ABSTRACT

The current increased demand for distributed applications in domains such as web services and cloud computing has significantly increased interest in concurrent programming. This demand in turn has resulted in new testing methodologies for such systems, which take account of the challenges necessary to test these applications. This paper presents a systematic review of the published research related to concurrent testing approaches, bug classification and testing tools. A systematic review is a process of collection, assessment and interpretation of the published papers related to a specific search question, designed to provide a background for further research. The results include information about the research relationships and research teams that are working in the different areas of concurrent programs testing.

Categories and Subject Descriptors

D.2.5 [Software Engineering]: Testing and Debugging; D.1.3 [Programming Techniques]: Concurrent Programming—Threading, Message Passing

General Terms

Systematic Review, Concurrent Program, Software Testing

Keywords

Concurrent program testing, testing tools, bug classification

1. INTRODUCTION

Concurrent applications are inevitably more complex than sequential ones. All concurrent software contains features such as nondeterminism, synchronization and inter-process communication which significantly increase the difficulty of validation and testing. A number of research studies have been conducted in concurrent program testing, investigating new test mechanisms and adapting different approaches from the classical approaches used for sequential program testing.

This paper presents a mapping of this research, classifying the results into three main contributions: 1) works that propose a new approach, mechanism or framework for concurrent programs test; 2) works that present a taxonomy, classification or discussion of concurrent bugs; and 3) works that present a tool or methodology to support concurrent program testing. The systematic review process was used to collect, conduct and analyze the available published papers. A systematic review is a process of assessment and interpretation of all available studies related to a research question or subject of specific interest, providing a background for further investigation [21].

An understanding of the systematic review process and how to implement it is becoming a key requirement for all researchers. It is a powerful resource that, if used correctly, can contribute with new research insights in a particular area or can provide an initial overview of the research area for a new researcher.

2. SYSTEMATIC REVIEW: PLANNING AND CONDUCTING

This systematic review was performed according to the process defined by Kitchenham and Charters [21]. This process is composed of three phases: 1) planning - definition of a protocol that specifies the plan that the systematic review will follow; 2) conducting - execution of the protocol planned and 3) reporting - divulgation of the results. For reasons of space, the paper only outlines the relevant information to understand the systematic review process. The full review is available in [2].

Three research questions were formulated, setting out the objectives of the systematic review: 1) What testing approaches have been proposed to test concurrent programs? 2) What type of bug taxonomy related to concurrent programs has been identified? 3) What tools have been developed to
test concurrent programs? Based on these questions, a first version of the research string was defined. This string was refined with the aid of a list of already-known primary studies, established by the authors of this paper. The selection of primary studies is governed by inclusion and exclusion criteria, specified in the planning phase. The digital libraries considered in this review were: ACM Digital Library (portal.acm.org), IEEE eXplore (ieeexplore.ieee.org), SCOPUS (scopus.com) and CITESEER (citeseerx.ist.psu.edu).

Searches were performed in the digital libraries and 1166 studies were obtained. From these studies, two different selections were produced, using different inclusion and exclusion criteria. In the first selection, we read title, keywords, abstracts and, when necessary, the introduction and conclusion of each study, and we selected 314 studies. In the second selection, a complete reading of the papers was undertaken and selected 188 papers, which were classified according to the key search questions: testing approach (166), bug taxonomy or classification (6) and testing tool (50). A paper can be classified in two or more questions depending on its contribution.

3. SYSTEMATIC REVIEW: SOME RESULTS

Table 1 shows a summary of the key results from this systematic review. For each search question, the papers are classified based on the parallel programming paradigm (message passing or multithreaded), the proposed technique and the programming language. Several contributions are related to the testing approach definition, which present the proposition of the different testing techniques (mainly for multithreaded parallel programs). Most of the testing tools concentrate on multithreaded Java programs.

Figure 1 shows the relationship between some selected authors, from the systematic review. The diagram contains frames that represent authors group, according to their different research areas. Frame A presents authors researching into monitoring, scheduling, preemption and model checking. Frame B presents authors that work with in testing tools development and authors researching into mechanisms to detect concurrent bugs, in general, using concurrent programs benchmarks. Frame C presents authors that work with model-based testing, reachability testing and deterministic execution. Frame D presents authors that work with structural testing criteria and support tools for coverage test of concurrent programs.

4. CONCLUSION

In this paper we have presented the key results of a systematic review applied to find relevant works in concurrent programs testing. This review was developed using the systematic review process defined by Kitchenham and Charters [21]. The results obtained show different groups of authors working in important and challenging fields, such as: nondeterminism, synchronization interleaving, concurrent bugs, testing tool and coverage measure. This research addresses the challenges to testing concurrent programs presented by Yang [55] in 1999.

Another review result is the construction of a diagram showing the relationship among authors. This diagram illustrates the subjects of interest for each author, this highlighting the collaborative networks. The knowledge of the key topics that are being researched and the people working in each area is important for the establishment of new collaborations.

5. ACKNOWLEDGMENTS

The authors would like to thank the Brazilian funding agencies CAPES and FAPESP, for the financial support, under Capes process 1191/10-1 and FAPESP processes: 2008/04614-5, 2009/04517-2, 2010/04935-6, 2010/02839-0.

6. REFERENCES

Table 1: Summary of the papers selected in the systematic review

<table>
<thead>
<tr>
<th>Category</th>
<th>Paradigm</th>
<th>Technique</th>
<th>Language</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>testing approach</td>
<td>message-passing</td>
<td>analysis static</td>
<td>-</td>
<td>[10]</td>
</tr>
<tr>
<td>testing approach</td>
<td>message-passing</td>
<td>mutation and</td>
<td>-</td>
<td>[51, 20, 44, 15]</td>
</tr>
<tr>
<td></td>
<td>multithreaded</td>
<td>perturbation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testing approach</td>
<td>multithreaded</td>
<td>model checking</td>
<td>Java, C</td>
<td>[17, 31, 35, 59, 36]</td>
</tr>
<tr>
<td>testing approach</td>
<td>message-passing</td>
<td>reachability testing</td>
<td>-</td>
<td>[16, 32, 5, 28, 19, 29, 52]</td>
</tr>
<tr>
<td></td>
<td>multithreaded</td>
<td>structural testing</td>
<td>Java, C/MPI,</td>
<td>[56, 22, 24, 23, 25, 49, 57]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ada, C/Pthread</td>
<td>[54, 58, 18, 49, 24, 43, 50]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[48]</td>
</tr>
<tr>
<td>testing approach</td>
<td>multithreaded</td>
<td>deterministic testing</td>
<td>Java, Ada</td>
<td>[6, 39, 12, 46, 7]</td>
</tr>
<tr>
<td>testing approach</td>
<td>multithreaded</td>
<td>search-based testing</td>
<td>Java</td>
<td>27</td>
</tr>
<tr>
<td>testing tool</td>
<td>multithreaded</td>
<td>static and dynamic analyses</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>testing tool</td>
<td>multithreaded</td>
<td>race detection</td>
<td>-</td>
<td>[41, 45, 26]</td>
</tr>
<tr>
<td>testing tool</td>
<td>multithreaded</td>
<td>mutation testing</td>
<td>-</td>
<td>[15]</td>
</tr>
<tr>
<td>testing tool</td>
<td>multithreaded</td>
<td>exhaustive testing</td>
<td>Java</td>
<td>[37, 38]</td>
</tr>
<tr>
<td>testing tool</td>
<td>multithreaded</td>
<td>controlled execution</td>
<td>Java</td>
<td>[13, 11]</td>
</tr>
<tr>
<td>testing tool</td>
<td>multithreaded</td>
<td>deterministic testing</td>
<td>-</td>
<td>[39]</td>
</tr>
<tr>
<td>testing tool</td>
<td>message-passing</td>
<td>structural testing</td>
<td>Ada, C/PVM,</td>
<td>[56, 1, 47, 22, 40]</td>
</tr>
<tr>
<td></td>
<td>multithreaded</td>
<td></td>
<td>C/MPI</td>
<td></td>
</tr>
<tr>
<td>testing tool</td>
<td>message-passing</td>
<td>reachability testing</td>
<td>-</td>
<td>[8]</td>
</tr>
<tr>
<td>bug taxonomy</td>
<td>multithreaded</td>
<td>bug patterns</td>
<td>-</td>
<td>[14, 3, 4, 34]</td>
</tr>
<tr>
<td>bug taxonomy</td>
<td>multithreaded</td>
<td>interleaving sequences in software transactional memory</td>
<td>-</td>
<td>[33]</td>
</tr>
</tbody>
</table>

Figure 1: Diagram with the relationship among authors

Architectural support for programming languages and operating systems, ASPLOS ’09, pages 85–96, New York, NY, USA, 2009. ACM.


[16] X. Gong, Y. Wang, Y. Zhou, and B. Li. On testing multi-threaded java programs. In Software


[44] A. Sen and M. Abadir. Coverage metrics for verification of concurrent systemc designs using mutation testing. In IEEE International High Level Design Validation and Test Workshop (HLDVT),


