Evaluating Software Testing Techniques: A Systematic Mapping Study

Mitchell Mayeda

University of Denver

Follow this and additional works at: https://digitalcommons.du.edu/etd

Recommended Citation

https://digitalcommons.du.edu/etd/1599

This Thesis is brought to you for free and open access by the Graduate Studies at Digital Commons @ DU. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ DU. For more information, please contact jennifer.cox@du.edu,dig-commons@du.edu.
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
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

1 Introduction ......................................................... 1  
1.1 Background ...................................................... 2  
1.2 Thesis Layout .................................................... 4  

2 Research Method .................................................... 5  
2.1 Definition of Research Questions ............................... 5  
2.2 Systematic Search ............................................... 7  
2.3 Study Selection .................................................. 8  
2.4 Data Mapping .................................................... 10  

3 Classification Schemes .............................................. 12  
3.1 Evaluation Method .............................................. 12  
3.2 Evaluation Dimension ......................................... 14  
3.3 Testing Technique Type ........................................ 15  
3.4 Contribution Type ............................................... 16  
3.5 Use of Mutation Analysis ...................................... 16  
3.6 Evaluation Quality ............................................. 17  

4 Evaluating Software Testing Techniques: A Map of the Field .... 20  
4.1 Publication Trends ............................................... 20  
4.1.1 Annual Activity Level ....................................... 20  
4.1.2 Main Publication Venues ..................................... 21  
4.1.3 Industry vs Academia ........................................ 22  
4.2 Context-Specific Mappings ..................................... 22  
4.2.1 Evaluation Method .......................................... 22  
4.2.2 Evaluation Dimension ...................................... 24  
4.2.3 Testing Technique Type ..................................... 25  
4.2.4 Contribution Type ........................................... 26  
4.2.5 Use of Mutation Analysis ................................... 28  
4.2.6 Evaluation Quality ......................................... 29  
4.2.7 Distribution of Evaluation Methods Over Time .......... 31  
4.2.8 Relation of Evaluation Method and Dimension .......... 32  
4.2.9 Relation of Mutation Analysis, Evaluation Method, and Technique Type ............................................. 34  
4.2.10 Relation of Author Affiliation, Evaluation Method, and Evaluation Dimension ........................................... 35
## List of Figures

2.1 Overview of the systematic mapping process ........................................ 6
2.2 Overview of the study selection process including the number of papers resulting from each step. .................................................. 11

4.1 Annual number of publications............................................................... 21
4.2 Percentage of contributions from industry and academia......................... 23
4.3 Distribution of primary study evaluations by method.............................. 24
4.4 Number of evaluations by dimension...................................................... 25
4.5 Percentage of white box and black box evaluations ................................ 26
4.6 Contribution type distribution ............................................................... 29
4.7 Mutation Analysis Distribution ............................................................. 30
4.8 Distribution of mutation analysis over time ............................................. 31
4.9 Distribution of evaluation methods over time ......................................... 32
4.10 Distribution of evaluations by method and dimension ............................ 33
4.11 Distribution of mutation analysis experiment papers ............................... 34
4.12 Distribution of mutation analysis case study papers ............................... 35
4.13 Distribution of mutation analysis black-box papers ................................ 36
4.14 Distribution of mutation analysis white-box papers ................................ 37
4.15 Distribution of evaluations by author affiliation, method, and dimension. 39
4.16 Relation of technique type, evaluation method, and evaluation dimension. 40
4.17 Decision tree for quickly locating entries in Table 4.6 .......................... 41
List of Tables

4.1 Main publication venues ................................................. 22
4.2 Guideline papers by category .......................................... 28
4.3 The number and percent of experiments that satisfy each of the experiment evaluation quality criteria ........................................... 30
4.4 Number and percent of case studies that satisfy each of the case study evaluation quality criteria ........................................... 31
4.5 Distribution of papers by evaluation method and evaluation dimension. 33
4.6 Papers belonging to each category combination ......................... 42
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?
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:

\[
\begin{align*}
\text{(evaluate OR validate OR assess)} \\
\text{AND} \\
\text{(effectiveness OR efficiency OR applicability OR scalability)} \\
\text{AND} \\
\text{("software testing" OR "software verification" OR "black-box testing" OR "white-box testing")} \\
\text{AND} \\
\text{(techniques)}
\end{align*}
\]
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.
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.
<table>
<thead>
<tr>
<th>Publication Venue</th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Testing, Verification and Reliability</td>
<td>33</td>
<td>9.85</td>
</tr>
<tr>
<td>International Symposium on Software Testing and Analysis</td>
<td>24</td>
<td>7.16</td>
</tr>
<tr>
<td>International Conference on Automated Software Engineering</td>
<td>17</td>
<td>5.07</td>
</tr>
<tr>
<td>International Conference on Software Engineering</td>
<td>15</td>
<td>4.48</td>
</tr>
<tr>
<td>International Conference on Software Testing</td>
<td>15</td>
<td>4.48</td>
</tr>
<tr>
<td>International Symposium on Foundations of Software Engineer ing</td>
<td>14</td>
<td>4.18</td>
</tr>
<tr>
<td>Empirical Software Engineering</td>
<td>13</td>
<td>3.88</td>
</tr>
<tr>
<td>International Conference on Software Testing, Verification and Validation</td>
<td>13</td>
<td>3.88</td>
</tr>
</tbody>
</table>

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
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.


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.
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.

![Pie chart showing percentages of white box, black box, and both evaluations.](image)

**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>
<thead>
<tr>
<th>Guideline Category</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Artifact</td>
<td>[239] <strong>Assessing Dependability with Software Fault Injection:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>A Survey</strong> Presents an overview of the state of the art in software</td>
</tr>
<tr>
<td></td>
<td>fault injection and insight on which approaches to apply in different</td>
</tr>
<tr>
<td></td>
<td>contexts.</td>
</tr>
</tbody>
</table>
Table 4.2: (continued)

<table>
<thead>
<tr>
<th>Guideline Category</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[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.</td>
</tr>
<tr>
<td></td>
<td>[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.</td>
</tr>
<tr>
<td></td>
<td>[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.</td>
</tr>
<tr>
<td>Evaluation Metric</td>
<td>[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.</td>
</tr>
<tr>
<td></td>
<td>[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.</td>
</tr>
<tr>
<td></td>
<td>[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</td>
</tr>
<tr>
<td>Guideline Category</td>
<td>Papers</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[328] <strong>Information gain of black-box testing</strong></td>
<td>Introduces a novel coverage criteria for assessing black-box tests based on information gain from test cases.</td>
</tr>
<tr>
<td>[81] <strong>On Use of Coverage Metrics in Assessing Effectiveness of Combinatorial Test Designs</strong></td>
<td>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.</td>
</tr>
<tr>
<td>[104] <strong>PBCOV: a property-based coverage criterion</strong></td>
<td>Proposes a new property-based criterion for assessing the adequacy of test suites.</td>
</tr>
<tr>
<td>[184] <strong>State Coverage: A Structural Test Adequacy Criterion for Behavior Checking</strong></td>
<td>Proposes state coverage, a new structural criterion for assessing the adequacy of test suites.</td>
</tr>
<tr>
<td>[293] <strong>Structural testing criteria for message-passing parallel programs</strong></td>
<td>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.</td>
</tr>
<tr>
<td>[122] <strong>The Risks of Coverage-Directed Test Case Generation</strong></td>
<td>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.</td>
</tr>
<tr>
<td>[297] <strong>Selecting V&amp;V Technology Combinations: How to Pick a Winner?</strong></td>
<td>Proposes a systematic method for evaluating verification and validation technique combinations.</td>
</tr>
<tr>
<td>[174] <strong>Towards a deeper understanding of test coverage</strong></td>
<td>Suggests coverage criteria should be calculated at different testing levels instead of for the test suite as a whole.</td>
</tr>
<tr>
<td>Guideline Category</td>
<td>Papers</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td>[143] Web Application Fault ClassificationAn Exploratory Study</td>
<td>Introduces a web application fault classification schema based on the exploration of two large, real-world web systems.</td>
</tr>
<tr>
<td>[82] The Impact of Students Skills and Experiences on Empirical Results: A Controlled Experiment with Undergraduate and Graduate Students</td>
<td>A controlled experiment investigating how the experience of human subjects in empirical studies evaluating effectiveness and efficiency can impact results.</td>
</tr>
<tr>
<td>[48] Towards a Semantic Knowledge Base on Threats to Validity and Control Actions in Controlled Experiments</td>
<td>Proposes a knowledge base of threats to validity to assist researchers in mitigating threats when planning experiments.</td>
</tr>
<tr>
<td>[283] The role of replications in Empirical Software Engineering</td>
<td>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.</td>
</tr>
<tr>
<td>[56] A Critical Analysis of Empirical Research in Software Testing</td>
<td>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.</td>
</tr>
<tr>
<td>[64] Towards Reporting Guidelines for Experimental Replications: A Proposal</td>
<td>Suggests publishing guidelines for experiment replications in order to &quot;increase the value of experimental replications&quot;.</td>
</tr>
</tbody>
</table>
Table 4.2: (continued)

<table>
<thead>
<tr>
<th>Guideline Category</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutation Analysis (c)</td>
<td>[150] A Generic Approach to Run Mutation Analysis</td>
</tr>
<tr>
<td></td>
<td>[144] An approach for experimentally evaluating effectiveness and efficiency of coverage criteria for software testing:</td>
</tr>
<tr>
<td></td>
<td>[169] Do Redundant Mutants Affect the Effectiveness and Efficiency of Mutation Analysis?</td>
</tr>
<tr>
<td></td>
<td>[127] Efficient mutation testing of multithreaded code</td>
</tr>
<tr>
<td></td>
<td>[334] Faster Mutation Testing Inspired by Test Prioritization and Reduction</td>
</tr>
<tr>
<td></td>
<td>[138] Measuring Effectiveness of Mutant Sets</td>
</tr>
<tr>
<td></td>
<td>[310] Mutants Generation For Testing Lustre Programs</td>
</tr>
<tr>
<td>Guideline Category</td>
<td>Papers</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td>[199] Mutation Testing in Practice using Ruby</td>
<td>Presents mutation operators for Ruby and guidelines for mutation testing based on experience from an industrial Ruby project.</td>
</tr>
<tr>
<td>[247] Mutation Testing Strategies using Mutant Classification</td>
<td>Proposes mutant classification strategies to assist in isolating equivalent mutants along with an experimental evaluation of the technique.</td>
</tr>
<tr>
<td>[146] Mutation Testing Techniques: A Comparative Study</td>
<td>An empirical comparison of four mutation testing techniques (operators at class level, operators at method level, all operators, and random sampling).</td>
</tr>
<tr>
<td>[170] The Major Mutation Framework: Efficient and Scalable Mutation Analysis for Java</td>
<td>Introduces a JUnit mutation analysis and fault seeding framework with claims of scalability and efficiency.</td>
</tr>
<tr>
<td>[237] The Use of Mutation in Testing Experiments and its Sensitivity to External Threats</td>
<td>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.</td>
</tr>
<tr>
<td>[83] Using Evolutionary Computation to Improve Mutation Testing</td>
<td>Introduces a mutation testing cost reduction technique that utilizes a genetic algorithm to produce a reduced set of mutants.</td>
</tr>
<tr>
<td>[257] Decreasing the cost of mutation testing with second-order mutants</td>
<td>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.</td>
</tr>
<tr>
<td>Guideline Category</td>
<td>Papers</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[230] Efficient JavaScript Mutation Testing</td>
<td>Proposes mutation operators specific to web applications and a mutation cost reduction technique.</td>
</tr>
<tr>
<td>[171] Efficient Mutation Analysis by Propagating and Partitioning Infected Execution States</td>
<td>Significant efficiency gains in mutation analysis using state infection conditions. The approach is also implemented and empirically evaluated on open source programs.</td>
</tr>
<tr>
<td>[182] Evaluating Mutation Testing Alternatives: A Collateral Experiment</td>
<td>Proposes second order mutation strategies and provides experimental results suggesting the strategies lead to significant cost reductions without considerably reducing test effectiveness.</td>
</tr>
<tr>
<td>[66] Exploring hybrid approach for mutant reduction in software testing</td>
<td>Introduces a hybrid mutation testing cost reduction technique.</td>
</tr>
<tr>
<td>[342] JDAMA: Java database application mutation analyser</td>
<td>Introduces a mutation analyzer useful for evaluating testing techniques applied to java database applications.</td>
</tr>
<tr>
<td>[147] Mutation Operators for Simulink Models</td>
<td>Proposes a set of mutation operators for Simulink models and provides a procedure for mutation testing of Simulink Models.</td>
</tr>
<tr>
<td>[225] Parallel mutation testing</td>
<td>Suggests enhancing the efficiency of mutation testing by utilizing parallel execution.</td>
</tr>
<tr>
<td>[226] Reducing mutation costs through uncovered mutants</td>
<td>Presents a mutation cost reduction technique that leverages the analysis of covered mutants to reduce the number of executions required.</td>
</tr>
<tr>
<td>Guideline Category</td>
<td>Papers</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>[128] <strong>Selective Mutation Testing for Concurrent Code</strong> &quot;Explores selective mutation techniques for concurrent mutation operators&quot; and provides an empirical study evaluating these techniques.</td>
</tr>
<tr>
<td></td>
<td>[344] <strong>Speeding-Up Mutation Testing via Data Compression and State Infection</strong> Speeds up mutation testing by filtering out executions using state infection information and grouping mutants with Formal Concept Analysis.</td>
</tr>
<tr>
<td></td>
<td>[212] <strong>Statistical Investigation on Class Mutation Operators</strong> 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.</td>
</tr>
<tr>
<td></td>
<td>[139] <strong>Topsy-Turvy: A Smarter and Faster Parallelization of Mutation Analysis</strong> Presents a new parallelization technique for mutation analysis.</td>
</tr>
<tr>
<td></td>
<td>[172] <strong>Using Conditional Mutation to Increase the Efficiency of Mutation Analysis</strong> Introduces a new efficiency optimization when performing mutation analysis called conditional mutation.</td>
</tr>
<tr>
<td></td>
<td>[211] <strong>X-MuT: A Tool for the Generation of XSLT Mutants</strong> Introduces mutation operators for the XSLT language along with their implementation in a tool and an evaluation of its effectiveness.</td>
</tr>
<tr>
<td>Mutation Analysis (m)</td>
<td>[86] <strong>A Variability Perspective of Mutation Analysis</strong> Introduces method for modeling mutation operators as a feature diagram for better and faster mutation analysis.</td>
</tr>
<tr>
<td></td>
<td>[87] <strong>Featured Model-based Mutation Analysis</strong> 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.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Category</th>
<th># of Experiments</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis Testing</td>
<td>39</td>
<td>18.22</td>
</tr>
<tr>
<td>Context Justification</td>
<td>98</td>
<td>45.79</td>
</tr>
<tr>
<td>Descriptive Statistics</td>
<td>160</td>
<td>74.77</td>
</tr>
<tr>
<td>Threats to Validity</td>
<td>100</td>
<td>46.73</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Case Study Evaluation Quality</th>
<th># of Case Studies</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research Questions</td>
<td>20</td>
<td>27.40</td>
</tr>
<tr>
<td>Triangulation</td>
<td>42</td>
<td>57.53</td>
</tr>
<tr>
<td>Threats to Validity</td>
<td>27</td>
<td>36.99</td>
</tr>
</tbody>
</table>

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.
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
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.

![Distribution of evaluations by method and dimension](image)

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
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
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.
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

<table>
<thead>
<tr>
<th>Context ID</th>
<th>Evaluation Method</th>
<th>Count</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Experiment</td>
<td>30</td>
<td>[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]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>6</td>
<td>[230], [199], [213], [330], [256], [247]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Experiment</td>
<td>24</td>
<td>[80], [92], [225], [292], [212], [344], [182], [66], [146], [254], [171], [170], [311], [258], [290], [172], [334], [273], [36], [144], [248], [181], [63], [234]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>4</td>
<td>[230], [139], [330], [100]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Experiment</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>2</td>
<td>[199], [330]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>1</td>
<td>[342]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Experiment</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Context ID</td>
<td>Evaluation Method</td>
<td>Count</td>
<td>Papers</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>E</td>
<td>Experiment</td>
<td>71</td>
<td>[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]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>15</td>
<td>[180], [202], [11], [30], [88], [335], [223], [124], [278], [112], [52], [265], [253], [308], [41]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>2</td>
<td>[304], [125]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>5</td>
<td>[25], [289], [60], [23], [134]</td>
</tr>
<tr>
<td>F</td>
<td>Experiment</td>
<td>54</td>
<td>[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]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>6</td>
<td>[202], [139], [124], [253], [35], [41]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Table 4.6: (continued)

<table>
<thead>
<tr>
<th>Context ID</th>
<th>Evaluation Method</th>
<th>Count</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Example</td>
<td>1</td>
<td>[149]</td>
</tr>
<tr>
<td></td>
<td>Experiment</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>2</td>
<td>[52], [268]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>1</td>
<td>[73]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>3</td>
<td>[173], [125], [117]</td>
</tr>
<tr>
<td>H</td>
<td>Experiment</td>
<td>1</td>
<td>[141]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Experiment</td>
<td>22</td>
<td>[207], [208], [129], [114], [55], [210], [324], [206], [314], [98], [176], [140], [311], [26], [126], [281], [93], [282], [190], [234], [318], [123]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>13</td>
<td>[233], [175], [178], [147], [59], [256], [10], [294], [227], [201], [6], [39], [260]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>1</td>
<td>[291]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>1</td>
<td>[26]</td>
</tr>
<tr>
<td>J</td>
<td>Experiment</td>
<td>10</td>
<td>[208], [92], [55], [70], [98], [311], [26], [87], [156], [234]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>7</td>
<td>[233], [175], [10], [294], [227], [6], [187]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>1</td>
<td>[26]</td>
</tr>
<tr>
<td>K</td>
<td>Experiment</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Table 4.6: (continued)

<table>
<thead>
<tr>
<th>Context ID</th>
<th>Evaluation Method</th>
<th>Count</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Analytic</td>
<td>1</td>
<td>[211]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Experiment</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Experiment</td>
<td>46</td>
<td>[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]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>25</td>
<td>[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]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>5</td>
<td>[224], [306], [151], [301], [195]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>3</td>
<td>[197], [28], [60]</td>
</tr>
<tr>
<td>N</td>
<td>Experiment</td>
<td>39</td>
<td>[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]</td>
</tr>
<tr>
<td>Context ID</td>
<td>Evaluation Method</td>
<td>Count</td>
<td>Papers</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>15</td>
<td>[279], [50], [44], [325], [323], [40], [89], [17], [163], [118], [158], [267], [347], [107], [41]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>5</td>
<td>[197], [58], [74], [45], [347]</td>
</tr>
<tr>
<td>O</td>
<td>Experiment</td>
<td>2</td>
<td>[99], [27]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>10</td>
<td>[288], [50], [157], [51], [276], [261], [313], [158], [27], [320]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>5</td>
<td>[229], [238], [177], [339], [306]</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>1</td>
<td>[272]</td>
</tr>
<tr>
<td>P</td>
<td>Experiment</td>
<td>4</td>
<td>[333], [79], [329], [5]</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>3</td>
<td>[279], [313], [242]</td>
</tr>
<tr>
<td></td>
<td>Analytic</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>1</td>
<td>[8]</td>
</tr>
</tbody>
</table>
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 evalu-
ations, 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. \cite{283} and \cite{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...

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


Lajos Cseppent and Zoltan Micskei. “Evaluating code-based test input generator tools”. In: Software Testing, Verification and Reliability 27.6 (2017), e1627.


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


G. Gay. “The Fitness Function for the Job: Search-Based Generation of Test Suites That Detect Real Faults”. In: 2017 IEEE International Conference on


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.


A. Mateen and K. Abbas. “Optimization of model based functional test case generation for android applications”. In: 2017 IEEE International Conference


Abhishek Rathore et al. “Application of Genetic Algorithm and Tabu Search in Software Testing”. In: Proceedings of the Fourth Annual ACM Bangalore Con-


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


