Ensinando Programação de Sistemas Concorrentes e Formas de Testar Tais Sistemas

Fagner Silva Martins (Bolsista), Emerson de Assis Silva (Voluntário), Josué da Silva Gomes Júnior (Voluntário), Ayla Débora Dantas de Souza Rebouças (Professora Orientadora) – CCAE/DCE - PROLICEN

1. Introdução

A indústria de desenvolvimento de software tem crescido cada vez mais, principalmente com o avanço das tecnologias e paradigmas computacionais. Porém este crescimento rápido traz alguns questionamentos quanto à qualidade dos sistemas produzidos, e demanda profissionais cada vez mais capacitados em teste de software.

Além disso, segundo Goetz (2006), desenvolver programas concorrentes é uma tarefa difícil, pois há mais fatores que podem dar errado em programas deste tipo. Sendo assim, melhorar a forma como se prepara profissionais para tal tipo de desenvolvimento é crucial. Entre as várias fases do processo de formação deste profissional, este trabalho se focou em desenvolvimento e testes. Testar é fundamental na Engenharia de Software (Bertolino 2007) e ajuda a garantir a qualidade desejada do produto final de software, fazendo parte das atividades de verificação e validação (V&V) (Sommerville 2006).

Dentre as abordagens para verificação e análise de sistemas de V&V, o teste é uma técnica em que se busca por possíveis defeitos, visando mostrar a presença de erros, e não sua ausência (Dijkstra 1969).

Tanto desenvolver quanto testar sistemas concorrentes é algo complexo. Vários alunos apontam que têm dificuldades em absorver o conteúdo "programação concorrente". Esse assunto pode ser explorado em diferentes disciplinas de um curso superior ou técnico em informática, como Sistemas Operacionais, Sistemas Distribuídos, etc. Mesmo assim, é um assunto abstrato e gera muita dificuldade. Além disso, alguns alunos apresentam dificuldades no teste automático de software.

Sendo assim, este trabalho descreve os passos seguidos pelos membros deste projeto do Prolicen que teve como objetivo geral o desenvolvimento de técnicas e objetos de aprendizagem para apoiar o ensino de Programação Concorrente e o teste de sistemas.

2. Metodologia

As atividades realizadas no decorrer do projeto foram: levantamento bibliográfico sobre ensino de programação concorrente e teste de software, elaboração de alguns objetos de aprendizagem (OA), aplicação e avaliação de um dos OAs na disciplina de POO por parte dos alunos.

3. Atividades Realizadas

As principais atividades realizadas foram: i) levantamento bibliográfico; ii) desenvolvimento de objetos de aprendizagem para apoiar o ensino de programação concorrente e teste de software; iii) avaliação de um dos OA; iv) escrita do relatório e de um artigo científico com resultados da pesquisa. Para o levantamento bibliográfico, foram utilizadas como fontes o portal ACM Digital Library e o IEEExplore. Além de buscas no Google Acadêmico à procura de trabalhos semelhantes. Para o desenvolvimento dos OAs, foram desenvolvidos protótipos simples que foram sendo evoluídos a partir da interação entre os alunos e a orientadora.

4. Resultados Obtidos

Nesta seção são apresentados os principais resultados alcançados a partir das atividades desempenhadas durante este projeto Prolicen.

4.1 Levantamento Bibliográfico sobre Ensino de Programação Concorrente

Alguns dos principais trabalhos relacionados ao ensino de programação concorrente foram os de Manickam e Aravind (2011), o de Toscani (2004) e o de Ortiz (2011).

O artigo de Manickam e Aravind (2011) trata da importância da programação concorrente e o uso de uma ferramenta de simulação (VITTI). Esta ferramenta foi criada para ensinar alguns algoritmos de sincronização simples ilustrando seu comportamento. Os autores enfatizam que há poucas ferramentas para o ensino de programação concorrente e falam sobre algumas como o JBACI e Convit.

O trabalho de Toscani (2004) apresenta o ambiente de programação Vale4 (V4), voltado exclusivamente para o uso acadêmico. Discute-se também a inadequação em ensinar programação concorrente com linguagens como Ada, Java e C. Todo o sistema da linguagem V4 foi implementado em SWI-Prolog. Boa parte do trabalho descrito por Toscani se destina à explicação das funcionalidades da linguagem V4, que não possui uma interface gráfica para auxiliar no aprendizado. .

O trabalho de Ortiz (2011) propõe que se introduza programação concorrente com a linguagem Erlang por ser mais adequada para o ensino deste tema do que linguagens como C / C + + / C# / Java. Após os testes, observou-se que no começo, os alunos sentiram um pouco de resistência, o que mudou com o tempo de uso.

Mesmo considerando estes trabalhos, observou-se que seria interessante criar um objeto de aprendizagem para apoiar o ensino de conteúdos introdutórios de programação concorrente em Java por considerá-la uma linguagem fortemente demandada pelo mercado e que tem um desempenho superior ao de outras linguagens como Erlang.

4.2 Levantamento Bibliográfico sobre Ensino de Teste de Software

Durante a revisão bibliográfica foram selecionados três principais trabalhos sobre ensino de testes unitários. Estes trabalhos são os de Jones e Chatmon (2001), Barriocanal et al. (2002) e Della Corte (2006).

O artigo de Jones e Chatmon (2001) trata da importância de testes de software na engenharia e na indústria de software e expõe a necessidade de um melhoramento no ensino de testes de software. Apresenta o framework SPRAE, que detalha alguns aspectos e passos a serem considerados para realizar testes com eficiência. Por fim é apresentada uma tabela com cada experiência e conhecimento a serem adquiridos.

Barriocanal et al. (2002) mostra os resultados obtidos no ensino de testes utilizando AUT (*Automated Unit Testing*) e a metodologia ágil XP em turmas introdutórias de programação do primeiro e segundo semestres de Ciências da computação. Segundo o autor, apenas cerca de 10% dos alunos que se submeteram à experiência conseguiram desenvolver testes satisfatoriamente. Della Corte (2006) revela que quanto mais cedo se inicia o ensino de testes, melhor é o aprendizado.

Segundo Patterson at al. (2003) "um bom testador deve ter habilidades e conhecimentos que são adquiridos na prática". A autora introduz o Módulo educacional integrado de OO e Testes de software, com o objetivo de ensinar gradualmente os conceitos de testes, em paralelo ao ensino de POO, relacionando apenas os conteúdos que se pretende ensinar. Foi desenvolvido um ambiente web chamado ProgTest, e integrado à ferramenta JABUTI para análise de cobertura de teste. O ProgTest avalia automaticamente os trabalhos enviados, promovendo um feedback tanto para o aluno, quanto para o professor, além de imparcialidade na avaliação.

4.3 Objeto de Aprendizagem para o Ensino Introdutório de Testes Automáticos com a Ferramenta JUnit

Com o objetivo de estimular o estudo de testes de software, foi desenvolvido um protótipo de um OA para auxiliar no desenvolvimento de testes na linguagem Java utilizando JUnit, framework desenvolvido em JAVA (Massol 2004) e introduzido em algumas disciplinas introdutórias de programação. Este OA apresentava alguns dos principais métodos do JUnit.. O objeto de aprendizagem em questão simula um diálogo com o usuário à procura de uma solução automática para o teste de um sistema de gerenciamento de turmas e disciplinas (um domínio comum para muitos alunos). São propostos alguns exercícios e soluções para os exercícios propostos. O objeto de aprendizagem propõe o problema, e detalha uma possível solução. Observa-se que o objeto de aprendizagem estimula

conhecimento tanto em programação, quanto em testes de software, estimulando a reflexão sobre o projeto (design) de seu software. Também é explorado o fator motivacional, pois a interação amigável e a forma como mostra a importância dos testes é relevante para a formação dos alunos, que às vezes veem esta tarefa como sendo exaustiva e desnecessária.

4.4 Objeto de Aprendizagem para o Ensino Introdutório de Programação Concorrente: "Quero Bolo"

Quero Bolo é uma ferramenta construída em Java com o intuito de apoiar na compreensão de conceitos iniciais da programação concorrente. Ela basicamente mostra uma animação, contendo três personagens (Cozinheiro, Mãe e filho), cujo comportamento pode ser influenciado pelo usuário, onde dois deles se movimentam: o Cozinheiro e a Mãe, representando duas threads cujo início e velocidade são controlados pelo usuário do OA. A cada movimento, são mostrados nos cantos superiores da tela, os estados das threads. É possível visualizar o código parcial destas threads, controlar sua velocidade, iniciá-las ou pausá-las, obter ajuda sobre a sua utilização ou conteúdo de programação concorrente.

Na animação, mostra-se uma mãe que vai à confeitaria buscar um bolo para seu filho e que aguarda até que este seja trazido, caso não esteja lá ainda. O cozinheiro vai buscar um bolo no fogão e o leva ao balcão. Ao clicar no botão iniciar, de cada boneco, são invocados os métodos start()[1] de ambas as threads, iniciando seu movimento. Quando o bolo é colocado em cima da mesa pelo cozinheiro, sua thread chama o método notify() acordando a thread do boneco da mãe, caso esta esteja esperando pelo bolo. Esta, ao pegá-lo, o leva para seu filho, concluindo então sua execução. Caso o boneco mãe chegue até a mesa e o bolo não esteja lá, é invocado o método wait(), pausando a execução da thread do boneco mãe até que o bolo tenha chegado.

Após o início da animação, o usuário poderá controlar a velocidade do boneco através de uma barra deslizante, redefinindo o valor utilizado pelo método sleep(timeout) das threads, entre zero e cem. O usuário também pode pausar qualquer thread, ativando o botão pausar, ou reiniciar o OA ativando o botão reiniciar. Ambas as funcionalidades só poderão ser executadas se uma das threads tiver sido ativada.

4.5. Avaliação do "Quero Bolo"

Com o término da primeira versão, foi realizada uma avaliação inicial por estudantes de uma turma de POO para a qual havia sido introduzido o conteúdo de programação concorrente. Foi feita uma apresentação em sala de aula e posteriormente foi feita a avaliação

por meio de um questionário online contendo quatorze perguntas discursivas e objetivas. Quatorze alunos participaram da avaliação. Uma das formas de identificar o nível de aceitação do OA por parte dos alunos foi perguntar se o indicariam para colegas. Dos alunos que responderam, 87% indicaria o OA para algum colega. Foi avaliado também o quanto a ferramenta poderia ajudar a absorver o conteúdo, sendo que quase 60% dos alunos responderam que a ferramenta poderia auxiliar em uma melhor absorção do assunto visto em sala de aula, confirmando a hipótese de utilidade do OA para apoiar a introdução desse conteúdo.

Aspectos positivos e negativos foram apontados pelos participantes da avaliação. A interface foi considerada amigável para desmistificar um assunto que causa certo receio aos alunos. Além disso, alguns consideraram o OA uma forma divertida de se entender o básico de threads em Java em pouco tempo. Entre os aspectos negativos estão: mais explicações sobre threads, a falta de recursos de áudio na ferramenta e o fato de não explorar outros aspectos da programação concorrente como o uso de semáforos e serão trabalhos futuros a se explorar.

5. Conclusão

De maneira geral, pôde se refletir bem sobre formas de melhor introduzir o ensino de programação concorrente e de testes de software, além de propor objetos de aprendizagem para auxiliar nesse projeto de aprendizagem.

Uma avaliação inicial mostrou que o “Quero Bolo” teve uma boa aprovação, sendo considerado um software didático e que facilita o aprendizado de programação concorrente. Este objeto de aprendizagem e o estudo de caso realizado foram descritos em um artigo submetido ao Simpósio Brasileiro de Sistemas de Informação.

6. Referências Bibliográficas

BARRIOCANAL, E. G.; URBÁN, M. S.; CUEVAS, I. A.; PÉREZ, P. D. An Experience in Integrating Automated Unit Testing Practices in an Introductory Programming Course. Department of Computer Science, DEI Laboratory. 2002

BERTOLINO, A. Software testing research: Achievements, challenges, dreams. In FOSE ’07: 2007 Future of Software Engineering, pages 85–103, Washington, DC, USA, 2007. IEEE Computer Society.

DELLA CORTE, C. K. Ensino Integrado de Fundamentos de programação e testes de Software. Dissertação de Mestrado, ICMC-USP, São Paulo, 2006.

GOETZ, B. Java Concurrency in Practice. Pearson, 2006.

DIJKSTRA, E.W. Notes on structured programming. Technological University Eindhoven, Netherlands Department of Mathematics, [Eindhoven], 1969.

JONES, E. L. J. e CHATMON, C. L. A Perspective on Teaching software Testing. Florida A&M University, 2001.

MANICKAM, V. e ARAVIND, A. If a picture is worth a thousand words, what would an animation be worth? In Proceedings of the 16th Western Canadian Conference on Computing Education, New York, NY, USA. 2011.

MASSOL, V. JUnit in Action. Manning. 2004.

ORTIZ, A. (2011). Teaching concurrency-oriented programming with erlang. In Proceedings of the 42nd ACM technical symposium on Computer science education, SIGCSE' 11, pages 195-200, New York, NY, USA. ACM.

SOMMERVILLE, I. Software Engineering. Addison-Wesley, 2006.

TOSCANI, S. S. Vale4 – Uma Linguagem Didática para Ensino de Programação Concorrente. 2004.