The Journal of Systems and Software 86 (2013) 934–950

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A scoping study on the 25 years of research into software testing in Brazil and an outlook on the future of the area

Vinicius Humberto Serapilha Durelli[a], Rodrigo Fraxino Araujo[a,b], Marco Aurelio Graciotto Silva[a,c], Rafael Alves Paes de Oliveira[a], Jose Carlos Maldonado[a], Marcio Eduardo Delamaro[a,*]

[a] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 13560-970 São Carlos, SP, Brazil
[b] Linux Technology Center, IBM Brazil, Hortolandia, Brazil
[c] Federal Institute of Education, Science and Technology of São Paulo, 13872-550 São João da Boa Vista, SP, Brazil

ARTICLE INFO

Article history:
Received 20 January 2012
Received in revised form 1 September 2012
Accepted 9 October 2012
Available online 18 October 2012

Keywords:
Software testing
Systematic mapping
Brazilian research

ABSTRACT

Over the past 25 years the Brazilian Symposium on Software Engineering (SBES) has evolved to become the most important event on software engineering in Brazil. Throughout these years, SBES has gathered a large body of studies in software testing. Aimed at providing an insightful understanding of what has already been published in such event, we have synthesized its 25-year history of research on software testing. Using information drawn from this overview we highlighted which software testing topics have been the most extensively surveyed in SBES literature. We have also devised a co-authorship network to depict the most prolific research groups and researchers. Moreover, by performing a citation analysis of the selected studies we have tried to ascertain the importance of SBES in a wider scenario. Finally, using the information extracted from the studies, we have shed light on the state-of-the-art of software testing in Brazil and provided an outlook on its foreseeable future.

1. Introduction

Throughout the past two and a half decades, SBES (from the acronym, in Portuguese, for Brazilian Symposium on Software Engineering) has established itself as an authoritative venue for software engineering (SE) research. Being the premier Brazilian conference on the topic, it has drawn the attention of researchers, academics, and practitioners alike, accumulating a large body of literature on almost every SE topic. Such comprehensive literature has featured technical papers on innovative research.

Given that software testing is an essential part of the SE literature, a significant part of SBES's 25-year publications addresses this subject. As far as we know the literature, before the previous edition of the conference, which held a 25th-anniversary special track, there were no in-depth studies focusing on giving an overview of what has been published in Brazil on software testing. Such special track received a number of contributions describing retrospective and prospective viewpoints on various SE topics. Several of these contributions indicated that software testing is one of the SE areas that account for the most papers appearing in SBES (Cavalcanti and Silva, 2011; Gomes et al., 2011a). Furthermore, acknowledging the importance of software testing among other SE topics, most contributions were centered on the evolution of this research area in Brazil and its worldwide impact. However, these software testing contributions investigated it from a narrow scope: Delamaro et al. (2011) focused solely on providing an overview of what had been published on structural and mutation testing, Lemos et al. (2011) reported on the types of empirical evaluations employed in the software testing-related studies, and Nakagawa and Maldonado (2011) concentrated on the architectural aspects of software testing tools.

Another downside of these previous studies is that none of them considered the proceedings from the tools session. We believe that such part of the conference has also brought significant contributions, gathering papers that report on advances in testing automation and widely used software testing tools. Two prominent examples of tools are (i) Proteum (Delamaro et al., 1993), which, according to Delamaro et al. (2011), has been extensively used by researchers throughout the world and (ii) JaBUTi (Java Bytecode Understanding and Testing) (Vincenzi et al., 2003), which has been used for structural testing research and spawned several derivatives.

In this sense, our previous work (Durelli et al., 2011) is a more comprehensive scoping study because it encompasses both the main track and the tools session proceedings. In fact, the underlying survey was carried out as a systematic mapping study. Systematic mapping is a synthesis method that involves searching into the literature to ascertain the nature, extent, and quantity of published research papers (i.e., primary studies) on a particular area

* Corresponding author. Tel.: +55 16 33738628.
E-mail address: delamaro@icmc.usp.br (M.E. Delamaro).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.10.012

of interest (Petersen et al., 2008). Mapping studies aggregate and categorize primary studies, yielding a synthesized view of a certain research area. Moreover, it differs from informal literature reviews because the approach used for searching is defined in a protocol and reported as an outcome. Such characteristics contribute towards the transparency and replicability of the mapping studies.

This paper is a revised and extended version of our previous study (Durelli et al., 2011) which outlines the results of the mapping study undertaken to classify and categorize evidence on software testing in the context of SBES. It provides a comprehensive look at the software testing research in Brazil. The primary contributions are the identification of the areas of software testing research most subjected to investigation, those that have not received much attention in these past 25 years, the analysis of the main contributions of Brazilian researchers on software testing, and the use of our findings to point out the areas on which the software testing research community needs to focus to meet the demands imposed by the types of systems that are being built today and will be built in the future. Aimed at obtaining an overall idea of the most prolific research groups and the degree of collaboration among authors of different universities, we have drawn authorship information from the selected papers to devise a co-authorship network. To investigate the external impact of SBES software testing literature, we analyzed the impact factor of the selected studies in several electronic databases. Also, in order to position this literature within an international scenario, we described surveys similar to our own. In doing so, we highlight whether these surveys mention Brazilian research, e.g., extended versions of papers published in SBES and tools developed by Brazilian researchers.

The remainder of this paper is organized as follows. Section 2 describes our mapping study protocol. Sections 3 and 4 describe how we selected and classified primary studies: Section 3 highlights the classification according to study type and Section 4 the classification according to contribution for software testing. Section 5 presents further details on how the authorship information drawn from the selected papers was used to devise a co-authorship network that depicts prominent research groups and their respective roles in SBES history. Section 6 estimates the relevance of SBES as a 25-year old venue by measuring the dissemination and impact of its software testing efforts. The collected information sets a baseline to compare SBES research against a broader scenario (Section 7) and topics on software testing research to be addressed in the near future (Section 8). Threats to validity are summarized in Section 9. Section 10 describes related software testing surveys, and we summarize and discuss our contributions in Section 11.

2. Systematic mapping process

Mapping studies follow a fivefold process (Petersen et al., 2008): (i) definition of research questions, (ii) performing the search for primary studies, (iii) screening of papers, (iv) keywording of abstracts, and (v) data extraction and mapping. In what follows we briefly describe how each of these steps was performed.

The research questions must embody the mapping study purpose. Since we aimed at determining the software testing topics that have been investigated in SBES and the productivity of researchers, our three research questions reflect this purpose as follows:

RQ_1: what test techniques have been most investigated?
RQ_2: who are the most prolific researchers?
RQ_3: what types of studies have been published?

The search for primary studies involved examining all previous SBES and its tools session proceedings. The examined proceedings are of one of two categories: hard-copy and soft-copy. SBES proceedings, for instance, were issued only as hard copies up to 1999. From 1999 on, these documents were issued as hard and soft copies (except in 2000 and 2003, when they were available only as hard copies). From 1999 to 2009, most soft copies (1999, 2001–2002, 2004, and 2006–2009) are available at BDBComp (Brazilian Digital Library on Computing). From 2009 onwards, main track proceedings have been indexed by IEEE Digital Library. From 1987 to 1991, the tools session proceedings were included in the main track. From 1992 onwards, they have been issued in their own proceedings. Notable exceptions are the 1994–1995, 1997, and 2000–2002 proceedings, which were again embodied in SBES main track proceedings.

Due to the heterogeneous nature of the proceedings, we could not use a unified search method to gather all studies. Furthermore, the distinct nature of the sources of studies precluded us from devising a search string during the conduction of this step. Rather, we managed to gather all SBES and tool session proceedings by (i) downloading soft copies from digital libraries, (ii) requesting soft and hard copies from fellow researchers, and (iii) borrowing hard copies from local libraries.

The third step, screening, aimed at determining the primary studies relevant to answer our research questions. To this end, we examined all proceedings, applying a set of inclusion and exclusion criteria to each study. We came up and applied the following inclusion and exclusion criteria:

- **Inclusion criterion:**
  - Any paper that described one or more studies regarding software testing was subjected to be included.
- **Exclusion criteria:**
  - Papers whose studies are not related to software testing (e.g., papers describing research on any of the other SE topics) were excluded.
  - Papers that report on insightful proposals for prospective software testing research, but do not apply them to define a test-related technology were excluded.
  - Elements of "gray literature", posters, tutorials, and panels were not included in our analysis.

From the quality assessment standpoint, the only requirement for a paper to be selected was that it had to have been published. During the study selection, we considered sufficient the peer review process the studies were subjected to. We avoided imposing many restrictions on primary study selection given that a broad overview of what had been published on software testing was intended.

The inclusion and exclusion criteria were applied using the following procedure. Each author chose a subset of proceedings to read. At first, based solely on title and abstract, we applied the previously established criteria to each paper. Afterwards, primary studies deemed as relevant according to our criteria were read through. As we read each study, we also performed a preliminary analysis and identified their main contributions. Eventually, we proceeded to do a consensus process to verify whether the pending papers met our criteria.

According to the information we gathered, throughout SBES history (from 1987 to 2011), 2659 papers were submitted, 865 of these papers were accepted. We ended up with 123 candidate papers. However, after a close investigation, the initial set was reduced to 104. Fig. 1 shows the frequency of primary studies we selected per year. According to the graph, research on software testing has been quite constant throughout the underlying 25-year period, except for the first years. Considering the selectivity of the conference and its broad coverage, it is fair to assume that software testing research has been playing a substantial role in SBES.

After selecting the papers to be analyzed, we performed the data extraction and analysis phases of the systematic review. First, we

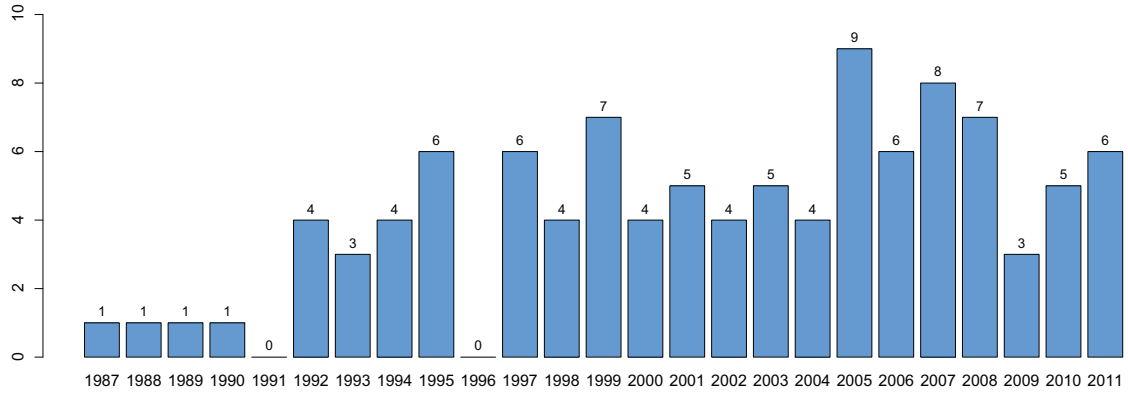


Fig. 1. Year-wise distribution of software testing primary studies.

defined the data to be extracted based on taxonomies for RQ_1 and RQ_3 or research evaluation metrics with regard to RQ_2. We analyzed and evaluated the corresponding research questions before assessing the overall impact of Brazilian research on software testing (Section 7). Each of the aforementioned research questions is addressed in the next sections.

3. Classification according to the study type

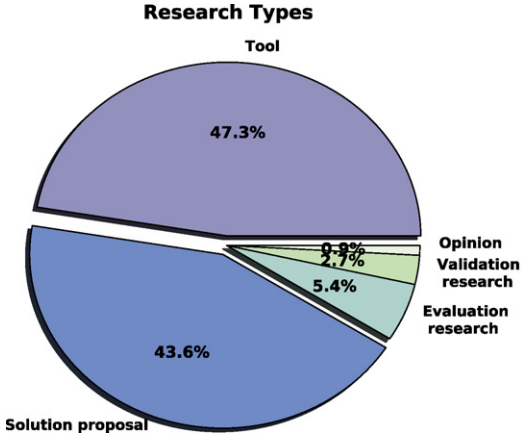
As previously stated, one of our research questions is concerned with categorizing SBES literature on software testing. Since formulating classification schemes to be used in the evaluation of papers is a controversial topic, rather than defining our own scheme, we decided to base our categorization on a scheme proposed by Wieringa et al. (2005). According to the authors, research can be classified into five categories: (i) validation research, (ii) evaluation research, (iii) solution proposals, (iv) opinion papers, and (v) experience papers. We could classify the analysed studies into four of these categories:

- **Solution proposal:** studies that report on a solution technique and argue for its usefulness, effectiveness, and relevance. The described solution technique is either novel or an extension of an existing technique. Studies in this category do not usually present in-depth validation of the described solution technique, but tend to describe a proof-of-concept by means of an example, a running prototype, or even a sound argument.
- **Evaluation research:** studies focusing on the evaluation of a problem or an implemented solution in practice or real settings. They include case studies, controlled experiments, etc.
- **Validation research:** studies that investigate proposed solutions which have not been implemented in practice. Such investigations are performed systematically by means of experiments, prototyping, etc.
- **Opinion:** also known as position papers, these studies contain the authors' point of view. In most cases, their claims are not supported by experimental evidence.

As the primary studies selected from the tools session proceedings did not fit well into the aforementioned categories, the classification scheme had to be extended. An additional research type category was devised to properly classify those primary studies:

- **Tool:** studies whose main contribution is a testing tool; often in the form of a research prototype, which automates one or several testing activities.

Fig. 2 shows the frequency of primary studies by research type. It is worth mentioning that some primary studies were assigned to more than one category, thereby affecting the frequency count. The sum of the frequencies shown in Fig. 2 (110) is larger than the total of selected studies (104). By analyzing the gathered data, it is evident that tool and solution proposal are by far the research type of most selected primary studies. If we take into consideration only the main track from SBES, the answer to RQ_3 is that most software testing studies are solution proposals. However, considering both SBES and tools session proceedings, it can be concluded that considerable research has been done on software testing automation: 52 out of the 104 primary studies describe software testing tools. This



Research Type	Frequency	Percentage
Tool	52	47.3%
Solution proposal	48	43.6%
Evaluation research	6	5.4%
Validation research	3	2.7%
Opinion	1	0.9%

Fig. 2. Distribution of primary studies by research type.

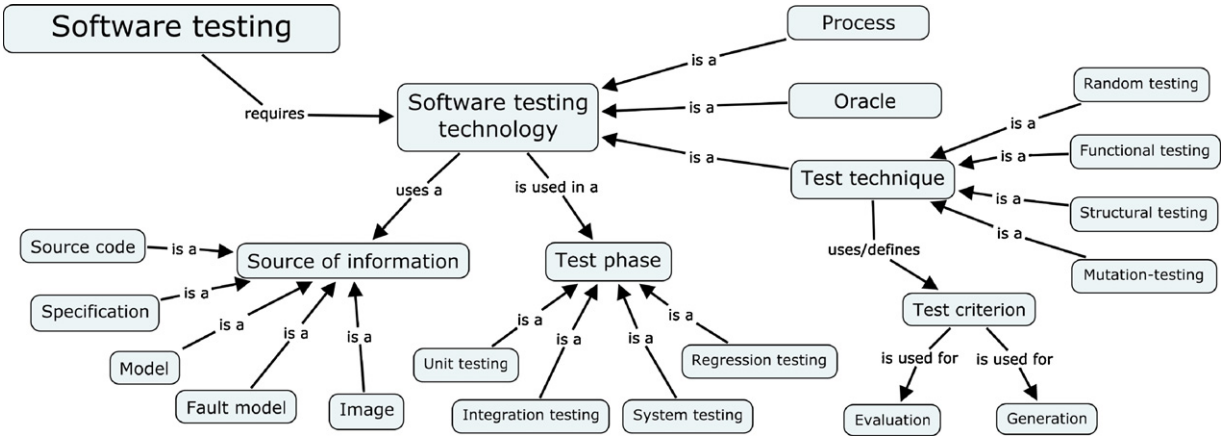


Fig. 3. Concept map used for the classification of the selected papers.

supports the notion that software testing requires implementing tools to support and evaluate the proposed solutions.

The gathered data also quantitatively characterize experiments published in SBES. As seen in Fig. 2, there has been a lack of experimental studies. From the final set of 104 primary studies, only 9 report on formal experiments investigating the benefits of the presented techniques. These 9 studies were further classified into two categories; 6 were included in the evaluation research category and 3 in the validation research category. Nevertheless, from observing the evolution of the type of papers published in SBES in the last five years, we believe that there is a tendency towards an increase in the number of papers combining characteristics of a solution proposal and either evaluation or validation research. An indicator of such tendency is the ever-increasing adoption of the evidence-based paradigm within software engineering research.

Only one primary study was classified as opinion paper. In fact, the latest call for papers of SBES technical research track does not even mention such types of contribution. However, distinct researchers are often invited as speakers, thereby somehow filling this gap with opinion panels.

4. Classification regarding software testing

Aiming to answer RQ_1, we classified each primary study contribution according to three characteristics: technology, source of information, and test phase. Prior to the selection phase, we established a classification scheme for the evidences to be extracted from the papers. Drawing from earlier, similar studies (Vegas et al., 2009; Bertolino, 2007), we defined a concept map of the main concepts in software testing, as shown in Fig. 3. As mentioned, the main categories are testing technology, source of information, and testing phase, following the rationale that each technology employed in testing requires a source of information and is applied during one or more testing phases. Technology is further classified into three subcategories: testing technique, oracle, and process. Testing techniques can be further classified as random, functional, structural or mutation-based testing. We also considered whether it was used for evaluation or generation of test cases.

Considering this concept map, we classified in Table 1 the evidences extracted from the selected papers. Analyzing its data, it is evident that structural and mutation testing have been widely used for evaluation of test cases. This reflects upon the choice of sources of information and testing phases, as structural testing techniques usually employ source code or models to derive test requirements, while mutation testing requires fault models; usually in the unit and integration phases. Research in techniques for generation of test cases is significant, but still modest when compared with its counterparts. It can also be observed that random and functional testing are the least studied areas at SBES, as well as system and regression testing.

Fig. 4 shows that structural criteria have been the most investigated in SBES. We argue that the possible reasons are the following: (i) up to half of our primary studies are based on source code and (ii) it is the most consolidated technique within academic circles. Nonetheless, mutation testing has been steadily expanding its presence since 1993 and, recently, functional testing has been drawing considerable attention. These

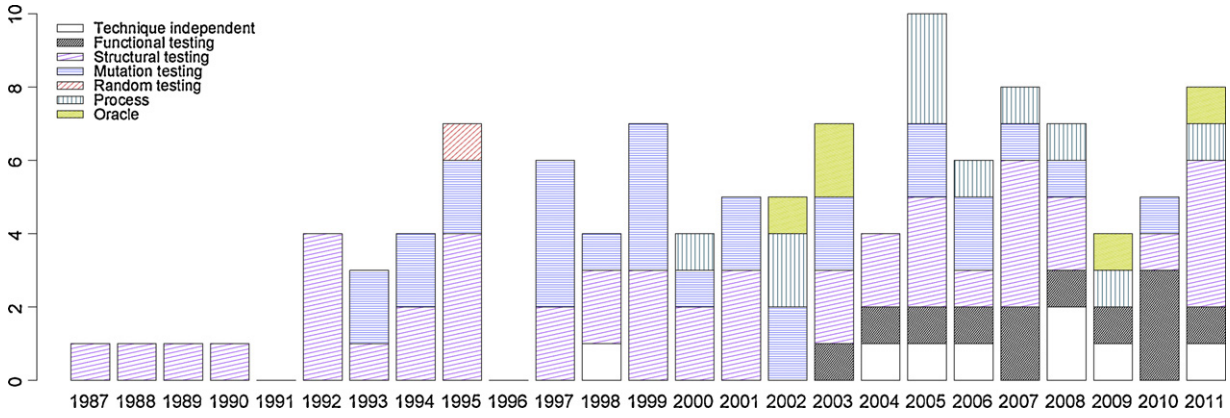


Fig. 4. Software testing technology per year.

Table 1
Classification of papers considering the concept map shown in Fig. 3.

Testing technology	Papers
Process	Herbert and Price (2000), dos Santos Domingues et al. (2002), Aranha and Borba (2002), Neto et al. (2005), Ré and Masiero (2005), Duarte et al. (2005), Neto and Travassos (2006), Aranha et al. (2008), Venâncio (ao et al., 2009), Gomes et al. (2011b)
Oracle	Simão et al. (2002), Oliveira et al. (2003), Farias and Machado (2003), de Oliveira et al. (2009), Souza et al. (2011a)

Testing technique	Evaluation	Generation
Random testing	Martins (1995)	Martins (1995)
Functional testing	Farias and Machado (2003), Barbosa et al. (2004), Rocha et al. (2005), Coelho et al. (2007), Silva et al. (2007), Souza et al. (2011a)	Barbosa et al. (2004), Silva et al. (2006), Araujo and Delamaro (2008), Besson et al. (2009), Albuquerque et al. (2010), Ferreira et al. (2010), de Sousa Santos et al. (2010)
Structural testing	Price et al. (1987, 1992), Maldonado and Chaim (1988), Chaim et al. (1989, 1998), de Alencar Price and Zorzo (1990), Júnior et al. (1992), Vergilio et al. (1992, 1994, 1995), Maldonado et al. (1992), Silva and Price (1994), Nakazato et al. (1995), Borges et al. (1995), Herbert and de Alencar Price (1995), Crespo et al. (1997), Vilela et al. (1998), Li et al. (1999), Spoto et al. (2000), Souza et al. (2001), Candolo et al. (2001), ao Alexandre Góes and Renaux (2001), Vincenzi et al. (2003), da Silva Simão et al. (2003, 2007), Nunes and de Melo (2004), Lemos et al. (2004), Nardi et al. (2005), Hausen et al. (2006), Franchin et al. (2007), Delamaro et al. (2007), Lemos and Masiero (2008), Eler et al. (2010), Cafeo and Masiero (2011), Neves and Masiero (2011), Souza et al. (2011a)	Vergilio et al. (1993, 1995), Nakazato et al. (1995), Herbert and de Alencar Price (1997), Bueno and Jino (1999), da Silva Simão et al. (2003, 2007), Abreu et al. (2007, 2005), Farias and Machado (2003), Nunes and de Melo (2004), Simão et al. (2005), Cartaxo et al. (2008)
Mutation testing	Mathur and Wong (1993), Delamaro et al. (1993, 1997), Fabbri et al. (1994a, 1995a, 1997), Wong et al. (1994), Martins (1995), Delamaro and Maldonado (1997), Barbosa et al. (1998), Nakagawa and Maldonado (1999), Sugeta et al. (1999), Vincenzi et al. (1999), de Carvalho et al. (1999), Simão and Maldonado (2000a,b), da Silva Simão and Maldonado (2001), Delamaro et al. (2001a), Simão et al. (2002), Emer and Vergilio (2002), Yano et al. (2003), Gerchman et al. (2005, 2008), Simão et al. (2005), Campanha et al. (2010)	do Rocio Senger de Souza and Maldonado (1997), Vacaro and Weber (2006), Mendes et al. (2006), Cruz Filho and Vergilio (2007)
Source of information		
Source code	Price et al. (1987, 1992), Maldonado and Chaim (1988), Chaim et al. (1989, 1998), de Alencar Price and Zorzo (1990), Júnior et al. (1992), Vergilio et al. (1992, 1993, 1994, 1995), Maldonado et al. (1992), Silva and Price (1994), Borges et al. (1995), Herbert and de Alencar Price (1995, 1997), Crespo et al. (1997), Vilela et al. (1998), Pinto and de Alencar Price (1998), Bueno and Jino (1999), Spoto et al. (2000), ao Alexandre Góes and Renaux (2001, 2001), Vincenzi et al. (2003), Farias and Machado (2003), Nunes and de Melo (2004), Lemos et al. (2004), Abreu et al. (2005), Ré and Masiero (2005, 2007), Nardi et al. (2005), Hausen et al. (2006), Biasi and Becker (2006), Mendes et al. (2006), Coelho et al. (2007), Franchin et al. (2007), Delamaro et al. (2007), Dantas et al. (2008), Lemos and Masiero (2008), Ré et al. (2008), Campanha et al. (2010), Eler et al. (2010), de Sousa Santos et al. (2010), Bernardo et al. (2011), Cafeo and Masiero (2011), Macedo et al. (2011), Neves and Masiero (2011), Souza et al. (2011a)	
Specification	da Silva Simão et al. (2003), Farias and Machado (2003), Rocha et al. (2005), Aranha et al. (2008), Besson et al. (2009), Venâncio (ao et al., 2009), Ferreira et al. (2010)	
Model	Fabbri et al. (1994a, 1995a, 1997), Nakazato et al. (1995), Martins (1995), Sugeta et al. (1999), Li et al. (1999), de Carvalho et al. (1999), Simão and Maldonado (2000a,b), Souza et al. (2001), Candolo et al. (2001), Oliveira et al. (2003), Fidelis and Martins (2004), Barbosa et al. (2004), Neto et al. (2005), Simão et al. (2005), Silva et al. (2006, 2007), Biasi and Becker (2006), da Silva Simão et al. (2007), Cruz Filho and Vergilio (2007), Araujo and Delamaro (2008), Cartaxo et al. (2008), Gerchman et al. (2008), Albuquerque et al. (2010), Gomes et al. (2011b)	
Fault model	Mathur and Wong (1993), Delamaro et al. (1993, 1997, 2001a), Wong et al. (1994), do Rocio Senger de Souza and Maldonado (1997), Delamaro and Maldonado (1997), Barbosa et al. (1998), Nakagawa and Maldonado (1999), Vincenzi et al. (1999), da Silva Simão and Maldonado (2001), Simão et al. (2002), Emer and Vergilio (2002), Yano et al. (2003), Gerchman et al. (2005), Vacaro and Weber (2006), Cruz Filho and Vergilio (2007)	
Image	de Oliveira et al. (2009)	
Testing phase		
Unit testing	Price et al. (1987, 1992), Maldonado and Chaim (1988), Chaim et al. (1989, 1998), de Alencar Price and Zorzo (1990), Júnior et al. (1992), Vergilio et al. (1992, 1993, 1995), Maldonado et al. (1992), Mathur and Wong (1993), Delamaro et al. (1993), Fabbri et al. (1994a, 1995a, 1997), Wong et al. (1994), Silva and Price (1994), Nakazato et al. (1995), Borges et al. (1995), Herbert and de Alencar Price (1995, 1997), do Rocio Senger de Souza and Maldonado (1997), Crespo et al. (1997), Vilela et al. (1998), Pinto and de Alencar Price (1998), Nakagawa and Maldonado (1999), Sugeta et al. (1999), Li et al. (1999), Bueno and Jino (1999), de Carvalho et al. (1999), Simão and Maldonado (2000a,b), Spoto et al. (2000), Souza et al. (2001, 2011a), Candolo et al. (2001), ao Alexandre Góes and Renaux (2001), Simão et al. (2002), Emer and Vergilio (2002), Vincenzi et al. (2003), Yano et al. (2003), Nunes and de Melo (2004), Lemos et al. (2004), de Castro Guerra et al. (2005), Abreu et al. (2005, 2007), Nardi et al. (2005), Rocha et al. (2005), Hausen et al. (2006), Biasi and Becker (2006), Coelho et al. (2007), Franchin et al. (2007), Delamaro et al. (2007), Araujo and Delamaro (2008), Cartaxo et al. (2008), Dantas et al. (2008), Lemos and Masiero (2008), Campanha et al. (2010), Eler et al. (2010), Bernardo et al. (2011), Macedo et al. (2011)	
Integration testing	Price et al. (1987), de Alencar Price and Zorzo (1990), Vergilio et al. (1994), Herbert and de Alencar Price (1995), Delamaro et al. (1997, 2001a), Delamaro and Maldonado (1997), Vilela et al. (1998), Pinto and de Alencar Price (1998), Vincenzi et al. (1999), Spoto et al. (2000), Barbosa et al. (2004), Nunes and de Melo (2004), Neto et al. (2005), Ré and Masiero (2005, 2007), Nardi et al. (2005), Hausen et al. (2006), Mendes et al. (2006), Abreu et al. (2007), Franchin et al. (2007), Cruz Filho and Vergilio (2007), Gerchman et al. (2008), Ré et al. (2008), Cafeo and Masiero (2011), Macedo et al. (2011), Neves and Masiero (2011)	
System testing	Aranha and Borba (2002), Farias and Machado (2003), Fidelis and Martins (2004), Gerchman et al. (2005), Silva et al. (2006, 2007), Vacaro and Weber (2006), Aranha et al. (2008), Besson et al. (2009), Albuquerque et al. (2010), Ferreira et al. (2010), de Sousa Santos et al. (2010), Gomes et al. (2011b)	
Regression testing	Fidelis and Martins (2004), Silva et al. (2006), Aranha et al. (2008)	

tendencies, as well as evidences extracted from the selected studies regarding their contributions to oracle, process, source of information and testing phase, are described in the following sections.

4.1. Source of information

Our analysis shows that software testing research has explored four distinct information sources: (i) source code, which is responsible for 49.1% of our primary studies, (ii) models, whose related studies account for 27.5%, (iii) fault-based constitutes 17.3%, (iv) software specification account for 5.1%, and (v) digital images make up 1% of all the selected papers. In the following, aimed at elucidating each of the subcategories, we describe several studies that fall into each subcategory.

An important paper that explores source code information in order to extract test requirements is the one by Delamaro et al. (2007). Broadly speaking, their paper is concerned with implementing structural testing by drawing information from an intermediate source code representation, i.e., Java bytecodes. The authors present techniques for analyzing bytecodes and implementing structural testing within two domains: relational database applications and aspect-oriented programs. The underlying techniques rely on JaBUTi (Vincenzi et al., 2003) coverage tool, which was originally designed to extract intra-method control-flow and data-flow testing requirements. Apart from Delamaro et al. (2007), several studies have extended JaBUTi (Franchin et al., 2007; Lemos and Masiero, 2008). Lemos and Masiero (2008), for instance, modified JaBUTi to deal with the new constructions (e.g., pointcuts) introduced by aspect-oriented programming. More specifically, they devised a structural integration testing approach for AspectJ and extended JaBUTi to provide automated support for their approach: the new version of the tool is called JaBUTi/PC-AJ. Since Lemos's and Masiero's study draws information from AspectJ constructions, which is a source code representation, it was also grouped into the source code subcategory.

As an example of study concerned with exploring high-level models to devise test requirements we can mention the study by da Silva Simão et al. (2007). The authors describe an experiment in which they evaluate some widely used coverage criteria for finite state machines (FSM). In a model subcategory, we can mention the study by Biasi and Becker (2006). Rather than FSM, the representation from which they derive information is Unified Modeling Language (UML) 2.0 Test Profile (U2TP). In fact, since U2TP can be used to specify a set of test cases, the authors extract such meta-data from these models and automate the generation of test drivers and stubs for the unit testing framework JUnit.

As for studies aimed at evaluating fault-based approaches, most of the studies in this subcategory have to do with research into mutation analysis. An example of primary study that was grouped into this subcategory is the one by Delamaro et al. (2001a). In their study, they define a set of mutation operators tailored towards Java concurrent programs. As reported by the authors, mutation operators for other programming languages had already been defined, however, due to the differences between the synchronization models used by the other concurrent programming languages and the one employed by Java, new operators were required to properly address concurrency in Java. Also in this category, the study of Fabbri et al. (1997) is concerned with the investigation of how effectively mutation analysis can be used to test statechart-based specifications. As a fundamental part of the proposed approach, the authors devised a mutation operator set that reflects the intrinsic characteristics of statecharts.

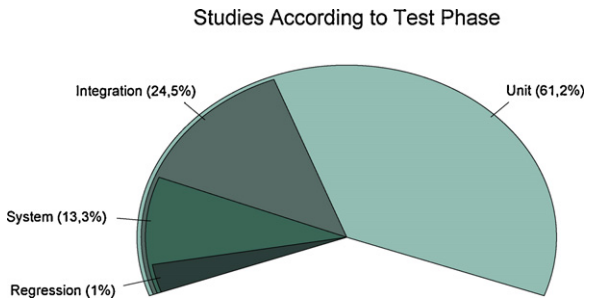


Fig. 5. A fan plot of the distribution of studies according to testing phase. It can be seen that unit testing is the largest slice and that the integration testing slice is roughly one third as large.

4.2. Testing phase

We classified the selected studies according to four distinct phases. Unit testing, through which individual units (methods, procedures, or functions) are tested. Integration testing, whose purpose is to verify the structure and behavior of subsystems by checking if each of them satisfies its corresponding function in the overall architecture. System testing, which aimed at determining whether all subsystems work as a whole and meet the underlying specification. Regression testing, which is performed whenever an element of the system under test (SUT) is modified.

The fan plot in Fig. 5 illustrates the relative quantities and differences between primary studies pertaining to each testing phase. At a glance one can notice that unit testing has been the testing phase most covered in SBES studies, whereas regression testing has been the least investigated one. Among the selected studies, 59 (which account for 61.2%) report on approaches tailored to unit testing. It is by far the testing phase that has been drawn most attention in SBES research since its beginning. Additionally, some of the most prominent SBES studies focus on unit testing. Mathur and Wong (1993) detail a research that intends to reduce the cost of mutation testing by randomly selecting the type of mutation or by using constrained mutation. Ten years later, Vincenzi et al. (2003) published a paper describing a coverage testing tool designed to test Java programs and Java-based components.

The first studies on integration testing were published in the earlier editions of the event (Price et al., 1987; de Alencar Price and Zorzo, 1990). Nevertheless, the first research addressing the issue more rigorously was published in 1994 (Vergilio et al., 1994). The authors explore infeasible paths to be covered, as a means to prevent or determine them when possible. Overall, SBES studies dealing with integration testing account for 24.5% so far. Summing up, 26 studies deal with this topic, from which 10 deal solely with integration testing and 16 studies also cover unit testing.

In SBES history, the necessity of specific techniques for testing a system and its structure as a whole only started to be addressed in 2002. Aranha and Borba (2002) present a testing process that uses code generators to improve the development productivity by focusing on performance and acceptance tests. Throughout SBES history, there have been 11 (which account for approximately 13.3%) studies on system-level testing.

Only 1.0% (3 studies) of the selected papers dealt with regression testing (Fidelis and Martins, 2004; Silva et al., 2006; Aranha et al., 2008). These studies describe testing tools tailored to automate the proposed regression testing activities. The first presents a web tool, which can help the storage and re-execution of test cases; the second can assist in the automatic execution and reporting of satellite on-board computer embedded software; and the third describes a tool that supports the measurement of complexity and effort for executing applications.

4.3. Software testing technique

Software testing techniques define the foundations to the systematic detection of faults given a testing criterion. Such techniques can be classified into exhaustive, random and partition testing. Exhaustive testing entails executing the artifact under test against every possible input data and condition (Mathur, 2008). Although it is a strong technique, it is not always feasible. Some of the factors that may hinder its application are (i) input data size, (ii) combination of conditions, and (iii) undecidable computing functions. Random testing defines statistic-based criteria to model the input space and sample data from the input space randomly (Mathur, 2008). Finally, partition testing defines subsets of the input domain given a criterion. As exhaustive testing is, in most situations, unfeasible, research has been devoted mostly to random and partition testing techniques.

SBES research on software testing has been mostly devoted to partition testing techniques: functional, structural and mutation testing. From 104 primary studies, only 1 dealt with random testing, 12 reported on functional testing approaches, 46 described researches related to structural testing, mutation-related research appeared in 29 studies, and technique independent research is described in 8 studies.

The Brazilian community of software testing strongly believes that research requires not only the definition of testing criteria, but the development of tools (which makes it feasible to apply novel techniques) and empirical evaluation of the results. As a matter of fact, the main contributions in SBES in terms of software testing research are related to (i) mutation testing of procedural programs and formal specifications of reactive systems and (ii) structural testing of Java and AspectJ programs. Such studies are supported by versions of Proteum and JaBUTi.

Regarding mutation testing, the most significant contribution in terms of tools is the Proteum family, which comprises tools for specification-based and source code based testing (Maldonado et al., 2000). As for specification-based mutation, early versions of the tools are Proteum/PN (Fabbri et al., 1994a) and Proteum/FSM (Fabbri et al., 1995a, 1999a). As mentioned in Section 4.1, the main step in the development of mutation analysis for finite state machines is the definition of a mutation operator set that covers typical mistakes made by designers during the creation of FSMs (e.g., missing transition) and minimal test requirements, such as covering all transitions (Fabbri et al., 1994b). Proteum/FSM applies the operators against specifications, creating mutated versions that are then evaluated by a test set. As a result, it yields a mutation score based on the relation of successful and failed test cases (disregarding equivalent mutants). Validation studies regarding mutation criteria for FSM were developed by de Carvalho et al. (1999), establishing essential operators for cost-effective constraint-mutation testing of applications.

FSMs are the basis for several formal specification techniques. For instance, Proteum/ST (Sugeta et al., 1999, 2001) implements mutation testing for statechart-based specifications by building on the operators defined by Proteum/FSM and adapting some of them to extended FSM (EFSM). In order to cope with EFSM, operators must deal with expressions and specific statechart features, such as hierarchy, history, and broadcasting (Fabbri et al., 1999b). Souza et al. (1999), by reusing FSM and EFSM operators, implemented mutation analysis for Estelle and Sugeta et al. (2004) approached SDL specifications. Instead of dealing with FSMs, Proteum/PN (Fabbri et al., 1994a, 1995b) and Proteum/CPN (Simão and Maldonado, 2000a,b) implement mutation analysis for Petri Nets specifications, using the same rationale as the other direct derivatives from Proteum.

Although Proteum/FSM has established a baseline for mutation testing employing FSM, most research contributions come from the realm of procedural programs. For instance, Proteum for C programs (Delamaro et al., 1993; Delamaro and Maldonado, 1996) defines a set of 71 operators for unit-level testing. Proteum/IM (Delamaro and Maldonado, 1997, 1996) defines mutation operators for integration-level, aiming at faults in the connections between two units: function call and parameters (Delamaro et al., 2001b). Finally, Proteum/IM 2.0 (Delamaro et al., 2000) unifies both approaches, providing unit and integration-level in a single tool. Besides implementing the aforementioned mutation testing criteria, Proteum/IM provides features that support the conduction of experimental studies, such as mutants management, execution configuration, and a script-based operation mode. These characteristics make Proteum/IM a suitable test bed for research on mutation testing (Delamaro et al., 2011).

Structural testing criteria represent another important contribution of Brazilian researchers. One of the first contributions was the definition of criteria based on the concept of potential uses, in which test requirements are defined under the assumption that a variable, defined in a given instruction block, can be used in any block where its alive Maldonado and Chaim (1988) and Delamaro et al. (2011). Testing criteria based on potential uses, such as potential-uses and potential du-paths, address an important limitation of software testing: the gap between all-uses criterion (Herman, 1976) and all paths in the presence of unfeasible paths (Frankl and Weyuker, 1988). A tool entitled POKE-TOOL was developed to automate the underlying criteria (Chaim et al., 1998). POKE-TOOL supports empirical evaluations, test data generation, debugging, and definition of reliability models.

Lessons learned in previous projects were applied to Java-based programs, establishing testing criteria at unit and integration level for object and aspect-oriented programs. JaBUTi (Vincenzi et al., 2003, 2005) is a test tool that implements control-flow and data-flow testing criteria for Java programs based upon Java bytecodes. Structural testing criteria based on control and data-flow have been implemented considering Java applications and particular features of the language, such as exceptions. Although structural testing criteria were initially devised for unit-level testing of object-oriented applications, several extensions have been created for aspect-oriented programs at unit and integration levels, providing relevant contributions to this domain (Chavez et al., 2011). Such extensions, supported by JaBUTi/AJ, address particularities regarding aspect-based applications, such as points and point cuts (Lemos et al., 2006, 2007; Lemos and Masiero, 2008), and integration testing considering pair-wise (Franchin et al., 2007; Lemos et al., 2009; Lemos and Masiero, 2011) and N-level integration (Cafeo and Masiero, 2011). A web service-based version of JaBUTi (Eler et al., 2009) is also used in education settings, integrating programming and testing evaluation using structural testing criteria (Souza et al., 2011a,b).

4.4. Oracle

It is a well-established fact that complete software testing automation entails determining whether a given outcome is correct. Thus, a testing oracle mechanism is required. A testing oracle is a method (program, function, set of data or values, etc.) used to verify whether the system under test behaves correctly on a particular execution (Baresi and Young, 2001).

Few efforts have focused on designing and implementing testing oracles. This shortage of papers that deal with testing oracles was reflected in the results of our analysis. Despite the importance of testing oracles to automate software testing, only 5 studies report research on this area. In the following, we describe these 5 studies.

Simão et al. (2002) describe a denotational semantics based oracle and a tool, `mudelgen`, that handles the language MuDeL. In this approach, a testing oracle is used to check the correctness of mutant

programs. Denotational semantics and Standard ML (SML) were used to set an environment in which the oracle can read the information on the semantics of the operators and compare them with the semantics of the generated mutants.

Oliveira et al. (2003) developed a testing oracle for Standard ML (SML) applications. Using specifications written in *Common Algebraic Specification Language* (CASL), SML-based oracles are generated. Furthermore, depending on the context in which they are inserted, the process generation can be configured, deriving testing oracles to specific domains.

Farias and Machado (2003) defined a functional testing method to verify software components using UML specifications as inputs to oracle testing. The software components behavior are defined in pre and post conditions written in Object Constraint Language (OCL), from which oracles are generated using the testing technique TOTEM (Briand and Labiche, 2001).

According to de Oliveira et al. (2009), one can choose to construct oracles defining assertions about the output presumed to be correct. Considering a GUI model, for example, it is possible to create a database that stores conventional characteristics (text or numbers) computed from the expected output (images). Then the actual output of the tested program is analyzed and the same conventional characteristics are computed and compared with the ones stored and used as references. In their approach, it is presented a mechanism for automating the testing oracle definition to systems with graphical outputs using image processing techniques.

Souza et al. (2011a) present the ProgTest: a web-based tool for the submission and automatic evaluation of programming assignments based on testing activities. Their paper outlines the scheme that they used to assess the assignments submitted by students. In general terms, to perform such an evaluation, the web system requires a model task that the authors called "oracle". This model represents the correct implementation of the activity. So, it yields an oracle able to evaluate the correction of assignments.

4.5. Process

In the software industry, settling on the proper software development process to develop a particular system may yield many competitive advantages for organizations. In general, the structure chosen for the development of a software system is set up by the process. The steps should be directly related and consistent with artifacts, people, resources, organizational structures and constraints of the system under development.

In this section we describe several studies published in SBES that report on software development processes. These papers describe processes used to implement a particular solution. The purpose of this subsection is to shed some light on the types of processes that have drawn attention of researchers whose papers were published in SBES. We also intend to identify possible processes that have been neglected by researchers. In doing so, we expect the neglected areas to receive special attention in the upcoming events.

A number of software development processes take into account software testing activities. However, a great deal of processes completely neglects testing activities. In the light of this fact, researchers have been investigating the relationship between testing activities and processes and the benefits that the former may produce. We describe a few of the papers that focus on this point of view.

A study that explores testing activities within the context of a development process is the one by Aranha and Borba (2002). They present a testing process that uses source code generators to improve the productivity of testing activities during development. Another example is the research of Neto and Travassos (2006), which describes a software development environment called Maraká. Such an environment allows for keeping track of software testing activities and the related documentation.

dos Santos Domingues et al. (2002) outline a tool called EvalTool, which implements support for the evaluation and selection of software testing tools. Broadly speaking, EvalTool explores the data obtained from tool evaluation forms, user reviews, and comparative studies, and generates reports that support decision-making activities related to the selection of testing tools. Ré and Masiero (2005) present a study that has to do with arranging classes and aspects in a way that minimizes the amount of stubs required for integration testing.

Processes are a well-established research area. Nevertheless, in the context of SBES and tools session proceedings, papers addressing processes have begun to emerge only in 2000 and currently account for only 10.5% of our selected primary studies. From observing these studies, we conjecture that what might have driven the focus to this subject is the widespread adoption of agile methodologies.

5. Co-authorship network

One of the goals of this study is the identification of the most prominent software testing researchers in SBES. Considering the 104 studies selected for analysis, 133 researchers contributed as authors, as shown in Fig. 6. The network of authors and papers clearly identifies the collaboration level among authors. The distinguished members of each group are usually at the center of each clique. This visual organization shows the main research groups in the Brazilian scenario, but it conceals quantitative information on the authors' contribution.

Aiming at identifying the most prolific researchers and answering RQ_2, we have compiled a ranking considering authors that have published at least three papers, as depicted in Table 2. Moreover, to show the relevance of SBES researchers and their underlying research in a wider scenario, we have also taken into account the authors *h*-index according to Google Scholar and Scopus. The *h*-index attempts to measure both productivity and impact of a published paper. For example, a researcher with an index of *h* has published *h* papers and each has been cited by others at least *h* times.

Despite being responsible for 93 primary studies, it is important to highlight that the studies were conducted collaboratively. Most of them ensued from the cooperation between advisors and advisees, which is the sort of collaboration that takes place in a typical Brazilian academic setting. It also turns out that once advisees establish their own research groups, most of them carried on collaborating with their former research group.

The network of authors and papers provides a bird's-eye view of the main groups and their contribution to software testing in SBES. Considering only the authors from Table 2 and their academic peers, such co-authorship network (Fig. 7) comprises 8 groups wherein most of the primary studies come from the most numerous research group (Group 2).

The first paper on software testing was published by the research group headed by Ana Price (Group 1). It was a short paper about a tool, PROTESTE (Price et al., 1987), which automates structural testing using control-flow-based criteria. Improved versions of this tool were later released, providing support for integration testing (de Alencar Price and Zorzo, 1990), data-flow criteria (Price et al., 1992), and integration testing with respect to control and data-flow criteria (Herbert and de Alencar Price, 1995). Apart from PROTESTE, her research group addressed test data generation (Herbert and de Alencar Price, 1997), testing of Smalltalk-based software (Pinto and de Alencar Price, 1998) and processes (Herbert and Price, 2000).

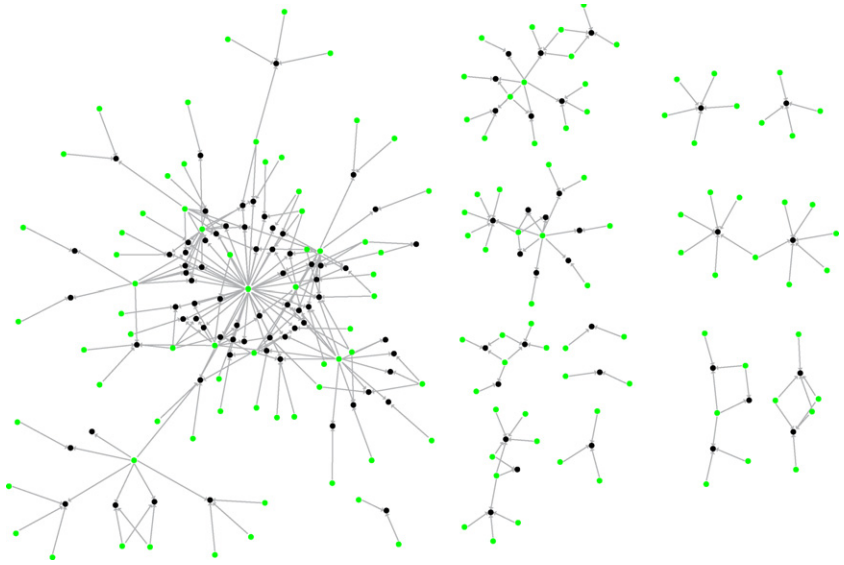


Fig. 6. Network of authors (green dots) and papers (black dots). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

In 1988, Mario Jino and José Maldonado established a new research group at the State University of Campinas (UNICAMP). Their first collaborative effort was a seminal paper whose main contribution was the potential uses criteria family (Maldonado and Chaim, 1988). It was a landmark in theoretical development for the largest research group on software testing in SBES, as it can be seen in Fig. 7 (Group 2). The group is led by Jino, Maldonado and Masiero. Its main focus is on structural and mutation testing. Regarding structural testing, the group has published studies on testing criteria based on source code for procedural languages (Chaim et al., 1989; Maldonado et al., 1992; Vergilio et al., 1992, 1993, 1994; Crespo et al., 1997; Vilela et al., 1999; Spoto et al., 2000; Hausen et al., 2006), aspect-oriented (Lemos et al., 2004; Ré and Masiero, 2005, 2007; Delamaro et al., 2007; Franchin et al., 2007; Lemos and Masiero, 2008; Ré et al., 2008), services (Eler et al., 2010), models (da Silva Simão et al., 2003, 2007), test data generation (Vergilio et al., 1995; Nakazato et al., 1995; Bueno and Jino, 1999; Candolo et al., 2001; Araujo and Delamaro, 2008) and testing tools (Júnior et al., 1992; Borges et al., 1995; Chaim et al., 1998; Vilela et al., 1998; Li et al., 1999; dos Santos Domingues et al., 2002; Vincenzi et al., 2003; Nardi et al., 2005; Simão et al., 2005). As for mutation testing, this research group focus on devising testing criteria based on source code (Delamaro et al., 1993, 1997, 2001a; Wong et al., 1994; Delamaro and Maldonado, 1997; Barbosa et al., 1998; Nakagawa and Maldonado, 1999; Vincenzi et al., 1999; da Silva Simão and Maldonado, 2001; Simão et al., 2002; Yano et al., 2003; Campanha et al., 2010) and models (Fabbri et al., 1994a, 1995a, 1997; do Rocio Senger de Souza and Maldonado, 1997; de Carvalho et al., 1999; Sugeta et al., 1999), Simão and Maldonado, 2000a,b, (da Silva Simão and Maldonado, 2001; Souza et al., 2001). It bears mentioning that two of the most well-known testing tools were designed and developed by members of Group 2, namely, JaBUTi (Vincenzi et al., 2003) and Proteum (Delamaro et al., 1993). After being made available to the general public, such tools have spawned a long list of follow-up

Table 2
Number of publications and H_{index} per author.

Rank	Name	Papers	H_i scholar	H_i scopus
1	José Carlos Maldonado	47	22	9
2	Mario Jino	16	8	3
3	Paulo César Masiero	15	15	6
4	Márcio Eduardo Delamaro	14	14	6
5	Adenilso da Silva Simão	12	5	4
6	Auri Marcelo Rizzo Vincenzi	9	10	5
7	Ana Maria de Alencar Price	8	2	1
	Sandra Camargo Pinto Ferraz Fabbri	8	9	3
	Silvia Regina Vergilio	8	8	4
8	Eliane Martins	7	10	7
	Marcos Lordello Chaim	7	3	1
9	Patricia Duarte de Lima Machado	6	8	4
	Simone do Rocio Senger de Souza	5	5	0
10	Juliana Silva Herbert	4	2	0
	Otávio Augusto Lazzarini Lemos	4	6	4
11	Edmundo Sérgio Spoto	3	2	1
	Eric W. Wong	3	22	11
	Guilherme Horta Travassos	3	17	9
	Paulo Henrique Monteiro Borba	3	20	9
	Plínio Roberto Souza Vilela	3	3	1
	Reginaldo Ré	3	2	0
	Taisy Silva Weber	3	3	2
	Wilkerson de Lucena Andrade	3	2	4

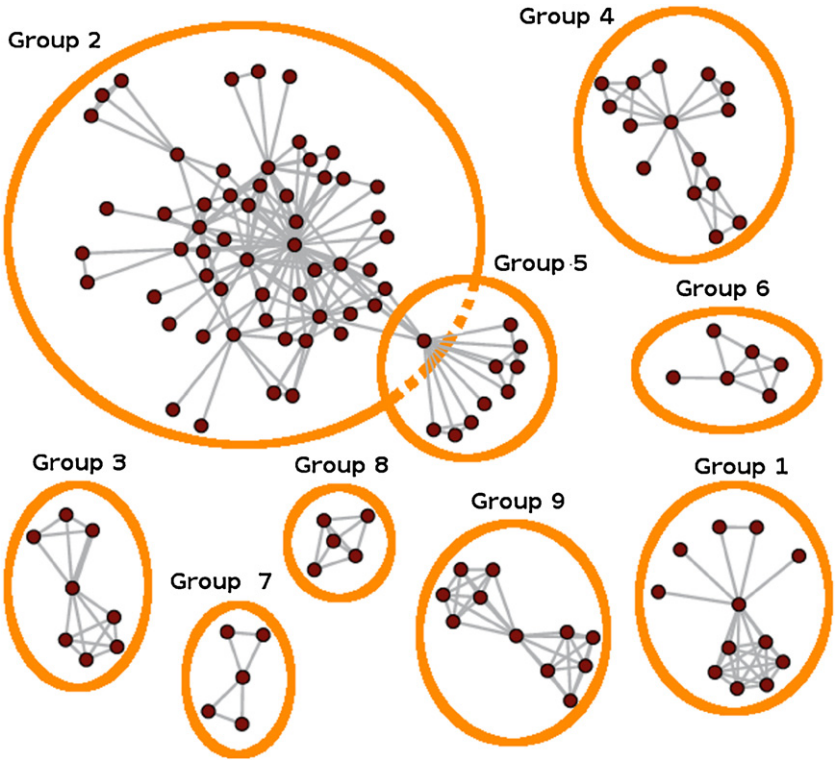


Fig. 7. Software testing research groups.

research projects that extended them to cope with different technologies (e.g., AspectJ) and criteria (e.g., mutation at the integration level).

In 2002, a third group was established and the software testing community began to spread. Led by Borba (Group 3), the first paper is concerned with testing web applications (Aranha and Borba, 2002). Later, his group worked on process estimation (Aranha et al., 2008) and model-based testing of product lines (Ferreira et al., 2010). Machado and her co-workers (Group 4) started their activity on model-based testing (Oliveira et al., 2003; Farias and Machado, 2003; Barbosa et al., 2004; Cartaxo et al., 2008) in 2003. As for testing tools, they designed and implemented a computational grid for software testing (Duarte et al., 2005).

In the following year, Eliane Martins, who had already authored some papers in SBES (Martins, 1995), established her own group (Group 5) focusing on regression testing (Fidelis and Martins, 2004), test data generation (Abreu et al., 2005, 2007) and testing of software components (de Castro Guerra et al., 2005). Weber (Group 6) and Travassos (Group 7) created their respective groups in 2005. Weber's research group has focused mostly on mutation testing using fault injection (Gerchman et al., 2005, 2008; Vacaro and Weber, 2006). Travassos and his collaborators have been tackling integration testing (Neto et al., 2005), testing within the context of software processes (Neto and Travassos, 2006), and test data generation (Albuquerque et al., 2010). Group 8 arose in 2006, led by Mattiello Francisco. Group 8 has been focusing on transferring testing technology to industry and testing of aerospace embedded systems (Silva et al., 2006, 2007). Finally, a young group driven by Bernardo et al. (2011) (Group 9) has been investigating aspect-oriented testing since 2011.

It is worth mentioning that Groups 5, 6, and 7 are headed by researchers who obtained their Ph.D. degrees in foreign institutions. Their relevance in the Brazilian scenario may indicate the potential benefit of interacting with international research groups. Furthermore, our study data also evince a strong collaboration between certain local researchers, e.g., Fabbri et al. (1997), and foreign researchers, such as Mathur and Wong (1993). Although in a small number, the presence of non-Brazilian authors corroborates the importance and outreach of the conference.

José Carlos Maldonado and Paulo César Masiero are among the most prolific researchers in terms of the number of studies on software testing published in SBES, as shown in Table 2. In addition, according to Gomes et al. (2011a), even taking into account all topics, not only software testing, they are the top two researchers with most papers published in SBES. This bears out the importance of software testing research in Brazil.

6. Citation analysis

To investigate the external impact of SBES software testing literature, we conducted a survey aiming at seeking data on the wide-ranging extent of citations of our primary studies. Our approach consisted in searching for citations related to our selected studies in several electronic databases, i.e., ACM, IEEE, Science Direct, and Google Scholar. ACM, IEEE, and Science Direct were selected because they are deemed as relevant scientific sources on software engineering. Google Scholar was also used due to its widespread coverage, holding even papers that were published in non-indexed events (which is the case of SBES's literature prior to 2009).

In order to retrieve statistical data with respect to the number of citations of each study, we used the aforementioned electronic databases, searching for occurrences of the title of each study. We found that the total amount of citations of our 98 primary studies is 276: 21 citations in ACM, 5 in IEEE, 9 in Science Direct, and 241 in Google Scholar.

Based on these numbers, we established an Impact Factor (IF) of our software testing studies. An IF is an index based on the frequency of citations of a given paper. Thus, the higher the IF of a certain paper, the more relevant the paper is. A three-year period

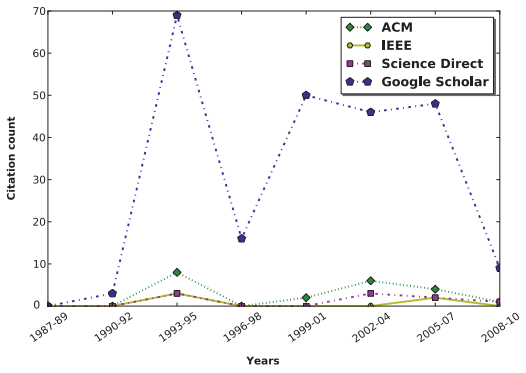


Fig. 8. Number of citations per year.

is generally used to calculate an IF. In this scenario, we calculated it by dividing the sum of studies of the previous three years regarding the evaluating date, which were cited in the same year, by the total number of software testing papers published in that period. This calculation was performed for each electronic database. Nonetheless, the results were not significant, as the IF was zero in most instances.

Our alternative was to collect the raw number of citations of each electronic database and group it by three-year ranges, as shown in Fig. 8. We can point out the papers of Wong et al. (1994) and Vincenzi et al. (2003), which received most citations. Wong et al. (1994) hold 5 citations in ACM, 3 in IEEE, 3 in Science Direct and 32 in Google Scholar. Vincenzi et al. (2003) feature 5 citations in ACM, 0 in IEEE, 2 in Science Direct and 23 in Google Scholar.

Another issue of the impact analysis of SBES, regarding other similar conferences, is that it publishes papers written in either Portuguese or English. Until 2011, 72 papers had been published in Portuguese while 32 were written in English. Although English is the *lingua franca* for scientific communication, it still lags behind Portuguese in SBES, as shown in Fig. 9. We believe that the number of studies written in English will raise now that the proceedings of the conference are published in the IEEEXplore digital library after the conference; we also believe that this will make the number of citations increase. Probably, prior to this period, the number of citations was negatively influenced by the complications concerning obtaining a published paper in hard copy or digital form.

7. An outlook on software testing in Brazil

Since the first edition of SBES, held in 1987, there have been studies on software testing. During the first six years, researchers emphasized the structural criterion. Since 1993, mutation testing has also been continuously investigated and reported in SBES. Only in 2000 results were published concerning other topics than testing criteria. In the last decade a myriad of research topics on software testing were investigated.

As technology advances, the need for improving the quality of software products also increases. Such advances in technology as well as the ever-increasing need for better testing approaches have made the following topics promising: test data generation, domain-specific approaches, and model based testing (Bertolino, 2007). We analyzed these topics in the light of our selected primary studies, aiming to indicate the areas that could be further explored in the future.

For the establishment of a testing theory, it is important to evaluate the effectiveness of a criterion. According to Harrold (2000), there is a demand for research studies that provide analytical, statistical, or empirical evidence of the effectiveness of a criterion in revealing faults. In particular, a common sense is to generally use a combination of testing techniques, even if one is deemed as more powerful, considering that they can handle different types of faults (Lyu, 1996).

SBES papers which seek empirical evidence to address evaluation and validation research were first published in 1992. Altogether they account for 9.2% of the primary studies. As far as we are concerned, this is a poor percentage. In addition, such academic empirical studies may not be representative in terms of scalability and complexity. They tend to be stronger in terms of internal validity, i.e., they provide accurate conclusions from a given data sample, but they lack support for external validity, i.e., they generalize the results to industrial contexts (Briand and Labiche, 2004). In our opinion, this trend will continue in the near future, since there is still a large gap between the cutting-edge research published in SBES and what may be practical in industrial settings.

Another challenge is to automate the generation of input data. Such area has always raised interest over the years, resulting in several studies that focus on this subject (McMinn, 2004). Nevertheless, according to Bertolino (2007), such efforts produced no significant impact on the industry until the mid-2000s.

In SBES, research in test data generation started to be addressed in 1993. Until its latest edition, 11.2% of the publications had addressed such a subject. Given that there is no constant presence of papers over the years dealing with test data generation, we consider that a greater emphasis could have been given to such area. The combination of advances in technologies such as symbolic execution, model checking, and static and dynamic analysis, coupled with progress in the standardization of models and the growth of the available computational resources, have provided positive expectations regarding the increasing automation of test data generation (Bertolino, 2007).

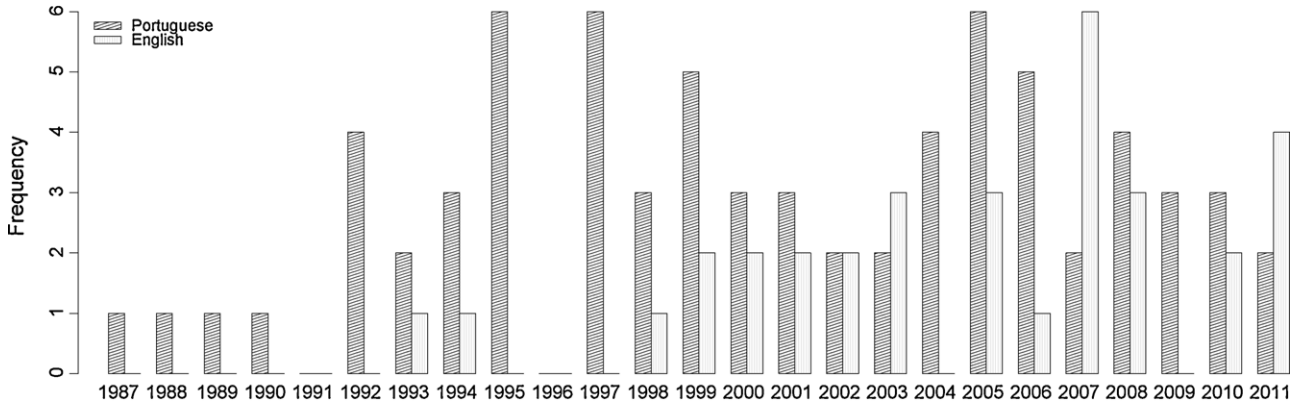


Fig. 9. Number of papers published in English and Portuguese per year.

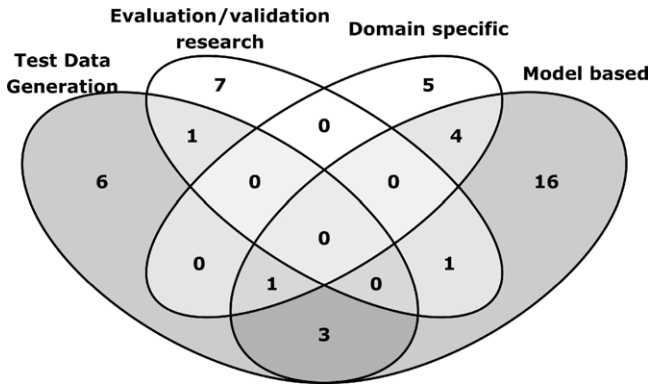


Fig. 10. Overview of published papers in SBES regarding future trend topics.

The first studies on model based testing published in SBES date from the mid-nineties. Although the idea comes from the mid-fifties by the generation of data from FSMs, the intention to apply it in real-world applications has been growing in recent years. In this approach, the testing activity takes place in a more abstract level, which can occur even before the software has been coded. This can lead to a more efficient process with a significant cost reduction and a final product with higher quality (Broy et al., 2005). Another benefit is that it can make it easier to automate the generation of test sets. SBES, which holds 25.5% of studies addressing model based testing, has contributed with its adoption. Currently, the level of industrial acceptance is not high (Bertolino, 2007). However, researchers have been placing substantial research efforts in this direction. In particular, the reactive systems industry has invested in this area by focusing on specific domains.

Domain-specific approaches have emerged as an efficient solution allowing field experts to express abstract specifications that meet their demands. The testing activity may be specific to the scope of an application. Thus, the testing task can make use of particular approaches, processes and tools, taking into consideration the domain requirements of the underlying product. Our results indicate that there has been little interest in developing software testing approaches geared towards specific domains. In SBES, the first paper was published in 2002. Altogether, 9.1% of the research are centered on such a subject.

Our results demonstrate that the research areas considered in our analysis have some overlap. A Venn diagram was drawn to better understand the distribution of these overlapping studies (Fig. 10). It is worth highlighting a paper from the tools session that overlaps 3 research areas, focusing on test data generation for embedded systems models (Araujo and Delamaro, 2008).

Although the event has already published research on most trend topics, some areas have not received much attention yet. For instance, software testing processes and testing oracles have not been sufficiently discussed in the venue yet. Agile methods and processes, which highlight the importance of software testing, are other examples of subjects that have not been covered in-depth in the venue. Although agile practices have been widely adopted, there are no studies that investigate the practices that explore the interplay between software testing and agile development, e.g., test-driven development (TDD).

The next section presents our speculations about future directions in this research area. These speculations are based upon the information we have gathered during this survey and the knowledge of the authors. When describing such speculations we also delve deeper into emergent research topics that we believe will become increasingly important as the area advances.

8. Our outlook on the future: trends that will drive software testing research

Due to advances in technology, many testing activities that defied automation in the past have become potential candidates to be automated. One of these activities is test case generation. We believe that research on such area will thrive in coming years. In particular, we can predict an increasing emphasis on automated test case generation from high-level models, e.g., use cases. Currently, there are both academic and industrial tools that automate the generation of test cases from models to varying degrees. Nevertheless, these tools are still work-in-progress and much has to be done to develop adequate techniques for solving the significant challenges posed to researchers.

As technological progress accelerates innovation, many technologies have come and gone in the last thirty years. After going through a phase of intense hype, most technologies end up falling from grace. More specifically, the most successful technologies go through what is best-known as Gartner's hype cycle (Fenn and Raskino, 2008), instead of falling into disuse. As some of these technologies have gained momentum, software testing researchers have tried to keep up with them by coming up with effective ways to test them. As far as we are concerned, the most recent example is the aspect-oriented paradigm (Kiczales et al., 1997). Over a decade ago, aspect-oriented programming was proposed in the literature as a means to achieve a better separation of concerns. As its proponents received a great deal of attention, industry-quality implementations were developed, e.g., AspectJ. Consequently, over the last few years, a myriad of approaches to properly test aspect-oriented programs have been devised.

Arguably, the aspect-oriented paradigm is past its prime (i.e., according to Gartner's hype cycle: *"peak of inflated expectations"* (Fenn and Raskino, 2008)). Thus, both researchers and practitioners have turned their attention towards emergent technologies that have generated a fair amount of buzz as cloud computing (Vaquero et al., 2008). We believe that there will be an increasing number of papers capitalizing on the popularity of emergent technologies in the foreseeable future. It can be anticipated that over the next five years researchers will be swayed by the benefits of cloud computing and significant investments will be made towards devising approaches and prototype tools to test cloud computing infrastructures. Naturally, several of these technology-driven research efforts will ultimately be published in SBES.

Embedded systems are now ubiquitous. Due to the fact that these systems are (i) embedded, (ii) depend on other systems, and (iii) do not run on devices that resemble conventional computers, they may not directly respond to external stimuli from the end user. Therefore, software testing researchers still need to come up with new approaches to scrutinize the execution of embedded systems, which poses significant research challenges. It bears mentioning that most of these efforts are currently ongoing.

We also believe that the time is ripe for taking research into testing oracles to the next level. Implementing testing oracles has been proven to be a demanding task. So far, such complexity has hampered the implementation of industry-quality testing oracles. Moreover, little research has focused on addressing this important part of software testing, and even fewer were published in SBES (as shown in Section 4). Such a shortage of research efforts makes testing oracles a promising area for those willing to take up a challenging issue. Particular attention should be paid to testing oracles for GUIs as well as reactive (i.e., event-driven) systems. We believe that a breakthrough in this topic would have enormous implications to the whole area.

In the 80s security was centered on high-end algorithms and mathematics (i.e., cryptography). By the 90s, security practitioners and researchers had shifted their focus from cryptography to

database protection. Circa early 2000s, security was mostly concerned with bullet-proofing the network. However, currently, most security vulnerabilities stem from faults in the software. Testing and security are starting to merge, which may open up possibilities for interdisciplinary research and further advancing both fields.

9. Threats to validity

We have given the same importance to tool session papers as to main track ones. That may be considered unfair and biased, as papers centered on tools are shorter. Nonetheless, the relevance and scientific rigor of several primary studies from the tools session is corroborated by the citation analysis shown in Section 6. As previously mentioned, a striking result to emerge from the data is that several primary studies on testing tools have achieved a considerable number of citations (Delamaro et al., 1993; Vincenzi et al., 2003).

Regarding the conference quality assessment perspective, after undergoing a rigorous peer review process, every paper accepted in the tools session needs to be presented at the conference. In addition, currently, one of the authors must dedicate a two-day time window to clarify any questions about the tool in question, providing more opportunities for interactions. Therefore, we argue that the inclusion of tool session studies is required to ensure that our overview encompasses all technical hurdles that have been mitigated by software testing researchers.

The current unavailability of an index for the entire SBES body of knowledge also poses a threat due to the fact that it may hamper the proper replication of our study. Furthermore, it cannot be ruled out the threat related to the subjective nature of our classification, wherein we categorized studies according to type and technology. Due to the lack of conventions for summarizing research, e.g., structured abstracts, determining the type of studies entailed some degree of subjective judgment. We tried to mitigate the subjective bias and improve validity by jointly deciding on polemical primary studies.

10. Related work

Our scoping study presents an in-depth look at the subjects that have been dealt with by the Brazilian software testing community over the past 25 years. Besides, based on our expertise and the investigated literature, we give our take on future directions in software testing research. Likewise, Bertolino (2003, 2007) also drew conclusions about the state of the art and future research thrusts in the area. Bertolino (2003) describes a general view of the area, trying to bring together all the relevant issues in a unified framework. Rather than focusing on providing a complete survey, her study emphasizes and describes an amalgam of theoretical and technical hurdles that have been keeping software testers from applying state-of-the-art approaches to solving practical software testing issues.

In her other survey, Bertolino (2007) starts off by recapping notable achievements from past and ongoing research (e.g., the systematic determination of test cases through testing criteria). In her overview of the area, these achievements are the starting point in a roadmap towards the asymptotic goals that she came up with. Along this roadmap are ongoing and emerging challenges that have been investigated by software testing researchers, so they stand for the directions the community should follow towards advancing the state of the art closer to the unattainable goals. Although both studies present broad overviews of the area, it is worth pointing out that they are somewhat biased towards Bertolino's own research and expertise. Also following a roadmap-based approach to summarizing research thrusts, similarly to Bertolino (2007), Harrold (2000) goes over several topics that according to her can lead to (i) practical testing methods, (ii) tools, and (iii) processes for fully supporting the execution of software testing activities. According to Harrold, along the underlying roadmap, the following topics play a pivotal role in pushing the state of the art forward: (i) devising better testing component-based strategies, (ii) coming up with better ways to test evolving systems, (iii) demonstrating the effectiveness of testing techniques through empirical studies, and creating effective testing processes. Moreover, aimed at making these strategies, approaches, processes, and techniques feasible for practitioners, Harrold remarks that it is necessary to bring to the fore the development of high-quality tools.

Jia and Harman (2011) directed their efforts into carrying out a survey with a narrower scope: summarizing the advances of one testing technique, namely, mutation analysis. Their survey provides an in-depth analysis of the technique, covering all important publications related to mutation testing since 1970–2009, including more than 390 papers. In addition, Master and PhD theses that have made a significant contribution to the technique in question were also taken into account during their survey. For instance, Delamaro's Master's thesis is listed among these selected documents. Furthermore, Jia and Harman give great emphasis to Proteum, which is one of the main results from Delamaro's research and has spawned a long list of follow-up research projects. It is also worth mentioning that the underlying survey also includes a paper by Simão, which is based on a previous version of his study that was published in SBES (da Silva Simão and Maldonado, 2001).

Based on the amount of papers gathered and their respective empirical results, Jia and Harman conclude that mutation analysis has reached a fairly high level of maturity and applicability. In a similar previous study, Offutt and Untch (2001) give an account of the history of the technique and outline existing optimization techniques for mutation testing. Rather than emphasizing all the existing research in mutation testing, in their survey, Usaola and Mateo (2010) focused solely on describing cost reduction techniques for mutation testing. According to Usaola and Mateo, tools that automate mutation analysis have come a long way, yet these tools are still lacking in features that would make them compelling to practitioners.

As highlighted in the study by Rafi et al. (2012), which reports on the results of a systematic literature review and a survey, not only tools that automate mutation testing offer a poor fit for practitioners' needs. Rafi et al. looked at the academic literature by performing a systematic review, in which they collected an initial set of 24.706 papers that were reduced to 25 primary studies. From analyzing these primary studies, they found that limitations were seldom described. Instead of being reported on these primary studies, limitations were reported mostly on experience reports. Rafi et al. conjecture that this is caused by publication bias regarding the benefits. Thus, they believe that, although the reported benefits were backed up by strong sources of evidence (e.g., experiments and case studies), future efforts in the area should evaluate the drawbacks of test automation through empirical studies. In addition, Rafi et al. also elaborated and conducted a survey to gauge the view of practitioners concerning the benefits and limitations of software testing automation. The results of such a survey show that the key benefits of test automation are reusability, repeatability, and effort savings in terms of test executions. As for the limitations, they found that the main problems related to test automation are the high cost associated with designing test cases and training the staff. Such a disparity between academia and industry is also reflected in SBES's literature. To the best of our knowledge, none of the testing tools described in the selected studies has been widely adopted in industrial settings.

Grindal et al. (2005) focus on strategies to select test data by combining input parameters following several approaches. More than 30 papers were analyzed by the authors between 1983 and 2003 in an attempt to collect all the knowledge related to combination strategies. The survey structure consists of (i) the definition of all the available combining strategies, along with their high level implementation algorithms; and (ii) the attempt of making a relation among the papers and their underlying strategy, followed by the establishment of a time-line regarding their employment. One of the positive aspects of such survey is the analysis and association of each strategy with coverage criteria. Furthermore, the paper also sheds some light on the size of generated test suites and a comparison among them.

The study of McMinn (2004) also addresses the automatic generation of test data. Their survey analyzes metaheuristic search techniques, which use heuristics in order to solve combinatorial problems at an acceptable computational cost. Examples of their applicability include structural and functional testing, exercising non-functional properties, such as worst case execution time of a segment of code and grey-box properties, such as attempts to stimulate error conditions. The survey reviews metaheuristic search techniques, emphasizing how they can be employed in structural testing. As we mentioned in Section 8, test case generation has been a long-standing challenge whose efforts have been hindered by the computational cost associated with the proposed solutions. Since both software and hardware capabilities have improved, costly computational strategies as metaheuristic search techniques have the potencial to become a promising way to cope with the test case generation process. Thus, we believe that future advances in test case generation will probably involve the strategies described by McMinn (2004).

11. Concluding remarks

This paper has provided an overview of the software testing literature published in SBES. The major contribution is a picture of this research area in Brazil. We believe that such an up-to-date overview of the event's history regarding software testing can benefit practice and future research efforts. During our scoping study, we privileged broadness rather than depth. Thus, we have described only fundamental characteristics of several studies that fall into each of the underlying categories, emphasizing the ones that contributed the most to outlining the big picture.

The result of our mapping study reveals that from the inception of SBES and its tool session (1987) to 2011 there has been at least a study per year reporting on software testing research; apart from 1991 and 1996. Structural testing is the criterion emphasized by most of these contributions. Another subject investigated by our mapping study is the source of information used in most studies. We found that source code along with high-level models have been by far the information sources most widely employed in academic settings.

The information drawn from our mapping study was also used to devise a co-authorship network, illustrating how authors have collaborated in the software testing research area and who are the most prolific ones. As each research group addresses a different topic, it would be important to improve the interaction between these groups, covering the gap on software process and establishing the foundations for a strong cooperation with the industry. As for the language most used by these research groups to write research papers, it seems that Portuguese is the language of choice of Brazilian researchers: 72 of the selected studies have been published in Portuguese (which account for ≈69%) whereas only 32 have been published in English.

We have also presented our outlook on the future of software testing. As described, we contend that there is still room for research on producing industry-quality tools. Furthermore, we believe that research aimed at coming up with strategies and tools to test emergent technologies will draw a great deal of attention. Along with emergent-technology-driven efforts, research on testing oracles and embedded systems will be on the rise in the near future.

The conclusions drawn from examining the data from the flagship software engineering conference in Brazil are the following. First, we need to come up with more effective mechanisms for technology transfer from academia to industry. There is little or no interaction between practitioners and researchers in Brazil. Second, empirical studies should be carried out using real-world programs, empirical evidence on testing techniques should rely on neither contrived examples nor toy programs. Third, there should be more research projects emphasizing problems faced be practitioners and there should be more collaboration: researchers should work alongside practitioners.

Acknowledgements

This research was funded by FAPESP (grant 2009/00632-1), CAPES (grant 0340-11-1), and CNPq (grants 141976/2008-0, 142381/2009-8 and 559915/2010-1). We would like to acknowledge all colleagues who contributed to this study, especially Jeff Offutt, Fabiano Ferrari, Adenilso Simão and Sandra Fabbri. We are also grateful to the anonymous reviewers for their truly insightful comments regarding the previous version of this document.

References

Abreu, B.T., Martins, E., Sousa, F.L., 2005. Automatic test data generation for path testing using a new stochastic algorithm. In: 19th SBES, Uberlândia, MG, Brazil, pp. 247–262.

Abreu, B.T., Martins, E., de Sousa, F.L., 2007. Generalized extremal optimization: a competitive algorithm for test data generation. In: XXI SBES, pp. 342–358.

Albuquerque, P.P.B., Massollar, J.L., Travassos, G.H., 2010. ModelT2: apoio ferramental à geração de casos de testes funcionais a partir de casos de uso. In: 24th SBES – 17th Tools Session, vol. 4, pp. 55–60.

de Alencar Price, A.M., Zorzo, A.F., 1990. Ambiente de apoio ao teste estrutural de programas. In: 4th SBES, pp. 169–182.

ao Alexandre Góes, J., Renaux, D.P., 2001. PERF – um ambiente de desenvolvimento voltado para testes temporais de software. In: 15th SBES – 8th Tools Session, pp. 404–409.

Aranha, E., de Almeida, F., Diniz, T., Fontes, V., Borba, P., 2008. Automated test execution effort estimation based on functional test specifications. In: 22nd SBES – 15th Tools Session, pp. 1–6.

Aranha, E., Borba, P., 2002. Testes e geração de código de sistemas web. In: 16th SBES, pp. 114–129.

Araujo, R.F., Delamaro, M.E., 2008. TeTooDS – testing tool for dynamic systems. In: 22nd SBES – 15th Tools Session, pp. 1–6.

Barbosa, D.L., Andrade, W.L., Machado, P.D.L., Figueiredo, J.C.A., 2004. SPACES – uma ferramenta para teste funcional de componentes. In: 18th SBES – 11th Tools Session, pp. 55–60.

Barbosa, E.F., Vincenzi, A.M.R., Maldonado, J.C., 1998. Uma contribuição para a determinação de um conjunto essencial de operadores de mutação no teste de programa C. In: 12th SBES, Maringá, PR, Brazil, pp. 103–120.

Baresi, L., Young, M., 2001. Test Oracles. Technical Report CIS-TR-01-02, University of Oregon, Dept. of Computer and Information Science, Eugene, OR, USA.

Bernardo, R., Sales, R., Castor, F., Coelho, R., Cacho, N., Soares, S., 2011. Agile testing of exceptional behavior. In: 25th SBES.

Bertolino, A.,2003. Software testing research and practice. In: Proceedings of the Abstract State Machines 10th International Conference on Advances in Theory and Practice, ASM'03. Springer-Verlag, Berlin, Heidelberg, pp. 1–21.

Bertolino, A.,2007. Software testing research: achievements, challenges, dreams. In: Future of Software Engineering 2007. IEEE Computer Society, Washington, DC, USA, pp. 85–103.

Besson, F.M., Beder, D.M., Chaim, M.L.,2009. Um conjunto de ferramentas para modelagem de casos de teste de aceitação de aplicações web. In: 23rd SBES – 16th Tools Session. IEEE Computer Society, Los Alamitos, CA, USA, pp. 13–18.

Biasi, L., Becker, K., 2006. Geração automatizada de drivers e stubs de teste para JUnit a partir de especificações U2TP. In: 20th SBES, pp. 33–48.

Borges, K.N., dos Santos Ramos, F., Maldonado, J.C., Chaim, M.L., Jino, M., 1995. POKE-TOOL versão CLIPPER – uma ferramenta para suporte ao teste

estrutural de programas baseado em análise de fluxo de dados. In: 9th SBES – 2nd Tools Session, 9, DI-UFPE, Recife, PE, Brazil, pp. 483–486.
Briand, L.C., Labiche, Y.,2001. A UML-based approach to system testing. In: 4th International Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools. Springer, London, UK, pp. 194–208.
Briand, L.C., Labiche, Y., 2004. Empirical studies of software testing techniques: challenges, practical strategies, and future research. SIGSOFT Software Engineering Notes 29, 1–3.
Broy, M., Jonsson, B., Katoen, J., Leucker, M., Pretschner, A. (Eds.), 2005. Model-Based Testing of Reactive Systems: Advanced Lectures, Volume 3472 of Lecture Notes in Computer Science. Springer-Verlag.
Bueno, P.M.S., Jino, M., 1999. Geração automática de dados e tratamento de não executabilidade no teste estrutural de software. In: 13th SBES, pp. 307–322.
Cafeo, B., Masiero, P., 2011. Contextual integration testing of object-oriented and aspect-oriented programs: a structural approach for java and aspectj. In: 25th SBES.
Campanha, D.N., Souza, S.R.S., Maldonado, J.C., 2010. Teste de mutação nos paradigmas procedimental e OO: uma avaliação no contexto de estrutura de dados. In: 24th SBES, vol. 1, pp. 91–100.
Candolo, M.A.P., da Silva Simão, A., Maldonado, J.C., 2001. MGASet – uma ferramenta para apoiar o teste e validação de especificações baseadas em máquinas de estado finito. In: 15th SBES – 8th Tools Session, pp. 386–391.
Cartaxo, E.G., Machado, P.D.L., Neto, F.G.O., ao, J., Ouriques, F.S., 2008. Usando funções de similaridade para redução de conjuntos de casos de teste em estratégias de teste baseado em modelos. In: 22nd SBES, pp. 1–16.
de Carvalho, R.A., Fabbri, S.C.P.F., Maldonado, J.C., 1999. Um estudo sobre a avaliação do custo de aplicação da análise de mutantes na validação de máquinas de estados finitos. In: 13th SBES, pp. 323–338.
de Castro Guerra, P.A., de Araújo, C.S., Rocha, C.R., Martins, E., 2005. CBDUnit – uma ferramenta para testes unitários de componentes. In: 19th SBES – 12th Tools Session, Uberlândia, MG, Brazil, pp. 1–6.
Cavalcanti, T.R., Silva, F.Q.B., 2011. Historical, conceptual, and methodological aspects of the publications of the Brazilian Symposium on Software Engineering: a systematic mapping study. In: 25th SBES, pp. 14–23.
Chaim, M., Maldonado, J., Jino, M., 1989. Modelando a determinação de potenciais DU-caminhos através de análise de fluxo de dados. In: 3rd SBES, pp. 239–251.
Chaim, M.L., Jino, M., Maldonado, J.C., 1998. POKE-TOOL – estado atual de uma ferramenta para teste estrutural de software baseado em análise de fluxo de dados. In: 12th SBES – 5th Tools Session, Maringá, PR, Brazil, pp. 37–45.
Chavez, C., Kulesza, U., Soares, S., Borba, P., Lucena, C., Masiero, P., Sant'Anna, C., Piveta, E., Ferrari, F.C., Castor, F., Coelho, R., Silva, L., Alves, V., Mendonca, N., Figueiredo, E., Camargo, V., Silva, C., Pires, P., Batista, T., Cacho, N., von Staa, A., Leite, J., Silveira, F., Lemos, O.A.L., Penteado, R., Delicato, F., Braga, R., Valente, M., Ramos, R., Bonifacio, R., Alencar, F., Castro, J., 2011. The AOSD research community in Brazil and its crosscutting impact. In: 25th SBES, pp. 72–81.
Coelho, R., Cirilo, E., Kulesza, U., von Staa, A., Lucena, C., Rashid, A., 2007. JAT framework: creating JUnit-style test for multi agent systems. In: 21st SBES – 14th Tools Session, pp. 40–46.
Crespo, A.N., Pasquini, A., Jino, M., Maldonado, J.C., 1997. Cobertura dos critérios potenciais-usos e a confiabilidade do software. In: 9th SBES, pp. 379–394.
Cruz Filho, P.N., Vergilio, S.R., 2007. Uma ferramenta para o teste de web services baseado em perturbação de dados dirigida por padrões. In: 21st SBES – 14th Tools Session, pp. 63–69.
Dantas, A., Gaudencio, M., Brasileiro, F., Cirne, W., 2008. Obtaining trustworthy test results in multi-threaded systems. In: 22nd SBES, pp. 33–48.
Delamaro, M., Chaim, M., Vincenzi, A., Jino, M., Maldonado, J., 2011. Twenty-five years of research in structural and mutation testing. In: 25th SBES, pp. 40–49.
Delamaro, M., Pezzè, M., Vincenzi, A.M.R., Maldonado, J.C., 2001a. Mutant operators for testing concurrent java programs. In: 15th SBES, pp. 272–301.
Delamaro, M.E., Maldonado, J.C., 1996. Proteum – a tool for the assessment of test adequacy for C programs. In: Pham, H., Elsayed, E.A. (Eds.), Conference on Performability in Computing Systems. Rutgers University, pp. 79–95.
Delamaro, M.E., Maldonado, J.C., 1997. Teste de integração: projeto de operadores para o critério mutação de interface. In: 9th SBES, pp. 413–428.
Delamaro, M.E., Maldonado, J.C., Jino, M., Chaim, M.L., 1993. Proteum: uma ferramenta de teste baseada na análise de mutantes. In: 7th SBES – Tools Session, pp. 31–33.
Delamaro, M.E., Maldonado, J.C., Mathur, A.P., 2001b. Interface mutation: an approach for integration testing. Transactions on Software Engineering 27, 228–247.
Delamaro, M.E., Maldonado, J.C., Nakagawa, E.Y., 1997. PROTEUM/IM: uma ferramenta de apoio ao teste de integração. In: 9th SBES – 4th Tools Session, pp. 487–490.
Delamaro, M.E., Maldonado, J.C., Vincenzi, A.M.R., 2000. Proteum/IM 2.0: an integrated mutation testing environment. In: Wong, W.E. (Ed.), Mutation 2000 Symposium, Advances in Database Systems. Kluwer Academic Publishers, San Jose, CA, pp. 91–101.
Delamaro, M.E., Nardi, P.A., Lemos, O.A.L., Masiero, P.C., Spoto, E.S., Maldonado, J.C., Vincenzi, A.M.R., 2007. Static analysis of java bytecode for domain-specific software testing. In: XXI SBES, João Pessoa, PB, Brazil, pp. 325–341.
Duarte, A.N., Cirne, W., Brasileiro, F., de Lima Machado, P.D., 2005. Gridunit: using the computational grid to speed up software testing. In: 19th SBES – 12th Tools Session, Uberlândia, MG, Brazil, pp. 33–38.
Durelli, V.H.S., Araujo, R.F., Graciotto Silva, M.A., Oliveira, R.A.P., Maldonado, J.C., Delamaro, M.E., 2011. What a long, strange trip it's been: past, present, and future perspectives on software testing research. In: 25th SBES, pp. 30–39.
Eler, M.M., Delamaro, M.E., Maldonado, J.C., Masiero, P.C., 2010. Built-in structural testing of web services. In: 24th SBES, vol. 1, pp. 71–80.
Eler, M.M., Endo, A.T., Masiero, P.C., Delamaro, M.E., Maldonado, J.C., Vincenzi, A.M.R., Chaim, M.L., Beder, D.M., 2009. JaBUTiService: a web service for structural testing of java programs. In: 33rd Annual IEEE Software Engineering Workshop, pp. 69–76.
Emer, M., Vergilio, S., 2002. Selection and evaluation of test data sets based on genetic programming. In: 16th SBES, pp. 82–97.
Fabbri, S.C.P.F., Maldonado, J.C., Delamaro, M.E., Masiero, P.C., 1995a. PROTEUM/FSM – uma ferramenta para apoiar a validação de máquinas de estado finito pelo critério análise de mutantes. In: 9th SBES – 2nd Tools Session, 9, DI-UFPE, Recife, PE, Brazil, pp. 475–478.
Fabbri, S.C.P.F., Maldonado, J.C., Delamaro, M.E., Masiero, P.C., 1999a. Proteum/FSM: a tool to support finite state machine validation based on mutation testing. In: XIX International Conference of the Chilean Computer Science Society (SCCC 99), pp. 96–104.
Fabbri, S.C.P.F., Maldonado, J.C., Masiero, P.C., 1997. Aplicação do critério análise de mutante na validação de especificações baseadas em Statecharts. In: 9th SBES, pp. 429–444.
Fabbri, S.C.P.F., Maldonado, J.C., Masiero, P.C., Delamaro, M.E., 1994a. Aplicação da análise de mutantes na validação de especificações baseadas em Redes de Petri. In: 8th SBES, Curitiba, PR, Brazil, pp. 423–437.
Fabbri, S.C.P.F., Maldonado, J.C., Masiero, P.C., Delamaro, M.E., 1994b. Mutation analysis testing for finite state machines. In: 5th International Symposium on Software Reliability Engineering (ISSRE), pp. 220–229.
Fabbri, S.C.P.F., Maldonado, J.C., Masiero, P.C., Delamaro, M.E., Wong, W.E., 1995b. Mutation testing applied to validate specifications based on petri nets. In: von Bochmann, G., Dssouli, R., Rafiq, O. (Eds.), 8th International Conference on Formal Description Techniques for Distributed Systems and Communications Protocols (FORTE), Volume 43 of IFIP Conference Proceedings. Chapman & Hall, SBC Porto Alegre, RS, Brazil, pp. 329–337.
Fabbri, S.C.P.F., Maldonado, J.C., Sugeta, T., Masiero, P.C., 1999b. Mutation testing applied to validate specifications based on statecharts. In: International Symposium on Software Reliability Systems, pp. 210–219.
Farias, C.M., Machado, P.D.L., 2003. Um método de teste funcional para verificação de componentes. In: 17th SBES, pp. 193–208.
Fenn, J., Raskino, M., 2008. Mastering the Hype Cycle: How to Choose the Right Innovation at the Right Time. Harvard Business School Press, Boston, MA, USA.
Ferreira, F., Neves, L., Silva, M., Borba, P., 2010. TaRGeT: a model based product line testing tool. In: 24th SBES – 17th Tools Session, vol. 4, pp. 67–72.
Fidelis, W.I.O., Martins, E., 2004. FireWeb: uma ferramenta de suporte aos testes de regressão de aplicações web. In: 18th SBES – 11th Tools Session, pp. 1–6.
Franchin, I.G., Lemos, O.A.L., Masiero, P.C., 2007. Pairwise structural testing of object and aspect-oriented java programs. In: XXI SBES, João Pessoa, PB, Brazil, pp. 377–393.
Frankl, P.G., Weyuker, E.J., 1988. An applicable family of data flow testing criteria. Transactions on Software Engineering 14, 1483–1498.
Gerchman, J., Jacques-Silva, G., Drebes, R.J., Weber, T.S., 2005. Ambiente distribuído de injeção de falhas de comunicação para teste de aplicações java de rede. In: 19th SBES, Uberlândia, MG, Brazil, pp. 232–246.
Gerchman, J., Menegotto, C.C., Weber, T.S., 2008. Geração de cargas de falha para campanhas de injeção de falhas a partir de modelos uml de teste. In: 22nd SBES, pp. 17–32.
Gomes, J.S., da Mota Silveira Neto, A., Cruzes, D.S., de Almeida, E.S., 2011a. 25 years of software engineering in Brazil: an analysis of SBES history. In: 25th SBES, pp. 4–13.
Gomes, R., Maciel, R., Silva, B., Araujo, F., Magalhaes, A., 2011b. Moderne: a model driven process-centered software engineering environment. In: 25th SBES.
Grindal, M., Offutt, J., Andler, S.F., 2005. Combination testing strategies: a survey. Software Testing, Verification, and Reliability 15, 167–199.
Harrold, M.J.,2000. Testing: a roadmap. In: Conference on the Future of Software Engineering. ACM, New York, NY, USA, pp. 61–72.
Hausen, A.C., Vergilio, S., Souza, S.R.S., Souza, P.S.L., ao, A.S.S., 2006. ValiMPI: uma ferramenta para teste de programas paralelos. In: 20th SBES – 13th Tools Session.
Herbert, J., Price, A., 2000. Ambiente de apoio ao teste cooperativo de software – testflow manager. In: 14th SBES – 7th Tools Session, pp. 380–382.
Herbert, J.S., de Alencar Price, A.M., 1995. PROTESTE+: ambiente de validação automática de qualidade de software através de técnicas de teste e de métricas de complexidade. In: 9th SBES – 2nd Tools Session, 9, DI-UFPE, Recife, PE, Brazil, pp. 487–490.
Herbert, J.S., de Alencar Price, A.M., 1997. Estratégia de geração de dados de teste baseada na análise simbólica e dinâmica do programa. In: 9th SBES, pp. 397–411.
Herman, P.M., 1976. A data flow analysis approach to program testing. Australian Computer Journal 8, 347–354.
Jia, Y., Harman, M., 2011. An analysis and survey of the development of mutation testing. IEEE Transactions on Software Engineering 37, 649–678.
Júnior, P.de.S.L., Maldonado, J.C., Jino, M., 1992. POKE-TOOL versão COBOL. In: 6th SBES – Tools Session, pp. 24–27.
Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.,1997. Aspect-oriented programming. In: ECOOP'97 – Object-Oriented Programming, Volume 1241 of Lecture Notes in Computer Science. Springer, Finland, pp. 220–242.
Lemos, O.A.L., Ferrari, F.C., Eler, M.M., Maldonado, J.C., Masiero, P.C., 2011. Evaluation studies of software testing research in the Brazilian Symposium on Software Engineering. In: 25th SBES, pp. 56–65.

Lemos, O.A.L., Ferrari, F.C., Masiero, P.C., Lopes, C.V.,2006. Testing aspect-oriented programming pointcut descriptors. In: 2nd Workshop on Testing Aspect-Oriented Programs. ACM, New York, NY, USA, pp. 33–38.
Lemos, O.A.L., Franchin, I.G., Masiero, P.C., 2009. Integration testing of object-oriented and aspect-oriented programs: a structural pairwise approach for java. Science of Computer Programming 74, 861–878.
Lemos, O.A.L., Masiero, P.C., 2008. Integration testing of aspect-oriented programs: a structural pointcut-based approach. In: 22nd SBES, pp. 49–64.
Lemos, O.A.L., Masiero, P.C., 2011. A pointcut-based coverage analysis approach for aspect-oriented programs. Information Sciences 181, 2721–2746.
Lemos, O.A.L., Vincenzi, A.M.R., Maldonado, J.C., Masiero, P.C., 2004. Teste de unidade de programas orientados a aspectos. In: 18th SBES, pp. 55–70.
Lemos, O.A.L., Vincenzi, A.M.R., Maldonado, J.C., Masiero, P.C., 2007. Control and data flow structural testing criteria for aspect-oriented programs. Journal of Systems and Software 80, 862–882.
Li, J.J., London, S., Vilela, P., Horgan, J.R., 1999. xSUDS-SDL: a tool for diagnosis and understanding software specifications. In: 13th SBES – 6th Tools Session, pp. 5–8.
Lyu, M.R. (Ed.), 1996. Handbook of Software Reliability Engineering. McGraw-Hill Inc., Hightstown, NJ, USA.
Macedo, A., Andrade, W., Machado, P., 2011. Realtimepco $\widehat{A}$ – a tool for real-time embedded systems testing execution. In: 25th SBES.
Maldonado, J., Chaim, M., 1988. Jino, Seleção de casos de testes baseada em fluxo de dados através dos critérios potenciais usos. In: 2nd SBES, pp. 24–35.
Maldonado, J.C., Delamaro, M.E., Fabbri, S.C.P.F., ao, A.S.S., Sugeta, T., Vincenzi, A.M.R., Masiero, P.C., 2000. Proteum: a family of tools to support specification and program testing based on mutation. In: Wong, W.E. (Ed.), MUTATION 2000: A Symposium on Mutation Testing for the New Century, Advances in Database Systems. Kluwer Academic Publishers, Hingham, MA, USA, pp. 113–116.
Maldonado, J.C., Vergílio, S.R., Chaim, M.L., Jino, M., 1992. Critérios potenciais usos: análise da aplicação de um benchmark. In: 6th SBES, pp. 357–370.
Martins, E., 1995. Integrando injeção de falhas e testes formais na validação da tolerância a falhas. In: 9th SBES, 9, DI-UFPE, Recife, PE, Brazil, pp. 223–239.
Mathur, A., Wong, W., 1993. Evaluation of the cost of alternate mutation strategies. In: 7th SBES, pp. 320–335.
Mathur, A.P., 2008. Foundations of Software Testing. Pearson Education, USA.
McMinn, P., 2004. Search-based software test data generation: a survey: research articles. Software Testing, Verification, and Reliability 14, 105–156.
Mendes, N., Moraes, R., Martins, E., Madeira, H., 2006. Jaca tool improvements for speeding up fault injection campaigns. In: 20th SBES – 13th Tools Session.
Nakagawa, E.Y., Maldonado, J.C., 1999. ITOOL – uma ferramenta para injeção de defeitos de software. In: 13th SBES – 6th Tools Session, pp. 49–52.
Nakagawa, E.Y., Maldonado, J.C., 2011. Contributions and perspectives in architectures of software testing environments. In: 25th Brazilian Symposium on Software Engineering (SBES), pp. 66–71.
Nakazato, K.K., Alexandrino, M., Maldonado, J.C., Fabbri, S.C.P.F., Masiero, P.C., 1995. MGASET – módulo de geração de seqüências de teste. In: 9th SBES – 2nd Tools Session, 9, DI-UFPE, Recife, PE, Brazil, pp. 479–482.
Nardi, P.A., Delamaro, M.E., Spoto, E.S., Vincenzi, A.M.R., 2005. Jabuti/bd: utilização de critérios estruturais em aplicações de bancos de dados java. In: 19th SBES – 12th Tools Session, Uberlândia, MG, Brazil, 45–50.
Neto, A.C.D., Lima, G.M.P.S., Travassos, G.H., 2005. FAROL: uma ferramenta de apoio à aplicação de heurísticas de ordenação de classes para teste de integração. In: 19th SBES – 12th Tools Session, Uberlândia, MG, Brazil, pp. 13–18.
Neto, A.D., Travassos, G.H., 2006. Maraká: apoio ao planejamento e controle de testes de software. In: 20th SBES – 13th Tools Session, pp. 85–90.
Neves, V., Masiero, P., 2011. Extensão da ferramenta jabuti/aj para teste de integração de programas orientados a objetos e a aspectos. In: 25th SBES.
de Nunes, P.R.de.A.F., de Melo, A.C.V., 2004. Ocongra – uma ferramenta para geração de grafos de fluxo de controle de objetos. In: 18th SBES – 11th Tools Session, pp. 7–12.
Offutt, J., Untch, R.H., 2001. Mutation Testing for the New Century. Kluwer Academic Publishers, Hingham, MA, USA, pp. 34–44.
Oliveira, K.A., Machado, P.D.L., Andrade, W.L., 2003. CASLTEST – test case, test oracle and test data generation from CASL specifications. In: 17th SBES, pp. 73–78.
de Oliveira, R.A.P., Delamaro, M.E., Nunes, F.L.S.,2009. O-FIm – oracle for images. In: 23rd SBES – 16th Tools Session. IEEE Computer Society, Los Alamitos, CA, USA, pp. 1–6.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering, pp. 71–80.
Pinto, I.M., de Alencar Price, A.M., 1998. Um sistema de apoio ao teste de programas orientados a objetos com uma abordagem reflexiva. In: 12th SBES, Maringá, PR, Brazil, pp. 87–102.
Price, A., Purper, C., Garcia, F., 1987. PROTESTE: projeto de uma ferramenta para teste de programas. In: 1st SBES, pp. 20–21.
Price, A., Rosa, F., Bins, I., Silva, J., Santi, L., Dias, P., Rombaldi, V., 1992. PROTESTE – ambiente de apoio ao teste de programas. In: 6th SBES – Tools Session, pp. 29–30.
Rafi, D., Moses, K., Petersen, K., Mantyla, M., 2012. Benefits and limitations of automated software testing: systematic literature review and practitioner survey. In: 7th International Workshop on Automation of Software Test (AST), pp. 36–42.
Ré, R., Masiero, P.C., 2005. Avaliação da abordagem incremental no teste de integração de programas orientados a aspectos. In: 19th SBES, Uberlândia, MG, Brazil, pp. 168–183.
Ré, R., Masiero, P.C., 2007. Integration testing of aspect-oriented programs: a characterization study to evaluate how minimize the number of stubs. In: XXI SBES, João Pessoa, PB, Brazil, pp. 411–426.
Ré, R., dos, A.L., Domingues, S., Masiero, P.C., 2008. Um catálogo de stubs para apoiar o teste de integração de programas orientados a aspectos. In: 22nd SBES, pp. 65–80.
Rocha, A.D., da Silva Simão, A., Maldonado, J.C., Masiero, P.C., 2005. Uma ferramenta baseada em aspectos para o teste funcional de programas java. In: 19th SBES, Uberlândia, MG, Brazil, pp. 263–278.
dos Santos Domingues, A.L., da Silva Simão, A., Vincenzi, A.M.R., Maldonado, J.C., 2002. EvalTool: um ambiente de apoio à avaliação e seleção de ferramentas de teste para programas orientados a objetos. In: 16th SBES, pp. 384–389.
Silva, J.B., Price, A.M.A., 1994. Métrica de complexidade de software baseado em critério de seleção de caminhos de teste. In: VIII SBES, Curitiba, PR, Brazil, pp. 471–485.
Silva, W., Junior, V.S., Mattiello-Francisco, M., Passos, D., 2006. QSEE-TAS: uma ferramenta para execução e relato automatizados de testes de software para aplicações espaciais. In: 20th SBES – 13th Tools Session.
Silva, W.P., Santiago, V., Vijaykumar, N.L., Mattiello-Francisco, F., 2007. SPAC: ferramenta para processamento e análise de dados científicos no processo de validação de software em aplicações espaciais. In: 21st SBES, pp. 70–76.
da Silva Simão, A., Maldonado, J.C., 2001. MuDeL: a language and a system for describing and generating mutants. In: 15th SBES, pp. 240–255.
da Silva Simão, A., Petrenko, A., Maldonado, J.C.,2007. Experimental evaluation of coverage criteria for FSM-based testing. In: XXI SBES. João Pessoa, PB, Brazil, pp. 359–374.
da Silva Simão, A., do, S., de Souza, R.S., Maldonado, J.C., 2003. A family of coverage testing criteria for Coloured Petri Nets. In: 17th SBES, pp. 209–224.
Simão, A., Maldonado, J., 2000a. Proteum-RS/PN: a tool to support edition, simulation and validation of Petri Nets based on mutation testing. In: 14th SBES, pp. 227–242.
Simão, A., Maldonado, J., 2000b. Proteum-RS/PN: uma ferramenta para apoiar a edição, simulação e validação de Redes de Petri baseada no teste de mutação. In: 14th SBES – 7th Tools Session, pp. 376–379.
Simão, A., Vincenzi, A., Maldonado, J., 2002. mudelgen: a tool for processing mutant operator descriptions. In: 16th SBES – 9th Tools Session, pp. 426–431.
Simão, A.d.S., Ambrósio, A.M., Fabbri, S.C.P.F., do Amaral, A.S.M.S., Martins, E., Maldonado, J.C., 2005. Plavis/fsm: an environment to integrate fsm-based testing tools. In: 19th SBES – 12th Tools Session, Uberlândia, MG, Brazil, pp. 57–62.
de Sousa Santos, I., Santos, A.R., de Alcântara dos Santos Neto, P., 2010. FERRARE GT: automação de testes de desempenho e estresse via testes funcionais. In: 24th SBES – 17th Tools Session, vol. 4, pp. 49–54.
Souza, D., Maldonado, J., Barbosa, E., 2011a. Uma contribuicao a submissão e avaliacao automática de trabalhos de programacao com base em atividades de teste. In: 25th SBES.
Souza, D.M., Maldonado, J.C., Barbosa, E.F.,2011b. ProgTest: an environment for the submission and evaluation of programming assignments based on testing activities. In: 24th IEEE-CS Conference on Software Engineering Education and Training (CSEET). IEEE Computer Society, Honolulu, HI, pp. 1–10.
do Rocio Senger de Souza, S., Maldonado, J.C., 1997. Avaliação do impacto da minimização de conjuntos de casos de teste no custo e eficácia do critério análise de mutantes. In: 9th SBES – 4th Tools Session, pp. 445–460.
Souza, S.R.S., Maldonado, J.C., Fabbri, S.C.P.F., 2001. FCCE: uma família de critérios de teste para validação de sistemas especificados em Estelle. In: 15th SBES, pp. 256–271.
Souza, S.R.S., Maldonado, J.C., Fabbri, S.C.P.F., Souza, W.L., 1999. Mutation testing applied to Estelle specifications. Software Quality Journal 8, 285–301.
Spoto, E., Jino, M., Maldonado, J., 2000. Teste estrutural de software: uma abordagem para aplicações de banco de dados relacional. In: 14th SBES, pp. 243–258.
Sugeta, T., Maldonado, J.C., Fabbri, S.C.P.F., 1999. PROTEUM-RS/ST – uma ferramenta para apoiar a validação de especificações statecharts baseada no critério análise de mutantes. In: 13th SBES – 6th Tools Session, pp. 41–44.
Sugeta, T., Maldonado, J.C., Fabbri, S.C.P.F., 2001. Proteum/ST: a tool to support statecharts validation based on mutation testing. In: Ideas 2001 – Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software, pp. 370–384.
Sugeta, T., Maldonado, J.C., Wong, W.E., 2004. Mutation testing applied to validate SDL specifications. In: Groz, R., Hierons, R.M. (Eds.), 16th IFIP International Conference on Testing of Communicating Systems (TestCom 2004), Volume 2978 of Lecture Notes in Computer Science, IFIP. Springer, Oxford, United Kingdom, pp. 193–208.
Usaola, M., Mateo, P., 2010. Mutation testing cost reduction techniques: a survey. IEEE Software 27, 80–86.
Vacaro, J.C., Weber, T.S., 2006. Injeção de falhas na fase de teste de aplicações distribuídas. In: 20th SBES, pp. 161–176.
Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M., 2008. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review 39, 50–55.
Vegas, S., Juristo, N., Basili, V.R., 2009. Maturing software engineering knowledge through classifications: a case study on unit testing techniques. IEEE Transactions on Software Engineering 35, 551–565.
Venâncio, J., ao, C.G., Mendes, E., Souza, E.,2009. RBTTool – uma ferramenta de apoio à abordagem de teste de software baseado em riscos. In: 23rd SBES – 16th Tools Session. IEEE Computer Society, Los Alamitos, CA, USA, pp. 7–12.
Vergilio, S., Maldonado, J., Jino, M., 1993. Uma estratégia para geração de dados de teste. In: 7th SBES, pp. 306–319.

Vergilio, S., Maldonado, J.C., Jino, M., 1992. Caminhos não executáveis na automação da atividade de teste. In: 6th SBES, pp. 343–356.
Vergilio, S.R., Maldonado, J.C., Jino, M., 1994. Caminhos não executáveis no teste de integração: caracterização, previsão e determinação. In: VIII SBES, Curitiba, PR, Brazil, pp. 453–467.
Vergilio, S.R., Maldonado, J.C., Jino, M., 1995. Geração de dados de teste: uma estratégia que preserva a hierarquia de critérios. In: 9th SBES, 9, DI-UFPE, Recife, PE, Brazil, pp. 211–222.
Vilela, P.R.S., Maldonado, J.C., Cruzes, D.S., Jino, M., 1998. ViewGraph: visualizing control flow graphs and call graphs. In: 12th SBES – 5th Tools Session, Maringá, PR, Brazil, pp. 29–35.
Vilela, P.R.S., Maldonado, J.C., Jino, M., 1999. Data flow based integration testing. In: 13th SBES, pp. 393–409.
Vincenzi, A.M.R., Maldonado, J.C., Barbosa, E.F., Delamaro, M.E., 1999. Operadores essenciais de interface: um estudo de caso. In: 13th SBES, pp. 373–391.
Vincenzi, A.M.R., Maldonado, J.C., Wong, W.E., Delamaro, M.E., 2005. Coverage testing of java programs and components. Science of Computer Programming 56, 211–230.
Vincenzi, A.M.R., Wong, W.E., Delamaro, M.E., Maldonado, J.C., 2003. JaBUTi: a coverage analysis tool for java programs. In: 17th SBES, pp. 79–84.
Wieringa, R., Maiden, N., Mead, N., Rolland, C., 2005. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirement Engineering 11, 102–107.
Wong, W.E., Maldonado, J.C., Delamaro, M.E., Mathur, A.P., 1994. Constrained mutation in C programs. In: VIII SBES, Curitiba, PR, pp. 439–452.
Yano, T., da Silva Simão, A., Maldonado, J.C., 2003. Proteum/SML: uma ferramenta de apoio ao teste de mutação para a linguagem Standard ML. In: 17th SBES, pp. 67–72.

Vinicius Humberto Serapilha Durelli is a Ph.D. candidate in Computer Science at the University of São Paulo, Brazil. As part of his doctoral studies, from 2011 to 2012, he was visiting scholar at the George Mason University, Virginia, USA. He received his M.S. in Computer Science from the Federal University of São Carlos in 2008. His current research interests include software testing, high-level language virtual machines, empirical software engineering, test-driven development, and refactoring. His main publications can be found on Lattes Platform (http://lattes.cnpq.br/8816910024419957).

Rodrigo Fraxino Araujo is a Ph.D. candidate in Computer Science at the University of São Paulo, Brazil. He holds a B.S. and a M.S. in Computer Science from Univem, Brazil. His research interests are software testing and embedded systems. A list of his main publications can be found on Lattes Platform (http://lattes.cnpq.br/3544209255871704). Currently, he is also an IBM employee.

Marco Aurelio Graciotto Silva is a Professor in the Federal Institute of Education, Science and Technology of São Paulo (IFSP) at São João da Boa Vista, Brazil. In 2001, he received a B.S. degree in Computer Science from the State University of Maringá (UEM) in Brazil. In 2005, he received his MS degree in Computer Science from the University of São Paulo, Brazil. In 2012, he received his D.S. degree in Computer Science also from the University of São Paulo. His research interests include learning objects, software engineering education, open source software testing, and requirements engineering. A list of his main publications can be found on Lattes Platform (http://lattes.cnpq.br/9383290036853173).

Rafael Alves Paes de Oliveira is a Ph.D. student in Computer Science at the University of São Paulo, Brazil. He graduated in Computer Science from Univem, Brazil. He received his M.S. in Computer Science from the University of São Paulo. His research interests are testing oracles, software testing, verification and validation, software engineering, and software quality. A list of his main publications can be found on Lattes Platform (http://lattes.cnpq.br/0793753941171478).

Jose Carlos Maldonado received his B.S. in Electrical Engineering/Electronics in 1978 from the University of São Paulo (USP/Brazil), his M.S. in Telecommunications/Digital Systems in 1983 from the National Space Research Institute (INPE/Brazil), his D.S. degree in Electrical Engineering/Automation and Control in 1991 from the University of Campinas (UNICAMP/Brazil), and his Post-Doctoral at Purdue University (USA) in 1995–1996. He worked at INPE (National Space Research Center) from 1979 up to 1985 as a researcher. In 1985 he joined the Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). His main research interests are related to software engineering, software testing and validation, experimental software engineering, software quality, SE education and training, web systems and reactive and embedded systems. A list of his main publications can be found on Lattes Platform (http://lattes.cnpq.br/8807333466702951).

Marcio Eduardo Delamaro graduated in Computer Science from Universidade Estadual de Campinas (1985), received his M.S. in Computer Science from the University of Sáo Paulo (1993), and his Ph.D. in Computational Physics from the University of Sáo Paulo (1997) with sandwich program at Purdue Univesity, United States. Conducted postdoctoral studies at the Politecnico di Milano, Italy, between 2000 and 2001 and earned the title of Associate Professor in the area of Software Engineering from the University of Sáo Paulo in 2005. He is currently associate professor in the Department of Computer Systems (SSC) at University of Sáo Paulo, Campus Sáo Carlos. He has experience in software engineering mainly in the areas of software testing, mutation testing and structural testing. A list of his main publications can be found on Lattes Platform (http://lattes.cnpq.br/2844974351441051).