Ensino de programação avançada incentivando a metacognição: uma experiência positiva usando Moodle+VPL

Mirtha L. F. Venero, Jesús P. Mena-Chalco

Centro de Matemática, Computação e Cognição, Universidade Federal do ABC
CEP: 09210-580 – Santo André – SP – Brasil

{mirtha.lina, jesus.mena}@ufabc.edu.br

Abstract. *The use of self-assessment tools is becoming one of the most popular resources for teaching programming. Nevertheless, there is a need to research the effectiveness of these tools and mechanisms to improve the learning strategies and performance of the students. This article reports an experience in the Moodle environment integrated with the automatic correction tool VPL for teaching advanced programming focusing on the development of metacognition. Although the methodology uses two different approaches, the results provide positive evidence of the effectiveness in using Moodle+VPL to support the mobilization of some metacognitive skills.*

Resumo. O uso de ferramentas de auto-avaliação está se tornando um dos recursos mais populares para o ensino de programação. Apesar disso, há necessidade de pesquisar a eficácia dessas ferramentas e os mecanismos para melhorar as estratégias de aprendizagem e o desempenho dos alunos. Este artigo relata uma experiência de uso de ambiente Moodle integrado com a ferramenta de correção automática VPL para o ensino de programação avançada com foco no desenvolvimento da metacognição. Apesar da metodologia usar duas abordagens diferentes, os resultados fornecem evidências positivas da efetividade do Moodle+VPL para apoiar a mobilização de algumas habilidades metacognitivas.

1. Introdução

O sucesso do ensino e aprendizagem em computação depende de diversos fatores, como o da construção das disciplinas, do uso de diversas ferramentas, assim como dos métodos de ensino e avaliação, que têm sido pesquisados de forma ativa ao longo dos anos (Luxton-Reilly et al. 2018). Em particular, as ferramentas para correção e geração de *feedback* de forma automática estão sendo cada vez mais usadas nas disciplinas introdutórias de programação (Ala-Mutka 2005, Ihantola et al. 2010, Keuning et al. 2018). Essas ferramentas também são apropriadas em matérias avançadas que requerem a solução de problemas usando algoritmos de maior complexidade (Enstrom et al. 2011). A correção manual desse tipo de atividades consome muito tempo, pois usualmente os programas têm um número grande de linhas de código que escondem erros sutis, às vezes difíceis de identificar corretamente até por monitores e docentes.

O uso de corretores automáticos diminui de forma significativa a carga de trabalho do professor e permite ao aluno obter um *feedback* rápido. No entanto, muitas vezes as submissões de soluções de programas são realizadas sem garantir uma correta

DOI: 10.5753/cbie.sbie.2019.279

interpretação do problema gerando diversas dificuldades metacognitivas (Prather et al. 2018). Por isso, para garantir o sucesso da aprendizagem, a avaliação automática deve estar inserida em ambientes de ensino que permitam estimular as habilidades metacognitivas como a reflexão, o monitoramento e a autorregulação (Metcalfe & Shimamura 1994, Bergin et al. 2005, Pimentel et al. 2005, Alhazbi et al. 2010, Kautzmann & Jaques 2016). O desenvolvimento dessas habilidades não acontece de forma natural (Flavell 1976, Shaft 1995, Mani & Mazumder 2013). Porém, poucos trabalhos descrevem o efeito de combinar o uso de ambientes de avaliação automática com estratégias metacognitivas (Loksa et al. 2016, Keuning et al. 2018, Prather et al. 2018, Rodriguez et al. 2018). Conforme apontado por Pettit e Prather (2017), há necessidade de preencher a lacuna de pesquisa sobre a eficácia dessas ferramentas para melhorar a aprendizagem dos alunos.

Este artigo relata a experiência na escolha e uso de um ambiente para o ensino da disciplina Algoritmos e Estruturas de Dados I na Universidade Federal do ABC (UFABC) que permitisse o desenvolvimento de estratégias de aprendizagem metacognitivas. A disciplina tem 48 horas de duração e a ementa inclui tópicos que cobrem noções básicas de complexidade de tempo, estruturas de dados lineares, árvores, busca e ordenação. A taxa de evasão e retenção na disciplina é maior de 35% devido a vários fatores. Primeiro, o projeto pedagógico da Universidade faz com que, mesmo que a maioria dos estudantes que cursam a matéria pretenda cursar Ciência da Computação, um número importante dos alunos tem como pretensão outros cursos específicos. Esses estudantes se matriculam na matéria com conhecimentos básicos de programação, porém sem terem cursado Programação Estruturada que é um dos pré-requisitos e introduz conceitos de programação na linguagem C. Para eles, o número de horas exige uma maior dedicação e trabalho independente.

A disciplina tem experimentado um aumento contínuo do número de estudantes sendo que no oferecimento de 2018 chegou a ser 200 com uma porcentagem de reprovação de 45% (incluindo as desistências). No ano atual 2019, a disciplina foi ofertada para mais de 300 alunos com o desafio adicional da ausência total de monitores. Por isso, a coordenação da disciplina decidiu discutir a metodologia e o plano de ensino a serem aplicados. No entanto, não houve consenso geral na escolha das ferramentas e somente dois professores aderiram à proposta de escolher um ambiente favorável ao desenvolvimento de estratégias metacognitivas. O artigo descreve a metodologia aplicada e os resultados da experiência e discute as formas de aperfeiçoar o método de ensino para melhorar a aprendizagem e o desempenho dos estudantes.

O artigo está organizado como a seguir. A seção 2 apresenta brevemente o referencial teórico que serviu de base para a escolha do ambiente e os métodos de ensino. A seção 3 descreve as metodologias adotadas, relata a experiência de uso da ferramenta de correção automática do Moodle+VPL e os resultados alcançados na disciplina. A seção 4 discute algumas reflexões sobre efetividade do Moodle+VPL para a mobilização das estratégias metacognitivas e seu efeito positivo para o aprendizado de programação avançada. A seção 5 apresenta as conclusões e os trabalhos futuros.

2. Referencial teórico para a escolha dos métodos e ambiente de ensino

O ponto de partida deste trabalho foi o artigo de Rodriguez et al. (2018) que pesquisaram as estratégias cognitivas e metacognitivas (propostas por Santos e Boruchovitch 2008) menos mobilizadas no processo aprendizagem de lógica de programação na modalidade semipresencial. Além disso, foram apresentados critérios e um *checklist* de recursos e ferramentas que podem auxiliar o professor a promover a inserção de cada critério na prática docente e apoiar o aluno nos estudos. Dentre esses critérios foram selecionados os seguintes como os dois mais importantes para serem desenvolvidos no contexto da disciplina Algoritmos e Estruturas de Dados I:

(1) planejamento das atividades de resolução de problemas (usando *checklist* de passos vs. resolução parcial)

(4) controle da ansiedade nas avaliações (usando avaliações curtas, *feedback* frequentes e simulados de provas).

2.1 Método de ensino

O método de ensino escolhido para mobilizar o critério **(1)** foi baseado no trabalho de Loksa et al. (2016) que propuseram quatro formas de intervenção dos professores para ajudar os estudantes a obter a solução de programação enquanto os incentivam a reconhecer, avaliar e refinar suas habilidades metacognitivas: *i) fornecer instruções explícitas sobre os objetivos e atividades envolvidas na resolução de problemas; ii) incentivar os alunos para descreverem os estados de solução de problemas; iii) fornecer meios físicos de representar os estágios de resolução de problemas para ajudar os alunos a monitorarem seu estado; iv) oferecer dicas contextuais que estimulem os alunos a reconhecerem as fases de resolução de problemas em que estão envolvidos*. Além disso, foram propostos seis estágios de resolução de problemas que podem ser revisitados de forma iterativa: 1) *(re) interpretar o problema; 2) procurar problemas análogos; 3) procurar soluções; 4) avaliar uma solução potencial; 5) implementar uma solução; 6) avaliar a solução implementada*.

Loksa et al. (2016) aplicaram a metodologia num acampamento de programação com o objetivo de ensinar conceitos, sintaxe e semântica de HTML, CSS e JavaScript com foco no *framework* React JavaScript. Os resultados apontaram que as quatro formas de estimular a metacognição e os seis estágios de resolução de problemas melhoraram a produtividade, promoveram a independência e aumentaram a autoeficácia dos jovens. Prather et al. (2018) combinaram essa metodologia com a estratégia de pensar em voz alta (PVA) para compreender as dificuldades metacognitivas que os programadores iniciantes enfrentam ao aprender a programar usando ferramentas de avaliação automáticas. O estudo identificou cinco dificuldades, três das quais têm a ver com insuficiente reflexão para construir um modelo conceitual correto do problema. No mais recente trabalho, Prather et al. (2019) realizaram um experimento sobre o efeito de fornecer dicas metacognitivas explícitas para ajudar na superação dessas dificuldades. Os resultados foram promissores apesar dos dados não se apresentarem conclusivos.

O método de ensino usado para mobilizar o critério **(4)** foi baseado na abordagem de Allen et al. (2018) que propõe usar a cada semana muitos pequenos programas (*many-small programs* - MSPs) em vez de um grande programa (*one-large*

program - OLP). Os autores concluíram que a abordagem MSP é menos intimidante para os estudantes que ganham confiança e habilidade começando pelos programas mais fáceis para depois trabalhar nos mais difíceis. Num trabalho posterior, eles pesquisaram se a abordagem MSP em cursos introdutórios de programação afeta o desempenho em disciplinas avançadas que usam a abordagem de OLP (Allen et al. 2019). A análise mostrou que o desempenho dos estudantes que fizeram o curso introdutório com MSPs foi um pouco melhor que os que usaram OLP.

2.2 Escolha do ambiente computacional

Tomando como base inicial o *checklist* de Rodriguez et al. (2018), foram selecionados as seguintes funcionalidades para a seleção da ferramenta: *Agenda e lembretes de conteúdo e atividades; Ferramentas de compilação/depuração; Ferramentas de testes/quiz/feedback automático; Fóruns, chat, redes sociais, grupos específicos.* Além disso, na revisão sistemática de literatura de Francisco et al. (2018) foi apresentado um conjunto de requisitos funcionais e não funcionais que devem ser atendidos por uma ferramenta para ensino de programação. Por isso, para a seleção do ambiente foram também considerados os seguintes requisitos funcionais: *Integração com Cursos, Monitoramento Desempenho dos Alunos, Diferentes Conteúdos/ Atividades, Geração de listas de exercícios, Feedback estático, dinâmico e personalizado, Detecção de Plágio, Usabilidade, Segurança, Documentação.*

Duas abordagens foram avaliadas para escolher um ambiente de código aberto com os requisitos acima. A primeira foi o uso de livros interativos que incorporam diversos recursos como texto, imagem e vídeo com edição, compilação e execução, visualização e avaliação de código. Nesta abordagem destacam-se plataformas de código aberto como OpenDSA (Fouh et al. 2013) e Runestone Interactive (Ericson et al. 2019) mas elas não incluem detecção de plágio. A segunda abordagem considerada foi o uso do Moodle (Modular Object-Oriented Dynamic Learning Environment), software livre sob a Licença Pública Geral GNU que pode ser usado em computadores, tablets ou celulares conectados à Internet (Cole & Foster, 2007).

Várias iniciativas têm integrado no Moodle os recursos para a avaliação automática de exercícios de programação. Dentre elas, destaca-se o Virtual Programming Lab (VPL) que é um plug-in do Moodle que permite a edição, execução e a correção de programas em diversas linguagens (e.g. C, C++, C#, Haskell, Java, Perl, PHP, Python e Ruby) e a detecção de plágio (Rodríguez-del-Pino et al. 2012). A execução do programa pode ser controlada limitando o tempo, tamanho da memória e do arquivo. O Moodle também tem sido integrado com juízes online como BOCA e URI (França et al. 2011, Chaves et al. 2013). No entanto, o VPL pode ser adaptado de forma mais simples para implementar funcionalidades adicionais pois a falta de documentação gera dificuldades para modificar o código fonte dos juízes (Francisco et al. 2018).

3. Desenvolvimento da disciplina usando Moodle+VPL

A disciplina foi inicialmente oferecida para 317 estudantes divididos em cinco turmas (duas no período matutino, uma no vespertino e duas no noturno) sob a responsabilidade de quatro docentes. Dois docentes aderiram à proposta de escolher a integração

Moodle+VPL para o desenvolvimento de estratégias metacognitivas. As turmas envolvidas (uma em cada período com 49, 47 e 82 estudantes) As outras turmas usaram como ambiente computacional Piazza e BOCA com 80 e 59 estudantes, resp. Os planos de ensino das turmas Moodle+VPL foram alinhados usando um cronograma e aulas teóricas similares. No entanto, nas aulas práticas e avaliações dessas três turmas foram usadas abordagens diferentes que serão explicadas a seguir.

O método de ensino escolhido para ser aplicado na turma vespertina (47 estudantes) foi baseado na abordagem de Loksa combinada com a estratégia PVA e OLP. O objetivo foi mobilizar principalmente a estratégias metacognitivas de planejamento e reflexão. Cada aula iniciou-se com a resolução conjunta entre professor e estudantes de um exercício de complexidade pequena sobre o conteúdo da aula teórica, aplicando os seis estágios de resolução de problemas. Em seguida, os alunos resolviam de forma independente um exercício similar usando a mesma estratégia com dicas de resolução dos docentes. Um exercício mais complexo era disponibilizado para desenvolvimento fora de sala de aula, mas com prazo de uma semana. Os exercícios deviam ser submetidos através do Moodle e avaliados automaticamente usando o VPL, com poucos casos de teste abertos e outros fechados construídos de forma manual e aleatória. O código VPL foi modificado para ser integrado com a ferramenta Valgrind[1] que permite detectar automaticamente erros no gerenciamento da memória. Nas outras duas turmas (131 estudantes), o foco para mobilizar a metacognição foi o controle da ansiedade e a autorregulação usando a abordagem MSP. Em cada aula prática os estudantes tentaram resolver de forma independente entre cinco e dez exercícios de pequena complexidade a serem submetidos e avaliados via Moodle+VPL, com todos os casos de teste abertos obtidos de forma manual. Em ambos casos, o prazo de todas as atividades no Moodle foi de uma semana.

Nas turmas Moodle+VPL também foram usados instrumentos de avaliação diferentes. Na turma vespertina foi usada uma prova com uma componente teórica escrita (duas perguntas) e uma pergunta prática para ser resolvida no computador e submetida via Moodle, porém sem avaliação automática. Nas turmas diurna e noturna as provas foram escritas seguindo a abordagem MSP com entre cinco e sete exercícios. Nessas duas turmas, conforme o checklist de Rodriguez et al. (2018), foram aplicados avaliações curtas frequentes e simulados das provas realizados sempre uma aula antes de cada prova. Além disso, nas três turmas foram usados desafios como bônus para melhorar a nota final. Na turma vespertina foi usado um desafio de programação que consistiu na resolução de um problema envolvendo os conhecimentos apreendidos ao longo do curso. Nas outras duas turmas foram usados dois desafios em forma de resumos escritos sobre conceitos teóricos complementares às aulas tratadas.

A Figura 1 ilustra os resultados obtidos na disciplina. O gráfico na parte superior mostra a distribuição de conceitos onde de “A” a “D” representam aprovação e os conceitos “F” e “O” reprovação por desempenho e frequência (consideradas desistências), resp. O gráfico na parte inferior mostra as porcentagens de aprovados e reprovados. A abordagem MSP obteve o melhor desempenho com uma porcentagem de aprovação de 81,54% e uma baixa evasão (6,92% de desistências). As turmas que

[1] http://www.valgrind.org, último acesso em julho de 2019.

usaram o Moodle+VPL tiveram porcentagens relativamente similares nos conceitos “A” e “D”; porém a maior reprovação por faltas foi na turma vespertina (25,53%). Nas turmas que não participaram da experiência a porcentagem de aprovação foi similar e menor de 50%. O fato da avaliação não ser unificada não permitiu obter resultados conclusivos sobre a melhor abordagem a ser aplicada na disciplina.

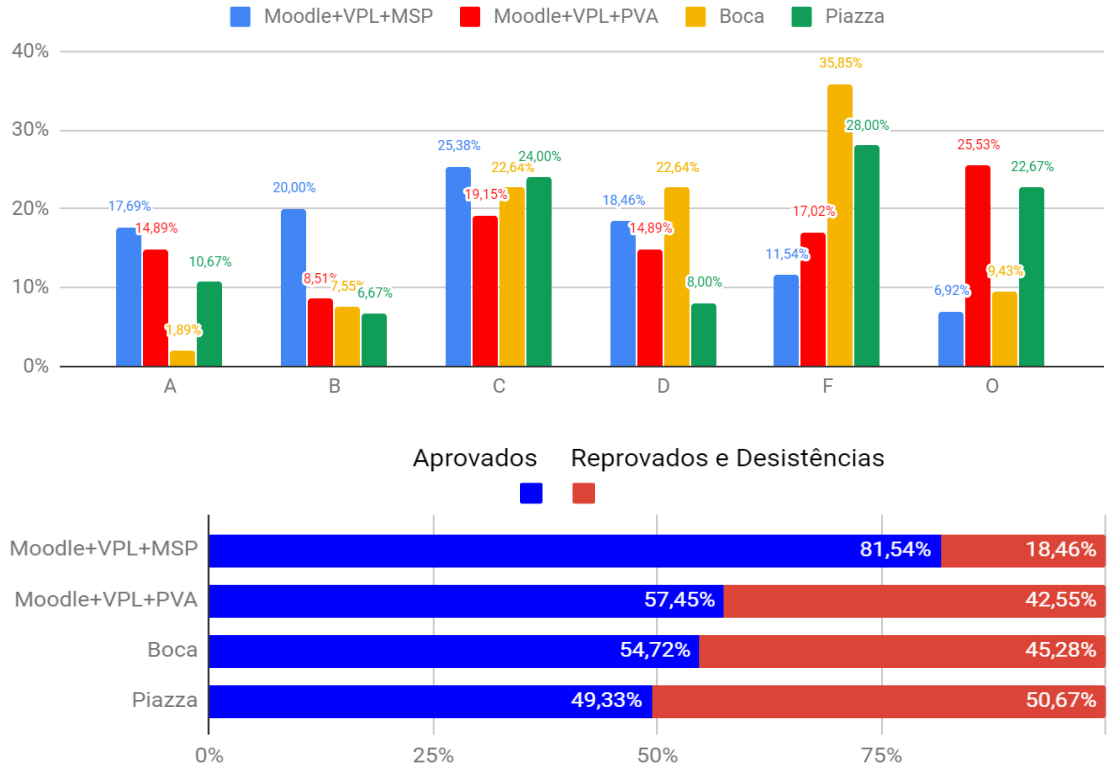

Figura 1. Gráficos com a distribuição de conceitos da disciplina em 2019.

4. Discussão

Os resultados da Figura 1 e as notas nas atividades práticas permitiram evidenciar que a abordagem MSP usando Moodle+VPL se mostrou eficaz em apoiar o desenvolvimento de estratégias de autorregulação para o controle da ansiedade nas avaliações. Para avaliar a efetividade da metodologia de Loksa combinada com a estratégia PVA na mobilização do planejamento e a reflexão foi realizada uma comparação usando o número médio de tentativas nas atividades. Para isso, foram comparadas as turmas MSP e PVA, escolhendo quatro exercícios da mesma complexidade ou similar. A Tabela 1 mostra que em média, o número de tentativas realizadas pelos estudantes da turma PVA foi menor que aqueles da turma diurna MSP; exceto no último exercício que foi levemente maior, apesar da complexidade ser maior pelo uso de listas ligadas em vez de vetores. Além disso, os casos de testes da turma MSP foram abertos enquanto na turma PVA o número foi maior e a maioria aleatórios e fechados. Na turma PVA, as submissões dos exercícios de maior complexidade (incluindo o desafio) também foram validadas de forma manual pelos professores.

Tabela 1: Número médio de tentativas para a submissão das atividades.

Turma	Árvore de Busca	Árvore AVL	Ordenação por Seleção	Ordenação por Inserção
MSP	8,47	6,58	5,55 (usando vetores)	9,55 (usando vetores)
PVA	5,14	5,6	4,8 (usando listas ligadas)	9,95 (usando listas ligadas)

Para analisar outros indicadores dos resultados da experiência, nas turmas Moodle+VPL foi aplicado um formulário de avaliação anônimo aos alunos inscritos na disciplina com perguntas sobre o curso pretendido, os materiais e ferramentas usados, o ambiente Moodle, as atividades práticas (exercícios, provas e desafios), tópicos de maior dificuldade, dentre outras. O preenchimento do formulário foi atividade voluntária realizada pelos estudantes após a divulgação preliminar dos conceitos finais. Nas turmas MSP, um total de 30 de estudantes (22,9%) responderam o formulário enquanto na turma PVA responderam 15 (31,9%). A Tabela 2 mostra algumas perguntas e as porcentagens da resposta em ambas turmas.

Tabela 2: Alguns resultados do formulário de avaliação da disciplina

Pergunta	Resposta	% MSP	% PVA
Curso Pretendido?	Computação	93,1	66,7
Trabalha?	Sim	30	60
O uso do Moodle é apropriado para a disciplina?	Sim	96,7	80
Usei os seguintes materiais para estudo	Exercícios no Moodle	96,7	100
As atividades práticas contribuíram para o meu aprendizado	Concordo	100	80
Aprendi mais com o desafio do que já sabia	Concordo	46,7	46,7

No formulário também foram coletados sugestões e comentários para melhorar o processo de ensino. Uma das respostas mais frequentes foi a disponibilizar mais casos de teste abertos. No entanto, de forma geral, disponibilizar todos os casos de teste faz com que as soluções sejam projetadas para atender somente esses casos sem considerar outras situações críticas ou critérios de eficiência. O teste às cegas contribui para reforçar os mecanismos de reflexão sobre a solução e autorregulação do aprendizado além melhorar as habilidades de programação e depuração. Os testes fechados também permitiram uma maior interação entre professores e estudantes para esclarecer dúvidas.

Por outro lado, é importante ressaltar que a estratégia MSP faz difícil a detecção de plágio mas acredita-se que os estudantes sentem menor necessidade de copiar dos colegas. Já na abordagem OLP foi detectado um caso de plágio, envolvendo estudantes com grandes dificuldades na matéria. Em geral, os estudantes com dificuldades rejeitam a correção automática (VPL) pois fazem um número grande de tentativas antes de obter sucesso em todos os casos de teste. Isso faz com que, conforme avança o curso, eles

desistam de concluir as atividades nas primeiras tentativas e acabam desistindo da disciplina. Como explicado na seção anterior, na turma vespertina que usou a abordagem OLP a porcentagem de desistência foi maior. No entanto, o horário das aulas (16-18h) e o fato de vários estudantes trabalharem ou fazerem estágios ou não terem como pretensão o curso de Computação podem ter causado um maior número de faltas.

Apesar de que a experiência não foi realizada em todas as turmas da disciplina, acredita-se que o estudo contribui para melhorar os resultados de ensino de forma global. Como mostra o gráfico da Figura 2, a porcentagem de reprovação, mesmo ainda sendo superior a 35%, é a menor dos últimos três anos correspondentes à reformulação da disciplina conforme o novo projeto pedagógico do curso. A experiência sugere que como método de ensino nas aulas práticas da disciplina podem ser combinadas as abordagens PVA e MSP com problemas simples e soluções parciais. Além de pequenas atividades avaliativas, para mobilizar o planejamento e a reflexão também são necessários exercícios independentes de maior complexidade que possam ser divididos em vários subproblemas a serem resolvidos usando os exercícios simples resolvidos nas aulas. O uso de desafios incentiva o estudante a um aprofundamento sobre tópicos ou conceitos mais sofisticados. A escolha de um ambiente com avaliação e feedback automático mostrou-se fundamental para o desenvolvimento da metacognição.

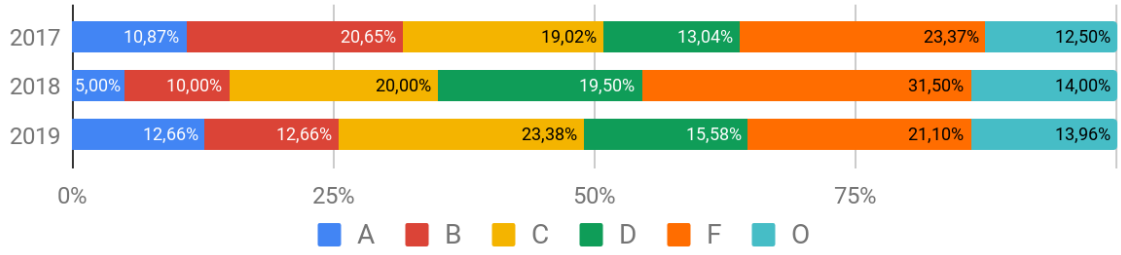

Figura 2. Distribuição de conceitos nos últimos três anos. Fonte: ProgGrad UFABC.

5. Conclusões e trabalhos futuros

Neste artigo foram apresentadas diferentes abordagens no ensino de conceitos de programação avançada. Os resultados apresentados fornecem evidências positivas da efetividade do uso do ambiente Moodle+VPL para mobilizar estratégias metacognitivas e como essa integração permite melhorar o aprendizado dos alunos. No entanto, o Moodle+VPL não fornece funcionalidades que ajudem a avaliar o desenvolvimento de habilidades de reflexão, monitoramento e autorregulação, como por exemplo relatórios, estatísticas, gráficos ou tabelas (por usuários e turmas) com, por exemplo, o número de envios realizados num período, o número de horas/dias na semana que os estudantes interagem com o sistema, os exercícios e casos de teste de maior dificuldade. Isso reforça a necessidade, já apontada por Prather et al. (2018), de definir mecanismos para implementar essas estratégias nos ambientes de aprendizagem e ferramentas de avaliação automática, bem como avaliar sua eficácia.

Como trabalho futuro pretende-se monitorar o desempenho dos estudantes que participaram na experiência em matérias subsequentes. O objetivo é obter outras evidências que permitam chegar num consenso para unificar os métodos de ensino e avaliação em todas as turmas da disciplina Algoritmos e Estruturas de Dados I e aplicar as melhores estratégias em outras disciplinas da UFABC que envolvem programação.

Agradecimentos

Os autores agradecem o Prof. Paulo Henrique Pisani pela colaboração no uso do VPL.

Referências

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

Alhazbi, S., Hassan, M. (2010). Fostering Self-Regulated learning in Introductory Computer Programming Course. APEC'2008 Education Reform Symposium in China. (2008). 21st Century Competencies.

Allen, J. M., Vahid, F., Downey, K., & Edgcomb, A. D. (2018). Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP). ASEE Annual Conference & Exposition.

Allen, J.M., Vahid, F., Edgcomb, A., Downey, K., & Miller, K. (2019) An Analysis of Using Many Small Programs in CS1. In Proc. 50th ACM SIGCSE, 585-591.

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In Proc. 1st ICER, 81-86.

Chaves, J. O. M., Castro, A. F., Lima, R. W., Lima, M. V. A. & Ferreira , K. H. A. (2013). MOJO: Uma Ferramenta de Auxílio à Elaboração, Submissão e Correção de Atividades em Disciplinas de Programação. In WEI - SBC.

Cole, J., & Foster, H. (2007). Using Moodle - teaching with the popular open source course management system (2. ed.). O'Reilly Community Press.

Dobre, I. (2015). Learning Management Systems for Higher Education - An Overview of Available Options for Higher Education Organizations, Procedia - Social and Behavioral Sciences, v. 180, p. 313-320.

Enström, E., Kreitz, G., Niemelä, F., Söderman, P., & Kann, V. (2011). Five years with kattis - Using an automated assessment system in teaching. Frontiers in Education.

Ericson, B., Cohen, J., & Miller, B. (2019) Using and Customizing Open-Source Runestone Ebooks for Computer Science Classes. In Proc ACM SIGCSE, 1240.

Flavell, J. H. (1976). Metacognitive aspects of problem solving. In L. B. Resnick (Ed.), The nature of intelligence, 231–235.

França, A., Soares, J., Gomes, D., & Barroso, G. C. (2011). Um sistema orientado a serviços para suporte a atividades de laboratório em disciplinas de técnicas de programação com integração ao ambiente Moodle. RENOTE, 9 (1).

Fouh, E., Breakiron, D., Elshehaly, M., Hall, T.S., Karavirta, V., & Shaffer, C.A. (2013). OpenDSA: using an active eTextbook to teach data structures and algorithms. In Proc. 44th ACM Technical Symposium on Computer Science Education, 734.

Francisco, R. E., Ambrósio, A. P. L., Pereira Júnior, C. X. & Fernandes, M. A.. (2018). Juiz online no ensino de CS1 - lições aprendidas e proposta de uma ferramenta. RBIE, 26 (3), 163-179.

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

Kautzmann, T., & Jaques, P. (2016). Training of the Metacognitive Skill of Knowledge Monitoring in Tutoring Systems. RBIE, 24(02), 22.

Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. ACM Transactions on Computing Education, 19(1), 3.

Loksa, D., Ko A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016) Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. In Proc. Conf. Human Factors in Computing Systems. ACM, 1449–1461.

Luxton-Reilly, A., Albluwi, I., Becker, B., Giannakos, M., Kumar, A., Ott, L.M., Paterson, J., Scott, M., Sheard, J. & Szabo, C. (2018) Introductory Programming: A Systematic Literature Review. In Proc. 23rd ACM ITiCSE , 55-106.

Mani M. & Mazumder Q. (2013) Incorporating metacognition into learning. In Proc. 44th ACM Technical Symposium on Computer Science Education. ACM, 53–58.

Metcalfe J. & Shimamura A. P. (1994) Metacognition: Knowing about knowing. MIT Press.

Pettit, R., & Prather, J. (2017). Automated assessment tools: Too many cooks, not enough collaboration. Journal of Computing Sciences in Colleges, 32(4), 113-121.

Pimentel E. P. P, Omar N., França V. F. (2005) Um Modelo para Incorporação de Automonitoramento da Aprendizagem em STI. RBIE, 13(1).

Prather, J., Pettit, R., McMurry, K., Peters, A., Homer, J. & Cohen, M. (2018). Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools. In Proc. ACM Conf. International Computing Education Research. 41-50.

Prather, J., Pettit, R., Becker, B.A., Denny, P., Loksa, D., Peters, A., Albrecht, Z. and Masci, K. (2019). First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts. In Proc. ACM SIGCSE , 531-537.

Rodriguez, C. L., Rocha, R. V., Goya D ; Venero, M. L. F.; Zampirolli, F. (2018). Critérios para inserção de estratégias cognitivas e metacognitivas no desenvolvimento de lógica de programação em ambientes virtuais de aprendizagem. Anais do SBIE.

Rodríguez-del-Pino, J. C., Royo, E. R., & Figueroa, Z. J. (2012). A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features. Proc. Int. Conf. e-Learning, e-Business, Enterprise Information System.

Santos, A. A. A., and Boruchovitch, E. (2008). Escala de estratégias de aprendizagem para Universitários, Manuscrito não publicado, Unicamp, Campinas, Brasil.

Shaft T. M. (1995) Helping programmers understand computer programs: the use of metacognition. ACM SIGMIS Database 26, 4 (1995), 25–46.