The Symposium on Search-Based Software Engineering: Past, Present and Future

Thelma Elita Colanzi[a,*], Wesley K. G. Assunção[b], Silvia R. Vergilio[c], Paulo Roberto Farah[c,d], Giovani Guizzo[e]

[a] *DIN - State University of Maringa (UEM), Maringa, Brazil*
[b] *COTSI, Federal University of Technology - Paraná (UTFPR), Toledo, Brazil*
[c] *DINF, Federal University of Paraná (UFPR), Curitiba, Brazil*
[d] *DESO, Santa Catarina State University (UDESC), Ibirama, Brazil*
[e] *CREST Centre - Department of Computer Science - University College London (UCL), London, UK*

Abstract

Context: Search-Based Software Engineering (SBSE) is the research field where Software Engineering (SE) problems are modelled as search problems to be solved by search-based techniques. The Symposium on Search Based Software Engineering (SSBSE) is the premier event on SBSE, which had its 11^{th} edition in 2019.
Objective: In order to better understand the characteristics and evolution of papers published at SSBSE, this work reports results from a mapping study targeting the proceedings of all SSBSE editions. Despite the existing mapping studies on SBSE, our contribution in this work is to provide information to researchers and practitioners willing to enter the SBSE field, being a source of information to strengthen the symposium, guide new studies, and motivate new collaboration among research groups.
Method: A systematic mapping study was conducted with a set of four research questions, in which 134 studies published in all editions of SSBSE, dated from 2009 to 2019, were evaluated. In a fifth question, 32 papers published in the challenge track were summarised.
Results: Throughout the years, 290 authors from 25 countries have contributed to the main track of the symposium, with the collaboration of at least two institutions in 46.3% of the papers. SSBSE papers have got substantial external visibility, as most citations are from different venues. The SE tasks addressed by SSBSE are mostly related to software testing, software debugging, software design, and maintenance. Evolutionary algorithms are present in 75% of the papers, being the most common search technique. The evaluation of the SBSE approaches usually includes industrial systems.
Conclusions: SSBSE has helped increase the popularity of SBSE in the SE research community and has played an important role on making SBSE mature. There are still problems and challenges to be addressed in the SBSE field, which can be tackled by SSBSE authors in further studies.

Keywords: Systematic mapping; Search-Based Software Engineering; Bibliometric analysis.

1. Introduction

Search Based Software Engineering (SBSE) is the research field that formulates Software Engineering (SE) problems as search problems. In this way, heuristic techniques are used to find near-optimal solutions to efficiently solve a large variety of problems associated to different SE tasks. The International Symposium on Search Based Software Engineering (SSBSE)[1] is the premier event on SBSE, with eleven editions so far. Over the past ten years, the symposium has drawn attention of researchers, academics, and practitioners alike, contributing to strengthen the field and to integrate the SBSE community, whilst gathering a large body of studies that are now relevant SBSE references.

Thanks to SSBSE we can say that the SBSE field is a mature and active field of research. In the literature we can find surveys reporting a wide view of SBSE on software bug fixing, project management, planning and cost estimation, software comprehension, refactoring, software slicing, service-oriented software engineering, compiler optimisation, quality assessment, and others [1–3]. Such surveys analyse the most used search-based algorithms and also point out research directions on SBSE. de Freitas *et al*. [4] presented a bibliometric analysis of the SBSE field, which, alongside the other surveys, show a growing number of SBSE papers, and an increasing number of addressed SE areas. Some of these areas have their own surveys, e.g., software testing [5], software design [6], requirements [7, 8], maintenance [9], and refactoring [10].

Despite the existing studies on SBSE, none offers a big picture of the SSBSE. Our previous work [11] synthesised the symposium's ten-year history in a systematic mapping [12] conducted over all the SSBSE proceedings. This previous work also analysed the addressed SE tasks and used Computational Intelligence (CI) techniques, similarly to aforementioned SBSE studies. But, differently from related work, it also provided

*Corresponding author
Email addresses: thelma@din.uem.br (Thelma Elita Colanzi), wesleyk@utfpr.edu.br (Wesley K. G. Assunção), silvia@inf.ufpr.br (Silvia R. Vergilio), paulo.farah@udesc.br (Paulo Roberto Farah), g.guizzo@ucl.ac.uk (Giovani Guizzo)

[1] https://ssbse.info/

additional analysis regarding the composition of steering and program committees, submission tracks, paper acceptance rate, and impact of the papers published. Such analysis allowed an overview of SSBSE and contributed to comprehend how the symposium has been evolving along the years.

In this paper, we conduct a deeper and updated analysis of our findings, add an analysis of the papers published in the Challenge Track, include a discussion about trends, and shed light upon the future of the SBSE field. Thereby, this paper presents findings regarding the past, present and future of the SSBSE. The main contributions are: i) to ascertain the impact and relevance of SSBSE, by reporting its main numbers and performing a citation analysis of the published works; ii) to devise a co-authorship network and depict the most prolific research groups, as well as the participation of the industry; iii) to point out the software engineering areas that have been most subjected to investigation as well as the ones that need more attention; iv) to identify the main CI techniques; v) to analyse how SBSE approaches have been evaluated; vi) to provide some recommendations to future SSBSE authors; and vii) to present trends and challenges that constitute research opportunities.

In our mapping, we followed the guidelines of Petersen *et al*. [12]. We established research questions to reflect our goals, adopted inclusion and exclusion criteria, and used a classification schema to categorise the studies and perform our analysis. Furthermore, we made available the raw data for future research and analysis.[2]

Studies like ours are important to corroborate the importance of the symposium and if it has been following up the main changes pointed out by the existing surveys and mappings of the SBSE field, as well as to evaluate its representativeness. As a result, we can observe that the symposium has followed the main changes in the field, playing an important role of contributing to SBSE's maturity.

The remainder of this paper is organised as follows. Section 2 reviews related work. In Section 3 we describe the adopted method: Research Questions (RQs), inclusion and exclusion criteria, how we collected the data, classification schema and some threats to validity. In Section 4 we present the results to answer each posed RQ and describe recommendations to new authors. Section 5 discusses trends, challenges and research opportunities for the SBSE field and symposium. Section 6 concludes the paper and points future directions based on our findings.

2. Related Work

The works that are most related to ours are surveys, systematic mappings, and systematic reviews in the SBSE literature. We have searched for papers on the ACM Digital Library, IEEE Xplore, Springer, Elsevier, and Scopus repositories using the following query: ("Search Based Software Engineering" OR "Search-based Software Engineering") AND ("Survey" OR "Systematic Mapping" OR "Systematic Review"). After collecting the results and removing duplicates, we manually filtered papers that did not fit in our scope.

Most papers found are focused on a particular task, area, or technique in SBSE. The latest major survey that covered the whole area was published in 2012 by Harman *et al*. [2]. We also found multiple papers on specific areas of SBSE, such as Search Based Software Testing (SBST) [3, 13, 14], Search Based Software Design (SBSD) [6, 15], Search Based Software Refactoring (SBSR) [10, 16], Search Based Software Project Management [17], and SBSE for Software Product Lines (SPL) [18–20].

Some other literature reviews focus on specific tasks, such as mining repositories using SBSE [21], test data generation [5, 22], genetic improvement of software [23], software requirements management [8, 24, 25], re-engineering of legacy applications into SPL [26], software module clustering [27], software project scheduling [28], cloud migration [29], and test resource allocation [30]. Specific techniques are also target of SBSE surveys, such as multi- and many-objective optimisation [31–33], hyper-heuristics [34], and interactive algorithms [35, 36].

Differently from the papers mentioned in this section, this paper is focused on mapping only the SBSE papers published on SSBSE. In our previous work [11], we addressed the development of SSBSE, but focusing mainly on the quantitative aspects of the symposium. In this extended and updated version, we present a deeper analysis of the results and also provide insights on trends and future challenges identified while reading the papers, whilst also drawing connections to SBSE as a whole.

3. Systematic Mapping Method

We adopt the systematic mapping process defined by Petersen *et al*. [12]. The process includes the following activities: i) definition of research questions; ii) conducting the search; iii) study selection; and iv) data extraction and classification. In this section we describe how these activities were conducted.

3.1. Research Questions

The main research goal of this work is to capture the SBSE field considering the papers published in the SSBSE editions. In order to reach such goal, we formulated the Research Questions (RQs) described as follows.

RQ1: What are the basic SSBSE numbers? To answer this RQ, we provide a quantitative analysis of the event: number of submitted and published papers along the years, acceptance rate, authors and committees characteristics, research groups, and collaborations.

RQ2: What is the external impact of SSBSE? To answer this RQ, we provide a citation analysis of the SSBSE papers, in order to evaluate the visibility and importance of publishing in the symposium. After 10 years of SSBSE, it would be interesting to discover whether SSBSE papers are cited by other

[2] https://doi.org/10.5522/04/12554366

authors in different venues. By analysing the number of citations, number of self-citations, and number of citations in other venues, we can evaluate if SSBSE papers are only cited by other SSBSE authors or if they are drawing attention from other researchers.

RQ3: What are the most common addressed SE areas and CI techniques? To answer this question, we provide a quantitative analysis of the addressed SE areas, employed CI techniques, and number of papers in each category. Besides, we analyse possible changes and trends over time. We also asked sub-questions such as: How have these areas been evolving? and What are the preferred CI techniques and how has this preference changed over time? Historically, software verification and validation is the most investigated area in SBSE, but which other SE areas have been also investigated? Moreover, we are interested in identifying what CI techniques have been employed nowadays in addition to evolutionary techniques (most common ones in SBSE [2]).

RQ4: How have the SBSE approaches been evaluated? To answer this question we provide an analysis of the experimental evaluation carried out in the papers, identifying applied statistical tests, subjects, type of case studies, and comparison measures. The main idea is to analyse experimental rigour employed in the studies published in the symposium and if it has changed over time.

RQ5: What are the main characteristics of the papers published in the SSBSE Challenge Track? To answer this RQ, we analyse which real-world software were used in the papers published in the referred track, which SE tasks were addressed, and which CI techniques were applied. It is also interesting to analyse whether the most addressed SE areas and the most applied CI techniques in this track are the same of the research track one, analysed in RQ3.

3.2. Conducting the search and study selection

As the scope of our mapping is the SSBSE, the proceedings of the eleven editions (2009-2019) are the source of studies. We consider the following inclusion/exclusion criteria.

- **Inclusion criteria**: studies published in the main track (full research paper) and in the challenge track of any edition of SSBSE.
- **Exclusion criteria**: studies not published in the main and challenge tracks, keynotes, posters or tutorials were not included.

3.3. Data extraction and classification

In order to answer the research questions we quantitatively analysed the extracted data. Regarding data collection, the abstract, introduction and conclusion paper sections were read in full, whereas the rest of the papers' content were read diagonally when necessary. The following pieces of information were extracted: publication year, authors, affiliations (universities and/or companies), city/country, software engineering task, used algorithms/techniques, statistical tests, quality indicators, subject systems used for evaluation, artefacts and tools involved in the experimentation, and other information needed to categorise the papers.

The papers of the research and challenge tracks were classified into categories in two dimensions: Software Engineering area and Computational Intelligence technique. For the first dimension, SE areas, we use the four first levels of the 2012 ACM Computing Classification System (CCS).[3] Table 1 contains the categories of the ACM CCS that have papers published in the SSBSE editions. With respect to the second dimension, CI techniques, five main categories were identified: Evolutionary Algorithm, Local Search, Swarm Intelligence, Machine Learning, and Others (techniques with lower number of papers).

The extracted data were used to answer the research questions. Furthermore, information about submissions and committees were extracted from the SSBSE proceedings in order to answer RQ1. To answer RQ2, the citations of the research track papers were extracted from Google Scholar as explained in Section 4.2.

3.4. Threats to validity

The construct validity is concerned to the research questions, mapping process, and metrics used to assess the results. To minimise this kind of threat, we had several discussions about the questions and goals of our search, and finally decided to standardise the mapping process. For this, we applied the guidelines proposed by Petersen *et al*. [12], while also using common mapping goals and metrics applied by different research studies in the literature [2, 3, 10] (e.g., citation count, authors analysis, sub-areas, and tasks classification). Another construct threat is related to the use of only full papers of the main track to answer RQ2 to RQ4, excluding short and student papers, for example. We took such decision because the other tracks of the symposium varied widely along the editions (Section 4.1.2), which could make the analysis biased towards editions with fewer tracks. Furthermore, papers other than full papers usually have a very narrow contribution and do not include sufficient information for a thorough analysis. Nevertheless, the challenge track has been occurring in all the last editions (since 2013) and is subject of a RQ apart (RQ5).

An external threat is the classification process for the Software Engineering areas. In order to mitigate this threat, we adopted the 2012 ACM Computing Classification System to classify the papers and avoided the creation of arbitrary subareas, hence the classification can be easily generalised and replicated by other authors. We could not identify the gender of two authors, thus we used Genderize.io API and both were defined as females.

The papers classification is tricky and very often subjective to the person reading the paper, thus this constitutes the major internal threat of our work. In order to minimise such a threat, the corpus of papers was divided equally to four authors, which read and classified the papers. After this initial process,

[3] `https://www.acm.org/publications/class-2012`

Table 1: SE Areas Classification according to ACM Computing Classification System (https://www.acm.org/publications/class-2012)

CCS - level 0	CCS - level 1	CCS - level 2	CCS - level 3
General and Reference	Cross-computing tools and techniques	Experimentation	Experimentation
	Document Types	Surveys and Overview	Survey
Social and professional topics	Professional topics	Management of computing and information systems	Project and people management
Software and its engineering	Software creation and management	Designing software	Requirements analysis
			Software design
		Search-based software engineering	SBSE
		Software development techniques	Automatic programming
		Software post-development issues	Maintaining software
			Software evolution
			Software reverse engineering
			Other
		Software verification and validation	Software defect analysis
	Software organization and properties	Extra-functional properties	Software performance
			Software reliability

the other author validated the classification by rechecking all papers.

4. Results and Analysis

In this section, each research question is answered as an outcome of the data extraction and analysis. The classification schema is used and different discussions are presented to analyse the obtained results.

4.1. RQ1 – SSBSE in numbers

The first edition of SSBSE occurred in 2009, in Windsor, United Kingdom (UK). Since then, the symposium took place in seven different countries. Nine editions occurred in Europe, one in South America (2014), and one in North America (2016). We observe that no edition has occurred in Asia or Oceania. Five editions were co-located with the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), and a few were co-located with other events, such as the International Conference on Software Maintenance and Evolution (ICSME) and the IEEE/ACM International Conference on Automated Software Engineering (ASE).

The symposium attracts researchers, students, lecturers, and members from industry. Each SSBSE edition had at least two keynotes, one from SE and other from the optimisation field, in a total of 27 keynotes, as well as 14 tutorials and 4 panels.

4.1.1. Committee characteristics

We observe the existence of distinct committees along the editions. The program committee is the only one present in all editions. The steering committee was created in 2012 (third edition). In the last editions (2017–2019) independent committees were created for the challenge, short and student papers tracks.

Regarding the program committee composition, the number of committee members varies from a minimum of 23 in 2017 to a maximum of 43 in 2014 with a high churn. But we do not observe a great variation in the number of represented countries along the editions, with a minimum of 9 (2011), a maximum of 13 (2014), and an average of 11.2. Some countries are represented in almost all editions such as UK, USA, Italy, Brazil, Spain, Ireland, Norway, Sweden, and Luxembourg. Countries that have appeared more recently are Romania and Korea.

A great variation and significant gender imbalance are observed when we consider the percentage of women in the committee.[4] This percentage varies from a minimum of 5% (2009) to a maximum of 29% (2019). The gender imbalance has been decreasing in the last years. This can be better visualised in Figure 1.(a). Regarding the steering committee (Figure 1.(b)) such imbalance has also been decreasing. In the first four editions it had only 1 woman in a total of 9 members (percentage of 11%). The maximum percentage of women is 33% in 2016 (3 out of 10 members). The number of countries represented in such a committee has been kept almost constant (around 5, with a maximum number of 8 distinct countries in 2012).

In spite of this gender imbalance, the percentage of women researchers in leadership positions in SSBSE is greater regarding other conferences and the Computer Science area [37]. We had a total of 50 chairs, 16 of which are women (32%). If we consider only the main track, this percentage is around 47% (7 out of 15). This imbalance has been decreasing in the last five years. Considering the main track, we observe a perfect balance since 2014; a woman and a man have been chosen for chairs since then.

4.1.2. Number of submissions and acceptance rate

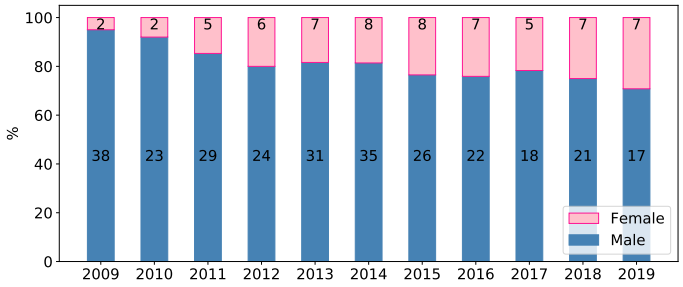
SSBSE has provided different tracks in its eleven editions. Some statistics about such tracks are presented in Table 2. The

[4] We manually checked the gender of committee members and authors by doing a web search in their profiles (e.g., Google Scholar, Microsoft Academic, Research Gate, and Linkedin).

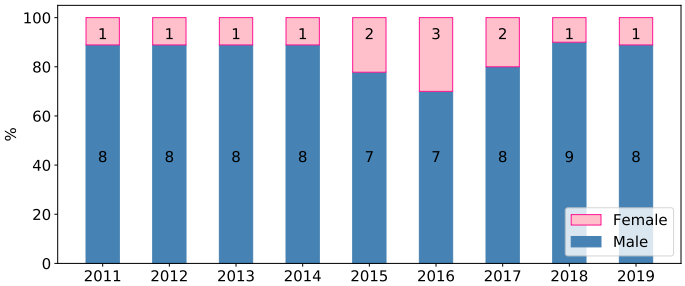
Table 2: SSBSE in Numbers. (COU: number of different countries submitting papers. TSUB: number of submissions including all tracks. SUB: number of submissions. ACC: number of accepted papers. Rate: percentage of acceptance. "-" means unknown or 0.)

Year	COU	TSUB	Full			Short/F.Abstract			Student			Challenge		
			SUB	ACC	Rate	SUB	ACC	Rate	SUB	ACC	Rate	SUB	ACC	Rate
2009	14	26	-	9		-	5	-	-	3	-	-	-	-
2010	-	36	-	14		-	-	-	-	3	-	-	-	-
2011	21	43	37	15	40.5	-	8	-	6	3	50	-	-	-
2012	20	38	34	15	44.1	-	3	-	4	2	50	-	-	-
2013	24	50	39	14	35.9	-	6	-	9	6	66.6	4	2	50
2014	19	51	32	14	43.7	3	1	33.3	8	3	37.1	8	4	50
2015	15	51	26	12	46.1	8	4	50	4	2	50	13	13	100
2016	20	48	25	13	52	9	4	44.4	7	4	57.1	7	7	100
2017	14	32	26	7	26.9	2	5	-	2	2	100	4	4	100
2018	10	21	12	12	100	8*	6	75	-	-	-	1	1	100
2019	17	28	16	9	56.25	9⋄	3	33.3	9⋄	3	33.3	1	1	100

* with Hot of the Press Track
⋄ just one track for short and student papers

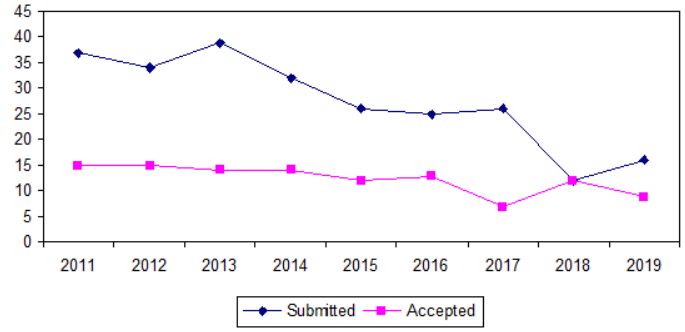


(a) Program Committee

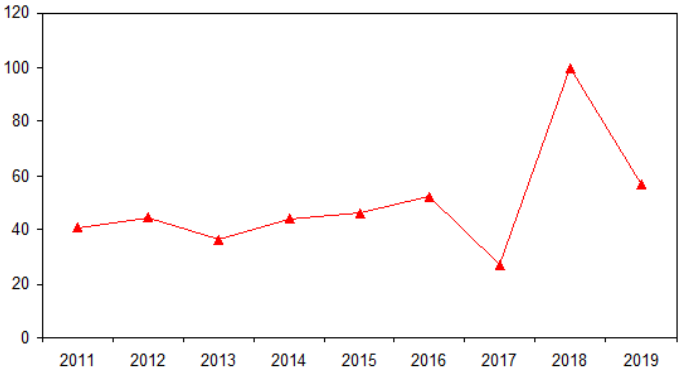


(b) Steering Committee

Figure 1: Committee Characteristics - Gender Imbalance



(a) Number of submitted and accepted papers



(b) Acceptance rate

Figure 2: Submissions in the track of full research papers

main track of full research papers and the student track occurred in all editions with independent chairs. We can see that the total number of submitted papers considering all tracks is greater in the period of 2013–2016. A similar fact can be observed considering the number of submitted papers for the full research papers and the student track.

The number of submitted and accepted papers of the full research papers, as well as, the acceptance rate can be better visualised in Figure 2. The years 2009 and 2010 were not included because we could not find the number of full papers submitted.

Regarding the main track of full research papers, the number of accepted papers varies from 7 (in 2017) to 15 (in 2011 and 2012). The acceptance rate of the main track falls in the range of 27% (2017) to 100% in (2018). These years of 2018 and 2017 are outliers. In 2018, a shepherding phase was added in the reviewing process, which may justify 100% of acceptance. In fact, we do not observe great variations in the acceptance rate before 2016, considering 2011–2016 the mean rate is 43%. After a period of growing and boom, we observed a decrease in the number of submitted papers, what might be justified by the recent inclusion of SBSE in the list of topics of several conferences.

The characteristics of the short papers track varied along the editions. The challenge track started only in 2013. Thus it is not possible to analyse the acceptance rate over the eleven years of both tracks.

In most editions, separated calls for short papers or fast abstracts were provided, with or without independent chairs. In some editions accepted short papers were originally submitted

Table 3: Author's churn

Year	New	Continued	Discontinued	Total	Churn
2009	24	0	0	24	0.0%
2010	32	6	18	38	133.3%
2011	34	8	30	42	89.4%
2012	34	16	26	50	80.9%
2013	34	8	42	42	68.0%
2014	39	5	37	44	92.8%
2015	35	4	40	39	79.5%
2016	39	6	33	45	100.0%
2017	13	3	42	16	28.9%
2018	31	4	12	35	193.7%
2019	25	1	25	26	71.4%

as full papers (2011–2013). In 2017, we had an independent call for short papers, but some papers submitted to the main track were also accepted as short papers. This is because the number of accepted short papers is greater than the submitted ones. The edition of 2017 included a journal-first papers track with 2 papers. The edition of 2018 included a Hot of the Press track that also included short/student papers with 6 papers. Other point that should be highlighted is that in 2019, just one track occurred including short and student papers.

Including all tracks, we had a corpus of 233 papers, 134 associated to the research track, published by IEEE in the first two editions, and by Springer since the third one. Because of this variety in the tracks (as mentioned in Section 3.4), the analysis conducted to answer RQ2 to RQ4 includes only the 134 full papers of the research track collected from all the SSBSE proceedings. The papers of the challenge track have special characteristics and are subject of RQ5.

4.1.3. Authorship

In the 134 full papers, we found 290 distinct authors. For those authors, we analysed their affiliations, and identified the most prolific authors and collaborations. The great majority of authors, equivalent to 94.8% (275 out 290), published fewer than 4 papers.

Table 3 quantifies unique authors who are new or returned to publish at SSBSE with at least one year without publication (column New), authors that published in the year before and remained publishing at least one paper in the event (column Continued), and authors who discontinued, i.e., that had an accepted paper in the previous year and did not publish any paper in the following year (Discontinued). Additionally, we calculated the yearly churn percentage, presented in the last column. This metric represents the authors' turnover. It is the percentage of new authors of a given year in relation to the number of authors of the previous year. The results indicate that few authors keep publishing along the years, 2012 was the edition with highest number of unique authors and 2017 was the year with the lowest. Churn rate is very high, the highest value was obtained for 2018 and the lowest for 2017.

We also investigated the number of countries represented by the authors. To this end we identified the country of all affiliations presented in the papers. Thus, if an author was affiliated to

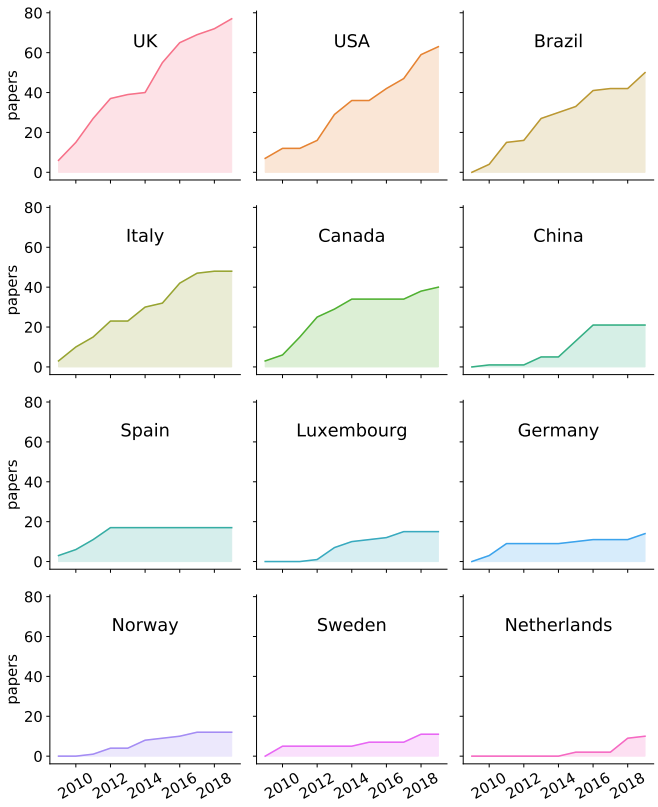


Figure 3: Contribution by countries

institutions of two countries, both were counted in our analysis. The analysis revealed that 25 different countries contributed to SSBSE. Figure 3 shows cumulative number of contributions of 12 distinct countries (48%).

The top 3 countries (12%) had a contribution of 43.7% and the top 5 countries (20%) had a contribution of 64%. We can observe that authors from the UK have contributed considerably more than other countries over all editions. Next, there are four countries that have been disputing the second position in the period: USA, Brazil, Italy and Canada. The USA have been maintaining the second place since 2013, tied with other countries in some years.

There is a third block, composed by: China, Spain, Luxembourg, Germany, Norway, Sweden and the Netherlands. There are some interesting aspects about these countries. We observe a big jump in the number authors from in years 2015 and 2016, which made them lead the number of contributions of this third group. We can also highlight the fact that Spanish authors participated actively only in the first four years of the event and, since 2014, nobody from Spain has published any other paper. Another important aspect of this group is that we observed an increase in the number of papers from Norway and Luxembourg, perhaps due to collaborations with other countries. Norway collaborated with other countries in 41.6% of published papers and Luxembourg in 30%.

Another analysis shown in Figure 4 presents the percentage of women that published papers. It varies from 8.3% in 2009 to 29% in 2013. The results show a big gender imbalance that has not been decreasing along the years.

Some authors belong to more than one kind of institution. Figure 5 displays the percentage of authors from universities, research foundations and industry. We can observe clearly that the great majority are from universities. One interesting point is that the research foundation Fondazione Bruno Kessler contributed in almost all editions, except 2013 and 2015. The percentage of papers exclusively from universities is 88.5%, exclusively from industry and also exclusively from research foundations is 4%, from universities in collaboration with industry is 3%, and from universities and research foundations is 0.5%. We noticed a modest participation of the industry.

4.1.4. Collaborations

To better identify the main SSBSE groups and collaborations we constructed a co-authorship network, which is shown in Figure 6. We observed that 46.3% of papers have external collaboration, that is, were published by authors from different institutions, and in 27.6% the institutions are from different countries. The University of Luxembourg formed the main group, collaborating with 11 different institutions. University College London collaborated with 8, Fondazione Bruno Kessler collaborated with 7, University of York and State University of Maringá with 5 and University of Sannio, Simula Research Laboratory, Università Della Svizzera Italiana, Federal University of Paraná, Technological Federal University of Paraná and Santa Catarina State University with 4 institutions. Moreover, there are many other collaborations with fewer connections.

We can conclude that there is a significant collaboration rate close to 50% and that the symposium plays an important role for this high percentage.

> ***RQ1:*** *SBSE has attracted participants from several countries. The mean number of countries represented in the committee is 13 and in the papers published is 25. Gender imbalance significantly varied from 5% to 30% for committee members and from 8% to 29% for authors. The total number of submitted papers was greater in the period of 2013 to 2016, and has declined in the last years. The great majority of the authors (94.8%) have published up to three papers. On average, churn rate of the event is high. The percentage of authors' collaboration is close to 50% in the papers.*

4.2. RQ2 – Citations Analysis and External Impact

This section presents results regarding the total number of citations of SSBSE editions and citations by individual papers in order to evaluate the impact of the symposium.

The number of citations was collected from Google Scholar (GS) on the 25^{th} October, 2019. All papers were individually evaluated, for which we collected their total number of citations (tagged as "Citations"), total number of citations excluding self-citations (tagged as "No Self-Citations"), and total number of citations excluding self-citations and citations by other SSBSE papers (tagged "External Citations").[5] Our citation analysis does not encompass the last edition of SSBSE (2019), because by the time we collected this data, the citations of 2019 papers had not been computed by Google Scholar yet.

In the past 10 years, SSBSE papers have received a total of 2 339 citations, of which 1 905 (81.4%) account for no self-citations and 1 809 (77.3%) represent external citations. The difference between the number of no self-citations and external citations is only 96 (4.1%), i.e., there are only 96 citations of SSBSE papers by different SSBSE authors and the remaining citations are from different venues.

Figure 7 depicts the average number of citations the papers received per year since they have been published. Figure 7.(a) shows the average citations by edition, whereas Figure 7.(b) shows the average by paper. Each SSBSE edition received on average 38.6 citations per year (39.8 median), of which 29.8 are no self-citations (32.4 median), and 28.3 are external citations (30.9 median). The SSBSE 2011 edition was the most cited one, with a total of 499 citations (62.4 per year).

Next we present some statistics by paper. On average, each SSBSE paper received 18.7 citations (13 median), of which 15.2 are no self-citations (8 median), and 14.5 are external citations (8 median). Moreover, each paper received on average 3.1 citations per year of its publication (2.4 median), of which 2.4 are no self-citations (1.7 median) and 2.3 are external citations (1.5 median). Even though the SSBSE 2011 edition was the most cited one per year in general, the SSBSE 2017 edition has a better average of citations by paper: each of the 7 papers received on average 10.3 citations, or 5.1 per year of their publication. Interestingly enough, this edition has the lowest number of published papers and acceptance rate (see Table 2).

Table 4 shows the 10 most cited papers from all editions. It is worth mentioning that these 10 papers have 736 citations (31.5% of all SSBSE papers combined). The paper authored by Arcuri and Fraser [38] is the most cited paper of the symposium.

Another interesting information is regarding SSBSE h-index and h5-index values [39]. The h-index counts the maximum number of h papers that have been cited at least h times. Such index can give a balanced assessment between quantity and quality (as measured by number of citations) of research papers. For example, an h-index of 100 means the 100 most cited papers have at least 100 citations. To increase the h-index to 101, the 101^{st} most cited paper must have at least 101 citations. Similarly, the h5-index compute the h-index for the papers published in the last 5 complete years (we consider the 5 years between 2014 and 2018). The SSBSE h-index is 28 and the h5-index is 13. As a matter of comparison, according to GS,[6] the h5-index

[5] To ease this task, we used Publish or Perish (`https://harzing.com/resources/publish-or-perish`), a tool that helps researchers look up information about papers, conferences, journals and others researchers in several repositories, including GS.

[6] `https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_softwaresystems`

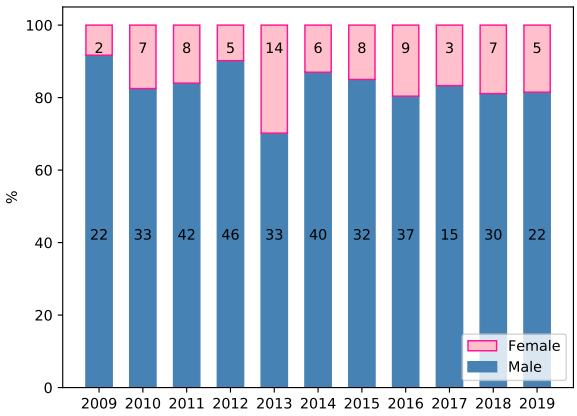


Figure 4: Authors Gender Imbalance

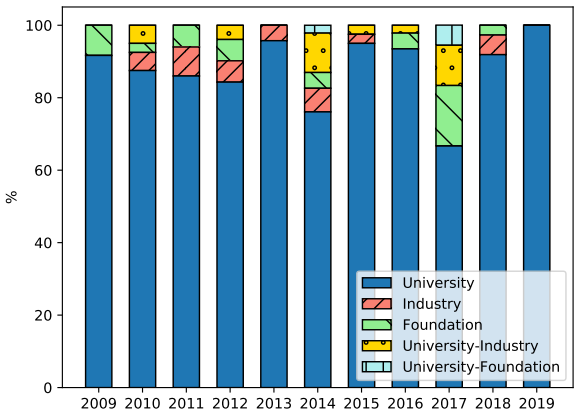


Figure 5: Source of Contributions

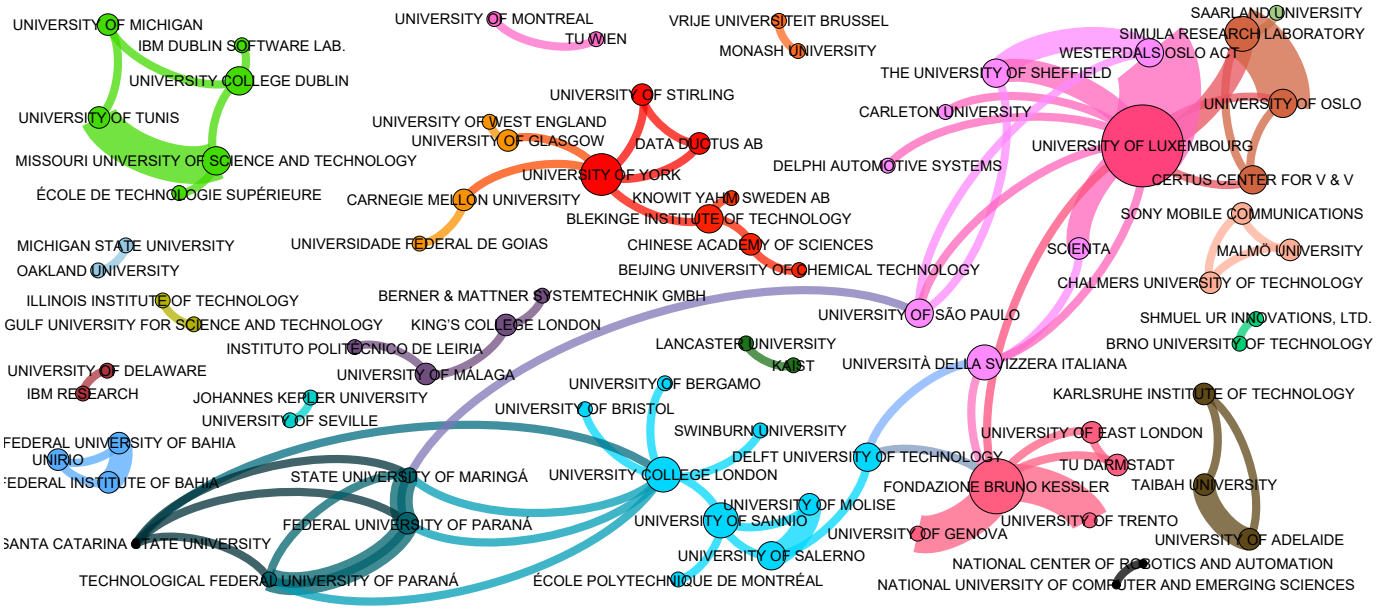
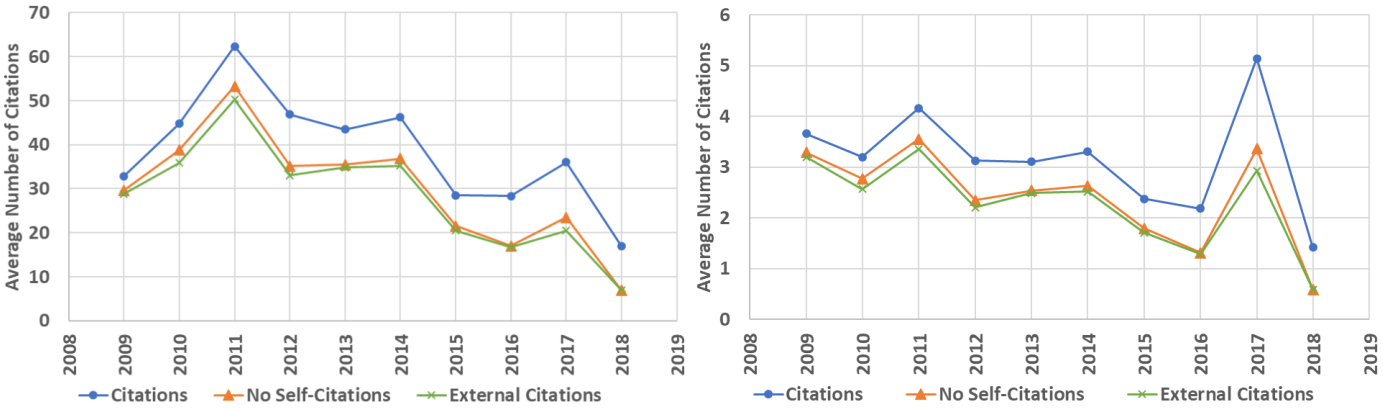


Figure 6: Collaboration network.



(a) Average by edition

(b) Average by paper

Figure 7: Average number of citations per year.

Table 4: Ranking of the 10 most cited SSBSE papers. C: citations, NS: no self-citations, E: external citations.

Year	Title	Authors	C	NS	E
2011	On Parameter Tuning in Search Based Software Engineering	Arcuri and Fraser	154	139	130
2009	An Improved Meta-Heuristic Search for Constrained Interaction Testing	Garvin *et al.*	85	79	77
2012	Evolving Human Competitive Spectra-Based Fault Localisation Techniques	Yoo	76	61	56
2011	Highly Scalable Multi Objective Test Suite Minimisation Using Graphics Cards	Yoo *et al.*	70	61	58
2009	A Study of the Multi-Objective Next Release Problem	Durillo *et al.*	66	61	57
2011	Ten Years of Search Based Software Engineering: A Bibliometric Analysis	de Freitas and Souza	62	61	57
2012	Putting the Developer in-the-Loop: An Interactive GA for Software Re-modularization	Bavota *et al.*	61	58	56
2012	Reverse Engineering Feature Models with Evolutionary Algorithms: An Exploratory Study	Lopez-Herrejon *et al.*	55	38	36
2010	The Human Competitiveness of Search Based Software Engineering	Souza *et al.*	54	46	40
2010	Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions	Ferruci *et al.*	53	42	40

of ACM/IEEE International Conference on Software Engineering (ICSE) is 75, IEEE Transactions on Software Engineering (TSE) is 48, IEEE Software is 41, IEEE/ACM International Conference on Automated Software Engineering (ASE) is 40, and ACM Transactions on Software Engineering and Methodology (TOSEM) is 28. Considering only external citations, the SSBSE h-index and h5-index values are respectively 25 and 11.

This close gap between no self-citations and external citations (both count and h-index) may indicate that the SSBSE papers have external visibility, as most of the citations are from different venues. Furthermore, this can also imply that such papers might have been used as source of inspiration for further research by the SE community.

We reported the number of citations as a measure of impact, however this might not be very accurate. As Ghezzi [40] stated in his keynote during the 31^{st} edition of ICSE, the most cited papers will not always represent the most influential ones. Sometimes, a paper is reported to be directly influenced by another paper, while having more citations. This can also be observed when comparing the rank of papers by citations count to the rank of most influential papers judged by the experts of the field. As shown by Ghezzi [40], the 8 most cited papers in the ICSE editions were elected as the most influential papers of that same year, but further down the rank, the most cited papers were not always selected as the most influential ones by experts.

A similar phenomenon happened with SSBSE. During the 10^{th} edition of the symposium in 2018, the community was asked to vote on the most influential paper of the past 10 years. The award was given to "The Human Competitiveness of Search Based Software Engineering" by de Souza *et al.* [41]. However, the award-winning paper is only the 9^{th} most cited paper (54 citations) with nearly a third of the citations of the 1^{st} one (154 citations).

All in all, the number of citations seemed to be the best metric of impact in the context of our work. This metric can be of some value, as a greater number of citations can tell more than smaller numbers, even though only about the visibility of papers. The best approach to evaluate the external impact of SSBSE papers would be to actually check the experts opinion, however, that is not a trivial task. Indeed, this could be done in future work with a more carefully designed impact evaluation with experts of top-tier software engineering venues.

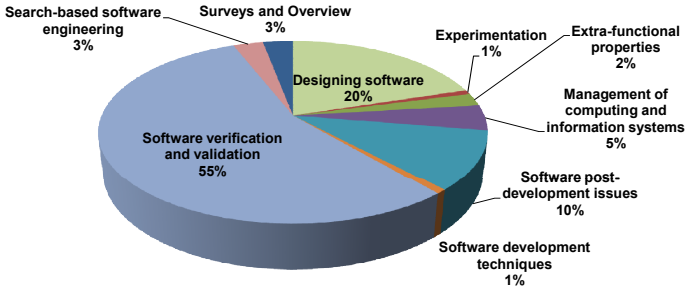


Figure 8: SE Areas Classification (level 2 of ACM CCS)

RQ2: *SSBSE papers have attracted substantial external visibility, as most of the citations are from different venues.*

4.3. RQ3 – Software Engineering Areas and Tasks

To answer RQ3, the papers were grouped by SE areas as described in Section 3. Figure 8 presents the percentages of studies by each identified SE area in the level 2 of ACM Computing Classification System. 55% of the papers are from Software Defect Analysis, which includes software testing and debugging, and 47% of them tackled some task related to software testing. This percentage is aligned with the software engineering wisdom that software testing takes about half of the whole development budget [3].

The SE area with the second greatest number of studies is Designing Software (20%). Such an area includes Requirements Analysis and Software Design. 10% of SSBSE papers deal with Software Post-development Issues (mainly Maintenance, Software Evolution and Reverse Engineering) and 5% of them are about Management of Computing and Information Systems, which involves People and Project Management. Other SE areas represent 7% of the overall publications: Search-based Software Engineering (3%), Surveys and Overview (3%), Extra-functional properties (2%), Software development techniques (1%) and Experimentation (1%).

Figure 9 shows the amount of papers published in SSBSE by year considering a more detailed level of SE area (level 3 of ACM Classification). Software Defect Analysis was addressed

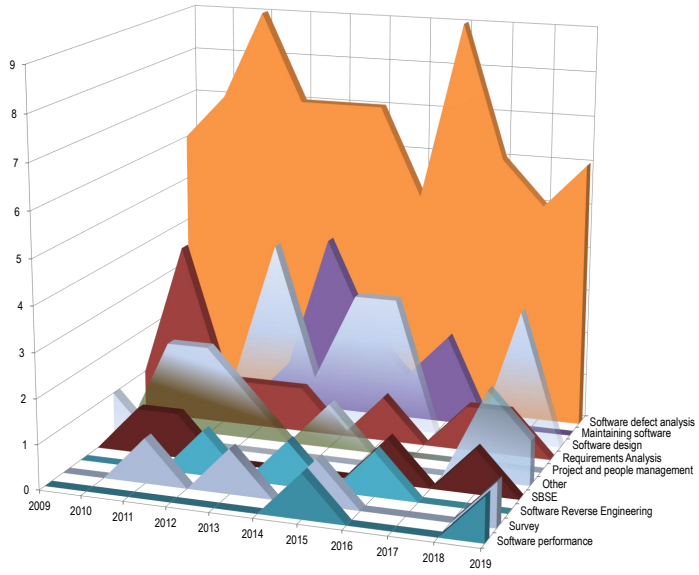


Figure 9: Amount of Papers published by Software Engineering Area (level 3 of ACM CCS)

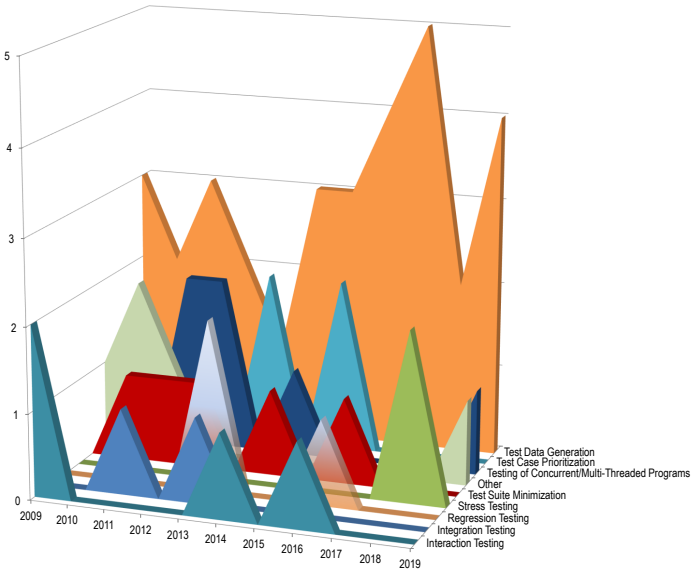


Figure 10: Amount of Papers published by Testing Tasks

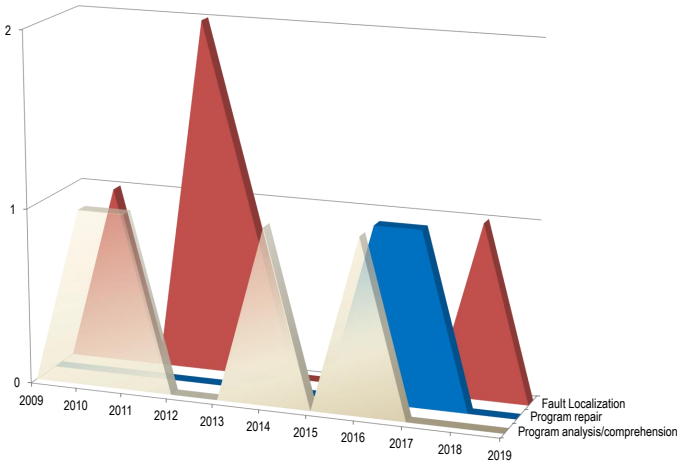


Figure 11: Amount of Papers published by Debugging Tasks

in every SSBSE edition. Maintenance, Software Design, Requirements Analysis and People and Project Management were the other most common SE areas for the level 3 of classification, representing, respectively, 7.5%, 12%, 8.2% and 4.5% of the SSBSE papers. Software Design and Requirements Analysis were investigated with regularity throughout the editions. The other SE areas were focused in less than 5% of the papers. SE areas such as Software Performance, Software Reliability, Automatic Programming and Experimentation have been addressed more recently.

The category "Other" includes papers about Software Reliability, Automatic Programming, Experimentation and Source-Code Authorship Definition. They have only one paper each.

4.3.1. Main tasks and problems

Figure 10 presents the tasks related to software testing. Test data generation was addressed in every SSBSE edition, comprising 50% of all software testing papers. The editions of the years 2017 and 2019 have five and four papers on this task, respectively. Defect prediction, test case evaluation and test management were tackled only in the first two editions. Test suite minimisation and testing of concurrent/multi-threaded programs were addressed over time. The last four editions contained papers on regression testing, stress testing, interaction testing, test suite minimisation, and bug identification.

Considering all editions, there are six papers about testing of concurrent/multi-threaded programs and five papers about test suite minimisation. Interaction testing and test case prioritisation were tackled by four papers. Regression testing was the subject of three papers, whereas integration testing and stress testing were addressed by two papers. The category "Other" includes papers about defect prediction, test case evaluation, test management, bug identification, and testability transformation for Java bytecode, each of which was addressed by only one paper.

The tasks related to software debugging are presented in Figure 11. Papers on debugging addressed tasks, such as fault localisation and program analysis, over time, and, more recently (2016 and 2017), program repair.

The Next Release Problem is the most addressed Requirements task, although the last publication about this problem was in 2015. This can indicate that the problem might have been satisfactorily solved. More recently, papers have focused on detection of incomplete requirements and non-functional requirements optimisation.

Regarding Software Design, most papers deal with architecture definition and model transformation in Model-Driven Engineering (MDE), followed by automatic software configuration, architecture improvement and software modularisation. After two years without publications in this area, three papers

addressing MDE were published in 2018.

Maintenance papers appeared between 2012–2016 and 55% of them addressed refactoring. The other SE tasks related to maintenance are bug prioritisation, code smells detection, non-functional properties optimisation, and automatic generation of maximally diversified versions, with one paper published for each task.

Most papers on Project and People Management deal with business process reduction and software project planning. Three surveys were published from 2011 to 2015. They addressed SBSE research analysis, metrics to search-based refactoring, and software requirement selection and prioritisation problems. Also, the survey about the papers of the 10 previous editions of SSBSE, which is extended in the present work, was published in SSBSE'2019.

Four papers treat SBSE over time, in the following order: SBSE evaluation, SBSE scalability, project decision making and online experimentation. Reverse engineering was applied to the Software Product Line approach in 3 published papers.

Tasks that have emerged in the last 4 years are the ones related to non-functional aspects (software performance, software reliability, non-functional properties optimisation and non-functional requirements optimisation), as well as program repair, stress testing, MDE, program synthesis and experimentation.

> ***RQ3:*** *The most common addressed SE area is Software Defect Analysis (55% of 134 papers). Test data generation is the most addressed task representing about 24% of overall full papers published in SSBSE editions.*

4.3.2. CI Techniques

Figure 12 shows the CI techniques used in the SSBSE papers over time. 75% of the papers applied (mono- or multi-objective) evolutionary algorithms. 24% applied local search, such as Hill-Climbing, Greedy, Simulated Annealing and Tabu Search. 4.8% used swarm intelligence algorithms (ACO and PSO). The category named "Other" (9% of the papers) includes algorithms such as Mathematical Optimisation, Mixed Integer Linear programming, Error-Correcting Graph Matching algorithm, Constraint Programming, Artificial Immune Recognition Systems, Random Search, etc. 4.5% of papers have also applied machine learning algorithms (Artificial Neural Network, Greedy Agglomerative Clustering or Multiple Regression).

22 out 134 papers used more than one CI technique. In some cases, different algorithms were used to compare which one has the best performance to solve the addressed problem. In other cases, algorithms from different CI techniques were combined to better solve a problem, which happened with the 4 papers that combined evolutionary algorithms and machine learning. Each one addressed the following tasks: refactoring, test data generation, test management, and automatic generation of maximally diversified versions.

As seen in Figure 12, since 2012 swarm intelligence has not been applied in SSBSE papers. The application of evolutionary

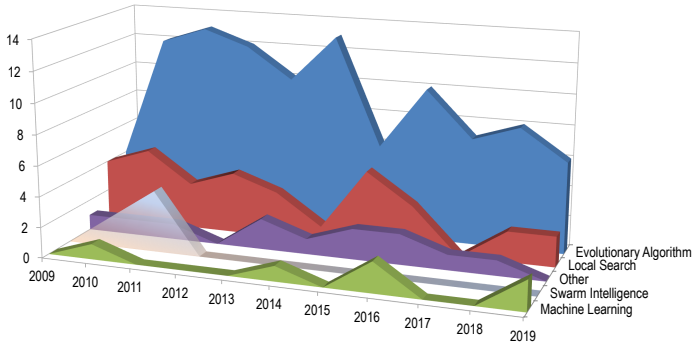


Figure 12: Amount of Papers published by CI Technique

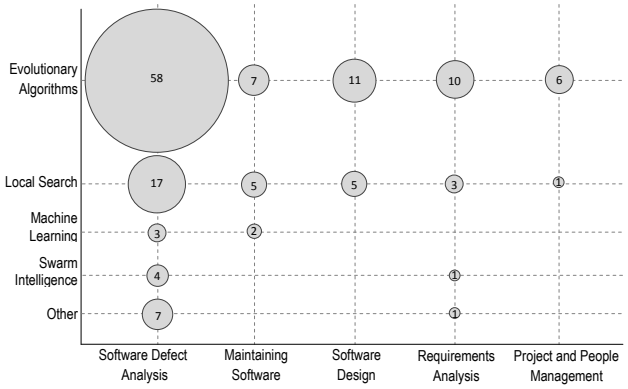


Figure 13: CI Techniques used by SE Areas

algorithms has also decreased over time. On the other hand, other CI techniques and machine learning algorithms have been increasingly used.

Figure 13 presents a bubble plot depicting the use of CI techniques to address SE problems of the five most common SE areas. Evolutionary algorithms are the most used technique followed by local search. Both are used by works of all areas. 26% of the overall publications employed some multi-objective evolutionary algorithm, mostly NSGA-II.

All kinds of CI techniques were used to solve problems related to Software Defect Analysis. For Requirements Analysis only machine learning algorithms were not used. Machine Learning was not used also for Software Design and People and Project Management. Swarm intelligence was used only in works on Software Defect Analysis and Requirements Analysis.

Table 5 lists the tasks most addressed by SSBSE papers, ordered from most to least used. Tasks addressed by only one paper are not shown. Test Data Generation is the most researched SE task, followed by Next Release Problem.

Table 6 lists the most used algorithms in SSBSE papers. Algorithms applied in only one paper were omitted. Genetic Algorithm is the most used, followed by NSGA-II.

Figure 14 shows a bubble chart correlating algorithms and SE tasks. The acronyms of tasks and algorithms are the same listed in Tables 5 and 6, respectively. Papers with only one occurrence of either SE task or algorithm are not shown. We can observe that the most frequent task and algorithm is Test Data

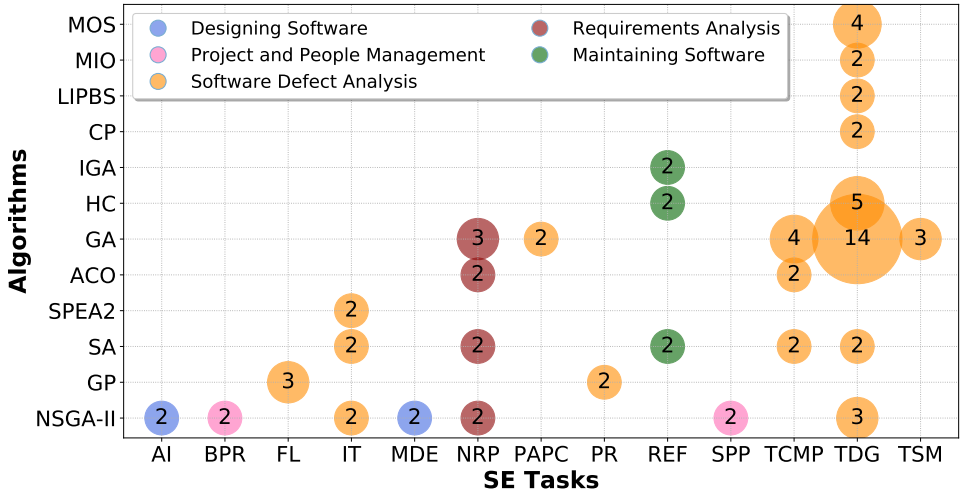


Figure 14: Algorithms used by SE Tasks

Table 5: Most addressed SE tasks.

Acronym	Task	Quantity
TDG	Test Data Generation	34
NRP	Next Release Problem	9
TCMP	Testing of Concurrent/Multi-Threaded Programs	8
REF	Refactoring	6
IT	Integration Testing	4
FL	Fault Localization	3
TSM	Test Suite Minimization	3
AI	Architecture Improvement	2
BPR	Business Process Reduction	2
IT	Interaction Testing	2
MDE	Model Transformation	2
PAPC	Program Analysis/Program Comprehension	2
PR	Program Repair	2
SPP	Software Project Planning	2

Table 6: Most used algorithms.

Acronym	Algorithm	Quantity
GA	Genetic Algorithm	26
NSGA-II	Non-dominated Sorting Genetic Algorithm-II	15
SA	Simulated Annealing	10
HC	Hill-Climbing	7
GP	Genetic Programming	5
MOSA	Many-Objective Sorting Algorithm	4
ACO	Ant Colony Optimization	4
CP	Constraint Programming	2
IGA	Interactive Genetic Algorithm	2
LIPBS	Linearly Independent Path based Search	2
MIO	Many Independent Objective algorithm	2
SPEA2	Strength Pareto Evolutionary Algorithm	2

Generation (TDG) and Genetic Algorithm (GA) with 14 occurrences. Analysing SE areas, most used algorithms in Software Defect Analysis are GA with 23 papers, Simulated Annealing (SA) with 6 and, in third, NSGA-II, Genetic Programming and Hill-Climbing with 5 occurrences. Designing Software and Project and People Management areas used only NSGA-II. Requirements Analysis used mostly GA (3 occurrences) followed by the algorithms NSGA-II, SA and Ant Colony Optimization, that tied with 2 papers. At last, papers of the Maintaining Software area used SA, Interactive Genetic Algorithm (IGA) and Hill-Climbing algorithms equally. It is worth mentioning that GA was the most used algorithm for Software Defect Analysis, Requirements Analysis and was not used by other SE areas. NSGA-II was the algorithm used for the highest number of different tasks. It was applied for all SE areas, except for Maintaining Software. On the other hand, Test Data Generation is the SE task that used the highest number of different algorithms.

4.3.3. Artefacts and Tools

The most common artefact used in Software Defect Analysis is source code. A few papers use class diagrams in order to predict defects and to perform regression testing using model transformation. A single paper used bug reports to identify bugs in the code. From the 32 papers that perform test data generation, 11 use the EvoSuite tool, arguably the most used and efficient test data generation tool in SBST. GenProg was evaluated in the two papers tackling program repair.

Studies classified into the Software Design area used models (such as class diagrams and metamodels) and source code as main artefacts. Models are mainly used in works about Architecture Improvement and MDE. jMetal and Evolutionary Computation for Java (ECJ) were the most cited tools to implement the evolutionary algorithms.

Requirements Analysis research commonly use requirements instances, models, and algorithms as main artefacts for optimisation. The tools used were jMetal, RELAX, MATLAB, and the robust optimisation framework.

Project and People Management works perform optimisation over project planning data sets with function points, architec-

tural models, and source code directly. The most common tools used in the case studies are JWebTracer, JBPRecovery, JBPFreqReducer, JBPEvo2, jMetal, and BPOntoManager.

The majority of papers with focus on Maintenance use source code, corresponding to 60%. The tools presented to deal with source code are Code-Imp and Ref-Finder. One study deals with Unified Modeling Language (UML) class diagram with a tool called MOREX+I (MOdel REfactoring by eXample plus Interaction, Ref-Finder). Refactoring operations, bug information, and Android apps bytecode are artefacts distinctly considered in other three papers. For these latter papers the authors did not mention specific tools.

> ***RQ3:*** *The most common CI technique applied is evolutionary algorithms (75% of the papers). Genetic Algorithm and NSGA-II are the most used algorithms. Evolutionary algorithms and machine learning were combined in some papers to better solve some SE problem. The most common artefact used is source code.*

4.4. RQ4 – Experimental Rigour

In this section we present aspects related to how proposed SBSE solutions are evaluated int the SSBSE papers, by detailing subjects, statistical tests, and quality indicators used.

4.4.1. Subjects

The subjects used for evaluating proposed approaches vary in their characteristics. For example, there are small computer programs, a.k.a. toy systems, typically used for proof of concept or educational purposes. For the purpose of quantitative evaluation we also commonly found synthetic systems/problems used as subjects in some studies. Toy/educational/synthetic subjects were used in 33.6% of the SSBSE papers. Some instances of this category are: Arcade Game Maker, Microwave Oven Software, and Service and Support System. Sometimes only small pieces of software are used in the experimentation such as Java classes.

Another category is industrial systems, present in 30.5% papers, which allows an evaluation of how SBSE solutions work in practice. We have examples of these subjects for different platforms such as desktop (Microsoft Word and ReleasePlanner), Web (Tudu, Oryx and Softslate Commerce), mobile (Sony Mobile and Android programs), embedded software (Adaptive headlight control, door lock control, and electric windows control), and MATLAB Simulink models.

The evaluations also used open source projects, representing 28.2%, commonly taken from repositories such as SourceForge,[7] GitHub,[8] SPLOT,[9] and Google Play.[10] Examples of this category are Eclipse, Mozilla, Apache Commons project, Apache Ant, ArgoUML, Azureus, Xerces-J, JHotDraw, AJHSQLDB, Health Watcher, Toll System, JFreeChart, Rhino, and GanttProject.

Another type of subjects are those obtained from publicly available datasets and benchmarks, observed in 7.6% of the papers. For example, Bench4BL dataset, F-Droid benchmark, SF100 Java benchmark, Siemens faulty program benchmark, and SEAMS repository.[11] We also observed the use of synthetic data, non-real artefacts, sometimes randomly generated, used to represent difficult problems or large artefacts, exposing the power of SBSE solutions.

4.4.2. Quality indicators and Comparison measures.

Quality indicators and comparison measures are mainly used to obtain information about solutions generated with SBSE approaches. These indicators and metrics are the base to design fitness functions, compare search-based algorithms, or reason about the search process.

For evaluating single solutions we observe measures such as absolute error, relative error, area under the curve, accuracy, completeness, robustness, accuracy, and customers satisfaction. When the goal is to compare two or more solutions, for instance solutions obtained with different algorithms, we can see measures such as size of the solutions (size of test suite, number of clusters, number of refactorings), precision and recall, and similarity to the desired solutions. Comparison among sets of solutions are mainly performed in multi/many-objective optimisation, where two Pareto fronts are compared using indicators such as coverage, generational distance, inverted generational distance, hyper-volume, euclidean distance to the ideal solution, and spread.

We also identified some metrics/measures related specifically to the SE area. For example, for Testing: average percentage of fault detected, fault detection effectiveness, branch coverage, code coverage, axiom coverage, method coverage, line coverage, percentage of parameter interactions covered, and mutation score. When optimising software for hardware, we found analysis related to exposures to data loss, temporal flexibility when diffusing data, simultaneous network partitions and tolerate dropped messages, data stall cycles, energy savings, non-functional improvement, network communication, and maximum call tree. In the area of Software Design, clustering algorithms are applied and measures such as the number of clusters, intra-edges, inter-edges, and number of isolated clusters are observed, as well as object-oriented metrics such as Chidamber and Kemerer, and QMOOD.

In addition to evaluating the solutions of SBSE algorithms, authors also evaluate the search-process. This is done by considering success rate (for instance, the percentage of times the branch was covered over 50 runs), performance of the algorithm, predictability, average number of executions taken to find the test data, degeneration of the search process, effect of penalisation, efficiency, stability, number of solutions generated, value of the elitist chromosome for each generation, scalability,

[7] `https://sourceforge.net/`
[8] `https://github.com/`
[9] `http://www.splot-research.org/`
[10] `https://play.google.com/store`
[11] `http://www.self-adaptive.org/`

Table 7: Statistical tests and Effect size measures used

Test/Measure	#Papers
Mann-Whitney-Wilcoxon U-test	53
Vargha-Delaney A12 effect size	25
Student's T-test	5
Kruskal-Wallis	4
Cliff's Delta	3
Kolmogorov-Smirnov test	3
Spearman's RC coefficient	3
Friedman test	3
Two-Tailed Test	2
Fisher Exact test	2
ANOVA	1
Bootstrap test	1
Cohen's d	1
Linear correlation	1
Scott-Knott test	1
Welch's t-test	1

execution time, time saving, time spent, and human competitiveness.

4.4.3. Statistical tests and Effect size measures.

SBSE approaches rely on CI techniques, which apply randomness in their search process. In order to identify a standard behaviour, commonly proposed approaches are run many times. For each run, metrics and measures, described in the last section, are collected and evaluated with statistical tests. The goal is to assess whether there are significant difference among results or not.

Table 7 presents the statistical tests and effect size measures that were applied in SSBSE papers. The last column of the table shows the number of papers using the tests. Among the 16 found tests/measures, Mann-Whitney-Wilcoxon U-test and Vargha-Delaney A12 effect size were by far the most commonly used, representing 71.6%. Regarding the use of statistical testing along the years, Figure 15 shows that tests and measures have been used since 2009, however, we can observe that after 2014 they have been used more frequently. In 2017 all papers used these tests/measures. In 2019 we observe a decrease in the number of papers applying statistical tests because there are more secondary and theoretical papers. Here is important to note that such statistical tests should be used carefully and together with qualitative analysis for properly evaluating SBSE solution [42].

RQ4: *The subjects used in the SSBSE papers for evaluating their approaches are toy/educational/synthetic examples (33.6%), industrial systems (30.5%), open source projects (28.2%), and few papers used datasets and benchmarks (7.6%). We observed a large number of quality indicators and comparison measures used in the papers. Regarding the analysis of results, in 93.3% of the papers we can observe evaluation results and among them 60% apply at least one type of statistical analysis.*

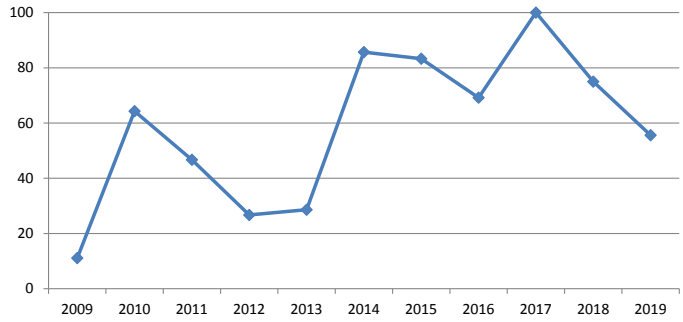


Figure 15: Percentage of papers that apply statistical tests or Effect size per year

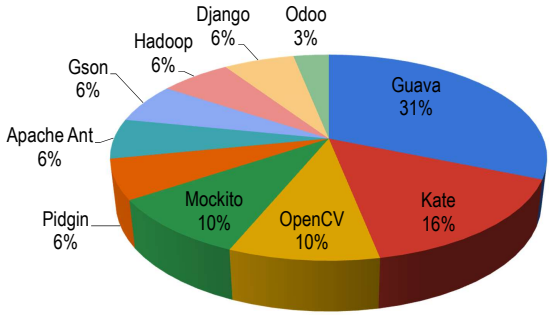


Figure 16: Challenge programs

4.5. RQ5 – Challenge Papers

The SSBSE Challenge Track is an opportunity for SBSE researchers to apply tools, techniques, and algorithms on open source software in order to carry out interesting analysis and uncover interesting insights related to them.

A total of 32 papers were published in the SSBSE Challenge Track (see detailed numbers in Section 4.1). They have authors affiliated to 21 different institutions. Thirteen papers (40%) were written in collaboration involving authors belonging to different institutions. However, the majority of papers were written by authors from six universities: University College London, University of York, Queen Mary University of London, University of Sheffield, University of South Carolina, and University of Stirling. Only one is from outside of the United Kingdom. 56% of the papers have at least one author affiliated to University College London. 12% of the papers have at least one author from the University of York. Each one of the other four universities appears in 9% of the papers.

Figure 16 presents the programs addressed by the 32 papers published in Challenge Track. They are programs written in Java, C, C++, and Python. The most used programs are Guava (10 papers) and Kate (5 papers).

Four papers generated test data for Guava. Three papers tackled its refactoring and the remaining papers focused on program comprehension, improving energy consumption, and regression testing together with test case prioritisation. Regarding Kate, two papers tackled its design focusing on the architecture improvement and component-based software design, whereas three papers explored other SE tasks, such as adding new functionality, program repair, and test data generation. We observed

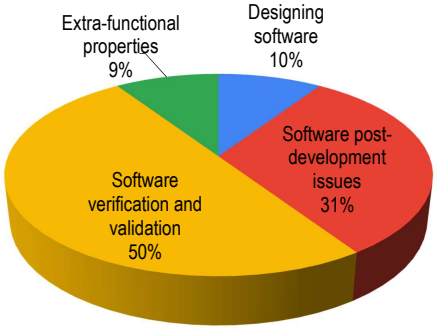


Figure 17: Challenge Papers - SE Areas Classification (level 2 of ACM CCS)

that all the programs were the goal of some testing task in at least one paper.

We classified and grouped the papers of Challenge Track following the same method used in RQ3 (see Section 3.3). Figure 17 presents the percentage of challenge papers by each identified SE areas in the level 2 of ACM Computing Classification System. 50% of the papers focus on Software Verification and Validation, 43.7% of which focusing on software testing. These percentage values are similar to those ones observed in the main track of SSBSE.

The other two most common SE areas are the same observed in the main track: Software post-development issues and Designing software. However, in the Challenge track, Software post-development issues is the second most common area, which is expected because this track deals with existing software that naturally have maintenance issues. Software post-development issues includes studies about maintenance (5 papers), software evolution (4 papers), and software comprehension (1 paper). Designing software includes problems related to architecture improvement (2 papers) and component-based software engineering (1 paper).

Figure 18 presents the amount of papers published by year considering the level 3 of ACM Classification. In this picture, Software defect analysis is subdivided in testing tasks and debugging tasks. Testing tasks were addressed in almost all years (2013–2018) including test data generation (31% of papers), stress testing (3%), test suite minimisation (3%), defect prediction (3%), and regression testing together test case prioritisation (3%). Program repair is the single task of debugging. It was addressed in 2015 and 2019 (6%).

Software design was tackled only in the earliest editions involving architecture improvement and component-based software engineering. Refactoring is the single maintenance task. It was regularly addressed from 2014 to 2017.

Software performance and Software evolution were mainly addressed in 2015 and 2016 — the two years with more challenge papers. Software performance includes tasks related to improving energy consumption and parameter optimisation. Two tasks are related to Software evolution: adding new functionality and automate software improvement. Other refers to program comprehension, which was the focus of a study published in 2015.

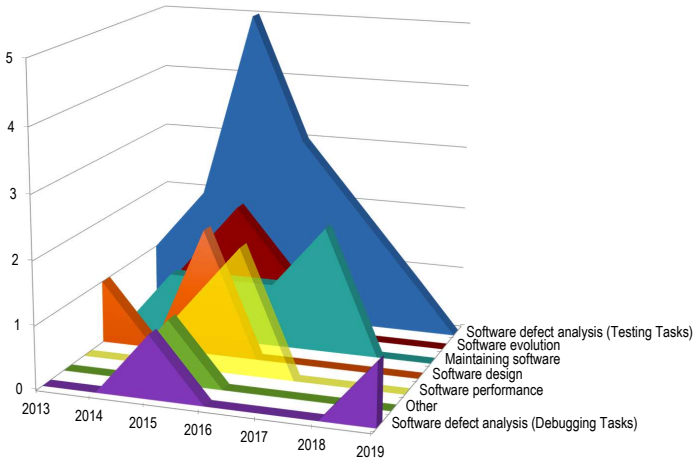


Figure 18: Challenge Papers - Amount of Papers published by Software Engineering Area (level 3 of ACM CCS)

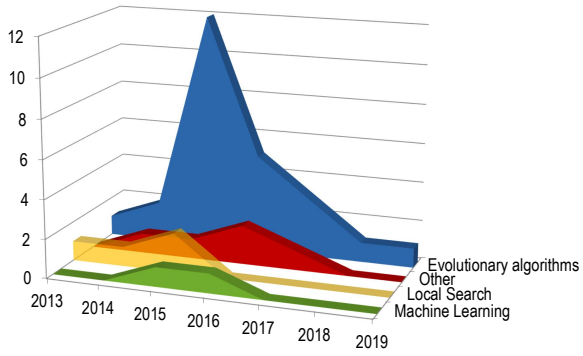


Figure 19: Challenge Papers - Amount of Papers published by CI Technique

In accordance with the main track, in the Challenge Track the most applied CI techniques are Evolutionary Algorithms (69% of the papers) and Local Search (11%). Some papers also applied Machine Learning (6%) and Other algorithms (14%).

Figure 19 shows the amount of CI techniques used by challenge papers by year. Evolutionary algorithms were used in all editions of the track (2013–2019). The most used algorithms of this kind are genetic algorithm (34% of 32 papers), NSGA-II (21%), and genetic programming (18%).

In this track, source code is also the main artefact used, representing 62% of the challenge papers (20 papers). 15 papers (47%) applied some statistical test in the experimental evaluation.

We consider that is natural that the same SE areas and tasks as well as the same algorithms are the most common in both tracks (main and challenge), since the SBSE researchers' expertise (who are usually authors of the main track) is employed to carry out the studies published in the Challenge Track.

RQ5: *Guava and Kate are the real-world software most used by the challenge papers. The most common addressed SE area in the Challenge Track is Software Defect Analysis (50% of 32 papers). Test data generation is the most addressed task representing about 31% of challenge papers. Evolutionary algorithms (GA and NSGA-II) are the most used CI technique.*

4.6. Recommendations to future SSBSE authors

During the screening of the 134 papers, we realised that some pieces of information are not presented in several papers, what makes the SBSE approaches not completely clear to the readers. Next we present some recommendations to future SSBSE authors in order to: i) provide writing practices to improve the definitions of SBSE approaches; ii) help authors to develop high quality studies; and iii) allow study replication.

- Make it clear which are the CI techniques and algorithms used, not just mention the tool or framework name, as it will improve readability and provide the reader with a more in-depth understanding of the paper. This is also important for sake of reproducibility;
- Clarify the ingredients of SBSE approaches that allow the application of CI techniques to solve the corresponding SE problem, namely the problem representation and the fitness function(s). Some (meta-)heuristics also need operators to modify candidate solutions, which should also be described in the text;
- Illustrative examples can be added in SBSE papers so that readers can easily understand the problem and the proposed solution. Authors can use toy programs to this mean;
- For the evaluation, authors should prefer using industrial systems or open source projects from different domains and sizes. This would make the findings more general and hopefully more practical in a real-world setting;
- To avoid threats regarding randomness of CI techniques, run your approaches many times and assess the results with statistical tests and effect size measures. Arcuri and Briand [43] present guidelines on how to evaluate randomised algorithms in software engineering, how to improve the statistical power of the experiments, and how to avoid errors. According to the authors, 1 000 independent runs would be ideal for a powerful statistical comparison, however, it might not always be feasible to perform this many independent runs due to computational power constraints. We observed that the most common number of runs is 30, but we cannot ensure this number would be enough for all cases. Ultimately, the authors should carefully consider the number of independent runs that can be performed in feasible time, and whether this number is enough for a powerful statistical comparison;
- The comparison to existing approaches is always recommended. This provides the reader with a baseline to reason about the benefits of the proposed solution. For approaches that deal with new problems, we recommend the authors to at least compare to random search as a matter of sanity check;
- Make the experimental package available. Additionally, provide as many resources as possible for other authors to replicate your study and/or to ease comparison. We suggest future researches to use Open Science Framework (OSF)[12] to increase the openness, integrity, accessibility, and reproducibility of scientific research.

5. Trends and Challenges

The analysis of the collected data allowed us to identify some trends as well as research opportunities in SBSE. The majority of identified trends and challenges are corroborated by literature as we pointed in what follows.

5.1. Trends and Problems Addressed so Far

The analysis conducted to answer RQ3 (Section 4.3) shows the area of Software Defect Analysis, including testing and debugging tasks, is the most addressed in the symposium. The application of search-based optimisation for testing problems is one of the origins of the SBSE field and SSBSE. But our findings show that, after ten years, such problems are still natural candidates for SBSE. Problems related to the test data generation task, addressed since the early SSBSE editions, keep raising interest [3]. On the other hand, some testing tasks have no longer received attention, such as test management, integration testing, and fault prediction, while others have recently being focused, such as fault localisation, program repair, and non-functional program improvement [23]. We observed some trends in the Software Testing area. Works have started addressing some more recent testing problems, but there are still many open issues [44]: flaky tests, time-consuming testing, untestable code, and complex and multi-platform testing.

The areas Requirement Analysis, Software Design, and Maintenance are less explored than Software Defect Analysis, but they have been investigated since the first editions and stayed active throughout the years (see Section 4.3). Works on these areas appear in almost all editions. On the other hand, the area People and Project Management, present in the beginning, has not been explored in the last five years. There are many research opportunities in these areas to address limitations of existing work. For instance, in the area of Software Design and Refactoring, most approaches only focus on improving cohesion and coupling and other simplistic quality attributes. Other aspects should be considered such as the development history, design problems, and constraint violations. Some challenges that need to be addressed are related to the lack of flexibility, and the need of an expert to validate the generated solutions.

[12] osf.io

Besides these limitations, we observe a trend that is to investigate problems related to non-functional aspects such as energy consuming and accessibility [45,46]. New problems to perform the tasks in all found areas have arisen, requiring investigations in emergent and new contexts such as Service-oriented architectures, Internet of Things (IoT), Highly-Configuration Systems (HCS), User Interface (UI) systems, and mobile applications.

5.2. SBSE Challenges

We can state that the SSBSE future depends on the SBSE field, in which we can identify several challenges to be addressed, mainly to tackle the growing complexity of the nowadays software systems and applications. The application contexts of SBSE currently encompasses a great number of objectives, constraints, and complex inputs as well as outputs, making most SBSE problems multi- and many-objective. However, dealing with such plethora of variables is a hard problem by itself [31, 32, 34].

Harman *et al*. [47] point some advantages of SBSE solutions. They are robust, scalable, realistic, and generic. Unfortunately, most SBSE works do not rigorously evaluate scalability and robustness of the generated solutions. It is still a challenge to propose solutions that are generic and useful for software engineers in a real-world setting.

The great majority of studies mention as future research or limitation the evaluation in real industrial scenarios. The literature reports challenges and issues related to the cooperation between academia and industry in software engineering areas [48]. They are also valid for SBSE, but in this field we can also mention other difficulties that hinder this cooperation, such as non-determinism of the algorithms, hard modelling of the search space, and high computational cost for high scale industrial software. These are barriers that still need to be considered and overcome.

Regarding robustness, we observe some challenges due to the dynamic aspects of the software systems. For example, requirements can change over time, thus adaptive SBSE approaches are required. Maybe the use of adaptive fitness functions and introducing the developer in the loop can help solve this challenge.

We found in our analysis different solutions or algorithms proposed for the same problem. A lot of effort is spent implementing the approaches and evaluating algorithms. A research opportunity is to automate these tasks, allowing automatic design of search-based approaches, as well as their evaluation. This can help in the practical issue regarding the choice of the best algorithm, search operator, fitness functions.

Despite each problem's peculiarity, many software engineering tasks have similarities that should be better explored by generic algorithms. However, algorithmic generality is a hard problem by itself, regardless of whether it is applied to software engineering problems or not.

Software engineers make decisions by simultaneously analysing different software artefacts such as requirements, test cases, architectural models, and source code. These artefacts and the areas that treat them are strongly connected, but most works are oblivious to their interactions. Although very challenging, meaningfully transferring information from one area to another and orchestrating artefact interactions can greatly benefit SBSE techniques.

Other practical issue is regarding the usefulness of generated solutions. Often the users do not recognise the solutions as feasible because they were not generated considering their needs and preferences. Involving the user in the loop has been investigated in the past [35, 36], but it is still an open challenge for the SBSE community. To name a few, these constitute some research opportunities: i) resolution of conflicting opinions provided by different users or different priorities given to the many objectives impacting the problem; ii) selection of the best moment and way to incorporate the preferences; and iii) to avoid human-fatigue in interactive approaches.

It is important for SBSE approaches to consider different levels of automation. They should be adaptive to make small decisions and invoke human participation to more fundamental ones. In this sense, the use of machine learning techniques seems to be a trend to learn the user behaviour and help in the decision making process. Furthermore, modern development environments allow the collection of large amount of data and diverse software repositories are available with the emergence of Software Engineering platforms (e.g. SourceForge and GitHub). The work of Nair *et al*. [21] calls this intersection as Data-Driven Search-Based Software Engineering, name for the field combining SBSE and Mining Software Repositories (MSR) techniques, and provides insights about how this combination can happen in many areas. The paper also discusses some open issues, regarding the human participation as well as optimisation of optimisers.

The software engineering community has seen a sudden rise in interest for Fuzzing [49], mainly due to the success of the American Fuzzing Loop (AFL)[13] tool developed by Google employees. Fuzzing tackles the generation of test data in a different way than SBSE. However, there might be multiple points of intersection between both fields, such as how to balance the exploration and exploitation when generating test inputs, or how to create readable test cases from the generated inputs. Using the best of two worlds can result in the creation of powerful tools such as EvoSuite, but with the efficiency and scalability of AFL. Making it happen shall be as challenging as it sounds promising.

5.3. Suggestions for Future SSBSE Editions

Considering our findings and what happens in other conferences of the software engineering area, next we present some mitigation actions that can steer future SSBSE editions and ideas to strength the symposium.

Answering RQ1, we observe a decreasing number of submissions in the last years (Section 4.1.2, Figure 2), as well as a modest industry participation in the authorship (Section 4.1.3, Figure 5). It is important to increase industry participation in

[13] `http://lcamtuf.coredump.cx/afl/`

SSBSE aiming at accelerating industrial adoption of the scientific endeavours and enabling the evaluation of SBSE approaches in real industrial scenarios. We believe that the inclusion of an industry track in SSBSE could help in this sense, as happens in other conferences such as the IEEE/ACM International Conference on Automated Software Engineering (ASE) and the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE).

Adding a mentoring program would be an alternative to increase the acceptance rate regarding submissions from new authors. Authors that never published in SSBSE research track and that intend to submit a paper to this track, could request to participate in the mentoring program. The Program Committee (PC) chairs could assign a member of the PC to make a preliminary assessment of the work and give feedback before the submission deadline, increasing the chances of less experienced authors, as well as the number of submissions.

Future editions of the symposium could be held in countries where there is a significant rate of contributions (see Figure 3). Canada and China, for example, have the 4^{th} and the 5^{th} greater number of contributions and never hosted an edition of SSBSE.

We observe that the challenge track is of great relevance for the symposium and some institutions are committed in submitting papers to this track, since they regularly published papers (see Section 4.5). In this way, we see that the inclusion of challenge cases related to industrial problems would attract the attention of the industry. Furthermore, in addition to the call for challenge solutions, the call for challenge cases, where the community propose the challenges to be taken into account, would be an alternative.

The women SSBSE leadership participation is rather good as pointed out by Figure 1. But Figure 4 shows a small participation regarding authorship. Gender bias is a major concern in software engineering discipline [37]. Gender diversity is important because it can help to share different skills, points of view and experiences, bringing and incorporating gender aspects of the customers and users in software engineering, expanding potential talents, among other aspects. Hence, increasing the participation of women in the symposium is of great value.

A repository of benchmarks for the different SBSE sub-areas could be organised and maintained by SSBSE. Such a repository would be useful for: (i) the evaluation of the SBSE approaches, (ii) contribute to solve the challenge of transferring information from one area to another and orchestrating artefact interactions, and (iii) divulge the symposium for whom is interested in using the benchmarks. The SSBSE chairs could call for the initial contributions to create the repositories and stimulate the collaborative creation of new benchmarks in some actions performed during the symposium.

Last but not the least, the reason why the number of submissions to SSBSE has decreased over time should be better investigate in future work.

6. Conclusion

In this paper we presented an overview of the history of SSBSE as well as results from a systematic mapping involving all full papers of the eleven proceedings. Our findings allow us to state that SSBSE papers have made some external impact on the SE research community. We found that most of the citations are from different venues and identified a close gap between no self-citations and external citations. This indicates SSBSE papers have external visibility.

Regarding the area of SE problems solved with CI techniques, software testing is still the main addressed task, but other problems have emerged, mostly related to non-functional requirements, program repair, MDE, and experimentation. Evolutionary algorithms remained the most used CI technique with machine learning to raise more interest in the past years.

We could observe along the years a wide range of subjects used by authors to evaluate their approaches. These subjects can also be used in new research. Moreover, in recent years authors are paying more attention to the use of statistical tests to better evaluate their results. But it is important to increase industry participation and the creation of repositories containing benchmarks regarding the different SBSE sub-areas.

To call attention and guide new authors willing to publish their papers and to participate in SSBSE, we presented a set of recommendations to improve their publications on understandability, replicability, and experimentation soundness. However, the recommendations are limited to what we observed during the papers screening.

There are still several problems to be addressed in the SBSE field as well as many challenges that we have previously stated, which can be tackled by SSBSE authors in further studies.

SSBSE has been a representative venue to divulge studies and put together academics, researchers and practitioners to discuss SBSE. Currently, the SBSE field is explicitly listed as a topic of interest of important conferences and journals. Given what we reported and discussed in this paper, we can state that SSBSE has helped increase the popularity of SBSE in the SE research community and has played an important role on making SBSE more mature.

Acknowledgment

This work was funded by CNPq (Grants 305968/2018-1, 428994/2018-0, and 408356/2018-9), and by the ERC advanced fellowship grant 741278: Evolutionary Program Improvement Collaborators (EPIC).

References

[1] M. Harman, S. A. Mansouri, Y. Zhang, Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications, Tech. rep., Department of Computer Science, King's College London (2009).

[2] M. Harman, S. A. Mansouri, Y. Zhang, Search-based Software Engineering: Trends, Techniques and Applications, ACM Comput. Surv. 45 (1) (2012) 1–61.

[3] M. Harman, Y. Jia, Y. Zhang, Achievements, open problems and challenges for search based software testing, in: International Conference on Software Testing, Verification and Validation, 2015.

[4] F. G. Freitas, J. T. Souza, Ten years of search based software engineering: A bibliometric analysis, in: M. B. Cohen, M. Ó Cinnéide (Eds.), Search-Based Software Engineering, 2011, pp. 18–32.

[5] P. McMinn, Search-based software test data generation: A survey, Software Testing Verification and Reliability 14 (2) (2004) 105–156.

[6] O. Räihä, A survey on search-based software design, Computer Science Review 4 (4) (2010) 203–249.

[7] Y. Zhang, A. Finkelstein, M. Harman, Search based requirements optimisation: Existing work and challenges, in: REFSQ '08, REFSQ '08, 2008, pp. 88–94.

[8] A. M. Pitangueira, R. S. P. Maciel, M. Barros, Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature, Journal of Systems and Software 103 (2015) 267 – 280.

[9] M. Di Penta, SBSE Meets Software Maintenance: Achievements and Open Problems, in: Search-Based Software Engineering, Vol. 7515, 2012, pp. 27–28.

[10] T. Mariani, S. R. Vergilio, A systematic review on search-based refactoring, Information and Software Technology 83 (2017) 14–34.

[11] T. E. Colanzi, W. K. G. Assunção, P. R. Farah, S. R. Vergilio, G. Guizzo, A review of ten years of the symposium on search-based software engineering, in: S. Nejati, G. Gay (Eds.), Search-Based Software Engineering, Springer International Publishing, 2019, pp. 42–57.

[12] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and Software Technology 64 (2015) 1 – 18.

[13] M. Khari, P. Kumar, An extensive evaluation of search-based software testing: a review, Soft Computing 23 (6) (2019) 1933–1946.

[14] D. Sharma, P. Chandra, Applicability of soft computing and optimization algorithms in software testing and metrics–a brief review, in: International Conference on Soft Computing and Pattern Recognition, Springer, 2016, pp. 535–546.

[15] I. Boussaïd, P. Siarry, M. Ahmed-Nacer, A survey on search-based model-driven engineering, Automated Software Engineering 24 (2) (2017) 233–294. doi:10.1007/s10515-017-0215-4.

[16] M. Mohan, D. Greer, A survey of search-based refactoring for software maintenance, Journal of Software Engineering Research and Development 6 (1) (2018) 3.

[17] F. Ferrucci, M. Harman, F. Sarro, Search-based software project management, in: Software Project Management in a Changing World, Springer, 2014, pp. 373–399.

[18] R. E. Lopez-Herrejon, L. Linsbauer, A. Egyed, A systematic mapping study of search-based software engineering for software product lines, Information and software technology 61 (2015) 33–51.

[19] R. E. Lopez-Herrejon, J. Ferrer, F. Chicano, A. Egyed, E. Alba, Evolutionary computation for software product line testing: an overview and open challenges, in: Computational Intelligence and Quantitative Software Engineering, Springer, 2016, pp. 59–87.

[20] M. Harman, Y. Jia, J. Krinke, W. B. Langdon, J. Petke, Y. Zhang, Search based software engineering for software product line engineering: a survey and directions for future work, in: 18th International Software Product Line Conference-Volume 1, ACM, 2014, pp. 5–18.

[21] V. Nair, A. Agrawal, J. Chen, W. Fu, G. Mathew, T. Menzies, L. Minku, M. Wagner, Z. Yu, Data-driven search-based software engineering, in: 15th International Conference on Mining Software Repositories, MSR '18, ACM, New York, NY, USA, 2018, pp. 341–352.

[22] M. Harman, Automated test data generation using search based software engineering, in: 2nd International Workshop on Automation of Software Test, IEEE Computer Society, 2007, p. 2.

[23] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, J. R. Woodward, Genetic improvement of software: a comprehensive survey, IEEE Transactions on Evolutionary Computation 22 (3) (2017) 415–432.

[24] E. Sodagari, M. Keyvanpour, Software requirements interaction management using search-based software engineering methods: A survey, in: 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), IEEE, 2017, pp. 0481–0486.

[25] A. M. Pitangueira, R. S. P. Maciel, M. de Oliveira Barros, A. S. Andrade, A systematic review of software requirements selection and prioritization using SBSE approaches, in: Symposium on Search Based Software Engineering, Springer, 2013, pp. 188–208.

[26] W. K. Assunção, R. E. Lopez-Herrejon, L. Linsbauer, S. R. Vergilio, A. Egyed, Reengineering legacy applications into software product lines: a systematic mapping, Empirical Software Engineering 22 (6) (2017) 2972–3016.

[27] V. Singh, Software module clustering using metaheuristic search techniques: A survey, in: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, 2016, pp. 2764–2767.

[28] A. V. Rezende, L. Silva, A. Britto, R. Amaral, Software project scheduling problem in the context of search-based software engineering: A systematic review, Journal of Systems and Software 155 (2019) 43 – 56. doi:10.1016/j.jss.2019.05.024.

[29] A. Abdelmabou, D. Jawawi, I. Ghani, A. Elsafi, A comparative evaluation of cloud migration optimization approaches: A systematic literature review, Journal of Theoretical and Applied Information Technology 79 (2015) 395–414.

[30] R. Pietrantuono, S. Russo, Search-based optimization for the testing resource allocation problem: research trends and opportunities, in: 2018 IEEE/ACM 11th International Workshop on Search-Based Software Testing (SBST), IEEE, 2018, pp. 6–12.

[31] A. Ramírez, J. R. Romero, S. Ventura, A survey of many-objective optimisation in search-based software engineering, Journal of Systems and Software 149 (2019) 382 – 395. doi:10.1016/j.jss.2018.12.015.

[32] A. S. Sayyad, H. Ammar, Pareto-optimal search-based software engineering (POSBSE): A literature survey, in: 2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), IEEE, 2013, pp. 21–27.

[33] S. Z. Qasim, M. A. Ismail, Research problems in search-based software engineering for many-objective optimization, in: 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), IEEE, 2017, pp. 1–6.

[34] J. M. Balera, V. A. de Santiago Júnior, A systematic mapping addressing hyper-heuristics within search-based software testing, Information and Software Technology 114 (2019) 176 – 189. doi:10.1016/j.infsof.2019.06.012.

[35] A. Ramírez, J. R. Romero, C. Simons, A Systematic Review of Interaction in Search-Based Software Engineering, IEEE Transactions on Software Engineering 45 (2019) 760–781. doi:10.1109/TSE.2018.2803055.

[36] T. N. Ferreira, S. R. Vergilio, J. T. de Souza, Incorporating user preferences in search-based software engineering: A systematic mapping study, Information and Software Technology 90 (2017) 55–69.

[37] S. Agarwal, N. Mittal, R. Katyal, A. Sureka, D. Correa, Women in computer science research: What is the bibliography data telling us?, SIGCAS Comput. Soc. 46 (1) (2016) 7–19.

[38] A. Arcuri, G. Fraser, On parameter tuning in search based software engineering, in: Search-Based Software Engineering, 2011, pp. 33–47.

[39] J. E. Hirsch, An index to quantify an individual's scientific research output, National Academy of Sciences 102 (46) (2005) 16569–16572.

[40] C. Ghezzi, Reflections on 40+ years of software engineering research and beyond: an insider's view, in: keynote address in 31st International Conference on Software Engineering, 2009.

[41] J. T. Souza, C. L. Maia, F. G. de Freitas, D. P. Coutinho, The human competitiveness of search based software engineering, in: Search-Based Software Engineering, 2010, pp. 143–152.

[42] R. L. Wasserstein, A. L. Schirm, N. A. Lazar, Moving to a world beyond "p < 0.05", The American Statistician 73 (sup1) (2019) 1–19. doi:10.1080/00031305.2019.1583913.

[43] A. Arcuri, L. Briand, A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering, Software Testing, Verification and Reliability 24 (3) (2014) 219–250. doi:10.1002/stvr.1486.

[44] E. Laukkanen, J. Itkonen, C. Lassenius, Problems, causes and solutions when adopting continuous delivery—a systematic literature review, Information and Software Technology 82 (2017) 55 – 79. doi:10.1016/j.infsof.2016.10.001.

[45] I. Manotas, J. Clause, L. Pollock, Exploring evolutionary search strategies to improve applications' energy efficiency, in: T. E. Colanzi, P. McMinn (Eds.), Search-Based Software Engineering, Springer International Publishing, Cham, 2018, pp. 278–292.

[46] K. M. Bowers, E. M. Fredericks, B. H. C. Cheng, Automated optimization of weighted non-functional objectives in self-adaptive systems, in: T. E. Colanzi, P. McMinn (Eds.), Search-Based Software Engineering, Springer International Publishing, Cham, 2018, pp. 182–197.

[47] M. Harman, P. McMinn, J. T. de Souza, S. Yoo, Search based software engineering: Techniques, taxonomy, tutorial, in: B. Meyer, M. Nordio (Eds.), Empirical Software Engineering and Verification: International Summer Schools, LASER 2008-2010, Elba Island, Italy, Revised Tutorial Lectures, Springer Berlin Heidelberg, 2012, pp. 1–59. doi:10.1007/978-3-642-25231-0_1.

[48] V. Garousi, K. Petersen, B. Özkan, Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review:, Information and Software Technology 79. doi:10.1016/j.infsof.2016.07.006.

[49] A. Zeller, R. Gopinath, M. Böhme, G. Fraser, C. Holler, The fuzzing book, in: The Fuzzing Book, Saarland University, 2019.