# Test Data Generation Based on GUI: A Systematic Mapping

Rodrigo Funabashi Jorge
Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Campo Grande, MS, Brazil
Email: rodrigo.funabashi@ufms.br

Márcio Eduardo Delamaro
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brazil
Email: delamaro@icmc.usp.br

Celso Gonçalves Camilo Junior
Instituto de Informática
Universidade Federal de Goiás
Goiânia, GO, Brazil
Email: celso@inf.ufg.br

Auri Marcelo Rizzo Vincenzi
Instituto de Informática
Universidade Federal de Goiás
Goiânia, GO, Brazil
Email: auri@inf.ufg.br

***Abstract*—For the general case, the complete automation of test data generation is an undecidable problem, and many researches employ meta-heuristics trying to find a reasonable partial solution. System testing performed via Graphical User Interface (GUI) imposes extra challenge for automation due to hundreds and often thousands of possibilities of events that can be generated. This work presents a study based on systematic mapping aiming at identifying the state of the art and the state of the practice on the automation of system testing carried out via GUI. We employed the traditional protocol of mapping study to support the data collection. The work was carried out from 6th February 2012 to 1st May 2013 resulting in the selection of 39 out of 598 primary studies obtained with the application of the search strings. Some of these works used, besides functional testing criteria, structural testing criteria to guide meta-heuristics. In relation to meta-heuristics, the distribution of work was more uniform, with a slight majority using Genetic Algorithms for test data generation. There are few research groups working on this subject. One particular author is responsible for authoring more than 30% of the selected primary studies and can be considered a reference in the generation of test data from GUI. Some research problems identified are 1) the difficult to represent all the possible GUI interactions without cause state explosion, 2) the need to evaluate the techniques on large software products, and 3) the complexity to automate the representation of the GUI interactions by reducing the number of infeasible sequences of actions.**

***Keywords–Systematic Mapping Study; System Testing; Testing through GUI; Automated Test Data Generation***

## I. INTRODUCTION

Modern software systems include various components that interact with each other to perform tasks. The correct behavior of these components is often verified by means of unit tests, integration, system and acceptance. In unit testing, the goal is to identify logic and implementation faults on each software unit. Integration testing is a systematic technique to integrate the software units in order to identify faults in the communication interface between them. The system test is performed after the system integration and aims to ensure that the software meets all functional, behaviour and performance requirements described in the specification. Finally, acceptance testing aims to verify whether the developed product meets the requirements specified by users [1]. Many of today's applications have a special component in the form of a Graphical User Interface (GUI). The GUI is composed by a set of control elements *widgets*, such as buttons, menu items and text boxes. The graphical interface is often the only piece of software which the user interacts. This way, it is necessary to test this interface in order to ensure product quality through the creation of test data in the form of input sequences. An input for a GUI application is a sequence of actions, such as clicking a button control or dragging and dropping GUI elements. It is observed that, in general, this type of test is performed during system and acceptance testing.

To generate a good set of test data, test designers should make sure that this set covers all the features of the system and, in the context of GUI also has exercised all possibilities of interface events. However, the difficulty in performing this task is twofold: to manage the domain size and to sequence the actions.

Within the first problem, unlike a system with a command line interface, a graphical user interface has many operations that need to be tested. A relatively small program, such as Microsoft WordPad has 325 possible GUI operations [2]. Therefore, the number of operations can easily be an order of extremely high magnitude for more complex programs [3].

Regarding the second issue, some functionality of the system can only be performed following a complex sequence of GUI events. For example, to open a file the user must enter the File menu and select the Open operation and then use a dialog box to specify the file name and complete the operation. Obviously, increasing the number of possible operations increases the sequencing problem exponentially, making it difficult to create test data manually.

Due to the difficulties related to the generation of test data which run via GUI, this work focused on the collection of primary studies in this context by applying a systematic mapping to verify what is available in the literature on this subject and which gaps can still be explored on further researches. In Section II, an explanation about GUI test, its features, limitations and related issues are presented. In Section III describes the planning, conduction, and analysis of the application of the systematic mapping. The conclusion is presented in Section IV.

## II. GUI TESTING

Graphical user interface is a type of interface that allows the user to interact with digital devices through graphical elements such as icons and other visual indicators, as opposed to the

 ISBN: 978-1-61208-367-4 

command line interface. The interaction is usually done via mouse or keyboard, on which the user is able to select symbols and manipulate them in order to get some practical result. These symbols are referred to *widgets* and are grouped into *kits*.

In our context, the testing via GUI means to perform a system or acceptance testing of a particular product to ensure it meets its specifications. This is normally done by using a range of test data.

To generate a good set of test data, designers should check that this set covers all the features of the system and, in the context of GUI also has exercised all possibilities of interface events. However, there are some open problems related to GUI testing [3]:

1) the huge amount of possible sequences from each state, i.e., in every state there are many alternative actions leading to an exceptionally large search domain. Furthermore, it is computationally expensive to generate and evaluate sequences, since the software needs to be started and all actions need to be performed in sequence. This requires efficient algorithms to explore the search space in an intelligent way to find optimal test sequences;
2) related to the generation of inputs for applications that explore button clicks and drag and drop operations components:
    a) map the GUI to determine the visible *widgets* and their properties. For example, the position of the components such as buttons and menu items;
    b) derive a set of permitted actions at each stage of implementation. For example, a visible button may be disabled and could not be pressed; and
    c) perform and record the test sequence making it possible to repeat (play) it again later.

To deal with the nature of the problems mentioned, the Artificial Intelligence is very applied, since these are optimization problems. Thus, the research area called *Search-based Software Engineering (SBSE)* has emerged, which deals with the application of mathematical optimization techniques to solve complex problems in the field of Software Engineering. According to the Software Engineering by Automated Search (SEBASE) website [4], that maintains a updated database about SBSE, 52% of the publications on SBSE focus on testing and debugging. This is due to the high cost of implementing these activities, which in general can spend 50% of the development cost [1]. Given this scenario a subarea of SBSE called *Search-Based Software Testing (SBST)* was created, focusing on the application of mathematical optimization techniques in solving problems in the context of testing activity. Therefore, the challenge is to automate the testing process as much as possible, and the generation of test data is, of course, an essential part of automation. Also according to the site SEBASE, only 6% of works in this area belong to Brazil.

## III. Application of Systematic Mapping

Systematic mapping is a type of literature review [5], in which it is conducted a broader review of primary studies to identify researches evidences and gaps, directing the focus of future systematic reviews, which tries to answer more specific research questions [5]. The systematic mapping study provides an overview of a research area, the amount, the type of research conducted, the results are available, in addition to the frequency of publications over time to identify trends [6]. There are many reasons for conducting a systematic mapping, the most common being [7]:

- to summarize the existing evidence regarding treatment or technology, for example, to summarize empirical evidence of the benefits and limitations of a specific method;
- to identify gaps in current research in order to suggest future areas of research; and
- to provide an overview/subsidy for advancing knowledge in new areas of research.

However, the mapping can also be used to examine the extent to which the empirical evidence supports/contradicts theoretical assumptions, or even to aid in the generation of new hypotheses [5], being composed of the following steps: planning, conducting and reporting [5].

### A. Planning

The planning of this systematic mapping, which describes the protocol that was established, was carried out from the adaptation of the protocol model presented by Petersen et. al. [6], that specifies the following elements [5]: research questions, search strategy and implementation, criteria for inclusion and exclusion, and data extraction and synthesis methods.

Research questions define the scope of the mapping. They guide the development of the remainder of the study and should be set according to the motivations of the study [6]. The research questions (RQ) were elaborated in order to find primary studies to understand and summarize evidences on the adoption of techniques for automatic generation of test data from GUI:

- $\mathbf{RQ}_1$: Do the techniques employed in GUI testing intend to cover a specific test criterion?
- $\mathbf{RQ}_{1.1}$: What is the test criterion?
- $\mathbf{RQ}_2$: What are the heuristics, techniques, algorithms or strategies used for automatic generation of test data from GUI?
- $\mathbf{RQ}_3$: Do the techniques for automatic generation of GUI test data require a data model that abstracts the GUI to perform the test generation?
- $\mathbf{RQ}_{3.1}$: If need, is the model generated automatically or manually?
- $\mathbf{RQ}_4$: What are the available tools and how do they support the automatic generation of test data from the GUI?
- $\mathbf{RQ}_5$: In what domain are automatically generating test data based on GUI applied?

Systematic mapping is a kind of secondary study in Software Engineering [8], identifying primary studies from several sources (databases). These sources can be classified into two main categories [8]: index engines and sites of editors. The index engines work with several publishers publications. One

 ISBN: 978-1-61208-367-4 

can cite SCOPUS as an example of index engines. The sites of editors refer to databases of online literature supplied by the editors to facilitate the recovery of the published literature. A popular site of editors in computing is the IEEE. However, as is the case of the ACM, some of these sources fall into two categories. The bases chosen in this study are ACM, IEEE, *Science Direct*, and SCOPUS, considered to be relatively efficient in conducting systematic reviews and mappings in the context of Software Engineering [9].

To build the search string, key concepts that wish to investigate were selected. From this, the synonyms, related terms and acronyms were identified. Related to the concept "Graphical User Interface" were *graphical user interface*, GUI e *web application*. For the concept "Automatic Test Data Generation" were *test data generation*, *test-data generation*, *generating test data*, *generate test data*, *automated testing*, *automation testing*, and *automation test*.

Based on the above key concepts, the default search string was built using the Boolean AND/OR connectors:

*("graphical user interface" OR "GUI" OR "web application") AND ("test data generation" OR "test-data generation" OR "generating test data" OR "generate test data" OR "automated testing" OR "automation testing" OR "automation test")*

A total of 10 articles were selected [10]–[19]. These articles have provided evidence that this search string is adequate, since all these items were returned after the application of search string in their respective bases.

To determine the relevancy of given primary study it must satisfy any Inclusion Criteria (IC) on the other hand it will be excluded by any Exclusion Criteria (EC). Our inclusion criteria are:

- **$IC_1$:** The study presents a case study or experience report using techniques for generating test data from GUI;
- **$IC_2$:** The study presents an investigation of the technical features to generate test data from GUI;
- **$IC_3$:** The study proposes methods for evaluating techniques for generating test data from GUI;
- **$IC_4$:** The study presents tools that use techniques to generate test data from GUI.

Primary studies considering different domains or presenting ideas in a vague way were excluded. To classify these studies the following exclusion criteria were identified:

- **$EC_1$:** The work is not related to any of the research questions;
- **$EC_2$:** The work was selected by another search string applied the same basis, sometimes with the keywords searched in the title, sometimes in the abstract. Thus, on these bases, the same work can be retrieved twice.
- **$EC_3$:** Lack of information about the work;
- **$EC_4$:** The work has already been selected by another source;
- **$EC_5$:** The work is not in English language.

Based on the inclusion and exclusion criteria, three stages were defined for the selection of works. The first was based on the analysis of keywords, title and abstract to decide whether the work may or may not be included. In the second stage, the introduction and conclusion were considered for analysis and third, the analysis was applied to the whole work.

For synthesis and extraction of data, some additional information to the research questions were collected, such as: which work focuses on web applications for generation of test data, whom authored the selected works, and what is the relation between the selected primary studies.

### B. Conducting

After the protocol specification, we started to apply the search strings on the selected databases. Observe that this step requires, sometimes, and adaptation of the default search string to satisfy specific constraints of a particular database search engine. The complete set of search strings were omitted for sake of space, but they can be found in [20]. The application and adaptation of search strings happened from 8 to 11 October 2012, which returned all the control articles previously mentioned in Section III-A. In Table I, one can get the number of items returned for each search string in each database. Columns $IC_n$ and $EC_n$ correspond to the inclusion and exclusion criteria defined in Section III-A, respectively. Numbers in these columns represent the total of primary studies included or excluded from analysis based on that specific inclusion or exclusion criterion.

Identification and selection of the work was performed in three steps: reading the title, keywords and abstract; reading the introduction and conclusion, and reading the whole paper. In the first step, using the JabRef [21], each work was analyzed by two experts applying the all the inclusion and some exclusion criteria ($EC_1$, $EC_2$, and $EC_3$), defined in Section III-A, thus helping organizing and cataloging the works. This analysis took place in the order of application of search strings, first considering the ones applied on titles (ACM-1 and IEEE-1) and subsequently the ones applied on abstracts (ACM-2 and IEEE-2), resulting in 98 works for the inclusion criteria representing 16,39% of the total.

However redundant works between databases had not yet been identified, that is, the exclusion criterion ($EC_4$) had not yet been applied. After the application of $EC_4$ and reading the introduction and conclusion (Step 2) the number of selected papers was reduced to 59, corresponding to 9,87% of the total, as shown in the summary presented in Table I. Observe that there is no primary study written on language different than English, therefore, the exclusion criterion $EC_5$ was not applied.

In the final selection process, third stage, studies were analyzed completely and thereafter 39 primary studies were selected to compose the mapping, 6,52% of the 598 primary studies initially selected. It is observed that this reduction rate is consistent with other surveys in the area [6][22].

### C. Analysis of the Results

Figure 1 presents the 39 selected primary studies organized by year as a directed graph. The arrows indicate a given primary study cite another one. Observe that from 1998 until the date of application of search strings, the majority of studies is concentrated in 2010, summing up 10 studies. However, the only primary study identified in 2001 was cited by 15 other works, and all studies from 2010 are referenced together by 8 other studies. Two studies that may be considered references

 ISBN: 978-1-61208-367-4 

TABLE I. Results of the Second Analysis of the Primary Studies

| Strings | IC1 | IC2 | IC3 | IC4 | EC1 | EC2 | EC3 | EC4 | Total IC (%) | Total EC (%) | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ACM-1 | 21 | 3 | 1 | 2 | 87 | 0 | 1 | 0 | 27 (23,48%) | 88 (76,52%) | 115 |
| ACM-2 | 2 | 0 | 0 | 0 | 7 | 15 | 0 | 0 | 2 ( 8,34%) | 22 (91,66%) | 24 |
| IEEE-1 | 10 | 1 | 0 | 1 | 53 | 0 | 2 | 13 | 12 (15,00%) | 68 (85,00%) | 80 |
| IEEE-2 | 1 | 0 | 0 | 0 | 5 | 19 | 0 | 1 | 1 ( 3,84%) | 25 (96,16%) | 26 |
| SCIENCE | 4 | 1 | 0 | 0 | 151 | 0 | 25 | 1 | 5 ( 2,75%) | 177 (97,25%) | 182 |
| SCOPUS | 11 | 0 | 1 | 0 | 134 | 0 | 1 | 24 | 12 ( 7,02%) | 159 (92,13%) | 171 |
| **AMOUNT** | 49 | 5 | 2 | 3 | 437 | 34 | 29 | 39 | 59 ( 9,87%) | 539 (90,13%) | 598 |

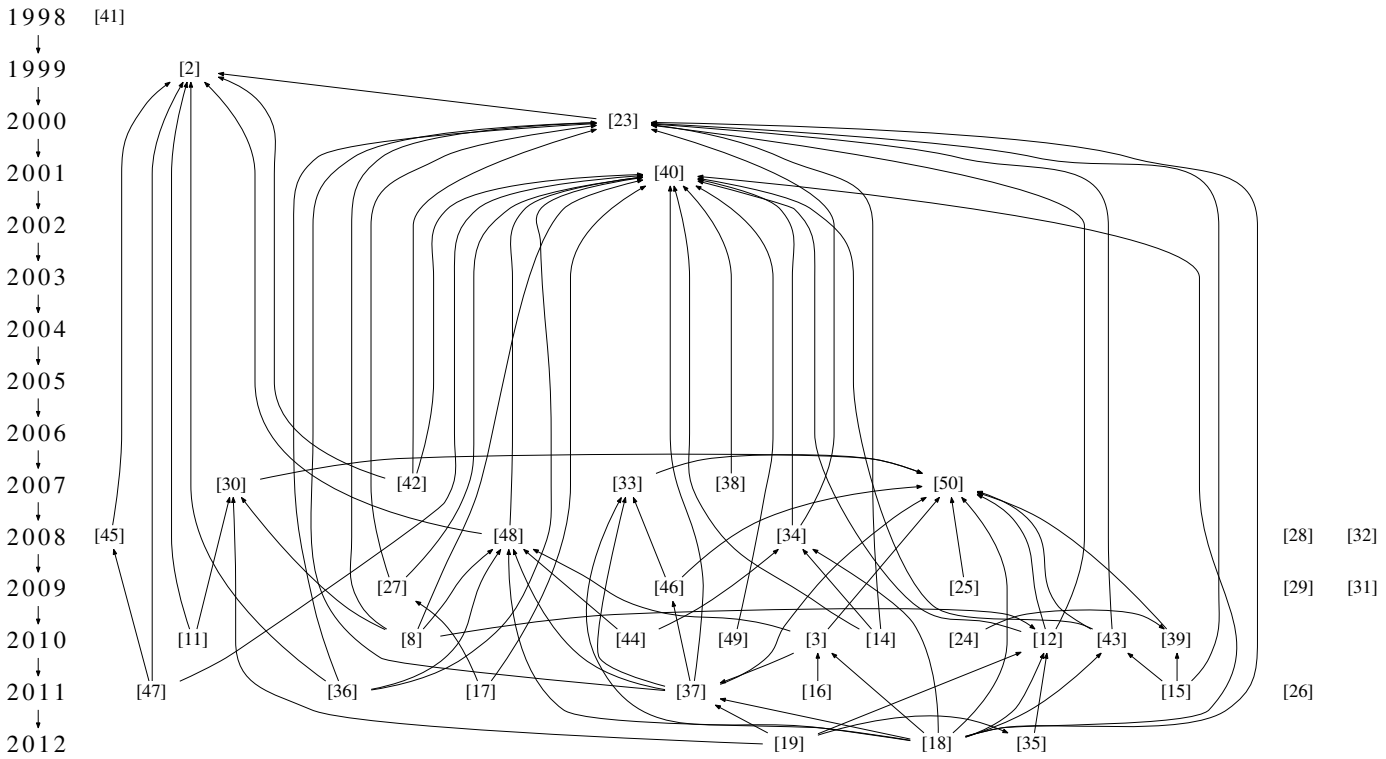


Figure 1. Distribution and Citation Between the Primary Studies

are the one authored by White and Almezen [23] and the one authored by Memon et. al. [2] with 11 and 15 citations each, respectively. The 39 selected primary studies involved 74 different authors of 27 affiliations (institutions) distributed in 13 countries. Most work within this context are authored by the same authors. An example is Dr. Atif M. Memon with participation, in approximately, 31% of the selected studies and can be considered a reference in the generation of test data from GUI. Highlight for the University of Maryland in the USA, appearing as an institution and country with more participation in the selected studies, 12 (30,8%) and 15 (38,5%), respectively.

Regarding the first research question $RQ_1$, the works do not identify a specific test criterion, but sometimes they mention which technique is used for generation of test data. Among the three traditional testing techniques, functional, structural, and fault-based, functional corresponds to 94,8%, since when the study did not mention what technique was used it was considered as functional since test data generation is based on GUI. Some works, however, besides the functional technique also use the source (structural technique) to guide its search technique or methodology for test data generation [15][26][27][48].

To answer the research question $RQ_2$, we analyzed the meta-heuristics and techniques generally used in studies for the generation of test data. In this case, the distribution of works was more uniform, with a small majority of 10 studies (25,6%) using Genetic Algorithms [3][14][26][32] [36][38][46][49].

Genetic algorithms are implemented as a computer simulation in which an initial population, in general randomly generated, representing a possible solution, is evolved to a better solution through iterations. The evolution occurs through generations. On each generation, the quality of the current solution in the population is evaluated, some individuals are selected for the next generation, and mutated or recombined to form a new population. The new population is then used as input to the next iteration of the algorithm.

Another meta-heuristic that was also used in three studies [16][18][34] was the Ant Colony Optimization. The algorithm was created by Marco Dorigo in 1992 [51] and was inspired in the behavior of ants searching for food. This is justified by the fact that a colony of insects are very organized and collective activities are carried out with self-organization. The idea is that ants move randomly in search of food, i.e., they conduct exploratory searches for possible solutions. Once one

 ISBN: 978-1-61208-367-4 

finds food, it returns to the nest depositing pheromone. The greatest amount of pheromone means that more ants found this path and deposited the pheromone, increasing the likelihood of this being the best or the shortest way. Thus, this path became a solution that was optimized according to the level of pheromone found in the trail.

Other meta-heuristics for generating test data were also used, as is the case studies [14] and [35] which applied *Q-Lerning* and the work of Raul et. al. [44] that used Particle Swarm Optimization. In addition to meta-heuristics, other techniques have been identified as the work [17] and [27] who have used ontologies.

A fact that has been observed and that should be explored is that several studies [14][15][29] need an initial model of the GUI's application to perform the generation of test data (research question $RQ_3$). Just the work of Mariani et al. [19] employed automatic generation of a model and produced the test data incrementally by traversing the GUI model of the application under test, requiring no human intervention. They used the *Q-Lerning*, a tool from the AI area that learns to interact with the application under test and to stimulate their functionality.

Responding to the fourth research question $RQ_4$, some tools that assist in generating test data were identified during the mapping. Most tools are complementary, i.e., they allow to obtain better results when combined. One of the most used tool in the selected studies was *GUI Testing Framework* (GUITAR) [45] which was used in 42% of the selected studies. This is a project supported by *National Science Foundation*, aiming at simplifying GUI testing by automatically generating test data to test the functionality of the program under test via the GUI. This tool is divided into four components that represent its main functions: *GUIRipper* that extracts information from the GUI of the application under testing; the *GUIStructure2Graph* that builds an event flow graph (EFG) with the GUI elements; the *TestCaseGenerator* which generates a set of test data based on the EFG (but without the use of meta-heuristics); and *GUIReplayer* responsible for running the program with the generated suite of tests. One of the studies that applied the tool was performed by Huang et al. [3]. They used genetic algorithms to correct invalid test data sets. The work consisted of two steps: generate a set of test and repair the test set containing viable sequences for this, used the EFG model generated by GUITAR.

The work of Mariani et al. [19] aims to implement and to evaluate a technique for generating test data focusing on interactive applications, i.e., applications that interact with users through a GUI or Web. The technique and tool developed and used are called AutoBlackTest, that works with the generation of a model and produces the test data incrementally by exercising the application under test. For this, it uses Q-Lerning, an optimization techniques in the area of Artificial Intelligence, that learns how to interact with the application under test to stimulate their functionality.

An important feature of this work is that the *AutoBlackTest* does not depend on an initial set for execution. The vast majority of current techniques depends on this data set and to generate GUI testing its works in two phases [37][40]: generates a model of the sequences of events that can be produced through the interaction with the GUI application under test; and generates a set of test data that covers sequences in the model.

The effectiveness of these techniques depends on the integrity of the original model. When the initial model is obtained by stimulation of the application under test with a simple sampling strategy that uses a subset of GUI actions to navigate and analyze the windows, the derived model is partial and incomplete [19]. Thus, the test data generated can ignore many interactions and windows not discovered in the initial phase.

To evaluate the proposal, Mariani et. al. [19] carried out a comparative empirical evaluation between *AutoBlackTest* with the GUITAR tool using four applications for *desktop* computers. In the empirical comparison between *AutoBlackTest* and GUITAR, when applied in sessions with 12 hours of testing, was conclusive that *AutoBlackTest* can generate test data that reach a higher code coverage and also reveals more flaws than GUITAR.

Finally, answering the research question $RQ_5$, most of the selected studies, approximately 95% use desktop application for generating test data from the GUI. Only the studies [26][31][32] applies the proposal in a Web context, thus showing that much can still be done in this area.

## IV. Conclusion

This study applied a systematic mapping of the literature between the years 1998 to 2012, on application of techniques for generating test data from the GUI of the application under test. 39 primary studies from different sources between regular and high-level conferences were selected, corresponding to 6,52% of the total number of studies identified by the search application. It was found that this percentage is justified due to two reasons: (1) there are many works that apply, evaluate and propose techniques to generate test data, but not using as reference the GUI; and (2) some works focus on generating test data to test the GUI itself and not use it as input for the generation of test data.

With respect to the five research question we investigated, we found that, in general, the proposed testing generation techniques employed most functional testing criteria for test set quality evaluation ($RQ_1$). In terms of meta-heuristic ($RQ_2$), Genetic Algorithm is employed in 10 out of 39 the primary studies, followed by ant colony employed in 3 out of 39 studies, and q-learn which were employed 2 out of 39 primary studies ($RQ_3$). In terms of automation ($RQ_4$) GUITAR (a tool to test GUI desktop applications) was used in 42% of the primary studies, reinforcing the result obtained by $RQ_5$ that almost all studies were performed in the context of desktop applications.

Therefore, based on primary studies identified and answers to the research questions, we can highlight some research areas in the context of testing from GUI to be explored, which is the main purpose of a systematic mapping study:

- development of a software environment that allows to abstract the GUI model automatically, providing subsidies so that test data can be run at any time and if there is a change or modification in the GUI, the model can be updated and reevaluated at any time;

 ISBN: 978-1-61208-367-4 

- conduction of experimental studies to compare the different test data generation techniques, identifying the main characteristics of each one;
- definition of a strategy to reduce the cost and increase efficiency in the generation of test data from GUI;
- adaptation of the representation of the techniques presented for generating test data from GUIs for Web applications; and
- conduction of systematic reviews considering more specific research questions about the use of meta-heuristics to support the automation of test data generation.

## ACKNOWLEDGMENT

The authors would like to thank the Brazilian funding agencies Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG), and Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul (FUNDECT) which support this work.

## REFERENCES

[1] G. J. Myers and C. Sandler, The Art of Software Testing. John Wiley & Sons, 2004.

[2] A. Memon, M. Pollack, and M. Soffa, "Using a goal-driven approach to generate test cases for guis," in Software Engineering, 1999. Proceedings of the 1999 International Conference on, May 1999, pp. 257–266.

[3] S. Huang, M. B. Cohen, and A. M. Memon, "Repairing gui test suites using a genetic algorithm," in Proceedings of the 2010 Third International Conference on Software Testing, Verification and Validation, ser. ICST'10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 245–254. [Online]. Available: http://dx.doi.org/10.1109/ICST.2010.39

[4] Y. Zhang, "Sbse repository," Página Web, Aug. 2013, disponível em: http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/. Acesso em: 12/12/2013.

[5] B. Kitchenham, S. Charters, D. Budgen, P. Brereton, M. Turner, S. Linkman, M. Jorgensen, E. Mendes, and G. Visaggio, "Guidelines for performing systematic literature reviews in software engineering," Software Engineering Group – School of Computer Science and Mathematics – Keele University, Keele, Staffs, ST5 5BG, UK, Tech. Rep. EBSE-2007-01, Jul. 2007.

[6] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering, ser. EASE'08. Swinton, UK, UK: British Computer Society, 2008, pp. 68–77. [Online]. Available: http://dl.acm.org/citation.cfm?id=2227115.2227123

[7] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, "Systematic review in software engineering," Systems Engineering and Computer Science Dept. - COPPE/UFRJ, Rio de Janeiro/RJ - Brazil, Technical Report RT-ES 679/05, 2005.

[8] L. Chen, M. A. Babar, and H. Zhang, "Towards an evidence-based understanding of electronic data sources," in International Conference on Evaluation and Assessment in Software Engineering (EASE2010). Keele, UK: BCS, Apr. 2010.

[9] 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 Trans. Softw. Eng., vol. 36, no. 6, Nov. 2010, pp. 742–762. [Online]. Available: http://dx.doi.org/10.1109/TSE.2009.52

[10] A. M. Memon, M. E. Pollack, and M. L. Soffa, "Hierarchical gui test case generation using automated planning," IEEE Trans. Soft. Eng., vol. 27, no. 2, Feb. 2001, pp. 144–155. [Online]. Available: http://dx.doi.org/10.1109/32.908959

[11] M. Cunha, A. Paiva, H. Ferreira, and R. Abreu, "Pettool: A pattern-based gui testing tool," in Software Technology and Engineering (ICSTE), 2010 2nd International Conference on, vol. 1, Oct. 2010, pp. 202–206.

[12] X. Yuan and A. M. Memon, "Generating event sequence-based test cases using gui runtime state feedback," IEEE Trans. Softw. Eng., vol. 36, no. 1, Jan. 2010, pp. 81–95. [Online]. Available: http://dx.doi.org/10.1109/TSE.2009.68

[13] ——, "Iterative execution-feedback model-directed gui testing," Inf. Softw. Technol., vol. 52, no. 5, May 2010, pp. 559–575. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2009.11.009

[14] A. Rauf, S. Anwar, M. A. Jaffer, and A. A. Shahid, "Automated gui test coverage analysis using ga," in Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations, ser. ITNG'10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1057–1062. [Online]. Available: http://dx.doi.org/10.1109/ITNG.2010.95

[15] S. Arlt, C. Bertolini, and M. Schäf, "Behind the scenes: An approach to incorporate context in gui test case generation," in Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, ser. ICSTW'11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 222–231. [Online]. Available: http://dx.doi.org/10.1109/ICSTW.2011.70

[16] S. Bauersfeld, S. Wappler, and J. Wegener, "An approach to automatic input sequence generation for gui testing using ant colony optimization," 2011, pp. 251–252.

[17] H. Li, H. Guo, F. Chen, H. Yang, and Y. Yang, "Using ontology to generate test cases for gui testing," Int. J. Comput. Appl. Technol., vol. 42, no. 2/3, Feb. 2011, pp. 213–224. [Online]. Available: http://dx.doi.org/10.1504/IJCAT.2011.045407

[18] Y. Huang and L. Lu, "Apply ant colony to event-flow model for graphical user interface test case generation," IET Software, vol. 6, no. 1, 2012, pp. 50–60.

[19] L. Mariani, M. Pezze, O. Riganelli, and M. Santoro, "Autoblacktest: Automatic black-box testing of interactive applications," Software Testing, Verification, and Validation, 2008 International Conference on, vol. 0, 2012, pp. 81–90.

[20] R. F. Jorge, "Geração de dados de teste a partir de gui: Um mapeamento sistemático," 2013, 24 junho 2014. [Online]. Available: http://www.inf.ufg.br/~auri/icsea2014/

[21] JabRef, "Ferramenta JabRef," Página do Projeto, Oct. 2013, disponível em: http://jabref.sourceforge.net/. Acesso em: 12/12/2013.

[22] E. EngstrÃm and P. Runeson, "Software product line testing a systematic mapping study," Information and Software Technology, vol. 53, 2011, pp. 2–13.

[23] L. White and H. Almezen, "Generating test cases for gui responsibilities using complete interaction sequences," in Proceedings of the 11th International Symposium on Software Reliability Engineering, ser. ISSRE'00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 110–124. [Online]. Available: http://dl.acm.org/citation.cfm?id=851024.856239

[24] C. Bertolini and A. Mota, "A framework for gui testing based on use case design," in Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, ser. ICSTW'10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 252–259. [Online]. Available: http://dx.doi.org/10.1109/ICSTW.2010.37

[25] S. Qian and F. Jiang, "An event interaction structure for gui test case generation," in Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, Aug. 2009, pp. 619–622.

[26] X. Peng and L. Lu, "A new approach for session-based test case generation by ga," in Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, May 2011, pp. 91–96.

[27] H. Li, F. Chen, H. Yang, H. Guo, W. C.-C. Chu, and Y. Yang, "An ontology-based approach for gui testing," in Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference - Volume 01, ser. COMPSAC'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 632–633. [Online]. Available: http://dx.doi.org/10.1109/COMPSAC.2009.92

 ISBN: 978-1-61208-367-4 

[28] X. Zhu, B. Zhou, J. Li, and Q. Gao, "A test automation solution on gui functional test," 2008, pp. 1413–1418. [Online]. Available: http://goo.gl/cxZ623

[29] Y. Hou, R. Chen, and Z. Du, "Automated gui testing for j2me software based on fsm," in Proceedings of the 2009 International Conference on Scalable Computing and Communications; Eighth International Conference on Embedded Computing, ser. SCALCOM-EMBEDDEDCOM'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 341–346. [Online]. Available: http://dx.doi.org/10.1109/EmbeddedCom-ScalCom.2009.67

[30] P. A. Brooks and A. M. Memon, "Automated gui testing guided by usage profiles," in Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, ser. ASE'07. New York, NY, USA: ACM, 2007, pp. 333–342. [Online]. Available: http://doi.acm.org/10.1145/1321631.1321681

[31] A. Shahzad, S. Raza, M. Azam, K. Bilal, Inam-Ul-haq, and S. Shamail, "Automated optimum test case generation using web navigation graphs," 2009, pp. 427–432. [Online]. Available: http://goo.gl/tiHCG8

[32] S. H. Kuk and H. S. Kim, "Automatic generation of testing environments for web applications," in Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02, ser. CSSE'08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 694–697. [Online]. Available: http://dx.doi.org/10.1109/CSSE.2008.1026

[33] X. Yuan, M. Cohen, and A. M. Memon, "Covering array sampling of input event sequences for automated gui testing," in Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, ser. ASE'07. New York, NY, USA: ACM, 2007, pp. 405–408. [Online]. Available: http://doi.acm.org/10.1145/1321631.1321695

[34] Y. Lu, D. Yan, S. Nie, and C. Wang, "Development of an improved gui automation test system based on event-flow graph," in Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02, ser. CSSE'08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 712–715. [Online]. Available: http://dx.doi.org/10.1109/CSSE.2008.1336

[35] G. Becce, L. Mariani, O. Riganelli, and M. Santoro, "Extracting widget descriptions from guis," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7212 LNCS, 2012, pp. 347–361. [Online]. Available: http://goo.gl/jmLntA

[36] A. Rauf, A. Jaffar, and A. Shahid, "Fully automated gui testing and coverage analysis using genetic algorithms," International Journal of Innovative Computing, Information and Control, vol. 7, no. 6, 2011, pp. 3281–3294. [Online]. Available: http://goo.gl/zsRadh

[37] X. Yuan, M. B. Cohen, and A. M. Memon, "Gui interaction testing: Incorporating event context," IEEE Trans. Softw. Eng., vol. 37, no. 4, Jul. 2011, pp. 559–574. [Online]. Available: http://dx.doi.org/10.1109/TSE.2010.50

[38] I. Alsmadi and K. Magel, "Gui path oriented test generation algorithms," in Proceedings of the Second IASTED International Conference on Human Computer Interaction, ser. IASTED-HCI'07. Anaheim, CA, USA: ACTA Press, 2007, pp. 216–219. [Online]. Available: http://dl.acm.org/citation.cfm?id=1698252.1698291

[39] C. Bertolini, A. Mota, E. Aranha, and C. Ferraz, "Gui testing techniques evaluation by designed experiments," in Proceedings of the 2010 Third International Conference on Software Testing, Verification and Validation, ser. ICST'10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 235–244. [Online]. Available: http://dx.doi.org/10.1109/ICST.2010.41

[40] A. M. Memon, M. L. Soffa, and M. E. Pollack, "Coverage criteria for gui testing," in Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering, ser. ESEC/FSE-9. New York, NY, USA: ACM, 2001, pp. 256–267. [Online]. Available: http://doi.acm.org/10.1145/503209.503244

[41] A. Beer, S. Mohacsi, and C. Stary, "Idatg: an open tool for automated testing of interactive software," in Computer Software and Applications Conference, 1998. COMPSAC'98. Proceedings. The Twenty-Second Annual International, Aug. 1998, pp. 470–475.

[42] M. Hayat and N. Qadeer, "Intra component gui test case generation technique," in Information and Emerging Technologies, 2007. ICIET 2007. International Conference on, Jul. 2007, pp. 1–5.

[43] X. Yuan and A. M. Memon, "Iterative execution-feedback model-directed gui testing," Inf. Softw. Technol., vol. 52, no. 5, May 2010, pp. 559–575. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2009.11.009

[44] R. Abdul, N. Ejaz, Q. Abbas, S. Rehman, and A. Shahid, "Pso based test coverage analysis for event driven software," 2010, pp. 219–224. [Online]. Available: http://goo.gl/7mdpDZ

[45] D. R. Hackner and A. M. Memon, "Test case generator for guitar," in Companion of the 30th international conference on Software engineering, ser. ICSE Companion'08. New York, NY, USA: ACM, 2008, pp. 959–960. [Online]. Available: http://doi.acm.org/10.1145/1370175.1370207

[46] X. Yuan, M. Cohen, and A. Memon, "Towards dynamic adaptive automated test generation for graphical user interfaces," in Software Testing, Verification and Validation Workshops, 2009. ICSTW '09. International Conference on, Apr. 2009, pp. 263–266.

[47] K. C. Chuang, C. S. Shih, and S. H. Hung, "User behavior augmented software testing for user-centered gui," in Proceedings of the 2011 ACM Symposium on Research in Applied Computation, ser. RACS'11. New York, NY, USA: ACM, 2011, pp. 200–208. [Online]. Available: http://doi.acm.org/10.1145/2103380.2103421

[48] Q. Xie and A. Memon, "Using a pilot study to derive a gui model for automated testing," ACM Transactions on Software Engineering and Methodology, vol. 18, no. 2, 2008, pp. 1–35. [Online]. Available: http://goo.gl/cXsdQi

[49] I. Alsmadi, "Using genetic algorithms for test case generation and selection optimization," 2010, pp. 1–4. [Online]. Available: http://goo.gl/xNZhRY

[50] X. Yuan and A. M. Memon, "Using gui run-time state as feedback to generate test cases," in Proceedings of the 29th international conference on Software Engineering, ser. ICSE'07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 396–405. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.94

[51] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," Evolutionary Computation, IEEE Transactions on, vol. 1, no. 1, 1997, pp. 53–66.

 ISBN: 978-1-61208-367-4 