Brazilian Journal of Development

Code teacher: uma ferramenta para correção automática de trabalhos acadêmicos de programação em Java

Codeteacher: a tool for automatic correction of academic programming jobs in Java

Recebimento dos originais: 03/12/2018
Aceitação para publicação: 07/01/2019

Francisco Alan de Oliveira Santos
Especialista em Engenharia de Produção pela Centro de Ensino Unificado de Teresina
Instituição: Instituto Federal de Educação do Maranhão – IFMA / Campus Coelho Neto
Endereço: MA-043, s/n – Olho D'Aguinha, CEP 65620-000, Coelho Neto-MA, Brasil
E-mail: franciscoalan.santos@ifma.edu.br

Mardoqueu Sousa Telvina
Bacharel em Ciência da Computação pela Faculdade de Educação São Francisco
Instituição: Instituto Federal de Educação do Maranhão – IFMA / Campus Coelho Neto
Endereço: MA-043, s/n – Olho D'Aguinha, CEP 65620-000, Coelho Neto-MA, Brasil
E-mail: mardoqueu.telvina@ifma.edu.br

Heleno da Silva Souza
Bacharel em Ciência da Computação pela Faculdade de Educação São Francisco
Instituição: Faculdade de Educação São Francisco – FAESF / Campus Coelho Neto
Endereço: Rua Abílio Monteiro, 1736, CEP 65725-000, Pedreiras-MA, Brasil
E-mail: helenosilva9@gmail.com

RESUMO

Exercícios práticos são essenciais no aprendizado de uma linguagem de programação. Com o objetivo de auxiliar o professor na avaliação de exercícios de programação, este artigo apresenta o CodeTeacher, uma ferramenta de análise e avaliação automática de código-fonte como alternativa para a otimização do processo de ensino-aprendizagem de programação. A intenção é fomentar a discussão na comunidade educacional em Computação e propor uma solução no sentido de encontrar novas abordagens para esse desafio, contribuindo para o progresso do ensino de programação.

Palavras-chave: Avaliação Automática, Exercícios de Programação, Java.

ABSTRACT

Practical exercises are essential tolearn a programming language. With the aim of assisting the teacher in the assessment of programming exercises, this paper presents CodeTeacher, a tool for automatic analysis and evaluation of source code as an alternative for the optimization of the teaching-learning programming process. The intention is to call the educational computing community to discussion and propose a solution to find new approaches to this challenge, thus contributing to the progress of teaching programming.

Keywords: Automatic Assessment, Programming Exercises, Java.

1 INTRODUÇÃO

Brazilian Journal of Development

O ensino de programação envolve a retenção de vários conhecimentos, como: noções de lógica, técnicas de programação, uso correto da sintaxe e recursos de uma linguagem, além da aplicação de boas práticas de desenvolvimento de *software*, dentre outros. A avaliação desses quesitos requer uma análise minuciosa por parte do professor, uma vez que é necessária a revisão do código produzido pelos alunos para identificar a manifestação do conhecimento adquirido, geralmente materializado na forma de trabalhos práticos[Prior 2003]. Devido à quantidade de detalhes a serem observados,faz-se necessário o acompanhamento individual para uma aprendizagem mais acurada.[Tobar*et al* 2001].

Porém, a tarefa de avaliar o conhecimento do aluno a partir da análise do código constitui um desafio para o ensino de programação, principalmente quando as turmas são extensas e a quantidade de turmas por professor também - realidade comum em cursos de programação[De Oliveira et al 2015]. Tal realidade acaba prejudicando, senão inviabilizando,a capacidade de avaliação do professor, dado o grande volume de trabalhos a serem corrigidos e as restrições de tempo para correção e entrega de notas[Nunes 2004],além de ser uma atividade repetitiva, trabalhosa e que pouco acrescenta ao docente. Por causar uma sobrecarga de atividades ao professor, tais fatores tendem a afetar a qualidade das avaliações.

Diante dessa dificuldade, surgiram alternativas para otimização desse processo por meio da automatização e execução de testes de código-fonte [Hollingsworth 1960, Ebrahimi 1994]. A tentativa de substituir a análise visual e execução manual de programas trouxe à luz ferramentas de correção assistida de trabalhos de programação. Contudo, embora tenha ocorrido uma considerável evolução desses sistemas [Romli et al. 2010], de acordo com De Oliveira et al.(2015), ainda há deficiências consideráveis com relação à efetividade dessas ferramentas, principalmente no que tange em avaliar se os objetivos educacionais almejados foram de fato alcançados.

Este artigo apresenta uma ferramenta de análise e avaliação automática de código-fonte como alternativa para a otimização do processo de ensino-aprendizagem de programação.A intenção é incentivar a discussão na comunidade educacional em Computação e propor uma solução no sentido de encontrar novas abordagens para esse desafio, contribuindo para o progresso do ensino de programação.

O texto está organizado como segue: Na Seção 2, apresentamos os trabalhos relacionados. Na Seção 3, apresentamos o *CodeTeacher* e descrevemos os tipos de análise que o compõem. Na Seção 4, explicamos funcionamento da ferramenta. A Seção 5 traz um estudo de caso para averiguar a viabilidade da ferramenta. E por fim, na Seção 6, concluímos com as considerações finais e trabalhos futuros.

Brazilian Journal of Development

1.1 TRABALHOS RELACIONADOS

Dentre as muitas ferramentas de apoio à prática da programação com as finalidades de submissão, execução e avaliação de exercícios, destacam-seo ProgTest, o PCodigo, o BOCA e o MOJO.

O ProgTest[De Souza et al. 2011] é um sistema de apoio automatizado à avaliação de submissões de programas escritos em Java. Juntamente com os programas também são submetidos seus respectivos casos de testes. O ProgTestcompila os programas do aluno e submete-os à execução dos testes.

O PCodigoé um complemento do Ambiente Virtual de Aprendizagem (AVA)Moodle[Moodle 2011]para execução em massa e deanálise de programas [De Oliveira et al 2015], desenvolvido na linguagem C, porém é aplicável a diferentes linguagens de programação. Integrado ao Moodle, recebe soluções de atividades de programação submetidas por alunos, executa-as e emite relatórios de avaliação para professores.

O BOCA *On-line Contest Administrator*[Campos and Ferreira 2004, França et al. 2011] é um sistema de internet para submissão de exercícios e correção *on-line* de código. É a plataforma atual usada nas competições de programação promovidas pela Sociedade Brasileira de Computação (SBC).

MOJO [Chaves 2013]é uma ferramenta que integra o conceito os Juízes On-line (JO) [Kurnia et al. 2001] ao Moodle. Consiste em um módulo que visa auxiliar o professor no processo de elaboração, submissão e correção de questões de programação.

O diferencial do *CodeTeacher* em relação às soluções já existentes, além de ser focado em Java, é que ele consiste em dar mais flexibilidade à avaliação do professor, permitindo uma análise mais holística. Além de orientar a avaliação por uma perspectiva pedagógica [Ihantola et al. 2010].Também possui a característica de ser extensível, isto é, sua arquitetura modular favorece a inclusão de novos recursos.

Consequentemente, com a possibilidade de proporcionar *feedback* imediato para o aluno, a ferramenta tem o potencial de obter maior engajamento deste, pois a conscientização do aluno sobre sua situação tende a causar uma reação no sentido de obter melhores resultados. Além de possibilitar maior transparência, tornando visual o processo de avaliação, expondo os resultados e reportando todos os interessados no aprendizado.

Para o professor, há ainda a possibilidade de percepção das deficiências de uma turma através da identificação de erros recorrentes em uma mesma avaliação, bem como a oportunização de um acompanhamento mais próximo do progresso da disciplina, com maior clareza sobre o rendimento individual e coletivo da turma.

Brazilian Journal of Development

1.2 CODE TEACHER

O *CodeTeacher*é uma aplicação *desktop* feita em Java, portanto independente de plataforma, que se fundamenta na abordagem de análise estática, dinâmica e estático-dinâmica[De Oliveira 2015]. É um sistema de análise de código-fonte baseada na configuração de critérios pré-definidos de avaliação. A aplicação foi feita para aceitar projetos em Java, independente da IDE de desenvolvimento utilizada, pois só precisa que as classes compiladas (arquivos com a extensão *.class*) sejam submetidas à apreciação da ferramenta.

Para detecção automática de inconformidades no código a partir da definição de critérios avaliativos pré-configurados pelo professor, é utilizado o recurso da programação reflexiva ou metaprogramação[Horstmann 2000]. Esse paradigma provê a capacidade de examinara estrutura e o estado (metainformações) de um programa e o poder dealterar seu comportamento em tempo de execução. Essa funcionalidade é provida por algumas linguagens de programação, dentre elas, o Java.

O foco atual de utilização do *CodeTeacher* concentra-se nas disciplinas iniciais e intermediárias de programação.É importante salientar também que a ferramenta abrange somente trabalhos práticos de codificação em Java, não abraçando análise textual de respostas a exercícios e questões subjetivas.

Quatro tipos de avaliação são possíveis, denominadas conforme segue:análise estrutural, análise comportamental,análise de saída padrão e análise conceitual. A seguir serão descritos os tipos de análise do *CodeTeacher*.

1.3 ANÁLISE ESTRUTURAL

Neste tipo de análise, são verificados os elementos estáticos do código, como declaração correta dos atributos e métodos de uma classe, podendo também ser verificados os seus modificadores de acesso, bem como se determinado membro é de classe ou instância.

Outra possibilidade é a análise do uso de herança por meio da navegação pela hierarquia de classes implementada pelo aluno. De modo similar,esta análise permite verificar se determinada classe implementa ou não uma dada interface. É possível ainda flexibilizar a análise considerando apenas alguns elementos na avaliação. Por exemplo, pode-se configurar um critério que avalie se há,no escopo definido, algum método que retorne dados e/ou receba parâmetros de determinados tipos, não sendo necessário informar o nome do método a ser buscado.

Para tornar mais livre a implementação do aluno, semcontudo, prejudicar a efetividade da avaliação, é possível lançar mão de expressões regulares, usando coringas na especificação de critérios. Pode-se, por exemplo, definir que deve existir um método cujo nome inicie com um

prefixo tal como “cadastrar...”. Para isso, seria preciso apenas informar no nome do método a seguinte expressão: “cadastrar*”,onde o asterisco (*) significaria qualquer cadeia de caracteres. Similarmente, seria possível verificar a existência de um método com o sufixo “?Dados”, sendo o sinal de interrogação correspondente a qualquer caractere.

1.4 ANÁLISE COMPORTAMENTAL

Para Ala-Mutka (2005) e Rahman et al. (2008), corretude e funcionalidade são importantes itens de avaliação. A análise comportamental consiste em atestar a corretude do código a partir de sua testagem funcional com vistas a simular o comportamento do programa em um ambiente ou cenário real.É uma análise baseada em entrada/saída que testa os serviços providos por um objeto, isto é, verifica-sea saída correta de um programa a partir de entradas fornecidas previamente e compara-se com resultados previstos. O código é executado e são checadas as respostas do objeto a estímulos externos, para isso, mensagens são passadas a um objeto no intuito de encontrar uma resposta esperada.O professor modela um conjunto de casos a serem avaliados e submete-os à apreciação da ferramenta e, através da combinação das entradas fornecidas e saídas previstas, é possível inferir a qualidade do programa quanto às suas funcionalidades, sendo possível com isso afirmar se ele corresponde aos requisitos especificados pelo professor.

Um exemplo de uso dessa abordagem pode ser a checagem do retorno de métodos a partir da invocação com passagem de uma lista parâmetros pré-estabelecidose da definição dos respectivos retornos esperados.

1.5 ANÁLISE DE SAÍDA PADRÃO

Nesta análise verifica-se se o código executado realiza a impressão de algum texto na saída padrão do sistema (*console*). É comum em disciplinas iniciais de programação a criação de programas que escrevam dados ou mensagens na saída padrão. Geralmente são dados resultantes de cálculos efetuados pela aplicação ou mesmo mensagens informativas. Normalmente essas informações são apresentadas de forma textual em uma interface de linha de comandos.A ferramentacaptura a saída padrão, interrompendo o fluxo de impressão e desviando-o para um *proxy* que armazena o conteúdo impresso e, em seguida, verifica se o que é impresso pelo programa do aluno é equivalente ao texto definido no critério de avaliaçãoconfigurado pelo professor. Este deve informar o conteúdo a ser impresso para que a comparação de igualdade seja realizada e, nesse caso, é considerado sucesso a correspondência exata entre os termos comparados.

Brazilian Journal of Development

1.6 ANÁLISE CONCEITUAL

Neste tipo de análise podem ser definidos padrões e métricas a serem contemplados. Esse mecanismo permite avaliar a aplicação de conceitos da Programação Orientada a Objetos (POO)como emprego de herança, polimorfismo, grau de encapsulamento, entre outros [Horstmann 2000]. Dessa forma, é possível descobrir se há classes abstratas que não são estendidas ou interfaces não implementadas, por exemplo. O uso de polimorfismo pode ser identificado e avaliado considerando, por exemplo, a presença de métodos sobrecarregados e/ou sobrescritos. Tais fatores podem ser enquadrados em uma escala de gradação definida pelo professor.Como exemplo, pode-se medir o nível de encapsulamento configurando um percentual mínimo de membros encapsulados a ser alcançado.

Embora ainda não estejam disponíveis no *CodeTeacher*, outros conceitos e métricas podem ser usados neste tipo de análise, como grau de coesão e acoplamento, pois novas métricas e conceitos podem ser agregados à ferramenta na forma de adição de funcionalidade, com a inclusão de *plug-ins* de extensão.

2 ETAPAS DA AVALIAÇÃO

A seguir são descritos os passos para utilização da ferramenta, todos os pormenores envolvendo o processo total de avaliação são detalhados nos tópicos a seguir. O fluxo de atividades é mostrado na Figura 1.

Figura 1. Etapas da avaliação

2.1 SELEÇÃO DE ARTEFATOS

Nesta etapa são indicados quais serão os elementos escolhidos para serem avaliados. Os ativos são selecionadosexplicitamente pelo usuário, primeiramente, através da indicação de quais diretórios devem ser acessados para buscar os artefatos de código a serem analisados, podendo ser, projetos, classes, ou pacotes de classes. Essa indicação determina o escopo da avaliação. A convenção utilizada pelo *CodeTeacher* para associar um determinado conjunto de artefatos de código a um aluno específico é que todos os arquivos contidos em uma pasta pertencem a um

Brazilian Journal of Development

mesmo aluno. Por isso, recomenda-se que haja uma pasta com o nome de cada estudante, pois o nome da pasta servirá de referência para identificar o aluno.

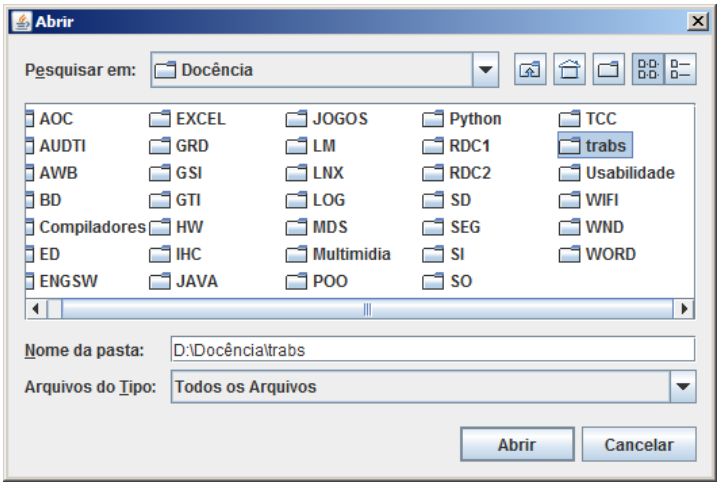

Figura 2. Seleção de artefatos

2.2 CONFIGURAÇÃO DE CRITÉRIOS

Uma vez que os elementos de *software* foram escolhidos, prossegue-se com a criaçãodos itens que irão compor a avaliação. Assim, um conjunto de critérios deve ser informado pelo professor para serem aplicados durante a análise. Os critérios são definidos usando-se a interface gráfica, onde são indicados quais serão os itens avaliativos considerados. O formato dos critérios varia de acordo com o tipo de análise, porém cada critério deve possuir um valor para compor a nota do aluno. A valoração dos critérios é estabelecida conforme são atribuídos pesos de acordo com o grau de importância considerado pelo avaliador. É neste momento que o professor alimenta o sistemacom seu julgamento prévio das competências esperadas do aluno com a realização da atividade prática em questão, no sentido de alcançar os objetivos de aprendizagem estabelecidos.

Depois de criados os critérios, há a possibilidade salvá-los em um arquivo para uso ou edição posterior, evitando assim que o trabalho de configurar todos os critérios seja repetido no caso de uma avaliação posterior.

Brazilian Journal of Development

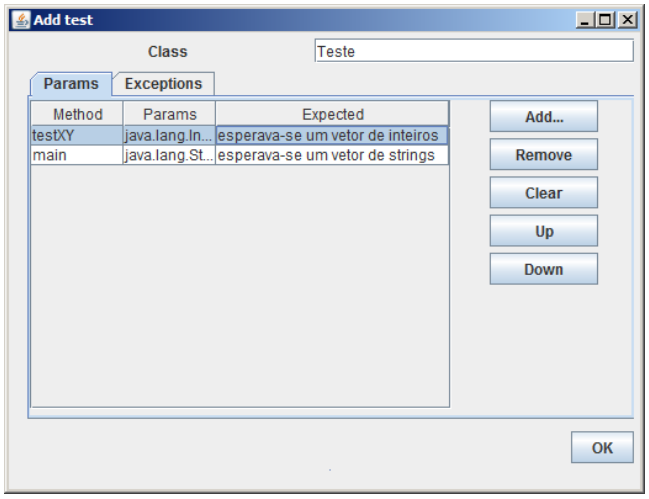

Figura 3. Configuração de critérios

2.3 EXECUÇÃO DE TESTES

A estratégia de execução é definida pelo professor, no sentido de tornar flexível o processo de avaliação. A execução dos testes pode ser em massa ou classe a classe, podendo ser alterada posteriormente, de forma que seja possível aplicar avaliações diferenciadas e com foco em aspectos determinados, a critério do professor. As notas dos alunos são calculadas de acordo com a quantidade total de pontos, obtida através do somatório de todos os critérios. Considera-se que, antes do início da execução, cada aluno possua todos os pontos, porém à medida que vão sendo encontradas irregularidades no seu código, os pontos correspondentes aos critérios não atendidos são debitados do total de pontos do aluno. Por fim, a nota final é contabilizada baseada no percentual de acertos do aluno.

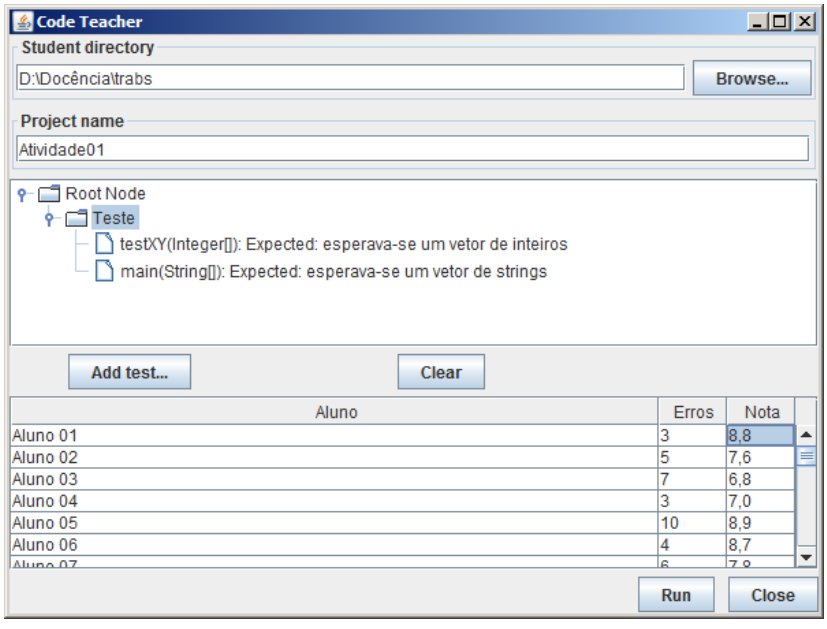

Figura 4. Execução de testes

Brazilian Journal of Development

2.4 RELATÓRIO DE DESEMPENHO

Após o término da execução de todos os testes, avança-se para a fase de predição das notas. Os resultados são tabelados e é gerado um relatório de desempenho onde constam as notas dos alunos da turma. Inicialmente o relatório é apresentado na sua forma resumida, contendo um sumário das notas, com informações estatísticas como média da turma e desvio-padrão. Porém, é possível analisar o desempenho individual obtendo um detalhamento do resultado de cada aluno. Outras opções de visualização também são disponibilizadas, como por exemplo, um extrato dos tipos de erros mais cometidos, percentual de acerto, índice de aproveitamento, entre outros. Com essasinformações complementares é possível identificar dificuldades generalizadas na turma e planejar estratégias para sanar tais deficiências.

Nesta fase, háainda uma opção de exportação dos dados resultantes para uma planilha, caso seja necessário manipular tais dados para geração de gráficos, por exemplo, ou mesmo para reportar os alunos acerca de seu desempenho. A geração do arquivo também serve de evidência documental para indicar a evolução da turma no decorrer da disciplina ministrada.

```
Tipo                  |Valor| Descrição
Método não encontrado |  5  | Método testXY(Integer[]) não encontrado na classe Test
Método não encontrado |  5  | Método main(String[]) não encontrado na classe Test
Total Erros: 10
Percentual Acertos: 80
Nota: 8.0
```

Figura 5. Relatório de Desempenho

3 ESTUDO DE CASO

Para atestar a efetividade do *CodeTeacher*como objeto de aprendizagem, ele foi adotado experimentalmente em uma instituição de ensino real, onde o conjunto de testes foi composto de respostas de exercícios de programação de uma turma regular do curso técnico em informática.Foram selecionadas 5atividades práticas de programação paracompor o corpus doexperimento.Para cada atividade foram coletadas respostas em formato de código-fonte, que foram utilizadascomo a base de soluções, totalizando 100 respostas.

Para a experimentação da solução proposta, as respostas foram submetidas às análises estrutural, comportamental, de saída padrão e conceitual, respectivamente, sendo a nota final a média aritmética entre estas notas. Para cada análise foram elaborados critérios avaliativos como sendo respostas-modelo, que levavam em consideração aspectos tidos como relevantes em um aluno iniciante em programação, envolvendo o uso básico dos principais elementos do Java, como sintaxe da linguagem, aspectos procedurais e recursos básicos de orientação a objetos. Os valores dos critérios foram atribuídos individualmente e de forma arbitrária por dois professores.Caso

houvesse divergência nas atribuições de valor de cada professor, estas eram confrontadas e passavam por um processo de revisão onde os dois discutiam até chegarem a um consenso.

Posteriormente, as respostas foram avaliadas manualmente por dois professores,que atribuíram notas de 0 a 10. Cada professor recebeu um *checklist* com os itens avaliativos idênticos aos critérios definidos na aplicação. Os professores deliberadamente assinalaramcada item como "Atendido" ou "Não atendido"e o cálculo da nota foi obtido a partir do percentual de itens atendidos. Este conjunto de exercícios já avaliados compõe a base de comparação do experimento.

4 RESULTADOS

Foi feita uma análise comparativa entre as notas obtidas via *CodeTeacher* e as notas atribuídas pelos professores. A Figura2 ilustra essa comparação.

Análise Estrutural

Questão	#1	#2	#3	#4	#5
Grau de Similaridade (%)	90,69	98,45	85,35	99,89	93,36
Média	93,55				

Análise Comportamental

Questão	#1	#2	#3	#4	#5
Grau de Similaridade (%)	91,87	95,34	89,32	91,46	88,87
Média	91,37				

Análise de SaídaPadrão

Questão	#1	#2	#3	#4	#5
Grau de Similaridade (%)	90,21	92,35	96,89	96,33	97,99
Média	94,75				

Análise Conceitual

Questão	#1	#2	#3	#4	#5

Brazilian Journal of Development

Grau de Similaridade (%)	90,69	85,81	75,35	79,89	73,26
Média	81				
Média Geral (%)	90,17				

Figura 2. Análise comparativa

Comparando as notas previstas e as notas reais, acorreção automática obteve precisão de 90,17%,o que demonstra a viabilidade do uso da ferramenta com grau de assertividade considerável. Contudo, é notória a necessidade de novos experimentos para assegurar um maior nível de aderência. Acreditamos que a adoção da ferramenta pode trazer benefícios rumo à sua maturação. Destacam-se ainda algumas observações sobre os resultados: percebeu-se que o grau de similaridade das avaliações foi mais homogêneo quando as notas eram abaixo de 3,0. Essa similaridade também aumentava quando as notas eram mais próximas de 10.

5 CONSIDERAÇÕES FINAIS

Foi apresentado o *CodeTeacher*, umaproposta de auxílio automatizado ao ensino de programação focada no aumento da produtividade do professor. No intuito de dar uma maior dinamicidade ao processo educacional através do uso sistemático da ferramenta, espera-se alcançar a redução do tempo de correção dos trabalhos, deixando assim o professor livre para se dedicar a outras práticas docentes, como elaboração de atividades,pesquisas, planejamento de aulas e confecção de material didático, além de acompanhamento de alunos.

Foi conduzido um estudo para avaliar seudesempenho junto a alunos e professores. A avaliação se deu por meio do uso da ferramenta para obtençãodos dados relativos à sua utilização.Os resultados possibilitaram uma demonstração satisfatória, porém não definitiva, sobre a verossimilhança das notas.

Como trabalhos futuros, prospecta-se o desenvolvimento de uma interface *web* para disponibilização on-line do serviço, bem como sua integração a plataformas móveis, além do desenvolvimento de funcionalidades complementares que agreguem valor ao propósito da ferramenta, como a capacidade de detecção de plágios. Também está nos planos a possibilidade da ferramenta ser anexada a um AVA (Ambiente Virtual de Aprendizagem) como o Moodle.

REFERÊNCIAS

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programmingassignments. *Computer Science Education*, 15(2):83–102.

Chaves, José Osvaldo M., Castro, Angélica F., Lima, Rommel W., Lima, Marcos ViniciusA., Ferreira ,Karl H. A. (2013). MOJO: Uma Ferramenta de Auxílio à Elaboração,Submissão e Correção de Atividades em Disciplinas dePrograma̧ção.In *XXI Workshop de Educação em Computação (WEI) - SBC 2013*,Maceió, AL.

Campos, C. and Ferreira, C. (2004). Boca: um sistema de apoio para competições deprogramação. In *XII Workshop de Educação em Computação (WEI) - SBC 2004*,Salvador, BA.

De Oliveira, Márcia G.; Oliveira, E. . Abordagens, Práticas e Desafios da Avaliação Automática de Exercícios de Programação. In: 4o. DesafIE - Workshop de Desafios da Computação Aplicada à Educação, 2015, Recife, PE. Anais do 4o. DesafIE, 2015. p. 1-10.

De Oliveira, M. G., Marques Ciarelli, P., and Oliveira, E. (2013). Recommendation of programming activities by multi-label classification for a formative assessment of students. *Expert Systems withApplications*.

De Oliveira, Márcia G.; Nogueira, Matheus de Araújo; Oliveira, Elias. Sistema de Apoio à Prática Assistida de Programação porExecução em Massa e Análise de Programas.In: *XIV Workshop de Educação em Computação (WEI) - SBC 2015*,Recife, PE.

De Souza, D., Maldonado, J., and Barbosa, E. (2011). Progtest: An environment forthe submission and evaluation of programming assignments based on testing activities.In *Software Engineering Education and Training (CSEE T), 2011 24th IEEE-CSConference on*, pages 1 –10.

Ebrahimi, A. (1994). Novice programmer errors: language constructs and plan composition. *International Journal of Human-Computer Studies*, 41(4):457 – 480.

França, A., Soares, J., Gomes, D., andG.C.Barroso (2011). Um sistema orientado aserviços para suporte a atividades de laboratório em disciplinas de técnicas de programaçãocom integração ao ambiente Moodle. *RENOTE - Revista Novas Tecnologias naEducação*, 9(1).

Hollingsworth, J. (1960). Automatic graders for programming classes. *Commun. ACM*,3(10):528–529.

Horstmann, Cay S; Cornell, Gary. Core Java 2. Vol.1: Fundamentos. Makron Books, 2000.

Ihantola, P., Ahoniemi, T., Karavirta, V., and Seppälä, O. (2010). Review of recent systemsfor automatic assessment of programming assignments. In *Proceedings of the10th Koli Calling International Conference on Computing Education Research*, KoliCalling '10, pages 86–93, New York, NY, USA. ACM.

Moodle – "A Free, Open Source Course Management System for Online Learning."(2011). Disponível em http://moodle.org/. Acesso em 26 de Agosto de 2017.Disponível em http://moodle.org/. Acessoem 17 de Março de 2011.

Prior, J. C. "Online assessment of SQL query formulation skills". In Proceedings of the Fifth Australasian Conference on Computing Education. Adelaide, Australia. 2003.

Romli, R., Sulaiman, S., and Zamli, K. (2010). Automatic programming assessment andtest data generation a review on its approaches. In *Information Technology (ITSim),2010 International Symposium in*, volume 3, pages 1186 –1192.

Rahman, K. A., Ahmad, S., Nordin, M. J., and Maklumat, F. T. D. S. (2008). The Design of an Automated C Programming Assessment Using Pseudo-code ComparisonTechnique.

Tobar, C. M. et al. "Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação". XII Simpósio Brasileiro de Informática na Educação, Vitória, ES. 2001.