Anais

Volume 02
ISSN 2178-6097

WTDSoft 2014

IV WORKSHOP DE TESES E DISSERTAÇÕES DO CBSOFT

COORDENADOR DO COMITÊ DE PROGRAMA
Eduardo Santana de Almeida - Universidade Federal da Bahia (UFBA)

COORDENAÇÃO DO CBSOFT 2014
Baldoino Fonseca - Universidade Federal de Alagoas (UFAL)
Leandro Dias da Silva - Universidade Federal de Alagoas (UFAL)
Márcio Ribeiro - Universidade Federal de Alagoas (UFAL)

REALIZAÇÃO
Universidade Federal de Alagoas (UFAL)
Instituto de Computação (IC/UFAL)

PROMOÇÃO
Sociedade Brasileira de Computação (SBC)

PATROCÍNIO
CAPES, CNPq, INES, Google

APOIO
Instituto Federal de Alagoas, Aloo Telecom, Springer, Secretaria de Estado do Turismo AL, Maceió Convention & Visitors Bureau, Centro Universitário CESMAC e Mix Cópia

Definição de um Framework para Avaliação Sistemática de Técnicas de Teste no Contexto de Programação Concorrente

Silvana M. Melo[1], **Orientadora: Simone R. S. Souza**[1]

[1] Instituto de Ciências Matemáticas e de Computação (ICMC) –
Universidade de São Paulo (ICMC/USP)
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

{morita,srocio}@icmc.usp.br

Resumo. *Este artigo apresenta um projeto de doutorado em andamento cujo objetivo é investigar as principais alternativas para a atividade de teste que têm sido empregadas no contexto de aplicações concorrentes, a fim de desenvolver um framework que auxilie a avaliação sistemática dessas técnicas, facilitando a realização de estudos futuros e, principalmente, ajudando os profissionais da academia e da indústria a selecionar a melhor técnica/ferramenta de teste considerando suas necessidades e diversos fatores de qualidade como satisfação, custo, eficácia e eficiência.*

1. Fundamentação Teórica

Com objetivo de identificar e eliminar defeitos presentes no produto de software, atividades de VV&T (Verificação, Validação e Teste) visam garantir a qualidade do software produzido. Dentre essas, o teste de software é uma atividade de garantia de qualidade que tem como objetivo identificar defeitos no produto em teste. A atividade de teste consiste de uma análise dinâmica do produto e é uma atividade relevante para a identificação e eliminação de erros que persistem.

No contexto de programas tradicionais (ou programas com características sequenciais), foram identificadas ao longo dos anos várias iniciativas de técnicas e critérios para a validação, as quais consideram duas questões importantes em relação à atividade de teste: 1) seleção de casos de teste, e 2) avaliação da adequação dos casos de testes em relação ao programa em teste [Beizer 1990, Coward 1988, DeMillo et al. 1978, Herman 1976, Laski and Korel 1983, Maldonado 1991, Mathur and Wong 1993, Rapps and Weyuker 1985]. Outra tendência é explorar e adaptar esses conceitos de teste para outros domínios de aplicação, tais como: Sistemas de Informação, Aplicações Concorrentes/Distribuídas, Sistemas de Tempo Real e Sistemas Embarcados.

Diferentemente dos programas tradicionais, a computação distribuída envolve programas concorrentes que interagem para realizar as tarefas. Essa interação pode ocorrer de forma sincronizada ou não, na qual esses programas (ou processos) podem ou não concorrerem pelos mesmos recursos computacionais. A construção de processos concorrentes requer o uso de primitivas para: definir quais processos serão executados em paralelo, iniciar e finalizar processos concorrentes e a garantir a coordenação entre os processos concorrentes enquanto estes estiverem executando [Almasi and Gottlieb 1989]. Desses três tipos de primitivas, destacam-se as primitivas necessárias à interação (comunicação e sincronização) entre os processos, devido à sua frequência de utilização, bem

como impacto no desempenho final do programa concorrente. A construção de programas concorrentes pode seguir dois diferentes paradigmas, classificados de acordo com a memória disponível [Almasi and Gottlieb 1989]. O paradigma de memória distribuída estabelece a comunicação e a sincronização entre os processos por meio da passagem de mensagens, no qual cada processo é executado por uma unidade de processamento que possui seu próprio espaço de endereçamento. No paradigma de memória compartilhada, a sincronização é obtida com a utilização de semáforos e monitores e variáveis compartilhadas são usadas para permitir a comunicação entre os processos que compartilham o mesmo espaço de endereçamento. Esses dois paradigmas apresentam características diferentes, as quais necessitam ser consideradas durante os testes.

O teste de aplicações concorrentes é mais complicado, pois além das dificuldades inerentes à atividade de teste, novos desafios são impostos devido principalmente ao comportamento não determinístico no qual diferentes sequências de sincronização podem ser obtidas com utilização de uma mesma entrada. Diversos trabalhos relevantes têm sido propostos na área de teste de programas concorrentes e eles podem ser classificados em diferentes abordagens, considerando por exemplo: análise estática ou dinâmica [Chen et al. 2009], paradigma de passagem de mensagem [Silva et al. 2005] ou memória compartilhada [Yang and Pollock 2003] e considerando a linguagem e/ou o uso de padrões [Sadowski and Yi 2009, Farchi et al. 2003].

Embora existam diversas propostas de técnicas e ferramentas para o teste de programas concorrentes, não há ainda uma maneira de auxiliar os profissionais que trabalham com programas concorrentes na indústria ou na academia, sobre qual dentre as inúmeras abordagens de teste existentes seria mais apropriada para as suas necessidades. Informações sobre custo-benefício, alcançabilidade e limitações são necessárias para a escolha da estratégia de teste adequada.

Segundo kitchenham et al. [Kitchenham et al. 2004], a proposta de novas metodologias pela academia, por si só, não é suficiente para a melhoria da qualidade do processo de software. Há que se considerar evidências sobre potenciais benefícios, limitações, custo e riscos associados a sua implantação na academia ou na indústria.

2. Caracterização do Problema

Dado o panorama atual das pesquisas desenvolvidas para o teste de programas concorrentes como apresentado na revisão sistemática em [Souza et al. 2011], pode-se notar a relevância e diversidade das pesquisas que propõe abordagens de teste para o software concorrente. Porém, até o momento nenhum trabalho que propõe uma metodologia de avaliação de técnicas de teste para a área de programação concorrente foi encontrado.

Nesse sentido, esta proposta de doutorado visa a investigar a definição de um framework de apoio para avaliação de técnicas e ferramentas de teste no contexto de programação concorrente. Tendo como base os trabalhos de Vos et al. [Vos et al. 2012] e Shull et al. [Shull et al. 2001], esse framework deverá apoiar os desenvolvedores de software concorrente na indústria e na academia, na escolha dos mecanismos de teste mais apropriados para os seus interesses e suas restrições de tempo/custo.

3. Metodologia e Estado Atual do Trabalho

Para o desenvolvimento deste projeto serão previamente reunidos e classificados os principais trabalhos que propõem metodologias de teste para o contexto da programação concorrente. A fim de auxilar a avaliação dessas metodologias, um framework será definido. Para isso, a metodologia proposta por Vos et al. [Vos et al. 2012] será instanciada para o contexto de programação concorrente, o que facilitará a realização de estudos secundários e formação de um corpo de evidência que juntos irão compor o framework, proporcionando uma base de conhecimento sólida, funcionando como base de pesquisa e norteando a avaliação sistemática de ferramentas/métodos/técnicas de teste propostos para este contexto.

Sendo assim, a metodologia de desenvolvimento do projeto pode ser dividida em quatro grandes fases, descritas a seguir:

- **Realização de uma revisão de literatura**: a fim de identificar, reunir e analisar pesquisas relacionadas às metodologias utilizadas para o teste de programas concorrentes, disponíveis na literatura. A replicação da revisão sistemática apresentada em [Souza et al. 2011], incluindo trabalhos mais recentes publicados na academia é um dos objetivos nessa fase.
- **Seleção e estudo das tecnologias de teste a serem investigadas**: com o objetivo de identificar as principais tecnologias propostas por pesquisadores e desenvolvedores no contexto do teste de programas concorrentes, trabalhos que disponibilizam metodologias consolidadas de teste serão selecionados com base nos resultados obtidos com a realização da revisão sistemática. Tais trabalhos serão classificados de acordo com suas contribuições.
- **Instanciação da metodologia proposta em [Vos et al. 2012] para o contexto de programação concorrente**: a metodologia de avaliação de técnicas e ferramenta de teste deve ser instanciada a fim de apoiar técnicas de teste que tratam características específicas das aplicações concorrentes, como: não determinismo, comunicação e sincronização de processos e *threads*.
- **Construção e disponibilização do corpo de conhecimento**: todos os resultados obtidos por meio da realização deste trabalho deverão ser reunidos para construção de um corpo de evidências disponibilizado como base de conhecimento.

Atualmente o trabalho encontra-se na fase de seleção e classificação das tecnologias a serem investigadas. Um mapeamento sistemático está sendo conduzido com objetivo de identificar as principais técnicas/ferramentas que têm sido propostas para o teste de programa concorrentes que servirão como base para o framework proposto. Um artigo científico com os resultados obtidos no mapeamento sistemático está sendo produzido e deverá ser submetido a um periódico ou evento da área.

4. Trabalhos Relacionados

Diversos trabalhos têm proposto técnicas e ferramentas para o teste de software concorrente, e tratam as mais diferentes abordagens, como: injeção de falhas [Artho et al. 2006], verificação formal [Wu et al. 2009], desenvolvimento dirigido a teste [Jagannath et al. 2010], execução controlada [Ball et al. 2009], teste de mutação [Gligoric et al. 2013], verificação de modelos [Yang et al. 2008], teste baseado em modelos [Aichernig et al. 2009], teste estrutural [Souza et al. 2014], análise simbólica

[Kundu et al. 2010], teste baseado em busca [Krena et al. 2010], teste de cobertura de *interleaving* [Yu et al. 2012], teste de concorrência probabilística [Burckhardt et al. 2010], teste de alcançabilidade [Lei and Carver 2006], métricas de cobertura [Deng et al. 2013] e geração de casos de teste [Xiaoan et al. 2009].

Tendo em vista a vasta diversidade de técnicas existentes, torna-se complexa a tarefa de escolha da técnica mais adequada a um determinado projeto. Nesse sentido, alguns trabalhos na área da programação sequencial tem explorado a avaliação de técnicas e ferramentas a fim de classificar quais apresentam maiores benefícios quando aplicadas a um determinado contexto [Kitchenham et al. 1995, Vos et al. 2012, Vegas et al. 2006, Pilar et al. 2014]. Esses estudos proporcionam aos profissionais de teste uma maneira de selecionar técnicas e ferramentas de maneira mais sistemática e metodológica, tomando como base evidências científicas.

Vos et al. [Vos et al. 2012] propõem um framework que disponibiliza uma metodologia para facilitar a realização de estudos de caso para avaliar e comparar técnicas e ferramentas de teste. A Figura 1 ilustra a metodologia, que propõe um framework genérico para auxiliar a instanciação de estudos primários (casos de teste) para avaliar técnicas e ferramentas de teste. Os estudos primários são reunidos em um corpo de evidência, que pode ser usado em estudos secundários (revisões de literatura) a fim de auxiliar os profissionais de teste a tomar decisões informadas sobre o uso de uma técnica e estimar o tempo/esforço que é necessário para sua implantação. O principal objetivo da metodologia é gerar estudos empíricos suficientes para servir como base de experiências documentadas e que sigam os princípios da Engenharia de Software Baseada em Evidências (ESBE), uma abordagem que procura integrar as melhores evidências de pesquisas com experiências práticas e valores humanos para auxiliar no processo de tomada de decisão sobre o desenvolvimento e manutenção de software [Kitchenham et al. 2004].

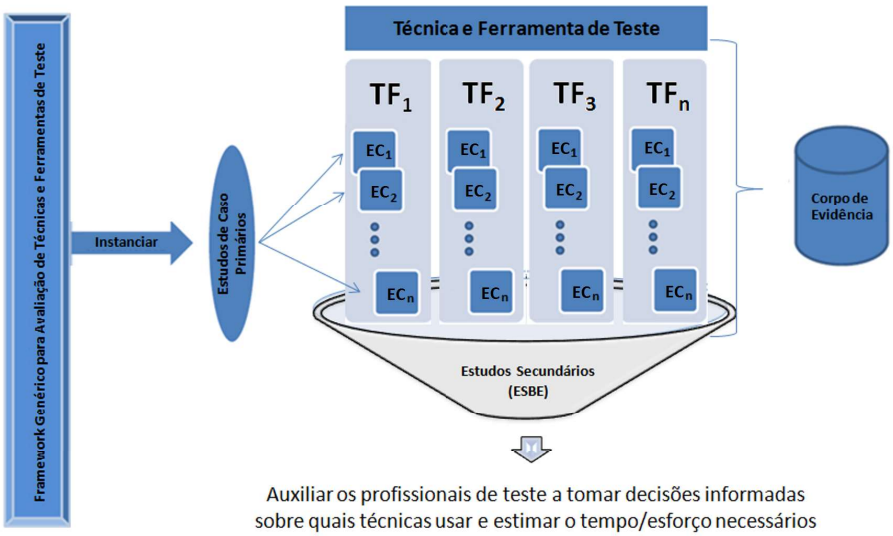

Figura 1. Metodologia para criação de um corpo de evidência [Vos et al. 2012].

O framework proposto por Vos et al. [Vos et al. 2012] define uma metodologia de avaliação de técnicas de teste apenas para o contexto da programação sequencial. Há a necessidade de extender o framework para o contexto da programação concorrente, pois

características específicas deste tipo de aplicação que influenciam diretamente na atividade de teste devem ser consideradas durante sua avaliação. Características como: o comportamento não determinístico, o elevado número de sincronizações a serem exercitadas durante o teste; a necessidade de detectar e corrigir erros típicos desse tipo de software, como: deadloks, condições de disputa, interferência e livelocks, não são encontradas em aplicações sequenciais e portanto não suportadas pelo framework proposto em [Vos et al. 2012].

5. Resultados Esperados

Com o desenvolvimento deste projeto de doutorado esperam-se as seguintes contribuições e resultados:

1. Desenvolvimento de um framework para auxiliar a avaliação sistemática de técnicas de teste para o contexto de programas concorrentes.
2. Disponibilização do material gerado nas avaliações conduzidas a fim de facilitar a realização de estudos secundários e comparações entre tecnologias de teste.
3. Contribuir para a transferência de tecnologia gerada na academia para centros desenvolvedores de software concorrente na academia e indústria.
4. Ampliar o uso do teste, hoje fortemente concentrado na academia, permitindo com isso a geração de novas demandas de teste, as quais realimentarão as pesquisas feitas formando-se um ciclo no desenvolvimento de novos conhecimentos.
5. Possibilitar o desenvolvimento de aplicações distribuídas com maior qualidade, através da criação de um framework de aplicação das técnicas de VV&T considerando diversos fatores como satisfação, custo, eficácia e eficiência.
6. Formação de recursos humanos em VV&T de aplicações distribuídas considerando desenvolver 1 projeto de iniciação científica atrelado ao projeto em questão e desdobramento para novos projetos de iniciação científica, mestrado e doutorado.

6. Avaliação dos resultados

A avaliação da metodologia proposta deverá ser feita por meio da condução de estudos empíricos na academia e indústria, visando aumentar a compreensão dos pesquisadores, tornando o processo de desenvolvimento de software mais previsível e contribuindo para o refinamento e aperfeiçoamento da tecnologia, auxiliando a busca por melhores práticas de teste. Segundo Shull et al. [Shull et al. 2001] a metodologia de avaliação de transferência de uma nova tecnologia para indústria deve sempre partir da definição conceitual até sua aplicação prática, com equilíbrio constante entre as necessidades de pesquisa e da indústria.

A proposta de utilização da metodologia no contexto industrial tem por objetivo a avaliação da aplicabilidade das abordagens definidas no ambiente acadêmico também em um ambiente prático real, principalmente considerando tecnologias que propõem ferramentas de suporte ao teste que podem atender as necessidades da indústria. Parcerias estão sendo estudadas entre profissionais da área de teste que trabalham com software concorrente tanto dentro da academia, quanto no âmbito industrial. Empresas parceiras de projetos relacionados a este trabalho são possíveis candidatas nessa tarefa de colaboração para validação do trabalho e devem ser contatadas durante a fase de definição e validação da metodologia.

Agradecimento

Os autores gostariam de agradecer a FAPESP, pelo apoio financeiro (processo número: 2013/05046-9).

Referências

Aichernig, B., Griesmayer, A., Schlatte, R., and Stam, A. (2009). Modeling and testing multi-threaded asynchronous systems with Creol. *ENTCS*, 243:3–14.

Almasi, G. S. and Gottlieb, A. (1989). *Highly parallel computing*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA.

Artho, C., Biere, A., and Honiden, S. (2006). Enforcer - efficient failure injection. *LNCS*, 4085:412–427.

Ball, T., Burckhardt, S., Coons, K. E., Musuvathi, M., and Qadeer, S. (2009). Preemption sealing for efficient concurrency testing. Technical report, Microsoft.

Beizer, B. (1990). *Software testing techniques*. Van Nostrand Reinhold Co., New York, NY, USA, 2 edition.

Burckhardt, S., Kothari, P., Musuvathi, M., and Nagarakatte, S. (2010). A randomized scheduler with probabilistic guarantees of finding bugs. In *ASPLOS*, pages 167–178, Pittsburgh, PA.

Chen, Q., Wang, L., Yang, Z., and Stoller, S. D. (2009). Have: Detecting atomicity violations via integrated dynamic and static analysis. In Chechik, M. and Wirsing, M., editors, *FASE*, volume 5503 of *LNCS*, pages 425–439. Springer.

Coward, P. D. (1988). A review of software testing. *Inf. Softw. Technol.*, 30:189–198.

DeMillo, R. A., Lipton, R. J., and Sayward, F. G. (1978). Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41.

Deng, D., 0022, W. Z., and Lu, S. (2013). Efficient concurrency-bug detection across inputs. In Hosking, A. L., Eugster, P. T., and Lopes, C. V., editors, *OOPSLA*, pages 785–802. ACM.

Farchi, E., Nir, Y., and Ur, S. (2003). Concurrent bug patterns and how to test them. In *IPDPS' 03*, pages 286.2–, Washington, DC, USA. IEEE Computer Society.

Gligoric, M., Zhang, L., Pereira, C., and Pokam, G. (2013). Selective mutation testing for concurrent code. In *ISSTA*, pages 224–234. ACM.

Herman, P. M. (1976). A data flow analysis approach to program testing. *Australian Computer Journal*, 8(3):92–96.

Jagannath, V., Gligoric, M., Jin, D., Rosu, G., and Marinov, D. (2010). Imunit: improved multithreaded unit testing. In *IWMSE '10*, pages 48–49, New York, United States. ACM.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. *Software, IEEE*, 12(4):52–62.

Kitchenham, B. A., Dybaa, T., and Jorgensen, M. (2004). Evidence-Based Software Engineering. In *Proceedings of ICSE 2004*, pages 273–281. IEEE Computer Society Press.

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

Kundu, S., Ganai, M. K., and Wang, C. (2010). Contessa: Concurrency testing augmented with symbolic analysis. *Lecture Notes in Computer Science (LNCS)*, 6174:127–131.

Laski, J. W. and Korel, B. (1983). A data flow oriented program testing strategy. *IEEE Trans. Softw. Eng.*, 9:347–354.

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

Maldonado, J. C. (1991). *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software*. PhD thesis, DCA/FEEC/UNICAMP, Campinas, SP.

Mathur, A. P. and Wong, E. W. (1993). An empirical comparison of mutation and data flow based test adequacy criteria. Journal of Software Testing, Verification, and Reliability 4(1): 9–31.

Pilar, M., Simmonds, J., and Astudillo, H. (2014). Semi-automated tool recommender for software development processes. *Electronic Notes in Theoretical Computer Science*, 302(0):95 – 109. Proceedings of the Latin American Computing Conference (CLEI 2013).

Rapps, S. and Weyuker, E. J. (1985). Selecting Software Test Data Using Data Flow Information. *IEEE Trans. Softw. Eng.*, 11(4):367–375.

Sadowski, C. and Yi, J. (2009). Tiddle: A trace description language for generating concurrent benchmarks to test dynamic analyses. In *WODA*.

Shull, F., Carver, J., and Travassos, G. H. (2001). An empirical methodology for introducing software processes. In *ESEC*, pages 10–14.

Silva, R., Pezzi, G., Maillard, N., and Diverio, T. (2005). Automatic data-flow graph generation of mpi programs. In *SBAC-PAD*, pages 93–100.

Souza, P. S. L., do Rocio Senger de Souza, S., and Zaluska, E. (2014). Structural testing for message-passing concurrent programs: an extended test model. *Concurrency and Computation: Practice and Experience*, 26(1):21–50.

Souza, S. R. S., Brito, M. A. S., Silva, R. A., Souza, P. S. L., and Zaluska, E. (2011). Research in concurrent software testing: a systematic review. In *PADTAD*, pages 1–5.

Vegas, S., Juzgado, N. J., and Basili, V. R. (2006). Packaging experiences for improving testing technique selection. *Journal of Systems and Software*, 79(11):1606–1618.

Vos, T. E. J., Marín, B., Escalona, M. J., and Marchetto, A. (2012). A methodological framework for evaluating software testing techniques and tools. In *QSIC*, pages 230–239.

Wu, M., Zhou, B., and Shi, W. (2009). A self-adaptive test framework for concurrent programs. In *MEDES*, pages 456–457, Lyon, France.

Xiaoan, B., Na, Z., and Zuohua, D. (2009). Test case generation of concurrent programs based on event graph. In *5th International Joint Conference on INC, IMS, and IDC (NCM 2009)*, pages 143–149, Seoul, Korea.

Yang, C.-S. D. and Pollock, L. L. (2003). All-uses testing of shared memory parallel programs. *Softw. Test., Verif. Reliab.*, 13(1):3–24.

Yang, Y., Chen, X., and Gopalakrishnan, G. (2008). Inspect: a runtime model checker for multithreaded C programs. Technical report, University of Utah.

Yu, J., Narayanasamy, S., Pereira, C., and Pokam, G. (2012). Maple: a coverage-driven testing tool for multithreaded programs. In Leavens, G. T. and Dwyer, M. B., editors, *OOPSLA*, pages 485–502. ACM.