Research in Concurrent Software Testing: A Systematic Review

Simone R. S. Souza
ICMC/USP
Universidade de São Paulo
São Carlos, SP, Brazil
srocio@icmc.usp.br

Maria A. S. Brito
ICMC/USP
Universidade de São Paulo
São Carlos, SP, Brazil
masbrit@icmc.usp.br

Rodolfo A. Silva
ICMC/USP
Universidade de São Paulo
São Carlos, SP, Brazil
adamshuk@icmc.usp.br

Paulo S. L. Souza
ICMC/USP
Universidade de São Paulo
São Carlos, SP, Brazil
pssouza@icmc.usp.br

Ed Zaluska
ECS
University of Southampton
Southampton, England
ejz@ecs.soton.ac.uk

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*

*PADTAD'*11, July 17, 2011, Toronto, ON, Canada
Copyright 2011 ACM 978-1-4503-0809-0/11/05 ...$10.00

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

[1] A. Bechini, J. Cutajar, and C. Prete. A tool for testing of parallel and distributed programs in message-passing environments. In *Electrotechnical Conference, 1998. MELECON 98., 9th Mediterranean*, volume 2, pages 1308 –1312 vol.2, may 1998.

[2] M. A. S. Brito, K. Felizardo, P. S. L. Souza, and S. R. S. Souza. Concurrent software testing: A systematic review. Technical Report 359, ICMC/USP, 2010.

[3] J. Cao, A. T. S. Chan, S. C. F. Chan, and N. K. C. Cheung. A robust monitor construct with runtime fault detection. *Concurrency Computation Practice and Experience*, 18(5):471–500, 2006.

[4] J. Cao, N. Cheung, and A. Chan. Run-time fault detection in monitor based concurrent programming. In *Dependable Systems and Networks, 2001. DSN 2001. International Conference on*, pages 357 –366, july 2001.

[5] R. Carver and J. Lei. A stateful approach to testing monitors in multithreaded programs. In *High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on*, pages 54 –63, nov. 2010.

[6] R. Carver and K. C. Tai. Deterministic execution testing of concurrent ada programs. In *Proceedings of the conference on Tri-Ada '89: Ada technology in context: application, development, and deployment*, TRI-Ada '89, pages 528–544, New York, NY, USA, 1989. ACM.

[7] R. Carver and K.-C. Tai. Replay and testing for concurrent programs. *Software, IEEE*, 8(2):66 –74, mar 1991.

[8] R. H. Carver and Y. Lei. A general model for reachability testing of concurrent programs. *Lecture Notes in Computer Science*, 3308:76–98, 2004.

[9] J. Chen and S. MacDonald. Towards a better collaboration of static and dynamic analyses for testing concurrent programs. In *Proceedings of the 6th Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging 2008, PADTAD'08*, 2008.

[10] M. Christakis and K. Sagonas. Detection of asynchronous message passing errors using static analysis. *Lecture Notes in Computer Science*, 6539 LNCS:5–18, 2011.

[11] S. Copty and S. Ur. Multi-threaded testing with AOP is easy, and it finds bugs! In *Lecture Notes in Computer Science*, volume 3648, pages 740–749, 2005.

[12] J. Devietti, B. Lucia, L. Ceze, and M. Oskin. DMP: deterministic shared memory multiprocessing. In *Proceeding of the 14th international conference on*

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

Category	Paradigm	Technique	Language	References
testing approach	message-passing	analysis static	-	[10]
testing approach	message-passing multithreaded	mutation and perturbation	-	[51, 20, 44, 15]
testing approach	multithreaded	model checking	Java, C	[17, 31, 35, 59, 36]
testing approach	message-passing multithreaded	reachability testing	-	[16, 32, 5, 28, 19, 29, 52] [30, 42, 53]
testing approach	message-passing multithreaded	structural testing	Java, C/MPI, Ada, C/Pthread	[56, 22, 24, 23, 25, 49, 57] [54, 58, 18, 49, 24, 43, 50] [48]
testing approach	multithreaded	deterministic testing	Java, Ada	[6, 39, 12, 46, 7]
testing approach	multithreaded	search-based testing	Java	[27]
testing approach	multithreaded	static and dynamic analyses	-	[9]
testing tool	multithreaded	race detection	-	[41, 45, 26]
testing tool	multithreaded	mutation testing	-	[15]
testing tool	multithreaded	exaustive testing	Java	[37, 38]
testing tool	multithreaded	controled execution	Java	[13, 11]
testing tool	multithreaded	deterministic testing	-	[39]
testing tool	message-passing multithreaded	structural testing	Ada, C/PVM, C/MPI	[56, 1, 47, 22, 40]
testing tool	message-passing multithreaded	reachability testing	-	[8]
bug taxonomy	multithreaded	bug patterns	-	[14, 3, 4, 34]
bug taxonomy	multithreaded	interleaving sequences in software transactional memory	-	[33]

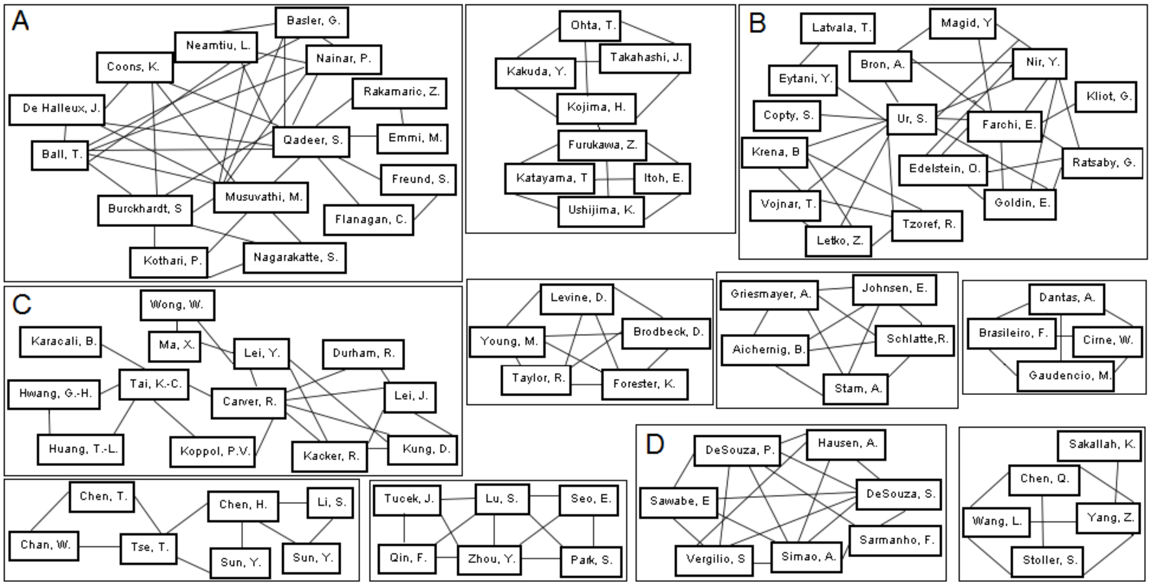

Figure 1: Diagram with the relationship among authors

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

[13] O. Edelstein, E. Farchi, E. Goldin, Y. Nir, G. Ratsaby, and S. Ur. Framework for testing multi-threaded java programs. *Concurrency Computation Practice and Experience*, 15(3-5 SPEC.):485–499, 2003.

[14] E. Farchi, Y. Nir, and S. Ur. Concurrent bug patterns and how to test them. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, page 7 pp., april 2003.

[15] M. Gligoric, V. Jagannath, and D. Marinov. Mutmut: Efficient exploration for mutation testing of multithreaded code. In *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, pages 55 –64, april 2010.

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

Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on, volume 1, pages 702 –706, aug 2007.

[17] S. Gradara, A. Santone, M. L. Villani, and G. Vaglini. Model checking multithreaded programs by means of reduced models. *Electronic Notes in Theoretical Computer Science*, 110:55–74, 2004.

[18] H.-F. Ho, G.-H. Chen, and T.-S. Kuo. Branch testing of concurrent programs using petri net models. *Computer Systems Science and Engineering*, 5(2):116–125, 1990.

[19] G.-H. Hwang, K.-C. Tai, and T.-L. Huang. Reachability testing: An approach to testing concurrent software. *International Journal of Software Engineering and Knowledge Engineering*, 5:493–510, 1995.

[20] V. Jagannath, M. Gligoric, S. Lauterburg, D. Marinov, and G. Agha. Mutation operators for actor systems. In *ICSTW 2010 - 3rd International Conference on Software Testing, Verification, and Validation Workshops*, pages 157–162, 2010.

[21] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University, 2007.

[22] H. Kojima, Y. Kakuda, J. Takahashi, and T. Ohta. A model for concurrent states and its coverage criteria. In *International Symposium on Autonomous Decentralized Systems, ISADS*, pages 1 –6, march 2009.

[23] P. V. Koppol, R. H. Carver, and K.-C. Tai. Incremental integration testing of concurrent programs. *IEEE Transactions on Software Engineering*, 28(6):607–623, 2002.

[24] P. V. Koppol and K.-C. Tai. An incremental approach to structural testing of concurrent software. In *Proceedings of the 1996 ACM SIGSOFT international symposium on Software testing and analysis*, ISSTA '96, pages 14–23, New York, NY, USA, 1996. ACM.

[25] H. Krawczyk and B. Wiszniewski. A method for determining testing scenarios for parallel and distributed software, 1996.

[26] B. Krena, Z. Letko, R. Tzoref, S. Ur, and T. Vojnar. Healing data races on-the-fly. In *Proceedings of the 2007 ACM Workshop on Parallel and Distributed Systems: Testing and Debugging, PADTAD'07*, pages 54–64, 2007.

[27] B. Krena, Z. Letko, T. Vojnar, and S. Ur. A platform for search-based testing of concurrent software. In *PADTAD 2010 - International Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging*, pages 48–58, 2010.

[28] Y. Lei and R. Carver. Reachability testing of semaphore-based programs. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 312 –317 vol.1, sept. 2004.

[29] Y. Lei and R. H. Carver. Reachability testing of concurrent programs. *IEEE Transactions on Software Engineering*, 32(6):382–403, 2006.

[30] Y. Lei, R. H. Carver, R. Kacker, and D. Kung. A combinatorial testing strategy for concurrent programs. *Software Testing Verification and Reliability*, 17(4):207–225, 2007.

[31] J. Li, D. Hei, and L. Yan. Correctness analysis based on testing and checking for openmp programs. In *ChinaGrid Annual Conference, 2009. ChinaGrid '09. Fourth*, pages 210 –215, aug. 2009.

[32] S. Q. Li, H. Y. Chen, and Y. X. Sun. A framework of reachability testing for java multithread programs. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 3, pages 2730 – 2734 vol.3, oct. 2004.

[33] J. Lourenço and G. Cunha. Testing patterns for software transactional memory engines. In *Proceedings of the 2007 ACM workshop on Parallel and distributed systems: testing and debugging*, PADTAD '07, pages 36–42, New York, NY, USA, 2007. ACM.

[34] S. Lu, S. Park, E. Seo, and Y. Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. *SIGOPS Oper. Syst. Rev.*, 42:329–339, March 2008.

[35] M. Musuvathi and S. Qadeer. Iterative context bounding for systematic testing of multithreaded programs. *SIGPLAN Not.*, 42:446–455, June 2007.

[36] M. Musuvathi and S. Qadeer. Fair stateless model checking. In *In PLDI 08: Programming Language Design and Implementation*, 2008.

[37] M. Musuvathi, S. Qadeer, and T. Ball. CHESS: A systematic testing tool for concurrent software. Technical report, MSR-TR-2007-149, Microsoft Research, 2007.

[38] M. Musuvathi, S. Qadeer, T. Ball, G. Basler, P. A. Nainar, and I. Neamtiu. Finding and reproducing heisenbugs in concurrent programs. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, OSDI'08, pages 267–280, Berkeley, CA, USA, 2008. USENIX Association.

[39] M. Olszewski, J. Ansel, and S. Amarasinghe. Kendo: efficient deterministic multithreading in software. *SIGPLAN Not.*, 44:97–108, March 2009.

[40] M.-Y. Park, S. J. Shim, Y.-K. Jun, and H.-R. Park. MPIRace-check: Detection of message races in MPI programs. *Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4459 LNCS:322–333, 2007.

[41] E. Pozniansky and A. Schuster. Efficient on-the-fly data race detection in multithreaded C++ programs. In *Parallel and Distributed Processing Symposium*, page 8 pp., april 2003.

[42] F. Pu and H.-Y. Xu. A feasible strategy for reachability testing of internet-based concurrent programs. In *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pages 1559 –1564, april 2008.

[43] F. S. Sarmanho, P. S. L. Souza, S. R. S. Souza, and A. Simão. Structural testing for semaphore-based multithread programs. *Lecture Notes in Computer Science*, 5101 LNCS:337–346, 2008.

[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)*,

pages 75 –81, june 2010.

[45] K. Sen and G. Agha. A race-detection and flipping algorithm for automated testing of multi-threaded programs. In *Haifa Verification Conference*, pages 166–182, 2006.

[46] H.-S. Seo, I. S. Chung, B. M. Kim, and Y. R. Kwou. The design and implementation of automata-based testing environment for java multi-thread programs. In *Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific*, pages 221 – 228, dec. 2001.

[47] P. L. Souza, E. T. Sawabe, A. S. Simão, S. R. Vergilio, and S. R. S. Souza. ValiPVM - a graphical tool for structural testing of PVM programs. *Lecture Notes in Computer Science*, 5205 LNCS:257–264, 2008.

[48] S. Souza, S. R. Vergilio, P. S. L. Souza, A. Simão, and A. C. Hausen. Structural testing criteria for message-passing parallel programs. *Concurrency Computation Practice and Experience*, 20(16):1893–1916, 2008.

[49] J. Takahashi, H. Kojima, and Z. Furukawa. Coverage based testing for concurrent software. In *28th International Conference on Distributed Computing Systems, ICDCS '08*, pages 533 –538, june 2008.

[50] R. N. Taylor, D. L. Levine, and C. D. Kelly. Structural testing of concurrent programs. *IEEE Transactions on Software Engineering*, 18(3):206–215, 1992.

[51] R. Vuduc, M. Schulz, D. Quinlan, B. D. Supinski, and A. Sæbjørnsen. Improving distributed memory applications testing by message perturbation. In *In Proceedings of Parallel and Distributed Systems: Testing and Debugging (PADTAD)*, 2006.

[52] W. E. Wong and Y. Lei. Reachability graph-based test sequence generation for concurrent programs. *International Journal of Software Engineering and Knowledge Engineering*, 18(6):803–822, 2008.

[53] W. E. Wong, Y. Lei, and X. Ma. Effective generation of test sequences for structural testing of concurrent programs. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, pages 539–548, 2005.

[54] Yang, , C. s. D. Yang, and L. L. Pollock. An algorithm for all-du-path testing coverage of shared memory parallel programs. In *In Sixth Asian Test Symposium*, pages 263–268, 1997.

[55] C.-S. D. Yang. *Program-Based, Structural Testing of Shared Memory Parallel Programs*. PhD thesis, University of Delaware, 1999.

[56] C.-S. D. Yang, A. L. Souter, and L. L. Pollock. All-du-path coverage for parallel programs. In *Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis*, ISSTA '98, pages 153–162, New York, NY, USA, 1998. ACM.

[57] R.-D. Yang and C.-G. Chung. A path analysis approach to concurrent program testing. In *Computers and Communications, 1990. Conference Proceedings., Ninth Annual International Phoenix Conference on*, pages 425 –432, mar 1990.

[58] R.-D. Yang and C.-G. Chung. Path analysis testing of concurrent programs. *Information and Software Technology*, 34(1):43–56, 1992.

[59] Y. Yang, X. Chen, and G. Gopalakrishnan. Inspect: A runtime model checker for multithreaded C programs. Technical report, UUCS-08-004, School of Computing, University of Utah, 2008.