Evaluation Studies of Software Testing Research in the Brazilian Symposium on Software Engineering

Otávio Augusto Lazzarini Lemos*, Fabiano Cutigi Ferrari†, Marcelo Medeiros Eler‡, José Carlos Maldonado‡, Paulo Cesar Masiero‡
*Science and Technology Department – Federal University of São Paulo at S. J. dos Campos – Brazil
Email: otavio.lemos@unifesp.br
†Computing Department – Federal University of São Carlos – Brazil
Email: fabiano@ufscar.br
‡Computer Systems Department – University of São Paulo at São Carlos – Brazil
Email: {mareler, jcmaldon, masiero}@icmc.usp.br

***Abstract*—Experimentation is the traditional way of identifying cause-effect relationships in scientific research. Lately, there has been an increasing understanding that experiments and other forms of evaluation should be more thoroughly disseminated among computer science and, in particular, Software Engineering (SE) researchers. Software testing (ST) is an important SE topic, where experiments are particularly valuable: since cost constraints and high effectiveness goals are common within this subfield, cost/benefit characteristics have to be adequately evaluated to deem a specific approach useful or not. This paper reports on a historical perspective of the evaluation studies present in ST research published in the Brazilian Symposium on Software Engineering (SBES). The survey characterizes the software testing-related papers published in the 24-year history of SBES, investigates the types of evaluation presented in these studies – when they were presented at all – and whether the number of evaluations has increased over the years. Additionally, the survey also brings a preliminary characterization of the Brazilian software testing community that adopts SBES as a vehicle to publish its research. Results show that the number of papers that present evaluation studies have significantly increased over the years. However, on the downside, amongst the papers that described some kind of evaluation, only 20% performed more rigorous evaluations (e.g. experiments or case studies in the industrial context), whereas 80% described exploratory, less rigorous case studies.**

I. Introduction

There is a common knowledge that science in general develops through theory and experimentation. Theory tries to define the nature and causes of a problem, while experimentation may confirm or refute such definitions. On the other hand, experimentation may also find new phenomena that can be explained through a theory [1]. Therefore, the application of experiments and other types of evaluation is considered essential to the development of any scientific field.

Several computer science researchers feel that our field has not yet insisted enough on experimentation (*e.g.* [2]). This is also true with respect to the software engineering (SE) community. Victor Basili is one of the voices that have been insisting on a more mature field of SE, where rigorous evaluation is part of any developed research. In fact, although the origins of SE can be dated back to the famous 1968 NATO conference, it cannot be said to have become an empirical science until the 1970s, with the advent of Basili's work [3]. Nevertheless, even though there has long been a positive encouragement on the application of adequate evaluation in SE, a 2005 survey showed that only 1.9% of the work published on prestigious SE venues until that date have applied controlled experiments [4].

The Brazilian Symposium on Software Engineering (SBES, from the Portuguese acronym) is the premier Brazilian SE conference. SBES has been held annually since 1987, summing up 24 editions up to 2010. Lately, the conference has been gathering nearly 500 people, including researchers, students, and practitioners working on the field [5]. If the Brazilian symposium is to grow into a respectable community, we must also take into account how research work published in SBES is being evaluated.

Software testing is a very important and prominent SE subfield and several papers published in the SBES history fall into this category. Since cost constraints and high effectiveness goals are common within software testing, every novel approach has to be adequately evaluated according to these characteristics to be deemed useful or not. In fact, experimentation and other types of evaluation is an essential part of research in software testing [6]. For instance, at many times one is interested in comparing the fault-detection effectiveness of testing criteria used to derive test cases. In this case experimentation is a handy tool to obtain evidence about this question [6].

In this paper we present a survey that serves as an initial assessment of the dissemination of adequate evaluation of software testing research in SBES. We have analyzed the 24 available SBES proceedings and characterized the evaluation studies presented in papers related to software testing. We categorized the papers within the software testing field, collected information about authors and affiliations, and classified the presented evaluations, looking into its evolution along the symposium's history.

Our assessment shows that the number of evaluations presented in SBES papers is increasing significantly along the years. However, our data also shows that there is still room for improvement in this area, specially with respect to the rigor of the conducted studies. In particular, we notice a lack of publications reporting controlled or quasi experiments involving software testing research. The remainder of this paper is structured as follows. Section II presents basic concepts about evaluation in SE and software testing and Section III describes the research method used to select and classify the papers in the survey and other characteristics of our study. Section IV presents the results of our survey and Section V discusses such results. Finally, Section VI concludes the paper, presenting some ideas to improve software testing evaluation in the future of SBES.

II. Background

There are many types of evaluation studies that can be applied to software engineering research. Zannier et al. [7] classifies them into the following: controlled experiment, quasi experiment, case study, exploratory case study, experience report, meta-analysis, example application, survey, and discussion. Each of these types has a different level of rigor. In this paper we use the same classification to characterize evaluation studies in software testing research published in SBES. In the following we synthesize Zannier et al.'s classification.

Controlled experiments apply random assignment of treatments to subjects, contain large sample sizes (> 10 subjects), formulate hypotheses, select an independent variable, and apply random sampling; while *quasi experiments* are controlled experiments with one or more of its characteristics missing. *Case studies* state a research question and unit(s) of analysis, report a logic link between data and propositions, provide criteria for interpreting findings, and are performed in "real-world" scenarios; while *exploratory case studies* are case studies with one or more of its characteristics missing.

Experience reports are retrospective reports with no propositions, do not necessarily contain answers to how or why some findings were attained, and often include lessons learned. *Meta-Analyses* analyze a body of similar studies to reach a common result; *example applications* only describe an application to assist the definition of the approaches (these are commonly alleged as "evaluations" or "validations" of the study). *Surveys* collect answers to structured or unstructured questionnaires given to participants; while *discussions* provide qualitative, textual, and opinion-related evaluation.

It is important to notice that, in general, SE is regarded as a discipline that needs to improve on the use of experiments and more rigorous forms of evaluation. However, as reported by Zannier et al. [7], the community seems to be evolving significantly with this respect over the years. For instance, over the lifetime of the International Conference on Software Engineering (ICSE) until 2006, there was a significant increase in the number of papers with an evaluation component. If this is true with respect to the international community, an important question is whether it it also holds to more local communities such as SBES, with respect to more specific fields such as Software Testing.

A. Software Testing Research and Evaluation

Software testing can be defined as the execution of a program against test cases with the intent of revealing faults [8]. The different testing techniques are defined based on the artifact used to derive test cases. Functional – or black-box – testing derives test cases from the specification or description of a program; structural – or white-box – testing derives test cases from implementations; fault-based testing derives test cases from fault models based on common mistakes committed by programmers; and model-based testing derives test cases from system specification models. To deem a software system *correct*, one could test every possible element of the system's input domain and check whether the output is consistent with the expected output. However, even for simple programs this is usually infeasible, because the input domains tend to be very large (imagine, for instance, the input space of a compiler system) [8]. Therefore, a large portion of testing research focus on proposing ways to select meaningful subsets of test cases to enhance the chance of revealing faults. Based on the categories of testing techniques described above, several testing selection criteria were proposed [9].

Besides testing techniques and criteria, there are many other aspects involved in the testing activity. For instance, in general, it is too expensive to test programs manually; therefore, software testing usually relies on tools to automate the test case generation, execution, and results gathering. After faults are revealed while testing the programs, they must be localized and fixed. This activity is usually not included under the *software testing* activity, being called *debugging*. Since it is closely related to testing, we decided to include papers concerned with it in our survey. Other topics that are important to software testing and were included are the following: *fault-injection*[1], which consists in intentionally introducing known failures into the system during its execution to evaluate if the system is robust enough to recover without crashing [10]; *regression testing*, which consists in selectively retesting a system to verify whether modifications have not caused unwanted effects [11]; and *testing strategy*, which consists in the way by

[1]Usually related to the system's *fault tolerance*.

which test case design methodologies are combined to provide an effective testing activity [8].

Different types of software testing research reclaim different types of evaluation. Some proposals might be easier to evaluate, while others might require more work. For instance, evaluating the effectiveness of a testing criterion might require the use of real applications, a large pool of test cases, and random selection of tests not to introduce bias in the test case generation (for instance, see [12]). In other cases, it might require only the simulation of an algorithm with different configurations. For instance, in the case of some approaches for automated test case generation, evaluation may consists only in running an implementation with different configurations and comparing the outcomes, which is an experiment easier to configure than a test criteria study. In any case, evaluation studies are very important for software testing, because we need approaches that are at the same time effective but also feasible. A researcher can only have evidence that a testing approach is useful or not only when it is adequately measured with respect to effectiveness and effort factors.

III. Study Setup

A. Research goals

Our main goal is to investigate the dissemination of software testing evaluations in SBES. We assess the increase in performed evaluations in terms of the percentage of papers with an evaluation component over the total number of published papers in a year. We define a paper as containing an evaluation component when it presents at least a study involving subjects – humans, programs, or specifications – and not only a single application example (studies that do not fall into the category of "application example" as defined in Section II).

A complementary goal of this survey is to characterize the software testing community that publishes papers in SBES. We do this by analyzing authors, schools, and topics involved in the selected publications.

B. Paper selection

The selection of the papers analyzed in this study was based on the proceedings of the 24 SBES editions, from 1987 to 2010. The papers published from 1987 to 1998, in 2000 and in 2003 are available only in printed format. Papers published from 1999 to 2008 (apart from 2000 and 2003) are also available online[2]. As of 2009, the SBES proceedings are also available in the IEEE Digital Library[3].

[2]http://www.lbd.dcc.ufmg.br:8080/bdbcomp/servlet/PesquisaEvento?evento=sbes – accessed in 02/08/2011.

[3]http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5336057 and http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5628346 – accessed in 02/08/2011.

The paper selection process was inspired by a process for running systematic mapping studies [13]. The first three authors of this paper performed the paper selection iteratively. In the first iteration we used an inclusion criterion which defines that relevant papers must be related to software testing. Firstly, we performed a preliminary analysis of the papers published in sessions related to *Verification, Validation and Testing* (VV&T) of the main track of each SBES proceedings. In the next step, we searched for papers related to software testing in the remaining parts of the proceedings, since some testing papers were allocated to other sessions (*e.g.* a paper on testing aspect-oriented programs can be allocated to the AOP session). Note that we have not considered SBES satellite events such as Tool Sessions and collocated workshops, given that our main goal was to evaluate SBES – *i.e.* its main track – as a vehicle to disseminate testing-related research that performs some kind of evaluation.

The second iteration was carried out by the same authors, hereafter called the *reviewers*. The identified papers were distributed amongst reviewers so that they could read the title, abstract and introduction aiming at identifying the papers that contained an evaluation component. In the third iteration the reviewers performed further analysis of the papers identified in the previous iteration to exclude "false positives" (*e.g.* papers that addressed bug fixing – *i.e.* maintenance – or other organizational matters). At this point, for each paper we collected relevant information into tables. Extracted details included authors' names, affiliations, testing approach addressed by the paper and, when applicable, the type and attributes of the reported evaluation. The next section presents some details about the classification schema we applied for the selected papers.

C. Paper classification

Since there are many elements involved in software testing, there are also several types of testing-related publications. Some of them focus on the proposal of a testing criterion, others focus on automating some aspect of the testing activity. There are also papers that evaluate testing criteria or varied testing strategies. Therefore, in a survey like the one reported herein, the large range of topics covered by software testing papers requires the adoption of some classification system that enables us to categorize the publications.

We classified the testing-related papers published in SBES according to two dimensions: *Technique* and *Type*. The first addresses the main testing-related technique investigated in a paper. Examples are white-box testing and automated test case generation. The *Type* dimension characterizes a paper according to its nature. While a paper may propose a software testing approach such as a novel family of criteria, another may be concerned with evaluating such family of criteria with respect to its efficacy and effectiveness.

The categories related to *Technique* are the following:

A	:	Automated test case generation
B	:	Black-box (Functional) Testing
D	:	Debugging
F	:	Fault-based Testing
I	:	Fault Injection and Fault Tolerance
M	:	Model-based Testing
R	:	Regression Testing
S	:	Testing Strategy
W	:	White-box (structural) Testing

The categories related to *Type* are the following:

A	:	Approach proposal,
E	:	Evaluation, when the paper evaluates some aspect of software testing
T	:	Tool, when the paper describes some testing tool or testing infrastructure implementation

As we will see in the next section, in some cases a paper can be classified in two categories of *Technique*. For example, a paper that describes an approach for deriving functional test cases based on the system's models is classified as **B** and **M**. Nevertheless, as far as possible we tried to assign a single category to each paper, according to the best related technique.

With respect to *Type*, we classified the papers according to their main contribution. For instance, in some cases a paper may propose a testing approach and at the same time evaluate it by means of an experiment. However, since the main contribution of the paper is the approach itself, we would classify such publication as an *approach proposal* paper, and not an *evaluation* paper.

IV. Results and Analysis

In this section we present the data gathered in our survey. We analyzed all available SBES proceedings from 1987 to 2010, and performed the selection process mentioned in the previous section. Firstly, we selected papers related to software testing; and secondly, we identified only those that contained an evaluation component. Among the papers with an evaluation component, we then identified the ones that presented more rigorous evaluation studies.

A. Selected papers

From the available SBES proceedings, we selected 55 papers that publish studies related to software testing. Tables III and IV present all papers and information about each. For each paper we present the year of publication, the title[4], the authors and their affiliation, the related testing technique and type, the evaluation type according to Zannier's classification [7] (or n/a in the absence of an evaluation component), and the type and number of subjects used in the evaluation. Such information is used to characterize the software testing community that publishes in the symposium, and to analyze the evolution of software testing evaluation studies published in SBES along the years.

[4]The titles of the papers written in Portuguese were translated to English. A list containing the original titles is presented in the Appendix at the end of this paper.

B. Community Characterization

In this section we present the results of our survey with respect to the characterization of the community that has published software testing papers in the history of SBES. With respect to scholars, there are 87 authors that appear in the software testing publications of our survey. Table I presents the top 16 ranked scholars. We selected only authors with more than two software testing papers presented at the symposium editions. To have an idea of how the same authors evaluated their published studies, in the same table we place the number of papers that present an evaluation component.

TABLE I
Top 16 Scholars Publishing Software Testing Research in SBES (1987-2010)

Scholar	# pubs.	# eval. pubs.
J. C. Maldonado	31	14
M. Jino	12	5
P. C. Masiero	10	3
M. E. Delamaro	7	2
S. R. Vergilio	6	1
S. C. P. F. Fabbri	5	2
O. A. L. Lemos	5	2
A. S. Simão	5	2
A. M. Price	4	0
S. R. S. Sousa	4	2
A. M. R. Vincenzi	4	2
A. Pasquini	3	3
E. Martins	3	2
A. N. Crespo	3	3
M. L. Chaim	3	0

There are 35 institutions involved in the papers of our survey. Table II presents the top 15 ranked institutions. We show institutions that appear at least in two software testing papers published in the symposium. Similar to the data for the authors, we also place the number of papers published by researchers with that institution that present an evaluation component.

With respect to the covered topics and types of software testing papers published in the history of SBES, Figure 1 presents charts with the data for each axis of our classification system. Note that the top covered topics were White-box testing, Fault-based testing, and Test case generation. This shows a trend of the software testing community that publishes in SBES and indicates topics that have been less covered

TABLE III
TESTING-RELATED PAPERS PUBLISHED IN SBES PROCEEDINGS (1/2).

Year	#	Title	Authors	Affiliation	Tech	Type	Eval	Subject type	#Subj.
1987	1	Visualizing the Control Flow of Programs*	A. M. Price; F. Garcia; C. Purper	UFRGS	W	A	n/a	-	-
	2	Controlled Execution of Programs*	J. R. V. da Silva; D. L. Segalin; R. Vieira; P. A. Azevedo;	UFRGS	D	T	n/a	-	-
	3	PROTESTE: Design of a Tool for Program Testing*	A. M. Price; C. Purper; F. Garcia	UFRGS	B	T	n/a	-	-
1988	4	Test Case Selection based on Data Flow through the Potential-Uses Criteria*	J. C. Maldonado; M. L. Chaim; M. Jino;	ICMC/USP, DCA/FEE/UNICAMP	W	A	n/a	-	-
1989	5	Modeling and Determining Potential DU-Paths through Data Flow Analysis*	M. L. Chaim; J. C. Maldonado; M. Jino	DCAI/UNICAMP, ICMC/USP	W	A	n/a	-	-
1990	6	Environment to Support Structural Testing of Programs*	A. M. A. Price; A. F. Zorzo	UFRGS	W	T	n/a	-	-
1992	7	Unfeasible Paths in the Testing Activity Automation*	S. R. Vergilio; J. C. Maldonado; M. Jino	DCAI/UNICAMP, ICMC/USP	W	A	Example Application	programs	-
	8	Potential-Uses Criteria: Analyzing the Application of a Benchmark*	J. C. Maldonado; S. R. Vergilio; M. L. Chaim; M. Jino	ICMC/USP, DCAI/UNICAMP	W	A	Example Application	-	-
1993	9	A Strategy for Generating Test Data*	S. R. Vergilio; J. C. Maldonado; M. Jino	UFPR, ICMC/USP, FEEC/UNICAMP	A	A	Example Application	-	-
	10	Evaluation of the Cost of Alternate Mutation Strategies	A. P. Mathur; Weichen E. Wong	Purdue University	F	E	Exploratory Case Study	programs	4
1994	11	Applying Mutant Analysis to the Validation of Petri Net-based Specifications*	S. C. P. F. Fabbri; J. C. Maldonado; P. C. Masiero; M. E. Delamaro	UFSCar, ICMC/USP, IFSC/USP	F	A	Example Application	-	-
	12	Constrained Mutation in C Programs	W. E. Wong; J. C. Maldonado; M. E. Delamaro; A. P. Mathur	Hughes Network Systems, ICMC/USP, Purdue University	F	E	Exploratory Case Study	programs	5
	13	Unfeasible Paths in Integration Testing: Characterization, Estimation and Determination*	S. R. Vergilio; J. C. Maldonado; M. Jino	UFPR, ICMC/USP, FEEC/UNICAMP	W	A	n/a	-	-
1995	14	Test Data Generation: A Strategy that Preserves Criteria Hierarchy*	S. R. Vergilio; J. C. Maldonado; M. Jino	UFPR, ICMC/USP, FEEC/UNICAMP	W	A	n/a	-	-
	15	Integrating Fault Injection and Formal Testing in the Validation of Fault Tolerance*	E. Martins	UNICAMP	I	A	n/a	-	-
	16	A G-Net Based Environment for Logical and Timing Analysis of Software Systems	A. Perkusich; J. C. A. Figueiredo	UFPB	M	T	n/a	-	-
1997	17	Potential-Uses Criteria Coverage and Software Reliability*	A. N. Crespo; A. Pasquini; M. Jino; J. C. Maldonado	FEEC/UNICAMP, ICMC/USP	W	E	Case Study	program	1
	18	Strategy for Test Data Generation based on Symbolic and Dynamic Program Analysis*	J. S. Herbert; A. M. A. Price	UFRGS	A	A	n/a	-	-
	19	Integration Testing: Design of Operators for the Interface Mutation Criterion*	M. E. Delamaro; J. C. Maldonado	IFSC/USP, ICMC/USP	F	A	n/a	-	-
	20	Applying the Mutant Analysis Criterion to the Validation of Statecharts-based Specifications*	S. C. P. F. Fabbri; J. C. Maldonado; P. C. Masiero	UFSCar, ICMC/USP	F	A	Example Application	-	-
	21	Evaluating the Impact of Test Set Minimization on the Cost and Efficacy of the Mutant Analysis Criterion*	S. R. S. Souza; J. C. Maldonado	UEPG, ICMC/USP	F	E	Exploratory Case Study	benchmark / program	1 / 5
1999	22	Automatic Data Generation and Non-Executabity Handling in Structural Software Testing*	P. M. S. Bueno; M. Jino	FEEC/UNICAMP	A, W	A	Exploratory Case Study	program	4
	23	A Study of the Cost Evaluation of Applying Mutant Analysis to the Validation of Finite State Machines*	R. A. Carvalho; S. C. P. F. Fabbri; J. C. Maldonado	UFSCar, ICMC/USP	F	E	Exploratory Case Study	FSM	10
	24	Interface Sufficient Operators: A Case Study*	A. M. R. Vincenzi; J. C. Maldonado; E. F. Barbosa; M. E. Delamaro	ICMC/USP, UEM	F	E	Exploratory Case Study	program	5
2000	25	A Binomial Software Reliability Model Based on Coverage of Structural Testing Criteria	A. N. Crespo; M. Jino; A. Pasquini; J. C. Maldonado	USF Campinas, ENEA Roma, FEEC/UNICAMP, ICMC/USP	W	A	Exploratory Case Study	program (a 10-KLOC program)	1
	26	Proteum-RS/PN: A Tool to Support Edition, Simulation and Validation of Petri Nets based on Mutation	A. S. Simão; J. C. Maldonado; S. C. P. F. Fabbri	ICMC/USP	F	T	Exploratory Case Study	specificatio ns (Petri nets)	5
	27	Structural Software Testing: An Approach for Relational Database Applications*	E. S. Spoto; M. Jino; J. C. Maldonado	UEM, FEEC/UNICAMP, ICMC/USP	W	A	Exploratory Case Study	programs	4

* Translated to English (original title in Portuguese)

Legend for Tech:
A: Automatic test case generation B: Functional (black-box) testing D: Debugging F: Fault-based testing I: Fault injection/ tolerance
M: Model-based testing R: Regression testing S: Testing Strategy W: Structural (white-box) testing

Legend for Type:
A: Approach proposal E: Evaluation
T: Tool and infrastructure

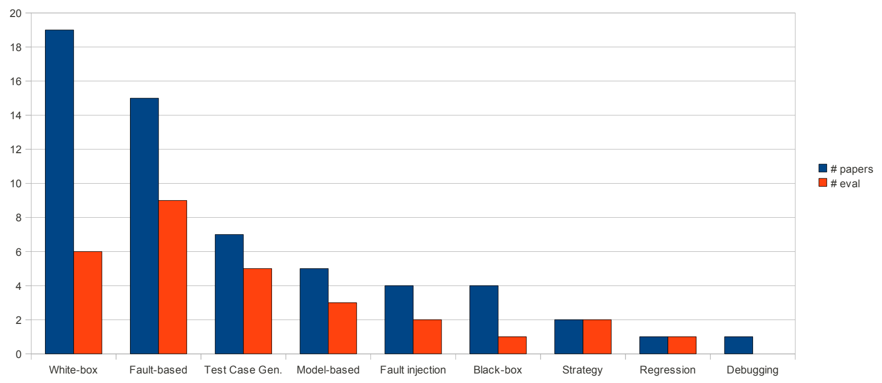
TABLE IV
TESTING-RELATED PAPERS PUBLISHED IN SBES PROCEEDINGS (2/2).

Year	#	Title	Authors	Affiliation	Tech	Type	Eval	Subject type	#Subj.
2001	28	Mudel: A Language and a System for Describing and Generating Mutants	A. S. Simão; J. C. Maldonado	ICMC/USP	F	T	n/a	-	-
	29	FCCE: A Testing Criteria Family for the Validation of Systems Specified in Estelle*	S. R. S. Souza; J. C. Maldonado; S. C. P. F. Fabbri	UEPG, ICMC/USP, UFSCar	W	A	n/a	-	-
	30	Mutant Operators for Testing Concurrent Java Programs	M. E. Delamaro; M. Pezzè; A. M. R. Vincenzi; J. C. Maldonado	UEM, Università di Milano, ICMC/USP	F	A	n/a	-	-
2002	31	Selection and Evaluation of Test Data Sets Based on Genetic Programming	M. C. F. P. Emer; S. R. Vergilio	UFPR	F	A	Exploratory Case Study	programs	4
	32	Tests and Code Generation for Web Systems*	E. Aranha; P. Borba	UFPE	A	T	Case Study	program	1
2003	33	A Method for Functional Testing for the Verification of Components*	C. M. Farias; P. D. L. Machado	UFCG	B	A	n/a	-	-
	34	A Family of Coverage Testing Criteria for Coloured Petri Nets	A. S. Simão; S. R. S. Souza; J. C. Maldonado	ICMC/USP	F	A	n/a	-	-
2004	35	Unit Testing of Aspect-Oriented Programs*	O. A. L. Lemos; A. M. R. Vincenzi; J. C. Maldonado; P. C. Masiero	ICMC-USP, UNIVEM	W	A	Exploratory Case Study	-	-
	36	Reuse in the Software Testing Activity to Reduce VV&T Cost and Effort in the Development and Re-engineering of Software*	M. I. Cagnin; J. C. Maldonado; A. Chan; R. Penteado; F. Germano	ICMC-USP, UFSCar	R	T	Exploratory Case Study	humans	1
	37	A Methodology for the Verification of Partial Systems Modeled with Object-Based Graph Grammar*	F. L. Dotti; F. Pasini; O. M. Santos	PUC-RS	M	A	n/a	-	-
2005	38	Distributed Environment of Communication Fault Injection for Testing of Network Java Applications*	J. Gerchman; G. Jacques-Silva; R. J. Drebes; T. S. Weber	UFRGS	I	T	n/a	-	-
	39	Automatic test data generation for path testing using a new stochastic algorithm	B. T. Abreu; E. Martins; F. L. Sousa	IC/UNICAMP, INPE	A	E	Exploratory Case Study	programs	1
	40	An Aspect-based Tool for Functional Testing of Java Programs*	A. D. Rocha; A. S. Simão; J. C. Maldonado; P. C. Masiero	ICMC-USP	B	T	n/a	-	-
2006	41	Automatic Generation of Test Drivers and Stubs for JUnit based on U2TP Specifications*	L. Biasi; K. Becker	PUCRS	A, M	T	Exploratory Case Study	program	1
2007	42	Static Analysis of Java Bytecode for Domain-Specific Software Testing	M. E. Delamaro; P. A. Nardi; O. A. L. Lemos; P. C. Masiero; E. S. Spoto; J. C. Maldonado; A. M. R. Vincenzi	UNIVEM, ICMC/USP, UNISANTOS	W	A	n/a	-	-
	43	Generalized Extremal Optimization: A Competitive Algorithm for Test Data Generation	B. T. Abreu; E. Martins; F. L. Souza	IC/UNICAMP, INPE	A	E	Exploratory Case Study	programs	7
	44	Experimental Evaluation of Coverage Criteria for FSM-Based Testing	A. S. Simão; A. Petrenko; J. C. Maldonado	ICMC/USP, Centre de Recherche Informatique de Montreal (CRIM)	M	E	Quasi Experiment	specificatio ns (FSMs)	?
	45	Pairwise Structural Testing of Object and Aspect-Oriented Java Programs	I. G. Franchin; O. A. L. Lemos; P. C. Masiero	ICMC/USP	W	A	n/a	-	-
	46	Integration Testing of Aspect-Oriented Programs: A Characterization Study to Evaluate how to Minimize the Number of Stubs	R. Ré; P. C. Masiero	UTFPR Campo Mourão and ICMC/USP	S	A	Exploratory Case Study	program	1
2008	47	Using Similarity Functions to Reduce Test Suites in Strategies for Model-based Testing*	E. G. Cartaxo; P. D. L. Machado; F. G. Oliveira Neto; J. F. S. Ouriques	UFCG	M, B	A	Exploratory Case Study	programs	3
	48	Generation of Faultloads for Testing Campaigns with Fault Injection from UML Testing Models*	J. Gerchman; C. Menegotto; T. S. Weber	UFRGS	I	A	Exploratory Case Study	-	-
	49	Obtaining Trustworthy Test Results in Multi-threaded Systems	A. Dantas; M. Gaudencio; F. Brasileiro; W. Cirne	UFCG	I	A	Exploratory Case Study	programs	1
	50	Integration Testing of Aspect-Oriented Programs: a Structural Pointcut-Based Approach	O. A. L. Lemos; P. C. Masiero	ICMC-USP	W	A	Example Application	-	-
	51	A Catalog of Stubs to Support the Integration Testing of Aspect-Oriented Programs *	R. Ré; A. L. S. Domingues; P. C. Masiero	UTFPR, ICMC-USP	S	E	Exploratory Case Study	-	-
2009	52	Applying Code Coverage Approach to an Infinite Failure Software Reliability Mode	A. N. Crespo; A. Pasquini; M. Jino; J. C. Maldonado	Deep Blue (Italy), ICMC-USP, FEEC-UNICAMP, CenPRA-MCT	W	A	Quasi Experiment	programs	1
2010	53	Characterising Faults in Aspect-Oriented Programs: Towards Filling the Gap between Theory and Practice	F. C. Ferrari; O. A. L. Lemos; R. Burrows; A. F. Garcia; J. C. Maldonado	ICMC-USP, Univ. of Lancaster, PUC-Rio, UNIFESP	F	E	Exploratory Case Study	programs	3
	54	Built-in structural testing of web services	M. M. Eler; M. E. Delamaro; J. C. Maldonado; P. C. Masiero	ICMC-USP	W	A	n/a	-	-
	55	Mutation Testing in Procedural and Object-Oriented Paradigms: An Evaluation of Data Structure Programs*	D. N. Campanha; S. R. S. Sousa; J. C. Maldonado	ICMC-USP	F	E	Experiment	programs	32

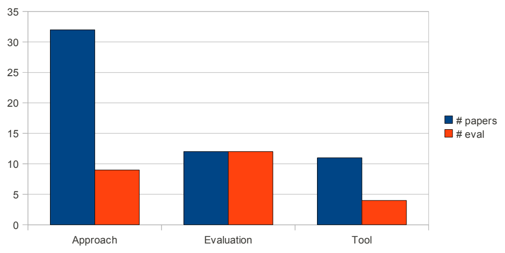
* Translated to English (original title in Portuguese)

Legend for Tech:
A: Automatic test case generation B: Functional (black-box) testing D: Debugging F: Fault-based testing I: Fault injection/ tolerance
M: Model-based testing R: Regression testing S: Testing Strategy W: Structural (white-box) testing

Legend for Type:
A: Approach proposal E: Evaluation
T: Tool and infrastructure



(a) Papers by Technique



(b) Papers by Type

Fig. 1. Charts of the covered topics and types of software testing papers.

TABLE II
TOP 15 INSTITUTIONS PUBLISHING SOFTWARE TESTING RESEARCH IN SBES (1987-2010)

Institution	# pubs.	# eval. pubs.
ICMC/USP	35	16
FEEC/UNICAMP	12	5
UFRGS	7	1
UFSCar	5	2
UFPR	4	1
UFCG	3	2
UEM	3	1
UTFPR	2	2
UNIVEM	2	1
IC/UNICAMP	2	2
UEPG	2	1
Purdue University	2	2
PUC-RS	2	1
INPE	2	2
IFSC/USP	2	0

in its lifetime. With respect to type, note that there are many more papers proposing approaches, and less focused on evaluations and tools. This also indicates a publication gap of experimentation papers, which are very important in this field.

C. Characterization and Evolution of Evaluation Studies

With respect to the evaluation studies present in the surveyed papers, note that 33 out of 55 papers performed some kind of evaluation, and 26 out of these 33 were categorized as experiments, quasi experiments, case studies or exploratory case studies. This means that approximately 47% of the whole set of analyzed papers contained an evaluation component (*i.e.* not only an application example).

To show how the numbers of papers with evaluation studies have evolved over the history of SBES, we analyze the paper data aggregated per triennium. We did this because we noticed that an annual analysis would present too much variability. Table V shows the number of papers that presented evaluation studies over the total number of published papers for each triennium. We covered all triennia from 1987 to 2010. Note that the 1990-1993 triennium skips 1991, because that year had no software testing publications, as can be noticed in Tables III and IV. The same applies to the 1994-1997 triennium.

To provide a visual representation of the data, Figure 2 presents a chart of the numbers presented in Table V. We draw lines between the data points only to provide an idea of the growth rate between periods. Note that the number of papers that present evaluation studies have significantly increased over the triennia. There is a noticeable upward trend, except for 1999-2001 triennium, which is an interesting outlier (note

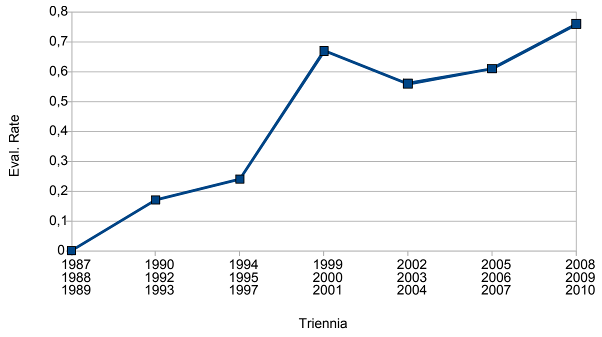


Fig. 2. Chart of the growth rate of papers with evaluation components per triennium.

TABLE V
EVOLUTION OF THE EVALUATION STUDIES.

Triennium	Eval. rate
1987-1989	0.00
1990-1993	0.17
1994-1997	0.24
1999-2001	0.67
2002-2004	0.56
2005-2007	0.61
2008-2010	0.76

the increase from the antecedent triennium and the decrease to the following triennium). In Section V we provide an in-depth analysis of these numbers.

Figure 3 presents a chart with the distribution of evaluation studies among the categories we analyzed. Note that the mass majority of papers applied exploratory case studies (21 papers), while only 2 papers presented case studies and other 2 quasi experiments, and only 1 paper presented a controlled experiment. This shows that while SBES has promoted the increase in application of evaluation studies along the years, the rigor of these studies were not strong. The next section discusses related issues in more detail.

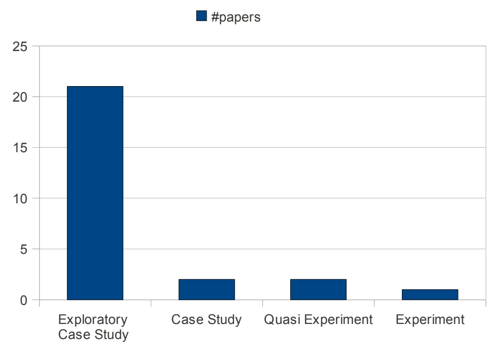


Fig. 3. Distribution of papers per evaluation type.

V. DISCUSSIONS

An interesting analysis to be conducted is the observation of the evolution in the rate of software testing papers that present evaluation components published along the SBES lifetime. Looking at Table V and Figure 2, we can see that no evaluations were presented in the first period, but in the second period, 17% of the papers had an evaluation component. In the last period, 71% of the papers presented evaluation components. To obtain the observed growth rate of the total period, we can use the following equation: $OR = \frac{FT-LT}{FT}$, where OR is the Observed Rate, FT is the First Triennium to present evaluations (1990-1993), and LT is the Last Triennium (2008-2010). This analysis shows that there was a 347% increase in the rate of evaluated papers along all the period. The average growth observed between triennia (i_3) can be computed by the following equation: $i_3 = (1 + OR)^{\frac{1}{5}} - 1$. Assigning 3.47 to OR, we reach the average growth of 34.92% in the rate of evaluated papers within subsequent triennia. These numbers suggest that, if no particular changes occur in the field, in the next triennium every – or close to every – software testing paper published in SBES will contain an evaluation component.

As observed in Section IV, there was an interesting outlier occurring in the 1999-2001 triennium. We believe this outlier can be explained by an increase in the awareness of the need for more serious evaluations in those years, which must have impacted in the number of evaluation studies. In fact, in 2005 an international survey of controlled experiments in software engineering [4] showed that 2000 was the year with the highest number of papers describing experiments, both in absolute and relative numbers. The subsequent years of 2001 and 2002 presented a decrease in the number of reported controlled experiments. The similarity perceived here with respect to software testing research shows a connection between the SBES community with the international Software Engineering community.

With respect to the distribution of papers according to the discussed testing technique, Figure 1(a) reveals that there is a gap between the number of publications related to the most frequent topic (white-box testing) and the number of evaluations: only 6 out of 19 white-box testing papers, *i.e.* 31%, reported some kind of evaluation. Other topics presented a better correlation between the number of publications and evaluations: strategy (100%), regression (100%), and test case generation (71%), for example.

Regarding the *Type*-related classification, Figure 1(b) shows that the novel approaches are hardly ever evaluated in the same paper: only 9 out of 32 papers, *i.e.* 28%, reported some kind of evaluation. This is an evidence that several approaches have been proposed in SBES but there are not a great concern with their evaluation. Papers describing some testing tool or related infrastructure implementation have also resulted in a low correlation between publications and evaluations (36%). Obviously, all papers aiming at evaluating some aspect of software testing present an evaluation study, however we decided to keep the column *Evaluation* in Figure 1(b) for the sake of completeness.

VI. Conclusions

This paper presented a survey with a historical perspective on the application of evaluation studies in software testing papers published in the Brazilian Symposium on Software Engineering (SBES). We have analyzed publications in the 24-year history of the symposium. Our data shows that our community has significantly improved in this subject, with a significant increase in the rate of evaluated testing-related publications. On the other hand, there is still much room to evolve with respect to the rigor of the performed evaluations. For instance, from a total of 26 papers that include an evaluation component, only a single software testing paper published in the history of the symposium has applied a rigorous controlled experiment, whereas only two others have presented results of quasi experiments. These three papers represent only 5% of the total number of testing-related papers published in the SBES' main research tracks.

Our survey also provides other interesting insights. For instance, we found out that publications about test case generation approaches were one of the most frequent to present an evaluation component (71%), and only 31% of papers on white-box testing – the dominant testing topic in SBES – have evaluated their proposals. This is consistent with the difference in difficulty in applying experiments for research work on those topics commented in Section II.

Another interesting result that showed up in our data was an outlier with respect to papers containing evaluations: the proceedings of the 1999-2001 triennium presented an uncommon increase in the rate of evaluated papers compared to the antecedent triennium. This result is consistent with a 2005 international survey of software engineering controlled experiments, which showed that 2000 was the year with the highest number of reported experiments in the analyzed period (1993-2002) [4]. In the future we intend to investigate further these questions, by looking into the correlation between SBES and the international software engineering/testing community.

Acknowledgments

We would like to thank Prof. Francisco Marcelo da Rocha from UNIFESP, who provided valuable help in the statistical procedures. We would also like to thank the financial support received from FAPESP (grants 2010/15540-2, 05/55403-6 and 2008/03252-2), CAPES and CNPq.

References

[1] D. G. Feitelson, "Experimental computer science (guest editor's introduction)," *Commun. ACM*, vol. 50, pp. 24–26, November 2007.

[2] P. A. Freeman, "Back to experimentation," *Commun. ACM*, vol. 51, pp. 21–22, January 2008.

[3] B. Boehm, H. D. Rombach, and M. V. Zelkowitz, *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[4] D. I. K. Sjoberg, J. E. Hannay, O. Hansen, V. By Kampenes, A. Karahasanovic, N.-K. Liborg, and A. C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE Trans. Softw. Eng.*, vol. 31, pp. 733–753, September 2005.

[5] A. Garcia, "CBSoft 2011 - SBES - Call for papers," 2011, available at: http://www.each.usp.br/cbsoft2011/ingles/sbes/chamada_sbes_en.html (accessed 16/05/2011).

[6] J. H. Andrews, L. C. Briand, and Y. Labiche, "Is mutation an appropriate tool for testing experiments?" in *Proc. of the 27th Int'l conference on Software engineering*, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 402–411.

[7] C. Zannier, G. Melnik, and F. Maurer, "On the success of empirical studies in the int'l conference on software engineering," in *Proc. of the 28th Int'l conference on Software engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 341–350.

[8] G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas, *The Art of Software Testing*, 2nd ed. John Wiley & Sons, 2004.

[9] A. P. Mathur, *Foundations of Software Testing*, 1st ed. Addison-Wesley Professional, 2008.

[10] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *IEEE Computer*, vol. 30, pp. 75–82, 1997.

[11] IEEE, "IEEE standard glossary of software engineering terminology," Institute of Electric and Electronic Engineers, Standard 610.12, 1990.

[12] Z. Lai, S. C. Cheung, and W. K. Chan, "Inter-context control-flow and data-flow test adequacy criteria for nesc applications," in *Proc. of the 16th ACM SIGSOFT Int'l Symposium on Foundations of software engineering*, ser. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, 2008, pp. 94–104. [Online]. Available: http://doi.acm.org/10.1145/1453101.1453115

[13] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. of the 12^{th} Int'l Conference on Evaluation and Assessment in Software Engineering (EASE)*. Bari - Italy: The British Computer Society, 2008, pp. 1–10.

APPENDIX: ORIGINAL PAPER TITLES

#	Translated Title	Original Title
1	Visualizing the Control Flow of Programs	Visualizando o Fluxo de Controle de Programas
2	Controlled Execution of Programs	Execução Controlada de Programas
3	PROTESTE: Design of a Tool for Program Testing	PROTESTE: Projeto de uma Ferramenta para Teste de Programas
4	Test Case Selection based on Data Flow through the Potential-Uses Criteria	Seleção de Casos de Teste Baseada em Fluxo de Dados Através dos Critérios Potenciais-Usos
5	Modeling and Determining Potential DU-Paths through Data Flow Analysis	Modelando a Determinação de Potenciais DU-Caminhos através da Análise de Fluxo de Dados
6	Environment to Support Structural Testing of Programs	Ambiente de Apoio ao Teste Estrutural de Programas
7	Unfeasible Paths in the Testing Activity Automation	Caminhos Não Executáveis na Automação das Atividades de Teste
8	Potential-Uses Criteria: Analyzing the Application of a Benchmark	Critérios Potenciais Usos: Análise de Aplicação de um Benchmark
9	A Strategy for Generating Test Data	Uma Estratégia para Geração de Dados de Teste
11	Applying Mutant Analysis to the Validation of Petri Net-based Specifications	Aplicação da Análise de Mutantes na Validação de Especificações Baseadas em Redes de Petri
13	Unfeasible Paths in Integration Testing: Characterization, Estimation and Determination	Caminhos Não Executáveis no Teste de Integração: Caracterização, Previsão e Determinação
14	Test Data Generation: A Strategy that Preserves Criteria Hierarchy	Geração de Dados de Teste: Uma Estratégia que Preserva a Hierarquia de Critérios
15	Integrating Fault Injection and Formal Testing in the Validation of Fault Tolerance	Integrando Injeção de Falhas e Testes Formais na Validação de Tolerância a Falhas
17	Potential-Uses Criteria Coverage and Software Reliability	Cobertura dos Critérios Potenciais-Usos e a Confiabilidade do Software
18	Strategy for Test Data Generation based on Symbolic and Dynamic Program Analysis	Estratégia de Geração de Dados de Teste Baseada na Análise Simbólica e Dinâmica do Programa
19	Integration Testing: Design of Operators for the Interface Mutation Criterion	Teste de Integração: Projeto de Operadores para o Critério Mutação de Interface
20	Applying the Mutant Analysis Criterion to the Validation of Statecharts-based Specifications	Aplicação do Critério Análise de Mutantes na Validação de Especificações Baseadas em Statecharts
21	Evaluating the Impact of Test Set Minimization on the Cost and Efficacy of the Mutant Analysis Criterion	Avaliação do Impacto da Minimização de Conjuntos de Casos de Teste no Custo e Eficácia do Critério Análise de Mutantes
22	Automatic Data Generation and Non-Executabiity Handling in Structural Software Testing	Geração Automática de Dados e Tratamento de Não Executabilidade no Teste Estrutural de Software
23	A Study of the Cost Evaluation of Applying Mutant Analysis to the Validation of Finite State Machines	Um Estudo sobre a Avaliação do Custo de Aplicação na Análise de Mutantes na Validação de Máquinas de Estados Finitos
24	Interface Sufficient Operators: A Case Study	Operadores Essenciais de Interface: Um Estudo de Caso
27	Structural Software Testing: An Approach for Relational Database Applications	Teste Estrutural de Software: Uma Abordagem para Aplicações de Banco de Dados Relacional
29	FCCE: A Testing Criteria Family for the Validation of Systems Specified in Estelle	FCCE: Uma Família de Critérios de Teste para Validação de Sistemas Especificados em Estelle
32	Tests and Code Generation for Web Systems	Testes e Geração de Código de Sistemas Web
33	A Method for Functional Testing for the Verification of Components	Um Método de Teste Funcional para Verificação de Componentes
35	Unit Testing of Aspect-Oriented Programs	Teste de Unidade de Programas Orientados a Aspectos
36	Reuse in the Software Testing Activity to Reduce VV&T Cost and Effort in the Development and Re-engineering of Software	Reuso na atividade de teste para reduzir custo e esforço de VV&T no desenvolvimento e na reengenharia de software
37	A Methodology for the Verification of Partial Systems Modeled with Object-Based Graph Grammar	Uma metodologia para a verificação de sistemas parciais modelados na gramática de grafos baseada em objetos
38	Distributed Environment of Communication Fault Injection for Testing of Network Java Applications	Ambiente Distribuído de Injeção de Falhas de Comunicação para Teste de Aplicações Java de Rede
40	An Aspect-based Tool for Functional Testing of Java Programs	Uma ferramenta baseada em aspectos para o teste funcional de programas Java
41	Automatic Generation of Test Drivers and Stubs for JUnit based on U2TP Specifications	Geração Automatizada de Drivers e Stubs de Teste para JUnit a Partir de Especificações U2TP
47	Using Similarity Functions to Reduce Test Suites in Strategies for Model-based Testing	Usando Funções de Similaridade para Redução de Conjuntos de Casos de Teste em Estratégias de Teste Baseado em Modelos
48	Generation of Faultloads for Testing Campaigns with Fault Injection from UML Testing Models	Geração de cargas de falha para campanhas de teste com injeção de falhas a partir de modelos UML de teste
51	A Catalog of Stubs to Support the Integration Testing of Aspect-Oriented Programs	Um catálogo de stubs para apoiar o teste de integração de programa orientados a aspectos
55	Mutation Testing in Procedural and Object-Oriented Paradigms: An Evaluation of Data Structure Programs	Teste de Mutação nos paradigmas Procedimental e OO: Uma avaliação no contexto de estrutura de dados