Information and Software Technology 55 (2013) 1679–1694

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

# Graphical user interface (GUI) testing: Systematic mapping and repository

Ishan Banerjee [a], Bao Nguyen [a], Vahid Garousi [b,c,*], Atif Memon [a]

[a] Department of Computer Science, University of Maryland, College Park, MD 20742, USA
[b] Electrical and Computer Engineering, University of Calgary, Calgary, Canada
[c] Informatics Institute, Middle East Technical University, Ankara, Turkey

ARTICLE INFO

*Article history:*
Received 30 August 2012
Received in revised form 11 March 2013
Accepted 11 March 2013
Available online 5 April 2013

*Keywords:*
Systematic mapping
GUI application
Testing
Paper repository
Bibliometrics

ABSTRACT

*Context: GUI testing* is system testing of a software that has a graphical-user interface (GUI) front-end. Because system testing entails that the entire software system, including the user interface, be tested as a whole, during GUI testing, test cases—modeled as sequences of user input events—are developed and executed on the software by exercising the GUI's widgets (e.g., text boxes and clickable buttons). More than 230 articles have appeared in the area of GUI testing since 1991.
*Objective:* In this paper, we study this existing body of knowledge using a systematic mapping (SM).
*Method:* The SM is conducted using the guidelines proposed by Petersen et al. We pose three sets of research questions. We define selection and exclusion criteria. From the initial pool of 230 articles, published in years 1991–2011, our final pool consisted of 136 articles. We systematically develop a classification scheme and map the selected articles to this scheme.
*Results:* We present two types of results. First, we report the demographics and bibliometrics trends in this domain, including: top-cited articles, active researchers, top venues, and active countries in this research area. Moreover, we derive the trends, for instance, in terms of types of articles, sources of information to derive test cases, types of evaluations used in articles, etc. Our second major result is a publicly-accessible repository that contains all our mapping data. We plan to update this repository on a regular basis, making it a "live" resource for all researchers.
*Conclusion:* Our SM provides an overview of existing GUI testing approaches and helps spot areas in the field that require more attention from the research community. For example, much work is needed to connect academic model-based techniques with commercially available tools. To this end, studies are needed to compare the state-of-the-art in GUI testing in academic techniques and industrial tools.


**Contents**



* Corresponding author at: Electrical and Computer Engineering, University of Calgary, Calgary, Canada. Tel.: +1 403 210 5412.
*E-mail addresses:* ishan@cs.umd.edu (I. Banerjee), baonn@cs.umd.edu (B. Nguyen), vgarousi@ucalgary.ca (V. Garousi), atif@cs.umd.edu (A. Memon).

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.03.004

## 1. Introduction

Whenever the number of *primary studies*—reported in *articles* (we use the term *article* to include research papers, book chapters, dissertations, theses, published experimental results, and published demonstrations of techniques)—in an area grows very large, it is useful to summarize the body of knowledge and to provide an overview using a secondary study [1]. A secondary study [2–5] aggregates and objectively synthesizes the outcomes of the primary studies. By "mapping the research landscape," a secondary study helps to identify sub-areas that need more primary studies. Because the synthesis needs to have some common basis for extracting attributes in the articles, a side effect of the secondary study is that it encourages researchers conducting and reporting primary studies to improve their reporting standard of such attributes, which may include metrics, tools, study subjects, limitations, etc.

In the field of Software Engineering (SE), a systematic mapping (SM) study is a well-accepted method to identify and categorize research literature [6,1]. An SM [2,7–12] study focuses on building classification schemes and the results show frequencies of articles for classifications within the scheme. These results become one of the outputs of the SM in the form of a database or *map* that can be a useful descriptive tool itself. An SM uses established searching protocols and has rigorous inclusion and exclusion criteria.

In this paper, we leverage the guidelines set by Petersen et al. [1] and Kitchenham and Charters [13] to create an SM for the area of *GUI testing*. We define the term GUI testing to mean that a GUI-based application, i.e., one that has a graphical-user interface (GUI) front-end, is tested solely by performing sequences of events (e.g., *"click on button"*, *"enter text"*, *"open menu"*) on GUI *widgets* (e.g., *"button"*, *"text-field"*, *"pull-down menu"*). In all but the most trivial GUI-based systems, the space of all possible event sequences that may be executed is extremely large, in principle infinite, e.g., consider the fact that a user of Microsoft Word can click on the *File* menu an unlimited number of times. All GUI testing techniques are in some sense *sampling* the input space, either manually [14,15] or automatically [16,17]. In the same vein, techniques that develop a GUI *test oracle* [18]—a mechanism that determines whether a GUI executed correctly for a test input—are based on sampling the output space; examining the entire output, pixel by pixel, is simply not practical [19,20]. Techniques for evaluating the adequacy of GUI test cases provide some metrics to quantify the test cases [21–23]. Techniques for regression testing focus on retesting the GUI software after modifications [24–26].

The above is just one possible classification of GUI testing techniques. The goal of our SM is to provide a much more comprehensive classification of articles that have appeared in the area since 1991 (our search revealed that the first paper on GUI testing appeared in 1991). Given that now there are regular events such as the International Workshop on TESTing Techniques & Experimentation Benchmarks for Event-Driven Software (TESTBEDS) [27–30] in the area, we expect this number to increase. We feel that this is an appropriate time to discuss trends in these articles and provide a synthesis of what researchers think are limitations of existing techniques and future directions in the area. We also want to encourage researchers who publish results of primary studies to improve their reporting standards, and include certain attributes in their articles to help conduct secondary studies. Considering that many computer users today use GUIs exclusively and have encountered GUI-related failures, research on GUIs and GUI testing is timely and relevant.

There have already been 2 smaller, preliminary secondary studies on GUI testing. Hellmann et al. [31] presented a literature review of test-driven development of user interfaces; it was based on a sample of 6 articles. Memon and Nguyen [32] presented a classification of 33 articles on model-based GUI test-case generation techniques. To the best of our knowledge, there are no other secondary studies in the area of GUI testing.

In our SM, we study a total of 230 articles. We formulate 3 sets of research questions pertaining to the research space of GUI testing, demographics of the studies and authors, and synthesis and interpretation of findings. We describe the mechanisms that we used to locate the articles and the set of criteria that we applied to exclude a number of articles; in all we classify 136 articles. Our most important findings suggest that there is an increase in the number of articles in the area; there has been lack of evaluation and validation, although this trend is changing; there is insufficient focus on mobile platforms; new techniques continue to be developed and evaluated; evaluation subjects are usually non trivial, mostly written in Java, and are often tested using automated model-based tools; and by far a large portion of the articles are from the US, followed by China.

We have published our SM as an online repository on Google Docs [33]. Our intention is to periodically update this repository, adding new GUI testing articles as and when they are published. In the future, we intend to allow authors of articles to update the repository so that it can become a "live" shared resource maintained by the wider GUI testing community.

The remainder of this paper is structured as follows. Section 2 presents background on GUI testing. Section 3 presents our goals and poses research questions. The approach that we used to select articles is presented in Section 4. Section 5 presents the process used for constructing the systematic map. Sections 6–8 present the results of the systematic mapping. Section 9 presents a discussion of results. Finally, Section 10 presents related work and Section 11 concludes with a summary of our findings and future work.

## 2. Background

A GUI takes events (mouse clicks, selections, typing in text-fields) as input from users, and then changes the state of its widgets. GUIs have become popular because of the advantages this "event-handler architecture" offers to both developers and users [34,35]. From the developer's point of view, the event handlers may be created and maintained fairly independently; hence, complex systems may be built using these loosely coupled pieces of code. From the user's point of view, GUIs offer many degrees of usage freedom, i.e., users may choose to perform a given task by inputting GUI events in many different ways in terms of their type, number and execution order.