Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação

Fábio P. Alves, Patricia Jaques

Instituto de Informática – Universidade do Vale do Rio dos Sinos (UNISINOS)
Avenida Unisinos, 950 – 93.022-000 – São Leopoldo – RS – Brasil

fabiopachecoalves85@gmail.com, pjaques@unisinos.br

Abstract. *For most students, programming courses are challenging and often frustrating. They demand logical knowledge, which can take time to be acquired by the student. For teacher, the high workload can hamper the students monitoring, who always expect to have immediate feedback to their questions. Thus, as a way of supporting programming courses, both for students and teachers, this paper describes a virtual environment composed of resources, such as custom feedback, facilities for interaction between student and teacher and to help teachers to monitor students. This system has undergone a quantitative user experience evaluation with 24 students of a programming course, and had satisfactory results, indicating the system value.*

Resumo. *Disciplinas de programação são desafiadoras e muitas vezes frustrantes para os estudantes universitários. Elas exigem um raciocínio lógico que pode demorar a ser adquirido. Para o professor, a alta carga de trabalho pode prejudicar no acompanhamento dos alunos, que esperam contar com retornos imediatas para suas dúvidas. Como forma de apoiar as disciplinas de programação, tanto para alunos como para professores, este artigo descreve um ambiente virtual composto de recursos como feedback personalizado, facilidades para a interação e auxílio ao professor para acompanhamento dos seus alunos, funcionalidades não encontradas de forma integrada nos trabalhos relacionados. Este sistema passou por uma avaliação quantitativa de experiência de uso por 24 alunos durante um semestre, e obteve resultados satisfatórios, indicando o valor agregado pelo sistema.*

1. Introdução

O ensino de programação nos cursos de tecnologia da informação tem uma grande importância por ser um dos pilares para a vida profissional do aluno. Alunos com alguma experiência profissional em informática conseguem ter esta percepção, porém, isso não é tão claro para aqueles sem muita experiência [Borges 2000].

Para que o aprendizado de programação seja efetivo é fundamental que o aluno pratique os conteúdos vistos em aula, metodologia conhecida como "*learning by doing*" [Anzai e Simon 1979]. Sendo assim o professor elabora trabalhos com desafios graduais de programação e disponibiliza para seus alunos. Uma vez concluído estes exercícios pelos alunos, começa o maior trabalho do professor, que é a correção destes trabalhos e a sinalização dos problemas encontrados. Não sendo somente isto, no decorrer da execução dos exercícios é comum e até esperado que os alunos acionem o professor

para esclarecer suas dúvidas e dificuldades encontradas. Devido ao grande volume de trabalho, o professor, geralmente, não consegue esclarecer imediatamente os pedidos dos seus alunos. Por outro lado, os estudantes, quando não prontamente atendidos, acabam se desmotivando ou até mesmo ignorando o aprendizado desta matéria.

Para apoiar o ensino de programação e buscar minimizar estas dificuldades, professores buscam recursos e ferramentas computacionais que possam lhes auxiliar nessas tarefas. Uma das ferramentas empregadas são os Juízes *Online*. As plataformas de Juízes *Online* utilizadas em competições de programação são capazes de executar os códigos submetidos a elas e informar se o programa funcionou corretamente, ou seja, se ele retorna a saída esperada para o conjunto de entradas teste. Porém, não possuem o recurso de fornecer um retorno apontando o tipo de erro ocorrido, por que ele aconteceu e nem onde este erro ocorreu. Este recurso é inexistente devido à natureza da sua utilização. Em competições de programação não se pode apontar os problemas encontrados; o próprio competidor deve tentar descobrir os erros para submeter novamente.

Alguns trabalhos como JOnline [Santos e Ribeiro 2011], MOJO [Chaves *et al* 2013] e BOSS [Joy, Griffiths e Boyatt 2005] estão fazendo uso destes recursos para apoiar as disciplinas de programação, utilizando a automatização fornecida pelos Juízes *Online* para apresentar mensagens para o aluno de forma instantânea. Os recursos de Juízes *Online* são também empregados para permitir que os alunos submetam seus trabalhos para os seus professores, e para que estes possam ter um controle sobre o que está sendo entregue, como é o caso do BOCA [Campos e Ferreira 2004]. Dentre estes trabalhos, não foi observado um aprofundamento no uso de mecanismos que facilitem a interação entre aluno e professor sobre dúvidas e erros dos códigos do aluno, limitando os trabalhos à simples comunicação via *chat* ou através de fórum. Além disso, com relação à correção automática, a grande maioria dos trabalhos utiliza apenas o simples retorno emitido pelo Juiz *Online*, o que não é suficiente para apoiar a aprendizagem de programação.

Visto estes pontos, este artigo descreve *feeper*, um ambiente virtual para apoiar o ensino/aprendizagem de programação no ensino superior. O trabalho desenvolvido fez uso de um mecanismo para melhorar a comunicação entre aluno e professor, permitindo sinalizar partes do código fonte escrito. Além disso, o *feedback* fornecido pelo Juiz *Online* foi personalizado para apresentar mensagens que indiquem ao aluno onde aconteceu o erro, por que ele aconteceu e como corrigi-lo e para permitir a execução de diferentes tipos de teste no programa do aluno.

Como diferencial, além da correção automática, o trabalho proposto emite dicas sobre os erros gerados pelo programa do aluno, permitindo que este consiga evoluir na correção do seu programa. O aperfeiçoamento da interação entre aluno e professor também foi foco desse trabalho. O programa permite que o aluno possa adicionar marcações em seu programa que serão exibidas ao professor, facilitando o entendimento, de ambas as partes, sobre em qual linha exata do código o aluno está tendo dúvidas e/ou dificuldades.

Foi realizada uma avaliação experimental do *feeper* com um grupo de 24 alunos que empregou a ferramenta nas aulas de laboratório de uma disciplina de programação durante um semestre. Ao final do experimento, foi aplicado um questionário com questões fechadas, seguindo a escala *Likert*, para avaliar a experiência dos alunos e

professor no uso da ferramenta. Este questionário objetivou avaliar a usabilidade da interface do sistema bem como a impressão dos estudantes e professor sobre suas funcionalidades. Os resultados do questionários mostraram que a ferramenta atende os requisitos básicos de usabilidade e que a ferramenta apoiou significativamente a interação entre aluno e professor, auxiliou os alunos a identificar erros nos seus códigos e a entender o porquê, assim como auxiliou o trabalho do professor de acompanhamento e *feedback* aos estudantes.

2. Trabalhos Relacionados

JOnline é um Juiz *Online* didático para o ensino de programação [Santos e Ribeiro 2011]. A proposta deste trabalho foi a de utilizar uma plataforma já existente de Juiz *Online* denominada BOCA [Campos e Ferreira 2004] e agregar a ela novas funcionalidades, entre as quais está a apresentação de dicas em língua portuguesa, programação colaborativa, entre outros recursos.

O BOCA é um Juiz *Online* desenvolvido para suportar competições de programação e que também pode ser utilizado para apoiar disciplinas de programação, através da submissão e correção automatizada de trabalhos desenvolvidos pelos alunos [Campos e Ferreira 2004]. Com o foco em competições de programação, o BOCA permite, além de criar as competições, enviar dúvidas aos juízes de prova, onde eles podem responder através de uma interface específica. Dentro desta interface exclusiva, os juízes podem avaliar os códigos submetidos e, além de obter a resposta do autojulgamento realizado pelo Juiz *Online*, reavaliar a solução proposta e escolher a melhor resposta para devolver aos times. O BOCA permite a submissão de códigos em *C*, *C++* e *Java*. Por ser um *Juiz Online*, o BOCA apenas sinaliza se o programa está funcionando corretamente ou não, mas não aponta onde encontra-se o erro e a causa.

Outro trabalho interessante proposto é o MOJO, uma ferramenta para auxiliar o professor em disciplinas de programação [Chaves *et al* 2013]. No MOJO o foco é o professor. Ele visa diminuir consideravelmente a sobrecarga de trabalho do docente na correção e apresentação dos resultados das atividades desenvolvidas pelos alunos. Como o foco é no professor, o MOJO recebe a resposta do Juiz *Online* e apresenta para o professor, para que este então dê um *feedback* para os seus alunos.

Lançado em 2005, o BOSS é um sistema de submissão e avaliação de código fonte que suporta a avaliação de trabalhos por meio da coleta de submissões, executando testes automatizados para correção e qualidade, verificando plágios, e fornecendo uma interface para obtenção de *feedbacks* [Joy, Griffiths e Boyatt 2005]. O BOSS realiza dois tipos de teste automatizado. O primeiro define entradas e saídas e executa o código inserindo as entradas e comparando as saídas, clássico teste caixa preta realizado pelos Juízes *Online*. Já o segundo tipo é aplicado somente para a linguagem *Java*, pois utiliza o *JUnit* [Beck e Gamma 2010], um framework com suporte à criação de testes automatizados para *Java*. Além disso, essa ferramenta permite aos professores executarem os testes automatizados, detectar plágios, visualizar os códigos submetidos, e fornecer um *feedback* para os alunos.

Como diferencial neste trabalho, para a interação entre alunos e professor, foi desenvolvida uma funcionalidade que permite registrar dúvidas e comentários diretamente nas linhas do código fonte do aluno, permitindo que ambos possam visualizar e discutir sobre o código fonte, diferente dos demais trabalhos relacionados

que fazem uso de *chats* simples e fóruns. Outro diferencial neste trabalho é a forma como os exercícios são cadastrados pelo professor, que permite atingir um alto nível de detalhamento na correção das respostas submetidas pelos alunos, através do cadastro de classes de teste e também de massa de dados para testes de entrada e saída de informação.

3. O feeper

O sistema desenvolvido recebeu o nome de *feeper*, um acrônimo entre as palavras "***fee****dback* **per**sonalizado", recurso este, tido como principal característica do sistema. O *feeper* consiste em uma ferramenta *web* para apoiar a aprendizagem de programação em classe, assim como em extraclasse. Na ferramenta, são disponibilizados exercícios práticos que são organizados de forma a desafiar o aluno de forma gradual na resolução e também acompanhar a sua evolução desde o início do uso desta ferramenta.

Para responder aos exercícios, o aluno utiliza um editor de código fonte, podendo criar múltiplas classes e também, caso necessário, realizar o *upload* de classes existentes. Através deste editor, é possível adicionar dúvidas para enviar ao professor, registrar anotações vinculadas às linhas do código fonte, e marcar o código fonte como favorito para uma futura consulta em uma listagem de favoritos. A Figura 1 ilustra o editor de código fonte com as funcionalidades disponíveis ao aluno.

Figura 1. Tela para edição de código fonte

Durante a execução destes exercícios, o aluno pode submeter a sua resposta para validação. Esta validação é feita de forma imediata, utilizando a tecnologia do Juiz *Online* para executar testes automatizados a fim de determinar se a resposta do aluno está correta ou não. Caso ocorram problemas durante esta validação, seja de compilação ou de falha na execução dos testes, o Juiz *Online* emite uma resposta personalizada. Quando um erro de compilação acontece, o Juiz *Online* captura o erro gerado pelo compilador para exibir ao aluno. Quando um erro de execução acontece, uma mensagem personalizada, editada pelo professor, é emitida. Para cada linha de teste

executado é possível emitir uma mensagem personalizada. Estes testes são divididos em duas partes, onde primeiro é executado o teste caixa preta, inserindo dados conhecidos, coletando as saídas geradas pelo programa e comparando com as saídas conhecidas e esperadas. O segundo modo de testes é a execução de classes de teste, onde para cada classe de teste cadastrada para o exercício é possível emitir uma mensagem personalizada de erro de execução e também uma mensagem personalizada de compilação.

A tecnologia do Juiz *Online* foi reaproveitada de um trabalho existente denominado CodeJudge [Guria, 2012], o qual foi selecionado devido à fácil utilização e customização. Ele foi desenvolvido para funcionar no sistema operacional *Linux* e interpretar as linguagens de programação *Java*, *C* e *C++*. Neste primeiro momento, foi apenas habilitado o uso da linguagem *Java*, deixando em aberto a possibilidade de expandir as linguagens as quais ele é capaz de interpretar. Em cima do CodeJudge foram aplicadas as personalizações anteriormente citadas.

A Figura 2 ilustra o processo executado pelo Juiz *Online*, exibindo as etapas de verificação, validação, execução de testes pré-cadastrados e emissão da resposta final sobre o código fonte do aluno. O processo inicia com a submissão do código fonte pelo aluno (conforme item (a) da Figura 2), o Juiz *Online* então solicita ao banco de dados os testes existentes para o exercício que o aluno está tentando responder (b). Os testes e os códigos são reunidos e executados teste por teste e, a cada execução, o Juiz Online compara a resposta gerada pelo programa do aluno com as respostas esperadas (c). Havendo falha na execução, então uma mensagem de erro é emitida ao aluno e o erro gerado pelo compilador é exibido. Havendo uma divergência entre a resposta gerada e a resposta esperada, então uma mensagem personalizada de alerta é emitida ao aluno, contendo orientações para que este consiga resolver o problema gerado (d). Essa mensagem de erro deve ser cadastrada pelo professor no seu módulo. Do contrário, então o programa é considerado correto e a resposta de sucesso é emitida ao aluno.

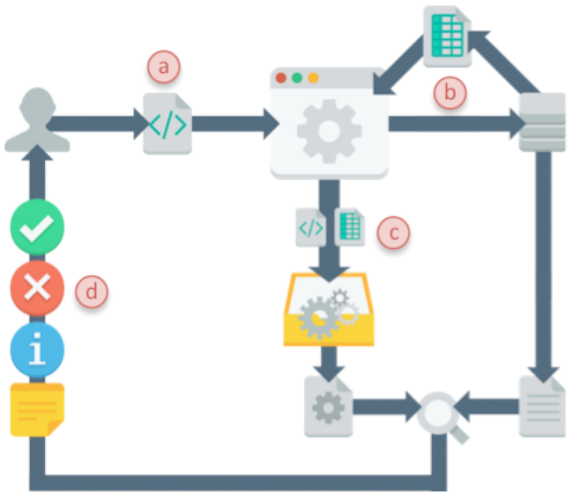

Figura 2. Fluxograma do Juiz *Online*

O aluno pode a qualquer momento registrar dúvidas para enviar ao professor. Estas dúvidas são vinculadas a uma linha de um código fonte (conforme linha 10 da Figura 1), permitindo que o aluno sinalize para o professor em qual linha do seu código ele está tendo dificuldades. O professor ao receber a mensagem consegue visualizar esta

marcação, e responder para ao aluno. Este recurso visa facilitar a comunicação entre o aluno e o professor, permitindo também que o professor ao revisar os códigos dos alunos, possa registrar os seus comentários e observações para os alunos. A Figura 3 ilustra um diálogo entre aluno e professor, registrado através desta funcionalidade.

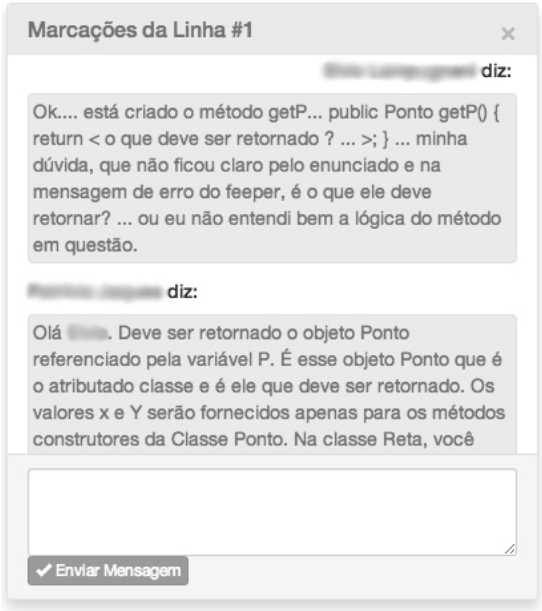

Figura 3. Conversa com o professor

Através do painel de acompanhamento de resultados, o aluno visualiza o *status* de cada exercício. Para cada exercício, é registrado um histórico de tentativas de submissão de respostas, exibindo todas as mensagens obtidas durante as sucessivas tentativas de resolução do exercício. Essa funcionalidade permite que o aluno consulte códigos antigos, visualizando estes através da ferramenta ou então através do *download* destes.

Para auxiliar o professor, o seu módulo conta com funcionalidades para acompanhar as tentativas dos alunos de solucionar os exercícios, permitindo que o professor interaja com o aluno, registrando comentários nos códigos dos alunos de forma simples e direta. Todas estas mensagens enviadas e recebidas estão disponíveis dentro da caixa de mensagens, desenvolvida como forma de centralizar todas as conversas entre professor e alunos. O professor pode também gerenciar a sua turma, para adicionar novos exercícios e liberá-los gradativamente conforme necessidade. Estes exercícios são mantidos pelo professor, podendo este inserir novos, modificar e até excluir exercícios. Através do cadastro dos exercícios, o professor poderá montar toda a inteligência para a correção do exercício, fornecendo mensagens que irão ajudar seus alunos quando estes encontrarem problemas para resolver o exercício.

Em um painel de resultados, o professor consegue visualizar o andamento dos exercícios, onde através de uma grade que exibe todos os alunos e todos os exercícios da turma é possível visualizar o *status* do exercício para cada aluno, se o estudante já conseguiu resolver o exercício ou se exercício está com algum outro problema. Ao clicar nesta sinalização, a ferramenta apresenta um histórico de respostas submetidas pelo aluno, onde todos os detalhes são exibidos, como a data da submissão, os códigos fontes utilizados e o *feedback* que o sistema apresentou para o aluno. A Figura 4 ilustra

a tela de resultados com o detalhamento de execução de determinado exercício para um estudante em específico.

Figura 4. Tela de resultados com detalhamento de execução

4. Avaliação

Uma avaliação experimental foi realizada em uma turma com 24 alunos de uma disciplina de programação em laboratório do primeiro semestre dos cursos de tecnologia da informação de uma universidade da grande Porto Alegre. O *feeper* foi utilizado pelos alunos e pelo professor durante um semestre de aula, mais especificamente dois meses. Após este período, para avaliar a experiência no uso da ferramenta, foi aplicado um questionário diferente para alunos e professor. As perguntas foram montadas utilizando a escala de *Likert*, onde os alunos deveriam escolher um valor entre 1 a 5 (discordo totalmente a concordo totalmente). Os dados foram analisados com a medida por moda (*Mo*) ou a resposta mais frequente, assim como a média (μ). Dos 24 alunos que utilizaram a ferramenta, apenas 17 responderam esse questionário final.

O questionário do aluno foi dividido em partes, onde a primeira parte serviu para identificar o aluno, e o seu nível de conhecimento em programação antes de cursar as cadeiras de programação e o seu nível de conhecimento atual. Foi possível então mapear a evolução que o aluno teve no decorrer do seu estudo, onde a pesquisa mostrou que o conhecimento inicial passou de nenhum ou muito baixo para um conhecimento intermediário. Por ser uma disciplina do primeiro semestre de curso, já era esperado que o conhecimento do aluno não atingisse o nível avançado.

Na segunda parte da pesquisa foram aplicadas perguntas relacionadas com a usabilidade da ferramenta. O resultado da pesquisa mostrou que os alunos tiveram muita facilidade para aprender a utilizar o *feeper* (*Mo*=5; μ=4,3) e que o *layout* da ferramenta facilitou muito o entendimento das funcionalidades (*Mo*=5; μ=4,4). Os alunos também indicaram a grande importância que um *layout* bonito e agradável tem em ferramentas

que objetivam o aprendizado (*Mo*=5; μ=4,4). De uma forma geral, a interatividade do *feeper* foi avaliada entre intermediária e boa (*Mo*=4; μ=3,6).

A última parte da pesquisa focou nas funcionalidades disponibilizadas pelo *feeper* e também na utilidade do uso de ferramentas que apoiam o aluno nas cadeiras de programação. A grande maioria dos alunos, de acordo com a pesquisa, acham que o uso de ferramentas para apoia-los nas cadeiras de programação tem muita utilidade (*Mo*=5; μ=4,5). A utilidade do *feeper* no aprendizado foi avaliada como intermediária, tendendo para boa (*Mo*=3; μ=3,5). A funcionalidade para envio (e recebimento) de mensagens ao professor foi muito bem avaliada, ficando empatada entre muito útil e útil (*Mo*=5; μ=3,7). A grande maioria dos alunos julgou que o retorno fornecido pelo *feeper* após corrigir seus códigos fonte foi útil (*Mo*=4; μ=3,6), porém as mensagens que validam as respostas objetivando ajudar a solucionar o erro teve dois picos na avaliação, no primeiro sendo avaliadas como úteis e no segundo pico avaliadas como um nível abaixo de intermediário (*Mo*=4; μ=3,4). É importante lembrar que as mensagens de *feedback* são cadastradas pelo professor e que, possivelmente, elas não foram tão efetivas pois era a primeira experiência do professor. Ao final do questionário foi disponibilizado um campo para que os alunos informassem sugestões de funcionalidades que julguem necessárias, onde surgiram ideias como apresentação de dicas e exemplos para solucionar os erros e a tradução de mensagens de erro geradas ao compilar o código fonte.

O questionário do professor também foi dividido em duas partes. Apenas um professor, o professor da turma onde o *feeper* foi aplicado, respondeu este questionário. Como diferencial, o questionário do professor tem, para cada pergunta na escala de *Likert*, um campo de texto para que o professor justifique sua resposta, permitindo agregar mais informação qualitativa no seu retorno.

A primeira parte do questionário apresentou as mesmas perguntas referentes à usabilidade da ferramenta e também à importância da usabilidade nesta categoria de ferramenta. De acordo com a pesquisa, o professor julgou ser fácil o aprendizado para usar a ferramenta, justificando que a ferramenta é bastante intuitiva, mas que estaria faltando apenas um tutorial para explicar o funcionamento do cadastro dos testes para os exercícios de programação, uma vez que o cadastro deve ser muito bem preenchido para gerar um retorno positivo ao aluno. O professor também julgou como muito importante que ferramentas desta categoria tenham o *layout* bonito e agradável. De forma geral, a interatividade e o layout aplicado no *feeper* foram muito satisfatórios.

A segunda parte do questionário foi referente as funcionalidades disponibilizadas para o professor e também na utilidade do uso de ferramentas que apoiam o aluno nas cadeiras de programação. O professor julgou que o uso de ferramentas para apoiarem os aprendizes é muito útil, pois podem fornecer um *feedback* mais individualizado e também mais rápido a eles. Com relação ao cadastro de exercícios, a funcionalidade foi avaliada como flexível, faltando uma ajuda para o professor, pois o cadastro de testes é bastante exaustivo e propício a erros humanos. O apoio fornecido pelo *feeper* aos alunos foi muito útil, uma vez que a ferramenta forneceu o retorno imediato aos alunos quando estes enviavam seus programas para validação. Outra funcionalidade que recebeu nota máxima foi a funcionalidade para envio de mensagens aos alunos, a qual foi muito elogiada pelo professor por ser extremamente útil, funcional e de fácil uso. A exibição do resultado do exercício foi

útil, mas, como depende diretamente do cadastro das validações feitas pelo professor, está suscetível a falhas. O tempo de resposta da ferramenta foi bem avaliado, com ressalva ao Juiz *Online*, que demorava um pouco a mais para corrigir o exercício do aluno. De acordo com a pesquisa, o professor ficou muito motivado para usar a ferramenta para apoiar as suas aulas de programação, justificando que ela facilitou muito seu trabalho na comunicação com os alunos para sinalizar que parte do código continha erro ou que precisava ser melhorado. Outro ponto ressaltado na justificativa do professor é que a ferramenta facilitou o acompanhamento das soluções dos alunos, permitindo que ele visualizasse todas as versões das respostas de forma extremamente rápida, agilizando a correção do exercício.

O questionário teve ainda uma pergunta listando todas as funcionalidades disponíveis para o professor, permitindo que este selecionasse múltiplas funcionalidades que mais lhe agradaram. Foram selecionadas as funcionalidades de envio de mensagens aos alunos, a visualização dos resultados dos exercícios e a recepção de dúvidas dos alunos permitindo a visualização do código fonte no mesmo momento.

Da mesma forma que no questionário disponibilizado para os alunos, o professor também pode sugerir outras funcionalidades importantes para melhorar o uso do *feeper*, tais como, uma funcionalidade para geração automática de testes, sistema de dicas em português para os alunos baseado nos principais erros de compilação dos alunos, inserção de técnicas de gamificação para aumentar a motivação dos alunos, e por fim uma evolução no editor de código fonte para permitir *debug* tanto para o professor quanto para o aluno.

5. Conclusão e Trabalhos Futuros

Este artigo apresentou uma ferramenta *online* para ser utilizada como apoio, para alunos e professores, em disciplinas de programação, oferecendo funcionalidades para tal. Dentre todas as funcionalidades, ganham destaque o recurso de *feedback* personalizado, onde, através de um cadastro de exercícios, o professor pode montar cenários de teste com mensagens personalizadas para quando o teste falhar. Esta mensagem é apresentada de forma imediata ao aluno, assim que este submeter sua resposta. Outro recurso muito importante apresentado neste trabalho é a facilidade criada para enviar mensagens ao professor e também para que o professor envie mensagens para seus alunos, a fim de comentar sobre partes específicas dos códigos dos programas dos seus alunos. Quando o estudante manda mensagens para o professor, esta mensagem é vinculada a um trecho de código fonte, o que permite destacar a linha a qual está ocorrendo sua dúvida. Para o professor, a facilidade de acompanhar a sua turma através de um painel de resultados dos exercícios, possibilita que este visualize através de uma grade como os seus alunos estão evoluindo na resolução dos exercícios, permitindo inclusive que o professor visualize um histórico de todas as respostas do aluno, com a data da resposta, os códigos utilizados e o *feedback* emitido pelo Juiz *Online*.

Os resultados deste trabalho foram obtidos através de uma avaliação da experiência de uso da ferramenta, onde um questionário com questões fechadas foi aplicado para os alunos e para o professor que usaram a ferramenta durante um semestre de uma disciplina de laboratório de programação de primeiro semestre de cursos em informática, resultados estes que, serviram para identificar se os objetivos foram ou não atingidos. A pesquisa indicou que as principais funcionalidades propostas foram úteis

para os avaliados, e que de fato agregaram para a sua experiência de uso. Com a avaliação também foi possível identificar pontos a serem melhorados, como por exemplo, automatizar a criação de cenários de teste, para diminuir o trabalho do professor, evitando também eventuais falhas humanas no momento do cadastro das mensagens de *feedback*, que são cruciais para uma boa avaliação dos exercícios. Outras sugestões dadas pelo professor e pelos alunos poderão ser adicionadas em trabalhos futuros.

Durante o desenvolvimento do *feeper* algumas funcionalidades foram identificadas como possíveis melhorias que possam ser aplicadas para aperfeiçoar o uso da ferramenta. Uma das principais melhorias identificadas, é o uso de gamificação como forma de motivação para os alunos. Outra funcionalidade que será bastante útil é a detecção de plágio, onde através de um componente será possível comparar códigos fonte históricos e obter um retorno que indique o percentual de chance de um código ser plágio.

Agradecimentos

O presente trabalho foi realizado com o apoio do CNPq.

Referências

Anzai, Y.; Simon, H. A., (1979). "The Theory of Learning by Doing", Carnegie-Mellon University, Pittsburgh.

Beck, K.; Gamma, E., (2010) "JUnit Cookbook", http://junit.sourceforge.net/doc/cookbook/cookbook.htm, novembro 2013, 15.

Borges, M. A., (2000) "Avaliação de uma metodologia alternativa para a aprendizagem de programação", Workshop de Educação em Computação, Anais do Congresso da SBC, Curitiba, PR.

Campos, C. P.; Ferreira, C. E., (2004) "BOCA: um sistema de apoio a competições de programação", Workshop de Educação em Computação, Anais do Congresso da SBC, Salvador, BA.

Chaves, J. O.; Castro, A.; Lima, R.; Lima, M. V.; Ferreira, K., (2013) "MOJO: uma ferramenta para auxiliar o professor em disciplinas de programação", Congresso Brasileiro de Ensino Superior a Distância, Belém, PA.

Guria, S., (2012) "CodeJudge", http://sankhs.com/codejudge, novembro 2013, 21.

Joy, M.; Griffiths, N.; Boyatt, R., (2005) "The BOSS Online Submission and Assessment System", ACM Journal on Educational Resources in Computing, v. 5, n. 3, pp. 1–28, setembro 2005.

Santos, J. C. S.; Ribeiro, A. R. L., (2011) "JOnline: proposta preliminar de um juiz online didático para o ensino de programação", XXII Simpósio Brasileiro de Informática na Educação, pp. 964–967, São Cristóvão, SE.