Higor Amario de Souza, 5619750.

MAC 5779 – Engenharia de Software Experimental

Protocolo da pesquisa: Experimento controlado.

Título: Depuração de programas com múltiplos defeitos.

Problema e contextualização

As atividades de teste e depuração de programas são responsáveis por uma parcela significativa do processo de desenvolvimento de programas. O esforço para detectar (teste), localizar e corrigir (depuração) defeitos consome entre 50% e 80% do desenvolvimento e manutenção (Collofello and Woodfield, 1989). A depuração de programas é geralmente realizada de forma manual pelos desenvolvedores (Jones et al., 2007).

Diferentes estudos propõem técnicas para automatizar a tarefa de depuração de programas, a maioria delas baseada em informações de cobertura de código (Naish et al., 2010; Wong et al., 2010; Mariani et al., 2011). A partir da cobertura obtida dos testes executados, tais técnicas indicam comandos mais suspeitos de conter defeitos. No entanto, os experimentos para avaliar a eficácia de localização das técnicas em geral são realizados em programas contendo um único defeito por versão. A eficácia de localização é medida pela quantidade de código que precisa ser verificado até que o defeito seja localizado. Nos casos em que a avaliação é feita em programas contendo múltiplos defeitos, o desempenho de localização é inferior ao desempenho em programas com um defeito por versão (Naish et al., 2010; Wong et al., 2010).

Na prática, ao procurar defeitos em programas reais, a quantidade de defeitos em um programa é desconhecida (Jones et al., 2007). Portanto, é necessário que as técnicas de localização de defeitos consigam indicar comandos suspeitos em programas com múltiplos defeitos de forma eficaz para que sejam adotadas em ambientes reais de desenvolvimento.

Experimento 1 – Comparação entre técnicas usando programas reais

O objetivo deste experimento é avaliar a eficácia e a eficiência de localização da técnica proposta, chamada Multiple Faults (MF), em comparação com ao menos uma técnica de localização existente. A eficácia será medida por meio da localização ou não dos defeitos dentro do tempo previsto. A eficiência será medida pela quantidade de cliques realizados nas listas geradas pelas técnicas até a localização do defeito.

Entre as possíveis técnicas para comparação, estão as propostas por Jones et al. (2007) e por Wong et al. (2010). A escolha depende da possibilidade de obter os programas que implementam essas técnicas. No caso de não ser possível obter esses programas, iremos desenvolver programas que representem as técnicas e validá-los comparando os resultados obtidos nos *benchmarks* usados nos artigos.

A escolha pela métrica de quantidade de cliques tem como objetivo medir a quantidade absoluta de esforço para localizar defeitos. Os experimentos serão conduzidos em programas reais de grande porte (programas contendo acima de 10.000 linhas de código para uso real, e não somente para experimentos científicos). Entre os critérios para seleção dos programas estão: programas de código-aberto, ou programas proprietários autorizados, que contenham testes automatizados; programas de diferentes domínios; programas escritos em Java (a princípio); e programas que estejam em um repositório de código.

Após selecionar os programas, será necessário identificar os defeitos existentes entre suas diferentes versões usando as informações do repositório. Esses defeitos serão mapeados e registrados, e assim será possível criar diferentes versões dos programas incluindo e excluindo os defeitos mapeados. Os resultados dos testes serão analisados para as versões contendo quantidade diferentes dos defeitos para estudar a interferência entre os defeitos.

As técnicas de localização serão avaliadas para as diferentes versões geradas dos programas, e será possível medir o desempenho de localização em diferentes condições, como programas contendo uma determinada quantidade de defeitos ou programas contendo defeitos que interferem em outros defeitos. A coleta dos resultados será automatizada.

Portanto, as variáveis independentes são a técnica de localização, a quantidade de defeitos no programa, o uso de defeitos que interferem em outros defeitos, o domínio do programa, o tamanho do conjunto de teste, as características dos defeitos e o tamanho do programas. A variável dependente é a quantidade de clique para localizar o defeito. As técnicas de localização serão avaliadas fixando os valores das variáveis independentes e variando uma a uma. Isso permitirá a análise de cada um dos fatores no experimento.

Entre as ameaças à validade interna esperadas estão os fatores citados acima. A escolha das quantidade de cada um desses fatores pode influenciar os resultados obtidos pelas técnicas. Entre as ameaças externas estão a generalização dos resultados, já que os programas utilizados podem não ser representativos de um modo geral.

A análise estatística será feita com uso do teste de hipótese de Wilcoxon rank-sum não-pareado para a quantidade de cliques supondo que a distribuição da amostra não será normal. Será avaliada a quantidade de cliques até localizar o defeito, uma medida de razão. O teste de Anderson-Darling será aplicado para avaliar a normalidade dos dados.

Experimento 2 – Uso prático da técnica proposta

O objetivo deste experimento controlado será avaliar a eficácia e a eficiência de localização de defeitos da técnica MF (MF), em comparação com o uso tradicional de depuração (T). Pretende-se avaliar se MF pode ser usada por desenvolvedores para localizar defeitos.

Utilizaremos o IDE Eclipse para o experimento. Será desenvolvido um plugin da técnica MF para possibilitar a avaliação de seu uso. O uso tradicional também será avaliado no Eclipse, em que o desenvolvedor usará o depurador simbólico do IDE para procurar pelo defeito.

A seleção dos participantes será feita por conveniência. Os participantes devem ser desenvolvedores universitários e/ou da indústria com conhecimento prévio do Eclipse e experiência em depuração de programas. É esperada a participação de 30 pessoas. Como comparação, Parnin e Orso (2011) contaram com 34 participantes em seu experimento sobre o uso de técnicas de depuração automatizada por programadores.

Ao aceitar participar do experimento, os participantes responderão a uma prova de conhecimentos sobre a linguagem Java e também a um questionário sobre informações pessoais como idade e tempo de experiência em depuração e programação. A prova será baseada em conteúdo de certificações na linguagem, contendo questões de diferentes níveis de conhecimento.

O resultado da prova e o questionário serão usados para avaliar o nível de conhecimento e o tempo de experiência em depuração e programação dos participantes, que serão classificados como iniciantes ou experientes. Os participantes serão separados em dois grupos contendo

desenvolvedores iniciantes e experientes usando a seleção por casamento.

Para o experimento, serão utilizados dois programas contendo dois defeitos cada. Os programas são de domínios diferentes (Ant e Commons-Math). O tempo para a localização de cada defeito será de 30 minutos. Os participantes receberão um treinamento sobre o uso do plugin MF, com duração de 30 minutos, incluindo um exercício prático.

Os grupos irão alternar entre grupo de controle e grupo experimental entre os dois programas. O grupo de controle usará o Eclipse para buscar pelos defeitos sem o plugin. Serão fornecidas informações sobre o resultado da execução dos testes JUnit para auxiliar os integrantes do grupo de controle. O grupo experimental utilizará o plugin MF.

A eficácia será medida por meio da localização ou não dos defeitos dentro do tempo previsto. A eficiência será medida pela quantidade de cliques realizados nas listas geradas pelas técnicas até a localização do defeito e o tempo gasto até a localização. Para medir a quantidade de cliques do grupo de controle será necessário criar um plugin para o Eclipse que faça essa medição. O plugin da técnica MF também contará o número de cliques. Ao localizar o defeito, o desenvolvedor clica em um botão que será colocado no Eclipse para encerrar a busca por aquele defeito.

O tempo total máximo para a realização do experimentos é de 3h30m por participante, incluindo a prova (1h), realizada em uma data anterior ao experimento, o treinamento (30m) e o experimento para localização dos quatro defeitos (2h).

Entre as ameaças à validade interna esperadas estão a seleção, que tentou-se minimizar usando a seleção por casamento e a prova de conhecimentos. Para evitar as ameaças de competição e desmoralização, os grupos alternarão entre grupo de controle e experimental de acordo com o programa utilizado. A ausência de um pré-teste minimiza as ameaças de contaminação, seleção-testagem, seleção-maturação e seleção-abandono.

Entre as ameaças externas estão a generalização da amostragem, já que a amostra não é representativa de população como um todo. No entanto, o experimento será importante para avaliar a capacidade prática de uso da técnica de depuração automatizada.

O tempo exigido para localizar os defeitos pode ser muito longo ou curto, e apenas a execução de um piloto pode indicar se o tempo está adequado.

A análise será feita com uso do teste de hipótese de Wilcoxon rank-sum não-pareado para a quantidade de cliques supondo que a distribuição da amostra não será normal. Os dados coletados serão tempo de localização e quantidade de cliques, duas medidas de razão. O teste de Anderson-Darling será aplicado para avaliar a normalidade dos dados.

Referências

Collofello JS, Woodfield S. Evaluating the effectiveness of reliability-assurance techniques. Journal of Systems and Software 9 (3), 191–195. 1989.

Jones, J. A.; Bowring, J. F.; Harrold, M. J. Debugging in parallel. In: Proceedings of the 2007 International Symposium on Software Testing and Analysis. New York, NY, USA: ACM, 2007. (ISSTA ’07), p. 16–26.

Kitchenham, B., Pfleeger S. Personal Opinion Surveys. Guide to Advanced Empirical Software Engineering Cap. 3. 2008.

Mariani, L.; Pastore, F.; Pezze, M. Dynamic analysis for diagnosing integration faults. Software Engineering, IEEE Transactions on, v. 37, n. 4, p. 486 –508. 2011.

Naish, L.; Lee, H. J.; Ramamohanarao, K. Statements versus predicates in spectral bug localization. In: Software Engineering Conference (APSEC), 2010 17th Asia Pacific. [s.n.], 2010. p. 375 –384.

Parnin, C. e Orso, A. Are automated debugging techniques actually helping programmers? In: Proceedings of the 2011 International Symposium on Software Testing and Analysis. New York, NY, USA: ACM, 2011. (ISSTA ’11), p. 199–209.

Wainer, J. Experimento em sistemas colaborativos. Sistemas Colaborativos, Cap. 24, p.405-432, 2011.

Wettel, R.; Lanza, M.; Robbes, R., Software systems as cities: a controlled experiment,. In: Proceedings of the 2011 International Conference on Software Engineering. Honolulu, HI, 2011. (ICSE 2011), p.551-560.

Wong, W. E.; Debroy, V.; Choi, B. A family of code coverage-based heuristics for effective fault localization. Journal of Systems and Software, v. 83, n. 2, p. 188–208, 2010.