing". en. In: *Software Testing, Verification and Reliability* 23.4 (2012), pp. 315–350. ISSN: 1099-1689. DOI: 10.1002/stvr.1471. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1471 (visited on 07/06/2018).

[226] Pedro Reales Mateo and Macario Polo Usaola. "Reducing mutation costs through uncovered mutants". en. In: *Software Testing, Verification and Reliability* 25.5-7 (2014), pp. 464–489. ISSN: 1099-1689. DOI: 10.1002/stvr.1534. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1534 (visited on 07/06/2018).

[227] Ernesto C. B. de Matos, Anamaria M. Moreira, and Joo B. de Souza Neto. "An empirical study of test generation with BETA". en. In: *Journal of the Brazilian Computer Society* 22.1 (Dec. 2016), p. 8. ISSN: 0104-6500, 1678-4804. DOI: 10.1186/s13173-016-0048-1. URL: http://link.springer.com/article/10.1186/s13173-016-0048-1 (visited on 07/08/2018).

[228] Lijun Mei, W. K. Chan, and T. H. Tse. "Data Flow Testing of Service Choreography". In: *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. ESEC/FSE '09. New York, NY, USA: ACM, 2009, pp. 151–160. ISBN: 978-1-60558-001-2. DOI: 10.1145/1595696.1595720. URL: http://doi.acm.org/10.1145/1595696.1595720 (visited on 07/07/2018).

[229] Atif M. Memon. "An event-flow model of GUI-based applications for testing". en. In: *Software Testing, Verification and Reliability* 17.3 (2007), pp. 137–157. ISSN: 1099-1689. DOI: 10.1002/stvr.364. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.364 (visited on 07/06/2018).

[230] S. Mirshokraie, A. Mesbah, and K. Pattabiraman. "Efficient JavaScript Mutation Testing". In: *Verification and Validation 2013 IEEE Sixth International Conference on Software Testing*. Mar. 2013, pp. 74–83. DOI: 10.1109/ICST.2013.23.

[231] Mehdi Mirzaaghaei and Ali Mesbah. "DOM-based Test Adequacy Criteria for Web Applications". In: *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. ISSTA 2014. New York, NY, USA: ACM, 2014, pp. 71–81. ISBN: 978-1-4503-2645-2. DOI: 10.1145/2610384.2610406. URL: http://doi.acm.org/10.1145/2610384.2610406 (visited on 07/07/2018).

[232] Supasit Monpratarnchai et al. "Automated Testing for Java Programs Using JPF-based Test Case Generation". In: *SIGSOFT Softw. Eng. Notes* 39.1 (Feb. 2014), pp. 1–5. ISSN: 0163-5948. DOI: 10.1145/2557833.2560575. URL: http://doi.acm.org/10.1145/2557833.2560575 (visited on 07/08/2018).

[233] Rodrigo M. L. M. Moreira et al. "Pattern-based GUI testing: Bridging the gap between design and quality assurance". en. In: *Software Testing, Verification and Reliability* 27.3 (2017), e1629. ISSN: 1099-1689. DOI: 10.1002/stvr.1629. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1629 (visited on 07/07/2018).

[234] S. Mouchawrab et al. "Assessing, Comparing, and Combining State Machine-Based Testing and Structural Testing: A Series of Experiments". In: *IEEE Transactions on Software Engineering* 37.2 (2011), pp. 161–187. ISSN: 0098-5589. DOI: 10.1109/TSE.2010.32.

[235] Miguel Nabuco and Ana C. R. Paiva. "Model-Based Test Case Generation for Web Applications". en. In: *Computational Science and Its Applications ICCSA 2014*. Lecture Notes in Computer Science. Springer, Cham, June 2014, pp. 248–262. ISBN: 978-3-319-09152-5 978-3-319-09153-2. DOI: 10.1007/978-3-319-

09153-2_19. URL: http://link.springer.com/chapter/10.1007/978-3-319-09153-2_19 (visited on 07/10/2018).

[236] Sunil Nair et al. “An Extended Systematic Literature Review on Provision of Evidence for Safety Certification”. In: *Information and Software Technology* (2014).

[237] Akbar Siami Namin and Sahitya Kakarla. “The Use of Mutation in Testing Experiments and Its Sensitivity to External Threats”. In: *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ISSTA ’11. New York, NY, USA: ACM, 2011, pp. 342–352. ISBN: 978-1-4503-0562-4. DOI: 10.1145/2001420.2001461. URL: http://doi.acm.org/10.1145/2001420.2001461 (visited on 07/08/2018).

[238] F. Naseer, S. ur Rehman, and K. Hussain. “Using meta-data technique for component based black box testing”. In: *2010 6th International Conference on Emerging Technologies (ICET)*. Oct. 2010, pp. 276–281. DOI: 10.1109/ICET.2010.5638474.

[239] Roberto Natella, Domenico Cotroneo, and Henrique S. Madeira. “Assessing Dependability with Software Fault Injection: A Survey”. In: *ACM Comput. Surv.* 48.3 (Feb. 2016), 44:1–44:55. ISSN: 0360-0300. DOI: 10.1145/2841425. URL: http://doi.acm.org/10.1145/2841425 (visited on 07/07/2018).

[240] Shimul Kumar Nath, Robert Merkel, and Man Fai Lau. “On the Improvement of a Fault Classification Scheme with Implications for White-box Testing”. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC ’12. New York, NY, USA: ACM, 2012, pp. 1123–1130. ISBN: 978-1-4503-0857-1. DOI: 10.1145/2245276.2231953. URL: http://doi.acm.org/10.1145/2245276.2231953 (visited on 07/07/2018).

[241] C. Nie et al. "Search Based Combinatorial Testing". In: *2012 19th Asia-Pacific Software Engineering Conference*. Vol. 1. Dec. 2012, pp. 778–783. DOI: 10.1109/APSEC.2012.16.

[242] Jeff Offutt and Chandra Alluri. "An industrial study of applying input space partitioning to test financial calculation engines". en. In: *Empirical Software Engineering* 19.3 (June 2014), pp. 558–581. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-012-9229-5. URL: http://link.springer.com/article/10.1007/s10664-012-9229-5 (visited on 07/08/2018).

[243] Jeff Offutt, Vasileios Papadimitriou, and Upsorn Praphamontripong. "A case study on bypass testing of web applications". en. In: *Empirical Software Engineering* 19.1 (Feb. 2014), pp. 69–104. ISSN: 1382-3256, 1573-7616. DOI: 10.1007/s10664-012-9216-x. URL: http://link.springer.com/article/10.1007/s10664-012-9216-x (visited on 07/08/2018).

[244] C. Pacheco et al. "Feedback-Directed Random Test Generation". In: *29th International Conference on Software Engineering (ICSE'07)*. May 2007, pp. 75–84. DOI: 10.1109/ICSE.2007.37.

[245] Bindu Madhavi Padmanabhuni and Hee Beng Kuan Tan. "Light-weight Rule-based Test Case Generation for Detecting Buffer Overflow Vulnerabilities". In: *Proceedings of the 10th International Workshop on Automation of Software Test*. AST '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 48–52. URL: http://dl.acm.org/citation.cfm?id=2819261.2819276 (visited on 07/08/2018).

[246] M. Papadakis and N. Malevris. "An Empirical Evaluation of the First and Second Order Mutation Testing Strategies". In: *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*. Apr. 2010, pp. 90–99. DOI: 10.1109/ICSTW.2010.50.

[247] Mike Papadakis and Yves Le Traon. "Mutation Testing Strategies Using Mutant Classification". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC '13. New York, NY, USA: ACM, 2013, pp. 1223–1229. ISBN: 978-1-4503-1656-9. DOI: 10.1145/2480362.2480592. URL: http://doi.acm.org/10.1145/2480362.2480592 (visited on 07/08/2018).

[248] Mike Papadakis and Nicos Malevris. "Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing". en. In: *Software Quality Journal* 19.4 (Dec. 2011), p. 691. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-011-9142-y. URL: http://link.springer.com/article/10.1007/s11219-011-9142-y (visited on 07/08/2018).

[249] Mike Papadakis, Nicos Malevris, and Maria Kallia. "Towards Automating the Generation of Mutation Tests". In: *Proceedings of the 5th Workshop on Automation of Software Test*. AST '10. New York, NY, USA: ACM, 2010, pp. 111–118. ISBN: 978-1-60558-970-1. DOI: 10.1145/1808266.1808283. URL: http://doi.acm.org/10.1145/1808266.1808283 (visited on 07/08/2018).

[250] Sangmin Park et al. "CarFast: Achieving Higher Statement Coverage Faster". In: *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. FSE '12. New York, NY, USA: ACM, 2012, 35:1–35:11. ISBN: 978-1-4503-1614-9. DOI: 10.1145/2393596.2393636. URL: http://doi.acm.org/10.1145/2393596.2393636 (visited on 07/08/2018).

[251] Soyeon Park, Shan Lu, and Yuanyuan Zhou. "CTrigger: Exposing Atomicity Violation Bugs from Their Hiding Places". In: *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XIV. New York, NY, USA: ACM, 2009, pp. 25–36. ISBN: 978-1-60558-406-5. DOI: 10.1145/1508244.1508249. URL: http://doi.acm.org/10.1145/1508244.1508249 (visited on 07/07/2018).

[252] M. Parthiban and M. R. Sumalatha. “GASE -an input domain reduction and branch coverage system based on Genetic Algorithm and Symbolic Execution”. In: *2013 International Conference on Information Communication and Embedded Systems (ICICES)*. Feb. 2013, pp. 429–433. DOI: 10.1109/ICICES.2013.6508273.

[253] Benny Pasternak, Shmuel Tyszberowicz, and Amiram Yehudai. “GenUTest: a unit test and mock aspect generation tool”. en. In: *International Journal on Software Tools for Technology Transfer* 11.4 (Oct. 2009), p. 273. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-009-0115-4. URL: http://link.springer.com/article/10.1007/s10009-009-0115-4 (visited on 07/09/2018).

[254] M. Patrick and Y. Jia. “Kernel Density Adaptive Random Testing”. In: *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Apr. 2015, pp. 1–10. DOI: 10.1109/ICSTW.2015.7107451.

[255] Kai Petersen et al. “Systematic Mapping Studies in Software Engineering”. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. EASE’08. Italy: BCS Learning & Development Ltd., 2008, pp. 68–77. URL: http://dl.acm.org.du.idm.oclc.org/citation.cfm?id=2227115.2227123.

[256] Florin Pinte, Norbert Oster, and Francesca Saglietti. “Techniques and Tools for the Automatic Generation of Optimal Test Data at Code, Model and Interface Level”. In: *Companion of the 30th International Conference on Software Engineering*. ICSE Companion ’08. New York, NY, USA: ACM, 2008, pp. 927–928. ISBN: 978-1-60558-079-1. DOI: 10.1145/1370175.1370191. URL: http://doi.acm.org/10.1145/1370175.1370191 (visited on 07/08/2018).

[257] Macario Polo, Mario Piattini, and Ignacio GarcaRodrguez. "Decreasing the cost of mutation testing with second-order mutants". en. In: *Software Testing, Verification and Reliability* 19.2 (2008), pp. 111–131. ISSN: 1099-1689. DOI: 10.1002/stvr.392. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.392 (visited on 07/07/2018).

[258] Pablo Ponzio et al. "Field-exhaustive Testing". In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. New York, NY, USA: ACM, 2016, pp. 908–919. ISBN: 978-1-4503-4218-6. DOI: 10.1145/2950290.2950336. URL: http://doi.acm.org/10.1145/2950290.2950336 (visited on 07/07/2018).

[259] S. Poulding, J. A. Clark, and H. Waeselynck. "A Principled Evaluation of the Effect of Directed Mutation on Search-Based Statistical Testing". In: *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. Mar. 2011, pp. 184–193. DOI: 10.1109/ICSTW.2011.36.

[260] M. Prasanna and K. R. Chandran. "Automated Test Case Generation for Object Oriented Systems Using UML Object Diagrams". en. In: *High Performance Architecture and Grid Computing*. Communications in Computer and Information Science. Springer, Berlin, Heidelberg, July 2011, pp. 417–423. ISBN: 978-3-642-22576-5 978-3-642-22577-2. DOI: 10.1007/978-3-642-22577-2_56. URL: http://link.springer.com/chapter/10.1007/978-3-642-22577-2_56 (visited on 07/09/2018).

[261] Elisa Puoskari et al. "Evaluating Applicability of Combinatorial Testing in an Industrial Environment: A Case Study". In: *Proceedings of the 2013 International Workshop on Joining AcadeMiA and Industry Contributions to Testing Automation*. JAMAICA 2013. New York, NY, USA: ACM, 2013, pp. 7–12. ISBN:

978-1-4503-2161-7. DOI: 10.1145/2489280.2489287. URL: http://doi.acm.org/10.1145/2489280.2489287 (visited on 07/08/2018).

[262] Rong-Zhi Qi, Zhi-Jian Wang, and Shui-Yan Li. "A Parallel Genetic Algorithm Based on Spark for Pairwise Test Suite Generation". en. In: *Journal of Computer Science and Technology* 31.2 (Mar. 2016), pp. 417–427. ISSN: 1000-9000, 1860-4749. DOI: 10.1007/s11390-016-1635-5. URL: http://link.springer.com/article/10.1007/s11390-016-1635-5 (visited on 07/09/2018).

[263] Xiao-Fang Qi et al. "Automated Testing of Web Applications Using Combinatorial Strategies". en. In: *Journal of Computer Science and Technology* 32.1 (Jan. 2017), pp. 199–210. ISSN: 1000-9000, 1860-4749. DOI: 10.1007/s11390-017-1699-x. URL: http://link.springer.com/article/10.1007/s11390-017-1699-x (visited on 07/09/2018).

[264] Khandakar Rabbi and Quazi Mamun. "An Effective t-way Test Data Generation Strategy". en. In: *Security and Privacy in Communication Networks*. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, Cham, Oct. 2015, pp. 633–648. ISBN: 978-3-319-28864-2 978-3-319-28865-9. DOI: 10.1007/978-3-319-28865-9_42. URL: http://link.springer.com/chapter/10.1007/978-3-319-28865-9_42 (visited on 07/09/2018).

[265] Zvonimir Rakamari. "STORM: Static Unit Checking of Concurrent Programs". In: *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 2*. ICSE '10. New York, NY, USA: ACM, 2010, pp. 519–520. ISBN: 978-1-60558-719-6. DOI: 10.1145/1810295.1810460. URL: http://doi.acm.org/10.1145/1810295.1810460 (visited on 07/08/2018).

[266] Abhishek Rathore et al. "Application of Genetic Algorithm and Tabu Search in Software Testing". In: *Proceedings of the Fourth Annual ACM Bangalore Con-*

ference. COMPUTE '11. New York, NY, USA: ACM, 2011, 23:1–23:4. ISBN: 978-1-4503-0750-5. DOI: 10.1145/1980422.1980445. URL: http://doi.acm.org/10.1145/1980422.1980445 (visited on 07/07/2018).

[267] Niloofar Razavi, Azadeh Farzan, and Sheila A. McIlraith. "Generating effective tests for concurrent programs via AI automated planning techniques". en. In: *International Journal on Software Tools for Technology Transfer* 16.1 (Feb. 2014), pp. 49–65. ISSN: 1433-2779, 1433-2787. DOI: 10.1007/s10009-013-0277-y. URL: http://link.springer.com/article/10.1007/s10009-013-0277-y (visited on 07/09/2018).

[268] Sion Ll Rhys, Simon Poulding, and John A. Clark. "Using Automated Search to Generate Test Data for Matlab". In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. GECCO '09. New York, NY, USA: ACM, 2009, pp. 1697–1704. ISBN: 978-1-60558-325-9. DOI: 10.1145/1569901.1570128. URL: http://doi.acm.org/10.1145/1569901.1570128 (visited on 07/08/2018).

[269] Jos Miguel Rojas, Gordon Fraser, and Andrea Arcuri. "Seeding strategies in search-based unit test generation". en. In: *Software Testing, Verification and Reliability* 26.5 (2016), pp. 366–401. ISSN: 1099-1689. DOI: 10.1002/stvr.1601. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1601 (visited on 07/06/2018).

[270] Per Runeson et al. *Case Study Research in Software Engineering: Guidelines and Examples*. 1st. Wiley Publishing, 2012. ISBN: 1118104358, 9781118104354.

[271] Neha Rungta, Eric G. Mercer, and Willem Visser. "Efficient Testing of Concurrent Programs with Abstraction-Guided Symbolic Execution". en. In: *Model Checking Software*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, June 2009, pp. 174–191. ISBN: 978-3-642-02651-5 978-3-642-02652-2. DOI:

10.1007/978-3-642-02652-2_16. URL: http://link.springer.com/chapter/10.1007/978-3-642-02652-2_16 (visited on 07/10/2018).

[272] Hazlifah Mohd Rusli et al. "A Comparative Evaluation of State-of-the-art Web Service Composition Testing Approaches". In: *Proceedings of the 6th International Workshop on Automation of Software Test*. AST '11. New York, NY, USA: ACM, 2011, pp. 29–35. ISBN: 978-1-4503-0592-1. DOI: 10.1145/1982595.1982602. URL: http://doi.acm.org/10.1145/1982595.1982602 (visited on 10/10/2018).

[273] F. Saglietti and F. Pinte. "Automated Unit and Integration Testing for Component-based Software Systems". In: *Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systems*. S&D4RCES '10. New York, NY, USA: ACM, 2010, 5:1–5:6. ISBN: 978-1-4503-0368-2. DOI: 10.1145/1868433.1868440. URL: http://doi.acm.org/10.1145/1868433.1868440 (visited on 07/08/2018).

[274] Jose Lorenzo San Miguel and Shingo Takada. "GUI and Usage Model-based Test Case Generation for Android Applications with Change Analysis". In: *Proceedings of the 1st International Workshop on Mobile Development*. Mobile! 2016. New York, NY, USA: ACM, 2016, pp. 43–44. ISBN: 978-1-4503-4643-6. DOI: 10.1145/3001854.3001865. URL: http://doi.acm.org/10.1145/3001854.3001865 (visited on 07/08/2018).

[275] Manoranjan Satpathy et al. "Efficient coverage of parallel and hierarchical stateflow models for test case generation". en. In: *Software Testing, Verification and Reliability* 22.7 (2011), pp. 457–479. ISSN: 1099-1689. DOI: 10.1002/stvr.444. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.444 (visited on 07/06/2018).

[276] Christian Schwarzl and Bernhard Peischl. "Generation of Executable Test Cases Based on Behavioral UML System Models". In: *Proceedings of the 5th Workshop on Automation of Software Test*. AST '10. New York, NY, USA: ACM, 2010, pp. 31–34. ISBN: 978-1-60558-970-1. DOI: 10.1145/1808266.1808271. URL: http://doi.acm.org/10.1145/1808266.1808271 (visited on 07/07/2018).

[277] Koushik Sen. "Effective Random Testing of Concurrent Programs". In: *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. ASE '07. New York, NY, USA: ACM, 2007, pp. 323–332. ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321679. URL: http://doi.acm.org/10.1145/1321631.1321679 (visited on 07/08/2018).

[278] Ohad Shacham et al. "Testing Atomicity of Composed Concurrent Operations". In: *Proceedings of the 2011 ACM International Conference on Object Oriented Programming Systems Languages and Applications*. OOPSLA '11. New York, NY, USA: ACM, 2011, pp. 51–64. ISBN: 978-1-4503-0940-0. DOI: 10.1145/2048066.2048073. URL: http://doi.acm.org/10.1145/2048066.2048073 (visited on 07/08/2018).

[279] Muzammil Shahbaz and Roland Groz. "Analysis and testing of black-box component-based systems by inferring partial models". en. In: *Software Testing, Verification and Reliability* 24.4 (2013), pp. 253–288. ISSN: 1099-1689. DOI: 10.1002/stvr.1491. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1491 (visited on 07/07/2018).

[280] D. Shannon et al. "Efficient Symbolic Execution of Strings for Validating Web Applications". In: *Proceedings of the 2Nd International Workshop on Defects in Large Software Systems: Held in Conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2009)*. DEFECTS '09. New York, NY, USA: ACM, 2009, pp. 22–26. ISBN: 978-1-60558-654-0. DOI:

10.1145/1555860.1555868. URL: http://doi.acm.org/10.1145/1555860.1555868 (visited on 07/07/2018).

[281] Donghwan Shin, Eunkyoung Jee, and Doo-Hwan Bae. “Comprehensive analysis of FBD test coverage criteria using mutants”. en. In: *Software & Systems Modeling* 15.3 (July 2016), pp. 631–645. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-014-0428-y. URL: http://link.springer.com/article/10.1007/s10270-014-0428-y (visited on 07/09/2018).

[282] Donghwan Shin, Eunkyoung Jee, and Doo-Hwan Bae. “Empirical Evaluation on FBD Model-Based Test Coverage Criteria Using Mutation Analysis”. en. In: *Model Driven Engineering Languages and Systems*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2012, pp. 465–479. ISBN: 978-3-642-33665-2 978-3-642-33666-9. DOI: 10.1007/978-3-642-33666-9_30. URL: http://link.springer.com/chapter/10.1007/978-3-642-33666-9_30 (visited on 07/09/2018).

[283] Forrest J. Shull et al. “The Role of Replications in Empirical Software Engineering”. In: *Empirical Softw. Engg.* 13.2 (Apr. 2008), pp. 211–218. ISSN: 1382-3256. DOI: 10.1007/s10664-008-9060-1. URL: http://dx.doi.org/10.1007/s10664-008-9060-1 (visited on 07/12/2018).

[284] Junaid Haroon Siddiqui and Sarfraz Khurshid. “Scaling symbolic execution using staged analysis”. en. In: *Innovations in Systems and Software Engineering* 9.2 (June 2013), pp. 119–131. ISSN: 1614-5046, 1614-5054. DOI: 10.1007/s11334-013-0196-9. URL: http://link.springer.com/article/10.1007/s11334-013-0196-9 (visited on 07/08/2018).

[285] Lucas Serpa Silva and Maarten van Someren. “Evolutionary Testing of Object-oriented Software”. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC ’10. New York, NY, USA: ACM, 2010, pp. 1126–1130. ISBN:

978-1-60558-639-7. DOI: 10.1145/1774088.1774326. URL: http://doi.acm.org/10.1145/1774088.1774326 (visited on 07/07/2018).

[286] M. Singh and V. M. Srivastava. "Extended firm mutation testing: A cost reduction technique for mutation testing". In: *2017 Fourth International Conference on Image Information Processing (ICIIP)*. Dec. 2017, pp. 1–6. DOI: 10.1109/ICIIP.2017.8313788.

[287] Yogesh Singh et al. "Systematic Literature Review on Regression Test Prioritization Techniques". In: *Informatica (Slovenia)* 36 (2012), pp. 379–408.

[288] Harry M. Sneed and Shihong Huang. "The design and use of WSDL-Test: a tool for testing Web services". en. In: *Journal of Software Maintenance and Evolution: Research and Practice* 19.5 (2007), pp. 297–314. ISSN: 1532-0618. DOI: 10.1002/smr.354. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/smr.354 (visited on 07/07/2018).

[289] C. Song, A. Porter, and J. S. Foster. "iTree: Efficiently Discovering High-Coverage Configurations Using Interaction Trees". In: *IEEE Transactions on Software Engineering* 40.3 (Mar. 2014), pp. 251–265. ISSN: 0098-5589. DOI: 10.1109/TSE.2013.55.

[290] Francisco Carlos M. Souza et al. "Strong Mutation-based Test Data Generation Using Hill Climbing". In: *Proceedings of the 9th International Workshop on Search-Based Software Testing*. SBST '16. New York, NY, USA: ACM, 2016, pp. 45–54. ISBN: 978-1-4503-4166-0. DOI: 10.1145/2897010.2897012. URL: http://doi.acm.org/10.1145/2897010.2897012 (visited on 07/07/2018).

[291] Maria Laura Pires Souza and Fbio Fagundes Silveira. "A Model-Based Testing Method for Dynamic Aspect-Oriented Software". en. In: *Computational Science and Its Applications ICCSA 2017*. Lecture Notes in Computer Science. Springer, Cham, July 2017, pp. 95–111. ISBN: 978-3-319-62406-8 978-3-319-62407-5. DOI:

10.1007/978-3-319-62407-5_7. URL: http://link.springer.com/chapter/10.1007/978-3-319-62407-5_7 (visited on 07/10/2018).

[292] S. R. S. Souza et al. "Empirical evaluation of a new composite approach to the coverage criteria and reachability testing of concurrent programs". en. In: *Software Testing, Verification and Reliability* 25.3 (2015), pp. 310–332. ISSN: 1099-1689. DOI: 10.1002/stvr.1568. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1568 (visited on 07/06/2018).

[293] S. R. S. Souza et al. "Structural testing criteria for message-passing parallel programs". en. In: *Concurrency and Computation: Practice and Experience* 20.16 (2008), pp. 1893–1916. ISSN: 1532-0634. DOI: 10.1002/cpe.1297. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1297 (visited on 07/06/2018).

[294] rica Ferreira de Souza, Valdivino Alexandre de Santiago Jnior, and Nandamudi Lankalapalli Vijaykumar. "H-Switch Cover: a new test criterion to generate test case from finite state machines". en. In: *Software Quality Journal* 25.2 (June 2017), pp. 373–405. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-015-9300-8. URL: http://link.springer.com/article/10.1007/s11219-015-9300-8 (visited on 07/08/2018).

[295] Sara E. Sprenkle, Lori L. Pollock, and Lucy M. Simko. "Configuring effective navigation models and abstract test cases for web applications by analysing user behaviour". en. In: *Software Testing, Verification and Reliability* 23.6 (2013), pp. 439–464. ISSN: 1099-1689. DOI: 10.1002/stvr.1496. (Visited on 07/06/2018).

[296] Matt Staats and Corina Psreanu. "Parallel Symbolic Execution for Structural Test Generation". In: *Proceedings of the 19th International Symposium on Software Testing and Analysis*. ISSTA '10. New York, NY, USA: ACM, 2010, pp. 183–

194. ISBN: 978-1-60558-823-0. DOI: 10.1145/1831708.1831732. URL: http://doi.acm.org/10.1145/1831708.1831732 (visited on 07/07/2018).

[297] P. Strooper and M. A. Wojcicki. "Selecting V V Technology Combinations: How to Pick a Winner?" In: *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*. July 2007, pp. 87–96. DOI: 10.1109/ICECCS.2007.40.

[298] Chungha Sung et al. "Static DOM Event Dependency Analysis for Testing Web Applications". In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. New York, NY, USA: ACM, 2016, pp. 447–459. ISBN: 978-1-4503-4218-6. DOI: 10.1145/2950290.2950292. URL: http://doi.acm.org/10.1145/2950290.2950292 (visited on 07/08/2018).

[299] Ana B. Snchez et al. "Variability testing in the wild: the Drupal case study". en. In: *Software & Systems Modeling* 16.1 (Feb. 2017), pp. 173–194. ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-015-0459-z. URL: http://link.springer.com/article/10.1007/s10270-015-0459-z (visited on 07/09/2018).

[300] Kunal Taneja, Yi Zhang, and Tao Xie. "MODA: Automated Test Generation for Database Applications via Mock Objects". In: *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*. ASE '10. New York, NY, USA: ACM, 2010, pp. 289–292. ISBN: 978-1-4503-0116-9. DOI: 10.1145/1858996.1859053. URL: http://doi.acm.org/10.1145/1858996.1859053 (visited on 07/08/2018).

[301] Hongyin Tang et al. "Generating Test Cases to Expose Concurrency Bugs in Android Applications". In: *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ASE 2016. New York, NY, USA: ACM, 2016, pp. 648–653. ISBN: 978-1-4503-3845-5. DOI: 10.1145/2970276.

2970320. URL: http://doi.acm.org/10.1145/2970276.2970320 (visited on 07/08/2018).

[302] A. F. Tappenden and J. Miller. "A Novel Evolutionary Approach for Adaptive Random Testing". In: *IEEE Transactions on Reliability* 58.4 (Dec. 2009), pp. 619–633. ISSN: 0018-9529. DOI: 10.1109/TR.2009.2034288.

[303] Andrew F. Tappenden and James Miller. "Automated Cookie Collection Testing". In: *ACM Trans. Softw. Eng. Methodol.* 23.1 (Feb. 2014), 3:1–3:40. ISSN: 1049-331X. DOI: 10.1145/2559936. URL: http://doi.acm.org/10.1145/2559936 (visited on 07/07/2018).

[304] D. N. Thi, V. D. Hieu, and N. V. Ha. "A Technique for Generating Test Data Using Genetic Algorithm". In: *2016 International Conference on Advanced Computing and Applications (ACOMP).* Nov. 2016, pp. 67–73. DOI: 10.1109/ACOMP.2016.019.

[305] Paul Thomson, Alastair F. Donaldson, and Adam Betts. "Concurrency Testing Using Schedule Bounding: An Empirical Study". In: *Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.* PPoPP '14. New York, NY, USA: ACM, 2014, pp. 15–28. ISBN: 978-1-4503-2656-8. DOI: 10.1145/2555243.2555260. URL: http://doi.acm.org/10.1145/2555243.2555260 (visited on 07/08/2018).

[306] C. Tian, S. Liu, and S. Nakajima. "Utilizing Model Checking for Automatic Test Case Generation from Conjunctions of Predicates". In: *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops.* Mar. 2011, pp. 304–309. DOI: 10.1109/ICSTW.2011.45.

[307] Tian Tian and Dunwei Gong. "Test data generation for path coverage of message-passing parallel programs based on co-evolutionary genetic algorithms". en. In: *Automated Software Engineering* 23.3 (Sept. 2016), pp. 469–500. ISSN: 0928-8910,

1573-7535. DOI: 10.1007/s10515-014-0173-z. URL: http://link.springer.com/article/10.1007/s10515-014-0173-z (visited on 07/09/2018).

[308] Nikolai Tillmann and Jonathan de Halleux. “PexWhite Box Test Generation for .NET”. en. In: *Tests and Proofs*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2008, pp. 134–153. ISBN: 978-3-540-79123-2 978-3-540-79124-9. DOI: 10.1007/978-3-540-79124-9_10. URL: http://link.springer.com/chapter/10.1007/978-3-540-79124-9_10 (visited on 07/09/2018).

[309] Javier Tuya et al. “A Controlled Experiment on White-box Database Testing”. In: *SIGSOFT Softw. Eng. Notes* 33.1 (Jan. 2008), 8:1–8:6. ISSN: 0163-5948. DOI: 10.1145/1344452.1344462. URL: http://doi.acm.org/10.1145/1344452.1344462 (visited on 10/10/2018).

[310] Le Van Phol, Nguyen Thanh Binh, and Ioannis Parissis. “Mutants Generation For Testing Lustre Programs”. In: *Proceedings of the Eighth International Symposium on Information and Communication Technology*. SoICT 2017. New York, NY, USA: ACM, 2017, pp. 425–430. ISBN: 978-1-4503-5328-1. DOI: 10.1145/3155133.3155155. URL: http://doi.acm.org/10.1145/3155133.3155155 (visited on 07/08/2018).

[311] Auri M. R. Vincenzi et al. “The Complementary Aspect of Automatically and Manually Generated Test Case Sets”. In: *Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation*. A-TEST 2016. New York, NY, USA: ACM, 2016, pp. 23–30. ISBN: 978-1-4503-4401-2. DOI: 10.1145/2994291.2994295. URL: http://doi.acm.org/10.1145/2994291.2994295 (visited on 07/07/2018).

[312] T. E. J. Vos et al. “A Methodological Framework for Evaluating Software Testing Techniques and Tools”. In: *2012 12th International Conference on Quality*

Software. 2012 12th International Conference on Quality Software. Aug. 2012, pp. 230–239. DOI: 10.1109/QSIC.2012.16.

[313] Tanja E. J. Vos et al. “Evolutionary functional black-box testing in an industrial setting”. en. In: *Software Quality Journal* 21.2 (June 2013), pp. 259–288. ISSN: 0963-9314, 1573-1367. DOI: 10.1007/s11219-012-9174-y. URL: http://link.springer.com/article/10.1007/s11219-012-9174-y (visited on 07/08/2018).

[314] N. Walkinshaw and G. Fraser. “Uncertainty-Driven Black-Box Test Data Generation”. In: *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. Mar. 2017, pp. 253–263. DOI: 10.1109/ICST.2017.30.

[315] Neil Walkinshaw et al. “Increasing Functional Coverage by Inductive Testing: A Case Study”. en. In: *Testing Software and Systems*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Nov. 2010, pp. 126–141. ISBN: 978-3-642-16572-6 978-3-642-16573-3. DOI: 10.1007/978-3-642-16573-3_10. URL: http://link.springer.com/chapter/10.1007/978-3-642-16573-3_10 (visited on 07/09/2018).

[316] Hongda Wang et al. “Generating effective test cases based on satisfiability modulo theory solvers for service-oriented workflow applications”. en. In: *Software Testing, Verification and Reliability* 26.2 (2015), pp. 149–169. ISSN: 1099-1689. DOI: 10.1002/stvr.1592. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1592 (visited on 07/07/2018).

[317] Huai Wang, W. K. Chan, and T. H. Tse. “Improving the Effectiveness of Testing Pervasive Software via Context Diversity”. In: *ACM Trans. Auton. Adapt. Syst.* 9.2 (July 2014), 9:1–9:28. ISSN: 1556-4665. DOI: 10.1145/2620000. URL: http://doi.acm.org/10.1145/2620000 (visited on 07/08/2018).

[318] S. Wang and J. Offutt. "Comparison of Unit-Level Automated Test Generation Tools". In: *2009 International Conference on Software Testing, Verification, and Validation Workshops*. Apr. 2009, pp. 210–219. DOI: 10.1109/ICSTW.2009.36.

[319] Y. Wang and Y. Wang. "Use Neural Network to Improve Fault Injection Testing". In: *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. July 2017, pp. 377–384. DOI: 10.1109/QRS-C.2017.69.

[320] Christian Wiederseiner et al. "An Open-Source Tool for Automated Generation of Black-Box xUnit Test Code and Its Industrial Evaluation". en. In: *Testing Practice and Research Techniques*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2010, pp. 118–128. ISBN: 978-3-642-15584-0 978-3-642-15585-7. DOI: 10.1007/978-3-642-15585-7_11. URL: http://link.springer.com/chapter/10.1007/978-3-642-15585-7_11 (visited on 07/09/2018).

[321] Andreas Windisch, Stefan Wappler, and Joachim Wegener. "Applying Particle Swarm Optimization to Software Testing". In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. GECCO '07. New York, NY, USA: ACM, 2007, pp. 1121–1128. ISBN: 978-1-59593-697-4. DOI: 10.1145/1276958.1277178. URL: http://doi.acm.org/10.1145/1276958.1277178 (visited on 07/07/2018).

[322] Claes Wohlin et al. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. ISBN: 3642290434, 9783642290435.

[323] P. Wojciak and R. Tzoref-Brill. "System Level Combinatorial Testing in Practice The Concurrent Maintenance Case Study". In: *Verification and Validation 2014 IEEE Seventh International Conference on Software Testing*. Mar. 2014, pp. 103–112. DOI: 10.1109/ICST.2014.23.

[324] S. Wu, Y. Wu, and S. Xu. "Acceleration of Random Testing for Software". In: *2013 IEEE 19th Pacific Rim International Symposium on Dependable Computing*. Dec. 2013, pp. 51–59. DOI: 10.1109/PRDC.2013.15.

[325] S. Xu et al. "A comparative study on black-box testing with open source applications". In: *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. May 2016, pp. 527–532. DOI: 10.1109/SNPD.2016.7515953.

[326] Wen Xu et al. "Designing New Operating Primitives to Improve Fuzzing Performance". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. New York, NY, USA: ACM, 2017, pp. 2313–2328. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134046. URL: http://doi.acm.org/10.1145/3133956.3134046 (visited on 07/08/2018).

[327] A. Yamada et al. "Greedy combinatorial test case generation using unsatisfiable cores". In: *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Sept. 2016, pp. 614–624.

[328] Linmin Yang, Zhe Dang, and Thomas R. Fischer. "Information gain of black-box testing". en. In: *Formal Aspects of Computing* 23.4 (July 2011), pp. 513–539. ISSN: 0934-5043, 1433-299X. DOI: 10.1007/s00165-011-0175-6. URL: http://link.springer.com/article/10.1007/s00165-011-0175-6 (visited on 07/08/2018).

[329] Kohsuke Yatoh et al. "Feedback-controlled Random Test Generation". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ISSTA 2015. New York, NY, USA: ACM, 2015, pp. 316–326. ISBN: 978-1-4503-3620-8. DOI: 10.1145/2771783.2771805. URL: http://doi.acm.org/10.1145/2771783.2771805 (visited on 07/07/2018).

[330] Hiroaki Yoshida et al. "FSX: Fine-grained Incremental Unit Test Generation for C/C++ Programs". In: *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ISSTA 2016. New York, NY, USA: ACM, 2016, pp. 106–117. ISBN: 978-1-4503-4390-9. DOI: 10.1145/2931037.2931055. URL: http://doi.acm.org/10.1145/2931037.2931055 (visited on 07/08/2018).

[331] M. El Youmi and B. Falah. "Testing web applications by unifying Fuzzy and All-Pairs techniques". In: *2014 International Conference on Multimedia Computing and Systems (ICMCS)*. Apr. 2014, pp. 547–551. DOI: 10.1109/ICMCS.2014.6911145.

[332] Mohammed I. Younis and Kamal Z. Zamli. "MC-MIPOG: A Parallel t-Way Test Generation Strategy for Multicore Systems". en. In: *ETRI Journal* 32.1 (2010), pp. 73–83. ISSN: 2233-7326. DOI: 10.4218/etrij.10.0109.0266. URL: http://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.10.0109.0266 (visited on 07/07/2018).

[333] Tingting Yu, Witty Srisa-an, and Gregg Rothermel. "An automated framework to support testing for process-level race conditions". en. In: *Software Testing, Verification and Reliability* 27.4-5 (2017), e1634. ISSN: 1099-1689. DOI: 10.1002/stvr.1634. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1634 (visited on 07/05/2018).

[334] Lingming Zhang, Darko Marinov, and Sarfraz Khurshid. "Faster Mutation Testing Inspired by Test Prioritization and Reduction". In: *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. ISSTA 2013. New York, NY, USA: ACM, 2013, pp. 235–245. ISBN: 978-1-4503-2159-4. DOI: 10.1145/2483760.2483782. URL: http://doi.acm.org/10.1145/2483760.2483782 (visited on 07/08/2018).

[335] Sai Zhang, Hao L, and Michael D. Ernst. "Finding Errors in Multithreaded GUI Applications". In: *Proceedings of the 2012 International Symposium on Software Testing and Analysis*. ISSTA 2012. New York, NY, USA: ACM, 2012, pp. 243–253. ISBN: 978-1-4503-1454-1. DOI: 10.1145/2338965.2336782. URL: http://doi.acm.org/10.1145/2338965.2336782 (visited on 07/07/2018).

[336] Wei Zhang et al. "ConSeq: Detecting Concurrency Bugs Through Sequential Errors". In: *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XVI. New York, NY, USA: ACM, 2011, pp. 251–264. ISBN: 978-1-4503-0266-1. DOI: 10.1145/1950365.1950395. URL: http://doi.acm.org/10.1145/1950365.1950395 (visited on 07/07/2018).

[337] Weixiang Zhang, Bo Wei, and Huisen Du. "An Output-Oriented Approach of Test Data Generation Based on Genetic Algorithm". en. In: *Algorithms and Architectures for Parallel Processing*. Lecture Notes in Computer Science. Springer, Cham, Nov. 2015, pp. 100–108. ISBN: 978-3-319-27160-6 978-3-319-27161-3. DOI: 10.1007/978-3-319-27161-3_9. URL: http://link.springer.com/chapter/10.1007/978-3-319-27161-3_9 (visited on 07/09/2018).

[338] Yucheng Zhang and Ali Mesbah. "Assertions Are Strongly Correlated with Test Suite Effectiveness". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pp. 214–224. ISBN: 978-1-4503-3675-8. DOI: 10.1145/2786805.2786858. URL: http://doi.acm.org/10.1145/2786805.2786858 (visited on 07/08/2018).

[339] W. Zheng and G. Bundell. "Model-Based Software Component Testing: A UML-Based Approach". In: *6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)*. July 2007, pp. 891–899. DOI: 10.1109/ICIS.2007.136.

[340] Wujie Zheng et al. "Random Unit-test Generation with MUT-aware Sequence Recommendation". In: *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*. ASE '10. New York, NY, USA: ACM, 2010, pp. 293–296. ISBN: 978-1-4503-0116-9. DOI: `10.1145/1858996.1859054`. URL: `http://doi.acm.org/10.1145/1858996.1859054` (visited on 07/08/2018).

[341] Hua Zhong, Lingming Zhang, and Sarfraz Khurshid. "Combinatorial Generation of Structurally Complex Test Inputs for Commercial Software Applications". In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. New York, NY, USA: ACM, 2016, pp. 981–986. ISBN: 978-1-4503-4218-6. DOI: `10.1145/2950290.2983959`. URL: `http://doi.acm.org/10.1145/2950290.2983959` (visited on 07/08/2018).

[342] Chixiang Zhou and Phyllis Frankl. "JDAMA: Java database application mutation analyser". en. In: *Software Testing, Verification and Reliability* 21.3 (2011), pp. 241–263. ISSN: 1099-1689. DOI: `10.1002/stvr.462`. (Visited on 07/05/2018).

[343] Yuqin Zhou, Taku Sugihara, and Yuji Sato. "Applying GA with Tabu list for Automatically Generating Test Cases Based on Formal Specification". en. In: *Structured Object-Oriented Formal Language and Method*. Lecture Notes in Computer Science. Springer, Cham, Nov. 2014, pp. 17–31. ISBN: 978-3-319-17403-7 978-3-319-17404-4. DOI: `10.1007/978-3-319-17404-4_2`. URL: `http://link.springer.com/chapter/10.1007/978-3-319-17404-4_2` (visited on 07/10/2018).

[344] Q. Zhu, A. Panichella, and A. Zaidman. "Speeding-Up Mutation Testing via Data Compression and State Infection". In: *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. Mar. 2017, pp. 103–109. DOI: `10.1109/ICSTW.2017.25`.

[345] Ziming Zhu, Xiong Xu, and Li Jiao. "Improved evolutionary generation of test data for multiple paths in search-based software testing". In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. June 2017, pp. 612–620. DOI: 10.1109/CEC.2017.7969367.

[346] Yunxiao Zou et al. "Virtual DOM Coverage for Effective Testing of Dynamic Web Applications". In: *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. ISSTA 2014. New York, NY, USA: ACM, 2014, pp. 60–70. ISBN: 978-1-4503-2645-2. DOI: 10.1145/2610384.2610399. URL: http://doi.acm.org/10.1145/2610384.2610399 (visited on 07/07/2018).

[347] R. S. Zybin et al. "Automation of broad sanity test generation". en. In: *Programming and Computer Software* 34.6 (Nov. 2008), pp. 351–363. ISSN: 0361-7688, 1608-3261. DOI: 10.1134/S0361768808060066. URL: http://link.springer.com/article/10.1134/S0361768808060066 (visited on 07/09/2018).