Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Centro de Informática

Pós-Graduação em Ciência da Computação

A SYSTEMATIC MAPPING STUDY ON SOFTWARE ENGINEERING TESTBEDS

Emanoel Francisco Spósito Barreiros

DISSERTAÇÃO DE MESTRADO

Recife
Fevereiro de 2011

Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Centro de Informática

Emanoel Francisco Spósito Barreiros

A SYSTEMATIC MAPPING STUDY ON SOFTWARE ENGINEERING TESTBEDS

Trabalho apresentado ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Sérgio Castelo Branco Soares*

Recife
Fevereiro de 2011

ACKNOWLEDGEMENTS

First I thank to God for the strength and perseverance I've been given. Without them I could not finish this course. I thank my parants, Manoel and Rejane, for providing the needed support since I was a tiny little person until now. Without you I could not finish this, be sure of that. Thanks to my brother Maurício and sister Manuela, for leting me stay up until late with the lights on. Thank you guys, you rock! I love you all!

Thank you to my advisor Sérgio Soares, who stood by me through the whole process, not only giving relevant insights that helped me drive this research, but also as a friend, having warm conversations about life and the future. Thanks man.

I thank the Informatics Center of the Federal University of Pernambuco for the great support provided to both students and professors. Thank you to all professors I had the opportunity to meet. Specially, thank you to all the members of the Software Productivity Group, who were always there to help and provided great discussions during our weekly meetings.

Thank you to my fiancée (soon my wife, I promise, honey!), Helaine Lins, for the patience and words of encouragement. You have helped in ways you don't even imagine. Love you!

Thanks to my good friends Adauto Almeida, Vinícius Garcia, and Juliana Saraiva, who have actively helped in this research. Of couse, also thanks for the uncountable and unforgettable moments during and after a long day (and night) of work. I hope our paths cross in the near future.

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong.

—RICHARD P. FEYNMAN

RESUMO

A pesquisa em Engenharia de Software precisa de mais evidências, principalmente em tópicos que envolvem interação humana, como manutenção de software. Muito embora pesquisa empírica esteja ganhando mais interesse, técnicas, metodologias e melhores práticas ainda estão em debate. Testbeds são ambientes empíricos nos quais pesquisadores podem planejar e executar estudos com a intenção de avaliar e/ou comparar tecnologias. O conceito é bem difundido em outras áreas, como Redes de Computadores, mas permanece pouco explorado na Engenharia de Software. O objetivo desta dissertação é agregar estudos relevantes que definem testbeds para avaliar tecnologias em Engenharia de Software. Para alcançar este objetivo, um Mapeamento Sistemático de Estudos da literatura de testbeds em Engenharia de Software foi executado. Baseado em buscas realizadas em mecanismos automatizados e buscas manuais em conferências importantes na área de Engenharia de Software, um total de 4239 estudos foi obtido. Treze (13) estudos primários foram identificados como relevantes e classificados de acordo com quatro perguntas de pesquisa. A partir da análise realizada, o estudo conclui que o uso de testbeds é benéfico em vários cenários. Importantes contribuições deste trabalho são: fornecer informações importantes para aqueles que desejam desenvolver testbeds, reduzir o tempo necessário para um tecnologia ser amplamente utilizada e disseminar o conceito pela comunidade.

Palavras-chave: Engenharia de Software, Engenharia de Software Empírica, Testbeds, Avaliação de Tecnologias, Transferência de Tecnologia.

ABSTRACT

Research on Software Engineering needs more evidence, primarily on topics that involve human interaction, such as software maintenance. Even though empirical research is growing in interest, techniques, methodologies and best practices are still in debate. In general, testbeds are empirical environments in which researchers are able to plan and conduct empirical studies to evaluate and/or compare technologies. The concept is well disseminated in other areas such as Computer Networks, but remains poorly explored in Software Engineering. The objective of this dissertation is to aggregate relevant studies that address the definition of testbeds to evaluate Software Engineering technologies. To accomplish that, a Systematic Mapping Study on the Software Engineering Testbeds literature was undertaken. Based on automated and manual searches in a set of engines and leading conferences in the field of Software Engineering, 4239 studies were obtained. A set of 13 primary studies were identified as relevant and classified according to four different research questions. From the analysis performed, the study concludes that the use of testbeds is beneficial in a number of scenarios. Important contributions of the current work are: to provide valuable information for those who want to develop new testbeds, to reduce the time taken for a new technology to be widely adopted, and disseminate the concept of testbeds throughout the community.

Keywords: Software Engineering, Empirical Software Engineering, Testbeds, Technology Evaluation, Technology Transfer.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Software is part of a system solution, a set of instructions that can be coded to execute in a computer environment. It includes the documentation needed to understand, transform, and use the solution. One of the dificulties researchers face is that software is developed, not produced [5]. From that sentence, another one can be extracted: software is manufactured by utilizing intellectual tools and creativiy of its developers. Human interaction plays a very important role in this scenario.

Software Engineering (SE) can be defined as the disciplined development and evolution of software systems based upon a set of principles, technologies, and processes [5]. As so, while SE matures, new ways of building knowledge must be introduced. The elements that compose technologies and their relations become more complex over time. In some cases, the creation of a concept or the introduction of a new technology cannot be based purely on theoretical work. They must be tried in situations that are similar to real ones to prove themselves as candidates to in the "real world". Empirical studies are performed in order to help researchers and practitioners better evaluate, predict, understand, control, and improve the software development process and product [4].

Empirical studies are activities that require a well defined process to guide the execution. In essence, the researcher defines an hypothesis and tests it using specifically designed processes in an iterative way. The information learned from the process is used to refine the hypothesis and develop new ones [4]. Empirical frameworks are important to help researchers build studies and analyze results in a systematic fashion, providing more reliable, comparable and replicable results [4, 6].

Even though frameworks proposed by the community aid researchers in many ways, the evaluation of technologies in SE still needs very specific guidelines. Testbeds provide means to systematically evaluate and compare technologies in SE. They have been used in different areas, but the use of testbeds to evaluate SE technologies has been first reported in 1988, by Robert Holibaugh, J. M. Perry and L. A. Sun [7]. In this seminal work, requirements and selection guidelines have been reported. The testbed aimed to provide an environment for the conduction of experiments to evaluate risks involved in developing software from reusable components and identify the information necessary to

apply reusable components during systems and software development.

Through the years, testbeds have helped researchers assess and compare technologies in SE. There is not, however, consensus in the community about how testbeds in SE should be built. It is important to aggregate the knowledge that has been produced by the SE community so far, in order to understand how testbeds work, the benefits they provide, and facilitate the construction of new testbeds by researchers. It is also not clear in which kinds of empirical studies testbeds are beneficial. The primary studies selected show the use of testbeds to help controlled experiments, experiments not involving human interaction and case studies. The use of testbeds to help the conduction of ethnographies, for example, was not found.

The motivation of this work is the need to produce better software, by allowing new and promising technologies to be early adopted by the general public. A mapping study that groups all available work in the field is of great support to researchers that intend to build new testbeds in SE.

1.1 PROBLEM AND RESEARCH QUESTIONS

In the scenario described above, the central research problem may be stated as:

> How can testbeds help researchers in evaluating technologies in SE? What should be considered in order to build a new testbed in SE?

Based on these fundamental questions, the current research tried to investigate the use of testbeds as a tool to conduct robust empirical studies in SE. Mainly, the research topics investigated through the use of testbeds have been identified, as well as how the testbeds are composed, their benefits, and evaluation methods. The research problem can be divided into four research questions, namely:

1. **RQ1:** Which specific software engineering fields are investigated using testbeds?
2. **RQ2:** Which elements the studies define for a testbed in software engineering?
3. **RQ3:** Which are the benefits of having a software engineering testbed?
4. **RQ4:** Which methods are used to evaluate the proposed testbed?

1.2 OBJECTIVES

The current study aims to investigate the use of testbeds in SE to empirically evaluate technologies by mapping the specific research topics investigated, the elements proposed

to compose the testbeds, the benefits the researchers claim by using them, and the evaluation methods used to assess the efficaccy of the proposed testbeds.

1.2.1 Specific Objectives

The current work aims at achieving the following specific objectives:

- Conduct a systematic mapping study on software engineering testbeds;
- Identify evidences of the topics investigated by the literature of testbeds in software engineering, the elements used to build testbeds, as well as the benefits of using them and the evaluation methods used to assess their efficacy.
- Promote the realization of empirical studies in software engineering.

1.3 DISSERTATION ORGANIZATION

The current work is organized as follows. Chapter 2 describes the background in empirical software engineering and the main concepts that concerns the field. Some difficulties regarding the conduction of empirical studies in software engineering are discussed. The experimental paradigm and an empirical framework are shown. It also addresses the Evidence-Based Software Engineering (EBSE) paradigm, focusing in Systematic Mapping Studies.

Chapter 3 describes the methodology used to conduct this work. The research is classified, the methodological board is presented as well as the main research steps. The protocol developed and used to guide the mapping study is also described.

Chapter 4 presentes the results of the mapping process. Research questions are answered and the data gathered is organized accordingly. General information about the studies is presented, such as the total amount of studies returned, the participation of each automated search engines in the set of primary studies selected, and the publication distribution over the years.

Chapter 5 presents the final considerations, conclusions obtained by this work, alongside with some threats to validity and future work. A final analysis of the work is also presented.

CHAPTER 2

BACKGROUND

In this chapter studies related to empirical software engineering, testbeds, and evidence-based software engineering are discussed.

2.1 EMPIRICAL STUDIES IN SOFTWARE ENGINEERING

Although computer science is an area of knowledge related to exact sciences, software engineering (SE) has very peculiar characteristics that strongly relate to the social sciences. For this reason, it is imperative to encourage the implementation of empirical studies that are able to evaluate the effectiveness of techniques, methodologies, and processes proposed in the area.

Experimentation, one kind of empirical study, allows the knowledge to be generated in a systematic, disciplined, quantifiable, and controlled way, especially in areas where human interaction dominates the scene [6]. For this reason, experiments are so common in social and behavioral sciences, which are closely related to human behavior and their interactions [8]. However, these characteristics make experiments very difficult to plan and evaluate, because it is not possible to generate accurate models, as opposed to Mathematics and Physics. Even if the experiment is well planned, there are still many variables that are very difficult to isolate.

One way to aid researchers is providing evidence of the benefits of a new process or methodology adoption by applying the process or methodology in some situation that is at least close to a real situation. However, most companies are not willing to risk a project to "assess" new methods and/or technologies. Empirical studies are essential in order to fill this gap in the industry, providing, to some extent, reliable data about a given technology, easing technology transfer. The issue has already been widely discussed by the SE community [6, 9, 10].

Empirical studies in SE started gaining momentum when the need for them began to be explained [4]. However, the nature of SE does not allow one to apply ordinary analytical processes common to other engineering areas. As previously mentioned, besides being virtually impossible to eliminate all variables that influence the results of the stud-

ies, activities in SE are inherently human and depend on their creativity. For example, an experienced developer might perform very differently when compared to an unexperienced one or knowledge of different programming paradigms can greatly influence results in studies related software development. Even worse, mood changes, personality traits or any other personal chatacteristic can easily contaminate results.

Rigorous empirical studies are very rare in the field, as described by Tichy et al. [11] and Zelkowitz et al. [12]. The main difficulties that lead to the scarcity of studies are:

- The difficulty of defining and encapsulating all information needed by the study;
- The difficulty of controlling the involved variables in the software development process;
- The human involvement, which adds countless variables to the study and generates the risk of contamination of the results;
- High cost to perform an experiment in software development due to the personnel allocation.

In this context, an alternative to offer a good set of empirical tools to evaluate technologies in controlled environments is to use a testbed. In essence, testbeds are a set of artifacts associated with a software system, together with the infrastructure required to run empirical studies, mostly controlled experiments, on that system [13]. By artifact we mean the software documentation, the experimental plan, test plans, source code, evolution versions, and any other software artifact that is used in a real project. With such artifacts, the testbed infrastructure (methods and tools) can support the execution of empirical studies, also favoring their replications, since the executed studies should also be part of the testbed, therefore available to other researchers. Usually, testbeds are assembled to assess specific technologies, such as tools that evaluate software architecture [13]. For example, Lindvall et al. [13] have chosen a system to serve as an object of study. Defects were seeded to the architecture and tools were applied to try to detect these faults. If the experiments and their replications follow the experimental plan, it is possible to say, with greater credibility, when an application performs better than other. A great benefit of a testbed is that after its configuration, the replications are much faster because it saves the time taken for the initial assembly (infrastructure creation and setup).

2.1.1 One Experimental Framework

According to Wöhlin et al. [6] and Basili et al. [4], the experimental process in SE comprises (i) *Definition*, (ii) *Planning*, (iii) *Operation*, (iv) *Interpretation* and (v) *Packaging*. Each of the phases is briefly presented next.

1. **Definition:** Before anything else, the study must be defined. The definition of the study can be divided in six parts:

   (a) **Motivation:** The reason why the study is to be carried. For example, a given study may intend to provide efficient tools to evaluate software dependability [14];

   (b) **Object:** The entity examined in the study. For example, a Software Architecture Evaluation (SAE) tool [14];

   (c) **Purpose:** May be defined as to assess the efficacy of a specific tool to detect architectural defects in a system specification [14];

   (d) **Perspective:** From which point of view the study is conducted. For example, the study can be performed in the perspective of project managers or programmers;

   (e) **Domain:** The domain in which the experiment is run, for example, a single team, or a project with several teams ;

   (f) **Scope:** Classification according to the size of teams and projects as seen in Table 2.1.

Table 2.1 Scope definition. Extracted from [4].

	#Projects	
#Teams per project	One	More than one
One	Single Project	Multi-project variation
More than one	Replicated project	Blocked-subject project

2. **Planning.** In this phase, the study is planned concerning its (i) design, (ii) criteria and (iii) measurement. The scope of the study is associated with analytical methods.

3. **Operation.** In this phase, the study is executed. The phase comprises (i) preparation, (ii) execution and (iii) analysis. The preparation may include a pilot study to confirm the experimental design. The execution collects the data as results of the experiments, which is analyzed later. Collection and analysis can take quantitative or qualitative approaches, or even a mix of both.

4. **Interpretation.** The results of the study are analyzed in order to draw conclusions about their impact. This phase can be split into three minor phases, namely (i) interpretation context, (ii) extrapolation and (iii) impact. Since experiments are usually executed in a controlled environment, the results must be extrapolated to try to obtain broader conclusions observing the study's limitations.

5. **Packaging.** The study must be packed to allow dissemination. All the assets used in the study must be assembled in a package. Examples of the assets packed are the experimental plans, software documentation (requirements document, architecture document, use cases), change scenarios, test plans. The results and statistical analysis may also be packed to facilitate replication and comparison.

2.2 SOFTWARE ENGINEERING TESTBEDS

Researchers still struggle when trying to evaluate SE technologies. Even though the empirical paradigm is constantly growing and being enhanced, there is still no consensus regarding the empirical evaluation of software engineering technology. In most sciences there are established mechanisms for performing experimentation with a set of well adapted tools for the purpose, e.g. particle accelerators in physics, microscopes in chemistry and radio telescopes for astronomy [14].

A large number of new technologies are being developed, like architecture description languages (ADLs) [15] and aspect-oriented software development (AOSD) [16]. However, only a small part of the technologies being developed is seeing use in real projects. This is partly due to the fact that in some cases, they are poorly evaluated. It is becoming more accepted by practitioners and researchers that empirical studies are able to reduce that gap by providing powerful tools, by assessing if a given technology is feasible and widely usable [13]. According to Lindvall et al. [13], the main obstacles to the existence of more empirical studies in SE are the insuficient specification of context variables, experimentation costs, and the amount of risks involved.

A great benefit of systematic empirical studies in software engineering is their amenability to replication. This capability is crucial for the transfer of knowledge because it allows

a specific study to be conducted in other environments and to be evaluated by other researchers [17]. Aiming to ease replication, one possibility is the creation of families of empirical studies [18], allowing other studies to be performed to confirm or not results obtained by other researchers. However, this is not enough to promote efficient replication of empirical studies, as we increasingly see the need for the existence of guidelines, processes and techniques to help in the planning, implementation, replication, and analysis of results [5, 10].

Another critical factor that hinders the use of new technologies is the little evidence and relevant data from prior experiences demonstrating the positive feedback on the technology use [19]. According to Redwine and Riddle [20], a new technology can take from 15 to 20 years to mature and be used by practitioners.

Software engineering testbeds try to reduce the risks of empirical studies in the field and provide more credibility to the studies that use them. The 13 primary studies selected by the mapping study claim 20 different benefits of using testbeds to evaluate technologies in software engineering. Couriously, nearly half of the benefits (9 out of 20) are cited by more than one study. This shows how the community is consonant with the concept and benefits of using testbeds in SE and how testbeds appear to be consistently improving the experience in empirical software engineering (ESE).

Testbeds are composed of a set of artifacts associated with a software system together with the infrastructure required to run empirical studies on that system [13]. Usually, testbeds are assembled to assess specific technologies, such as tools for evaluating software architecture [13]. For the latter, a system was chosen to serve as an object of study. Defects were seeded to the architecture and tools were applied to try to detect these faults. If the experiments and their replications follow the experimental plan, we can say with greater certainty, when an application performs better than another. A great benefit of a testbed is that after its configuration, the replications are much faster because it saves the time taken for the initial assembly (creation of infrastructure).

2.3 EVIDENCE-BASED SOFTWARE ENGINEERING (EBSE)

In the late 80s and early 90s medical studies were facing a difficult phase, in which failure was a constant research result. In clinical analysis, experts were drawing conclusions that, in many cases, were different from the results of systematic reviews [21].

In a scenario where badly designed studies could mean the death of human subjects, medical researchers have adopted the evidence-based paradigm. In the late 90s, the publication rate of evidence-based work in medical sciences have increased greatly, driving

the creation of several journals specialized in systematic reviews [21].

Even though the paradigm has been proposed to fulfill medical sciences' needs, it is also applicable to science fields in which knowledge building through the systematic review of the literature is positive. Hence, according to Kitchenham et al. [21] the goal of EBSE is *to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software.*

As Petersen et al. argue [22], as a research area matures, the number of reports and results have a significant increase. It is important for the community to keep track of these advances in the form of systematic literature reviews. Recently, the use of systematic literature reviews has been widely adopted [23, 24, 25].

As such, Systematic Literature Reviews (SLR) and Systematic Mapping Studies (SMS) emerged as two important tools to aggregate and build knowledge in Software Engineering (SE). According to Kitchenham and Charters [23], systematic reviews have the following advantages:

- The well-defined methodology makes it less likely that the results of the literature are biased, although it does not protect against publication bias in the primary studies;
- They can provide information about the effects of some phenomenon across a wide range of settings and empirical methods. If studies give consistent results, systematic reviews provide evidence that the phenomenon is robust and transferable. If the studies give inconsistent results, sources of variation can be studied;
- In the case of quantitative studies, it is possible to combine data using meta-analytic techniques. This increases the likelihood of detecting real effects that individual smaller studies are unable to detect.

In return, systematic reviews require much more effort to be carried out when compared to traditional literature reviews. SLRs are particularly concerned in answering very narrow questions, usually intending to build new knowledge based on the aggregation of existing studies in a given field. SMSs are usually employed when the researcher wants to explore a given area, when there is little evidence on a particular field or to guide a future SLR or further studies.

The results of a systematic mapping study are presented in a higher level of granularuty, allowing the identification of evidence clusters and evidence deserts. In summary, the main differences between SLRs and SMSs are [23]:

- Mapping studies generally have broader research questions driving them and often ask multiple research questions;
- The search terms for mapping studies will be less highly focussed than for systematic reviews and are likely to return a very large number of studies, for a mapping study however this is less of a problem than with large numbers of results during the search phase of the systematic review as the aim here is for broad coverage rather than narrow focus;
- The data extraction process for mapping studies is also much broader than the data extraction process for systematic reviews and can more accurately be termed a classification or categorization stage. The purpose of this stage is to classify papers with sufficient detail to answer the broad research questions and identify papers for later reviews without being a time consuming task;
- The analysis stage of a mapping study is about summarising the data to answer the research questions posed. It is unlikely to include in depth analysis techniques such as meta-analysis and narrative synthesis, but totals and summaries. Graphical representations of study distributions by classification type may be an effective reporting mechanism;
- Dissemination of the results of a mapping study may be more limited than for a systematic review; limited to commissioning bodies and academic publications, with the aim of influencing the future direction of primary research.

2.4 SUMMARY

Empirical studies are of great importance to software engineering because of its characteristics, which relates it to social sciences. By conducting empirical studies in SE, researchers are able to build knowledge incrementaly. This can reduce the gap that exists between industry and academia. Empirical studies are very difficult to run because of both operational complexity and costs.

Testbeds appear as a good alternative to those that intend to systematically evaluate technologies in SE. They can reduce the time taken for a study to be conducted because researchers can, depending on the situation, reuse all the assets used in the first run. Examples of assets that can be reused are: experimental plan, test plans, source code of the applications used, and metrics suite.

Some testbeds have to the evaluate technologies in SE have already been defined, but they seem to be very different from each other. It is understandable, to some extent, because they were designed to evaluate very different technologies. It is important, however, to understand how they are composed if a researcher intends to build new testbeds. EBSE helps the researcher to aggregate knowledge from the literature in a systematic fashion. By using strict protocols, SLRs and SMSs are capable of generating trustful studies and presenting insightful results.

CHAPTER 3

METHODOLOGY

This chapter discusses the methdology used to conduct the research. Section 3.1 shows the philosophical stance and methodological approaches used as well as the research steps taken. Section 3.2 describes the protocol used to guide the systematic mapping study conducted.

3.1 CLASSIFICATION AND RESEARCH STEPS

The present research uses the inductive approach, based on qualitative data. It is also based on the systematic mapping study method [22] to collect and analyze primary studies.

According to Marconi and Lakatos [26], the *inductive* method is a mental process by which one can, using sufficiently validated data, infer a general or universal truth not contained in the studied parts. The inductive approach is characterized by three phases:

1. **Observation of phenomena:** Observe the facts and analyze them aiming to discover the causes of their manifestation;

2. **Discover relation among them:** By using comparations, approximate the facts and phenomena, aiming to discover the constant relation among them.

3. **Generalization of relation:** Generalize the relations found in the previous phase to similar phenomena and facts, many of them still not observed yet.

The Systematic Mapping Study (SMS) method has gained much attention in software engineering and was inspired by medical research. Briefly, systematic reviews (SR) exhaustively queries the literature to collect relevant primary studies, performs an in-depth analysis and classifies those estudies according to the defined research questions. Yet, it discusses the results and provides relevant insights about them. Compared to ordinary reviews, SRs have many benefits: a well-defined methodology reduces bias, a wider range of situations and contexts can allow more general conclusions, and use of statistical meta-analysis can have conclusions that individual studies in isolation cannot [23, 22].

The nature of the research variables is mainly qualitative. Qualitative research is characterized by the analysis of deeper aspects, describing the complexity of human behavior. It provides more detailed analysis about the investigations, habits, attitudes, and behavior tendencies. It is also characterized by not using statistical instruments to argument conclusions [26].

3.1.1 Classification According to Cooper

Systematic reviews of the literature have drawn great attention of the community. A taxonomy was proposed by Harris M. Copper to try to classify and evaluate systematic literature reviews [27]. Originally proposed as a classification system for systematic literature reviews in education and psychology, it is fairly applicable to systematic literature reviews in general. The classification according to Cooper's taxonomy can be found in Table 3.1.

Table 3.1 Classification according to Cooper's taxonomy.

Characteristic	Category
Focus	Research outcomes Research methods Practices of applications
Goal	Integration (Generalization) Identification of Central Issues
Perspective	Neutral Representation
Coverage	Exhaustive
Organization	Methodological
Audience	Specialized Researchers Practitioners

The focus of a review concerns the research's central issue. In general, reviews focus on one or more of four areas: research outcomes, research methods, theories, and practices or applications. They are not mutually excludent, and it is common for reviews to have more than one focus.

The goal concerns what the researcher expects to conclude. For reviews, the most common goal is to organize and synthesize past literature pertaining to the same issue. Basically, literature reviews can have a combination of three different goals: Integration, Criticism and Identification of Central Issues. When a review aims to critique the literature, the researcher intends to show that the results from previous studies are not justifiable. The thid goal is the identification of central issues, in which the intention of

the researcher is to provide suggestions about how the problems in a certain area emerged.

The perpective influences how the researcher's point of view influences the discussion of the literature. There are two possible roles of reviewer's perpective: neural representation and espousal of position. In the former, the one tries to not "contaminate" the study with one's position. The researcher tries to represent all sides, in a fashion similar to that employed by the original authors. In the latter perspective, the authors may choose to omit particular studies if they don't contribute to the arguments posed by the authors.

The fourth characteristic of a review is its coverage. It is related to the decisions made by the authors on how to search for studies and how to select them. There are four different types of coverage: Exhaustive, Exhaustive with Selective Citation, Representative, and Central (or Pivotal). In the first one, exhaustive coverage, the authors are comprehensive in the presentation of works. They make an effor to include all the literature related to the topic under consideration, or at least most of it. The second converage type, exhaustive with selective citation, is similar to the exhaustive coverage, but only a subset of works is described in the resulting paper. In the representative coverage the authors present only a set of works that are representative of many other works in a field. Samples are used to simplify large sets of studies. The authors argument the characteristics that make the selected sample illustrative of a larger group. In the last type of coverage, central (or pivotal), the reviewer concentrates on studies that are pivotal to the area under investigation. This may include material that initiated a line of thinking, introduced new methods or engendered an important debate. Studies can employ more than one coverage strategy. Exhaustive and exhaustive with selective citation are clearly exclusive, but the others can be used in conjunction.

The fifth characteristic dictates how a paper is organized. Reviews can be arranged historically, when the works are presented in the chronological order in which the were published, conceptually, when the studies are grouped in categories of similar ideas, and methodologically, when studies that apply similar methods are grouped under the same subtopics.

The audience to which the study is directed to is the last characteristic. Reviews can be written for specialized researchers, general researchers, policy makers, practitioners or the general public. This characteristic is more related to the author's writing style and the terminology used.

3.1.2 Research Steps

The research started with an ad hoc search on the literature of software testbeds. The concept of testbed has been widely used in other Computer Science areas, such as Computer Networks, Artificial Intelligence and Telecommunications. It is strongly related to the idea of technology evaluation in a controlled environment and, in SE it is not explored as much.

It was clear that there was no consensus on the community on how to use testbeds in SE. Also, it was not clear how a testbed could be built from scratch. For example, which structural elements are necessary, how to evaluate, and which software engineering topics have already been investigated using testbeds are questions not easily answered. By analyzing this initial set of studies, many important benefits have been identified when using testbeds, which helps to stress the importance testbeds have for improving the empirical paradigm in software engineering. No systematic study that brings together the knowledge of the area was found.

Based on that, a mapping study was planned and executed. The problem was formalized in the form of four research questions and a research protocol was developed. From then on, the entire research has been guided by this protocol. At the end, analysis results from the selected primary studies are shown, mapped according to the specified questions.

The SMS was chosen because of the exploratory nature of the questions asked (see Section 3.2.1). According to Kitchenham et al. [23], the questions asked on an SMS are much broader and general, in contrast to the ones asked on an SLR, that must be very precise and narrow. Yet, it is not common to have more than two research questions on a SLR. In fact, this work addresses four research questions. The steps for each main phase are depicted in Figure 3.1. Figure 3.2 details the evaluation process that is carried out in the study selection step.

3.2 SYSTEMATIC MAPPING STUDY PROTOCOL

This section presents the protocol used in the SMS conducted in this work. On the preliminary readings, it was observed that there was no solid concept wrapping the testbed definition in SE. Therefore, an SMS to collect studies that could give researchers relevant insight on such topic is appropriate. With the mapping results in hand, one can extrapolate and define new testbeds taking advantage of lessons learned and best practices from similar studies.

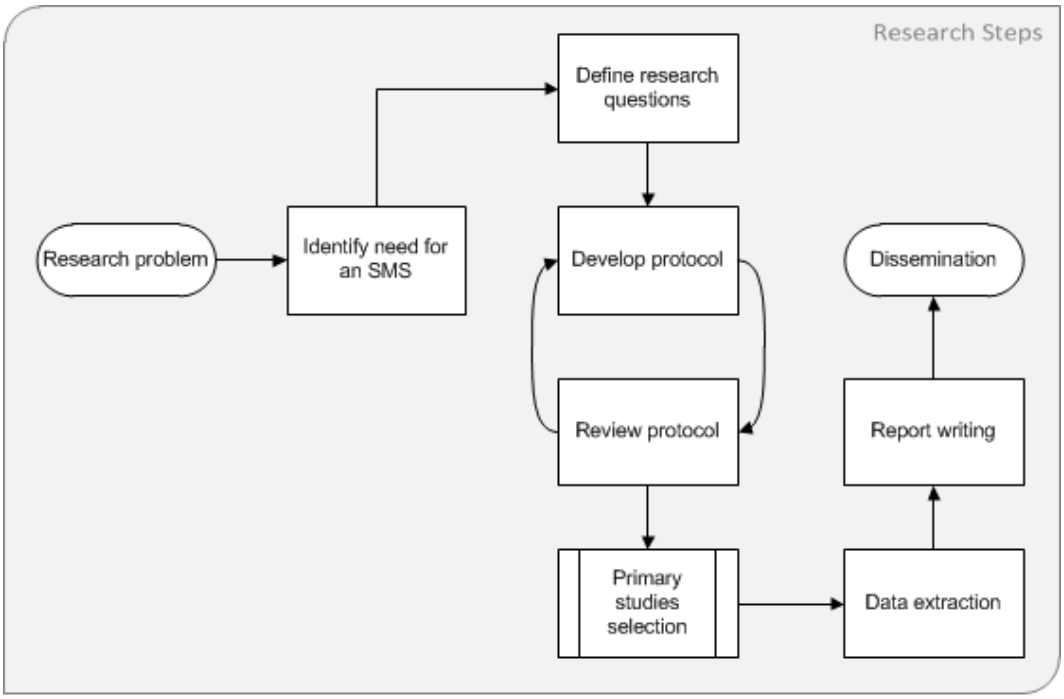

Figure 3.1 Detailed research steps.

Initially the intention was to perform an SLR, but based on the research questions, an SMS was the correct choice. The main difference between these two types of studies is that when one performs an SLR, one is concerned in answering very narrow questions (in most cases, only one or two questions). For example, Kitchenham et al. [28] have performed an SLR to try to determine under what circumstances individual organisations would be able to rely on cross-company based estimation models. In this research, the following research questions were used:

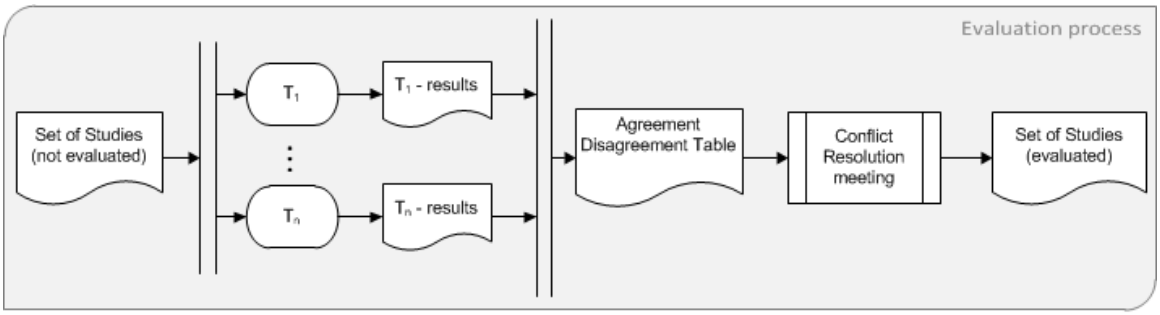

Figure 3.2 Steps for the selection process.

1. What evidence is there that cross-company estimation models are not significantly worse than within-company estimation models for predicting effort for software/Web projects?

2. Do the characteristics of the study data sets and the data analysis methods used in the study affect the outcome of within-company and cross-company effort estimation accuracy studies?

In contrast, Pretorius and Budgen [29] have conducted a systematic mapping study to collect evidence on the forms and models used in UML. In such study, the authors have elaborated the following research questions:

1. How many papers have reported on empirical evaluations on the models and forms used in the UML?

2. Which aspects of the UML have been most subjected to empirical study?

3. Which aspects of the UML could be recommended as topics for further empirical study?

4. Which research methods are most frequently employed in empirical studies of the UML?

It is easy to observe that the first study is concerned with answering narrow questions, while the second is rather focused on generating a map of the knowledge pertaining to a the area of UML modeling.

When one performs a SMS, one is interested on exploring a given field of study and, for example, mapping the "deserts" of knowledge in that field. Usually, a mapping study have more questions, and most importantly, they are more general when compared to SLR questions. Both studies, however, require a protocol. Next, the defined protocol is described. The guidelines defined by Kitchenham and Charters at the Software Engineering Group at Keele University [23] were used.

According to Kitchenham et al. [21], SLRs and SMSs protocols should have the following steps:

1. Identification of the need of a systematic review;

2. Formulation of a focused review question;

3. A comprehensive, exhaustive search for primary studies;

4. Quality assessment of included studies;
5. Identification of the data needed to answer the research question;
6. Data extraction;
7. Summary and synthesis of study results (meta-analysis);
8. Interpretation of the results to determine their applicability;
9. Report writing.

The current study did not follow all the steps defined by Kitchenham. However, the steps taken by this research were strongly influenced by the ones defined in the latter study.

3.2.1 Research Questions

This secondary study aims to search the literature and map primary studies that define testbeds on SE. The following research questions were identified as relevant to our purpose:

1. **RQ1:** Which specific software engineering fields are investigated using testbeds?
2. **RQ2:** Which elements the studies define for a testbed in software engineering?
3. **RQ3:** Which are the benefits of using a software engineering testbed?
4. **RQ4:** Which methods are used to evaluate the proposed testbed?

3.2.2 Search Process

It is not uncommon to set a time period in which the search is performed, but this policy is generally used when the search is executed solely manually. In some cases (as of this mapping study) the search is done both manually and through automated search engines. A set of automated search engines were used, namely: IEEE Computer Society Digital Library, ACM Digital Library, Scopus, Citeseer, and Springer Link. Manual searches were also performed on the proceedings of the last edition of the International Conference on Software Engineering (ICSE '10), Empirical Software Engineering and Measurement (ESEM '09) [1], Evaluation and Assessment in Software Engineering (EASE '10) and on

[1]The search was performed prior to the 2010 edition.

the past twelve months of the Empirical Software Engineering Journal (from June 2009 to June 2010). Even though there was a time constraint on the manual searches, the automated ones were performed without this limitation. Researchers have reported that automated searches are flawed in a several ways [30], hence, we preferred to not only rely on such mechanisms to try to reduce the risks of not selecting relevant studies that still have not been indexed by the search engines. The range of manual searches is still small, but an extension is scheduled to occur in a future iteration of this work.

Searches were performed using the following keywords: software engineering, testbed, family of experiments, technology evaluation, technology comparison, technology transition, software maturity, software adoption, replication, replicability, replicate. A general query to be used on the selected search engines was built:

> "software engineering" AND (testbed OR "family of experiments" OR "technology evaluation" OR "technology comparison" OR "technology transition" OR "software adoption" OR "software maturity" OR "replicability" OR "replicate" OR "replication" OR "experiment protocol" OR "study protocol" OR "experiment process" OR "experiment guidelines" OR "study process" OR comparability OR comparable OR compare)

3.2.3 Study Selection Criteria and Procedure

The mapping study process suggests that the papers are screened by at least two researchers in order to avoid biased evaluations. Each paper returned by the initial search was, then evaluated regarding their relevance to the mapping study. In this initial step, only the title, keywords and abstracts were taken into account. It is important to stress that only papers that were clearly out of scope were excluded in this phase, keeping all potential primary studies for further analysis.

After the initial selection, the full version of each paper is obtained so that a more detailed analysis can be performed. The following criteria should be applied in order to include or exclude the paper from the review:

- It is a real paper, not presentation slides or extended abstract;
- The paper is not a duplicate;
- The paper defines a testbed (or at least a process, framework, etc., that is concerned with evaluation of technologies);

- The paper is related to software engineering.

3.2.4 Relevance Evaluation

The process of evaluating the relevance of the primary studies is important, for example, for providing more inclusion/exclusion criteria, for providing a means of weighing the papers when results are being synthesized, and to guide recommendations for further research [23]. The relevance assessment was performed through questions that should be answered with three possible values:

- Yes (Y): If the question is fully answered, this answer counts the value 1.0;
- Partly (P): If the question is partly answered, this answer counts the value 0.5;
- No (N): If the question is not answered or not satisfactorily answered, this answer counts the value 0.0.

There were five questions to be answered, which makes possible for each paper to score from 0.0 to 5.0. The questions try to measure, to some extent, both the methodological quality and relevance to the mapping study. The defined questions are:

1. Does the paper define the concept of testbed?
2. Has the paper evaluated the testbed it proposes?
3. Does the testbed evaluation allows replication?
4. Does the study define a process for using the testbed?
5. Does the study point out the benefits of having and/or using a testbed?

After this grading, the primary studies were classified in five different groups: *Very Good* (between 4.5 and 5.0), *Good* (between 3.0 and 4.0), *Regular* (between 1.5 and 2.5) and *Poor* (between 0.0 and 1.0). The intention was to create four evenly distributed classes, since it was not possible, it was preferred to keep the highest class to try not induce bias towards better classification of the studies.

3.2.5 Data Extraction Strategy

The data extraction was performed using three forms. Form A is used to list the studies that were included in the mapping. It shows general data, such as year of publication,

authors and code for further reference in the mapping study. Form B is used to show the studies excluded in the second phase of evaluation. In this form, the most important information found is the reason why the study has been excluded. Form C is used to extract all relevant information to answer the research questions. These forms can be seen in Appendix C.

3.3 SUMMARY

This chapter presented the methodology used in the study. The inductive approach and positivist philosophical stance were adopted. The study is mainly qualitative and also adopts the systematic mapping study method to collect and analyze primary studies.

A protocol was developed to guide the research. The study identified a set of relevant keywords to run through the search engines. The studies returned by the search process have be classified according to four research questions that were developed to address the research problem. A quality assessment process is also described.

CHAPTER 4

EXECUTION AND RESULTS

This chapter provides the results and execution of the protocol defined in Chapter 3. The results are grouped in three categories, namely:

- **Data Extraction and Analysis**: General data about the results are presented, such as the amount of primary studies returned by each search engine, the count of selected papers for each engine and so forth;
- **Evidence Mapping**: The data regarding the answers to the research questions are discussed.
- **Discussion on Results**: In this section we present a discussion on the main results found by the systematic research.

4.1 DATA EXTRACTION AND DATA ANALYSIS

By executing the protocol defined in Chapter 3, the primary studies were obtained. The search string was applied in each search engine, which resulted in 4239 candidate studies. In separate, IEEE returned 2141 studies, ACM returned 100 studies, Citeseer returned 393 studies, Scopus returned 1507 studies and Springer returned 97 studies. Manual searches were also performed on the proceedings of ESEM 2009[1], EASE 2010[2] and ICSE 2010[3] but no relevant studies were found. Yet, manual searches were performed on past editions of the Empirical Software Engineering Journal[4] ranging from June 2009 to June 2010, and only one relevant primary study was found. Figure 4.1 depicts the amount of studies returned by each search engine.

Scopus was the search engine that returned the second higher number of primary studies. However, one very interesting phenomenon occurred: all relevant studies returned by Scopus were also obtained by the other search engines. This may lead to some interesting

[1] http://www.csc2.ncsu.edu/conferences/esem/
[2] http://www.scm.keele.ac.uk/ease/ease2010.html
[3] http://www.sbs.co.za/ICSE2010/
[4] http://www.springerlink.com/content/1382-3256

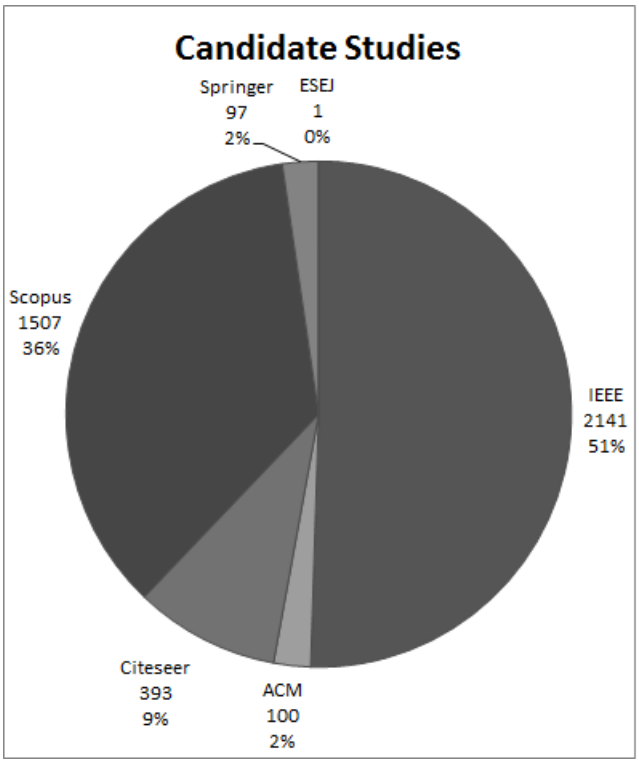

Figure 4.1 Amount of primary studies returned by the automated search engines.

series of studies to evaluate the efficiency of the automated search engines. In comparison to other SMSs in the literature, the current study selected a small number of relevant studies (13 out of 4239), which show that the research topic is still in its beginning. The reader can find the representativeness of each search engine on the chart depicted on Figure 4.2.

The amount of primary studies returned by the application of the search string to the search engines was very high (4239 in total). This number was greatly reduced in the first screening session, which consisted of evaluating the title, abstract and keywords, leaving only 106 candidate primary studies. After the second phase, which consisted of screening the whole paper, the final set of papers was defined. The final set of primary studies sum 13 studies, which are listed at Appendix A. The 93 papers that were excluded after the second screening are listed at Appendix B. Each one is accompanied with the reason why it was excluded. The number of primary studies is very small when compared to the amount returned by the queries, but this problem arises when the search process uses automated seach engines [23]. The evolution of the screening process is shown in Table 4.2.

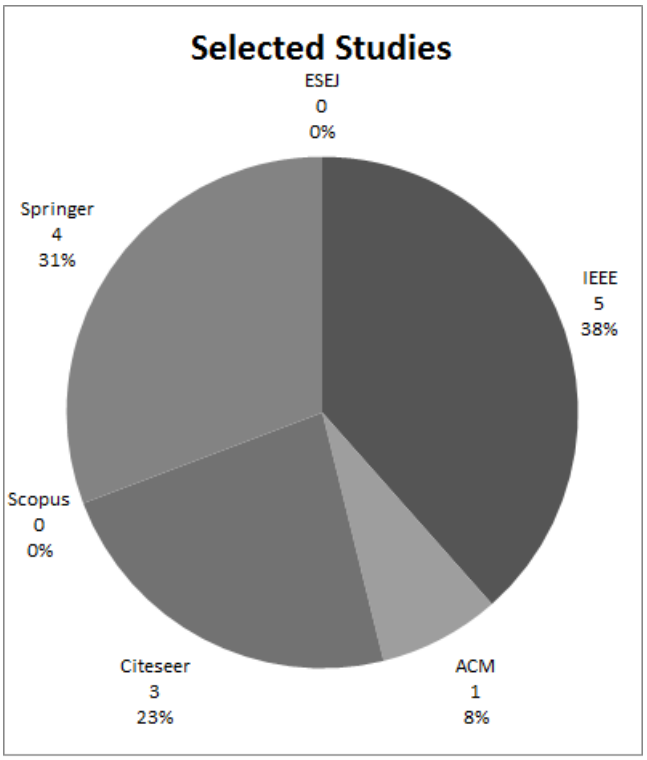

Figure 4.2 Representativeness of each search engine on the selected studies.

The search through automated engines did not impose a time constraint regarding the publishing year. The selected papers were published from 1988 to 2009. The distribution of the papers over the years is shown in Figure 4.3

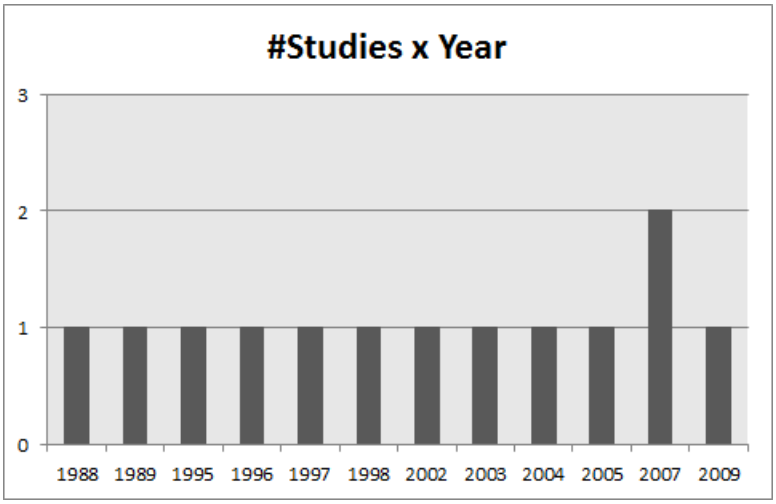

Figure 4.3 Distribution of studies by year.

There were 74 different authors in the 13 selected studies. The ones that published more than one paper are presente in Table 4.1. The authors come from 28 institutions, based on 11 different countries. The chart on Figure 4.4 shows the participation of each country in the selected papers.

Table 4.1 Authors who have published more than one paper about testbeds in SE.

Author	#Studies
Forrest Shull	3
Mikael Lindvall	2
Ioana Rus	2
Paolo Donzelli	2
Atif Memon	2
Marvin Zelkowitz	2
Chris Ackermann	2
Sima Asgari	2
Victor Basili	2
Lorin Hochstein	2
Roseanne Tvedt	2
Daniel Pech	2
Alexander Lam	2
Barry Boehm	2

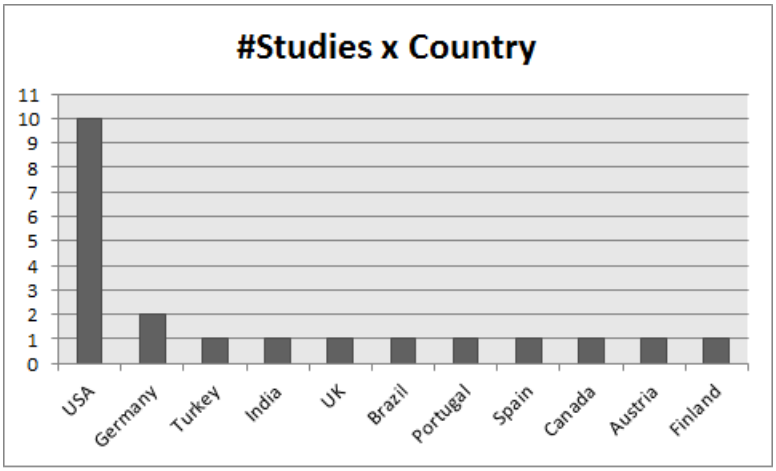

Figure 4.4 Distribution of studies by coutry.

Table 4.2 Evolution of the screening process.

Primary Studies Selection						
Sources	Studies Returned	#1 Selection	#2 Selection			
			Excluded			Included
		Potentially Relevant Studies	Irrelevant	Repeated/Duplicate	Incomplete	Primary Studies
IEEE	2141	22	16	1	0	5
ACM	100	13	8	4	0	1
Citeseer	393	17	12	2	0	3
Scopus	1507	38	28	10	0	0
Springer	97	15	11	0	0	4
ESEJ	1	1	0	1	0	0
TOTAL	4239	106	75	18	0	13

4.1.1 Relevance Evaluation

Using the definitions in Section 3.2.4, it was possible to grade each primary study regarding its relevance to the mapping. After this grading, the articles were classified in four different groups: *Very Good* (between 4.5 and 5.0), *Good* (between 3.0 and 4.0), *Regular* (between 1.5 and 2.5) and *Poor* (between 0.0 and 1.0). The results of this evaluation can be seen in Table 4.3. It is important to notice that none of the primary studies was classified as *Poor*. Each primary study was given a code, that can be seen in Table 4.3. In some sections of the current paper, individual studies can be referenced by this code.

4.2 EVIDENCE MAPPING

In this section the research questions are answered. In order to systematically answer thoses questions, particular sections of each primary studies were extracted and showed in the form of evidences. Section 4.2.1 maps the software engineering fields that are investigated using the concept of testbeds. Section 4.2.2 describes the components that the primary studies define as parts of a composed structure named testbed. Section 4.2.3 discusses the benefits claimed by the authors when using testbeds to evaluate technology in software engineering. Finally, Section 4.2.4 enlists the evaluation methods that were used by the authors to evaluate the testbeds that were proposed. From now on, for

Table 4.3 Results of the relevance evaluation.

Code	Primary Study	Ref.	Grade	Class.
PS01	A Requirements Engineering Testbed: Concept, Status and First Results	[31]	3.5	Good
PS02	Experimenting with Software Testbeds for Evaluating new Technologies	[14]	4.5	Very Good
PS03	An Evolutionary Testbed for Software Technology Evaluation	[13]	5.0	Very Good
PS04	Experiences in Developing and Applying a Software Engineering Technology Testbed	[19]	4.5	Very Good
PS05	Effect of Object Orientation on Maintainability of Software	[32]	2.5	Regular
PS06	Phase I Testbed Description: Requirements and Selection Guidelines	[7]	2.0	Regular
PS07	On the Contributions of and End-to-End AOSD Testbed	[33]	2.5	Regular
PS08	Communication Metrics for Sotware Development	[34]	3.5	Good
PS09	A Framework for Evaluating Software Technology	[1]	3.5	Good
PS10	Using Empirical Testbeds to Accelerate Technology Maturity and Transition: The SCRover Experience	[35]	3.5	Good
PS11	A Testbed for Analyzing Architecture Description Languages	[36]	2.5	Regular
PS12	A Family of Experiments to Investigate the Influence of Context on the Effect of Inspection Techniques	[2]	3.5	Good
PS13	Empirical Studies in ESERNET	[3]	1.5	Regular

the sake of clarity, an individual primary study will be reference by its code, which is comprised of the letters “PS” followed by a number indicating its order. Hence, in the current work, primary studies range from **PS01** to **PS13**. The primary studies and their correspondent code are at Table 4.3;

4.2.1 RQ1 - Research Topics

The objective of this question was to map all the software engineering research fields covered by research using testbeds. In total, 16 different topics were explored by the primary studies. The topics were identified based on the description given by the authors of the primary study. In general, authors are very clear when describing the topic the testbed is applied to. The amount of investigated research topics is greater than the total amount of primary studies because some studies study more than one research field. The most investigated topic was “Software Architecture”.

The evidences extracted from the primary studies are summarized in Table 4.4 and described in the remaining of this section.

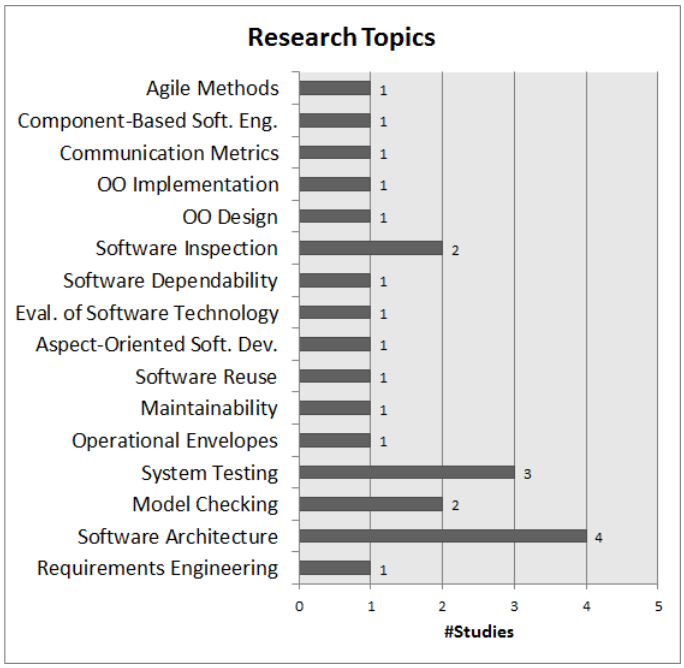

Figure 4.5 Distribution of studies by research topic.

Table 4.4 Summary of papers by research topic.

Research Topics	References - PS: Primary Studies	# of studies (%)
Requirements Engineering	PS01	1 (8%)
Software Architecture	PS02, PS03, PS04, PS11	4 (31%)
Model Checking	PS02, PS04	2 (15%)
System Testing	PS02, PS04, PS13	3 (23%)
Operational Envelopes	PS04	1 (8%)
Maintainability	PS05	1 (8%)
Software Reuse	PS06	1 (8%)
AO Development	PS07	1 (8%)
Evaluation of Software Technology	PS09	1 (8%)
Software Dependability	PS10	1 (8%)
Software Inspection	PS12, PS13	2 (8%)
OO Design	PS13	1 (8%)
OO Implementation	PS13	1 (8%)
Communication Metrics	PS08	1 (8%)
Component-Based Software Engineering	PS13	1 (8%)
Agile Methods	PS13	1 (8%)

Each research topic is be briefly described and the primary studies that are related to the topic are listed alongside with their correspondent evidences showing the relation.

Requirements Engineering

Requirements are used to express the stakeholders' needs in a given computer system. In general, requirements are precise enough so that the development team can implement a solution that will satisfactorily meet client's need. When requirements are well discussed and doubts cleared in this phase, the team can have savings of almost two orders of magnitude when compared to cases where the errors persist to other phases of development. The topics cited by the studies are highlited in bold in the extracted pieces of text from the original primaty studies.

The evidence that included the study in this topic is as follows:

- **PS01** — *"This paper describes a Requirements Engineering Testbed currently being developed and used by the Rome Air Development Center."*

Software Architecture

According to the IEEE standard named ANSI/IEEE 1471-2000, later adopted by ISO/IEC JTC1/SC7 as ISO/IEC 42010:2007, *"Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution".*

Next the reader can see the evidence that included the studies in this topic:

- **PS02** — *"The technology discussed in this paper is of the latter kind and is called Implementation-oriented software architectural evaluation."*
- **PS03** — *"The particular technology chosen for initial experimentation is called software architecture evaluation (SAE)."*
- **PS04** — *"The technologies evaluated came from the fields of* **architecture***, testing, state-model checking, and operational envelopes."*
- **PS11** — *"The two major objectives of the task are to develop (a) an initial testbed for analyzing ADLs ..."*

When the authors use testbeds to study Software Architecture, the Software Architecture Evaluation (SAE) subtopic appears in three of four papers (PS02, PS03, PS04). The other primary study (PS11) uses a testbed to evaluate ADLs (Architecture Description Languages).

System Testing

System testing (either of hardware or software) is a discipline that is capable of checking whether the product is compatible with the system requirements that were previously contracted by the clients. Usually, the tests are conducted in a controlled environment, where the system is expected to run. By following test scenarios, which are a kind of script, test analysts can assert the product compliance. The primary studies that were included in this category are shown next:

- **PS02** — *"We are expanding the testbed to allow for experiments using standardized execution-based technologies, i.e., software testing."*
- **PS04** — *"The technologies evaluated came from the fields of architecture,* **testing***, state-model checking, and operational envelopes."*
- **PS13** — *"ESERNET members have been performing empirical studies in six research areas: Inspections,* **Testing***, Object-Oriented design, Object-Oriented implementation, Component-based Software Engineering, and Agile methods."*

Operational Envelopes

Operational Envelope is a concept related to system dependability. An operation envelope is defined as the environment or conditions in which the system will work dependably. In other words , the system can be garanteed to work properly if and only if the system runs in this previously "contracted" environment. In this sense, testbeds are used to develop a technology that aims to help to identify operational envelopes prior to system execution and report environmental conditions in which the system will fail. The primary study that investigates such topic is shown next:

- **PS04** — *"The technologies evaluated came from the fields of architecture, testing, state-model checking, and* **operational envelopes***."*

Maintainability

Software maintenance activities are defined as any activity that is done after the software product has been delivered. Maintenance costs can easily surpass the whole development cost. Hence, studies that try to investigate how well a given approach or technology performs in maintenance activities are very important. In the current mapping study, a primary study that addresses theses issues was selected. The evidence is:

- **PS05** — *"In this report, we describe an effort to experimentally study the maintainability of object oriented software, as compared to non-OO software."*

Software Reuse

According to Krueger [37], "software reuse is the process of creating software systems from existing software rather than building software systems from scratch". In the same work, Krueger discusses some approaches that can be used to employ software reuse. But, how can one evaluate whether one approach is better than other in a given scenario? Detailed evaluation, with a set of carefully selected criteria must be defined in order to have a fair and sound evaluation. The evidence that shows the study belonging to this category is:

- **PS06** — *"The primary objective of the Reuse Project testbed is to provide the physical framework to acquire, use, and evaluate existing reusable components. The testbed will also support executing a set of carefully controlled reuse experiments on problems of realistic size and complexity and gaining experience and understanding of software reusability. The elements of the Reuse Testbed are hardware, software tools, a software development environment, a library for existing reuse parts, and a candidate development problem. This paper will establish the guidelines for selecting the elements of the testbed that serve the above purposes."*

Aspect-Oriented Software Development

Aspect-oriented software development (AOSD) is emerging as a new programming paradigm [16]. AOSD techniques are gaining increased attention from both academic and industrial institutions. In order to facilitate the adoption of such techniques it is paramount that empirical studies demonstrate the benefits (and drawbacks) of this new approach. The study selected by this mapping study is depicted next, alongside with its evidence:

- **PS07** — *"As such, this paper outlines our initial effort on the design of a testbed that will provide end-to-end systematic comparison of AOSD techniques with other mainstream modularization techniques."*

Evaluation of Software Technology

The evaluation of software technology is very important in organizations that are concerned with the impacts of decisions regarding, for example, the selection of tools and their application. Just to mention a few, impacts may include [1]:

- Initial technology cost acquisition;

- Long-term effect on quallity, time to market, and cost of the organization's products and services when using the technology;
- Training and support services' impact of introducing the technology.

The study selected by this mapping study is depicted next, alongside with its evidence:

- **PS09** — *"The authors' experimental framework can help companies evaluate a new software technology by examining its features in relation to its peers and competitors through a systematic approach that includes modeling experiments."*

Sotware Dependability

Dependability is the ability that a system has to be reliable and trustful. As opposed to what one might think, systems are not dependable when users' needs are met, but when they never do something that could not be predicted. Dependability is not always needed. Usually, mission critical systems enforce this characteristic, as in space missions. The study selected by this mapping study is depicted next, alongside with its evidence:

- **PS10** — *"This paper is an experience report on a first attempt to develop and apply a new form of software: a full-service empirical testbed designed to evaluate alternative software dependability technologies, and to accelerate their maturation and transition into project use."*

Software Inspection

Software inspection is an activity characterized by the execution of a well defined process by trained individuals that look for defects in software work products. Inspection is one of the most used quality focused techniques employed by industry. The main objective of inspections is to reach a consensus on given deliverable and decide wether it is approved for use or not. The studies selected by this mapping study are depicted next, alongside with their evidences:

- **PS12** — *"In designing the experiment family, we selected software inspections as the specific software development technology to serve as testbed for our experimental method."*
- **PS13** — *"ESERNET members have been performing empirical studies in six research areas:* **Inspections***, Testing, Object-Oriented design, Object-Oriented implementation, Component-based Software Engineering, and Agile methods."*

OO Design

The object-oriented paradigm is currently the dominating paradigm in software systems engineering. In the past, a software project did not have the same formal rigour that the project of a bridge had, for example. OO design methods have emerged to try to address this problem. Empirical studies are important to investigate approaches that can improve the design of object-oriented systems. The study selected by this mapping study is depicted next, alongside with its evidence:

- **PS13** — *"ESERNET members have been performing empirical studies in six research areas: Inspections, Testing,* **Object-Oriented design***, Object-Oriented implementation, Component-based Software Engineering, and Agile methods."*

OO Implementation

Even though a system has been well designed, if it has a poor execution, the results my be not satisfactory. In this sense, some methodologies have emerged to reduce the risks of the implementation phase. The study selected by this mapping study is listed next, alongside with its evidence:

- **PS13** — *"ESERNET members have been performing empirical studies in six research areas: Inspections, Testing, Object-Oriented design,* **Object-Oriented implementation***, Component-based Software Engineering, and Agile methods."*

OO Communication Metrics

Communication plays an important role in software development. A team that does not communicate well is doomed to failure. Communication can establsihed through many different mechanisms, such as emails and memoranda. Communication can be even harder when projects are bigger and involve more people. The primary study selected by this mapping study developed a testbed to evaluate communication metrics. It is evidenced below:

- **PS08** — *"We then define communication metrics more formally and present a test-bed for their design and experimentation."*

Component-Based Software Engineering

The component-based software engineering (CBSE) research field aims to improve quality and productivity of software systems by separation of concerns in the form of

components. This is difficult specially to small to medium sized companies, which tend not to have a well defined process. In order to achieve that, many tools, methods and techniques have been developed to aid CBSE [3]. The study selected by this primary study is depicted below, alongside with its evidence:

- **PS13** — *"ESERNET members have been performing empirical studies in six research areas: Inspections, Testing, Object-Oriented design, Object-Oriented implementation,* **Component-based Software Engineering***, and Agile methods."*

Agile Methods

Research on agile methods is still in its beginning. The agile ideas came from industry, which had problems with requirements changing too often, technologies that evolved very rapidly and short time-to-market [3]. To achieve that, several methods were developed based on an expert experience base, which led to a handful of good practices that form the foundations of these methods. The study selected by this primary study is depicted below, alongside with its evidence:

- **PS13** — *"ESERNET members have been performing empirical studies in six research areas: Inspections, Testing, Object-Oriented design, Object-Oriented implementation, Component-based Software Engineering, and* **Agile methods***."*

4.2.2 RQ2 - Testbed Components

The objective of this question is to query the literature to discover what the studies that use testbeds as tools define as elements of such testbeds. In a prior informal review, most studies diverged when defining the components of the testbed. It is important to researchers to understand which are the most used elements and how they are employed in case new testbeds are to be built. Yet, it is also important to gather knowledge in case it is necessary to create entirely new elements.

Among other things, testbeds are very specific guidelines to perform empirical studies. They are tailored to accomplish particular objectives, and as so, they can be very different from each other. But in essence, they can be divided in two major groups: the ones that list elements as activities and the ones that list elements as generic blocks.

Testbeds that list elements as activities will be referenced in the text as Step-Based Testbeds (SBT). They are straightforward and easier to understand. In general, these testbeds also include a series of assets that are used throughout the execution of the

study. These assets may be pieces of source code, documents or a set of seeded faults to be intentionally added to the source code. A type of asset can be present or not, depending on the intentions of the testbed designer or what the testbed aims to investigate. The main difference to the second type of testbeds is that the first one does show ordered steps.

Testbeds that list elements as generic blocks will be named Block-Based Testbeds (BBT) hereafter. In general, they are a bit harder to understand in comparison to SBTs, since a list of steps is usually more natural to follow. They show, however, generic blocks. These blocks can contain several kinds of assets, like the ones described in the latter paragraph. In some cases, the blocks are linked together, so the reader can have better idea of how they relate and how assets present (or created) in a different block can be used in another block.

As previosly mentioned, testbeds are inherently very different if they aim to investigate different factors. The elements defined by each primary study will be detailed next, grouped by the types defined in this section (SBT and GBT).

Step-Based Testbeds (SBTs)

Step-based testbeds are basically, as mentioned earlier in this section, an ordered set of steps. The primary studies that were grouped in this set are:

- **PS03** — *For building, using, and evolving the testbed we performed the following activities:*

  1. *Select application in relevant domain.*
  2. *Select family of technologies.*
  3. *Select one technology within the family.*
  4. *Prepare artifacts necessary for experimentation.*
  5. *Conduct study to create a testbed baseline.*
  6. *Define faults to be seeded.*
  7. *Design and conduct experiment.*
  8. *Analyze and document results from experiment.*
  9. *Improve testbed based on analysis.*
  10. *Improve technology based on analysis.*
  11. *Verify testbed usefulness.*

- **PS11** — *Our testbed development approach involves four major activities which reflect these guiding principles. These activities are:*

  1. *Establish criteria to use as a basis for assessing key ADL properties.*
  2. *Develop scenario-based methods for objectively applying the criteria to ADLs to determine their properties.*
  3. *Select a representative set of ADLs and supporting tools to analyze, and obtain and install the tools to support the hands-on, scenario-based analyses.*
  4. *Apply the methods and criteria to the selected ADLs and document the results and lessons learned.*

- **PS13** — *The execution of each empirical study was structured according to the widely accepted Quality Improvement Paradigm (QIP). The Quality Improvement Paradigm is a comprehensive framework for systematically managing improvement by using experiments and measurement. Quality Improvement Paradigm consists of six steps that are repeated iteratively:*

  1. *Characterise. Understand the current situation and establish a baseline.*
  2. *Set goals. Quantifiable goals are set and given in terms of improvement.*
  3. *Choose process. Based on the characterisation and the goals, choose the appropriate processes and supporting methods and tools.*
  4. *Execute. The study or project is performed and the results are collected for evaluation purposes.*
  5. *Analyze. The outcome (current practices, possible problems, and findings) is studied and future possible improvements are identified.*
  6. *Package. The experiences are packaged to build the basis for future improvement projects.*

Block-Based Testbeds (BBTs)

In this category, it is harder to identify the elements of the testbed. In this case, it is not always found a clear definition of the elements. For every primary study, evidences are presented and the elements are highlighted in the text. The primary studies that were classified as block-based testbeds are shown next:

- **PS01** — *The testbed is based on a new* **process model for requirements engineering activities** *which provides a detailed description of the fundamental activities occurring during requirements engineering and their relationships to design activities. The testbed's current implementation includes a* **requirements analysis method** *and* **two requirements validation techniques based on rapid protoytyping and simulation**.

- **PS02** — *The testbed is a* **set of programs**, **data**, *and* **supporting documentation** *that allows researchers to test their new technology on a standard software platform. An important component of this testbed is the* **Unified Model of Dependability (UMD)**, *which was used to elicit dependability requirements for the testbed software. With a* **collection of seeded faults and known issues of the target system**, *we are able to determine if a new technology is adept at uncovering defects or providing other aids proposed by its developers.*

- **PS04** — *A brief description of each component in the testbed's architecture is given below along with why the component is important to the architecture.*

  - **Instrumentation** — *an instrumentation class will help the evaluators collect data for their evaluation report. Without this feature, an evaluator will have to spend more time in figuring out how to collect the data they need.*

  - **Seeded Defect Engine and Defect Pool** — *seeded defects will help the evaluators estimate how well their technology finds defects and it prevents evaluators from building their technology to just pass a certain test or operational scenario. In addition, seeded defects give a measurement to evaluators on how well their technologies can find defects and what kind of defects the technology can and cannot find.*

  - **Code** — *a software system is needed to evaluate the technology on.*

  - **Specifications** — *specifications are needed if the organization wishes to test more than just technologies that work on code. Specifications provide the basis for the evaluation, along with the code. For example, architecture specifications are needed for architecture technology evaluations. Without specifications, evaluators whose technology requires specifications will not be able to use the testbed or it will increase the amount of time the evaluator conducts the evaluation since they will have to produce the specifications on their own. Thus, without specifications, the scope and usefulness of the testbed will be limited.*

  - **Mission/Scenario Generator** — *allows evaluators to do a more thorough evaluation of their technology since it will be operating under several scenarios and not just one. Also, prevents an evaluator from developing their technology to work under a specific circumstance.*

- **PS05** — *One approach for experimental evaluation is to have* **testbed of many problems** *where for each problem both OO software and regular software (i.e. function oriented software) have been developed. That is, for a given specification, there are* **two software - one OO and one regular - that have been developed** *to satisfy the* **specifications**. *Once the testbed is there, then the maintenance experiments can be conducted by changing the specifications and then independently making changes to both object oriented software and regular software and evaluating the maintenance exercises through the use of* **some metrics**.

- **PS06** — *The testbed's* **hardware and system software** *will provide a distributed computing environment with file-server capability for the storage of reusable components and other artifacts of the development process. These requirements also address system and software test capabilities. The* **software development environment** *of the testbed will support all phases of the development and have an integrated software engineering database for the storage and retrieval of reusable components and newly developed software engineering artifacts. The testbed will support* **a variety of reusable components**, *domain-independent and domain-dependent. Example domains include data structure packages, missile navigation components and tools, avionic designs, user interface tools, and signal processing subsystems. The testbed requirements are general, because reuse has a large intersection with environments, methods, and tools for software engineering. The* **guidelines for selecting the testbed hardware and software** *have been derived from the requirements. A testbed must be an open system into which one can integrate new tools and techniques as they become available. It should also be representative of the facilities available to MCCR contractors today.*

- **PS07** — *The testbed itself consists of three core elements: (i)* **Applications** *that contain a variety of crosscutting concerns; (ii)* **AO and non-AO approaches** *that are applied to a common application to generate artifacts; (iii)* **a suite of metrics** *associated with a variety of internal and external software attributes; and (iv)* **a set of metric results** *that have been gathered from applying the metric suites mentioned in (iii) to the artifacts produced.*

- **PS08** — *To investigate the value of communication metrics, we have used as a test-bed* **the senior level software engineering project courses at Carnegie Mellon University** *(15-413 and 15-499). In these project courses, students are taught software engineering by working on a* **real project**, *with* **real clients** *and* **real deadlines**. *They are organized into teams that need to cooperate for the project to succeed. Such a test-bed represents an ideal environment for experimental software engineering, providing sufficient realism while allowing for controlled observation of important project parameters.*

- **PS09** — This primary study [1] did not provide a brief piece of text detailing its testbed. Instead, it provided a figure, which is presented as Figure 4.6.

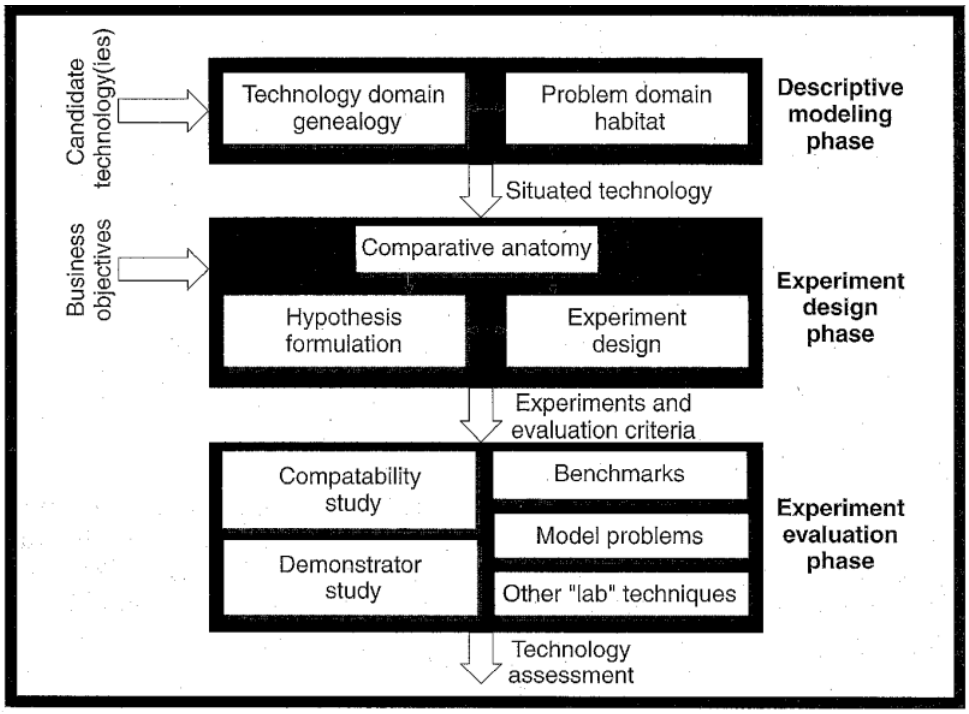

Figure 4.6 Elements of the testbed proposed by Brown and Wallnau. Extracted from [1].

- **PS12** — Differently from the latter, the current primary study did provide a textual definition, but it is far too long. The blocks defined by this study can be summarized by the Figures 4.7 and 4.8.

- **PS13** — This primary study is very peculiar since it belongs to both categories, providing activities and blocks. The BBT part of this study is depicted in the right

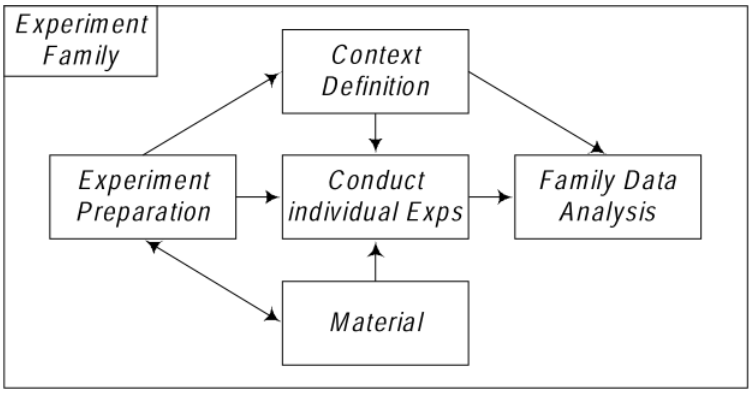

Figure 4.7 Elements of the first part of the testbed proposed by Ciolkowski, Shull and Biffl. Extracted from [2].

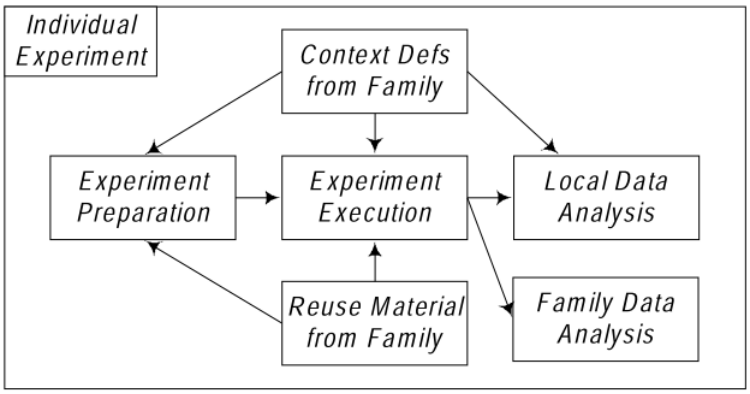

Figure 4.8 Elements of the second part of the testbed proposed by Ciolkowski, Shull and Biffl. Extracted from [2].

side of Figure 4.9, stressed by a red square and named ESERNET Project. As the reader may see, the left side of the figure shows the SBT part of this testbed.

4.2.3 RQ3 - Benefits of Using Testbeds

The objective of this question is to map all the benefits researchers might have when using testbeds as a tool to evaluate technologies. There were a total of 19 benefits cited by the authors of the primary studies. Most papers have claimed more than one benefit from using testbeds. Curiously, only one primary study (PS05) provides no benefits from using testbeds. Table 4.5 lists all the benefits claimed by the authors of the primary studies. For each benefit, are presented the papers that have concluded such benefit, alongside with its count and percentage in relation to the full set of the 13 selected studies.

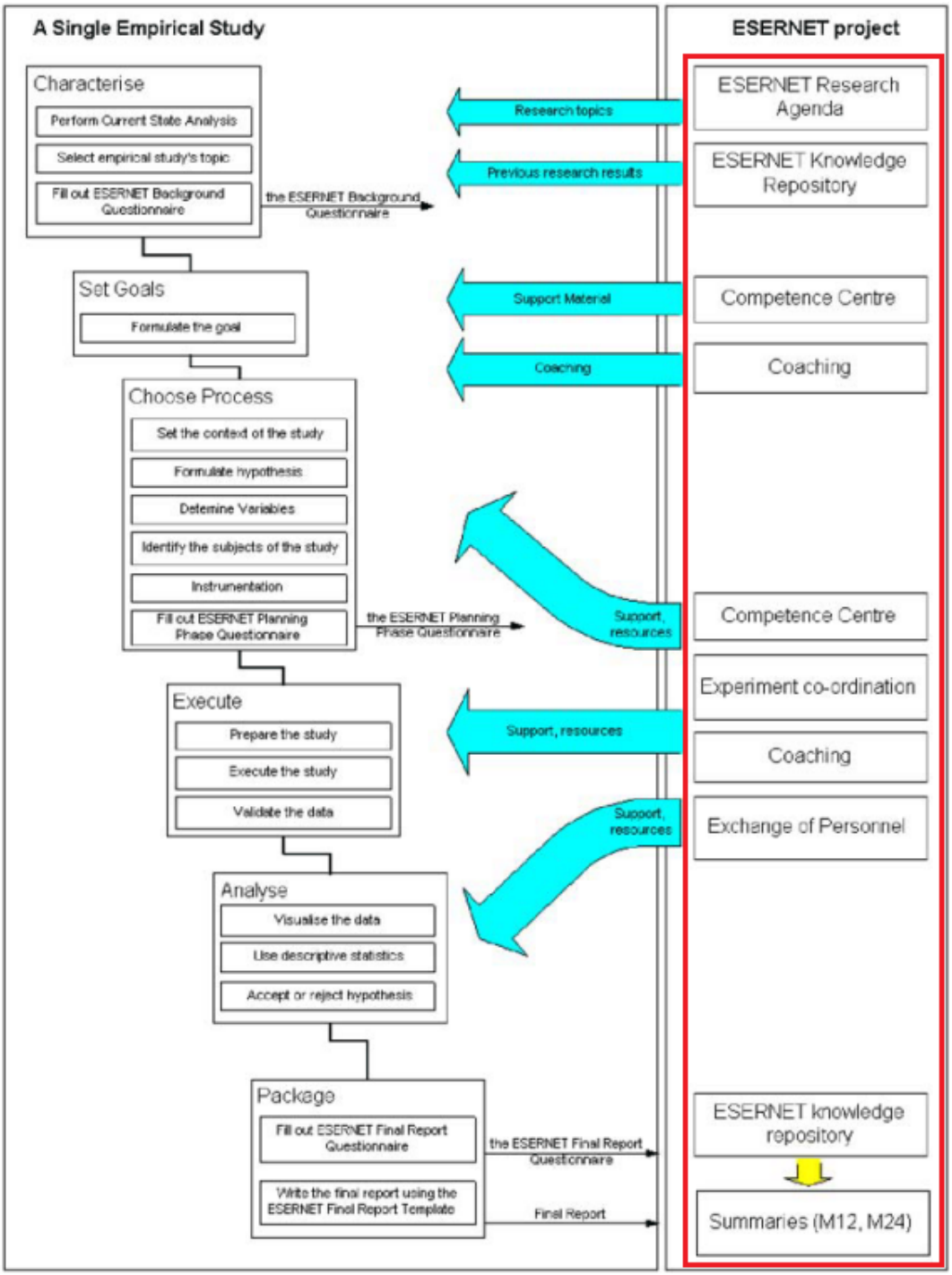

Figure 4.9 Elements of the testbed proposed by Sandelin and Vierimaa. Extracted from [3].

It is clear that the use of testbeds can make empirical evaluation of technologies easier. Five studies have reported that the use of testbeds reduces the experimentation cost.

This is due to the fact that when a testbed is built, many artifacts are created, such as experimental plans, requirements documents or a pool of seeded defects. Naturally, the artifacts generated differ from one study to another, but in essence, these artifacts can be reused later in other studies, be it a replication of the same study or not. Besides, the studies also claim that the use of testbeds can ease results comparison, fostering replication and facilitating the technologies comparison. These findings show the importance of testbeds in software engineering.

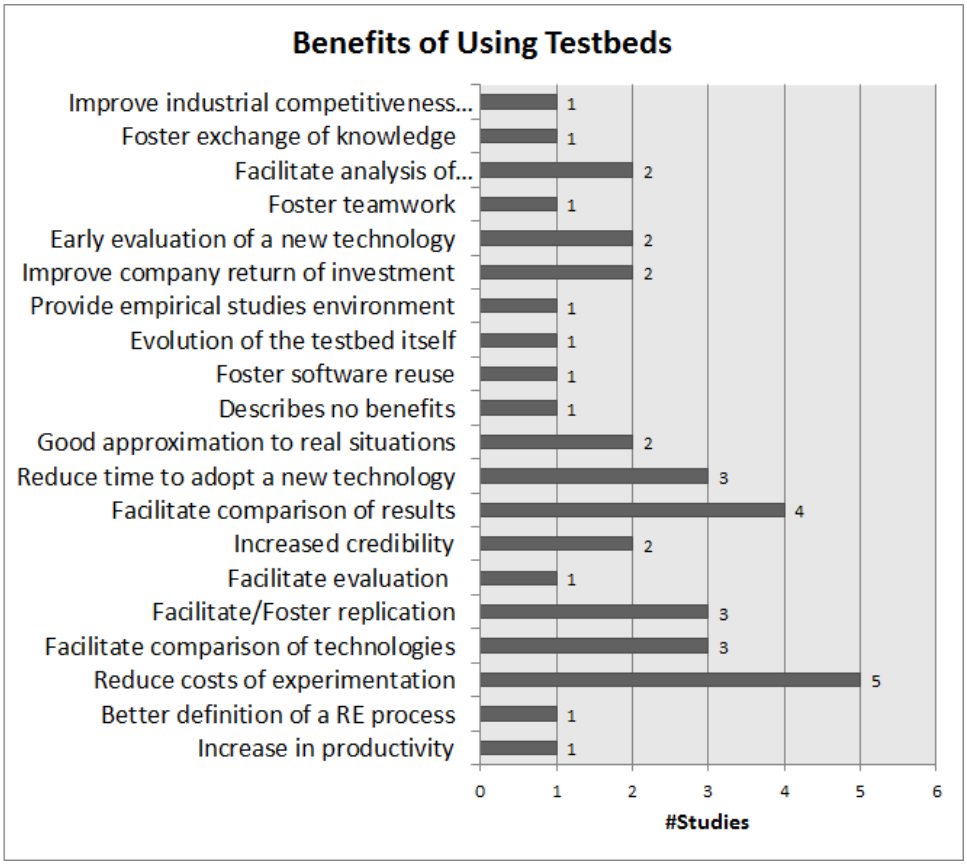

Figure 4.10 Benefits described by the selected studies.

4.2.4 RQ4 - Evaluation Methods

This question aims to identify the scientific methods employed to evaluate the proposed testbeds. It is important to identify the methods/techniques employed by the authors to evaluate the testbeds because this might be valuable information to researchers that intend to build their own testbed. The results from the current analysis will help us to better plan our evaluation strategy.

Table 4.5 Summary of the benefits from using testbeds

Benefits	References	# of studies (%)
Increase in development productivity	PS01	1 (8%)
Better definition of a RE process	PS01	1 (8%)
Reduce costs of experimentation	PS02, PS03, PS04, PS11, PS12	5 (38%)
Facilitate comparison of technologies	PS02, PS07, PS11	3 (23%)
Facilitate/Foster replication	PS02, PS03, PS13	3 (23%)
Facilitate evaluation	PS02	1 (8%)
Increased credibility	PS03, PS12	2 (15%)
Facilitate comparison of results	PS03, PS04, PS11, PS12	4 (31%)
Reduce time to adopt a new technology	PS03, PS04, PS10	3 (23%)
Good approximation to real situations	PS04	1 (8%)
Foster software reuse	PS06	1 (8%)
Evolution of the testbed itself	PS07	1 (8%)
Provide empirical studies environment	PS08	1 (8%)
Improve company return of investment	PS09, PS10	2 (15%)
Early evaluation of a new technology	PS10, PS11	2 (15%)
Foster teamwork	PS10	1 (8%)
Facilitate analysis of complementarities among technologies	PS10, PS11	2 (15%)
Foster exchange of knowledge	PS13	1 (8%)
Improve industrial competitiveness and innovation potential	PS13	1 (8%)

Only two papers from the set of primary studies did not provide evaluation of the testbed (PS07 and PS08). Most of the remaining primary studies employed more than one evaluation strategy. Figure 4.11 show the methods employed by the authors to evaluate the testbeds. Table 4.6 shows the mapping of each paper regarding the evaluation methods used. Primary study PS13 has used the post-mortem evaluation method. This method is characterized by project documentation evaluation and interview with the subjects of the studies being analyzed [3]. The evidences are shown in Table 4.6.

4.3 ANALYSIS AND DISCUSSION

Despite the small number of selected primary studies, the current mapping study was able to find evidence to answer the research questions. In terms of quality, more than

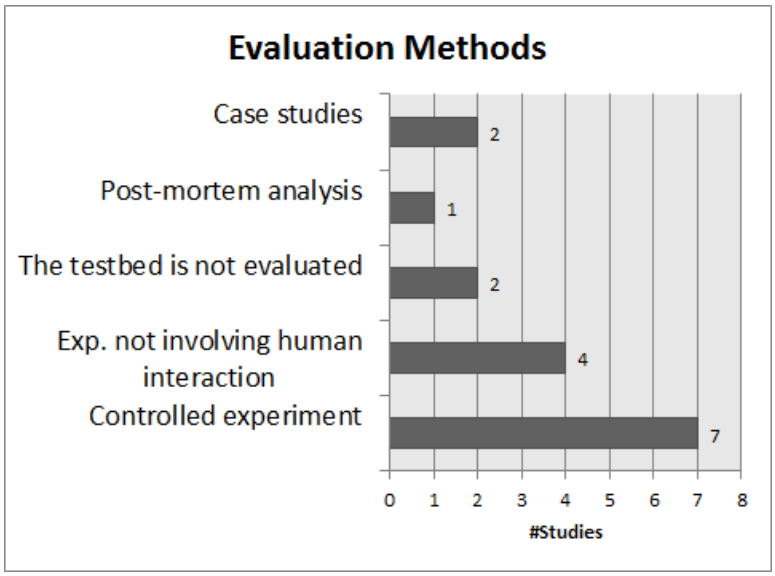

Figure 4.11 Methods used to evaluate the testbeds and their occurrence.

Table 4.6 Summary of evaluation methods.

Evaluation Methods	References	# of studies (%)
Controlled experiment	PS01, PS02, PS05, PS06, PS09, PS12, PS13	7 (54%)
Exp. not involving human interaction	PS02, PS03, PS04, PS10	4 (31%)
Post-mortem analysis	PS13	1 (8%)
Case studies	PS08, PS13	2 (15)%

half of the papers scored *Good* or *Very Good* (8 of 13, 61.54%).

The mapping study identified 16 different research topics in which testbeds are actively used as an empirical evaluation tool. This shows how versatile testbeds can be and how easily the concept can be applied to evaluate technologies in a great number of different scenarios. This is also supported by the fact that even in an individual study, some testbeds proposed were used to investigate more than one research topic.

When mapping the elements the studies described as building blocks of the testbeds, the authors diverged the most. This is perfectly understandable, however, because the testbeds are to be applied to evaluate very different technologies. Yet, in some cases testbeds can be one part of a bigger process. In those cases, testbeds can be tailored to better fit the process, what has influence in the testbed's organization.

Regarding the benefits the use of testbeds enables, the studies have agreed to some extent. In essence, testbeds can be taken as guidelines to the execution of empirical studies that try to evaluate technologies in software engineering. Half (10 of 20) of the benefits claimed from the studies have been cited by more than one paper and many of the benefits make the empirical environment in which the studies are run stronger.

The following can be highlited:

- Reduce costs of experimentation;
- Facilitate/Foster replication;
- Facilitate evaluation;
- Facilitate comparison of results;
- Good approximation to real situations;
- Provide empirical studies environment;
- Foster excahnge of knowledge;
- Evolution of the testbed itself.

The highlited benefits are directly related to the improvement of the empirical paradigm by providing better tools and granting greater power to researchers that take advantage of testbeds in software engineering.

Only two of the 13 papers do not evaluate the testbeds they propose. *Controlled Experiment* was the most used method to evaluate the testbeds, which indicates the preference of the community for such approach. As an example, PS01 performed a controlled experimento to evaluate a requirements engineering testbed. In this study, two experiments were carried. The experiments consisted of developing two system prototypes. In the first experiment, analysts were 3.5 faster than they would be, if conventional techniques were used. On the second experiment, the factor was 6 times faster. The experiment design was not shown, and the author argues that the increase in productivity from the first to the second experiment was due to the learning curve of the techniques used in the testbed.

It is important to know that such methods have been successfully applied to testbeds evaluation because researchers that intend to build their own testbeds can take lessons learned described by the studies and avoid the most common mistakes. It is also important to stress that testbeds seem to be better applied to only a subset of empirical

studies. The selected primary studies do not report any use of testbeds to aid ethnographies, action-resaerch, surveys or quasi-experiments. Even though it seems hard to use testbeds to aid ethnographies and surveys, further investigation is needed to argument the use of testbeds to kinds of empirical studies other than controlled experiments and case studies.

4.4 TOWARDS A SOFTWARE MAINTENANCE TESTBED

From the analysis of the papers selected by the mapping study, it is possible to conclude that testbeds are empirical environments, mostly used to aid researchers in conducting experiments and case studies in SE. An ongoing study or our research group is to develop a software maintenance testbed. In order to achieve that, several elements must be built and integrated. Initial ideas for the software maintenance are presented next.

The study proposed the following elements:

- Assets repository;
- Guidelines;
- Metrics suite;
- Knowledge base;
- Applications;
- Information system.

The assets repository is important to keep all the assets that are used in the research. Several assets are created during the definition, planning, operation, interpretation and packaging phases. Such assets my include experimental plans, use case and architecture documents, manuals and so forth.

Guidelines can help the researcher in selecting research methods that are better suited for the objectives of the study. It is intended for the testbed to contain guidelines for performing controlled experiments, quasi-experiments, case studies, action research, surveys and ethnographies. Adauto Almeida identified that there are several types of mechanisms to guide empirical studies in software engineering [38]: guidelines, lessos learned, frameworks, methods, processes, paradigms, techniques, and templates.

The concept of measurement is tightly coupled with the testbed concept of since one of the main objectives of testbeds is to allow systematic comparison of technologies. To

define a metrics suite is important to establish a good base for comparison. If two or more studies evaluate different, but competitive, technologies, they can only be compared if they have collected the same measures. A pre-defined metrics suite also reduces the time spent by researcher when they need to select metrics for their particular studies. To effectively allow the use of a metrics suite, all metrics must be carefully specified. An ongoing study is conducting a systematic mapping study to identify metrics specifics to software maintenance.

The knowledge base is one of the most important elements of the testbed. Every study executed using the testbed must be returned to the tesbed to compose the knowledge base. The experience obtained while using the testbed is a valuable information to aid researchers in future studies and also help to enhance the testbed itself.

Every empirical study in SE must use at least one application in some phase of the study. Some studies measure the software, other studies measure communication among participants, for example, but all of them need applications. Marcelo Moura, in his Master's Thesis [39] defined several attributes to classify applications and maintenance scenarios, which aid researchers in selecting applications to perform empirical studies.

Several elements and phases of the testbed can be easier performed with the help of an information system (IS). For example, the applications can be easily kept in the IS and a search mechanism can be implemented to automate the query for a specific application. The assets can be hosted in the IS, kept grouped by study and versioned, among other uses.

4.5 SUMMARY

The search process returned a total number of 4239 papers. From that amount, 13 primary studies were selected. IEEE and Sprinter were the sources that had more relevant studies selected (5 and 4 primary studies, respectively). The studies provide evidence that testbeds have been successfully used to the evaluation of technologies in SE. From the 74 different authors that have been identified, 14 have published more than one paper on the subject.

The primary studies investigate 16 different research topics. The most investigated topic was "Software Architecture", with 30.77% of the studies. Regarding the composing elements of testbeds, the studies can be grouped under categories: Step-Based Testbeds (SBT) and Block-Based Testbeds (BBT). The main difference between the categories is that SBTs provide elements as numbered activities, while BBTs provide a block structure.

The primary studies claim a total of 20 differents benefits of using testbeds to eval-

uate technologies in SE. Nearly half of the benefits are claimed by at least two studies, which gives more confidence to the conclusion that testbeds are effective to evaluate SE technologies.

Researchers tend to use controlled experiments as the main evaluation method of SE testbeds. Since SE is mainly based on human activities and interactions, controlled experiments provide tools to isolate variables and better analyze results. Experiments not involving human interaction also appeared as an important method. They were mainly employed when the study tried to evaluate the performance of tools that were applied to assets produced during the development process.

CHAPTER 5

FINAL CONSIDERATIONS

In this chapter, the final considerations of the study are presented. Some threats to validity are discussed.

5.1 THREATS TO VALIDITY

The first threat to validity concerns the search strategy employed. Since we mainly used automated search engines, relevant studies may not have been included in the set of selected studies. Even though we dedicated some time to identify relevant keywords, a particular study that used a different term might be missing. Some more recent studies may be missing because the search engines may not have indexed them. Yet we tried to reduce this risk by manually searching the last proceeding of the following conferences: *Empirical Software Engineering and Measurement* (ESEM'09), *Evaluation and Assessment in Software Engineering* (EASE'10) and *Internation Conference on Software Engineering* (ICSE'10). We also manually searched the last 12 months of the *Empirical Software Engineering Journal* (from June'09 to June'10).

The small number of selected papers is also a threat. Even though the automated search has been executed without any time limitation, only 13 papers were considered relevant. Despite the small set of primary studies, more than half of them were classified as, at least, *Good* papers after the quality evaluation.

The quality assessment process was not as performed, as proposed by Kitchenham and Charters [23]. Rather then evaluating the validity and bias in the selected work, we tried to evaluate the relevance of the studies considering the mapping goals. The conduction of quality analysis defines an extension of the current study.

The data was extracted by only one researcher, which configures a threat [23]. However, this phase was supervised by the student's advisor, and eventual checks have been performed by another researcher of the group, who also worked in other phases of the mapping.

5.2 FUTURE WORK

The results achieved by this mapping study will help our research group build a new testbed for the evaluation of software maintenance technologies. The testbed will help reseachers in empirical studies, from planning to analysis, packaging, and dissemination. Another researcher, from the same research group has undertaken a systematic mapping study to identify the existing guidelines to empirical SE studies. The objective of these guidelines is to provide valuable aid to researchers that need to plan empirical studies in SE and try to establish a stepping stone in these activities for the testbed. Guidelines to controlled experiments, surveys and case studies have been identified by the mapping study. The identification of the most used metrics for aspect-oriented software maintenance is being conducted by a third researcher. With a set of specalized metrics, researchers will be able to choose the ones that best suit their work. Moura [39] has proposed a benchmarking framework to help researchers classify and select applications to use in empirical studies on aspect-oriented software maintenance. Several aspects of applications is taken into account, such as version, presence of source code, available versions, just to name a few. With the results found by the mentioned studies, the group is able to build a much more robust testbed by utilizing the "best practices" reported by the SE community.

Event though the mapping study identified 13 primary studies that describe the definition and use of testbeds in SE, none of them is concerned with running the evaluations in a corporate environment. Organizations have difficulty in adopting new technologies. In some cases, a technology can take up to 20 years to be widely adopted [20].

Currently, the technology transition is an ill-defined, non-repeatable and inefficient process [40]. The need for the sharing of empirical knowledge in SE has already been stressed [41], but researchers constantly face two problems: (1) How to encourage researchers to use materials provided by other researchers? (2) How to encourage researchers to make material available to others in an appropriate form? These problems are even bigger when the corporate environment is taken into account. In many cases, industrial secrets can make the difference in being the leading company in the market and failing. The use of managerial frameworks to run in-company evaluations, however, can be of great help to reduce the time to adopt new technolgies. This appears to be a good opportunity to expand the use of testbeds beyond academia.

As a means to improve the current mapping, backward search will be performed. This activity searches for relevant papers in the references of the primary studies. This allows us to cover a large set of directly relevant papers, in which the probability of

finding relevant papers is higher. We also intend to search manually in more issues of the Empirical Software Engineering Journal and in more proceedings of the conferences to try to reduce even more the risk of automated search.

A more thorough quality assessment will be performed. Publication bias and validity of the selected studies will be evaluated according to the guidelines proposed by Kitchenham and Charters [23].

5.3 CONCLUSIONS

This mapping attempted to perform a systematic literature review for studies that report the development of testbeds in software engineering. After the initial evaluation of 4239 papers, this mapping study selected 13 relevant primary studies. Using the four research questions, we were able to satisfactorily find evidences that showed testbeds as an important and efficient tool to empirically evaluate technologies in software engineering as well as examples of the elements that a testbed encompasses.

The mapping process was reported, which makes it easier for other researchers to evaluate the current study. The Scopus search engine presented a good rate of relevant papers. From the 38 candidate studies selected, 10 papers were found relevant, the highest absolute amount. However, the other search engines also returned all relevant papers returned by Scopus. It was preferred, hence, to attribute these papers to their original publisher (IEEE, ACM and Springer). This is interesting because in the future, mechanisms like Scopus can be even more powerful and concentrate all the queries needed in only one place, reducing the researchers' effort in this step of the study.

The vast majority of papers (12 of the 13) have described several benefits of using testbeds to evalute technologies in software engineering. Many of the papers have reported similar benefits. This shows that the community is consonant with improvements the use of testbeds can bring to the empirical paradigm.

Even though the use of testbeds can have a positive impact in the results of empirical studies in software engineering, this study showed that they are not being well explored by the community. One of the most important goals of this study is to disseminate the testbed concept allowing a greater number of researchers to take advantage of software engineering testbeds.

APPENDIX A

INCLUDED PRIMARY STUDIES

The information presented in this appendix is detailed in Section C.7 of Appendix C. Because of space limitations, the only information displayed is **ID**, **year**, **source**, **author** and **title**. The reader can check the full reference in the bibliography of this work.

ID	Year	Source	Reference
PS01	1989	IEEE	William E. Rzepka. *A requirements engineering testbed: concept, status and first results* [31]
PS02	2007	Springer	Mikael Lindvall, Ioana Rus, Paolo Donzelli, Atif Memon, Marvin Zelkowitz, Aysu Betin-Can, Tevfik Bultan, Chris Ackermann, Bettina Anders, Sima Asgari, Victor Basili, Lorin Hochstein, Jörg Fellmann, Forrest Shull, Roseanne Tvedt, Daniel Pech, Daniel Hirschbach. *Experimenting with Software Testbeds for Evaluating new Technologies* [14]
PS03	2005	Springer	Mikael Lindvall, Ioana Rus, Forrest Shull, Marvin Zelkowitz, Paolo Donzelli, Atif Memon, Victor Basili, Patricia Costa, Roseanne Tvedt, Lorin Hochstein, Sima Asgari, Chris Ackermann, Dan Pech. *An evolutionary testbed for software technology evaluation* [13]
PS04	2009	Springer	Alexander Lam, Barry Boehm. *Experiences in Developing and Applying a Software Engineering Technology Testbed* [19]
PS05	1997	IEEE	G. Aditya Kiran, S. Haripriya, Pankaj Jalote. *Effect of Object Orientation on Maintainability of Software* [32]
PS06	1988	IEEE	Robert Holibaugh, J. M. Perry, L. A. Sun. *Phase I Testbed Description: Requirements and Selection Guidelines* [7]
PS07	2007	ACM	Phil Greenwood, Alessandro Garcia, Awais Rashid, Eduardo Figueiredo, Claudio Sant'Anna, Nelio Cacho, Americo Sampaio, Sergio Soares, Paulo Borba, Marcos Dosea, Ricardo Ramos, Uira Kulesza, Thiago Bartolomei, Monica Pinto, Lidia Fuentes, Nadia Gamez, Ana Moreira, Joao Araujo, Thais Batista, Ana Medeiros, Francisco Dantas, Lyrene Fernandes, Jan WLoka, Christina Chavez, Robert France, Isabel Brito. *On the Contributions of and End-to-End AOSD Testbed* [33]
PS08	1998	IEEE	Allen H. Dutoit, Bernd Bruegge. *Communication Metrics for Software Development* [34]
PS09	1996	IEEE	Alan W. Brown, Kurt C. Wallnau. *A Framework for Evaluating Software Technology* [1]
PS10	2004	IEEE	Barry Boehm, Jesal Bhuta, David Garlan, Eric Gradman, LiGuo Huang, Alexander Lam, Ray Medachy, Nenad Medvidovic, Kenneth Meyer, Steven Meyers, Gustavo Perez, Kirk Reinholtz, Roshanak Roshandel, Nicolas Rouquette. *Using Empirical Testbeds to Accelerate Technology Maturity and Transition: The SCRover Experience* [35]

ID	Year	Source	Reference
PS11	1995	Citeseer	Richard Creps, Paul Kogut, Frank Svoboda. *A Testbed for Analyzing Architecture Description Languages* [36]
PS12	2002	Citeseer	Marcus Ciolkowski, Forrest Shull, Stefan Biffl. *A Family of Experiments to Investigate the Influence of Context on the Effect of Inspection Techniques* [2]
PS13	2003	Springer	Toni Sandelin, Matias Vierimaa. *Empirical Studies in ESERNET* [3]

APPENDIX B

EXCLUDED PRIMARY STUDIES

The list of excluded papers. In paper 93, *ESEJ* stands form Empirical Software Engineering Journal.

ID	Year	Source	Reference	Criteria
1	2007	IEEE	F. Parandoosh. *Evaluating Agent-Oriented Software Engineering Methodologies* In 2nd International Workshop on Soft. Computing Applications, 2007. SOFA 2007.	Irrelevant
2	1995	IEEE	M. Shaw, R. DeLine, D.V. Klein, T.L. Ross, D.M. Young, G. Zelesnik. *Abstractions for software architecture and tools to support them* In IEEE Transactions on Software Engineering	Irrelevant
3	2008	IEEE	T. Goldschmidt, R. Reussner, J. Winzen. *A case study evaluation of maintainability and performance of persistency techniques* In Proceedings of the International Conference on Software Engineering	Irrelevant
4	2007	IEEE	Phil Greenwood, Alessandro Garcia, Awais Rashid, Eduardo Figueiredo, Claudio Sant'Anna, Nelio Cacho, Americo Sampaio, Sergio Soares, Paulo Borba, Marcos Dosea, Ricardo Ramos, Uira Kulesza, Thiago Bartolomei, Monica Pinto, Lidia Fuentes, Nadia Gamez, Ana Moreira, Joao Araujo, Thais Batista, Ana Medeiros, Francisco Dantas, Lyrene Fernandes, Jan WLoka, Christina Chavez, Robert France, Isabel Brito. *On the Contributions of and End-to-End AOSD Testbed* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by ACM)
5	1997	IEEE	Rodion M. Podorozhny, Leon J. Osterweil. *Criticality of modeling formalisms in software design method comparison* In Proceedings of the International Conference on Software Engineering	Irrelevant
6	1994	IEEE	Xiping Song, Leon J. Osterweil. *Experience with an approach to comparing software design methodologies* In IEEE Transactions on Software Engineering	Irrelevant
7	1994	IEEE	P. Fowler, L. Levine. *From theory to practice: technology transition at the SEI* In Proceedings of the Hawaii International Conference on System Sciences	Irrelevant

ID	Year	Source	Reference	Criteria
8	2005	IEEE	A. H. Elamy, B. H. Far. *Utilizing incomplete block designs in evaluating agent-oriented software engineering methodologies* In Canadian Conference on Electrical and Computer Engineering	Irrelevant
9	2006	IEEE	A. H. Eden, T. Mens. *Measuring software flexibility* In IEE Proceedings: Software	Irrelevant
10	1999	IEEE	Victor R. Basili, Forrest Shull, Filippo Lanubile. *Building Knowledge through Families of Experiments* In IEEE Transactions on Software Engineering	Irrelevant
11	2007	IEEE	Miguel Goulão, O Brito E Abreu. *Modeling the Experimental Software Engineering Process* In 6th International Conference on the Quality of Information and Communications Technology, 2007. QUATIC 2007.	Irrelevant
12	2008	IEEE	M.G. Mendonca, J. C. Maldonado, M. C. F. de Oliveira, J. Carver, C. P. F. Fabbri, F. Shull, G. H. Travassos, E. N. Hohn, V. R. Basili. *A Framework for Software Engineering Experimental Replications* In 13th IEEE International Conference on Engineering of Complex Computer Systems, 2008. ICECCS 2008.	Irrelevant
13	2009	IEEE	N. H. Awang, W.M.N.W. Kadir, S. Shahibuddin. *Comparative Evaluation of the State-of-the Art on Approaches to Software Adaptation* In 4th International Conference onSoftware Engineering Advances, 2009. ICSEA '09.	Irrelevant
14	2010	IEEE	N. Pettersson, W. Lowe, J. Nivre. *Evaluation of Accuracy in Design Pattern Occurrence Detection* In IEEE Transactions on Software Engineering	Irrelevant
15	2000	IEEE	N. Medvidovic, R. N. Taylor. *A classification and comparison framework for software architecture description languages* In IEEE Transactions on Software Engineering	Irrelevant
16	1998	IEEE	J. Miller, M. Wood, M. Roper. *Further Experiences with Scenarios and Checklists* In Empirical Software Engineering	Irrelevant
17	1989	IEEE	J. Bayer, N. Melone. *A critique of diffusion theory as a managerial framework for understanding adoption of software engineering innovations* In The Journal of Systems and Software	Irrelevant
18	2005	ACM	Luciano Rodrigues Guimarães, Plínio Roberto Souza Vilela. *Comparing software development models using CDM* In SIGITE '05: Proceedings of the 6th conference on Information technology education	Irrelevant
19	2004	ACM	Barry Boehm, Jesal Bhuta, David Garlan, Eric Gradman, LiGuo Huang, Alexander Lam, Ray Medachy, Nenad Medvidovic, Kenneth Meyer, Steven Meyers, Gustavo Perez, Kirk Reinholtz, Roshanak Roshandel, Nicolas Rouquette. *Using Empirical Testbeds to Accelerate Technology Maturity and Transition: The SCRover Experience* In International Symposium on Empirical Software Engineering, 2004. ISESE '04. Proceedings.	Duplicate (originally published by IEEE)

ID	Year	Source	Reference	Criteria
20	2008	ACM	T. Goldschmidt, R. Reussner, J. Winzen. *A case study evaluation of maintainability and performance of persistency techniques* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by IEEE)
21	1997	ACM	Bernd Bruegge, Allen H. Dutoit. *Communication metrics for software development* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by IEEE)
22	1997	ACM	Rodion M. Podorozhny, Leon J. Osterweil. *Criticality of modeling formalisms in software design method comparison* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by IEEE)
23	2001	ACM	Serge Demeyer. *Towards a software evolution benchmark* In Proceedings of International Workshop on Principles of Software Evolution (IWPSE2001)	Irrelevant
24	2006	ACM	B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, Z. He, Z. Liming.*Evaluating guidelines for empirical software engineering studies* In Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering	Irrelevant
25	2006	ACM	S. Vegas, N. Juristo, A. Moreno, M. Solari, P. Letelier. *Analysis of the influence of communication between researchers on experiment replication* In Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering	Irrelevant
26	2004	ACM	F. Shull, M. G. Mendonca, V. Basili, J. Carver, J. C. Maldonado, S. Fabbri, G. H. Travassos, M. C. Ferreira. *Knowledge-Sharing Issues in Experimental Software Engineering* In Empirical Software Engineering	Irrelevant
27	2010	ACM	N. Pettersson, W. Lowe, J. Nivre. *Evaluation of Accuracy in Design Pattern Occurrence Detection* In IEEE Transactions on Software Engineering	Irrelevant
28	2000	ACM	N. Medvidovic, R. N. Taylor. *A classification and comparison framework for software architecture description languages* In IEEE Transactions on Software Engineering	Irrelevant
29	1998	ACM	J. Miller, M. Wood, M. Roper. *Further Experiences with Scenarios and Checklists* In Empirical Software Engineering	Irrelevant
30	1995	Citeseer	M. Shaw, R. DeLine, D.V. Klein, T.L. Ross, D.M. Young, G. Zelesnik. *Abstractions for software architecture and tools to support them* In IEEE Transactions on Software Engineering	Duplicate (originally published by IEEE)
31	1997	Citeseer	Rodion M. Podorozhny, Leon J. Osterweil. *Criticality of modeling formalisms in software design method comparison* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by IEEE)
32	2001	Citeseer	Serge Demeyer. *Towards a software evolution benchmark* In Proceedings of International Workshop on Principles of Software Evolution (IWPSE2001)	Irrelevant
33	1999	Citeseer	Victor R. Basili, Forrest Shull, Filippo Lanubile. *Building Knowledge through Families of Experiments* In IEEE Transactions on Software Engineering	Irrelevant

ID	Year	Source	Reference	Criteria
34	1996	Citeseer	Andrew Brooks, John Daly, James Miller, Marc Roper, Murray Wood. *Replication of Experimental Results in Software Engineering* In IEEE Transactions on Software Engineering	Irrelevant
35	1998	Citeseer	Victor Basili, Forrest Shull, Filippo Lanubile. *Using Experiments to Build a Body of Knowledge* In Perspectives of System Informatics	Irrelevant
36	1994	Citeseer	A. Brooks, J. Daly, J. Miller, M. Roper, M. Wood. *Replication's Role in Experimental Computer Science* By Department of Computer Science, University of Strathclyde, Glasgow, U.K.	Irrelevant
37	1999	Citeseer	Victor Basili, Forrest Shull, Filippo Lanubile. *Building Knowledge through Families of Software Studies: An Experience Report* By Computer Science Department of the University of Maryland	Irrelevant
38	1996	Citeseer	V. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, M. Zelkowitz. *Packaging Researcher Experience to Assist Replication of Experiments* In Proceeding of the ISERN Meeting	Irrelevant
39	2007	Citeseer	Miguel Goulão, O Brito E Abreu. *Modeling the Experimental Software Engineering Process* In 6th International Conference on the Quality of Information and Communications Technology, 2007. QUATIC 2007.	Irrelevant
40	2008	Citeseer	L. Hochstein, T. Nakamura, F. Shull, N. Zazworka, V. R. Basili, M. V. Zelkowitz. *Chapter 5 An Environment for Conducting Families of Software Engineering Experiments* In Advances in Computers	Irrelevant
41	2008	Citeseer	F. J. Shull, J. C. Carver, S. Vegas, N. Juristo. *The role of replications in Empirical Software Engineering* In Empirical Software Engineering	Irrelevant
42	2000	Citeseer	N. Medvidovic, R. N. Taylor. *A classification and comparison framework for software architecture description languages* In IEEE Transactions on Software Engineering	Irrelevant
43	1998	Citeseer	J. Miller, M. Wood, M. Roper. *Further Experiences with Scenarios and Checklists* In Empirical Software Engineering	Irrelevant
44	2004	Scopus	Barry Boehm, Jesal Bhuta, David Garlan, Eric Gradman, LiGuo Huang, Alexander Lam, Ray Medachy, Nenad Medvidovic, Kenneth Meyer, Steven Meyers, Gustavo Perez, Kirk Reinholtz, Roshanak Roshandel, Nicolas Rouquette. *Using Empirical Testbeds to Accelerate Technology Maturity and Transition: The SCRover Experience* In International Symposium on Empirical Software Engineering, 2004. ISESE '04. Proceedings.	Duplicate (originally published by IEEE)
45	1989	Scopus	William E. Rzepka. *A requirements engineering testbed: concept, status and first results* In Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences	Duplicate (originally published by IEEE)
46	2007	Scopus	F. Parandoosh. *Evaluating Agent-Oriented Software Engineering Methodologies* In 2nd International Workshop on Soft. Computing Applications, 2007. SOFA 2007.	Irrelevant

ID	Year	Source	Reference	Criteria
47	1995	Scopus	M. Shaw, R. DeLine, D.V. Klein, T.L. Ross, D.M. Young, G. Zelesnik. *Abstractions for software architecture and tools to support them* In IEEE Transactions on Software Engineering	Irrelevant
48	2009	Scopus	A. Lam, B. Boehm. *Experiences in Developing and Applying a Software Engineering Technology Testbed* In Empirical Software Engineering	Duplicate (originally published by Springer)
49	2008	Scopus	T. Goldschmidt, R. Reussner, J. Winzen. *A case study evaluation of maintainability and performance of persistency techniques* In Proceedings of the International Conference on Software Engineering	Irrelevant
50	2007	Scopus	Phil Greenwood, Alessandro Garcia, Awais Rashid, Eduardo Figueiredo, Claudio Sant'Anna, Nelio Cacho, Americo Sampaio, Sergio Soares, Paulo Borba, Marcos Dosea, Ricardo Ramos, Uira Kulesza, Thiago Bartolomei, Monica Pinto, Lidia Fuentes, Nadia Gamez, Ana Moreira, Joao Araujo, Thais Batista, Ana Medeiros, Francisco Dantas, Lyrene Fernandes, Jan WLoka, Christina Chavez, Robert France, Isabel Brito. *On the Contributions of and End-to-End AOSD Testbed* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by ACM)
51	2007	Scopus	Mikael Lindvall, Ioana Rus, Paolo Donzelli, Atif Memon, Marvin Zelkowitz, Aysu Betin-Can, Tevfik Bultan, Chris Ackermann, Bettina Anders, Sima Asgari, Victor Basili, Lorin Hochstein, Jörg Fellmann, Forrest Shull, Roseanne Tvedt, Daniel Pech, Daniel Hirschbach. *Experimenting with Software Testbeds for Evaluating new Technologies* In Empirical Software Engineering	Duplicate (originally published by Springer)
52	2005	Scopus	Mikael Lindvall, Ioana Rus, Forrest Shull, Marvin Zelkowitz, Paolo Donzelli, Atif Memon, Victor Basili, Patricia Costa, Roseanne Tvedt, Lorin Hochstein, Sima Asgari, Chris Ackermann, Dan Pech. *An evolutionary testbed for software technology evaluation* In Innovations in Systems and Software Engineering	Duplicate (originally published by Springer)
53	2005	Scopus	G. Aditya Kiran, S. Haripriya, Pankaj Jalote. *Effect of Object Orientation on Maintainability of Software* In Conference on Software Maintenance	Duplicate (originally published by IEEE)
54	1998	Scopus	Allen H. Dutoit, Bernd Bruegge. *Communication Metrics for Software Development* In Proceedings of the International Conference on Software Engineering	Duplicate (originally published by IEEE)
55	1996	Scopus	Alan W. Brown, Kurt C. Wallnau. *A Framework for Evaluating Software Technology* In IEEE Software	Duplicate (originally published by IEEE)
56	2007	Scopus	V. R. Basili, M. V. Zelkowitz, D. I. K. Sjøberg, P. Johnson, A. J. Cowling. *Protocols in the use of empirical software engineering artifacts* In Empirical Software Engineering	Irrelevant
57	1997	Scopus	Rodion M. Podorozhny, Leon J. Osterweil. *Criticality of modeling formalisms in software design method comparison* In Proceedings of the International Conference on Software Engineering	Irrelevant

ID	Year	Source	Reference	Criteria
58	1994	Scopus	Xiping Song, Leon J. Osterweil. *Experience with an approach to comparing software design methodologies* In IEEE Transactions on Software Engineering	Irrelevant
59	2001	Scopus	Serge Demeyer. *Towards a software evolution benchmark* In Proceedings of International Workshop on Principles of Software Evolution (IWPSE2001)	Irrelevant
60	1994	Scopus	P. Fowler, L. Levine. *From theory to practice: technology transition at the SEI* In Proceedings of the Hawaii International Conference on System Sciences	Irrelevant
61	2005	Scopus	A. H. Elamy, B. H. Far. *Utilizing incomplete block designs in evaluating agent-oriented software engineering methodologies* In Canadian Conference on Electrical and Computer Engineering	Irrelevant
62	2003	Scopus	Toni Sandelin, Matias Vierimaa. *Empirical Studies in ESERNET* In Lecture Notes in Computer Science	Duplicate (originally published by Springer)
63	2009	Scopus	M. Thongmak, P. Muenchaisri. *Maintainability metrics for aspect-oriented software* In International Journal of Software Engineering and Knowledge Engineering	Irrelevant
64	2006	Scopus	B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, Z. He, Z. Liming.*Evaluating guidelines for empirical software engineering studies* In Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering	Irrelevant
65	2006	Scopus	A. H. Eden, T. Mens. *Measuring software flexibility* In IEE Proceedings: Software	Irrelevant
66	1999	Scopus	Victor R. Basili, Forrest Shull, Filippo Lanubile. *Building Knowledge through Families of Experiments* In IEEE Transactions on Software Engineering	Irrelevant
67	2007	Scopus	Miguel Goulão, O Brito E Abreu. *Modeling the Experimental Software Engineering Process* In 6th International Conference on the Quality of Information and Communications Technology, 2007. QUATIC 2007.	Irrelevant
68	2008	Scopus	M.G. Mendonca, J. C. Maldonado, M. C. F. de Oliveira, J. Carver, C. P. F. Fabbri, F. Shull, G. H. Travassos, E. N. Hohn, V. R. Basili. *A Framework for Software Engineering Experimental Replications* In 13th IEEE International Conference on Engineering of Complex Computer Systems, 2008. ICECCS 2008.	Irrelevant
69	2009	Scopus	N. H. Awang, W.M.N.W. Kadir, S. Shahibuddin. *Comparative Evaluation of the State-of-the Art on Approaches to Software Adaptation* In 4th International Conference on Software Engineering Advances, 2009. ICSEA '09.	Irrelevant
70	2009	Scopus	D. Falessi, M. A. Babar, G. Cantone, P. Kruchten. *Applying empirical software engineering to software architecture: challenges and lessons learned* In Empirical Software Engineering	Irrelevant
71	2008	Scopus	L. Hochstein, T. Nakamura, F. Shull, N. Zazworka, V. R. Basili, M. V. Zelkowitz. *Chapter 5 An Environment for Conducting Families of Software Engineering Experiments* In Advances in Computers	Irrelevant

ID	Year	Source	Reference	Criteria
72	2008	Scopus	F. J. Shull, J. C. Carver, S. Vegas, N. Juristo. *The role of replications in Empirical Software Engineering* In Empirical Software Engineering	Irrelevant
73	2008	Scopus	B. Kitchenham. *The role of replications in empirical software engineering-a word of warning* In Empirical Software Engineering	Irrelevant
74	2007	Scopus	D. I. K. Sjøberg. *Knowledge acquisition in software engineering requires sharing of data and artifacts* In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	Irrelevant
75	2006	Scopus	S. Vegas, N. Juristo, A. Moreno, M. Solari, P. Letelier. *Analysis of the influence of communication between researchers on experiment replication* In Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering	Irrelevant
76	2004	Scopus	Luqi, L. Zhang, M. Saboe. *The dynamical models for software technology transition* In International Journal of Software Engineering and Knowledge Engineering	Irrelevant
77	2004	Scopus	F. Shull, M. G. Mendonca, V. Basili, J. Carver, J. C. Maldonado, S. Fabbri, G. H. Travassos, M. C. Ferreira. *Knowledge-Sharing Issues in Experimental Software Engineering* In Empirical Software Engineering	Irrelevant
78	2010	Scopus	N. Pettersson, W. Lowe, J. Nivre. *Evaluation of Accuracy in Design Pattern Occurrence Detection* In IEEE Transactions on Software Engineering	Irrelevant
79	2000	Scopus	N. Medvidovic, R. N. Taylor. *A classification and comparison framework for software architecture description languages* In IEEE Transactions on Software Engineering	Irrelevant
80	1998	Scopus	J. Miller, M. Wood, M. Roper. *Further Experiences with Scenarios and Checklists* In Empirical Software Engineering	Irrelevant
81	1989	Scopus	J. Bayer, N. Melone. *A critique of diffusion theory as a managerial framework for understanding adoption of software engineering innovations* In The Journal of Systems and Software	Irrelevant
82	2007	Springer	V. R. Basili, M. V. Zelkowitz, D. I. K. Sjøberg, P. Johnson, A. J. Cowling. *Protocols in the use of empirical software engineering artifacts* In Empirical Software Engineering	Irrelevant
83	2006	Springer	B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, Z. He, Z. Liming.*Evaluating guidelines for empirical software engineering studies* In Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering	Irrelevant
84	1998	Springer	Victor Basili, Forrest Shull, Filippo Lanubile. *Using Experiments to Build a Body of Knowledge* In Perspectives of System Informatics	Irrelevant
85	2009	Springer	D. Falessi, M. A. Babar, G. Cantone, P. Kruchten. *Applying empirical software engineering to software architecture: challenges and lessons learned* In Empirical Software Engineering	Irrelevant

ID	Year	Source	Reference	Criteria
86	2008	Springer	F. J. Shull, J. C. Carver, S. Vegas, N. Juristo. *The role of replications in Empirical Software Engineering* In Empirical Software Engineering	Irrelevant
87	2008	Springer	B. Kitchenham. *The role of replications in empirical software engineering-a word of warning* In Empirical Software Engineering	Irrelevant
88	2007	Springer	D. I. K. Sjøberg. *Knowledge acquisition in software engineering requires sharing of data and artifacts* In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	Irrelevant
89	2004	Springer	F. Shull, M. G. Mendonca, V. Basili, J. Carver, J. C. Maldonado, S. Fabbri, G. H. Travassos, M. C. Ferreira. *Knowledge-Sharing Issues in Experimental Software Engineering* In Empirical Software Engineering	Irrelevant
90	1998	Springer	J. Miller, M. Wood, M. Roper. *Further Experiences with Scenarios and Checklists* In Empirical Software Engineering	Irrelevant
91	2008	Springer	A. Brooks, M. Roper, M. Wood, J. Daly, J. Miller. *Replication's Role in Software Engineering* In Guide to Advanced Empirical Software Engineering (Book)	Irrelevant
92	1995	Springer	Shari Pfleeger. *Experimental design and analysis in software engineering* In Annals of Software Engineering	Irrelevant
93	2007	ESEJ	Mikael Lindvall, Ioana Rus, Paolo Donzelli, Atif Memon, Marvin Zelkowitz, Aysu Betin-Can, Tevfik Bultan, Chris Ackermann, Bettina Anders, Sima Asgari, Victor Basili, Lorin Hochstein, Jörg Fellmann, Forrest Shull, Roseanne Tvedt, Daniel Pech, Daniel Hirschbach. *Experimenting with Software Testbeds for Evaluating new Technologies* In Empirical Software Engineering	Duplicate (originally published by Springer)

APPENDIX C

THE SYSTEMATIC MAPPING STUDY PROTOCOL

C.1 TEAM

- Emanoel Francisco Spósito Barreiros (Author)
  - Contact: efsb@cin.ufpe.br
  - Affilitation: Informatics Center - Federal University of Pernambuco
- Sérgio Castelo Branco Soares (Revisor)
  - Contact: scbs@cin.ufpe.br
  - Affilitation: Informatics Center - Federal University of Pernambuco
- Adauto Trigueiro de Almeida Filho (Revisor)
  - Contact: ataf@cin.ufpe.br
  - Affilitation: Informatics Center - Federal University of Pernambuco

C.2 INTRODUCTION

Systematic Reviews (SR) are effective tools to gather knowledge that is commonly spread elsewhere. By following the steps defined in a previously defined protocol, researchers are able to aggregate relevant studies in a replicable and verifiable way. More specifically, Systematic Mapping Study (SMS) is a method that provides an overview of a research field, making possible to identify, quantify and analyze types of research and available results [22]. According to Kitchenham [23] an SMS is a broad review of primary studies of a specific area which aims to identify the available evidences of the investigated topic. As so, an SMS is classified as a secondary study, since it needs evidences described by other studies to build knowledge.

Kitchenham and Charters [23] enlist several reasons to perform systematic reviews:

- " To summarise the existing evidence concerning a treatment or technology e.g. to summarise the empirical evidence of the benefits and limitations of a specific agile method.";

- "To identify any gaps in current research in order to suggest areas for further investigation.";
- "To provide a framework/background in order to appropriately position new research activities."

Systematic Literature Reviews (SLR) are very similar to mapping studies and is also a very important method associated to Evidence-Based Software Engineering (EBSE) [21]. Different from ordinary literature reviews commonly found in most studies, both methods demand a protocol to guide the research, defining search strategies, inclusion and exclusion criteria and so forth. SLRs, however, are different from SMSs because they aim to answer very narrow research questions, while SMSs are concerned with mapping the knowledge in a given area, identifying deserts of knowledge and directing further research. In many cases, SMSs proceed SLRs, providing valuable information and optimizing the whole process of research.

Both SLRs and SMSs need protocols to be executed. These protocols must be developed beforehand and provided to the community after the study is concluded. This allows the community to assess the correctness of the process employed and evaluate its results. This is an important advantage over ordinary literature reviews, since the knowledge is auditable. In this sense, the protocol used in the current mapping study is presented next.

C.3 SCOPE OF THE STUDY

Since the current study involves a systematic search on the software engineering literature, the population of the study is composed of all papers that have reported some sort of empirical studies in the evaluation of tehcnologies in software engineering. The universe of studies shall be obtained by executing automated and manual searches, as detailed in Section C.5.

The mapping study is concerned only with studies that have defined testbeds to evaluate of technologies in SE. However, the search has to be broad enough to reduce the risk of not selecting relevant papers that employ similar empirical environments but name it differently. Hence, other kinds of empirical studies can be included if they define a similar structure to testbeds, but use different terms to define it.

It is expected that the results of the mapping study can enhance the knowledge in the area by providing aggregated information regarding the use of testbeds in software engineering. The evidences shown in this work are important to fulfill this task. Empirical

studies on SE are run mainly in academic environments, which characterizes the current mapping study as being of academic context.

The selected studies must be limited to the empirical software engineering area. Since many studies have reported testbeds in other areas, a great amount of irrelevant studies would generate a high overhead.

C.4 RESEARCH QUESTIONS

The whole idea behind the current mapping study came up when wondering: *How can we improve empirical studies in software engineering? How can researchers effectively and systematically evaluate technologies in software engineering?* This pushed the research to search for tools that could be used to effectively and efficiently evaluate technologies in software engineering. Testbeds have emerged as a concept that has been widely used in other areas, but poorly explored in software engineering.

The current secondary study aims to query the literature and map primary studies that defined testbeds for software engineering. The following research questions were identified as relevant to the purpose of the study:

1. **RQ1:** Which specific software engineering fields are investigated using testbeds?
2. **RQ2:** Which elements the studies define for a testbed in software engineering?
3. **RQ3:** Which are the benefits of using a software engineering testbed?
4. **RQ4:** Which methods are used to evaluate the proposed testbed?

C.5 SEARCH PROCESS

To try to assess the largest amount of studies possible in relatively small amount of time, several automated search engines shall be used:

- IEEE Computer Society Digital Library [1];
- ACM Digital Library [2];
- Scopus [3];

[1] http://ieeexplore.ieee.org
[2] http://portal.acm.org
[3] http://www.scopus.com

- Citeseer [4];
- Springer Link [5].

Also, manual searches must be performed on the proceedings of the last editions of a few leading conferences in the empirical SE field and one journal:

- International Conference on Software Engineering (ICSE '10);
- Empirical Software Engineering and Measurement (ESEM '09) [6];
- Evaluation and Assessment in Software Engineering (EASE '10);
- Past twelve months of the Empirical Software Engineering Journal (from June 2009 to June 2010)

Even though there was a time constraint on the manual searches, the automated ones were performed without this limitation. Researchers have reported that automated searches are flawed in several points [30, 23], hence it is important to not only rely on such mechanisms in order to try to reduce the risks of not selecting relevant studies.

The relevant terms identified were the following:

- **Software Engineering**: software engineering;
- **Testbed**: testbed, family of experiments, technology evaluation, technology comparison, technology transition, software maturity, software adoption, replication, replicability, replicate.

Some keywords my seem not to be related to testbeds. These keywords were chosen to attempt to find more relevant studies and include those that use concepts related to "testbeds" but do not name them accordingly. Since the concept is not widely disseminated, this situation emerged as probable. A general query to be used on the selected search engines was built:

[4] http://citeseer.ist.psu.edu
[5] http://www.springerlink.com
[6] The search was performed prior to the 2010 edition.

> "software engineering" AND (testbed OR "family of experiments" OR "technology evaluation" OR "technology comparison" OR "technology transition" OR "software adoption" OR "software maturity" OR "replicability" OR "replicate" OR "replication" OR "experiment protocol" OR "study protocol" OR "experiment process" OR "experiment guidelines" OR "study process" OR comparability OR comparable OR compare)

C.5.1 Inclusion and Exclusion Criteria

One important detail to consider is the inclusion and exclusion of papers. The researcher must be very careful when analyzing which papers will and will not be included because some interesting papers can be erroneously excluded as a result of a misconducted analysis.

In the current mapping study, all papers that are clearly out of scope shall be removed early in the process based on the analysis of the keywords, title and abstract. After the initial selection, the full versions of each paper must be obtained so that a more detailed analysis can be performed. The following criteria must be applied in order to include or exclude the paper from the mapping:

- It is a real paper, not a Power Point© presentation or extended abstract (tech reports, PhD Thesis and MSc Dissertations are considered as well);
- The paper is not a duplicate;
- The paper defines a testbed (or at least a process, framework, etc., that is concerned with evaluation of technologies);
- The paper is related to software engineering.

C.6 SELECTION PROCESS

The selection process is described next:

1. The researcher utilizes the automated search engines and applies the search string as a filter. The manual searches are also performed. In this case, only potentially relevant papers are added to the list of papers, to avoid unecessary work. As a result, a list of potential primary studies is generated. In this step, many papers may be returned. Worse, many papers that are not related to research topic may

be present. In this case, all papers that are clearly out of scope are removed from the mapping. This is accomplished by analyzing the title, keywords, abstract and relevance to the topic and research questions. No record is kept in this phase. The papers must be reviewed by at least 2 researchers to avoid bias;

2. The list of papers generated by the researchers is compared and a final list is created. If there is any disagreement regarding the permanence of the paper in the mapping, it is kept to avoid the removal of potentially relevant papers;

3. Each paper in the list is obtained and screened by the researchers. In this phase, the whole paper is evaluated. The inclusion and exclusion criteria are applied and the relevance to the topic and research questions is evaluated again. Every paper must be evaluated by at least two researchers;

4. The final set os studies is documented. The included studies are documented using Form A. Excluded studies are documented using Form B. The data extraction is performed using Form C only in the set of included studies. The referred forms are detailed in Section C.7.

Whenever studies have to be evaluated, disagreements regarding their permanence in the mapping may arise since more than one researcher evaluates each paper. The evaluation process can follow a simple list of steps, which is depicted in Figure C.1.

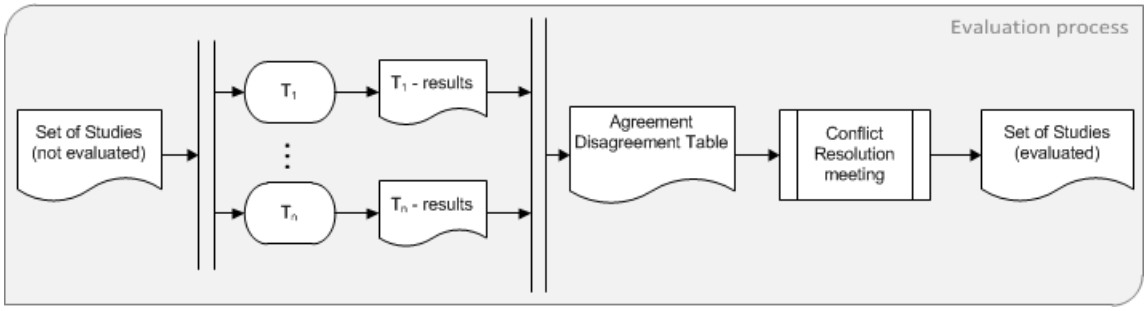

Figure C.1 The process of evaluation considering the conflict resolution meeting.

In this general process, the researchers can be divided into teams. Each team is given a set of studies to be evaluated. After the evaluation, each team should create a report describing the studies that were included and the ones that were removed. Eventual disagreements shall be resolved in a meeting with the presence of the teams. The teams should present their opinions on the inclusion or exclusion of the studies. The result of

this meeting is a report that describes the papers that have been included and excluded from the mapping. Every excluded paper must be justified. It is a good practice to have a senior researcher supervising the meetings to aid in its conduction and conflicts resolution.

Small variations are needed depending on when the process is used during the whole mapping process. In the initial round of evaluation, when all studies returned by search engines are evaluated, reasons for the exclusion of papers (and the paper themselves) do not have to be documented. On the other hand, in the second round of evaluation, every paper (included or excluded) must be documented.

C.6.1 Relevance Evaluation

The process of evaluating the relevance of the primary studies is important, for example, for providing more inclusion/exclusion criteria, for providing a means of weighing the papers when results are being synthesized, and to guide recommendations for further research [23]. The relevance evaluation was performed through questions that should be answered with three possible values:

- Yes (Y): If the question is fully answered; this answer counts the value 1.0;
- Partly (P): If the question is partly answered; this answer counts the value 0.5;
- No (N): If the question is not answered or not satisfactorily answered; this answer counts the value 0.0.

There were five questions to be answered, which makes possible for each paper to score from 0.0 to 5.0. The questions try to measure, to some extent, both the methodological quality and relevance to the mapping study. The defined questions are:

1. Does the paper define the concept of testbed?
2. Has the paper evaluated the testbed it proposes?
3. Does the testbed evaluation allows replication?
4. Does the study define a process for using the testbed?
5. Does the study point the benefits of having and/or using a testbed?

After this grading, the primary studies were classified in four different groups: *Very Good* (between 4.5 and 5.0), *Good* (between 3.0 and 4.0), *Regular* (between 1.5 and 2.5)

and *Poor* (between 0.0 and 1.0). The intention was to create four evenly distributed classes, since it was not possible, it was preferred to keep the highest class with less possible values to try to reduce bias.

C.7 DATA EXTRACTION

To help the extraction of information from the selected and excluded studies, three forms were used. They are shown in the following sections.

C.7.1 Form A

This form is used to collect general information from the selected primary studies. The following information is collected from each of the papers:

- **ID**: An identifier for the paper. It is useful to reference the paper in several parts of the mapping;
- **Source**: The publisher of the paper;
- **Year**: The year of publication;
- **Title**: The title of the paper;
- **Author**: The list of authors of the paper;
- **Institution**: The list of institutions that took part in the work;
- **Country**: The list of countries where the institutions are based.

C.7.2 Form B

The excluded papers must also be documented. One important information to stress is the reason why the paper was excluded. The information extracted from each excluded paper is:

- **ID**: An identifier for the paper. It is useful to reference the paper in several parts of the mapping;
- **Source**: The publisher of the paper;
- **Year**: The year of publication;

- **Title**: The title of the paper;
- **Author**: The list of authors of the paper;
- **Criteria used for exclusion**: The criteria used by the researcher to exclude the paper from the mapping.

C.7.3 Form C

Form C is used to extract the relevant information from the primary studies. In addition to the information collected in Form A, the following information is collected:

- **Evaluation Date**: When the paper was evaluated;
- **Researcher**: Researcher that extracted the information and filled the form;
- **Research Question 1**: Which specific software engineering fields are investigated using testbeds?
- **Research Question 2**: Which elements the studies define for a testbed in software engineering?
- **Research Question 3**: Which are the benefits of using a software engineering testbed?
- **Research Question 4**: Which methods are used to evaluate the proposed testbed?
- **Notes**: Eventually some insight from the researcher can be documented in this field.

C.7.4 Data Synthesis

With the collected data, a more detailed analysis shall be performed in order to build maps that represent the knowledge generated by the research. This synthesis relate the topics investigated with the use of testbeds, the elements the studies define as being the building blocks of testbeds, the benefits researchers take advantage when using testbeds and the methods they used to evaluate the proposed testbeds.

The synthesis focus on presenting the frequencies of the publications in each category to allow the visualization of which category has been emphasized in previous research, identifying deserts of knowledge and helping researchers to drive new research in poorly

investigated fields. This mapping study presents maps in the form of frequeny tables, relating each category with the reference of the papers belonging the category and the overall frequency, and by building bars, columns and pie charts.

BIBLIOGRAPHY

[1] Alan W. Brown and Kurt C. Wallnau. A framework for evaluating software technology. *IEEE Softw.*, 13:39–49, September 1996. (document), 4.3, 4.2.1, 4.2.2, 4.6, A

[2] Marcus Ciolkowski, Forrest Shull, and Stefan Biffl. A family of experiments to investigate the influence of context on the effect of inspection techniques. 2002. (document), 4.3, 4.7, 4.8, A

[3] Toni Sandelin and Matias Vierimaa. Empirical studies in esernet. 2765:39–54, 2003. (document), 4.3, 4.2.1, 4.9, 4.2.4, A

[4] V R Basili, R W Selby, and D H Hutchens. Experimentation in software engineering. *IEEE Trans. Softw. Eng.*, 12(7):733–743, 1986. (document), 1, 2.1, 2.1.1, 2.1

[5] Victor R. Basili. The experimental paradigm in software engineering. In *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions*, pages 3–12, London, UK, 1993. Springer-Verlag. 1, 2.2

[6] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000. 1, 2.1, 2.1.1

[7] Robert Holibaugh, J. M. Perry, and L. A. Sun. Phase i testbed description: Requirements and selection guidelines. Technical report, Software Engineering Institute, Carnegie-Mellon University, September 1988. 1, 4.3, A

[8] Colin Robson. *A Resource for Social Scientists and Practitioners-Researchers*. Wiley-Blackwell, 2 edition, 2002. 2.1

[9] Walter F. Tichy. Should computer scientists experiment more? - 16 excuses to avoid experimentation. *IEEE Computer*, 31:32–40, 1997. 2.1

[10] Victor R. Basili. The role of experimentation in software engineering: past, current, and future. In *ICSE '96: Proceedings of the 18th international conference on Software engineering*, pages 442–449, Washington, DC, USA, 1996. IEEE Computer Society. 2.1, 2.2

[11] Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz. Experimental evaluation in computer science: a quantitative study. *J. Syst. Softw.*, 28(1):9–18, 1995. 2.1

[12] Marvin V. Zelkowitz and Dolores R. Wallace. Experimental models for validating technology. *Computer*, 31(5):23–31, 1998. 2.1

[13] Mikael Lindvall, Ioana Rus, Forrest Shull, Marvin Zelkowitz, Paolo Donzelli, Atif Memon, Victor Basili, Patricia Costa, Roseanne Tvedt, Lorin Hochstein, Sima Asgari, Chris Ackermann, and Dan Pech. An evolutionary testbed for software technology evaluation. *Innovations in Systems and Software Engineering*, 1:3–11, April 2005. 2.1, 2.2, 4.3, A

[14] Mikael Lindvall, Ioana Rus, Paolo Donzelli, Atif Memon, Marvin Zelkowitz, Aysu Betin-Can, Tevfik Bultan, Chris Ackermann, Bettina Anders, Sima Asgari, Victor Basili, Lorin Hochstein, Jörg Fellmann, Forrest Shull, Roseanne Tvedt, Daniel Pech, and Daniel Hirschbach. Experimenting with software testbeds for evaluating new technologies. *Empirical Softw. Eng.*, 12(4):417–444, 2007. 1a, 1b, 1c, 2.2, 4.3, A

[15] Gregory Abowd, Robert Allen, and David Garlan. Using style to understand descriptions of software architecture. *SIGSOFT Softw. Eng. Notes*, 18:9–20, December 1993. 2.2

[16] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean marc Loingtier, and John Irwin. Aspect-oriented programming. In *ECOOP*. SpringerVerlag, 1997. 2.2, 4.2.1

[17] A. Brooks, M. Roper, M. Wood, J. Daly, and J. Miller. *Replication's Role in Software Engineering*, pages 365–379. Springer London, 2008. 2.2

[18] Lorin Hochstein, Taiga Nakamura, Forrest Shull, Nico Zazworka, Martin Voelp, Marvin V. Zelkowitz, and Victor R. Basili. An environment of conducting families of software engineering experiments. Technical report, University of Maryland, 2007. 2.2

[19] Alexander Lam and Barry Boehm. Experiences in developing and applying a software engineering technology testbed. *Empirical Software Engineering*, 14(5):579–601, 2009. 2.2, 4.3, A

[20] Samuel T. Redwine, Jr. and William E. Riddle. Software technology maturation. In *ICSE '85: Proceedings of the 8th international conference on Software engineering*, pages 189–200, Los Alamitos, CA, USA, 1985. IEEE Computer Society Press. 2.2, 5.2

[21] Barbara A. Kitchenham, Tore Dybå, and Magne Jorgensen. Evidence-based software engineering. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 273–281, Washington, DC, USA, 2004. IEEE Computer Society. 2.3, 3.2, C.2

[22] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. *12th International Conference on Evaluation and Assessment in Software Engineering*, pages 71–80, 2008. 2.3, 3.1, 3.1, C.2

[23] Software Engineering Group. Guidelines for performing systematic literature reviews in software engineering. Technical report, School of Computer Science and Mathematics, Keele University, July 2007. 2.3, 3.1, 3.1.2, 3.2, 3.2.4, 4.1, 5.1, 5.2, C.2, C.5, C.6.1

[24] Tore Dybå, Vigdis By Kampenes, and Dag I.K. Sjøberg. A systematic review of statistical power in software engineering experiments. *Information and Software Technology*, 48(8):745 – 755, 2006. 2.3

[25] Jo E. Hannay, Dag I.K. Sjøberg, and Tore Dybå. A systematic review of theory use in software engineering experiments. *IEEE Transactions on Software Engineering*, 33:87–107, 2007. 2.3

[26] Marina de Andrade Marconi and Eva Maria Lakatos. *Metodologia Científica*. Editora Atlas, São Paulo, 5 edition, 2010. 3.1, 3.1

[27] Harris Cooper. Organizing knowledge syntheses: A taxonomy of literature reviews. *Knowledge, Technology and Policy*, 1:104–126, 1988. 10.1007/BF03177550. 3.1.1

[28] Barbara A. Kitchenham, Emilia Mendes, and Guilherme H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, 33:316–329, 2007. 3.2

[29] Rialette Pretorius and David Budgen. A mapping study on empirical evidence related to the models and forms used in the uml. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '08, pages 342–344, New York, NY, USA, 2008. ACM. 3.2

[30] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80:571–583, April 2007. 3.2.2, C.5

[31] W.E. Rzepka. A requirements engineering testbed: concept, status and first results. In *System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, volume 2, pages 339–347 vol.2, 3-6 1989. 4.3, A

[32] G. Aditya Kiran, S. Haripriya, and Pankaj Jalote. Effect of object orientation on maintainability of software. pages 114–121. IEEE, Piscataway, NJ, United States, 1997. 4.3, A

[33] Phil Greenwood, Alessandro Garcia, Awais Rashid, Eduardo Figueiredo, Claudio Sant'Anna, Nelio Cacho, Americo Sampaio, Sergio Soares, Paulo Borba, Marcos Dosea, Ricardo Ramos, Uira Kulesza, Thiago Bartolomei, Monica Pinto, Lidia Fuentes, Nadia Gamez, Ana Moreira, Joao Araujo, Thais Batista, Ana Medeiros, Francisco Dantas, Lyrene Fernandes, Jan Wloka, Christina Chavez, Robert France, and Isabel Brito. On the contributions of an end-to-end aosd testbed. In *EARLYASPECTS '07: Proceedings of the Early Aspects at ICSE*, page 8, Washington, DC, USA, 2007. IEEE Computer Society. 4.3, A

[34] Allen H. Dutoit and Bernd Bruegge. Communication metrics for software development. *Proceedings - International Conference on Software Engineering*, pages 271–281, 1997. 4.3, A

[35] B. Boehm, J. Bhuta, D. Garlan, E. Gradman, LiGuo Huang, A. Lam, R. Madachy, N. Medvidovic, K. Meyer, S. Meyers, G. Perez, K. Reinholtz, R. Roshandel, and

N. Rouquette. Using empirical testbeds to accelerate technology maturity and transition: the scrover experience. pages 117–126, 19-20 2004. 4.3, A

[36] R. Creps, Presenter Richard Creps, and Day Tuesday. A testbed for analyzing architecture description languages. 1995. 4.3, A

[37] Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24:131–183, June 1992. 4.2.1

[38] Adauto Trigueiro de Almeida Filho. Um mapeamento sistemático de mecanismos para guiar estudos empíricos em engenharia de software. Master's thesis, Federal University of Pernambuco, March 2011. 4.4

[39] Marcelo Moura. Um benchmarking framework para avaliação da manutenibilidade de software orientado a aspectos. Master's thesis, University of Pernambuco, 2008. 4.4, 5.2

[40] Luqi, L. Zhang, and M. Saboe. The dynamical models for software technology transition. *International Journal of Software Engineering and Knowledge Engineering*, 14(2):179–205, 2004. 5.2

[41] D.I.K. Sjøberg. Knowledge acquisition in software engineering requires sharing of data and artifacts. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4336 LNCS:77–82, 2007. 5.2