Execução paralela de programas como suporte ao teste de mutação

Stevão Alves de Andrade[1], **Orientador: Márcio Eduardo Delamaro**[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

{stevao,delamaro}@icmc.usp.br

Abstract. *This paper presents a Master's Degree project under development that aims to propose a solution for decreasing the computational cost involved in applying mutation testing. There are being investigated alternatives for parallelization of test run by applying techniques of concurrent programming. Thus, studies are being conducted aiming to identify phases of testing that require a greater cost during its execution, in order to improve efficiency when applying a solution with use of parallelization techniques.*

Resumo. *Este artigo apresenta um projeto de mestrado em desenvolvimento que tem como objetivo propor uma solução para a redução do custo computacional envolvido na aplicação do teste de mutação. Estão sendo investigadas alternativas para paralelização da execução do teste aplicando técnicas de programação concorrente. Neste sentido, estudos vêm sendo conduzidos com objetivo de identificar as fases do teste que exigem maior custo durante sua execução, com intuito de melhorar sua eficiência ao aplicar uma solução com uso de técnicas de paralelização.*

1. Introdução

Dentre as técnicas de verificação e validação de software, a atividade de teste é uma das mais utilizadas, constituindo-se em uma técnica para fornecer evidências da confiabilidade de um produto, consistindo em uma análise dinâmica do produto [Delamaro et al. 2007].

Em geral, os critérios de teste de software são estabelecidos a partir de três técnicas: a funcional, a estrutural e a baseada em erros. Na técnica baseada em erros, que é o foco deste trabalho, os critérios e requisitos de teste são derivados a partir dos erros típicos cometidos no processo de desenvolvimento de software [Myers et al. 2004].

Um dos critérios baseados em erros é o Teste de Mutação. Esse critério surgiu com o objetivo de avaliar a efetividade/qualidade de um conjunto de casos de teste. Os erros típicos de um domínio ou paradigma de desenvolvimento são caracterizados e desenvolvidos como operadores de mutação que, durante a atividade de teste, geram versões modificadas (mutantes) do programa em teste. A intenção, com isso, é auxiliar na seleção de casos de teste que demonstrem que os erros modelados pelos operadores de mutação não estão presentes no programa em teste [Delamaro and Maldonado 1999].

Além de ser uma técnica eficaz quando aplicada para testar um programa, essa técnica tem sido amplamente utilizada para avaliar a eficácia de outras técnicas de validação

em revelar defeitos. Experimentos são conduzidos para avaliar o quanto que uma técnica ou critério de teste é capaz de revelar defeitos modelados pelos operadores de mutação. A vantagem do uso de teste de mutação nesse contexto é o fato dessa abordagem apresentar um processo bem definido e preciso para semear os defeitos no produto a ser avaliado. Isso facilita a replicação dos resultados em outros experimentos [Andrews et al. 2005].

2. Caracterização do Problema

Um dos maiores problemas para a aplicação do teste de mutação está relacionado ao seu alto custo computacional, uma vez que o número de mutantes gerados, mesmo para pequenos programas, pode ser muito grande. Esse custo está relacionado ao tempo de análise dos mutantes, normalmente realizado manualmente [Delamaro et al. 2007] e com o tempo necessário para executar os mutantes gerados, representado pela seguinte equação [Mateo and Usaola 2013]:

$$Mt = Gt * Nm + Et * Nt * (Nm + 1)$$

Onde *Mt* é o tempo total da aplicação da técnica, *Gt* é o tempo para a geração de um único mutante, *Et* é o tempo de execução para um caso de teste, *Ntc* é o número total de casos de teste e *Nm* o número de mutantes gerados.

Abordagens como a geração automática de casos de testes possibilitam a verificação do sistema de maneira intensiva. A geração de dados de teste é um processo de identificação de dados do domínio de entrada válidos para um programa de acordo com os critérios de teste e é utilizada com objetivo de verificar se uma dada implementação contém falhas [Korel 1990]. Quando aplicada em conjunto ao teste de mutação, os mutantes gerados precisam ser executados diversas vezes com um número elevado de casos de teste, o que faz com que a eficiência na execução do critério se torne ainda mais importante. Além disso, avaliações experimentais necessitam que mutantes gerados sejam executados com todo o domínio de casos de teste disponíveis, para que os dados referentes ao processo possam ser colhidos para análise.

Várias estratégias foram desenvolvidas com o objetivo de reduzir esses custos, dentre as quais se destacam a utilização de arquiteturas de hardware avançadas para reduzir o tempo de execução dos mutantes [Choi and Mathur 1993], a determinação de um conjunto reduzido e efetivo de operadores de mutação[Barbosa 1998, Mathur and Wong 1993, Offutt et al. 1996] e abordagens para reduzir o número de mutantes gerados com uso de análise estatística de anomalias de fluxo de dados [Marshall et al. 1990].

Uma importante técnica para redução de custo é a execução paralela, a qual é amplamente explorada em outros contextos, como previsão de tempo, dinâmica de fluídos, processamento de imagens, química quântica entre outros [Quinn 2004]. Em [Jia and Harman 2011] é apresentado um *survey* sobre o teste de mutação e são descritas técnicas para a paralelização da execução do teste de mutação. Em geral as abordagens concentram-se em explorar diferentes arquiteturas computacionais disponíveis na época. Com os avanços computacionais atuais, novas abordagens podem ser exploradas visando aperfeiçoar essa paralelização, incluindo o uso de *clusters*.

3. Caracterização da Contribuição

O objetivo deste trabalho é propor uma alternativa para melhorar a eficiência da aplicação do teste de mutação reduindo o tempo de resposta envolvido na aplicação da técnica. Pretende-se desenvolver um modelo para a aplicação do teste de mutação utilizando técnicas de programação concorrente com objetivo de aperfeiçoar os recursos computacionais utilizados durante essa atividade, considerando questões fundamentais como disponibilidade e escalabilidade.

Com base nos objetivos do projeto e nas atividades a serem realizadas durante a sua condução, esperam-se que os seguintes resultados sejam atingidos:

- Contribuir com a atividade de teste de mutação, por meio da definição de um modelo para a aplicação do teste de mutação adaptado aos moldes de programas concorrentes;
- Instanciar para a ferramenta *Proteum* [Delamaro 1993] o modelo proposto, objetivando aumentara eficiência da aplicabilidade do critério;
- Viabilizar a aplicação do teste de mutação em conjunto com outras abordagens como, por exemplo, geração automática de dados de teste, de forma a tornar uma alternativa prática para verificação intensiva do produto em teste;
- Gerar resultados e discussões com base na experimentação dos resultados obtidos e comparar com outras abordagens existentes.

4. Estado Atual do Trabalho

Para fornecer os subsídios teóricos necessários para a conclusão dos objetivos deste projeto, inicialmente foi realizada uma investigação de conceitos básicos relacionados à área de engenharia de software e programação concorrente.

Entre as atividades que estão sendo desenvolvidas, pode-se destacar:

- Decomposição da atividade do teste de mutação em pequenas porções com objetivo de identificar quais atividades podem ser executadas de maneira concorrente ou em paralelo;
- Criação do grafo de dependência entre de tarefas identificadas. Com base na decomposição das atividades, o objetivo é identificar as porções de trabalho que são dependentes entre si;
- Mapear as tarefas que podem ser executadas em diferentes processadores;
- Implementação dos algoritmos de balanceamento de carga apresentados em [Mateo and Usaola 2013];
- Adequação da ferramenta de teste de mutação *Proteum* [Delamaro 1993] para execução em paralelo, fazendo uso dos algoritmos de balanceamento de carga.

5. Trabalhos Relacionados

Existem diversos trabalhos na literatura que tratam sobre a execução paralela de mutantes. Em um *survey* sobre mutação [Jia and Harman 2011] identificaram três linhas de investigação: *(i)* execução de mutantes em máquinas de única instrução e múltiplos dados (SIMD), *(ii)* execução de mutantes em máquinas de múltiplas instruções e múltiplos dados (MIMD) e *(iii)* uso de algoritmos seriais otimizados. Além dessas abordagens, também

é possível destacar o trabalho apresentado em [Mateo and Usaola 2013], que versa sobre algoritmos de balanceamento de carga.

Esses trabalhos se relacionam com este trabalho de Mestrado por apresentarem abordagens semelhantes a qual está sendo desenvolvida neste. No entanto, diferenciam-se por serem abordagens aplicadas em contextos diferentes. Esses trabalhos, em sua maioria, servem como base para adaptação de padrões e criação de métricas que vêm sido desenvolvidos neste projeto de Mestrado.

6. Avaliação dos Resultados

Pretende-se avaliar os resultados obtidos aplicando métricas de avaliação de desempenho. A avaliação de desempenho tem como objetivo avaliar e analisar o desempenho do software, com intuito de entender melhor o seu comportamento. Além disso, fornece informações importantes para tomadas de decisões de projeto e implementação, necessárias para a melhoria de um produto. Nesse sentido, entende-se que é a abordagem mais apropriada a ser utilizada para validar os resutados obtidos por meio de experimentos.

Dentre as avaliações que pretende-se realizar, destacam-se:

- Instanciar a abordagem de execução paralela do teste de mutação, proposta na ferramenta de teste de mutação *Proteum*;
- Avaliar, por meio de experimentos científicos, o ganho em eficiência *(speedup)* da versão paralela da ferramenta *Proteum* em relação à versão sequencial existente;
- Realizar estudo comparativo à abordagem apresentada em [Mateo and Usaola 2013], com intuito de verificar o comportamento dos algoritmos de balanceamento de carga apresentados, quando aplicados à ferramenta de teste de mutação *Proteum*.

Referências

Andrews, J., Briand, L., and Labiche, Y. (2005). Is mutation an appropriate tool for testing experiments? In *International Conference on Software Engineering (ICSE/´05)*, pages 15–21, St. Louis, Missouri, USA.

Barbosa, E. F. (1998). Uma contribuição para a determinação de um conjunto essencial de operadores de mutação no teste de programas C. Master's thesis, ICMC-USP, São Carlos, SP.

Choi, B. J. and Mathur, A. P. (1993). High-performance mutation testing. *The Journal of Systems and Software*, 1(20):135–152.

Delamaro, M. E. (1993). Proteum - um ambiente de teste baseado na análise de mutantes. Master's thesis, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC/USP), São Carlos, SP.

Delamaro, M. E. and Maldonado, J. C. (1999). Interface mutation: Assessing testing quality at interprocedural level. In *International Conference of the Chile an Computer Science Society (19th SCCC)*, pages 78–86.

Delamaro, M. E., Maldonado, J. C., and Jino, M. (2007). Introdução ao teste de software. In Delamaro, M. E., Jino, M., and Maldonado, J. C., editors, *Introdução ao Teste de Software*, pages 1 – 7. Elsevier.

Jia, Y. and Harman, M. (2011). An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678.

Korel, B. (1990). Automated software test data generation. *IEEE Trans. Softw. Eng.*, 16(8):870–879.

Marshall, A., Hedley, D., Riddell, I., and Hennell, M. (1990). Static dataflow-aided weak mutation analysis (sdawm). *Information and Software Technology*, 32(1):99 – 104. Special Issue on Software Quality Assurance.

Mateo, P. R. and Usaola, M. P. (2013). Parallel mutation testing. *Software Testing, Verification and Reliability*, 23(4):315–350.

Mathur, A. P. and Wong, W. E. (1993). Evaluation of the cost alternate mutation strategies. In *VII Simpósio Brasileiro de Engenharia de Software (VII SBES)*, pages 320–334, Rio de Janeiro, RJ.

Myers, G., Sandler, C., Badgett, T., and Thomas, T. (2004). *The Art of Software Testing*. Business Data Processing: A Wiley Series. Wiley.

Offutt, A. J., Rothermel, A. J., Untch, R. H., and Zapf, C. (1996). An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering Methodology*, 5(2):99–118.

Quinn, M. (2004). *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Higher Education. McGraw-Hill Higher Education.