Desenvolvimento baseado em testes com aplicação de critérios de teste no ensino introdutório de computação

Yuri Rafael Grajefe Feitosa
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, PR, Brasil
yurirafael@alunos.utfpr.edu.br

Marco Aurélio Graciotto Silva
Universidade Tecnológica Federal do Paraná, Campo Mourão, PR, Brasil
magsilva@utfpr.edu.br

José Augusto Fabri
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, PR, Brasil
fabri@utfpr.edu.br

Ensinar os alunos a programar é um processo complexo [7]. De modo geral, existe uma linha de desenvolvimento que deve ser seguida, começando pela lógica, depois a lógica de programação e por fim os algoritmos [3]. Dentre as abordagens para ensino de programação [9], o teste de software pode ser considerado para tratar a resolução de problemas [4]. Assim, observam-se indícios de que o teste de software criterioso pode contribuir no processo de aprendizagem e desenvolvimento de código [11].

Um caso de teste consiste em um conjunto de entradas, condições de execução e saída esperada para um programa [5]. A reflexão sobre a elaboração de um caso de teste na fase introdutória do ensino de programação pode induzir o aluno a pensar melhor na resolução de um determinado problema computacional e, consequentemente, chegar a uma resposta de forma eficiente, com menos tentativas e erros no percurso [2]. Em estudo correlato, observou-se que o uso de critérios de teste em conjunto com o aprendizado dirigido a teste contribuem para a melhoria da qualidade dos programas e dos casos de teste [8]. No entanto, naquele trabalho foi considerado um conjunto limitado de critérios de teste e não foi considerada a relação entre casos de teste, requisitos de teste e objetivos de aprendizagem associados a cada atividade de aprendizagem.

O objetivo desta pesquisa consiste em estabelecer uma abordagem de desenvolvimento orientado a testes (TDD – *Test-Driven Development*) que empregue critérios de teste de forma escalonada e associada a objetivos de atividades de aprendizagem de programação. Desta forma, busca-se aperfeiçoar o desenvolvimento da lógica computacional dos alunos nos primeiros degraus de aprendizagem de programação nos cursos de computação e áreas afins.

A abordagem proposta consiste em conciliar os critérios de teste de software ao TDD na elaboração da lógica computacional do problema proposto com a finalidade de reduzir a curva de aprendizado do aluno. TDD é uma técnica iterativa de design de software associada a teste de software organizada em três etapas: definição dos casos de teste, implementação do código e refatoração [1]. Um critério de teste permite avaliar o software quanto a um determinado tipo de erro, podendo ser associado a objetivos de atividades de aprendizagem. Desta forma, critérios de teste de software podem ser utilizados para orientar o aluno a pensar em resolver problema de maneira eficiente, evitando abordagens de tentativa e erro.

Como exemplo, considere o cenário de alunos ingressante dos cursos de computação (CS1), submetidos a uma série de exercícios, na linguagem C, com a finalidade de avaliar o método proposto. Essas atividades têm o nível de dificuldade escalonado, começando com a soma de dois números, depois um problema que envolva estrutura de decisão, após isso, decisão e repetição, assim por diante. Tais atividades são avaliadas de forma automática por juízes online, considerando conjuntos de casos de teste e implementação de referência, ambos definidos pelo instrutor, que satisfazem determinados critérios e que são pertinentes aos objetivos de aprendizagem.

Para cada exercício, o aluno deve criar casos de teste antes da implementação do programa. Como é uma disciplina introdutória à programação, não seria razoável que definisse casos de teste automatizados. Como alternativa, cada caso de teste deve ser especificado de forma simplificada, informando-se apenas os dados de entrada e o resultado esperado como parte da documentação [6, 10]. Cada caso de teste deve ser processado pela ferramenta associada à abordagem proposta, fornecendo os dados de entrada como parâmetros da função ou do programa e comparando o resultado obtido com o resultado esperado. Os resultados podem ser obtidos tanto em relação ao programa do aluno quanto à solução de referência.

Em um primeiro momento, é esperado que o caso de teste falhe em relação ao código do aluno (dada a sua inexistência). Neste instante, ele prossegue para a implementação de sua solução até que o caso de teste passe. A execução do caso de teste com relação à solução de referência deve ser utilizada para discernir um erro de programação de um erro de compreensão do problema, o que deve ser informado após repetidas tentativas sem perceber o equívoco. Quando a solução do aluno satisfizer os casos de teste criados, os resultados de falha do teste da solução dele com a solução de referência devem ser considerados para guiá-lo a criar novos casos de teste, explorando requisitos de teste e problemas típicos relacionados aos conceitos sendo aprendidos.

Como forma de verificação da eficácia da abordagem proposta, será feita a comparação com a abordagem tradicional de ensino, considerando instrumentos de avaliação como FCS1 [12].

Ao considerar os critérios de teste para auxiliar a construção de programas, a presente proposta busca induzir a melhora na aprendizagem dos alunos, principalmente no desenvolvimento da lógica computacional. Portanto, espera-se que os alunos pensem mais antes de resolver um problema e que consequentemente a solução desse problema seja feita da melhor forma possível.

EduComp'21, Abril 27–30, 2021, Jataí, Goiás, Brasil (On-line)

REFERÊNCIAS

[1] Kent Beck. 2002. *Test-Driven Development: By Example* (1 ed.). Addison-Wesley Professional, EUA. 240 pages.

[2] Stephen H. Edwards. 2004. Using software testing to move students from trial-and-error to reflection-in-action. In *35th SIGCSE Technical Symposium on Computer*

Science Education (35 ed.) (Norfolk, VA, EUA). ACM, New York, NY, EUA, 26–30. https://doi.org/10.1145/971300.971312

[3] André Luiz Villar Forbellone and Henri Frederico Eberspächer. 2005. (3 ed.). Pearson, São Paulo, SP, Brasil. 218 pages.

[4] Pei Hsia and Frederick E. Petry. 1980. A Framework for Discipline in Programming. *Transactions on Software Engineering* SE-6, 2 (March 1980), 226–232. https://doi.org/10.1109/TSE.1980.234479

[5] ISO, IEC, and IEEE. 2017. ISO/IEC/IEEE 24765:2017 – Systems and software engineering – Vocabulary. Padrão Internacional. , 525 pages. https://doi.org/10.1109/IEEESTD.2017.8016712

[6] Vesa Lappalainen, Jonne Itkonen, Ville Isomöttönen, and Sami Kollanus. 2010. ComTest: a tool to impart TDD and unit testing to introductory level programming. In *15th Annual Conference on Innovation and Technology in Computer Science Education* (15 ed.) (Bilkent, Ankara, Turquia). ACM, New York, NY, EUA, 63–67. https://doi.org/10.1145/1822090.1822110

[7] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (23 ed.) (Larnaca, Cyprus). ACM, New York, NY, EUA, 55–106. https://doi.org/10.1145/3293881.3295779

[8] Bruno Henrique Pachulski Camara and Marco Aurélio Graciotto Silva. 2016. A Strategy to Combine Test-Driven Development and Test Criteria to Improve Learning of Programming Skills. In *47th ACM Technical Symposium on Computing Science Education* (47 ed.) (Memphis, TN, EUA). ACM, New York, NY, EUA, 443–448. https://doi.org/10.1145/2839509.2844633

[9] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A Survey of Literature on the Teaching of Introductory Programming. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR 2007)* (Dundee, Scotland). ACM, New York, NY, EUA, 204–223. https://doi.org/10.1145/1345443.1345441

[10] Python Software Foundation. 1999. doctest – Test interactive Python examples. Software. https://docs.python.org/3/library/doctest.html

[11] Lilian Passos Scatalon, Jeffrey C. Carver, Rogério Eduardo Garcia, and Ellen Francine Barbosa. 2019. Software Testing in Introductory Programming Courses: A Systematic Mapping Study. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (50 ed.) (Minneapolis, MN, EUA). ACM, New York, NY, EUA, 421–427. https://doi.org/10.1145/3287324.3287384

[12] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1 Knowledge. In *42nd ACM Technical Symposium on Computer Science Education* (42 ed.) (Dallas, TX, EUA). ACM, New York, NY, EUA, 111–116. https://doi.org/10.1145/1953163.1953200