

# The State of the Art in Controlled Experimentation on Testing Techniques and an Evaluation Criteria for Controlled Experiments

Hyunsook Do

Department of Computer Science and Engineering  
University of Nebraska - Lincoln  
Lincoln, NE

January 19, 2006

## Abstract

The importance of empirical studies in software engineering has been recognized by many researchers, but the number of empirical studies is relatively small compared to other disciplines. The quality of individual studies also has not been evaluated well enough to build up knowledge about empirical studies in that field and to provide guidelines for future studies. To investigate the current state of empirical studies, we perform two surveys: a survey of papers that evaluate empirical studies in software engineering, and a survey of papers that provide experimentation guidelines. Based on results from these surveys and additional evaluation criteria identified, we establish criteria for the evaluation of empirical studies on testing techniques. Using this criteria, we evaluate empirical studies of software testing techniques and expose some problems in performing empirical studies. This report intends to provide insights regarding how empirical studies of testing techniques should be performed.

## 1 Introduction

Empirical studies are necessary when researchers or practitioners need to evaluate new techniques or methods, and to make decisions seeking more cost-effective solutions. Wohlin et al. [39] state that the essence of empirical studies is as follows: *“The major reasons for carrying out quantitative empirical studies is the opportunity of getting objective and statistically significant results regarding the understanding, controlling, prediction, and improvement of software development. Empirical studies are important input to the decision-making in an improvement seeking organization.”*

In software engineering, however, empirical studies have been poorly performed compared to other disciplines, such as the natural sciences, applied mathematics, medical research, and psychiatry [17, 38]. Tichy et al. [38] found that over 50% of articles in software engineering published in 1993 lack an empirical evaluation, while the fraction of papers lacking quantitative evaluation in Optical Engineering and Neural Computation is only 15% and 12%, respectively. Zelkowitz et al. [40] and Do et al. [7] also found similar results. About a third of the papers in Zelkowitz et al.’s study (software engineering articles covering 1985, 1990, and 1995) had no experimental validation. Do et al. narrowed their focus to testing techniques, and found that about 53% of the papers did not perform empirical validation (six major software engineering publications covering from 1994 to 2003).

This lack of empirical evidence is likely to hamper the improvement of the software quality, and will probably lead practitioners in an organization to make decisions about adopting new software technologies or methods by intuition and not by scientific evidence [10]. Intuition may work and provide a good starting point, but it must be followed by empirical validation [37]. Performing empirical studies can be very expensive due to the intrinsic complexity of experimentation and the considerable time required. Such empirical studies, however, can produce worthy benefits for practitioners and researchers. That is, the results from experiments can produce far more profits than the cost that a company expends on the experimentation. For example, Tichy [37] illustrated that a five-year lead in software inspections based on in-house experiments at Lucent Technologies is yielding benefits for that company.

Over the decades, the importance of empirical studies in software engineering has been emphasized by several researchers [2, 10, 18, 37, 39], and a study by Do et al. [7] reports that the number of research papers that conduct empirical studies is increasing. This is a positive indication that more software engineers are aware of the importance of performing empirical studies. In that sense, empirical studies in software engineering are maturing in a quantitative way, however, we are not certain whether they are performed in a valid way. Kitchenham et al. [18] point out that even research areas that are considered to be more mature than computer science with respect to empirical studies have problems with performing empirical studies. For example, they show that 40% of papers published in the British Journal of Psychiatry in 1993 had statistical errors. Thus it is important to diagnose the current state of empirical studies to provide future direction in performing such studies.

This report evaluates empirical studies of software testing techniques, focusing on studies of the techniques themselves (we exclude studies with human subjects), provides an overview of those studies, and exposes open problems in both our empirical understanding of techniques, and our understanding of how the techniques should be investigated. Among empirical studies, we focused on the controlled experiment in particular. To achieve these objectives, we required the following preliminary investigations:

1. Survey papers that evaluate empirical studies in software engineering.
2. Survey papers that provide experimentation guidelines.
3. Establish criteria for the evaluation of empirical studies on testing techniques.

Section 2 presents the surveys of empirical studies in software engineering: a survey of reviews of empirical studies and a survey of experimentation guidelines. Section 3 presents criteria for evaluating empirical studies on testing techniques. Section 4 presents evaluation of existing controlled experiments on testing techniques. Section 5 discusses open problems and some implications for empirical studies. Section 6 presents conclusions.

## 2 A Survey of Empirical Studies

Early surveys on experimental computer science by Feldman et al. [9] and McCracken et al. [22] in 1979 present problems in performing experiments in computer science, such as poor support from industry and government, and lack of engagement in experimentation by the computer scientists. An article by the computer science and telecommunications board [36] in 1994 points out that the support for experimental computer science is still not sufficient, and experimental computer scientists may face challenges in building their academic careers.

Software engineering research also faces the same problems [38], and as a consequence we often find limited empirical evidence supporting software methods and techniques. To provide an overview of the current state of empirical studies in software engineering, this section conducts two surveys regarding empirical studies. First, we research studies that performed evaluations of empirical studies of software engineering. Second,

we investigate studies that provide insights on how empirical studies should be performed. The first survey provides us insight on the current status of systematic reviews in software engineering research. The second survey provides guidelines from which to formulate criteria for systematic reviews of papers we consider in this report. After presenting results from the two surveys, we discuss observations and problems identified in those surveys.

## 2.1 A Survey of Reviews of Empirical Studies

While there are not many, there are nonetheless some reviews of empirical studies in software engineering that have been performed. This section presents chronologically five studies which have different focuses and evaluation schemes. The first three studies evaluated general software engineering studies, and the last two studies focused on studies of testing techniques.

These studies observed that several important aspects of empirical research are neglected by researchers: studies often fail to state a precise specification of the problem, give proper interpretation of results, and/or detailed descriptions of data. However, they found a positive aspect to current empirical studies: the number of empirical studies is increasing, and more computer scientists are proving willing to conduct empirical studies.

The detailed descriptions are as follows. For each study, we present the source of the survey (for example, what years of what publications were surveyed by a study), the evaluation criteria that the study used, and results and observations made by the study. The item “results” is presented only if a study provides statistical or numerical data, otherwise it is discussed in the “observations.”

1. Basili et al. [2]: This study provides a framework for experimentation and evaluates existing empirical studies based on the proposed framework. This study focuses on examining the quality of empirical studies, and identifies some problems in experimentation.
  - *Source of survey:* Experimental studies on software engineering between years 1972-1985 (40 papers). No selection criteria are given.
  - *Criteria for evaluation:* The evaluation criteria are given in Section 2.2.
  - *Observations:*
    - There is no “universal model” in software engineering. Experimental studies should consider the vast differences among environments and among people (there are an enormous number of factors that differ across environment).
    - Studies need a precise problem specification.
    - The experimental planning process should include a series of experiments (replication).
    - Some papers do not define their data well enough to enable a comparison of results across many projects and environments.

- The presentation of experimental results should include appropriate qualification (generalization issue) and adequate exposure (graphical representation for data) to support their proper interpretation
2. Tichy et al. [38]: This study surveys research articles in computer science and two other fields, optical engineering and neural computation, and compares them in a quantitative way.
- *Source of survey:* All issues from 1991 to 1993 of the ACM Transactions on Computer Systems (TOCS), all issues from 1992 to 1993, and numbers 1 and 2 in 1994 of the ACM Transactions on Programming Languages and Systems (TOPLAS), all issues in 1993 of the IEEE Transactions on Software Engineering (TSE), all issues in 1993 of the SIGPLAN Conference on Programming Language Design and Implementation (PLDI), a random sample of 50 titles from the set of all works published by the ACM in 1993, all issues in 1993 of Neural Computation (NC), and numbers 1 and 3 from 1994 of Optical Engineering (OE). In total, 403 papers were examined.
  - *Criteria for evaluation:* They classify papers into five categories: formal theory, design & modeling, empirical work, hypothesis testing, and other. Then they assess design & modeling papers based on the amount of space devoted to the description of experimental evaluation. They claim that the amount of space can reflect the quality of the experiment<sup>1</sup>.
  - *Results:* Percentage of papers without empirical evaluation: CS samples - 43%, OE samples - 14%. The number of articles in NC and OE devoting over 20% of their space to experimental evaluation is larger than that in CS samples (67% in OE vs 31% in the random CS sample). Samples related to software engineering (TSE and TOPLAS) are worse than the random CS sample.
  - *Observations:*
    - The results suggest that a large proportion of CS publications may not meet standards long established in the natural and engineering sciences.
    - The youth of the computer science field is not a sufficient explanation for poor standards because the NC field is only six years old.
    - Computer scientists have neglected to develop adequate measuring techniques.
    - Many computer scientists agree that standards need to be raised, but they are reluctant to take the first step: making a tremendous effort to build up measurement guidelines and expertise, not being rewarded (slow careers).
3. Zelkowitz et al. [40]: The study defines 12 validation models for experimentation, classifies articles in software engineering using those models, and analyzes them in a quantitative way.
- *Source of survey:* All issues from the years 1985, 1990, and 1995 of the IEEE Transactions on Software Engineering (TSE), IEEE Software, and International Conference on Software Engineering (ICSE). In total, 562 papers were examined.

---

<sup>1</sup>Please refer section 3.3 in their study for more detailed descriptions.

- *Criteria for evaluation:* The authors develop 12 different experimental models with respect to how data was collected, and group them into three broad categories: observational (project monitoring, case study, assertion, field study), historical (literature search, legacy, lessons learned, static analysis), and controlled (replicated, synthetic, dynamic analysis, simulation).
  - *Results:* Lessons learned and case studies are the most prevalent validation models (about 10% for each). A third of the papers relied on assertion and about a third of the papers had no experimental validation. The percentage of the papers having no experimental validation dropped from 36% in 1985 to 29% in 1990 to 19% in 1995.
  - *Observations:*
    - Too many papers have no experimental validation.
    - Many papers use an informal form of validation (assertion).
    - Researchers often fail to state their goals clearly.
    - Researchers often fail to state how they validate their hypotheses.
    - Experimentation terminologies are used very loosely.
4. Juristo et al. [15]: This study examines papers on testing techniques, and evaluates them using the testing technique family-basis. The evaluation is performed in a qualitative way.
- *Source of survey:* Papers on testing techniques from the past 25 years. No selection criteria are given.
  - *Criteria for evaluation:* The authors measure the maturity level of testing techniques using the four criteria: laboratory study, formal analysis, laboratory replication, and field study. The evaluation is made with respect to the family in which the testing technique is a member: random, functional, control-flow, data-flow, mutation, regression, and improvement.
  - *Observations:*
    - Many studies are based solely on qualitative graph analysis.
    - The response variables examined are of limited use in practice.
    - The artifacts (programs and faults) are not representative.
    - More experiments and replications need to be conducted to generalize results.
5. Do et al. [7]: This study examines articles on empirical studies of testing techniques, and evaluates them for the artifact utilization point of view. The analysis is done in a quantitative way.
- *Source of survey:* All issues and proceedings from two journals and four conferences over the period 1994 to 2003: IEEE Transactions on Software Engineering (TSE), ACM Transactions on Software Engineering and Methodology (TOSEM), ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), ACM/IEEE International Conference on Software Engineering (ICSE), ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE),

and the IEEE International Conference on Software Maintenance (ICSM). In total, 1,995 papers were examined.

- *Criteria for evaluation:* Of the 1,995 full research papers, the authors identify 224 papers on topics involving software testing issues, and select 107 papers that reported results of empirical studies for further analysis. The 107 papers are analyzed with respect to characteristics: the type of empirical study performed (controlled experiment, case study, and example), number of programs used as sources of data, number of versions used as sources of data, whether test suites were utilized, whether fault data were utilized, and whether the study involved artifacts provided by or made available to other researchers.
- *Results:* 11.2% of the papers concerned software testing. Of the testing-related papers, 47.7% reported empirical studies. Of the empirical study papers, 34.5% were controlled experiments, and 56% were case studies. 32.7% utilized data from one program, 44% utilized multiple versions, 35.5% utilized fault data, and 22.4% involved artifact sharing.
- *Observations:*
  - Over 50% of studies do not have empirical validation.
  - The number of controlled experimental studies is small, but the trends are changing: 27.5% in the first five years (1994-1998), and 38.8% in the second five years (1999-2003).
  - Researchers are becoming increasingly willing to conduct controlled experiments, and are increasing the extent to which they utilize shared artifacts.

## 2.2 A Survey of Experimentation Guidelines

There is plenty of literature that provides experimental design and data analysis guidelines that can be applied in science and engineering areas in general [6, 14, 24]. One set of guidelines by Montgomery [24] recommends seven steps for an experiment: (1) recognition of the problem, (2) choice of factors, levels, and ranges, (3) selection of independent/dependent variables, (4) choice of experimental design, (5) performing the experiment, (6) statistical analysis of the data, and (7) conclusions and recommendations. Depending on the literature, these steps break down into more detailed processes, or some steps merge, but their principle guidelines are not different.

Based primarily on these general guidelines, several software engineering researchers have established empirical study guidelines for software engineering and have created important foundations to assist researchers in performing empirical studies [2, 10, 16, 19, 18, 26, 27, 28, 29, 30, 31, 39]. Among those in our survey, we focused on guidelines for performing experiments, and present the results of the survey.

Table 1 presents experimentation steps defined in five different sources [2, 16, 18, 28, 39]. They bear similarity to those from the general guidelines mentioned in the beginning of this section. They use slightly different terminologies and groupings, but deliver conceptually similar processes. In particular, Basili et

al. and Wohlin et al. describe common components, with the exception of presentation and packaging in Wohlin et al.'s guidelines. Juristo et al. consider some factors that are specifically relevant to an experiment in software engineering, but we omit them in this survey because their considerations are focused on human subjects.

Detailed descriptions of Table 1 are as follows.

1. Basili et al. [2]: This study describes a framework for experimentation in software engineering, which defines four steps.
  - Definition: Define six parts - motivation of the study, object (the primary entity examined in a study), purpose, perspective (developer or customer), domain (programs or human subjects), and scope (single/multi project, replicated project, or blocked subject project).
  - Planning: Contain three parts - design (factorial or block design), criteria (direct/indirect reflection of cost/quality: cost, reliability, programmer understanding, etc.), and measurement (capture the cost/quality aspects: fault detection cost, fault detection effectiveness, etc.).
  - Operation: Contain three parts - preparation (subject expertise, object instrumentation, etc.), execution (collect and validate data), and analysis (check assumptions before applying statistical tests)
  - Interpretation: Contain three parts - interpretation context (statistical results should be interpreted in the context of the study), extrapolation (representativeness of the study), and impact.
2. Pfleeger [28]: This study defines six steps for experimentation.
  - Conception: Define goals of the study, and state clearly the objective of the study (should be stated as a question you want answered).
  - Design: Creates a hypothesis, and generates a formal design to test the hypothesis (identify subject/object, treatment, and independent/dependent variables )
  - Preparation: Setup environment for execution, such as a tool purchase, or hardware configuration.
  - Execution: Collect data by applying the treatment to the experimental subjects/objects.
  - Analysis: Review all measurements, and apply statistical analyses.
  - Dissemination and decision-making: Draw a conclusion based on the results from the analysis phase.
3. Wohlin et al. [39]: Authors define five steps for experimentation. The first four steps are similar to those defined by Basili et al., but provide more detailed descriptions.
  - Definition: Determine why the experiment is conducted. Components to be considered are the object of study, purpose, quality focus (the primary effect under the study, such as effectiveness, or cost), perspective (whose viewpoint, i.e., developer or customer), and context (environment in which the experiment is run).

Table 1: The Experiment Processes.

Steps	Basili et al. [2]	Pfleeger [28]	Wohlin et al. [39]	Juristo et al. [16]	Kitchenham et al. [18]
1	<p>Definition</p> <ul style="list-style-type: none"> <li>- motivation</li> <li>- object</li> <li>- purpose</li> <li>- perspective</li> <li>- domain</li> <li>- scope</li> </ul>	<p>Conception</p> <ul style="list-style-type: none"> <li>- goals</li> <li>- objective</li> </ul>	<p>Definition</p> <ul style="list-style-type: none"> <li>- object of study</li> <li>- purpose</li> <li>- quality focus</li> <li>- perspective</li> <li>- context</li> </ul>	<p>Definition</p> <ul style="list-style-type: none"> <li>- objectives</li> <li>- hypothesis</li> </ul>	<p>Experimental context</p> <ul style="list-style-type: none"> <li>- background info</li> <li>- research hypothesis</li> </ul>
2	<p>Planning</p> <ul style="list-style-type: none"> <li>- design</li> <li>- criteria</li> <li>- measurement</li> </ul>	<p>Design</p> <ul style="list-style-type: none"> <li>- hypothesis formulation</li> <li>- subject/object selection</li> <li>- treatment identification</li> <li>- variables selection</li> </ul>	<p>Planning</p> <ul style="list-style-type: none"> <li>- context selection</li> <li>- hypothesis formulation</li> <li>- variables selection</li> <li>- subject selection</li> <li>- experiment design</li> <li>- instrumentation</li> <li>- validity evaluation</li> </ul>	<p>Design</p> <ul style="list-style-type: none"> <li>- define the experimental unit</li> <li>- subject/object selection</li> <li>- variables selection</li> <li>- define the parameter</li> </ul>	<p>Design</p> <ul style="list-style-type: none"> <li>- identify the population</li> <li>- define sampling process</li> <li>- define the treatment allocation</li> <li>- define the experimental unit</li> </ul>
3	<p>Operation</p> <ul style="list-style-type: none"> <li>- preparation</li> <li>- execution</li> <li>- analysis</li> </ul>	<p>Preparation</p> <ul style="list-style-type: none"> <li>- environment setup</li> </ul> <p>Execution</p> <ul style="list-style-type: none"> <li>- measure</li> </ul>	<p>Operation</p> <ul style="list-style-type: none"> <li>- preparation</li> <li>- execution</li> <li>- data validation</li> </ul>	<p>Execution</p> <ul style="list-style-type: none"> <li>- collect data</li> </ul>	<p>Execution and data collection</p> <ul style="list-style-type: none"> <li>- define all software measures</li> </ul>
4	<p>Interpretation</p> <ul style="list-style-type: none"> <li>- interpretation context</li> <li>- extrapolation</li> <li>- impact</li> </ul>	<p>Analysis</p> <ul style="list-style-type: none"> <li>- review measurements</li> <li>- statistical procedures</li> </ul>	<p>Analysis and Interpretation</p> <ul style="list-style-type: none"> <li>- descriptive statistics</li> <li>- data set reduction</li> <li>- hypothesis testing</li> </ul>	<p>Analysis</p> <ul style="list-style-type: none"> <li>- examination of data</li> <li>- statistical inference</li> </ul>	<p>Analysis</p> <ul style="list-style-type: none"> <li>- graphical examination of data</li> <li>- check the assumptions of the test</li> </ul> <p>Presentation of data</p> <ul style="list-style-type: none"> <li>- describe all statistical procedures used</li> <li>- present the raw data</li> <li>- provide appropriate descriptive statistics</li> <li>- make appropriate use of graphics</li> </ul> <p>Interpretation</p> <ul style="list-style-type: none"> <li>- define the population</li> <li>- statistical and practical importance</li> <li>- define the type of study</li> <li>- specify any limitations of the study</li> </ul>
5		<p>Dissemination and discussion-making</p> <ul style="list-style-type: none"> <li>- draw conclusions</li> </ul>	<p>Presentation and Package</p> <ul style="list-style-type: none"> <li>- present the findings</li> </ul>		

- Planning: Focus on how the experiment is conducted.  
The components are context selection (off-line vs on-line, student vs professional, toy vs real problems, and specific vs general), hypothesis formulation, variable selection (independent and dependent variables), selection of subjects (connected to the generalization of results from the experiment), experiment design (describes how the tests are organized and run), instrumentation (environment setup, such as fault seeding in a program, or interview preparation), validity evaluation (threats to internal, external, construct, and conclusion validity).
  - Operation: The components are preparation, execution, and data validation. The experiment is carried out to collect the data that should be analyzed. Collected data should be validated by the experimenter.
  - Analysis and interpretation: The components are descriptive statistics (gives understanding of data set distribution), data set reduction (identify and address properly), and hypothesis testing (find appropriate statistical test procedure based on data distribution, draw conclusions, and discuss statistical/practical significance of results).
  - Presentation and package: Define an experiment report outline: introduction, problem statement, experiment planning, operation, data analysis, interpretation, discussion, and conclusions.
4. Juristo et al. [16]: Authors define four experimentation steps focusing on statistical data analysis, considering experimentation in software engineering. They provide software engineering specific factors we need to consider when we perform an experiment, but most of those factors are related to the human subject specific factors, such as learning, experience, boredom, enthusiasm effect, etc.
- Definition: Define objectives of the study, and formulate hypothesis for the experiment.
  - Design: Components required in the design phase are experimental units, subjects/objects, independent/dependent variables, and parameters (the characteristics of software that is invariable throughout the experiment).
  - Execution: Run experiment and collect data.
  - Analysis: Examination of data is recommended before applying statistical analysis. The statistical inference is made after statistical analysis.
5. Kitchenham et al. [18]: This study proposes a set of guidelines for empirical studies in software engineering. In the following description, we omitted human-subject specific guidelines.
- Experimental context: Background and related information for the study needs to be presented. The research hypotheses are defined.
  - Experimental design: Identify the population, and define selection process for subjects/objects, process for assigning treatments, and experimental units.
  - Execution of the experiment and data collection: Define all software measures (entity, attribute, unit, and counting rules).

- Analysis: Graphical examinations are recommended before undertaking detail analysis, and an assumption check should be performed on the test procedure.
- Presentation of results: Presentation<sup>2</sup> of results should be detailed enough to allow replications of the study. Depending on statistical packages, one might get slightly different results, so it is a good idea to specify what package has been used in the study. Provide appropriate graphical representations for data comprehension, and appropriate descriptive statistics.
- Interpretation of results: Define the population and the type of study. The statistical and practical importances might be different, so it is important to interpret results from both perspectives. Any limitations on the study need to be specified.

## 2.3 Observations and Some Problem Areas

In sections 2.1 and 2.2, we presented the current state of reviews of empirical studies and experimentation guidelines. Through these surveys we made the following observations and identified some problems that should be improved.

### 1. *Small number of systematic reviews:*

During the survey process, we found that there is a limited number of papers that conduct reviews of empirical studies. Moreover, some reviews do not have systematic evaluation criteria or a definition of the population (such as how they selected papers for reviews). For example, Basili et al. and Juristo et al. do not provide information on how they select papers for reviews. In the absence of this information, it is difficult to draw correct inferences from the results of the study. Kitchenham et al. [17] urge that software engineers need to perform more systematic reviews to establish evidence-based software engineering. They recommend that systematic reviews should use appropriate search methodologies, and justify the method of search, such as the choice of journals and the selection of papers.

### 2. *Few reviews of quality of empirical studies:*

Many reviews focus on the quantitative aspects, for example, how many papers do not have any empirical validation, how many papers utilize fault data, etc. The quantitative analysis is an important activity that identifies a general (or specific) trends in attributes we are interested in, but we also need qualitative analysis that provides insights on how well individual studies were performed. Juristo et al.'s review [16] provides the quality point of view of testing techniques, which they describe as the maturity level of a technique. We need more of this type of review to build up knowledge about empirical studies in a specific area that researchers are interested in, and to provide guidelines for studying that area in future research.

---

<sup>2</sup>The usage of presentation here is different from that in Wohlin's study, which referred to documentation. Here, presentation means presenting analyzed data.

### 3. *No standard experimentation guidelines in software engineering:*

During the survey of experimentation guidelines, we found that some terminologies are not standardized, so researchers tend to use terminologies familiar to them. For example, Basili et al. use “extrapolation”, while some others use “generalization”, which is more popularly used in the experimentation community. The “state variable” (“response variable”) and “independent variable” (“dependent variable”) are used interchangeably. Juristo et al. are using “parameter” which is defined as “any characteristic of software project that is to be invariable throughout the experimentation.” It appears to be an environment or setting for experiment, but use of the terminology is not clear.

Some guidelines explicitly recommend graphical examination of data and checking of assumptions before applying statistics procedures, but some make no mention whatsoever. Interpretation of test results is the most important part of the experiment since it answers specific research questions, and often research in software engineering carries industrial interests. It is thus important to report statistical and practical significance of experiment results. Most of the guidelines surveyed here miss this point.

According to the study by Kitchenham et al. [17], medical research has standard guidelines for conducting and reporting an experiment, which is called the “CONSORT statement”. Thus all the most important medical journals follow those guidelines. Software engineers also need this sort of standard set of guidelines for conducting empirical studies.

## **3 Criteria for the Evaluation of Empirical Studies on Testing Techniques**

A software system is evolving continuously to improve its functionality and meet various demands from users and the software development environment. As a result, developers or testers need to test a system whenever changes are made. Testing and maintenance, however, are expensive, particularly when a system grows large. Thus many techniques and tools are developed to be cost-effective and evaluated through empirical studies. However, in most cases, the studies are independent, and there is no aggregation of efforts, replication, or meta-analysis from similar studies, except with multiple studies conducted by the same researchers. A study by Brooks et al. [5] in 1996 found that external replication in software engineering is very rare, and reviews in the area of software testing and maintenance, where empirical work is relatively commonplace, make no mention of external replication work. Possible reasons for this phenomenon are a lack of detailed descriptions of experiments, and more fundamentally a lack of standard experimentation guidelines in software engineering as described in Section 2.3.

As a first step, to address these problems, this section presents criteria for the evaluation of empirical studies of testing techniques, primarily based on the survey in Section 2, with the additional evaluation criteria for testing techniques we identified.

1. *The objectives of the study:*

One of the problems observed in reviews of empirical studies is that researchers often fail to state a precise problem specification. This criterion considers whether the objectives of the study are stated clearly. As recommended by most of the experiment guidelines in Section 2.2, for a formal experiment, specific hypotheses or research questions should be formulated from the objectives of the study. However, researchers often neglect to formulate a specific hypothesis that makes clearer the detailed interests of the study.

2. *The experimental design:*

We adopt a part of design guidelines from Kitchenham et al. [18]:

- Define the sampling process of objects: In practice, it is hard to obtain software programs by random sampling. Open source software hosts such as SourceForge and Apache Jakarta, however, can be one of the pools from which we can obtain random samples in a limited sense. Unlike human subjects, which are typically available for one time experimental use, researchers who perform experiments with non-human objects often share object programs for their experiments [7]. In either case, a study needs to specify how objects obtained.
- Define treatment: Defining treatment helps to understand the cause and effect relationship clearly.
- Define the process for treatment assignment: The treatment allocation to objects should be unbiased. The unbiased allocation is unlikely to happen when using object programs instead of human subjects, but still a process should be specified for treatment assignment to make the experiment process clear.

In addition to these, all detailed attributes of objects should be specified. For example, studies on testing techniques possibly carry the following attributes: lines of code (or number of classes), the number of functions (or methods), the number of faults in a program, the type of faults (real, seeded, and mutated faults), the number of versions of a program, the test suite size, the type of test suite (specification-based, code-based, coverage-based, etc), the number of mutants, and any other characteristics relevant to the experiment. This information helps to understand the experiment results and the complexity of the experiment. Any detailed descriptions of the experiment design improve the chances of replication studies.

3. *Study setting:*

Depending on software tools and hardware devices, experiment results can be slightly different. To provide a transparent view of the collected data, environment setups that affect results should be specified. For example, when you measure an execution time of a program by applying different techniques under a specific software tool, you need to provide a specification for the machine and the name (source) of the software tool, and when important, a description of the software tool. This

criterion is also very important if researchers want to replicate the study.

4. *Data collection (measure):*

When we collect data, we should be certain that a measure is valid for both input and output, and detailed descriptions of data collection procedures are helpful for understanding the experiment and conducting replications. The unit of collected data should be specified clearly. Sometimes researchers forget to specify a unit of measure. For instance, Andrews et al. [1] measure execution time using a specific tool, but no unit of time is specified. It can be seconds or hours, which represent very different scales, thus readers cannot interpret the the results correctly.

For testing techniques, possible measures are: fault detection rate, test suite reduction rate, run time for testing tools/methods, coverage rate, etc.

5. *Data examination and analysis:*

Before performing a formal data analysis, it is important to examine collected data through graphical presentations. By doing this, we can understand how data is distributed (normal or non-normal), whether data includes outliers (and if so, how extreme), and whether the variabilities between data sets to be compared are severe. This gives us an initial idea about what type of statistical procedures should be applied to the collected data, and serves as a check of assumptions before applying statistical procedures.

Depending on the distribution of data, its size, and the variability between data groups to be compared, the choice between parametric and non-parametric analysis is made. Parametric tests are more powerful than non-parametric tests, but we should make sure that the data meets the assumptions required for parametric tests. Since different statistics packages may give slightly different results, specifying what statistics package the study used is important.

6. *Interpretation:*

The most important part of the experiment processes is interpretation of results by connection to the research questions. A study may find statistical significance in the results, but there may no practical significance, and vice versa. Montgomery [24] illustrates this point well using the following example: an engineer may determine that a modification to an automobile fuel injection system may produce a true mean improvement in gasoline mileage of 0.1 mi/gal. This is a statistically significant result. However if the cost of the modification is \$1000, then the 0.1 mi/gal difference is probably too small to be of any practical value.

7. *Limit of the study:*

Researchers should report any limitations of the study, such as threats to the validity of the experiment. There are four types of threats to validity [39]: internal, external, construct, and conclusion validity. Researchers need to discuss at least internal and external validity [18].

- Internal validity: Internal validity concerns confounding factors or biases that can affect the independent variables with respect to causality.
- External validity: External validity concerns the generalization of results.
- Construct validity: Construct validity concerns whether the experiment setting actually reflects the construct under study.
- Conclusion validity: Conclusion validity concerns issues that affect the ability to draw the correct conclusions about relations between the treatment and the outcome of an experiment.

#### 8. *Replicability:*

Replication is very important from a scientific point of view. Replication studies can confirm the results of an original study, and so can build knowledge of the study in a significant way, contributing to the generalization of results. According to a study by Zelkowitz et al. [40], too many experiments have an advocacy problem. A third of papers from their survey relied on advocated study. This problem can be partially solved through external replication.<sup>3</sup>

The replicability of the study is closely related to how detailed the study presents the experiment design and data collection/measure processes. In particular, for the independent researchers (external replication), the replicability is critical. Brooks et al. [5] perform several replicated studies and find that even in a doctoral thesis, several details were absent from the experiment report, which introduced uncertainty to their replication study and sources of variability between the experiments.

To judge a replicability of studies by other researchers, we examine four components (objects (details on attributes of objects and source of objects), treatments (description of techniques used), study setting, and data collection) for each study, and whether the study provides detailed information about those components. We categorized them into three levels: highly, moderately, and poorly replicable.

#### 9. *Long-term and lifetime view:*

As mentioned in Fenton et al.'s study [10], the long-term view could lead to conclusions very different from the short-term view. Their study shows one example from the NASA Goddard Software Engineering Conference, which investigated the benefits of using Ada instead of Fortran. It was reported that Ada was not effective in the first set of projects, but after three major Ada developments, results showed that there were significant benefits to using Ada instead of Fortran.

A long-term assessment is more important for studies on testing techniques because testing is tightly related to system evolution. There has been considerable research on testing and regression testing. Only recently have people started to evaluate the techniques empirically. Typically these evaluations treat testing and regression testing as a one-time process, whereas what we are really interested in is

---

<sup>3</sup>There are two types of replications: internal and external replications [5, 16]. The internal replications are run within the experiment itself in order to establish the statistical validity of the experiment. The external replications are run by independent researchers in order to build confidence in the results of the experiment and check on the findings of other researchers.

testing the cost-benefits across system’s entire lifetime. Ignoring system lifetime may lead to incorrect valuation of techniques. For example, a testing technique  $A$  may be beneficial compared to a comparable technique  $B$  for a base version of a system, but when the system evolves, the conclusion may not hold. Thus we consider whether studies treat testing a system as a lifetime process (long-term view) or a one-time process (short-term view).

10. *Industrial significance:*

Often studies draw myopic conclusions, and thus they miss important factors when they evaluate techniques. Their findings can be very significant and useful to industry, so researchers should link their findings to industrial contexts and provide a prospective view on potential benefits resulting from their techniques. For example, savings as indicated by some measure on a small program may not be very beneficial, but for a large commercial program, it may be a very significant savings. Software engineering research is tightly related to the software industry as medical research is related to the pharmaceutical industry, and it is considered to play a pioneering role in testing new ideas prior to applying them in industry. Thus correlating results from the study with industrial contexts and suggesting potential benefits of the results are a necessary and important part of the process for a better software engineering future.

## 4 The State of the Art in Testing Techniques

In this section, we analyze a set of ACM Transactions on Software Engineering and Methodology (TOSEM) and IEEE Transactions on Software Engineering (TSE) papers reviewed by Do et al. [7], focusing on controlled experiments. These papers are from all issues from these venues, in the period 1994 to 2003. We analyze 15 papers in total, and an analysis is based on criteria that is defined in the previous section. In the analysis process, we focus only on the experimental point of view, so discussion regarding the context of a study is outside the scope of this study.

Table 2 summarizes the results of our analysis. The description of each column is as follows:

- Papers: Lists the order of the paper’s appearance in the detailed descriptions (1, 2, ..., or 15).
- Research Questions: Specifies whether a study states specific research questions (yes or no).
- Design: Specifies whether a study describes all necessary components in the experimental design, such as objects, treatments, and treatment assignment (yes or no).
- Study Setting: Specifies whether a study describes the environmental setting or tools that are necessary to all objects and treatments, such as machine specifications (yes or no). When a study contains multiple settings and describes some part of them, we specify “yes/no.”
- Data Collection: Specifies whether a study describes data collection and measure processes (yes or no)
- Analysis: specifies whether a study performs statistical analysis (yes or no).

Table 2: Summary of Testing Techniques.

Papers	Research Questions	Design	Study Setting	Data Collection	Analysis	Interpretation	Limit	Replicability	Long-term View	Industrial Practice
1	yes	yes	yes	yes	no	yes/yes	I/E/C	moderate	no	no
2	no	yes	yes	yes	no	yes/yes	I/E/C	high	no	yes
3	no	yes	yes/no	yes	yes	yes/yes	-	moderate	no	no
4	yes	yes	yes	yes	yes	yes/yes	I/E/C	high	no	yes
5	no	yes	yes	yes	no	yes/no	-	moderate	no	no
6	yes	yes	yes	yes	no	yes/yes	I/E/C	high	no	no
7	yes	yes	no	yes	no	yes/yes	I/E/C	high	no	no
8	yes	yes	yes	no	no	yes/no	-	poor	no	no
9	yes	yes	no	yes	yes	yes/yes	E	high	no	no
10	no	yes	no	no	no	yes/no	I/E	poor	no	no
11	yes	yes	yes	yes	no	yes/yes	E	high	no	no
12	yes	yes	no	yes	yes	yes/no	I/E/C	high	no	no
13	no	yes	yes	yes	no	yes/no	I/E/C	high	no	no
14	no	yes	yes	yes	no	yes/yes	E	high	no	no
15	yes	yes	yes	yes	yes	yes/yes	I/E/C	high	no	no

Table 3: Replicability Checklist.

Papers	Object Description	Object Source	Technique Description	Study Setting	Data Collection Details	Unit of Measure	Replicability
1	yes	yes	yes	yes	yes	no	moderate
2	yes	yes	yes	yes	yes	yes	high
3	yes	yes	yes	yes/no	yes	yes	moderate
4	yes	yes	yes	yes	yes	yes	high
5	yes	yes	no	yes	yes	yes	moderate
6	yes	yes	yes	yes	yes	yes	high
7	yes	yes	yes	no	yes	yes	high
8	no	no	yes	yes	no	yes	poor
9	yes	yes	yes	no	yes	yes	high
10	yes	no	yes	no	no	yes	poor
11	yes	no	yes	yes	yes	yes	high
12	yes	yes	yes	no	yes	yes	high
13	yes	yes	yes	yes	yes	yes	high
14	yes	yes	yes	no	yes	yes	high
15	yes	yes	yes	yes	yes	yes	high

- Interpretation: Specifies whether a study interprets results with respect to the research questions and considers practical issues related to results ((yes or no)/(yes or no) - first one indicates relevant interpretations are made, and second one indicates practical issues are discussed.).
- Limit: Specifies whether a study describes the limitations of the study, such as threats to internal, external, construct, and conclusion validity. Since none of studies we consider address threats to conclusion validity, we denote this column using the validity initials (I/E/C).
- Replicability: Specifies the degree of replicability of a study (high, moderate, or poor). Decisions are

made from Table 3, considering whether the following elements are addressed in the study: object description details, source of object, technique implementation details, study setting, data collection details, and unit of measure. Based on this information, the degree of replicability is decided. Two exceptions are made for the first and third studies. These studies satisfy most of the requirements, but in each one component poses a high risk to replication.<sup>4</sup>

- Long-term View: Specifies whether a study considers a long-term view of the lifetime of techniques (yes or no).
- Industrial Practice: Specifies whether a study considers results in any industrial contexts (yes or no).

The detailed descriptions for each study are as follows: We describe each study in alphabetical order by author’s name. Some studies do not provide some of the criteria we consider. In that case, we omit them from the descriptions. Some studies define multiple objectives or research questions. In that case, we present only one for brevity. We omit the details on “Interpretation” for brevity; instead we describe whether the results are interpreted with respect to the research questions or objectives.

1. General test result checking with log file analysis (Andrews et al. [1], TSE 2003).

- The objectives of the study: This study states the motivation of the study and two specific research questions.  
Motivation: What is the most useful application of formal and semi-formal methods to the process of unit testing?  
Research question: How efficient is unit testing with test result checking done by oracles written in LFAL?
- The experimental design:  
Object: Four programs (C programs, LOC, number of conditional statements, number of mutants), implemented by authors.  
Treatment: Two data selection methods (“cust” and “rand”), and two test result checking methods (“lfa” and “ref”).  
Treatment assignment: All combinations of treatments are applied to all programs.
- Study setting: 300-MHz Sun Ultra 10 running SunOS 5.7 and a mutation tool (detailed description is not given).
- Data collection (measure): The running time from ten replications (the unit is not specified), the number of mutants killed, and the ratio of running time and mutants killed between every pair of treatment combinations.
- Data analysis: No statistical analyses or graphical data examinations are performed.
- Interpretation: Relevant interpretation is made with respect to the research questions. Practical issues are discussed.
- Limitations of study: Threats to internal, external, and construct validity are discussed.
- Replicability: Moderate.

---

<sup>4</sup>The first study uses a mutation tool for the program instrumentation, but the tool is not available publicly, and detailed descriptions are omitted. The third study uses a reverse engineering tool to determine the attributes of objects. Hence their results are dependent on that particular reverse engineering tool.

2. A Comparative study of coarse- and fine-grained safe regression test-selection techniques (Bible et al. [3], TOSEM 2001)

- The objectives of the study: This study states objectives, but does not give specific research questions.  
Objectives: Empirically investigate the relative cost-benefit tradeoffs involved in utilizing TestTube or DejaVu for regression test selection.
- The experimental design:  
Object: Seven Siemens programs, *space*, and *player* (C programs, LOC, number of functions, number of versions, test pool size, and test suite average size), from multiple outside sources.  
Treatment: Two test selection methods (TestTube and DejaVu) and a “retest-all” method.  
Treatment assignment: All treatments are applied to all programs.
- Study setting: Sun Microsystems SPARCstation 1+ with 24MB of physical memory for the execution runs on the Siemens programs and *player*, and Sun Microsystems UltraSparc 1 with 50MB of physical memory for the execution runs on *space*.
- Data collection (measure): Precision (percentage of test cases selected by each technique) and cost (time, in seconds)
- Data analysis: Graphical examinations of data and statistical data are given, but no statistical analyses are performed.
- Interpretation: Relevant interpretation with respect to the research questions is made. Practical issues are discussed.
- Limitations of study: Threats to internal, external, and construct validity are discussed.
- Replicability: High.
- Industrial practice: A cost model for comparing regression techniques is discussed in terms of the product development phases.

3. An investigation of graph-based class integration test order strategies (Briand et al. [4], TSE 2003).

- The objectives of the study: This study states a broad objective. No detailed research question is given.  
Objective: Empirically evaluate three graph-based integration testing strategies.
- The experimental design:  
Object: Four Java programs (LOC, number of classes, relationships in UML, and cycles), from class project and multiple outside sources.  
Treatment: Three graph-based integration testing strategies (Briand et al., Le Traon et al. and Tai et al.).  
Treatment assignment: All treatments are applied to all programs.
- Study setting: Personal computer (500MHz, 128MB), and a reverse engineering tool (Java source to Object Relation Diagram).
- Data collection (measure): The number of stubs, attribute coupling, and method coupling.
- Data analysis: Graphical examination of data and check of assumptions before applying a non-parametric test. Wilcoxon Rank-Sum nonparametric test is performed.
- Interpretation: Relevant interpretation is made with respect to the research questions. Practical issues are discussed.
- Replicability: Moderate.

4. Test case prioritization: A family of empirical studies (Elbaum et al. [8], TSE 2002).
  - The objectives of the study: The study states specific research questions.  
Research questions: Can version-specific test case prioritization improve the rate of fault detection of test suites?
  - The experimental design:  
Object: Seven Siemens programs, *space*, *grep*, *flex* and *QTB* (C programs, LOC, number of versions, test pool size, and test suite average size), from multiple outside sources.  
Treatment: 18 prioritization techniques.  
Treatment assignment: All treatments are applied to all programs.
  - Study setting: Uses Aristotle program analysis system, Proteum mutation system, and Unix diff utility.
  - Data collection (measure): APFD value (the weighted average of the percentage of faults detected, formula is given).
  - Data analysis: Graphical examination of data is performed. Assumption checks for ANOVA test are not made. ANOVA and Bonferroni analysis procedures are applied and statistical data is shown.
  - Interpretation: Relevant interpretation is made with respect to the research questions. Practical issues are discussed.
  - Limitations of study: Threats to internal, external, and construct validity are discussed.
  - Replicability: High.
  - Industrial practice: Cost-benefit analysis is given in an industrial context.
5. The chaining approach for software test data generation (Ferguson et al. [11], TOSEM 1996).
  - The objectives of the study: The study states a goal, but no specific research questions are given.  
Goal: Compare the following methods of test data generation: random, path-oriented, goal-oriented, and the chaining approach (level 1 and level 3).
  - The experimental design:  
Object: 11 Pascal programs (LOC, number of nodes, and number of branches), implemented by authors.  
Treatment: four test data generation methods.  
Treatment assignment: All treatments are applied to all programs.
  - Study setting: TESTGEN test data generation system and a PC with a 60MHz Pentium processor.
  - Data collection (measure): Success rate (a formula is given), coverage (the percentage of nodes for which at least one try was successful in finding input data during the experiment), average/maximum time of a successful search (unit is seconds), and average/maximum time of an unsuccessful search (unit is seconds).
  - Data analysis: The raw data is presented. No graphical examination and no statistical analysis is performed.
  - Interpretation: Relevant interpretation is made with respect to the research goal, but detailed discussion is not provided.
  - Replicability: Moderate.
6. An empirical study of regression test selection techniques (Graves et al. [12], TOSEM 2001)

- The objectives of the study: The study states an objective and gives specific research questions.  
Goal: Examine the relative costs and benefits of several regression test selection techniques.  
Research questions: How do techniques differ in terms of their ability to reduce regression testing cost?
  - The experimental design:  
Object: Seven Siemens programs, *space*, and *player* (C programs, LOC, number of functions, number of versions, test pool size, and average test suite size), from multiple outside sources.  
Treatment: Seven test selection techniques (safe, dataflow, minimization, random(25), random(50), random(75), and retest-all).  
Treatment assignment: All treatments are applied to all programs.
  - Study setting: Uses Aristotle program analysis system.
  - Data collection (measure): Average reduction in test suite size and fault detection effectiveness.
  - Data analysis: Graphical examination of data is performed. No statistical tests are applied. Cost-benefit analysis between techniques is given.
  - Interpretation: Experiment results are interpreted in the context of the research questions. Practical issues are discussed.
  - Limitations of study: Threats to internal, external, and construct validity are discussed.
  - Replicability: High.
7. Empirical studies of a prediction model for regression test selection (Harrold et al. [13], TSE 2001).
- The objectives of the study: The study states a hypothesis.  
Hypothesis: Given a system under test  $P$ , a regression test suite  $T$  for  $P$ , and a selective regression testing method  $M$ , it is possible to use information about the coverage relation  $covers_M$  induced by  $M$  over  $T$  and the entities of  $P$  to predict whether or not  $M$  will be cost-effective for regression testing future versions of  $P$ .
  - The experimental design:  
Object: Seven Siemens programs (C programs, LOC, number of functions, number of versions, test pool size, and average test suite size), from the Siemens Corporate Research.  
Treatment: Two test selection methods (DejaVu and TestTube).  
Treatment assignment: All treatments are applied to all programs.
  - Data collection (measure): The cost of applying a testing method and the cost of executing programs on the test cases selected by that testing method.
  - Data analysis: Graphical data regarding the deviation between predicted and actual test selection percentages is shown. No statistical analysis is performed.
  - Interpretation: Test results are interpreted in the context of the hypothesis. Practical aspects are discussed.
  - Limitations of study: Threats to internal, external, and construct validity are discussed.
  - Replicability: High.
8. Incremental integration testing of concurrent programs (Koppol et al. [20], TSE 2002).
- The objectives of the study: The study states specific research questions.  
Research questions : How much state space reduction does the algorithm “Reduce” achieve during incremental testing?

- The experimental design:
    - Object: Three CCS programs (no specific information regarding attributes is given).
    - Treatment: Two program slice methods (T-synchronizations and L-synchronizations).
    - Treatment assignment: All treatments are applied to all programs.
  - Study setting: 300MHz Pentium computer with 64MB RAM.
  - Data collection (measure): Ureduction rate (percentage) - no detailed description on how they collected this data is given. The average sizes of the program slices using two coverage criteria. Test adequacy - no definition is given.
  - Data analysis: Statistical data is provided, but no formal statistical analyses are performed.
  - Interpretation: All questions are answered, but no further discussion is provided.
  - Replicability: Poor. to support replication.
9. Using spanning sets for coverage testing (Marre et al. [21], TSE 2003)
- The objectives of the study: The study states a specific research question.
    - Research question: Investigate whether using spanning sets to make testing cheaper might produce, as a negative effect, a significant loss in the fault-detection effectiveness.
  - The experimental design:
    - Object: Seven Siemens programs (C programs, LOC, number of faulty versions, and number of test cases), from the Aristotle Research Group.
    - Treatment: Two test suite selection methods to reach a certain coverage (using spanning sets and random) and two coverage criteria (all-branches and all-uses).
    - Treatment assignment: All-branches and two test selection methods are applied to all programs, and all-uses and two test selection methods are applied to five programs.
  - Data collection (measure): Test size to obtain a certain coverage and number of detected faults. Raw data is presented.
  - Data analysis: Graphical examination of data is not performed. Check of assumptions for F-test is not performed. F-test results are reported, but a confidence value for F-test is not provided (such as the p-value).
  - Interpretation: Test results are interpreted in the context of the research question. Practical aspects are discussed.
  - Limitations of study: Threats to external validity are discussed.
  - Replicability: High.
10. Generation software test data by evolution (Michael et al. [23], TSE 2001).
- The objectives of the study: The study states some objectives, but does not provide specific research questions.
    - Objectives: Investigate how the GAs and random test generation perform on increasingly complex synthetic programs.
  - The experimental design:
    - Object: Ten C programs (LOC), generated by a CASE tool (one program).
    - Treatment: Four test generation methods.
    - Treatment assignment: All treatments are applied to all programs.
  - Data collection (measure): The highest percentage of test requirements satisfied by any single

run of the test generator among a series of five such runs. Best, worst, and pointwise mean performance over 10 separate attempts by each system to achieve complete condition-decision coverage. Raw data is shown in the table. Plots for the data are shown.

- Data analysis: No statistical tests are performed. Comparison between methods is made based on raw data.
- Interpretation: The hypotheses are not clear enough to understand the objective of the study, but discussion of the results makes it clear. No practical aspects are discussed.
- Limitations of study: Threats to internal and external validity are discussed.
- Replicability: Poor.

11. An experimental determination of sufficient mutant operators (Offutt et al. [25], TOSEM 1996).

- The objectives of the study: The study states a specific hypothesis.  
Hypothesis: Test sets that kill all mutants under selective mutation will have a high coverage level (high mutation score) under nonselective mutation.
- The experimental design:  
Object: Ten Fortran-77 programs (the number of statements, the number of mutants).  
Treatment: Four selective mutant sets for each program.  
Treatment assignment: All treatments are applied to all programs.
- Study setting: Uses Godzilla test data generator.
- Data collection (measure): Mutation score (averaged over the five test sets) and the savings obtained by selective mutation in terms of the number of mutants.
- Data analysis: Raw data is provided. No graphical examination of data is performed. No statistical tests are performed.
- Interpretation: Discussion is relevant to research questions. Practical issues are discussed
- Limitations of study: Threats to external validity are discussed.
- Replicability: High.

12. A methodology for testing spreadsheets (Rothermel et al. [32], TOSEM 2001)

- The objectives of the study: The study states specific research questions.  
Research questions: How effective are du-adequate test suites at revealing faults in spreadsheets?
- The experimental design:  
Object: Eight spreadsheets (number of cells/expressions/DU-associations/conditions, number of faulty versions, test pool size, average test suite size), implemented by students.  
Treatment: Two test suite generation methods (du-adequate and random).  
Treatment assignment: All treatments are applied to all programs.
- Data collection (measure): A test suite's efficacy (a formula is given).
- Data analysis: Graph examination of data is performed, but no assumption checks are conducted before applying statistical tests. Wilcoxon Rank-Sum test is applied and the p-value is provided.
- Interpretation: Relevant interpretation is made with respect to the research questions. No practical issues are discussed.
- Limitations of study: Threats to construct, internal, and external validity are discussed.
- Replicability: High.

13. A safe, efficient regression test selection technique (Rothermel et al. [33], TOSEM 1997).

- The objectives of the study: The study states objectives, but does not give specific research questions.  
Objective: Investigate whether proposed algorithms could reduce the cost of regression testing at the intraprocedural and interprocedural levels.
- The experimental design:  
Object: Seven Siemens programs and *player* (C programs, LOC, procedures, nodes, edges, versions, and tests), from multiple outside sources.  
Treatment: Two test selection algorithms (intraprocedural and interprocedural).  
Treatment assignment: All treatments are applied to all programs.
- Study setting: Uses Aristotle analysis tool and Sun Microsystems SPARCstation 10 with 128MB RAM.
- Data collection (measure): Percentage of tests selected, time required to run all tests, time required to run selected tests, and time required to select tests.
- Data analysis: Graphical data is shown. No statistical tests are performed.
- Interpretation: Relevant interpretation is made with respect to hypothesis. No practical issues are discussed.
- Limitations of study: Threats to internal, external, and construct validity are discussed.
- Replicability: High.

14. Empirical Studies of a Safe Regression Test Selection Technique (Rothermel et al. [34], TSE 1998) <sup>5</sup>.

- The objectives of the study: The study states an objective.  
Objective: Investigate the effects on test selection of using non-coverage-based test suites instead of coverage-based test suites.
- Study setting: Sun Microsystems SPARCstation 1+ with 24MB RAM.
- The experimental design:  
Object: Seven Siemens programs (C, LOC, number of functions, number of versions, and test pool size), from the Siemens Corporate Research.  
Treatment: Two types of test suites (non-coverage-based and coverage-based).  
Treatment assignment: All treatments are applied to all programs.
- Data collection (measure): Percentage of tests selected, average interquartile ranges, execution time to run all tests/selected tests, and time to perform analysis.
- Data analysis: Graphical data is shown. No statistical tests are performed. The comparison between the results from the two types of test suites is made based on average interquartile ranges.
- Interpretation: Relevant interpretation is made with respect to the objective of the study. Practical issue is discussed.
- Limitations of study: Threats to external validity are discussed.
- Replicability: High.

15. Prioritizing test case for regression testing (Rothermel et al. [35], TSE 2001).

- The objectives of the study: The study states research questions.  
Research questions: Can test case prioritization improve the rate of fault detection of test suite?

---

<sup>5</sup>This study performs five empirical studies, but only one of them is relevant to experiment, so we describe that particular study in this report.

- The experimental design:
  - Object: Seven Siemens programs and *space* (C programs, LOC, number of versions, number of mutants, test pool size, and average test suite size), from multiple outside sources.
  - Treatment: Nine prioritization techniques.
  - Treatment assignment: All treatments are applied to all programs.
- Study setting: Uses Aristotle analysis program and Proteum mutation system.
- Data collection (measure): APFD (a weighted average of the percentage of faults detected).
- Data analysis: Graphical examination of data is performed. ANOVA and Bonferroni tests are applied (SAS package), but no assumption checks for ANOVA test is performed.
- Interpretation: Relevant interpretation is made with respect to research questions. Practical implications are discussed.
- Limitations of study: Threats to internal, external, and construct validity are discussed.
- Replicability: High.

## 5 Discussions and Open Problems

In this section, we discuss observations drawn from our analysis and open problems in understanding testing techniques from an empirical point of view.

### 1. *Researchers often neglect to formulate specific research questions:*

Of the 15 papers we reviewed, only nine studies clearly stated specific research questions. Others provided only goals or objectives, which are often too general. For example, Briand et al. [4] state the objective of their study as follows: “Empirically evaluate three-based integration testing strategies.” We cannot determine what the study intends to investigate from such a description. The objective should specify at least those attributes of testing strategies that they want to compare, such as the number of stubs the testing strategies require.

### 2. *Statistical analysis is not well established:*

To draw meaningful conclusions from a study, we must perform statistical analysis and interpret the results. The papers we considered performed formal experiments, but surprisingly the number of studies that applied statistical analysis is very small. Only five studies analyze their data using statistical procedures. Even more problematic is that only one study among the five checks assumptions before applying statistical tests. Without checking assumptions, statistical tests may be incorrectly applied. Kitchenham et al. point out this issue as follows: The researchers were likely to find “statistically significant” results that did not really exist.

Graphical examination of data is always a useful way to check how a data set is distributed and whether outliers exist. For example, if a data distribution does not look normal and the sample size is small, then we need to consider applying a nonparametric test instead of a parametric test. Based on a judgement of the nature of data, one might find alternative ways to display data, such as transformation to a different scale, or other appropriate statistical procedures.

3. *Practical importance should be drawn from results:*

Researchers often overlook the practical implications of the results. They may find a statistical significance from data analysis, but as mentioned in Section 3, sometimes statistically significant results may not be practically significant and vice versa. Even in the absence of statistical analysis, it is important to interpret results from a practical point of view.

Ten of the studies discuss the practical implications of their results. It is an encouraging sign that many researchers are trying to interpret their results in a practical way, but this should be more widespread.

4. *Value to industrial practice is not addressed:*

Practical implications of the results are relevant to industrial practice, but techniques require scalability since in most cases experiments are done on small size software compared to commercial products. Since industry tends to be sensitive to cost-benefit tradeoffs when they consider whether to adopt new techniques, studies need to be aware of the projection of techniques into industrial contexts if they believe that their techniques have practical benefits. Kitchenham et al. [17] point out that research results are not in widespread use in industry because researchers address issues that are not relevant to industry, or present results that are incomprehensible to industrial decision makers.

Only two studies address this issue in their discussion: Bible et al. discuss the cost model for comparing regression techniques in terms of the product development phases, and Elbaum et al. perform a cost-benefit analysis considering the industrial context (such as translating potential savings from techniques into savings in the organization). Researchers should think one step further in their studies and not just interpret the results in a limited context.

5. *Long-term and lifetime views are overlooked:*

Despite its importance, none of the studies consider long-term or lifetime views of the techniques. It is not an easy task to consider these factors because it requires complex and large scale experiments. As mentioned in Section 3, one study by NASA found that after three major Ada development projects, the results showed that there were significant benefits to Ada over Fortran. Investigating three major development project periods is a large experiment and is likely impossible in a laboratory research environment. However, in some areas, particularly those in testing, we can investigate testing techniques' effectiveness in a static environment by using software programs having several versions that can be obtained from open-source software hosts, or (preferably) from software companies. This task is still not easy, but it is worthwhile to investigate techniques in this manner, as it may make a significant contribution to the testing community.

6. *Replicability requires substantial infrastructure support:*

Even though descriptions of experiment processes and all relevant components are detailed enough to perform replication studies, there exists a fundamental problem that may introduce uncertainty to replication studies: studies on testing techniques tend to be very dependent on tools utilized during

the experiment. For example, Briand et al.'s study [4] uses a reverse engineering tool to build objects from Java programs on which they conduct experiment. Hence their results are dependent on that particular reverse engineering tool. Without being able to obtain this tool, a replication study would be confounded by its use.

One possible solution to this problem is to build an infrastructure that helps with artifact sharing between researchers. A study by Do et al. [7] discusses several challenges that researchers face when they conduct an experiment, and addresses the replicability problem as one of those challenges. In fact, these challenges mainly arise from a lack of infrastructure support in software engineering. As suggested in their study, by expanding the infrastructure that provides various artifacts, we can expect a large number of independent replication studies.

7. *Presentation and documentation are not well organized:*

A well organized study provides a clear message and helps readers to better understand the significance of the study. In particular, when an experiment setup or design is complex, it is especially important. For example, if a study performs a significant and successful experiment, but its presentation is too poor to transmit its message clearly to readers, its significance will not be recognized by others.

During the evaluation of these papers, we find that presentation of experiment processes is problematic. Some papers introduce research questions in the introduction section, instead of in the experiment section. All relevant information to an experiment should appear in the experiment section. Readers may forget research objectives when they reach the experiment section, and thus they are forced to find where research objectives are stated. Some papers tend to describe entire experiment processes and results in one section without any break points. This style makes it difficult to understand what is described. It can be understood, but requires more time than would otherwise be necessary.

We recommend that the experiment section in a study should utilize a standard template that can be generally understood. Having a standard template, in turn, helps researchers to remember to report all the important information about an experiment.

## 6 Conclusions

We have provided an overview of the current state of controlled experimentation on testing techniques, and established an evaluation criteria for studies based on two surveys of empirical studies. During the evaluation of those studies in Section 4, we identified several areas for improvement in empirical experimentation.

While we recognize problems with experimentation from a specific set of papers, we also project some problems in a different light. The papers reviewed in Section 4 are considered to be high quality papers in software engineering, and were selected from preeminent journal publications. Thus the problems described in this report raise concerns. First, if journal papers exhibit problems in experimentation, then these problems may be more serious in conference papers, which have space constraints and tend to be simplified. Second,

papers appearing in journals typically go through a lengthy review process, and thus these papers are significantly revised in response to comments from reviewers. They are corrected if they contain flaws in the study. Yet we still see problems in these studies as we described in the previous section. We conjecture that reviewers or editors may not be aware of the importance of experimentation and thus may not have appropriate knowledge to recognize the problems we discussed. There is no quick remedy to these problems. Software engineers need to be aware of these problems and build a consensus that performing empirical studies is an essential activity for providing scientific evidence and inputs to the decision-making processes in an organization.

Criteria established in this report may miss additional important factors we need to consider, but we hope this report provides insight regarding how empirical studies in software engineering should be performed, and in particular the important factors researchers have overlooked in past experiments on testing techniques.

## References

- [1] J. Andrews and Y. Zhang. General test result checking with log file analysis. *IEEE Transactions on Software Engineering*, 29(7):634–648, July 2003.
- [2] V. R. Basili, R. W. Selby, and D. H. Hutchens. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, 12(7):733–743, July 1986.
- [3] J. Bible, G. Rothermel, and D. Rosenblum. Coarse- and fine-grained safe regression test selection. *ACM Transactions on Software Engineering and Methodology*, 10(2):149–183, April 2001.
- [4] L. Briand, Y. Labiche, and Y. Wang. An investigation of graph-based class integration test order strategies. *IEEE Transactions on Software Engineering*, 29(7), July 2003.
- [5] A. Brooks, J. Daly, M. Miller, M. Roper, and M. Wood. Replication of experimental results in software engineering. Technical Report ISERN-96-10, ISERN, 1996.
- [6] A. Dean and D. Voss. *Design and Analysis of Experiments*. Springer, 1999.
- [7] H. Do, S. Elbaum, and G. Rothermel. Infrastructure support for controlled experimentation with software testing and regression testing techniques. In *Proc. Int'l. Symp. Empirical Software Engineering*, pages 60–70, August 2004.
- [8] S. Elbaum, A. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. *IEEE Transactions on Software Engineering*, 28(2):159–182, February 2002.
- [9] J.A. Feldman and W. R. Sutherland. Rejuvenating experimental computer science - A report to the national science foundation and others. *Comm. ACM*, 22(9):497–502, September 1979.

- [10] N. Fenton and S. Pfleeger. Science and substance: A challenge to software engineers. *IEEE Software*, pages 86–95, July 1994.
- [11] R. Ferguson and B. Korel. The chaining approach for software test data generation. *ACM Trans. Softw. Eng. Meth.*, 5(1), January 1996.
- [12] T. L. Graves, M. J. Harrold, J.-M. Kim, A. Porter, and G. Rothermel. An empirical study of regression test selection techniques. *ACM Trans. Softw. Eng. Meth.*, 10(2):184–208, April 2001.
- [13] M. J. Harrold, D. Rosenblum, G. Rothermel, and E. Weyuker. Empirical studies of a prediction model for regression test selection. *IEEE Trans. Softw. Eng.*, 27(3):248–263, March 2001.
- [14] K. Hinkelmann and O. Kempthorne. *Design and Analysis of Experiments: Introduction to Experimental Design*. John Wiley and Sons, 1999.
- [15] N. Juristo, A. Moreno, and S. Vegas. Reviewing 25 years of testing technique experiments. *Empirical Software Engineering: An International Journal*, pages 7–44, March 2004.
- [16] N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, 2001.
- [17] B. Kitchenham, T. Dyba, and M. Jorgensen. Evidence-based Software Engineering. In *Proc. Int'l. Conf. Softw. Eng.*, 2004.
- [18] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, August 2002.
- [19] B. Kitchenham, L. Pickard, and S. Pfleeger. Case studies for method and tool evaluation. *IEEE Software*, pages 52–62, July 1995.
- [20] P. Koppol, R. Carver, and K. Tai. Incremental integration testing of concurrent programs. *IEEE Transactions on Software Engineering*, 28(6), June 2002.
- [21] M. Marre and A. Bertolino. Using spanning sets for coverage testing. *IEEE Transactions on Software Engineering*, 29(11), November 2003.
- [22] D. D. McCracken, P. J. Denning, and D. H. Brandin. An ACM executive committee position on the crisis in experimental computer science. *Comm. ACM*, 22(9):503–504, September 1979.
- [23] C. Michael, G. McGraw, and M. Schatz. Generation software test data by evolution. *IEEE Transactions on Software Engineering*, 27(12), December 2001.

- [24] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, New York, fourth edition, 1997.
- [25] A. Offutt, G. Rothermel, R. Untch, and C. Zapf. An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering and Methodology*, 5(2), April 1996.
- [26] D. Perry, A. Porter, and L. Votta. Empirical Software Engineering: A Roadmap. In *Proceedings International Conference of Software Engineering*, pages 345–355, May 2000.
- [27] S. L. Pfleeger. Design and analysis in software engineering. Part 1: The language of case studies and formal experiments. *ACM SIGSOFT Software Engineering Notes*, 19(4):16–20, October 1994.
- [28] S. L. Pfleeger. Experimental design and analysis in software engineering. Part 2: How to set up an experiment. *ACM SIGSOFT Software Engineering Notes*, 20(1):22–26, January 1995.
- [29] S. L. Pfleeger. Experimental design and analysis in software engineering. Part 3: Types of experimental design. *ACM SIGSOFT Software Engineering Notes*, 20(2):14–16, April 1995.
- [30] S. L. Pfleeger. Experimental design and analysis in software engineering. Part 4: Choosing an experimental design. *ACM SIGSOFT Software Engineering Notes*, 20(3):13–15, July 1995.
- [31] S. L. Pfleeger. Experimental design and analysis in software engineering. Part 5: Analyzing the data. *ACM SIGSOFT Software Engineering Notes*, 20(5):14–16, December 1995.
- [32] G. Rothermel, M. Burnett, L. Li, C. Dupuis, and A. Sheretov. A methodology for testing spreadsheets. *ACM Trans. Softw. Eng. Meth.*, 10(1), January 2001.
- [33] G. Rothermel and M. J. Harrold. A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology*, 6(2):173–210, April 1997.
- [34] G. Rothermel and M. J. Harrold. Empirical studies of a safe regression test selection technique. *IEEE Transactions on Software Engineering*, 24(6):401–419, June 1998.
- [35] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold. Prioritizing test case for regression testing. *IEEE Transactions on Software Engineering*, 27(10):929–948, October 2001.
- [36] Computer Science and Telecommunications Board. Academic careers for experimental computer scientists and engineers. *Comm. ACM*, 37(4):87–90, April 1994.
- [37] W. Tichy. Should computer scientists experiment more? *IEEE Computer*, 31(5):32–40, May 1998.
- [38] W. Tichy, P. Lukowicz, L. Prechelt, and E. Heinz. Experimental evaluation in computer science: a quantitative study. *Journal of Systems and Software*, 28(1):9–18, January 1995.

- [39] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslen. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, 2000.
- [40] M. Zelkowitz and D. Wallace. Experimental models for validating technology. *IEEE Computer*, pages 23–31, May 1998.