Optimus: Mecanismo de otimização de execução de testes em sistemas autoadaptativos

Isabely do Nascimento Costa[1]
Orientadora: Rossana M. C. Andrade[1] **— Co-orientador: Ismayle S. Santos**[2]

[1]Grupo de Redes de Redes de Computadores, Engenharia de Software e Sistemas
Universidade Federal do Ceará (UFC) - 60455-760 – Fortaleza – CE – Brasil
Mestrado em Ciência da Computação
Data de início: 10 de Maio de 2021 – Data de término: 15 de Abril de 2024
Data da aprovação da proposta de dissertação: 11 de Agosto de 2023

[2]Universidade Estadual do Ceará (UECE) – Fortaleza, CE - Brasil

isabelycosta@alu.ufc.br, rossana@ufc.br,ismayle.santos@uece.br

Abstract. *Dynamically adaptive systems (DAS) are capable of modifying themselves automatically according to their environment. These dynamic adaptations bring flexibility to the system but can also result in failures during execution. For DAS, various testing approaches have been proposed to solve the main challenges, one of which is runtime testing. However, there is a lack of studies related to the effectiveness and cost of executing tests at runtime and, as a result, this research proposes a mechanism to reduce the cost of execution and aid in the effectiveness of executing tests at runtime to contribute to the identification of faults in DAS. Link to video: https://youtu.be/hb72qtNdI7g*

Resumo. *Os sistemas dinamicamente adaptativos (DAS) são capazes de modificar-se automaticamente de acordo com o ambiente no qual estão inseridos. Essas adaptações dinâmicas trazem flexibilidade ao sistema, mas também podem resultar em falhas durante a sua execução. Para os DAS, várias abordagens de teste foram propostas como forma de resolver os principais desafios, sendo uma delas o teste em tempo de execução. No entanto, há uma carência de estudos relacionados a eficácia e custo de execução de testes em tempo de execução e, em razão disso, esta pesquisa propõe um mecanismo para diminuir o custo de execução e auxiliar na eficácia da execução de testes em tempo de execução com o objetivo de contribuir para a identificação de falhas em DAS.*

1. Introdução

Os sistemas de informação modernos estão cada vez mais complexos. Isso se deve ao crescente uso de dispositivos móveis e consequentemente com a necessidade de ininterruptamente os mesmos funcionarem em qualquer ambiente. Para suprir essa necessidade, a indústria de software precisou se adaptar as tais demandas utilizando-se de sistemas altamente distribuídos. Contudo, desenvolver, configurar e manter esses sistemas é uma tarefa muito difícil, sujeita a erros e custosa (KRUPITZER et al., 2015). Uma solução para esse problema é a autoadaptação, de modo que espera-se que o *software* autoadaptativo

cumpra os seus requisitos em tempo de execução, em resposta a alterações (SALEHIE; TAHVILDARI, 2009).

As adaptações dinâmicas dos sistemas enquanto eles estão sendo operados podem levar a alterações inseguras em tempo de execução e, em seguida, podem levam a novos riscos de bugs e interações inesperadas (LAHAMI; KRICHEN, 2021). No contexto dos DAS, as abordagens de teste tradicionais são ineficazes devido às características inerentes a esses sistemas. Detectar falhas neste tipo de sistema de forma eficaz não é uma tarefa trivial (SIQUEIRA; FERRARI; SERIKAWA et al., 2016). Dessa maneira, a atividade de teste para sistemas DAS se torna complexa (SIQUEIRA; FERRARI; SOUZA et al., 2021). A partir disso, o teste em tempo de execução tem potencial para ser uma solução apropriada para a validação de sistemas dinamicamente adaptativos (LAHAMI; KRICHEN; JMAIEL, 2013) e podem atuar de várias formas para resolução dos desafios mencionados (SANTOS, E. B. d., 2020). Contudo, há uma carência em abordagens que gerenciem e mantenham de forma eficiente os testes em tempo de execução podendo gerar preocupações como relacionadas a custo de execução e evolução de casos de teste (LAHAMI; KRICHEN, 2021). Em razão disso, se faz necessária a criação de um mecanismo de geração de sequências de casos de teste mais assertivas e com menor custo.

Com o objetivo de contribuir para a identificação de falhas em sistemas DAS, esta pesquisa propõe um mecanismo utilizando algoritmos de otimização para geração de sequências de casos de testes mais eficazes e com menor custo de execução (*e.g.* consumo de recursos e tempo). Para que seja possível alcançar o objetivo geral da pesquisa, os seguintes objetivos específicos foram enumerados: Modelar o problema como uma ou mais funções *fitness*[1]; Implementar um novo mecanismo para geração de sequências de casos de teste utilizando a função fitness e avaliar em uma abordagem de teste o mecanismo proposto comparando diferentes algoritmos em relação a custo e eficácia.

2. Fundamentação Teórica

2.1. Sistemas Dinamicamente Adaptativos (DAS)

Os "sistemas baseados em componentes distribuídos" podem mudar dinamicamente durante sua execução contínua sem fim. Geralmente, essas mudanças dinâmicas são necessárias para fornecer sistemas mais confiáveis, para apagar deficiências detectadas, ou para apoiar o desenvolvimento rápido dos requisitos dos usuários e a crescente variabilidade dos ambientes de execução (LAHAMI; KRICHEN, 2021). Esses sistemas são chamados de *Dynamically Adaptive Systems* (DAS). Os DAS fornecem as chamadas propriedades de autogestão, como a autoconfiguração, a autorrecuperação na presença de falhas, a auto-otimização e a autoproteção contra ameaças (KRUPITZER et al., 2015). Para alcançar um comportamento adaptativo, as propriedades básicas do sistema são: autoconsciência e consciência de contexto. A autoconsciência descreve a capacidade de um sistema de estar ciente de si mesmo. Consciência de contexto significa que o sistema é ciente de seu ambiente operacional, o chamado contexto (KRUPITZER et al., 2015). A dinâmica de adaptação, pode ocorrer em tempo de execução. Um Sistema Dinamicamente Adaptativo (DAS) é um sistema de software com adaptação em tempo de execução

[1]Uma função *fitness* é uma função objetiva que é utilizada para avaliar soluções (ARRIETA et al., 2019). Esta é necessária para que o algoritmo possa determinar a proximidade entre a solução dada e a solução ótima (BAJAJ; SANGWAN, 2019).

ativada (KRUPITZER et al., 2015; SANTOS, I. d. S., 2017).

2.2. Teste de software

O teste de software segundo a ISO (*International Organization for Standardization*) 29119, seria um conjunto de atividades realizadas para facilitar a descoberta e/ou avaliação de propriedades de um ou mais itens (HASS, 2014). A atividade de teste leva cerca de metade do custo total de desenvolvimento de software, sendo um processo demorado e caro (BAJAJ; SANGWAN, 2019). Dadas as restrições de tempo e custo, uma das principais questões do teste se torna: qual subconjunto de todos os casos de teste possíveis tem a maior probabilidade de detectar a maioria dos erros (MYERS; SANDLER; BADGETT, 2013). Segundo (COPELAND, 2004), existem cinco critérios básicos para definir até onde deve-se testar um software que são: Critérios de cobertura, Taxa de descoberta de defeitos, Custo marginal de encontrar o próximo defeito; Consenso da equipe e Definição do chefe.

A principal característica do sistema dinamicamente adaptativo é que ele pode adaptar-se em tempo de execução de acordo com a informação do contexto. Este atributo traz vários desafios para a atividade de teste de software (SANTOS, I. d. S., 2017), como: Muitas das adaptações são realizadas em tempo de execução (SIQUEIRA; FERRARI; SOUZA et al., 2021), não sendo possível analisá-las durante o tempo de desenvolvimento; a quantidade de cenários gerados a partir de alternativas de adaptação ainda pode ser muito grande e inviável do ponto de vista do teste (SIQUEIRA; FERRARI; SOUZA et al., 2021). O teste em tempo de execução tem potencial para ser uma solução para a validação de sistemas DAS, devido a dificuldade de identificar em tempo de desenvolvimento todo contexto operacional possível que um DAS pode encontrar em tempo de execução. Segundo (LAHAMI; KRICHEN; JMAIEL, 2015) o teste em tempo de execução (ou *Runtime Testing*) é definido como um método de teste que é realizado em ambiente de execução final de um sistema quando o sistema ou uma parte dele está operacional (FREDERICKS; RAMIREZ; CHENG, 2013).

2.3. Otimização em Engenharia de Software (SBSE)

A conciliação entre técnicas de otimização e Engenharia de Software ficou conhecida como Otimização em Engenharia de Software, em inglês *Search-Based Software Engineering* (MAIA; SOUZA, 2013). Dada a importância da fase de Teste de Software, a subárea denominada *Search-Based Software Testing* (SBST) se destaca em SBSE (MCMINN, 2011). A SBST é a utilização de técnicas de pesquisa meta-heurística otimizada para automatizar ou automatizar parcialmente uma tarefa de teste (MCMINN, 2011).

Segundo (HARMAN; JONES, 2001), apenas dois componentes são necessários para aplicar o SBSE: Uma representação (codificação) do problema e a definição da função de fitness. As soluções candidatas são evoluídas e são avaliadas em um processo iterativo até que uma condição de parada seja cumprida. Como resultado, soluções ótimas são encontradas para o problema (PÉREZ et al., 2021). A função *fitness* (ou função quantitativa) é necessária para que o algoritmo possa discriminar entre soluções promissoras e más. Esta função determina a proximidade entre a solução dada e a solução ótima. Para a abordagem de objetivo único, a função *fitness* é a função de objetivo do problema, que maximiza ou minimiza para obter soluções ótimas. Por outro lado, a abordagem multi-

objetiva tem várias funções objetivo para cada meta, que se maximizam ou minimizam individualmente para obter as soluções ideais (BAJAJ; SANGWAN, 2019).

3. Metodologia

A metodologia proposta foi baseada a partir dos objetivos de pesquisa citados na Seção 1. Inicialmente, foi realizada uma pesquisa na literatura em busca de revisões sistemáticas da literatura (do inglês, *Systematic Literature Review* - SLR) que contribuíssem para a justificativa deste trabalho. A partir dessa pesquisa, identificou-se a necessidade de alcançar trabalhos atuais e que estivessem voltados para o objetivo deste trabalho de mestrado. Dado que a SLR mais atual encontrada abrangia trabalhos publicados somente até 2020. Em razão disto, uma revisão sistemática da literatura foi realizada. Esta SLR foi conduzida seguindo a *guideline* de Kitchenham et al. (KITCHENHAM; BUDGEN; BRERETON, 2016) e teve como objetivo identificar os desafios, abordagens, fatores de influência e tendências em testes de sistemas dinamicamente adaptativos. Ademais, uma análise de mecanismos de otimização existentes na literatura foi feita utilizando-se de revisões sistemáticas focadas em SBST ou SBSE.

Com base nos dados obtidos pelas etapas anteriores para implementação do mecanismo, seguindo os passos definidos por (HARMAN; JONES, 2001) para aplicação de SBSE, serão definidas: uma representação do problema que seja passível de manipulação simbólica e uma função *fitness* do problema de acordo com os objetivos indicados na seção 4. Em seguida, serão selecionados algoritmos genéticos com as características necessárias para aplicação da função *fitness*. Por fim, um ou mais algoritmos genéticos serão escolhidos para implementação do mecanismo de geração de sequências de casos de teste. Para avaliar serão utilizadas duas métricas amplamente utilizadas (ELBAUM; MALISHEVSKY; ROTHERMEL, 2000): a *Average Percentage of Faults Detected* (APFD) que seria a percentagem média de falhas detectadas e a *Average Percentage of Blocks Covered* (APBC) que seria relacionada a percentagem de cobertura de blocos de código. Ao fim da implementação do mecanismo, o mesmo será conectado ao RETAkE (RuntimE Testing of dynamically Adaptive systEms) (SANTOS, E. B. d., 2020), apresentado na Seção 5, para a execução de experimentos controlados seguindo as etapas do processo de experimentação definido por (WOHLIN et al., 2012).

4. Proposta

A partir dos dados obtidos pelas etapas anteriores da pesquisa (Leitura de revisões sistemáticas e execução da SLR), onde pode-se observar a carência de abordagens de testes que utilizam de mecanismos de otimização e um desafio relacionado a custo de execução, o mecanismo (*Optimus*) de geração de sequências de testes proposto para esse trabalho utilizará de algoritmos genéticos, função fitness multi-objetiva, tendo como objetivos: diversidade de contexto; cobertura de código e custo de execução. Estes objetivos foram escolhidos de acordo com o objetivo principal da pesquisa. A diversidade de contexto será um critério para alcançar sequências de testes mais eficazes, a cobertura de código está relacionado uma métrica para verificar se a diversidade de contexto alcança um grau relativamente ótimo de cobertura de código do sistema sob teste e o custo seria para selecionar conjuntos de casos de teste que obtessem maior cobertura por menor custo. Em resumo, o objetivo de otimização da solução seria a maximização da diversidade de contexto, maximização da cobertura do código e minimização do custo de execução. Vale

ressaltar que os algoritmos não foram selecionados, em razão de que têm-se a intenção de se testar alguns algoritmos e até mesmo utilizar mais de um para elaboração do mecanismo. A partir das definições supracitadas, o problema será modelado como uma ou mais funções fitness e um novo mecanismo (Optimus) para geração de sequências de casos de teste utilizando uma ou mais funções fitness.

5. Trabalhos relacionados

Na literatura foram identificados trabalhos que utilizam otimização para auxiliar na execução de abordagens de teste em sistemas DAS. Entretanto, foi identificada uma lacuna em relação a um mecanismo de otimização de sequência de testes em sistemas DAS.

Maggio e Mandrioli (MANDRIOLI; MAGGIO, 2022) apresentam uma abordagem para aspectos não funcionais. O objetivo desta abordagem é fornecer garantias empíricas sobre o comportamento do sistema. Para isso, os autores definiram um problema de otimização para avaliar o desempenho da camada de adaptação do software. Essa abordagem permite a exploração de grandes espaços de entrada e configuração, e pode fornecer uma quantificação da confiança do sistema durante sua execução. Já o RETAkE de (SANTOS, E. B. d., 2020) é uma abordagem para executar o teste em tempo de execução com base na variabilidade do contexto do sistema e na modelação de características. O RETAkE testa o mecanismo de adaptação, permitindo a verificação das suas regras de adaptação com o modelo de variabilidade do sistema. O teste em tempo de execução é apoiado pela verificação das propriedades comportamentais. Esta abordagem gera sequências de testes, contudo de forma aleatória e utilizando o algoritmo de Hamming para calcular a distância de Hamming entre o contexto do estado atual do DAS no DFTS (*Dynamic Feature Transition System*) em relação aos seus vizinhos.

Em resumo, diferente dos trabalhos mencionados anteriormente, o objetivo deste trabalho é trazer um mecanismo de otimização de sequências de testes que possa ser utilizado de forma ampla em abordagens de testes em sistemas DAS. De forma que as abordagens que busquem os objetivos de diversidade de contexto, cobertura de código e custo de execução possam utilizar desse mecanismo.

6. Considerações finais

Este documento apresentou a proposta para o mestrado até o momento focado nos temas Sistemas Dinamicamente Adaptativos, Teste de Software e Otimização em Engenharia de Software. Em relação ao status da pesquisa, iniciou-se o estudo para definição da função *fitness* do mecanismo e tendo como contribuições esperadas: a publicação da revisão sistemática da literatura, publicação de um artigo com resultados comparativos a partir da aplicação dos algoritmos genéticos e por fim; a publicação dos resultados encontrados pela pesquisa ao final da avaliação.

Referências

ARRIETA, A. et al. Search-based test case prioritization for simulation-based testing of cyber-physical system product lines. **Journal of Systems and Software**, 2019.

BAJAJ, A.; SANGWAN, O. P. A systematic literature review of test case prioritization using genetic algorithms. **IEEE Access**, IEEE, v. 7, p. 126355–126375, 2019.

COPELAND, L. **A practitioner's guide to software test design**. Artech House, 2004.

ELBAUM, S.; MALISHEVSKY, A. G.; ROTHERMEL, G. Prioritizing test cases for regression testing. In: PROCEEDINGS of the 2000 ACM SIGSOFT international symposium on Software testing and analysis. 2000. P. 102–112.

FREDERICKS, E. M.; RAMIREZ, A. J.; CHENG, B. H. Towards run-time testing of dynamic adaptive systems, 2013.

HARMAN, M.; JONES, B. F. Search-based software engineering. **Information and software Technology**, Elsevier, v. 43, n. 14, p. 833–839, 2001.

HASS, A. M. **Guide to advanced software testing**. Artech House, 2014.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, P. **Evidence-based software engineering and systematic reviews**. CRC press, 2016.

KRUPITZER, C. et al. A survey on engineering approaches for self-adaptive systems. **Pervasive and Mobile Computing**, Elsevier, v. 17, p. 184–206, 2015.

LAHAMI, M.; KRICHEN, M. A survey on runtime testing of dynamically adaptable and distributed systems. **Software Quality Journal**, Springer, p. 1–39, 2021.

LAHAMI, M.; KRICHEN, M.; JMAIEL, M. Runtime testing framework for improving quality in dynamic service-based systems, 2013.

LAHAMI, M.; KRICHEN, M.; JMAIEL, M. Runtime testing approach of structural adaptations for dynamic and distributed systems. **International Journal of Computer Applications in Technology**, 2015.

MAIA, C. L. B.; SOUZA, J. T. de. UMA PROPOSTA DE OTIMIZAÇÃO PARA SELEÇÃO DE CASOS DE TESTES PARA AUTOMAÇÃO, 2013.

MANDRIOLI, C.; MAGGIO, M. Testing Self-Adaptive Software With Probabilistic Guarantees on Performance Metrics: Extended and Comparative Results. **IEEE Transactions on Software Engineering**, 2022.

MCMINN, P. Search-based software testing: Past, present and future. In: IEEE. 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops. 2011. P. 153–163.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The art of software testing**. John Wiley & Sons, 2013.

PÉREZ, F. et al. Empowering the Human as the Fitness Function in Search-Based Model-Driven Engineering. **IEEE Transactions on Software Engineering**, IEEE, 2021.

SALEHIE, M.; TAHVILDARI, L. Self-Adaptive Software: Landscape and Research Challenges, 2009.

SANTOS, E. B. d. RETAkE: Abordagem para teste em tempo de execução de sistemas dinamicamente adaptativos, 2020.

SANTOS, I. d. S. TestDAS: Testing method for dynamically adaptive systems, 2017.

SIQUEIRA, B. R.; FERRARI, F. C.; SOUZA, K. E. et al. Testing of adaptive and context-aware systems: approaches and challenges. **Software Testing, Verification and Reliability**, Wiley Online Library, v. 31, n. 7, e1772, 2021.

SIQUEIRA, B. R.; FERRARI, F. C.; SERIKAWA, M. A. et al. Characterisation of challenges for testing of adaptive systems. In: PROCEEDINGS of the 1st Brazilian Symposium on Systematic and Automated Software Testing. 2016. P. 1–10.

WOHLIN, C. et al. **Experimentation in software engineering**. Springer Science & Business Media, 2012.