Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação

Leandro S. G. Carvalho, David B. F. Oliveira, Bruno F. Gadelha

Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Octávio, 6200 – Coroado I – CEP: 69077-000 – Manaus – AM – Brasil
{galvao,david,bruno}@icomp.ufam.edu.br

Resumo. *A alta reprovação em disciplina introdutória de programação para estudantes não ligados à área de computação levou um grupo de professores a adotar uma metodologia híbrida de ensino. Ela mesclava aulas presenciais com atividades baseadas em um juiz online desenvolvido por um dos autores. A ferramenta possibilitou que os estudantes praticassem mais exercícios de programação, com feedback imediato, proporcionando um aumento na taxa de aprovação. A percepção dos alunos, de forma geral positiva, também foi analisada, por meio do Método de Explicitação do Discurso Subjacente.*

Abstract. *The high failure rate in introductory programming course for non-information technology students led a group of teachers to adopt a blended learning approach. It balanced face-to-face lessons with activities based on an online judge developed by one of the authors. The tool enabled students to practice programming with immediate feedback, which provided a significant increase in the pass rate. The perception of students, in general favorable, was also analyzed through the Underlying Discourse Unveiling Method.*

1. Introdução

A disciplina de Introdução à Programação de Computadores (IPC) é ministrada para quatorze cursos de engenharia e de ciências exatas na Universidade Federal do Amazonas (UFAM). Nesses cursos, a programação exerce um papel de atividade-meio, e não de atividade-fim. Dessa forma, de maneira geral, esses estudantes apresentam uma menor motivação em aprender os conteúdos previstos na disciplina.

Em especial, IPC é ofertada durante o primeiro período letivo em oito desses quatorze cursos, para cerca de 300 alunos. Tal público apresenta duas características desafiadoras para o ensino de qualquer disciplina: falta de maturidade do estudante em perceber a importância do conteúdo para sua formação acadêmica e profissional; e evasão devida à falta de afinidade com o curso em que acabou de ingressar. Outras dificuldades são profusamente relatadas na literatura [Chaves et al. 2103, Paes et al. 2013, Pelz et al. 2012, Píccolo et al. 2010], e por isso são resumidas na

Figura 1. Elas contribuem de alguma maneira para que a taxa de aprovação em IPC seja baixa, cerca de 40%.

Para tratar tal problema, adotou-se uma metodologia de ensino híbrido (*blended learning*) de modo a efetivamente aumentar a taxa de aprovação em IPC. O objetivo do presente trabalho é apresentar a ferramenta de juiz online em torno da qual se fundamentou a metodologia de ensino híbrido, bem como os resultados atingidos.

DOI: 10.5753/cbie.sbie.2016.140

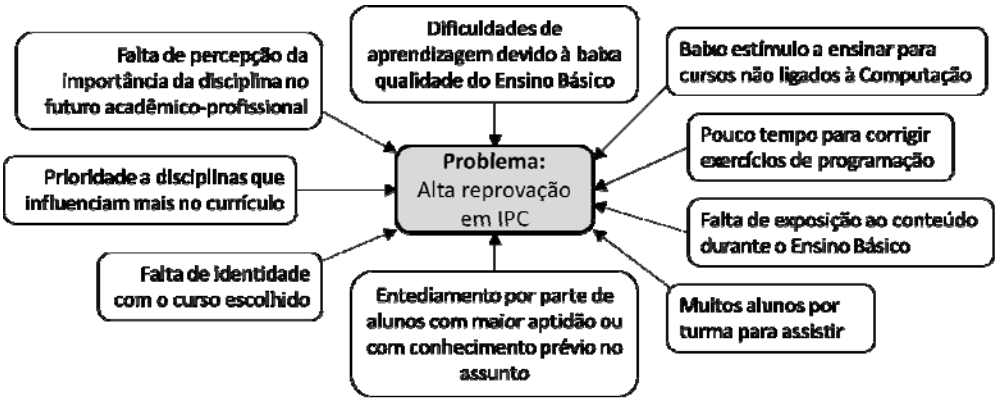


Figura 1 - Desafios ligados ao problema de pesquisa.

O artigo está organizado da seguinte forma: na Seção 2, descreve-se a metodologia híbrida adotada em 2015; a Seção 3 apresenta alguns juízes online disponíveis na literatura, enfocando-se na Seção 4 o desenvolvido e utilizado neste estudo de caso; a metodologia de avaliação da ferramenta é descrita na Seção 5, e os resultados discutidos na Seção 6; por fim, a Seção 7 apresenta as conclusões.

2. Metodologia híbrida de ensino-aprendizagem

Nesta seção, apresenta-se a fundamentação teórica para a adoção da metodologia híbrida de ensino-aprendizagem e, em seguida, esta é descrita em detalhes.

2.1 Planejamento da metodologia de ensino-aprendizagem

O aprendizado em disciplinas de programação baseia-se na prática e repetição regular de exercícios [Paes et al. 2013]. Porém, além do hábito contínuo, é vital que o estudante receba *feedback* rápido, a fim de localizar seus erros, compreender a origem destes e remediá-los. Assim, evita-se que os estudantes fiquem frustrados na tentativa de aprender os princípios da programação de computadores [Pelz et al. 2012].

Nesse mesmo sentido, Ihantola et al. (2015) afirmam que "a avaliação contínua durante um curso de programação garante que os estudantes pratiquem bastante, bem como obtenham *feedback* sobre a qualidade de suas soluções". Isso acontece porque a avaliação orienta a aprendizagem e serve de *feedback* tanto para o aluno quanto para o professor, seja de um tema em específico, seja do curso como um todo.

Dessa forma, os professores envolvidos no ensino de IPC constataram que, para resolver o problema da baixa taxa de aprovação, teriam que modificar a metodologia de ensino para uma abordagem mais prática. Contudo, verificou-se que a solução não se resumiria a simplesmente aumentar a quantidade de exercícios. O enunciado destes deveriam ser melhor concebidos, de tal forma a atingir os seguintes objetivos:

1. Oferecer diferentes contextos de aplicação da programação, a fim motivar os estudantes [Giraffa et al. 2015]; e
2. Exercitar uma diversidade de habilidades ligadas à programação, tais como rastreamento de código, identificação e correção de erros, construção de código, reutilização de código, entre outras [Hazzan et al. 2014].

Porém, verificou-se que mais tempo seria necessário para preparar enunciados variados e para corrigir os exercícios. Com efeito, adotar ingenuamente uma abordagem

prática exige mais tempo do docente, se os moldes tradicionais de ensino forem mantidos. Por isso, decidiu-se que a correção de exercícios teria que ser automatizada. No entanto, tal automatização deveria estar claramente articulada com as demais atividades envolvidas no ensino de programação. Portanto, buscou-se no paradigma de *Blended Learning* (ensino híbrido) as lições para fundamentar a metodologia de ensino-aprendizagem descrita na Seção 2.2.

Segundo Horn e Staker (2015), "ensino híbrido é qualquer programa educacional formal no qual um estudante aprende, pelo menos em parte, por meio do ensino online, com algum elemento de controle do estudante sobre o tempo, o lugar, o caminho e/ou o ritmo". Uma de suas principais vantagens é incorporar os benefícios das atividades de aprendizagem tradicionais face-a-face (síncrona) e online (assíncrona). Consequentemente, tal metodologia proporciona maior flexibilidade aos estudantes, tanto em termos de local como de espaço para a aprendizagem.

2.2 Descrição da metodologia de ensino-aprendizagem

Ferramentas

As seguintes ferramentas foram utilizadas como apoio à metodologia de ensino híbrido:

- O juiz online **CodeBench**, desenvolvido por um dos autores deste artigo e descrito na Seção 4. Foi utilizado para apoiar a atividade "laboratório de codificação" e as avaliações parciais. Por meio dele, os estudantes exercitavam as habilidades de elaboração, correção e reutilização de código.
- O ambiente virtual de aprendizagem (AVA) **Colabweb**, uma instância do ambiente Moodle. Seu propósito era apoiar a atividade "laboratório de exercícios" (*quiz*) e a avaliação final. Por meio dele, os estudantes realizavam exercícios conceituais e relativos à habilidade de rastreamento de código. Era também utilizado para divulgar informações complementares sobre a disciplina.
- **Spyder** (https://pythonhosted.org/spyder), uma IDE para a linguagem Python. Foi adotado por oferecer recursos que auxiliam o aprendizado do programador iniciante, tais como: destacador de sintaxe, enumerador de linhas, visualizador do estado de variáveis, bibliotecas Numpy e MathPlotLib já instaladas, e integração do editor de scripts e do *shell* de execução em uma só janela.

Equipe e papéis

Para implementar a metodologia híbrida, dividiu-se a equipe segundo três papéis:

1. **Professor**: docente da instituição, responsável por preparar e ministrar as aulas de abertura de módulo, e elaborar as questões dos laboratórios de codificação, das avaliações parciais e dos laboratórios de exercício (*quizzes*).
2. **Tutor**: mestrando ou doutorando, responsável por tirar as dúvidas dos estudantes durante os laboratórios de codificação e de exercícios, e auxiliar o professor na condução das avaliações práticas no juiz online.
3. **Gerente do juiz online**: mestrando, responsável por cadastrar os laboratórios e as avaliações no juiz online, elaborar scripts de resposta (gabarito) para todas as questões, bem como os respectivos casos de teste.

Na experiência de 2015, foram alocados um professor e um tutor para cada uma das três turmas que participaram do piloto. Uma única pessoa foi alocada como gerente

do juiz online para todas as três turmas. Se um professor ou um tutor precisasse se ausentar da instituição, podia ser substituído por seu respectivo par.

Balanceamento entre aulas presenciais e online

As aulas de IPC foram organizadas em um módulo **introdutório**, de duas aulas presenciais para familiarização com as ferramentas, e mais sete módulos **temáticos**, de quatro aulas cada. A primeira e a quarta aula de cada módulo temático eram presenciais. As aulas 2 e 3 eram de presença facultativa, pois requeriam o uso do juiz online e do AVA. As atividades podiam ser realizadas de forma individual ou colaborativa, mas a submissão era individual. O tutor estava sempre presente no laboratório de informática para atender os estudantes com dificuldade nos temas.

A divisão entre os laboratórios de codificação e de exercícios tinha finalidade de planejamento. Na prática, ambas as atividades eram liberadas na data de abertura do módulo, e encerradas no dia da avaliação parcial. Nas aulas 2 e 3, o aluno tinha a flexibilidade de tirar dúvidas sobre qualquer uma das atividades, bem como procurar qualquer um dos três tutores, a depender do horário que lhe fosse mais conveniente.

Avaliação

Foram atribuídas notas aos laboratórios de codificação e de exercícios, a fim de estimular sua realização. Todavia, as notas das avaliações parciais que encerravam os módulos tiveram maior peso sobre a nota final. Ademais, os módulos iniciais, que abordaram assuntos simples (variáveis, estrutura sequencial), tiveram menor peso em relação aos módulos finais, que abordaram assuntos complexos (estrutura de repetição, matrizes). Tal distinção objetivou motivar dois perfis de alunos: aqueles com dificuldade no início da disciplina, a fim de que recuperassem nota; e aqueles com facilidade no início da disciplina, a fim de que não se afastassem dos estudos.

3. Sistemas juízes online

Juízes online são sistemas de correção automática de códigos-fonte comumente empregados como ferramentas de apoio pedagógico em disciplinas de programação. Tais sistemas são capazes de receber códigos desenvolvidos pelos usuários como resposta a determinados exercícios de programação, informando imediatamente se o código submetido está correto ou não. Basicamente, essa análise de corretude é feita mediante execução do código e comparação da saída retornada com uma saída esperada.

Grande variedade de juízes online está disponível na literatura científica. Por exemplo, o sistema The Huxley [Paes et al. 2013] possui centenas de problemas de programação cadastrados, e permite que os alunos submetam soluções na forma de códigos-fonte desenvolvidos em diferentes linguagens de programação. Através de análise sintática do código submetido e de testes de aceitação, o aluno é imediatamente informado se seu código é uma solução válida para o problema que tentou solucionar.

Um exemplo mais recente de juiz online é a ferramenta *feeper* [Alves e Jaques, 2014], que além de corrigir automaticamente os códigos dos alunos, é capaz de emitir dicas personalizadas de como solucionar os erros presentes em tais códigos. Outro diferencial dessa ferramenta é que ela favorece a interação entre aluno e professor, e permite a esse último acompanhar a evolução de todos seus alunos.

Além dos sistemas propostos pela literatura, existem inúmeros exemplos de sistemas juízes online disponíveis na Web, normalmente abertos para o público em

geral. Dentre os mais conhecidos no Brasil está o sistema URI Online Judge (www.urionlinejudge.com.br), que foi criado para servir de apoio e complemento de estudos para estudantes de Engenharias e Ciência da Computação. O sistema URI oferece um ambiente de iteração entre alunos, facilitando a troca de experiências, além de possibilitar que alunos iniciantes obtenham auxílio de estudantes mais avançados. Outros juízes online disponíveis na Web são Code Chef (www.codechef.com), UVa (uva.onlinejudge.org), HackerRank (www.hackerrank.com) e run.codes (we.run.codes).

Outra ferramenta que vale destaque é o MOJO [Chaves et al, 2013], um módulo de integração entre juízes online e o Moodle desenvolvido para diminuir o trabalho de elaborar questões e de corrigir códigos dos alunos.

Além de fins pedagógicos, juízes online também podem ser usados para apoiar competições de programação, na avaliação dos códigos submetidos pelos participantes. Um exemplo é o BOCA (*BOCA Online Contest Administrator*), adotado nas Maratonas de Programação organizadas pela SBC (Sociedade Brasileira de Computação).

Na próxima seção, apresentamos o CodeBench, juiz online desenvolvido na UFAM com o objetivo de proporcionar aos professores melhor controle da classe durante a aplicação de avaliações presenciais, além de servir de ambiente para resolução de exercícios de programação como os sistemas aqui comentados.

4. O sistema CodeBench

O CodeBench é um juiz online desenvolvido na UFAM com o propósito de automatizar a correção dos exercícios de programação. Nele, os professores podem disponibilizar exercícios de programação para seus alunos, que por sua vez podem codificar soluções e submetê-las através da interface do sistema (Figura 2). Assim que o aluno submete uma solução para um dado exercício, o CodeBench informa se ela está correta ou não.

Para julgar a corretude de um código submetido por um aluno, o CodeBench segue dois passos principais:

1. *Análise sintática do código*, em que se verifica se o código submetido possui algum erro sintático, de acordo com a gramática da linguagem de programação adotada pelo professor da disciplina. Caso haja algum problema sintático com o código, o aluno é imediatamente informado sobre a natureza do problema, bem como a linha do código onde ele se encontra.
2. *Análise lógica do código*, em que se verifica se o código desenvolvido pelo aluno é capaz de solucionar corretamente o problema proposto pelo professor.

A análise lógica dos códigos é feita por meio de *casos de testes,* cadastrados pelo professor ao criar uma questão. Cada caso de teste possui dois valores: um *valor de entrada*, passado como entrada para o programa do aluno, e um *valor de saída*, que é a saída correta para o valor de entrada informado. Por exemplo, considere-se um exercício que solicite a elaboração de um programa que imprima o fatorial de um valor dado como entrada. Um caso de teste válido para corrigir esse problema seria {*valor de entrada*: 5*, valor de saída*: 120}. Nesse exemplo, para verificar a corretude de um código submetido, o CodeBench executará os seguintes passos: (i) executar o código e verificar se há erros de sintaxe; (ii) caso não haja erro de sintaxe, submeter o valor 5 como entrada para o programa; e (iii) verificar se o programa imprime o valor 120. Caso o programa imprima o valor 120 para a entrada 5, o CodeBench informa que a questão foi solucionada corretamente.

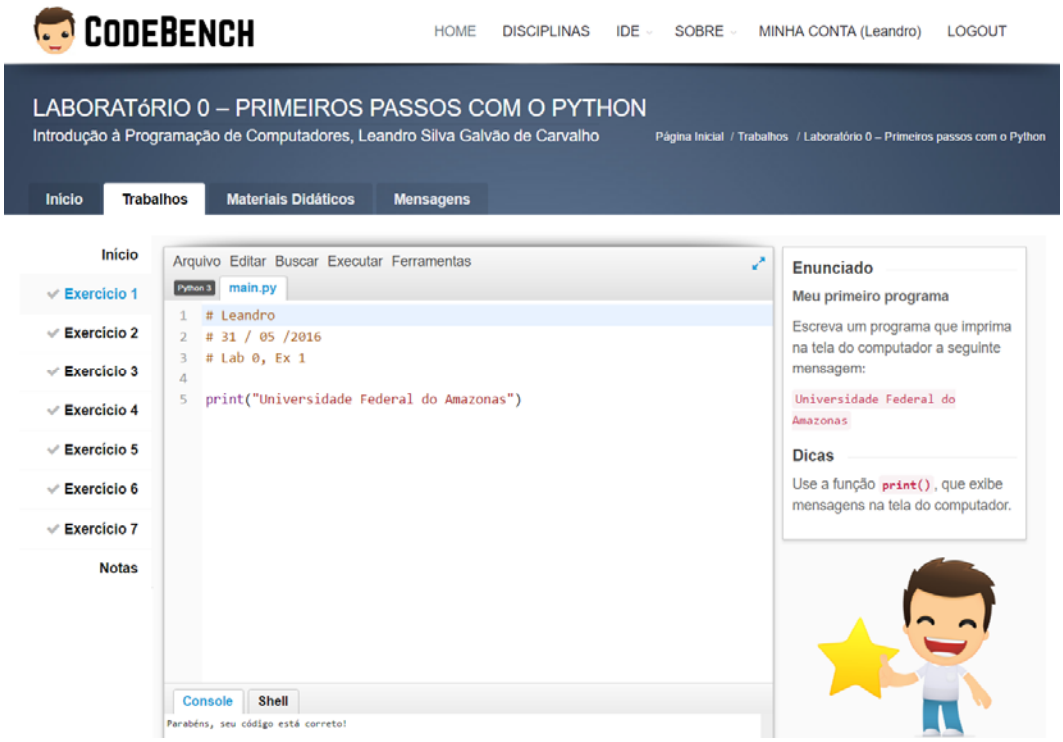


Figura 2 - Captura de tela de uma questão submetida com sucesso ao CodeBench.

Caso o sistema notifique que o código-solução não está correto, seja por erros de sintaxe ou de lógica, o aluno poderá rever seu código e ressubmetê-lo tantas vezes quantas desejar, até o prazo estipulado pelo professor ao cadastrar a questão. Essa política de ressubmissão de códigos incentiva o aluno a identificar e solucionar os erros por si próprio, ao contrário da abordagem tradicional, em que o aluno tem que esperar pela correção do professor, sem oportunidade de identificar os erros e corrigir seus códigos. Tal abordagem mimetiza o desenvolvimento de código na vida profissional, em que os programadores precisam identificar os erros de seus códigos e solucioná-los.

Além de poupar o tempo dos professores com a automatização da correção dos exercícios, o CodeBench também facilita a elaboração ou seleção de tais exercícios. Para isso, o sistema possui um banco contendo pouco mais de 900 exercícios de programação, que podem ser escolhidos pelos professores durante a criação de trabalhos. Sempre que um novo exercício é cadastrado no sistema, ele passa a integrar o banco e fica imediatamente disponível para todos os demais professores. Dessa forma, o CodeBench também pode ser visto como um ambiente colaborativo de elaboração de questões, onde todos os professores são favorecidos com o trabalho de todos os demais.

Os alunos podem desenvolver código no próprio IDE (Ambiente de Desenvolvimento Integrado) do CodeBench ou em outro recomendado pelo professor, e submete os códigos pela interface do CodeBench. Adicionalmente, o CodeBench dispõe de uma interface com a qual o professor pode conferir os códigos submetidos por seus alunos, verificando quais casos de teste passaram ou não com sucesso.

O registro de um aluno no CodeBench é por autoinscrição. Ou seja, qualquer pessoa pode se inscrever e se matricular em uma turma, pois o CodeBench não interage com o sistema de controle acadêmico da UFAM. Por um lado, isso confere liberdade para outras instituições de ensino utilizarem o CodeBench. Por outro, requer do professor um trabalho de cotejamento entre a planilha de turma do CodeBench e a da

turma registrada institucionalmente. Já o registro de professores e tutores só pode ser validado mediante aval do administrador do sistema.

5. Metodologia de Avaliação

A metodologia híbrida foi aplicada pela primeira vez durante o primeiro semestre letivo de 2015, em três das oito turmas de IPC compostas por calouros, aqui designadas por "grupo experimental". As outras cinco são designadas por "grupo de controle". Em ambos os grupos, mediram-se as taxas de aprovação, reprovação por nota e reprovação por frequência. Esses valores somam 100%, uma vez que não é permitido que os estudantes tranquem matrícula durante os dois primeiros períodos letivos.

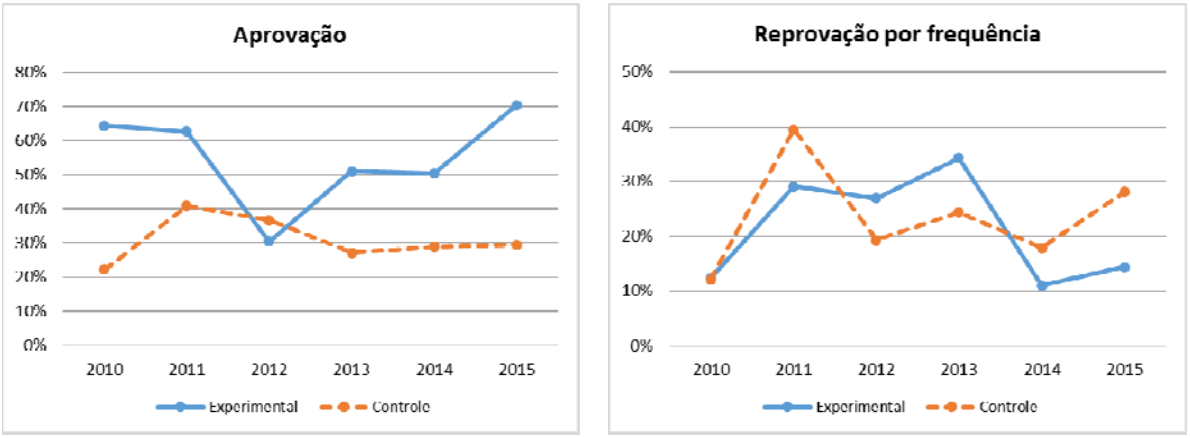
Ao final da disciplina, solicitou-se que os alunos das turmas experimentais avaliassem a metodologia híbrida por meio do preenchimento de um questionário. Este consistia de perguntas sobre o entendimento e a dificuldade de realizar as atividades propostas e a opinião acerca da metodologia utilizada. A partir dos dados levantados, realizou-se uma análise qualitativa sobre os comentários dos participantes nas perguntas abertas, por meio do Método de Explicitação do Discurso Subjacente (MEDS) [Nicolaci-da-Costa 2007]. Para tanto, as respostas foram sistematicamente comparadas em busca de recorrências. Os resultados são apresentados na Seção 6 a seguir.

6. Análise dos Resultados

Nesta seção, são apresentados os resultados quantitativos (aprovação, reprovação), bem como os qualitativos (percepção dos estudantes) a respeito da metodologia híbrida e da ferramenta de juiz online adotada no ensino de programação.

6.1 Taxas de Aprovação e Reprovação em IPC

A Figura 3(a) apresenta a evolução da taxa de aprovação em IPC de 2010 a 2015. A linha cheia representa as três turmas que participaram do grupo experimental, nas quais a metodologia híbrida foi aplicada somente no ano de 2015, e a linha tracejada representa as cinco turmas do grupo controle.



(a) (b)

Figura 3 - Evolução das taxas de aprovação (a) e de reprovação por frequência (b) na disciplina de IPC.

Com efeito, observa-se uma forte inclinação positiva no segmento entre 2014 e 2015 para o grupo experimental, em comparação com o segmento de inclinação quase

zero no referente ao grupo controle. Quantitativamente, este é um forte indício de que a metodologia híbrida aumentou a taxa de aprovação em IPC, como planejado.

Um traço marcante na Figura 3(a) é o comportamento atípico registrado no ano de 2012, que provavelmente se deve à paralização das aulas de IPC causada pela greve das universidades federais, fato que não ocorreu em 2015, apesar de uma nova greve.

A Figura 3(b) apresenta a evolução da taxa de reprovação por frequência em IPC de 2010 a 2015. Esse indicador pode mesclar em um só valor tanto a desistência do curso como a desistência da disciplina de IPC. Mesmo assim, nota-se que a inclinação do segmento entre 2014 e 2015 relativo ao grupo experimental é menor que a inclinação do segmento relativo ao grupo de controle. Este é um indício de que a metodologia híbrida manteve os alunos em aula mais do que a metodologia tradicional.

6.2 Questionário

A partir da análise qualitativa segundo o método MEDS [Nicolaci-da-Costa 2007] sobre as respostas dos estudantes sobre o questionário aplicado ao fim da disciplina no grupo experimental, foram identificadas as categorias apresentadas na Tabela 1.

Tabela 1 - Categorias que emergem da análise interparticipantes do MEDS.

#	Categoria	Ocorrências
1	*Feedback* rápido	14
2	Facilidade no uso de ferramentas	9
3	Qualidade do material de apoio	12
4	Disponibilidade dos sistemas e material de apoio	11
5	Quantidade de exemplos de teste	9
6	Correção binária de códigos	12
7	Flexibilidade de local para resolver as atividades	23
8	Estudo autônomo	4
9	Divisão do tempo com outras disciplinas	4

A seguir, são transcritas e comentadas as respostas consideradas mais marcantes em cada categoria. Tais respostas são destacadas em itálico e reproduzidas da mesma forma como foram escritas pelos estudantes.

O *feedback* rápido (categoria 1) proporcionado pelas ferramentas de apoio teve bastante relevância na motivação dos estudantes. Estes demonstraram gostar de saber rapidamente seu desempenho nas atividades, conforme ilustra a citação "*Foi super prático e bom, nos mostrava onde estava o erro, com a intenção de nos ajudar a verificar onde estava o erro!*".

Com respeito ao uso das ferramentas de suporte adotadas na disciplina (categoria 2), os alunos relataram que não tiveram dificuldades e aprovavam seu uso devido ao rápido *feedback* que recebiam. Um aluno afirmou: "*É bem simples de usar, quando os professores nos auxiliam. Inclusive, até melhor, já que temos acesso às notas no momento em que terminamos de realizar determinado exercício*". Os trechos sublinhados ressaltam a facilidade do uso e reitera a importância de um rápido *feedback* nas suas atividades. Outras citações também ilustram a facilidade no uso das ferramentas: "*Fazer os quizzes foi fácil pq o Colabweb é um sistema fácil de lidar"* e *"O ambiente era bem simples e facil de entender*".

Sobre a qualidade do material de apoio disponibilizado aos alunos (categoria 3), foi relatado que eram satisfatórios, de linguagem de fácil entendimento e que realmente ajudavam na realização das atividades, como diz esta citação: "*Havia tipo um "tutorial" explicando detalhadamente questões semelhates*". A disponibilidade tanto do material de apoio quanto das ferramentas de suporte na disciplina (categoria 4) foi outro ponto de destaque nos relatos dos alunos, que afirmavam: "*Porque foi possível acessar os arquivos de apoio no Colabweb*" e "*porque podia consultar a apostila e o Spyder*".

Com respeito às atividades de codificação, os alunos destacaram a insuficiência de exemplos para testes dos scripts propostos, como se observa nesta citação: "*Achei que faltavam mais exemplos para testar o programa*". Outros não relataram a mesma dificuldade, como se observa na citação: "*Por que eram bem explicativos e no material disponibilizado no Colabweb continha muitos exemplos*".

Um aspecto muito criticado foi a forma de correção dos scripts pelo CodeBench (categoria 6). Ele atribuía de forma binária nota máxima ou mínima para cada script enviado, de modo que alguns alunos relataram dificuldades para entender o que faziam de certo ou errado em cada atividade de codificação, como exemplifica a citação: "*Sim, pois com esse sistema ou você acerta tudo ou erra tudo, não existe meio termo. As vezes seu código possui uma pequena falha e você pode zerar pois ele mão considera o resto do seu programa na hora de computar a nota. Sem dizer que o programa compilado precisa estar muito igual ao do CodeBench para ser aceito*".

A interação com professores e tutores durante o andamento da disciplina (categoria 12) foi outro tópico recorrente nas respostas dos alunos ao questionário. Foi destacada a importância dessa interação para o entendimento e a boa execução das atividades propostas durante o curso, conforme ilustram as citações: "*alguns possuíam um nível de dificuldade maior, requisitando ajuda aos monitores e professor*", "*... mas consegui me sair bem perguntando do tutor o do professor*"; e "*As aulas e o contato com os tutores tornavam as resoluções mais simples*".

Por fim, as características inerentes à metodologia híbrida adotada na disciplina também foram percebidas pelos alunos, que destacaram a flexibilidade de local para fazerem as atividades (categoria 7), a possibilidade do estudo autônomo (categoria 8) e uma melhor divisão do tempo com as outras disciplinas cursadas no semestre (categoria 9). As citações a seguir exemplificam respectivamente cada categoria citada: "*Foi bom ter o CodeBench pois ele substituiu a aula presencial com o professor e podemos responder em qualquer lugar*"; "*nós conseguiríamos ver se conseguimos fazer sozinhos as atividades e se empenhar mais no curso*"; e "*Esse sistema traz bastante vantagens, dentre elas é a possibilidade de resolver questões em casa por conta própria. Isso facilita porque é possível conciliar IPC com as outras matérias do curso*".

7. Conclusões e Trabalhos Futuros

Cada estudante aprende a programar em um ritmo próprio, conforme sua base de conhecimentos prévios e motivação pessoal. O ensino híbrido flexibiliza o ritmo de aprendizagem de tal maneira a se adequar a tais individualidades. Dessa forma, juízes online, que provêm feedback automático sobre a corretude de programas submetidos, são ferramentas fundamentais para motivar o aprendizado e a autonomia.

Neste trabalho, apresentou-se o CodeBench, um juiz online desenvolvido como ferramenta de apoio a uma metodologia de ensino híbrido de programação. Verificou-se que essa abordagem proporcionou um aumento nos índices de aprovação, que saltaram

de 50% para 70% entre 2014 e 2015 nas turmas experimentais, ao passo que se manteve em torno de 30% nas turmas controle. A percepção dos estudantes foi, em geral, positiva e motivou a continuidade da iniciativa em 2016, com aprimoramentos.

Como trabalho em andamento, o juiz online foi completamente remodelado com base no feedback dos estudantes e dos professores. Na nova versão, o professor pode cadastrar três casos de teste, e a pontuação do aluno pode ser proporcional ao número de casos verificados com sucesso. Uma IDE foi incorporada ao CodeBench, reduzindo a confusão inicial do aprendiz em lidar com três ferramentas distintas. Além disso, o aluno agora pode opinar sobre a dificuldade percebida sobre o exercício, informação esta que poderá ser cruzada futuramente com o tempo de codificação, número de tentativas de submissão e desempenho da turma, a fim de auxiliar o professor a decidir sobre o balanceamento das atividades propostas.

Referências

Alves, F. P.; Jaques, P. (2014). Um Ambiente Virtual com *Feedback* Personalizado para Apoio a Disciplinas de Programação. In Anais do XXV Simpósio Brasileiro de Informática na Educação (Vol. 25, No. 1, p. 1078).

Chaves; J. O. M.; Castro; A. F.; Lima; R. W.; Lima; M. V. A.; Ferreira; K. H. (2013). Integrando Moodle e Juízes Online no Apoio a Atividades de Programação. In Anais do Simpósio Brasileiro de Informática na Educação (Vol. 24, No. 1, p. 244).

Giraffa, L.; Muller, L.; Moraes, M. C. (2015). Ensinando Programação apoiada por um ambiente virtual e exercícios associados a cotidiano dos alunos: compartilhando alternativas e lições aprendidas. In Anais dos Workshops do Congresso Brasileiro de Informática na Educação (Vol. 4, No. 1, p. 1330).

Hazzan, O.; Lapidot, T.; Ragonis, N. (2014). Guide to teaching computer science: an activity-based approach, 2ed. Springer.

Horn, M. B.; Staker, H. (2015). Blended: usando a inovação disruptiva para aprimorar a educação. Porto Alegre: Penso.

Ihantola, P.; Ahoniemi, T.; Karavirta, V.; Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In Proc. of the 10th Koli Calling International Conference on Computing Education Research (pp. 86-93).

Nicolaci-da-Costa, A. M. (2007) O Campo da Pesquisa Qualitativa e o Método da Explicitação do Discurso Subjacente (MEDS). In: Psicologia: Reflexão e Crítica. vol.20 no.1. ISSN: 0102-7972. RS, Porto Alegre.

Paes, R.B.; Malaquias, R.; Guimarães, M.; Almeida, H. (2013). Ferramenta para a Avaliação de Aprendizado de Alunos em Programação de Computadores. In Anais dos Workshops do Congresso Brasileiro de Informática na Educação (Vol. 2, No. 1).

Pelz, F. D.; Jesus, E. A.; Raabe, A. L. (2012). Um Mecanismo para Correção Automática de Exercícios Práticos de Programação Introdutória. In Anais do XXIII Simpósio Brasileiro de Informática na Educação (Vol. 23, No. 1).

Píccolo, H. L.; Sena, V. F.; Nogueira, K. B.; Silva, M. O.; Maia; Y. A. N. (2010). Ambiente Interativo e Adaptável para ensino de Programação. In Workshop sobre Educação em Computação, pp. 555–566.