UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n° 0006/2011

Threats to Validity in Search-based Software Engineering Empirical Studies

Márcio de Oliveira Barros
Arilo Claudio Dias Neto

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Relatórios Técnicos do DIA/UNIRIO, No. 0006/2011
Editor: Prof. Sean W. M. Siqueira Abril, 2011

Threats to Validity in Search-based Software Engineering Empirical Studies

Márcio de Oliveira Barros[1], Arilo Claudio Dias Neto[2]

[1] Postgraduate Information Systems Program – UNIRIO
Av. Pasteur 458, Urca – Rio de Janeiro, RJ – Brazil
[2] Computer Science Department – UFAM - Postgraduate Program in Informatics
R. Gen. Rodrigo Octávio Jordão Ramos, 3000 – Setor Norte – Manaus, AM – Brazil

{marcio.barros@uniriotec.br, arilo@dcc.ufam.edu.br}

Resumo. O uso de experimentos para avaliar e comparar algoritmos tem sido uma tendência recente na área de computação. O mesmo pode ser dito para a área de engenharia de software, onde os principais estudos experimentais nesta área foram publicados, normalmente, a menos de vinte anos. Experimentos em *search-based software engineering* (SBSE), ou engenharia de software baseada em buscas, compartilham as limitações provenientes da imaturidade de ambas suas áreas de origem (engenharia de software e otimização combinatória). Uma dessas limitações está relacionada à carência de uma lista de ameaças à validade que podem afetar experimentos em SBSE. Neste trabalho, é proposta uma lista de ameaças à validade para experimentos em SBSE baseada em um framework proposto na área de Engenharia de Software Experimental (ESE). Ainda é apresentada uma abordagem sistemática para avaliar quão abrangentes um artigo científico é em relação à descrição de tais ameaças. Ao aplicar esta abordagem a artigos científicos publicados nas duas primeiras edições do *International Symposium on Search-Based Software Engineering* (SSBSE), percebeu-se que as principais limitações do projeto experimental dos experimentos descritos nesses artigos estão relacionadas à descrição de ameaças à validade *internas*, *externas* e *de constructo*, enquanto que ameaças à validade de conclusão são mais bem abordadas por estes artigos.

Palavras-chave: Engenharia de Software baseada em buscas; estudos empíricos; ameaças à validade.

Abstract. The usage of experiments to evaluate and compare algorithms is a recent trend. The same can be said for software engineering, since major experimental studies in the area are usually no more than twenty-years old. Search-based software engineering (SBSE) experiments share the limitations born out of the immaturity in both its source areas. One of these limitations regards the lack of a list of validity threats that may affect SBSE experiments. In this work, we propose a list of validity threats for SBSE experiments based on the framework proposed in the Empirical Software Engineering field. We also present a systematic approach to evaluate to which extent a research paper addresses each of these threats. By applying this approach to papers published in the first two editions of the International Symposium on Search-Based Software Engineering, we perceived major limitations in addressing internal, external, and construct validity threats, while conclusion threats are better handled by these papers.

Keywords: Search-based software engineering; empirical studies; validity threats.

Index

Threats to Validity in Search-Based Software Engineering Empirical Studies

1. Introduction

The usage of experimental analysis to evaluate and compare algorithms is a recent trend [1]. It has been gaining attention from researchers who are fonder of programming than applying the theoretical derivations required to evaluate the worst-case and average-case performance of the algorithms under analysis. It has also being sought as a bridge to connect the theory on algorithms to practice, as experimentation relies on implementing the algorithms and evaluating their effectiveness and efficiency on resolving a given set of (usually, practical and real) problems.

Search-based software engineering (SBSE) is a research field whose goal is to use search-based optimization algorithms to automate the construction of solutions to software engineering problems [2] [3]. SBSE links the fields of algorithms and software engineering and its technical literature shows that the practical perspective of the later is imposed upon the former. Therefore, we observe a large number of SBSE research proposals being evaluated through experimentation instead of theoretical analysis. However, bringing together the specificities of experimentation in software engineering and algorithms is no trivial task.

Recently, Ali *et al*. [4] presented a systematic review on how empirical investigations are conducted in search-based software testing (SBST), a research area within the larger SBSE research community. As part of the paper summarization process that underlies the systematic review, the authors proposed a list of validity threats to SBSE experiments. Validity threats are potential risks that are involved in the design and execution of empirical studies [5]. These threats may limit the studies' ability to yield reliable results or their generalization to a larger population than the sample instances used in the experiments. The proposed validity threats follow the general framework proposed by [5], which is composed of four major classes: conclusion, internal, construct, and external validity threats. However, the description of the proposed threats is limited (possibly due to space restrictions) and incomplete, based on our experience and formerly published SBSE experiments.

In this paper, we propose a list of validity threats that may be considered while designing a SBSE experiment. The list was built upon validity threats proposed by [4] and uses the same framework proposed by [5]. We also present a systematic procedure to ascertain to which extent a research paper assesses these threats. By applying this procedure, we analyzed all full-papers published in the proceedings of the first two editions of the Symposium on Search-Based Software Engineering (SSBSE), covering papers from 2009 (9 papers) to 2010 (14 papers). We perceived major limitations in addressing internal, external, and construct validity threats, while conclusion threats are generally better handled by these papers.

Besides this introduction, this paper is composed of 4 sections. Section 2 presents the proposed validity threats to SBSE experiments. Section 3 presents the approach to evaluate how validity threats are addressed in a research paper. Section 4 presents our evaluation regarding the assessment of validity threats in SSBSE papers. Finally, in Section 5 we draw some conclusions and propose further research.

2. Validity Threats to SBSE Experiments

In the following paragraphs, we propose a set of the validity threats that may affect SBSE experimental studies. These threats are based in our experience, on research papers describing how to conduct experiments to evaluate search algorithms (mainly the work of Johnson [1]), on papers reporting on SBSE experiments, and on a former list of validity threats (the work of Ali *et al*. [4]). The proposed threats follow the general framework proposed by Wohlin *et al*. [5], which group them into four major classes: conclusion, internal, construct, and external threats. To separate new threats from those originally proposed by Ali *et al*. [4], the former are marked with a "[+]" sign.

2.1. Conclusion Validity Threats

These threats are concerned with the relationship between treatment and outcome. The empirical design must make sure that there is a statistical relationship between the parts involved. Major conclusion threats that may affect a SBSE experiment include:

- **Not accounting for random variation:** meta-heuristic algorithms are usually related to stochastic random-number generators. For instance, the initial population of a genetic algorithm is usually randomly generated and the starting point of a hill-climbing search is commonly selected at random. Therefore, a single run of an experimental study upon a given instance may yield results that carry the benefits of a favorable initial random selection or the prejudices of a badly selected random starting point [1]. Therefore, all experiment should be executed several times for each instance;
- **Lack of good descriptive statistics:** since experiments must collect data from many execution cycles for each instance, these data must be summarized in a meaningful way to allow drawing conclusions. At a minimum, papers should show the average of the observed results or the percentage of successful runs, but preferably some measure of the variation of observed results, such as their standard deviation, min-max range or a box-plot, could also be presented [4];
- **Lack of a meaningful comparison baseline:** SBSE experiments usually compare results observed from running two or more algorithms (or algorithmic configurations) upon the same set of instances. The baseline for comparing a new algorithm may be a distinct meta-heuristic search approach, a local search procedure, or a non-heuristic approach. Frequently, random search is used as a basic verification, showing that the problem cannot be adequately solved by selecting random solutions in the search space and, therefore, a systematic search procedure is required [4]. Anyway, to draw reliable conclusions based on such a comparison, the baseline must be representative of the best-known solution so far;
- **[+] Lack of formal hypothesis and statistical tests:** the comparison described in the former paragraph must be based on a formal hypothesis and must be evaluated by a proper statistical test. By proper, we mean a statistical test that adheres to the characteristics of the underlying data under evaluation, through parametric or non-parametric statistical inference procedures.

2.2. Internal Validity Threats

If a relationship is observed between treatment and outcome, the experiment design must guarantee that it is a causal relationship and not the result of a factor upon which the researcher has no control. Major internal threats that may affect a SBSE experiment include:

- **Poor parameter settings:** the selected parameters for the proposed technique or comparison baseline are not explicitly presented in the experimental design. By hiding parameter settings, the designer may favor one or another technique, possibly limiting the generalization of observed conclusions. Moreover, a complete description of parameter values is required for the experiment to be reproducible – an important quality of any empirical study;
- **[+] Lack of discussion on code instrumentation:** the source code used in an experiment may hide specific tweaks or instrumentations to favor certain instances or algorithms, thus influencing the observed results. Johnson [1] suggests that the source code should be made publicly available, allowing other researchers to reproduce and inspect the experiment;
- **[+] Lack of clear of data collection tools and procedures:** the steps executed to collect information from real-world instances or to generate random instances used in the experiments must be precisely described to make sure that these aspects are not influencing the observed results by selecting favorable instances;
- **[+] Lack of real problem instances:** Software Engineering is a practical science and as such must handle real problems. Although randomly-generated instances may be useful to address the behavior of a new technique in certain situations, they may lack of properties found in real-world instances. Since, that may influence the conclusions drawn from the experiment, researchers must use structured random instances combined with a reasonable set of real instances [1].

2.3. Construct Validity Threats

These threats are concerned with the relations between theory and observation, ensuring that the treatment reflects the construct of the cause and that the outcome reflects the construct of the effect. Major construct threats that may affect a SBSE experiment include:

- **Lack of assessing the validity of cost measures:** the number of fitness evaluations is accepted as the most general and useful cost measure [1]. When measuring the cost of executing an algorithm by any other means, the researcher must justify the selected metric;
- **Lack of assessing the validity of effectiveness measures:** these measures must be relevant to the underlying problem and an improvement in their values must be related to an improvement in the quality of the solution. Ideally, these metrics must have a clear interpretation under the problem of interest. Therefore, the selection of proper effectiveness measures is relevant to make sure that observed results sustain or refuse the proposed theory;
- **[+] Lack of discussing the underlying model subjected to optimization:** the environment under which software is developed is usually complex, involving both technical and social issues. Therefore, any model describing a software-related problem is a simplification of the real world. When manipulating such models to propose a theory, one must be sure to discuss their limitations and how these simplifications may influence their practical applications.

2.4. External Validity Threats

These threats are concerned with the generalization of observed results to a larger population, outside the sample instances used in the experiment. Major external threats to SBSE experiments include:

- **[+] Lack of a clear definition of target instances:** no generalization is possible if researchers cannot understand the instances used in an empirical study. Therefore, any experimental design must provide a clear definition of the selected instances;
- **[+] Lack of a clear object selection strategy:** given the aspects addressed by the algorithmic approach under evaluation, the research must clearly depict how the instances used in the experiment were selected, designed, randomly-generated, or collected from real-world problems. The researcher must also justify why these instances are used in the experiment at hand;
- **[+] Lack of evaluations for instances of growing size and complexity:** Software Engineering techniques are designed to handle systems and teams that may vary in size and complexity. For instance, while a given software project may have a few requirements, other may have thousands. Therefore, a SBSE approach must be evaluated across a breadth of problem instances, both varying in size and complexity, to provide an assessment on the limits of the new technique.

Given the former classification, we evaluated how research papers published in the first editions of the Symposium on Search-Based Software Engineering (SSBSE) assess validity threats. To perform this evaluation, we developed a systematic approach to assess how research papers address validity threats. This approach is presented in the next section.

3. Evaluating the Assessment of Validity Threats

We propose a qualitative, questionnaire-based evaluation framework to assert to which extent a research paper addresses validity threats to SBSE experiments. The questionnaire is composed of 17 questions, each addressing a particular validity threat. These questions are answered qualitatively, indicating that the paper provides a Limited (L), Partial (P), or Complete (C) assessment of the threat at hand.

3.1. Questionnaire used for Assessment of Validity Threats

Below, we present the questions comprising the questionnaire and the interpretation of each possible qualitative answer. Not all response possibilities (L, P, and C) are available for all questions. Also, we cannot claim that these questions provide a complete framework to address validity threats that may affect SBSE experiments: in the present state of our research, they represent the best compilation of experience and issues reported in related papers, but their effectiveness is yet to be assessed.

Q1: *Is the experiment run many times for each instance?*

L: the experiment is run less than ten times[1] for each instance.

C: the experiment is run ten or more times for each instance.

Q2: *Is the data collected from the experiments well-summarized?*

L: no precise information collected from the experiments is presented.

P: only average values of the collected information are presented.

C: average and dispersion values of the collected information are presented.

Q3: *Is there a meaningful comparison baseline?*

[1] Ten runs was a rule-of-thumb limit proposed by Ali *et al*. in [4].

L: no comparison baseline is provided.

P: one or more heuristic algorithms are used as comparison baseline.

C: one or more non-heuristic algorithms are used as comparison baseline.

Q4: Are the hypothesis formally presented?

L: no clear description of the hypothesis under interest is given.

P: a clear, though textual, description of the hypothesis is given.

C: a formal description of the hypothesis is given.

Q5: Are statistic inference tests used?

L: no statistic inference tests are used to compare results to the baseline.

C: at least one statistic inference test is used to compare results to the baseline.

Q6: Does the paper discuss the model subjected to optimization and its limitations?

L: a new problem and a new model are presented, but not discussed.

P: a recurrent problem is presented; a model is presented, but not discussed.

C: a model is presented and its practical limitations are discussed.

Q7: Does the paper discuss the validity of cost measures?

L: execution time, individual or interaction counts are used and not justified.

P: the former measures are used; their limitations are addressed and justified.

C: the number of fitness evaluations or an equivalent metric is used.

Q8: Does the paper discuss the validity of effectiveness measures?

L: an unclear effectiveness metric is used.

P: a set of well-defined metrics are used, but without clear justification.

C: a set of well-defined metrics are used and clearly justified.

Q9: Does the paper discuss on any instrumentation made to the code?

L: no mention to which code was used.

P: a code structure, flow-chart or reference is presented.

C: a code structure or reference is presented and tweaks are discussed.

Q10: Is the code used to run the studies available for third-party researchers?

L: code is unavailable and is not discussed.

P: code is available for third-part use, but instances are not.

C: code and used instances are available for third-part use.

Q11: Does the paper describe the data collection procedures used in the experiment?

L: no mention is made on how the data supporting effectiveness and efficiency measures was collected.

P: limited discussion on how the data was collected.

C: software tools or procedures used to collect data are presented.

Q12: Does the paper present a clear definition of target instances?

L: target instances are not clearly described.

P: major attributes of target instances are described, but details are lacking.
C: target instances are precisely described.

Q13: Does the paper present a clear object selection strategy?
L: the reasons to use the proposed instances are not clear.
C: the proposed instances are selected in order to fill some validity threat.

Q14: Does the paper accounts for variation on instance size in the experiments?
L: variation on instance size is not addressed.
P: two instances of different size are used in the experiments.
C: more than two instances of different size are used in the experiments.

Q15: Does the paper accounts for variation on instance complexity in experiments?
L: variation on instance complexity is not addressed.
P: two instances of different complexity are used in the experiments.
C: more than two instances of distinct complexity are used in the experiments.

Q16: Does the paper use real-world instances in the experiments?
L: no real-world instance is used in the experiments.
P: up to two real-world instances are used in the experiment.
C: more than two real-world instances are used in the experiment.

Q17: Does the paper clearly present the parameter values used in the experiments?
L: no parameter value or incomplete data is presented for the algorithms.
C: precise parameter values are presented for all algorithms.

The next subsection presents the data collected from all 23 analyzed papers according to the questionnaire proposed above (questions Q01 to Q17).

3.2. Data Collected from SSBSE papers

In the following paragraphs, we analyze the assessment of validity threats in the full papers published in the first two editions of the International Symposium on Search-Based Software Engineering (SSBSE). Table 1 presents the list of the analyzed SSBSE papers.

Table 1. List of full papers published in the editions 2009 and 2010 of SSBSE

Year	Paper ID	Author	Title	SW Eng Issue
2009	P01-2009	A. Marchetto and P. Tonella	Search-Based Testing of Ajax Web Applications	Software Testing
	P02-2009	B. Garvin, M. Cohen, M. Dwyer	An Improved Meta-Heuristic Search for Constrained Interaction Testing	Software Testing
	P03-2009	S. Kpodjedo, F. Ricca, G. Antoniol, P. Galinier	Evolution and Search Based Metrics to Improve Defects Prediction	Software Metrics
	P04-2009	J. Durillo, Y. Zhang, E. Alba, A. Nebro	A Study of the Multi-Objective Next Release Problem	Software Requirements
	P05-2009	D. Kim and S. Park	Dynamic Architectural Selection: A Genetic Algorithm Based Approach	Software Architecture
	P06-2009	M. Shevertalov, J. Kothari, E. Stehle, S. Mancoridis	On the Use of Discretized Source Code Metrics for Author Identification	Software Metrics
	P07-2009	U. Khan, I. Bate	WCET Analysis of Modern Processors Using Multi-Criteria Optimisation	Software Testing

	P08-2009	A. Arcuri	Full Theoretical Runtime Analysis of Alternating Variable Method on the Triangle Classification Problem	Software Testing (Runtime Analysis)
	P09-2009	K. Ghani and J. Clark	Widening the Goal Posts: Program Stretching to Aid Search Based Software Testing	Software Testing
2010	P01-2010	G. Lu, R. Bahsoon, X. Yao	Applying Elementary Landscape Analysis to Search-based Software Engineering	Release Planning
	P02-2010	P. McMinn	Does Program Structure Impact the Effectiveness of the Crossover Operator in Evolutionary Testing?	Software Testing
	P03-2010	S. Yoo	A Novel Mask-Coding Representation for Set Cover Problems with Applications in Test Suite Minimization	Software Testing
	P04-2010	Y. Zhang and M. Harman	Search Based Optimization of Requirements Interaction Management	Software Requirements
	P05-2010	P. Tonella, A. Susi, F. Palma	Using Interactive GA for Requirements Priorization	Software Requirements
	P06-2010	J. Sagrado, I. Águila, F. Orellana	Ant Colony Optimization for the Next Release Problem: A Comparative Study	Software Requirements
	P07-2010	W. Afzal, R. Torkar, R. Feldt, G. Wikstrand	Search-based Prediction of Fault-Slip-Through in Large Software Projects	Software Testing
	P08-2010	F. Ferrucci, C. Gravino, R. Oliveto, F. Sarro	Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions	Project Management
	P09-2010	K. Lakhotia, M. Harman, H. Gross	AUSTIN: A tool for SBST for the C Language Evaluation on Deployed Automotive Systems	Software Testing
	P10-2010	F. Lindlar, A. Windisch	A Search-Based Approach to Functional Hardware-in-the-Loop Testing	Software Testing
	P11-2010	M. Sheverlatov, K. Lynch, E. Stehle, C. Rorres, S. Mancoridis	Using Search Methods for Selecting and Combining Software Sensors to Improve Fault Detection in Autonomic Systems	Software Testing
	P12-2010	J. Xiao, W. Afzal	Search-based Resource Scheduling for Bug Fixing Tasks	Software Testing, Project Management
	P13-2010	J. Souza, C. Maia, F. Freitas, D. Coutinho	The Human Competitiveness of SBSE	Software Requirements, Project Management, Software Testing
	P14-2010	F. Asadi, G. Antoniol, Y. Gueheneuc	Concept Location with GA: A Comparison of Four Distributed Architectures	Software Maintenance

Table 2 presents the summary of data collected from the questionnaires answered for each full paper published in the editions 2009 and 2010 of SSBSE

Table 2. Data collected from 2009 and 2010 SSBSE papers

Paper –ID	*Q 01*	*Q 02*	*Q 03*	*Q 04*	*Q 05*	*Q 06*	*Q 07*	*Q 08*	*Q 09*	*Q 10*	*Q 11*	*Q 12*	*Q 13*	*Q 14*	*Q 15*	*Q 16*	*Q 17*
P01-2009	L	P	C	L	L	P	C	C	C	L	C	P	C	C	L	L	C
P02-2009	C	P	C	P	L	C	C	C	C	P	P	P	L	L	P	C	C
P03-2009	C	P	C	P	C	P	C	P	P	C	P	P	C	L	L	P	L
P04-2009	C	C	C	P	C	P	P	C	L	C	L	C	L	C	C	L	C
P05-2009	C	P	C	L	L	L	C	L	L	L	L	P	C	P	L	L	C
P06-2009	L	L	C	L	L	C	C	L	L	L	C	L	C	P	P	C	C
P07-2009	C	C	L	L	L	C	P	L	L	L	L	P	C	C	C	C	C
P08-2009	C	C	L	C	L	C	C	L	L	C	L	C	C	C	P	L	C
P09-2009	C	C	C	L	L	P	L	L	C	C	P	C	C	C	P	L	C
P01-2010	C	C	C	P	C	P	L	L	L	L	P	P	C	C	C	L	C
P02-2010	C	P	C	P	C	P	C	C	L	L	P	C	C	C	P	L	L
P03-2010	C	C	P	P	C	P	P	P	L	L	C	C	L	C	P	C	C
P04-2010	L	L	P	P	L	C	C	P	L	L	P	C	C	C	P	L	C
P05-2010	C	C	C	P	C	P	L	C	C	L	P	C	L	L	L	P	C
P06-2010	C	C	P	P	L	P	L	P	L	L	P	C	L	L	L	P	C

P07-2010	C	C	C	P	C	L	L	P	P	L	C	C	L	L	L	P	C
P08-2010	C	C	C	P	C	P	L	L	P	L	P	P	C	L	L	P	C
P09-2010	C	C	C	C	C	P	C	C	C	C	P	C	C	C	C	C	C
P10-2010	L	P	C	L	L	P	L	C	L	L	P	P	C	L	L	P	C
P11-2010	L	P	C	L	L	P	L	C	P	L	C	P	L	L	L	P	C
P12-2010	C	P	C	L	C	C	L	P	P	L	C	C	L	L	L	P	C
P13-2010	C	C	C	P	C	P	P	P	P	L	P	P	L	P	L	L	C
P14-2010	L	C	P	L	L	C	L	P	C	L	C	C	L	P	L	P	C

The next section analyzes the data collected from all 23 analyzed papers regarding the assessment of validity threats according to the framework proposed in the section 2.

4. Data Analysis

Table 3 presents the number of papers for each qualitative category and validity threat. Alongside with each validity threat, we present the questions that address the threat. When two questions are presented for the same threat, the highest answer in the proposed qualitative scale is considered for evaluation purposes and accountability. Percentile values are truncated before the decimals and this may yield lines that do not sum up to 100%.

Table 3. Validity threats assessment in SBSE research papers

Threat Type	Validity Threat	L	A	C
Conclusion	Not accounting for random variation (Q1)	6 (26%)	n/a	17 (74%)
	Lack of good descriptive statistics (Q2)	2 (9%)	8 (35%)	13 (56%)
	Lack of a meaningful comparison baseline (Q3)	2 (9%)	4 (17%)	17 (74%)
	Lack of formal hypothesis and statistical tests (Q4, Q5)	8 (35%)	3 (13%)	12 (52%)
Internal	Poor parameter settings (Q17)	2 (9%)	n/a	21 (91%)
	Lack of discussion on code instrumentation (Q9, Q10)	9 (39%)	5 (22%)	9 (39%)
	Lack of description of data collection procedures (Q11)	4 (17%)	12 (52%)	7 (30%)
	Lack of real problem instances (Q16)	9 (39%)	9 (39%)	5 (22%)
Construct	Lack of validating cost measures (Q7)	10 (43%)	4 (17%)	9 (39%)
	Lack of validating effectiveness measures (Q8)	7 (30%)	8 (35%)	8 (35%)
	Lack of discussing the underlying model (Q6)	2 (9%)	14 (61%)	7 (30%)
External	Lack of a definition of target instances (Q12)	1 (4%)	10 (43%)	12 (52%)
	Lack of a object selection strategy (Q13)	10 (43%)	n/a	13 (57%)
	Lack of instances of growing size (Q14)	9 (39%)	4 (17%)	10 (43%)
	Lack of instances of growing complexity (Q15)	12 (52%)	7 (30%)	4 (17%)

Regarding conclusion validity, we observe that most SSBSE papers account for the random variation on heuristic search algorithms, provide a good summary of the data collected during the experiments using proper statistics, and use a meaningful comparison baseline. Most exceptions to these rules, especially regarding a proper assessment of the random nature of heuristic search algorithms, are justified by long execution cycles, having the algorithm to interact with humans or slow-response devices. On the other hand, the formal proposition of hypothesis and their evaluation through statistic inference tests is an area where experiments can be improved.

Regarding internal validity, we observe that major problems with SSBSE papers are related to lacking a proper discussion on the instrumentation of the source code and on the usage of real world instances in the experiments. Johnson [1] observes that the ability to reproduce most experiments may be hampered due to the lack of information about the code. Regarding the usage of real instances, it can be observed, the usage of real instances in SBSE experiments is still limited.

Regarding construct threats, major problems with SSBSE research papers are related to the selection of cost measure and the description of the underlying model

submitted to optimization. While the number of fitness function evaluations is considered as the reference measure to describe the cost of executing an algorithm, several papers use the number of iterations (which depends on population size and may be not comparable between different algorithms) or wall-clock time (which depends on computer characteristics and CPU/memory load) to measure this cost. Regarding the description of the underlying model subjected to optimization, SSBSE papers present the behavior formerly reported by [1], which indicates that most papers still lack a detailed discussion of their underlying models.

Finally, regarding external threats, major problems are related to the lack of instances varying in size and complexity in the experimental studies. A diverse set of instances is required to evaluate the scalability of a SBSE research proposal. Since SBSE is mostly concerned with large scale problems, such a scalability assessment is very important. However, we observe that few research papers report results for more than 2 instances of different size (43%) and complexity (17%). Moreover, not all studies present a clear description of the selected instances and a discussion on how to measure their size and complexity to observe their influence in the collected results.

5. Conclusions

In this paper we presented a list of validity threats that may affect the design of experimental studies aiming to evaluate search-based software engineering propositions. The proposed threats represent an extension from a former study on SBSE validity threats [4], according to a set of guidelines to organize empirical studies to evaluate algorithms [1]. To evaluate to which extent these threats are addressed in research papers, we propose a questionnaire to assess the proposed threats. We have conducted an analysis of 23 SBSE papers using the proposed questionnaire and concluded that while conclusion threats are well addressed by current papers, the assessment of internal, external, and construct threats can be severely improved.

6. Acknowledgements

The authors would like to express their gratitude for FAPERJ, CNPq, INCT-SEC, and FAPEAM, the research agencies that financially supported this project.

7. References

[1] D. S. Johnson, “A Theoretician's Guide to the Experimental Analysis of Algorithms”, *Data Structures, Near Neighbor Searches, and Methodology: Proceedings of 5th & 6th DIMACS Implementation Challenges,* American Mathematical Society, Providence, 2002, 215-250

[2] M. HARMAN, B.F. JONES, “Search-based Software Engineering”, Information and Software Technology, 2001, pp. 833-839

[3] M. Harman, “The Current State and Future of Search Based Software Engineering”, *Future of Software Engineering (FOSE’07),* 2007

[4] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, “A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation”, *IEEE Transactions on Software Engineering,* vol. 36 (6), November/December 2010, 2010, pp. 742-762.

[5] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, A. Wesslen, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 2000.