Free and Open Source Software Development and Research: Opportunities for Software Engineering

Fabio Kon, Paulo Meirelles, Nelson Lago
FLOSS Competence Center
Department of Computer Science - IME
University of São Paulo, Brazil
{fabio.kon,paulormm,lago}@ime.usp.br

Antonio Terceiro, Christina Chavez, Manoel Mendonça
Software Engineering Lab (LES)
Department of Computer Science
Federal University of Bahia, Brazil
{terceiro,flach,manoel.mendonca}@dcc.ufba.br

***Abstract*—Free/Libre/Open Source Software (FLOSS) communities have produced a large amount of valuable software that is directly or indirectly used daily by any person with access to a computer. The field of Software Engineering studies processes, mechanisms, tools, and frameworks for the development of software artifacts. Historically, however, most of Software Engineering research and education does not benefit from the large and rich source of data and experimental testbeds offered by FLOSS projects and their hundreds of millions of lines of working code. In this paper, we discuss how Software Engineering research and education can greatly benefit from the wealth of information available in the FLOSS ecosystem. We then evaluate how FLOSS has been used, up to now, by papers published in the Brazilian Symposium on Software Engineering. Finally, we present an agenda for the future, proposing concrete ways to exploit the synergies between research and education in Software Engineering and FLOSS projects.**

I. Introduction

Software Engineering (SE) research aims at advancing our understanding of the process of software development and its outcomes. As in other fields of research, knowledge must be built on top of verifiable evidence. However, it is not uncommon to see Software Engineering research based on non-public data and non-reproducible methodologies, or even based on no data at all.

Moreover, the education of future software engineers (and possibly SE researchers) should be based on the study of existing real-world software, very much like the education of future artists include extensive review of past artistic achievements and styles or the education of future architects is based on the study of classical buildings and construction styles.

Free/Libre/Open Source Software (FLOSS)[1] offers SE researchers the opportunity of basing their research on abundant and publicly available data, freely accessible data analysis, and software development tools. These are basic building blocks for reproducible and extensible science. Accordingly, SE apprentices can apply the concepts that they learn in the classroom in real-world situations made available by FLOSS projects.

[1]In this work, the acronym "FLOSS" is used as a representative for "Free Software", "Open Source Software" (OSS) and "Free/Open Source Software" (FOSS).

The international SE community has already recognized the potential of FLOSS in both research and education [1], [2], as well as opportunities for future research [3]. In this paper, we discuss important opportunities brought by FLOSS to the SE community, and analyze whether or not the Brazilian SE community is taking advantage of them. We performed a study over papers published in the Brazilian Symposium on Software Engineering (SBES) which, from the top of its 25 years, is the main Brazilian venue of publication for SE researchers. We analyzed research papers published between 1999 and 2010, tools papers published between 2001 and 2010 and SE education papers published between 2008 and 2010.

In this analysis, we classified SE research according to the way FLOSS was used in research reported by SBES papers and compared these categories with the discussed opportunities. Based on that, we propose an agenda to better take advantage of FLOSS in SE research and education.

The remainder of this paper is organized as follows. Section II presents definitions and basic concepts about the FLOSS ecossystem, and Section III discusses opportunities that FLOSS brings for SE researchers and educators. In Section IV, we describe our study and its results. In Section V, we discuss the results in light of the identified opportunities. We conclude the paper in Section VI by proposing an agenda for fostering the synergies between FLOSS and SE in Brazil.

II. About FLOSS

"FLOSS" is a broad acronym that refers to software that promotes user freedom, does not discriminate users or uses and, at the same time, is based on a collaborative, efficient, and open development process. FLOSS basically allows users to use, study, modify, and redistribute the software with virtually no restrictions except, sometimes, preventing users to impose restrictions on other users. In order to make this possible, access to the source code is a necessary condition.

Typically, such software exists by means of development projects that are centered around some publicly-accessible source code, where both developers and users interact with each other mainly over the Internet. The code is necessarily licensed under terms that comply with either the Free Software

definition [2] or the Open Source Definition [3].

Most software recognized as "Free Software" or "Open Source" actually complies with both definitions, even if their specific terms are quite different from one another; The use of a specific terminology reflects the desire to emphasize either the ethical or the technical aspects of the same phenomena. The acronym FLOSS (Free/Libre/Open-Source Software), which we use in this text, is gaining strong acceptance, as it encompasses both major terminologies.

A. Why FLOSS?

FLOSS offers benefits from social, ethical, technical, and economic points of view [4], [5]. From a social point of view, FLOSS may be one of the many elements in easing access to technological resources to the population at large. Also, the use and support of FLOSS in e-government may offer more transparency and ease access to services and data to citizens. Finally, source code access and sharing eases learning and promotes participation in the development of software that might otherwise impose restrictions beyond those embodied in the Law.

From an ethical point of view, if a resource may be easily shared, preventing this should only be pursued under very specific circumstances and with very good motivation. The growth of the FLOSS model in the last decades undermines arguments in favor of such restrictions for software, as its sharing does not prevent other benefits to society.

From the technical and economical points of view, FLOSS offers a development model that may result in high quality code that gets adapted quickly to different situations with lower direct costs. It also leverages the development effort of different companies or individuals into common products, reducing the duplication of effort.

B. FLOSS Development Process

From a Software Engineering point of view, the most interesting aspect of FLOSS is its development process. A FLOSS project starts when an individual developer, or an organization, decides to make a software product publicly available on the Internet so that it can be freely used, modified and redistributed.

After an initial version is released and advertised in the appropriate communication channels, early adopters will start using the product. Some of these early adopters may be software developers themselves, who will be able to review the source code and propose changes that fix defects for their specific environments or add features for their own use cases. These changes may be sent back to the original developer(s) in the form of patches[4] . The project leader(s) will review the proposed changes and apply them (or not) to the official version, so that when a new release is made, end users will have access to these new functionalities or bug fixes.

In the course of time, each release of the product may have more features and be more portable than its predecessor due to contributions from outside developers. The most frequent contributors may gain trust from the initial developer(s), and receive direct write access to the official source code of the project, becoming able to make changes directly to the official version of the product.

This development process is often driven by means of a version control system (VCS). While the repository is often publicly available for read access, write access is restricted to a limited group of developers. Other developers will need their patches to be reviewed by a developer with the needed privileges in order to get their contributions into the project's official repository.

Besides the version control system, most FLOSS projects use a fairly standardized environment for collaboration between their developers: an issue tracker (or bug tracker) for organizing pending, ongoing and future activities; one or more mailing lists for coordination and communication between the team; and usually a web-based content management system (such as a wiki) for community-driven documentation writing.

The process of developers joining FLOSS projects and gaining responsibility may range from very informal to very formal, depending on the project. Small projects normally have informal procedures for that, e.g. one existing developer just offers an account in the version control system to the new contributor. Larger projects, on the other hand, may have more "bureaucratic" processes for accepting a new developer with privileged access to the project's resources. This "bureaucracy" may involve filling forms with an application, signing terms of copyright transference and other procedures. [6], [7]

The following characteristics make FLOSS projects different enough from "conventional" software projects, making them interesting objects of study:

- **Source code availability.** Source code of FLOSS projects is always available on the Internet, since most projects have a publicly-accessible version control repository.
- **User/developer symbiosis.** In most FLOSS projects the developers are also users of the software, and they also provide requirements. Maybe because of that, several free software projects do not have explicit requirement documents, and the development proceeds at a pace adequate for the developers to satisfy their own needs.
- **Non-contractual work.** A large amount of work in FLOSS projects is done in a non-contractual fashion. This does not imply that the developers are necessarily volunteers, but only that there is no central management with control over all of the developers' activities.
- **Work is self-assigned.** The absense of a central central management with control over the contributors' activities promotes work self-assignment: volunteer developers tend to work on the parts of the project that most appeal to them, and employed developers will work on the parts that are of most interest to their employers.

[2] `http://www.gnu.org/philosophy/free-sw.html`

[3] `http://www.opensource.org/docs/definition.html`

[4] A patch is a file that describes the changes to one or more files, normally of textual content, by describing which lines to remove and which lines to add. After receiving a patch the developer can reproduce the changes proposed by the sender in his/her own copy of the source code.

- **Geographical Distribution.** In most FLOSS projects the developers are spread among several different locations in the world. In projects with high geographical dispersion, communication is mostly performed through electronic means.

The Software Engineering literature tends to portrait FLOSS as a homogeneous phenomenon[8], but most of these characteristics do not apply to all FLOSS projects, and some of them may be manifested in different ways across projects.

III. FLOSS Opportunities for Academic Software Engineering

FLOSS brings multiple benefits for Software Engineering in an academic scenario, providing benefits both for education and research.

A. Using FLOSS for Software Engineering Education

Both FLOSS development and development processes taught in traditional SE textbooks address the challenges of multi-person construction and maintenance of multi-version programs [9], but with development processes, work practices, and project forms that differ significantly and in interesting ways [3]. This raises the question of whether FLOSS development could be useful for educational purposes in SE and computer science undergraduate courses, by exposing students and teachers to different aspects of professional practice. Such exposure may encompass a wide range of issues and activities that includes problem solving, management, ethical and legal concerns, written and oral communication, working as part of a team, and remaining current in a rapidly changing discipline [10].

This issue has been exploited in different international forums [11], [12], [13], [14], [15], [16]. The perspective of using FLOSS for educational purposes has brought a series of interesting opportunities and challenges into the Software Engineering Education agenda, such as:

- Involving students and faculty in large-scale FLOSS projects to provide them with real-world experience and an understanding of the issues found in large, complex software projects [16], [13];
- Reinvigorating the CS/SE curriculum and faculty members [13]. FLOSS promotes project and problem-based learning in which developers work on projects that interest them; by working on interesting and meaningful projects, they can learn concepts, skills, and aptitudes [17];
- Exploiting successful open source projects, in which software is highly modular and APIs are well documented, for teaching principles and good practices [15];
- Providing quantitative data from real, open and freely available source code on which to perform analysis and base decisions [10].

Evidently, the use of FLOSS also brings difficulties, such as:

- Because of its intrinsic characteristics, FLOSS development is not tipically representative of many traditional development methodologies. Therefore, its usefulness in SE education is restricted to only part of SE topics or to methodologies related to common FLOSS development processes;
- Academic experience on FLOSS may be seen as less important for the preparation of professionals in environments where proprietary software is the norm, especially if it would result in less academic experience with other widespread technologies;
- Involving teachers and students in FLOSS-related studies brings specific difficulties, for example finding projects that gave appropriate size and complexity (not big enough to be unbearable and not small enough to be trivial), allow exploring the required course topics and have enough documentation for starters.

B. Using data from FLOSS projects in Software Engineering Research

While data from private software development is often scarce and may impose high acquisition costs, FLOSS project data is plenty and readily available for free to anyone with an Internet connection. Private data usually cannot be shared for replicating research results due to issues such as confidentiality requirements and non-disclosure agreements, whereas FLOSS data is already public anyway [1].

FLOSS projects can serve as an abundant source of data for Software Engineering research. Hundreds of thousands of projects make available their source code, which can be analyzed either manually or by automated tools. For example, in a recent work, we studied 6773 projects using automated tools and showed how the source code metrics affected the attractiveness of the projects, in terms of number of downloads, page hits, and project members [18]. In another study, our group analyzed the impact of changes in the licenses of 756 FLOSS projects over 44 months to show how these changes affect the attractiveness of these projects [19]. In yet another study we analyzed the full history of 7 FLOSS web server projects, containing 13553 software changes, in order to investigate developers' level of participation in the project as a predictor for variations in structural complexity [20].

Most FLOSS projects have all their communication media and development resources – version control system, bug tracking system, mailing lists, documentation websites – publicly available on the Internet, and through them researchers can monitor, track and analyze the activity of the development group working in a certain software artifact. Such data is being used extensively in contemporary Software Engineering research, and systems like bicho [21], Analizo [22] and others can help automate the acquisition and analysis of this kind of data from publicly accessible repositories on the Web.

Again, it should be noted that FLOSS does not cover all possible data relevant for SE; some methodologies and topics cannot be assessed by means of FLOSS data analysis. FLOSS-based research, however, is not limited to the previously existing software engineering research sub-areas: studying FLOSS in itself is also a promising research area. FLOSS development

groups have been producing large and complex products using methodologies and processes that are sometimes substantially different from "conventional" software development. Understanding how FLOSS works and its differences and similarities with regard to "conventional" software engineering practices can help researchers and practitioners to enhance the software development practice in general.

Research on FLOSS development has brought a series of interesting topics into the Software Engineering research agenda, such as the social structure of development communities [23], [24], [6], [7], communication and work flow patterns in FLOSS projects, [25], [26], developer evolution and participation [27], [20], and attractiveness of FLOSS projects [19], [18].

C. FLOSS software products in research activities

This opportunity applies not only to Software Engineering research, but also to Computer Science research in general, as well as to any research in which there is software development. Both researchers and society can benefit from the interplay between FLOSS and research activity.

Releasing the source of software produced in research activities meets the basic scientific principle of reproducibility. The scientific method demands that, when describing a scientific achievement in a paper, researchers must provide all the details required for an independent group of scientists to reproduce the experiment to validate it, checking whether or not the same results are achieved. When software plays an important role in a scientific study, either as subject under study or as a data analysis tool, making it publicly available under a FLOSS license will facilitate the results to be expanded and built upon.

Another benefit of releasing research-originated software as FLOSS is fostering technology transfer from research institutions to society: being freely licensed allows research prototypes to be enhanced into production-class products without the researchers having to be (necessarily) involved indefinitely.

There are several examples of successful FLOSS products that were initially developed in the context of research activities: the PostgreSQL relational database management system and the BSD family of operating systems were created at the University of California, Berkeley; the Xen virtualization technology was initially developed at the University of Cambridge; the Ginga platform for digital television was developed in PUC-Rio and Federal University of Paraíba and later chosen as the official standard for Brazilian National Digital Television system. It's hard to imagine the consequences for the software technology landscape if these projects have been kept as private research prototypes.

Technology transfer also happens the other way around: the fact that researchers can build their work on top of a vast amount of existing software tools, middleware, and environments enable researchers to go deeper into their research without having to waste time, effort, and financial resources on reinventing the wheel. In 1676, Isaac Newton stated that he had been able to see further by standing on the shoulders of giants. FLOSS enables today's researchers to do the same, by reusing software components that are required for their research but that are not the core objective of the scientific advancement they seek. Researchers may use existing FLOSS systems such as the Linux operating system, the GCC compiler, the Eclipse framework for IDEs, or the OpenNebula Cloud Computing middleware and test their ideas by changing just a small portion of them. This way researchers can concentrate on the exact point they want to explore and advance science and technology more rapidly with more reliable results.

IV. FLOSS in SBES

To evaluate where the Brazilian Software Engineering research community stands with regard to FLOSS, we carried out a study on the papers published in the Brazilian Symposium on Software Engineering (SBES). We have examined: (i) main track papers between 1999 and 2010, (ii) tools session papers between 2001 and 2010, and (iii) Software Engineering Education Forum (*Fórum de Education em Engenharia de Software* – FEES) papers from 2008, its first edition, to 2010. This section describes this study and its findings.

A. Data acquisition

We obtained the full text of SBES main track, tools session, and FEES papers from the following sources:

- Main track papers from SBES 1999, 2001, 2002, 2004, 2006, 2007, 2008, and 2009 were obtained from the Brazilian Computer Science Digital Library[5] ("Biblioteca Digital Brasileira de Computação" – BDBComp), maintained by UFMG's Databases Laboratory.
- Main track papers from SBES 2005 were obtained contacting the program chair and from SBES 2010 were obtained from the IEEE Xplore Digital Library[6].
- Tools papers from SBES 2001 and 2002 were also obtained from BDBComp. Because SBES tools session papers from most years are not available on-line, we got 2004, 2005, 2006, 2007, 2008, 2009, and 2010 papers contacting the program chair of each edition.
- FEES papers from 2008 and 2009 were obtained at the FEES homepage hosted by PUC-Rio[7], and 2010 papers were obtained from CBSoft 2010 organizers.

TABLE I
MAIN TRACK PAPERS ANALYZED, BY YEAR

Year	Analyzed	Mention FLOSS	Relative frequency
1999	26	0	0.00
2001	20	0	0.00
2002	18	2	0.11
2004	17	1	0.06
2005	21	2	0.10
2006	19	0	0.00
2007	23	2	0.09
2008	19	1	0.05
2009	24	6	0.25
2010	19	11	0.58

[5] http://www.lbd.dcc.ufmg.br/bdbcomp
[6] http://ieeexplore.ieee.org
[7] http://fees.inf.puc-rio.br/

TABLE II
TOOL PAPERS ANALYZED, BY YEAR

Year	Analyzed	Mention FLOSS	Relative frequency
2001	18	0	0.00
2002	19	3	0.16
2004	15	2	0.13
2005	12	2	0.17
2006	25	7	0.28
2007	14	2	0.14
2008	11	3	0.27
2009	12	4	0.33
2010	16	7	0.44

The number of main track and tools papers analyzed by year are presented in Tables I and II, respectively. FEES papers were 14 from 2008, 8 for 2009 and 8 for 2010. Despite not being able to obtain every single paper since 1999, the papers we obtained gave us a reasonable approximation of what happened during the period. We analyzed a total amount of 378 papers: 206 from the main track, 142 from the tools session, and 30 from the Software Engineering Education Forum.

The PDF files were processed by a script that extracted their text, and matched their contents against the following terms: "software(s) livre(s)", "ferramenta(s) livre(s)", "ferramenta(s) aberta(s)", "software(s) aberto(s)", "código aberto", "repositório(s) de software", "free software", "open source", "open software", "libre software", "software repository", "OSS", "FLOSS", "FOSS", and "OSSD". To validate this identification process of the papers related to FLOSS, we checked all of them manually by reading the abstract and the sections that include one of the terms above.

Tables I and II present the raw results for the SBES main track and tools session, respectively. The tables presents the number of papers analyzed in each year, how many of them mentioned FLOSS, and the relative frequency of papers mentioning FLOSS (i.e. the ratio between the number of papers mentioning FLOSS and the total of papers). We can see that the number of papers mentioning FLOSS in both the main track and the tools session have been increasing, getting to around 50% in their 2010 editions.

The source code for the data extraction and analysis scripts we used is available together with the sources for this paper itself[8], and is licensed under the GNU General Public License version 3 or any later version.

B. Analysis of research papers mentioning FLOSS

We investigated in which way the FLOSS terms were used in these 25 papers. For that, we classified them in 6 different categories, according to our interpretation of each paper. Each paper was classified in exactly one category, as shown in Table III. The first category comprised three papers that did not mention FLOSS explicitly in their text but just referenced bibliography whose title mentioned FLOSS [28], [29], [30]. The other categories are detailed below.

[8]https://gitorious.org/flosspapers/cbsoft2011-sbes25

1) Example or Comparison: Papers that are not focused on FLOSS but used FLOSS as an example or comparison. **Yamaguti and Price** [31] presented a web-based reflective architecture. They argued that any FLOSS CASE tool is an example of "open object", a concept defined in their paper. **Maciel *et al.*** [32] explained that there are several FLOSS and proprietary MDD/MDA tools with different features that can use their proposed approach for model-driven process. **Araújo and von Staa** [33] describe a tool, called SDiff, that compares documents using their syntactic structure. They compared SDiff to KDiff, which is FLOSS tools also used to analyse documents.

2) Running FLOSS tools: Some of the analyzed papers described the development or use of FLOSS tools to carry out their research. **Santos and Schiel** [34] presented a study about the interoperability of data between different domains on the Internet, applying the Resource Description Framework (RDF). They used PostgreSQL (a FLOSS database) in one of their case studies. **Mendonça *et al.*** [35] among their contributions presented a refactoring framework, called RefaX. They used Jikes, which is an adaptation from IBM's FLOSS Java compiler to implement a Java version of Refax, caled Refax4Java. **Maciel and Yano** [36] proposed a workflow language based Web Services and a FLOSS run-time environment to this language. **Gimenes *et al.*** [37] presented a process for component-based product line for Workflow Management Systems. The implementation of a proposed product line was basead on FLOSS frameworks and tools such as JHotDraw, JacORB, CORORB, MySQL, and ObjectBridge. **Dantas *et al.*** [38] proposed an approach to test multi-threaded systems, using OurGrid, a FLOSS peer-to-peer grid middleware. Also, they developed a FLOSS testing framework called Thread-Control. **Torres *et al.*** [39] proposed the Model Driven Java Persistence API (MD-JPA). A FLOSS plug-in was developed to validate and work with MD-JPA.

3) FLOSS tool as target: Several of the selected papers used FLOSS tools as a target to test another tool, framework, or environment. **Souza and Mendonça** [40] defined a reverse engineering environment to extract and detect "implied scenarios" from traces. They tested the proposed environment against MyPetStore, a customized version of PetStore, which is a FLOSS tool developed by Sun MicroSystems to illustrate programming resources available in the J2EE platform. **Ferrari *et al.*** [41] performed an empirical study to quantify, document, and classify faults uncovered in several releases of three Aspect-Oriented systems. One of these systems was iBATIS, a Java-based FLOSS framework for object-relational data mapping. **Macia *et al.*** [42] also used iBATIS as target in a study on Aspect-Oriented programming. They presented a set of metric-based strategies to detect recurring code smells in existing AO systems. **Dantas and Garcia** [43] analyzed two software products, among them the iBATIS tool, to perform an analysis of the relationship between advanced programming techniques and the trade-off of software reuse and stability.

4) Data from FLOSS: Part of the papers used data from FLOSS projects to test their tools and theories. **Colaço Jr.**

TABLE III
ANALYSIS OF RESEARCH PAPERS MENTIONING FLOSS

Category	2002	2004	2005	2007	2008	2009	2010	Total
Reference	0	0	0	1	0	1	1	3
Example or Comparison	1	0	0	0	0	1	1	3
Running FLOSS tools	1	1	2	0	1	1	0	6
FLOSS tool as target	0	0	0	1	0	0	3	4
Data from FLOSS	0	0	0	0	0	2	3	5
Research about FLOSS	0	0	0	0	0	1	3	4
Total	2	1	2	2	1	6	11	25

et al. [44] analyzed 18 large Brazilian industrial projects trying to use association rules obtained from mining their source code repositories to predict pairs of files that would need to be changed together in the future. They compared their results with other results from a previous similar study with eight FLOSS projects (Eclipse, GCC, Gimp, JBOSS, JEdit, KOffice, Postgres, and Python). They found that, in that industrial environment, the prediction power was even higher than with the FLOSS projects. **Ferreira *et al.*** [45] collected source code metrics from 40 Java FLOSS projects to perform an empirical study and propose reference values for Object-Oriented software metrics. **Carneiro *et al.*** [46] conducted an experimental study to assess a multiple views approach based on concern-driven software visualization resources, arguing that visual views support code smell detection. To perform their study, they analyzed five consecutive versions of a FLOSS project called MobileMedia, an application for photo, music, and video on mobile devices. **Costa and Barros** [47] analyzed eigth FLOSS projects (Azureus, Eclipse ANT, Jena, JMule, JUnit, JVI, Poor Man CMS, and Sweet Home 3D) to perform an experimental study to observe if the adoption of the Common-Closure principle improves a set of software design metrics, based on a proposed technique to organize the classes into packages according to this principle. **Netto *et al.*** [48] used data from Eclipse Bugzilla to test a method based on a genetic algorithm to find the closest optimal bug correction task schedules, according to the relevance of information from bug repositories. As a practical result, they suggested a better schedule to Eclipse IDE developers.

5) Research about FLOSS: Finally, there are papers that focus on FLOSS in itself. **Costa *et al.*** [49] described Transflow, a tool aimed at analyzing data about the co-evolution of the source code and the developers' participation and social interaction in FLOSS projects. **Cavalcanti *et al.*** [50] discussed about the bug report duplication problem that can have a negative influence on software maintenance, in particular, because it increases time spent on report analysis and validation. They conducted a study about that using eight FLOSS projects (Bugzilla, Eclipse Epiphany, Evolution, Firefox, GCC, Thunderbird, and Tomcat). **Terceiro *et al.*** [20] studied how the participation of developers in FLOSS projects can be associated with the structural complexity of the project source code. **Meirelles *et al.*** [18] analyzed source code attributes as predictors for the attractiveness of FLOSS projects, i.e., its ability to attract and retain users and developers, which is crucial for the success of the project.

C. Analysis of tools papers mentioning FLOSS

We analyzed 30 tools papers in which FLOSS terms are mentioned explicitly to identify if they described actual FLOSS tools. We separated them in four different groups according to how these tools papers mentioned FLOSS and whether they explicitly mentioned a distribution license and provided the location of a source code repository. Table IV shows the distribution of these tools papers according to our classification.

Two tools are not FLOSS, but their respective papers cited FLOSS in their bibliography: **DDE** by Garcia *et.al* [51] and **X-CORE** by Oliveira *et.al* [52]. Other four tools are not FLOSS but their authors promised that they would be FLOSS in the future: **ControlPro** by Moro *et.al* [53], **ReqODE** by Martins *et.al* [54], **CRISTA** by Porto *et.al* [55], and **StArt** by Zamboni *et.al* [56]. Another group of four tools are not FLOSS but mentioned FLOSS in their papers because they depend on FLOSS products: **Mudelgen**, by Simão *et.al* [57], depends on Bison and Flex, which are FLOSS compiler development tools. **GAW**, by Mangan *et.al* [58], is a Plug-in for Eclipse IDE. **BAST**, by Cavalcanti *et.al* [59], uses a MySQL dabase. **ModelT2**, by Albuquerque *et.al* [60], is related to the FLOSS UML tool BOUML.

The majority of the papers mentioning FLOSS present their respective tool as FLOSS, but 14 of them did not specify a license or repository. Additionally, their respective websites were either unavailable or did not contain this information. Although the authors had the intention to make their tools available as FLOSS, they did not succeed in doing so. Even in the best cases when the tool is actually available, they cannot be considered FLOSS because they either do not made the source code available or did not specify a license[9] that complies with either the Free Software defition or the Open Source definition. This way **C&L** by Felicíssimo *et.al* [61], **SearchEngine** by Sales *et.al* [62], **Merlin** by Mrack *et.al* [63], **WebAPSEE** by Lima *et.al* [64], **Codipse-Req** by Brito and Vasconcelos [65], **Captor** by Shimabukuro Jr. *et.al* [66], **BSmart** by Gomes *et.al* [67], **Batcave** by Marinho *et.al* [68], **TeTooDS** by Araujo and Delamaro *et.al* [69], **FastInterface** by Oliveira and Lula Jr. *et.al* [70], **RBTTool** by Venâncio *et.al* [71], **Fermine** by Almeida *et.al* [72], **AssistME**

[9]Copyright law in most countries assumes that, unless the author explicitly states otherwise, every work of creation such as software is distributed under "All Rights Reserved" terms, so that third parties are prevented from making derived works or even redistributing the work. That is why, whenever a software product does not specify a distribution license, we must assume it is not FLOSS.

TABLE IV
ANALYSIS OF TOOLS PAPERS MENTIONING FLOSS

Context	2002	2004	2005	2006	2007	2008	2009	2010	Total
Reference	1	0	0	0	1	0	0	0	2
Promise to be FLOSS	0	0	1	1	0	1	0	1	4
use FLOSS	1	1	0	0	0	0	1	1	4
FLOSS – no license and/or no repository	0	1	0	5	1	2	2	3	14
FLOSS	1	0	1	1	0	0	1	2	6
Total	3	2	2	7	2	3	4	7	30

by Queiroz *et.al* [73], **ComSCId & DMAsp** by Parreira Jr. *et.al* [74], cannot be actually be considered as FLOSS tools.

Finally, only six tool papers provided all the information needed to verify that they described actual FLOSS tools. Reis [75] described in details the concepts and features of **Bugzilla**, a well-known FLOSS issue tracking tool. Duarte *et.al* [76] described **GridUnit**, a tool to execute software automated tests in grid environments. It is available under the LGPL 2.1 license and its source code is available in a CVS repository on SourceForge. Paiva *et.al* [77] developed **MVCASE** to support their model for design rationale capture and implementation. It is available under the BSD license and its source code can be accessed from an Subversion repository. Meirelles *et.al* [78] presented **Crab**, the first version of a tool for configuration and interpretation of source code metrics. It was released under the BSD license in a Subversion repository. It was later rewritten, renamed to Kalibro, relicensed under the LGPL license, and made available from from a Git repository. Terceiro *et.al* [22] described **Analizo**, a free, multi-language, extensible source code analysis and visualization toolkit that calculates a fair number of source code metrics, generates dependency graphs, and makes software evolution analysis. Analizo source code can be accessed from a Git repository and was released under the GPL version 3 license. Ferreira *et.al* [79] created **TaRGeT** to reduce the testing costs via a systematic approach that generates test suites from use case specifications. Its source code can be accessed from a Subversion repository and is licensed under the MIT license.

D. Analysis of the Software Engineering Education Forum papers

The Software Engineering Education Forum had no published paper since 2008 and 2010 that addresses the use of FLOSS in SEE. We had a paper accepted for the 2011 edition [80], in which we discuss preliminar results of our experience with the use of FLOSS in SEE.

V. DISCUSSION

The international SE community promotes important conferences in which FLOSS is one of the main topics of interest. For example, the ACM/IEEE International Conference on Software Engineering (ICSE) has a "Software Engineering in Practice" track that explicitly solicits quality research papers addressing new development methods that "depart from traditional software engineering, such as free and open software".

In addition, the International Federation for Information Processing (IFIP) has a dedicated working group (WG 2.13) to promote research on FLOSS. Between 2001 and 2005, IFIP organized the Open Source Software Engineering workshop series at ICSE. Later, IFIP supported the Emerging Trends in FLOSS Research and Development workshop series from 2007 to 2010 at ICSE. Since 2005, IFIP WG 2.13 promotes the International Conference on Open Source Systems (the OSS conferences). OSS'2011 is being held for the first time in Brazil at Salvador, just one week after SBES'2011.

Brazil, on the other hand, has still little tradition in FLOSS-centered research. The only partial exception is the Free Software Workshop at FISL (International Forum on Free Software) that happens every year since 2000 in Porto Alegre, Brazil.

In this study, we analyzed the majority of papers published over the last 10 years in SBES to understand how the interest on FLOSS evolved during this period. Such interest was first noticed in 2002, when two research papers we identified as being related to FLOSS were either mentioning FLOSS superficially or reporting on the usage of a FLOSS tool. Reporting on usage of FLOSS tools was already positive, since the results obtained are easier to reproduce than when private, unavailable tools are used.

Recently, the interest in FLOSS not only grew in the numbers, but also changed in nature. FLOSS moved to a more central role in the SBES community research work. We can observe three significant trends in the last two editions of SBES, which are described below.

First, we have seen a significant increase in the usage of FLOSS project data for validating general Software Engineering hypotheses, techniques, and tools [44], [45], [46], [47], [48]. This benefits Brazilian Software Engineering research by showcasing the possibility of studying data from real, active software development projects.

Second, we have witnessed the growing interest in the study of the development processes used by FLOSS projects [49], [50], [20], [18]. By understanding how FLOSS projects work, which challenges they face and how they succeed (or not succeed), we can improve the general body of knowledge on software development.

Third, we also found out that an increasing number of Brazilian researchers are taking advantage of the opportunities in releasing their tools as FLOSS. Two papers in the main track reported that they released research that produced FLOSS products [38], [39], as well as five papers in the tools session [75], [77], [78], [22], [79]. Not only their research is easier to reproduce, but those researchers can also benefit from outside collaboration in the improvement of their tools.

Unfortunately, another 14 tool paper authors were willing to make their tools available as FLOSS, but, as far as we could verify, they did not succeed on that.

These evolution trends match the opportunities we presented in Sections III-B and III-C. We believe, however, that the Brazilian Software Engineering community could be taking more advantage of these opportunities, and that we are very late in comparison with the international scenario. Nonetheless, we also believe that this is a good start.

With regard to Software Engineering Education, there are innovative experiences in using FLOSS explicitly as an educational tool for software development in both undergraduate and graduate levels in universities such as USP and UFBA[80]. We expect these experiences to become more widely spread in Brazil and their results to be presented in academic forums such as FEES, which has not been the case up to the present date.

VI. Proposed Agenda

Brazil has been an important player in the FLOSS International scenario. GNU/Linux has been used as a platform for research and education in several Brazilian universities since the beginning of the 1990s. Brazil hosts one of the largest and most important FLOSS events in the world: FISL. Finally, Brazilian researchers and developers have developed FLOSS systems such as Lua, Ginga/NCL, and Noosfero, which are used and well known internationally.

However, the use of FLOSS as a significant subject of Software Engineering research in Brazil is still incipient. FLOSS offers hundreds of millions of lines of code to be analyzed by researchers. Hundreds of thousands of FLOSS projects with their tens of thousands of development teams post to the web detailed information about their work and the way they interact. Nevertheless, few researchers in Brazil look at this rich set of information to base their work; meanwhile, many works rely on unrealistic toy prototypes to assess their research results.

With regard to education, university courses throughout the country usually face difficulties such as tight schedules, unbalanced student bodies, and lack of local expertise on the FLOSS ecosystem. As a result, instructors often feel pressured to choose an easier solution: either adopting a more theoretical approach in which software is not developed or focusing on the development of small proof-of-concept systems. The opportunity to learn by having hands-on experience on real-world, existing FLOSS projects is often missed.

To change this situation, we propose an agenda and invite members of the Brazilian Software Engineering community to collaborate on refining, improving, and implementing it. We propose the following actions:

1) Encouraging that software research uses public data and FLOSS tools to process and analyze data. This will promote openness and reproducibility of experimental studies, enabling them to be replicated or refined by other groups, which is a fundamental principle of science.
2) Increase the number of venues welcoming results from FLOSS research. This can be accomplished, for example, by (i) reshaping the FISL Workshop on Free Software to become a venue for publication of research results with original scientific-technological contributions and/or (ii) explicitly mentioning FLOSS as a topic of interest for traditional SE venues such as SBES itself, ESELAW (Experimental Software Engineering Latin American Workshop), and SBQS (Brazilian Symposium on Software Quality).
3) Creating a repository for publishing the FLOSS tools presented in the tools sessions of Brazilian software engineering conferences and workshops. This repository must support long-term storage of files for public downloading as well as documentation about how to publish them as FLOSS properly.
4) Fostering the use of FLOSS in SE Education. This can be achieved by (i) creating a repository of educational material, like MIT's open courseware, focused on FLOSS education, development, and exploitation. This material should be created collaboratively and shared by educators at no cost; (ii) creating a forum where SE educators can discuss and share experiences in the use of FLOSS for SE education.
5) Fostering the creation of FLOSS Competence Centers throughout the country and having them join the international network of FLOSS Competence Centers[10].

The Brazilian Computer Society (SBC) could host a special interest group about FLOSS to coordinate the pursuit of such agenda. Despite being of particular interest to the Software Engineering special commission, FLOSS is also a topic transversal to all Computer Science areas. It would be interesting to have a forum, in the context of SBC, where the several researchers with interest in FLOSS could share experiences and pursue this agenda.

The authors intend to push this agenda forward and invite all interested colleagues to get in touch. To achieve that, we created an online community in the Brazilian Free Software social network[11], where we already started working on items 3 and 4 of the proposed agenda. We can also use this space to discuss how to accomplish the other items.

VII. Future Work

In this work, we performed a literature review on the available digital data (SBES research and tools papers). However, this review was not complete nor systematic.

Therefore, as future work, we plan to conduct a systematic review [81] to establish a more controlled process of conducting the research review, as well as to gather and analyze all existing research and tool papers from all SBES editions, available on the BDBComp and other media as well.

The development of such a systematic approach may bring improvements in precision of the data and the reliability of

[10] http://www.flosscc.org.
[11] http://softwarelivre.org/sbc

the information, and allow other researchers to reproduce the review in different time and contexts, being able to judge the adequacy and modify the chosen criteria and standards.

ACKNOWLEDGMENTS

This research was supported by the Brazilian National Research Council (CNPq), grants 304804/2009-6 and 566164/2008-6, and partially supported by the National Institute of Science and Technology for Software Engineering (INES), CNPq grant 573964/2008-4. The authors are grateful to former SBES Tools program chairs, who provided the papers of the events they have coordinated.

REFERENCES

[1] W. Scacchi, "Free/open source software development: recent research results and emerging opportunities," in *Proceedings of the 6th joint meeting of the European Software Engineering Conf. and the ACM SIGSOFT Int'l Symposium on Foundations of Software Engineering, 2007, Dubrovnik, Croatia, September 3-7, 2007, Companion Papers*. ACM, 2007, pp. 459–468.

[2] A. Aksulu and M. R. Wade, "A comprehensive review and synthesis of open source research," *J. AIS*, vol. 11, no. 11, 2010.

[3] W. Scacchi, "The future of research in free/open source software development," in *Proc. of the FSE/SDP workshop on Future of software engineering research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 315–320.

[4] F. Kon, "O software aberto e a questão social," Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, Tech. Rep. RT-MAC-2001-07, maio 2001. [Online]. Available: http://www.ime.usp.br/~kon/papers/RT-SoftwareAberto.pdf

[5] E. S. Raymond, *The Cathedral & the Bazaar* . Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1999.

[6] C. Jensen and W. Scacchi, "Modeling recruitment and role migration processes in ossd projects," in *In Proceedings of the 6th Int'l Workshop on Software Process Simulation and Modeling*, 2005.

[7] ——, "Role migration and advancement processes in ossd projects: A comparative case study," in *ICSE '07: Proceedings of the 29th Int'l Conf. on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 364–374.

[8] T. Østerlie and L. Jaccheri, "A critical review of software engineering research on open source software development," in *Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design*. Gdansk University Press, 2007, pp. 12–20.

[9] D. L. Parnas, "Software engineering or methods for the multi-person construction of multi-version programs," in *Programming Methodology*, ser. Lecture Notes in Computer Science, C. Hackl, Ed., vol. 23. Springer, 1974, pp. 225–235.

[10] "Computing curricula – software engineering volume (ccse)," 2004, http://sites.computer.org/ccse.

[11] M. J. Wolf, K. Bowyer, D. Gotterbarn, and K. Miller, "Open source software: intellectual challenges to the status quo," in *Proc. of the 33rd SIGCSE technical symposium on Computer Science Education*, ser. SIGCSE '02. New York, NY, USA: ACM, 2002, pp. 317–318.

[12] K. J. O'Hara and J. S. Kay, "Open source software and computer science education," *J. Comput. Small Coll.*, vol. 18, pp. 1–7, February 2003.

[13] D. A. Patterson, "Computer science education in the 21st century," *Commun. ACM*, vol. 49, pp. 27–30, March 2006.

[14] R. K. Raj and F. Kazemian, "Using open source software in computer science courses," in *Frontiers in Education Conf. , 36th Annual*, IEEE. San Diego, CA, USA: IEEE, 2006, pp. 21 – 26.

[15] J. Nandigam, V. N. Gudivada, and A. Hamou-Lhadj, "Learning software engineering principles using open source software," in *Frontiers in Education Conf. , 38th Annual*, IEEE. Saratoga Springs, NY, USA: IEEE, 2008, pp. 18 – 23.

[16] G. W. Hislop, H. J. Ellis, A. B. Tucker, and S. Dexter, "Using open source software to engage students in computer science education (panel)," *SIGCSE Bull.*, vol. 41, pp. 134–135, March 2009.

[17] B. D. Faber, "Educational models and open source: resisting the proprietary university," in *Proc. of the 20th annual Int'l Conf. on Computer documentation*, ser. SIGDOC '02. New York, NY, USA: ACM, 2002, pp. 31–38.

[18] P. Meirelles, C. Santos Jr., J. Miranda, F. Kon, A. Terceiro, and C. Chavez, "A study of the relationships between source code metrics and attractiveness in free software projects," *Software Engineering, Brazilian Symposium on*, vol. 0, pp. 11–20, 2010.

[19] C. D. S. Jr., M. B. Cavalca, F. Kon, J. Singer, V. Ritter, D. Regina, and T. Tsujimoto, "Intellectual property policy and attractiveness: a longitudinal study of free and open source software projects," in *CSCW*, 2011, pp. 705–708.

[20] A. Terceiro, L. R. Rios, and C. Chavez, "An empirical study on the structural complexity introduced by core and peripheral developers in free software projects," *Brazilian Symposium on Software Engineering*, vol. 0, pp. 21–29, sep 2010.

[21] I. Herraiz, D. I. Cortazar, and F. R. Hernández, "FLOSSMetrics: Free/Libre/open source software metrics," *Software Maintenance and Reengineering, European Conf. on*, vol. 0, pp. 281–284, 2009.

[22] A. Terceiro, J. Costa, J. a. Miranda, P. Meirelles, L. R. Rios, L. Almeida, C. Chavez, and F. Kon, "Analizo: an extensible multi-language source code analysis and visualization toolkit," in *CBSOFT-Ferramentas*, 2010.

[23] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 3, pp. 309–346, 2002.

[24] K. Crowston, K. Wei, Q. Li, and J. Howison, "Core and periphery in free/libre and open source software team communications," *Hawaii Int'l Conf. on System Sciences*, vol. 6, p. 118, 2006.

[25] H. Masmoudi, M. d. Besten, C. d. Loupy, and J.-M. Dalle, ""peeling the onion": The words and actions that distinguish core from periphery in bug reports and how core and periphery interact together." in *5th IFIP WG 2.13 Int'l Conf. on Open Source Systems*, vol. 299. Springer, 2009, pp. 284–297.

[26] M. J. Scialdone, N. Li, R. Heckman, and K. Crowston, "Group maintenance behaviors of core and peripherial members of free/libre open source software teams," in *5th IFIP WG 2.13 Int'l Conf. on Open Source Systems*, vol. 299. Springer, 2009, pp. 298–309.

[27] J. Costa, F. Santana, and C. Souza, "Understanding open source developers' evolution using transflow," in *Groupware: Design, Implementation, and Use, 15th Int'l Workshop, CRIWG 2009, Peso da Régua, Douro, Portugal, September 13-17, 2009. Proceedings*, 2009, pp. 65–78.

[28] J. P. F. de Oliveira, T. Brito, S. R. Jr, and G. Elias, "Um Serviço de Repositório Compartilhado e Distribuído para Suporte ao Desenvolvimento Baseado em Componentes," in *XXI SBES*, 2007.

[29] J. a. G. Prudêncio, L. Murta, and C. Werner, "On the selection of concurrency control policies for configuration management," in *Proc. of the 2009 XXIII Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2009, pp. 155–164.

[30] L. L. Silva, K. R. Paixao, S. d. Amo, and M. d. A. Maia, "Software evolution aided by execution trace alignment," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 158–167.

[31] M. H. Yamaguti and R. T. Price, "Uma Arquitetura Reflexiva Baseada na Web para Ambiente de Suporte a Processo," in *XVI SBES*, 2002.

[32] R. S. P. Maciel, B. C. d. Silva, A. P. F. Magalhães, and N. S. Rosa, "An integrated approach for model driven process modeling and enactment," in *Proc. of the 2009 XXIII Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2009, pp. 104–114.

[33] T. de Araújo and A. von Staa, "Sdiff: A comparison tool based in syntactical document structure," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*, 27 2010-oct. 1 2010, pp. 30 –39.

[34] D. S. A. Santos and U. Schiel, "RDF na Interoperabilidade entre Domínios na Web," in *XVI SBES*, 2002.

[35] N. Mendonça, P. H. M. Maia, L. A. Fonseca, and R. M. C. Andrade., "Building Flexible Refactoring Tools with XML," in *XVII SBES*, 2004.

[36] L. A. H. da S. Maciel and E. T. Yano, "Uma Linguagem de Workflow Para Composição de Web Services - LCWS," in *XIX SBES*, 2005.

[37] I. M. de Souza Gimenes, R. Nishimura, E. A. de Oliveira Junior, F. R. Lazilha, U. Kulesza, and C. J. P. Lucena, "A Component-based Product Development Process for a Workflow Management System Product Line," in *XIX SBES*, 2005.

[38] A. Dantas, M. Gaudencio, F. Brasileiro, and W. Cirne, "Obtaining

Trustworthy Test Results in Multi-threaded Systems," in *XXII SBES*, 2008.

[39] A. Torres, R. Galante, and M. S. Pimenta, "Towards a uml profile for model-driven object-relational mapping," in *Proc. of the 2009 XXIII Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2009, pp. 94–103.

[40] F. Souza and N. Mendonça, "Um Ambiente para Detecção de Cenários Implícitos a partir de Rastros de Execução," in *XXI SBES*, 2007.

[41] F. C. Ferrari, R. Burrows, O. A. L. Lemos, A. Garcia, and J. C. Maldonado, "Characterising faults in aspect-oriented programs: Towards filling the gap between theory and practice," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 50–59.

[42] I. Macia, A. Garcia, and A. v. Staa, "Defining and applying detection strategies for aspect-oriented code smells," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 60–69.

[43] F. Dantas and A. Garcia, "Software reuse versus stability: Evaluating advanced programming techniques," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 40–49.

[44] M. C. Júnior, M. Mendonça, and F. Rodrigues, "Mining software change history in an industrial environment," in *Proc. of the 2009 XXIII Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2009, pp. 54–61.

[45] K. A. M. Ferreira, M. A. S. Bigonha, R. S. Bigonha, H. C. Almeida, and L. F. O. Mendes, "Reference values for object-oriented software metrics," in *Proc. of the 2009 XXIII Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2009, pp. 62–72.

[46] G. d. F. Carneiro, M. Silva, L. Mara, E. Figueiredo, C. Sant'Anna, A. Garcia, and M. Mendonca, "Identifying code smells with multiple concern views," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 128–137.

[47] M. d. F. Costa and M. d. O. Barros, "Evaluating the implications of a package design principle upon software maintainability," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 138–147.

[48] F. Netto, M. O. Barros, and A. C. F. Alvim, "An automated approach for scheduling bug fix tasks," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 80–89.

[49] J. M. R. Costa, F. W. Santana, and C. Souza, "Using transflow to analyze open source developers' evolution," in *Proc. of the 2009 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2009, pp. 165–175.

[50] Y. Cavalcanti, P. A. S. Neto, E. Almeida, D. Lucredio, C. Cunha, and S. Meira, "One step more to understand the bug report duplication problem," in *Proc. of the 2010 Brazilian Symposium on Software Engineering*. IEEE Computer Society, 2010, pp. 148–157.

[51] V. C. Garcia, V. Fontanette, A. B. Perez, A. A. Bossonaro, and A. F. Prado, "DDE - draco domain editor," in *IX Sessão de Ferramentas – XVI SBES*, 2002, pp. 378–383.

[52] J. Oliveira, T. Brito, A. Oliveira, S. Rabelo, and G. Elias, "X-CORE: Um Serviço de Repositório Compartilhado e Distribuído de Componentes de Software," in *XIV Sessão de Ferramentas – XXI SBES*, 2007.

[53] R. D. Moro, J. C. Nardi, and R. de Almeida Falbo, "ControlPro: Uma Ferramenta de Acompanhamento de Projetos Integrada a um Ambiente de Desenvolvimento de Software," in *XI Sessão de Ferramentas – XIX SBES*, 2005.

[54] A. F. Martins, J. C. Nardi, and R. de Almeida Falbo, "ReqODE: Uma Ferramenta de Apoio à Engenharia de Requisitos Integrada ao Ambiente ODE," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[55] D. de Paula Porto, M. Mendonça, and S. Fabbri, "CRISTA - Code Reading Implemented with Stepwise Abstraction," in *XV Sessão de Ferramentas – XXII SBES*, 2008.

[56] A. Zamboni, A. D. Thommazo, E. Hernandes, and S. Fabbri, "StArt – Uma Ferramenta Computacional de Apoio à Revisão Sistemática," in *CBSOFT-Ferramentas*, 2010.

[57] A. da Silva Simão, A. M. R. Vincenzi, and J. C. Maldonado, "mudelgen: A Tool for Processing Mutant Operator Descriptions," in *IX Sessão de Ferramentas – XVI SBES*, 2002.

[58] M. A. S. Mangan, I. A. da Silva, and C. M. L. Werner, "GAW: uma Ferramenta de Percepção de Grupo Aplicada no Desenvolvimento de Software," in *XI Sessão de Ferramentas – XVIII SBES*, 2004.

[59] Y. a. C. Cavalcanti, C. E. A. da Cunha, E. S. de Almeida, and S. R. de Lemos Meira, "BAST: A Bug Report Analysis and Search Tool," in *XVI Sessão de Ferramentas – XXIII SBES*, 2009.

[60] P. Albuquerque, J. L. Massollar, and G. H. Travassos, "ModelT2: Apoio Ferramental à Geração de Casos de Testes Funcionais a partir de Casos de Uso," in *CBSOFT-Ferramentas*, 2010.

[61] C. H. Felicíssimo, J. C. S. do Prado Leite, K. K. Breitman, and L. F. da Silva, "C&L: Um Ambiente para Edição e Visualização de Cenários e Léxicos," in *XI Sessão de Ferramentas – XVIII SBES*, 2004.

[62] E. de O. Sales, S. F. de Freitas, and R. Q. Reis, "Uma Ferramenta para Recuperação de Modelos de Processo de Software Reutilizáveis," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[63] M. Mrack, Álvaro de Freitas Moreira, and M. Pimenta, "Merlin: Interfaces CRUD Em Tempo de Execução," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[64] A. Lima, A. Costa, B. França, C. A. L. Reis, and R. Q. Reis, "Gerência Flexível de Processos de Software com o Ambiente WebAPSEE," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[65] R. A. Brito and A. M. L. de Vasconcelos, "Definiçãoo e Implementação de um Modelo de Rastreamento para Engenharia de Requisitos Distribuída," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[66] E. S. Junior, P. Masiero, and R. Braga, "Captor: Um Gerador de Aplicações Configurável," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[67] B. E. G. Gomes, A. M. Moreira, D. B. P. Déharbe, and K. K. de O. Moraes, "A Ferramenta BSmart para o Desenvolvimento Rigoroso de Aplicações Java Card com o Método Formal B," in *XIV Sessão de Ferramentas – XXI SBES*, 2007.

[68] E. Marinho, V. Medeiros, D. Déharbe, B. Gomes, and C. Tavares, "A Ferramenta Batcave para a Verificaçãao de Especificações Formais na Notaçãao B," in *XIV Sessão de Ferramentas – XXI SBES*, 2007.

[69] R. F. Araujo and M. E. Delamaro, "TeTooDS - Testing Tool for Dynamic Systems," in *XV Sessão de Ferramentas – XXII SBES*, 2008.

[70] K. M. A. de Oliveira and B. L. Jr., "Desenvolvimento Evolutivo de Interfaces com o Usuário em uma Abordagem Baseada em Modelos e Múltipla Prototipagem: FastInterface," in *XVI Sessão de Ferramentas – XXIII SBES*, 2009.

[71] J. Venâncio, C. Gusmão, E. Mendes, and E. Souza, "RBTTool – Uma Ferramenta de Apoio à Abordagem de Teste de Software baseado em Riscos," in *XVI Sessão de Ferramentas – XXIII SBES*, 2009.

[72] G. T. de Almeida, B. A. Ramos, M. M. F. Neto, M. S. Reis, and M. R. dos S. Barcelos Aline P. V. de Vasconcelos, "Ferramenta de Apoio à Engenharia de Requisitos Integrada a um Ambiente Colaborativo de C0digo Aberto," in *CBSOFT-Ferramentas*, 2010.

[73] C. Queiroz, F. Castor, and N. Cacho, "AssistME – uma Ferramenta para Auxiliar a Refatoração para Aspectos de Tratamento de Exceccões," in *CBSOFT-Ferramentas*, 2010.

[74] P. A. P. Jr., H. A. X. Costa, V. V. de Camargo, and R. A. D. Penteado, "ComSCId & DMAsp: Identificação de Interesses Transversais e Recuperação de Modelos de Classes Anotados a partir Aplicações OO Java," in *CBSOFT-Ferramentas*, 2010.

[75] C. Reis, "Uma Visão Geral do Bugzilla, uma Ferramenta de Acompanhamento de Alterações," in *IX Sessão de Ferramentas – XVI SBES*, 2002.

[76] A. N. Duarte, W. Cirne, F. Brasileiro, and P. D. de Lima Machado, "GridUnit: Using the Computational Grid to Speed up Software Testing," in *XI Sessão de Ferramentas – XIX SBES*, 2005.

[77] D. M. B. Paiva, D. Lucrédia, and R. P. de Mattos Fortes, "MVCASE - Incluindo Design Rationale para Auxílio a Modelagem em Projetos de Pesquisa," in *XIII Sessão de Ferramentas – XX SBES*, 2006.

[78] P. R. M. Meirelles, R. Cóbe, S. Hanazumi, P. Nunes, G. Challco, S. Martins, E. Morais, and F. Kon, "Crab: Uma Ferramenta de Configuração e Interpretação de Métricas de Software para Avaliação de Qualidade de Código," in *XVI Sessão de Ferramentas – XXIII SBES*, 2009.

[79] F. Ferreira, L. Neves, M. Silva, and P. Borba, "TaRGeT: a Model Based Product Line Testing Tool," in *CBSOFT-Ferramentas*, 2010.

[80] C. Chavez, A. Terceiro, P. Meirelles, F. Kon, and C. S. Jr., "Using free/libre/open source for software engineering education," in *CBSoft 2011 - SBES - FEES*, mar 2011.

[81] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, August 2002.