Avaliando ambientes para ensino de programação com suporte para o desenvolvimento da metacognição

Sabrina M. Rodrigues[1], Mirtha L.F. Venero[2], Carla L. Rodriguez[2], Denise H. Goya[2], Rafaela V. Rocha[2]

[1] Bacharelado em Ciência e Tecnologia, [2] Centro de Matemática, Computação e Cognição - Universidade Federal do ABC (UFABC) - Santo André, SP, Brasil

[1]sabrina.m@aluno.ufabc.edu.br, [2]{mirtha.lina, c.rodriguez, denise.goya, rafaela.rocha}@ufabc.edu.br

Abstract. *The growing number of students in programming courses imposes several challenges for teachers, including the choice of a proper teaching environment and resources. Given the difficulty of attending students in a personalized way, the teaching methods and tools should be chosen in a way that supports the development of metacognitive awareness. This article presents an evaluation of four open-source programming environments that allow them to be compared on the basis of criteria that support the mobilization of metacognitive learning strategies.*

Resumo. *O crescente número de estudantes nos cursos de programação impõe vários desafios para os docentes, dentre eles a escolha de um ambiente e recursos de ensino apropriados. Ante a dificuldade de atender os estudantes de forma personalizada, os métodos e ferramentas de ensino devem ser escolhidos de forma que apoiem o desenvolvimento da consciência metacognitiva. Este artigo apresenta uma avaliação de seis ambientes de livre acesso para o ensino de programação que permite compará-los com base em critérios que suportam a mobilização de estratégias de aprendizagem metacognitivas.*

1. Introdução

O processo de ensino e aprendizagem de programação impõe tanto para o professor quanto para o aluno grandes desafios envolvendo a taxonomia de Bloom [Bloom et al. 1956]. Dentre eles destacam-se ensinar e aprender a: interpretar, abstrair e representar situações reais usando habilidades matemáticas; compreender, analisar e resolver os problemas computacionais envolvidos nessas situações; e testar, avaliar, aprimorar as soluções propostas. Medeiros et al. (2019) apontam como os principais desafios para o professor a escolha de métodos e ferramentas apropriados, manter a motivação e engajamento dos estudantes e a escalabilidade e personalização do ensino. O número cada vez mais crescente de alunos nos cursos de programação tem gerado a necessidade de desenvolver uma ampla gama de ferramentas que auxiliem nesse processo. A revisão sistemática de Luxton-Reilly et al. (2018) fornece uma classificação das ferramentas usando diversas categorias, dentre elas os ambientes de edição e programação; bibliotecas, APIs e ferramentas de compilação e depuração; ferramentas de design, visualização, colaboração e jogos; e de avaliação e monitoramento de progresso.

A escolha de um ambiente de ensino e ferramentas apropriadas depende de vários fatores, por exemplo os objetivos e características do curso, o tipo de matéria

DOI: 10.5753/cbie.wcbie.2019.417

(introdutória ou avançada), a modalidade de ensino (presencial, semipresencial, a distância), o público-alvo, etc. Às vezes a dificuldade de escolher um ambiente se dá pela abundância de opções e falta de critérios e padrões de comparação para fazer a seleção formal e rigorosa dentre diferentes tipos [Brusilovsky et al. 2014, Francisco et al. 2018, Keuning, Jeuring e Heeren 2018]. Além disso, Brusilovsky et al. (2014) consideraram que outra dificuldade é a de customizar as ferramentas às suas necessidades, ressaltando a importância de incluir conteúdos de aprendizagem inteligente (SLC- *Smart Learning Content*).

As ferramentas SLC têm como características serem interativas oferecendo recursos de visualização, simulação, suporte à codificação e a resolução de problemas com avaliação automática e *feedback*. O *feedback* de tipo formativo deve incluir informações para modificar o pensamento ou o comportamento do estudante, com a finalidade de melhorar a aprendizagem [Shute 2008]. Ele constitui um dos recursos que permite desenvolver estratégias de aprendizagem metacognitivas como a autorregulação [Ott et al. 2016]. A metacognição consiste no conhecimento sobre o próprio conhecimento e a natureza das tarefas cognitivas [Flavell 1976]. A consciência metacognitiva permite ao estudante planejar, regular, monitorar e avaliar seu próprio conhecimento [Schoenfeld 1987, Derry e Hawkes 1993]. A importância de promover o desenvolvimento da consciência metacognitiva é cada vez maior, dado o aumento do número de estudantes e a diversidade de formação. A necessidade de avaliar como as ferramentas apoiam os processos metacognitivos tem sido apontada em diversos trabalhos [Hudesman et al. 2013, Keuning, Jeuring e Heeren 2018, Prather et al. 2018].

Este artigo apresenta uma avaliação de seis ambientes para o ensino de programação com base em critérios que permitem comparar suas funcionalidades no apoio à metacognição, ie. na forma de contribuir para aumentar a consciência dos alunos no conhecimento dos seus processos de aprendizagem. A pesquisa está motivada no trabalho de Rodriguez et al. (2018) que apresentou um conjunto de critérios e recursos para a inserção de estratégias de aprendizagem pouco usadas pelos estudantes de programação de uma disciplina semipresencial da Universidade Federal do ABC. No entanto, nenhum ambiente para o ensino de programação foi mencionado como apropriado segundo esses critérios.

O foco deste artigo está em ambientes de código aberto ou acesso livre, em particular aqueles que usam múltiplas linguagens, paradigmas e métodos. Diversos autores têm apontado a falta desses sistemas como uma das razões pelo constante desenvolvimento de novas ferramentas; por isso é necessário unificar esforços para o aprimoramento das existentes [Ihantola et al. 2010, Pettit e Prather 2017]. Além disso, numa revisão sistemática recente sobre ensino de programação no Brasil, Borges et al. (2018) perceberam que os estudos que propuseram uma ferramenta representaram uma quantidade significativa dentre os trabalhos investigados (36%), mas tiveram pior avaliação quando comparados com outros trabalhos e/ou soluções.

O artigo está organizado como a seguir. As seções 2 e 3 apresentam revisões da literatura que permitiram selecionar algumas ferramentas de código aberto e critérios de avaliação. A seção 4 inclui a comparação de seis ferramentas escolhidas que permite medir se há suporte para o desenvolvimento de habilidades metacognitivas. Por fim, a seção 5 apresenta considerações finais e trabalhos futuros.

2. Referencial teórico sobre ferramentas de ensino de programação com avaliação automática

Neste trabalho, foram realizadas duas revisões da literatura sobre artigos dos últimos dez anos, ambas de caráter narrativo ou exploratório, conforme Rother (2007). A primeira revisão teve como objetivo identificar ferramentas de código aberto com funcionalidades que contribuem para o desenvolvimento de habilidades metacognitivas no ensino de programação. O ponto de partida do processo de busca foram duas revisões sistemáticas recentes da literatura de programação no mundo [Luxton-Reilly et al. 2018] e no Brasil [Borges et al. 2018]. A primeira explorou a literatura de programação introdutória no período de 2003 até 2017, abordando os estudantes, o ensino, o currículo e a avaliação. A segunda apresentou um mapeamento dos trabalhos publicados de 2012 até 2016 sobre metodologias e ferramentas de ensino de programação em duas das principais plataformas de publicação de trabalhos em Informática em Educação do Brasil: o Portal de Publicações da CEIE e a revista Renote. A partir delas, foram pesquisadas outras referências e encontradas algumas ferramentas.

Além disso, foi realizada uma busca por título, palavras-chaves e resumo nas bases Periódicos Capes, *IEEEXplorer Digital Library*, *ACM Digital Library*, *Science Direct* e *Scopus*. Nelas foram usadas *strings* que podem ser resumidas no seguinte padrão e seu correspondente em português: (*environment* OR *tool* OR *system* OR *platform* OR *teach** OR *learn** OR *systematic review)* AND *(* OR *program** OR *CS1*) AND (*metacogniti** OR *planning* OR *self-regulat** OR *self-assessment* OR *automat** *assessment* OR *reflection* OR *monitoring* OR *feedback*). Após a busca, foram excluídos os artigos descrevendo ferramentas não disponíveis para uso livre na Internet, incompletas, não estáveis ou de código não aberto. Os ambientes contidos nesses trabalhos podem ser classificadas em três grandes grupos que serão explicados a seguir.

As **ferramentas de avaliação automática (AA)** têm um papel importante no desenvolvimento da autoavaliação e a autorregulação [Ott et al. 2016, Keuning et al. 2018]. Segundo Ihantola et al. (2010) as ferramentas de avaliação automática podem ser divididas em duas grandes categorias: para competições de programação (usualmente conhecidas como *juízes online*) e para *educação*. Entre os exemplos de juízes online usados no Brasil estão o BOCA [De Campos e Ferreira 2004] e o URI [Bez, Tonin e Rodegheri 2014]. Exemplos de ferramentas utilizadas para educação são TestMyCode [Vihavainen et al., 2013] e Jutge.org [Petit et al. 2018]. Na revisão sistemática sobre geração automática de *feedback* de Keuning, Jeuring & Heeren (2018) foi apresentada uma lista de 101 ferramentas, dentre as quais se destacam onze que incluem exercícios em múltiplas linguagens e paradigmas e que podem ser resolvidos usando múltiplas estratégias, e.g. o VPL [Rodríguez-del-Pino et al. 2012]. A lista contém apenas um sistema tutor inteligente no qual o *feedback* inclui explicações sobre metacognição.

A integração de **sistemas de gestão de aprendizagem** (*Learning Management Systems* - LMSs), como e.g. Moodle, Sakai e Canvas, com outras ferramentas usando *plug-ins* é uma das formas mais usadas de adequar os cursos às necessidades do ensino e aprendizado de programação [Rößling et al. 2008, Ihantola et al. 2010]. Segundo Dobre (2015), o Moodle é o líder no mercado de ambientes de gestão de aprendizagem de código aberto. A plataforma fornece uma infraestrutura flexível que tem permitido criar um extenso diretório de *plug-ins*. Dentre os mais populares para o ensino de

programação estão o VPL e CodeRunner [Lobb e Harlow 2016], além de juízes online como BOCA, Online Judge [Zhigang et al. 2012] e o SPOJ Brasil e URI [Chaves et al. 2013]. No *survey* sobre juízes online de Wasik et al. (2018) a integração de LMS com juízes é chamada de **plataforma de desenvolvimento**.

Os **livros eletrônicos interativos** constituem uma das abordagens SLC mais recentes e cada vez mais populares no ensino e aprendizagem da programação. Diferentemente dos livros eletrônicos (*ebooks*) tradicionais, além de texto, imagens, vídeos e *quizzes*, os *ebooks* podem conter elementos interativos que permitem ao usuário interagir e customizar simulações, visualizações, execuções de código e avaliações automáticas [Korhonen et al. 2013]. Além disso, os livros interativos podem fornecer informação acerca do número de acessos e a distribuição do tempo por estudante para cada recurso, conteúdo, módulo ou exercício o que possibilita monitorar a frequência de uso e, com isso, a motivação do estudante. Como ferramentas SLC, os livros interativos permitem uma aprendizagem invertida mais ativa, centrada no aluno, produzindo maior autonomia e engajamento [Pollari-Malmi et al. 2017]. Dentre as plataformas de desenvolvimento de livros interativos destacam-se a OpenDSA [Fouh et al. 2016] e Runestone Interactive [Miller e Ranum 2014].

3. Critérios de comparação de ambientes de ensino de programação

Com o objetivo identificar critérios usados para avaliar ou comparar as ferramentas no ensino e aprendizagem de programação, foi realizada uma segunda revisão da literatura exploratória guiada pela seguinte pergunta de pesquisa: **quais critérios são usados para avaliar ferramentas de ensino de programação?** Neste caso as strings de busca usadas seguiram o seguinte padrão (e seu correspondente em português): (*tool* OR *system* OR *platform* OR *teach** OR *learn** OR *program** OR *CS1* OR *comput**) AND (e*valuation* OR *criteri** OR *taxonomy* OR *classification* OR *requirement*). Dentre os artigos obtidos, foram excluídos os trabalhos que não incluem requisitos ou critérios funcionais ou não funcionais que permitam comparar ferramentas de ensino e aprendizagem de programação.

Segundo a revisão sistemática de Francisco et al. (2018) sobre juízes online, um ambiente de ensino e aprendizagem de programação ideal deve atender certos requisitos **funcionais** e **não funcionais**, além de permitir a configuração dos recursos de aprendizagem (aqui classificados como de **design**). Os requisitos **funcionais** apontados pelos autores foram: *integração com cursos, monitoramento de desempenho dos alunos, diferentes conteúdos/atividades, geração de listas de exercícios, nível de dificuldade dos exercícios, feedback estatístico e personalizado e detecção de plágio*. Por outro lado, Rodriguez et al. (2018) consideram que especificamente as seguintes funcionalidades são necessárias para mobilizar algumas habilidades de metacognição: *agenda e lembretes de conteúdo e atividades; ferramentas de compilação; ferramentas de testes/correção/feedback automático; fóruns, chat, redes sociais, grupos específicos*. O trabalho mais antigo [Coull e Duncan 2011] também aponta como critérios a inclusão de *ferramentas de compilação*, *integração com cursos* e *diferentes atividades*.

Dentre os requisitos **não funcionais,** Francisco et al. (2018) apontaram como importantes os seguintes: *usabilidade*, *integração, segurança, escalabilidade* e *disponibilidade*. Queirós e Leal (2012) consideram três níveis de *interoperabilidade* para de avaliação de exercícios de programação: nível 0 quando permite a configuração

de dados selecionando os já existentes ou adicionando novos; nível 1 quando suporta importar/exportar dados de/para outras ferramentas; e nível 2 quando o sistema também suporta a comunicação com outras ferramentas através de serviços web. Os autores analisaram a interoperabilidade de vários ambientes considerando os exercícios, os usuários e os resultados das avaliações. Korhonen et al. (2013) propuseram um conjunto de requisitos de **design** para livros interativos, dentre os quais ressaltamos o suporte para a *criação e customização de conteúdo interativo*.

Neste artigo, devido à limitação de espaço, foram considerados dez requisitos, mostrados na Tabela 1. A primeira coluna da tabela associa cada requisito funcional e de design com uma das três habilidades metacognitivas envolvidas na autorregulação segundo Schoenfeld (1987): avaliar se você entende o conteúdo/problema (reflexão); planejar a estratégia de estudo/solução (planejamento); monitorar o andamento do processo e avaliar a solução (monitoramento). O planejamento determina quais objetivos secundários devem ser atingidos e em que ordem, enquanto o monitoramento é a habilidade de realizar autoverificações no processo [Derry e Hawkes, 1993]. Dessa forma, por exemplo, as agendas e lembretes favorecem o planejamento enquanto os fóruns, chat e grupos estimulam a reflexão. Apesar disso, um critério pode dar suporte a várias habilidades: por exemplo, o *feedback* automático favorece o monitoramento do processo de resolução, como também a reflexão sobre a compreensão do problema.

Tabela 1: Critérios de comparação de ambientes de programação com suporte a metacognição.

Habilidade	Critérios	Nível 0	Nível 1	Nível 2
Reflexão	**Diferentes conteúdos e atividades**	Um único tipo	Vários tipos separados	Vários tipos SLC combinados
Reflexão	**Customização do SLC**	Manual	Centrado no Instrutor	Híbrido e Personalizado
Reflexão	**Fóruns, chat, grupos específicos**	Manual	Centrado no Instrutor	Híbrido e Personalizado
Reflexão	**Detecção de plágio**	Manual	Centrado no Instrutor	Híbrido e Personalizado
Planeja-mento	**Agenda e lembretes de atividades**	Manual	Centrado no Instrutor	Híbrido e Personalizado
Monitora-mento	**Monitoramento de desempenho**	Somativo	Formativo	Personalizado
Monitora-mento	***Feedback* automático**	Somativo	Formativo	Personalizado
	Interoperabilidade	Manual	Importar/Exportar	Uso repositórios e serviços
	Integração	Manual	Unidirecional	Bidirecional
	Usabilidade	Uma LP	Algumas LPs	Muitas LPs

Seguindo a proposta de Queirós e Leal (2012), para cada critério foram considerados três níveis de suporte à habilidade metacognitiva: 0, 1 e 2. No entanto, o significado dos níveis varia dependendo do critério conforme apresentado de forma resumida na Tabela 1. Para requisitos funcionais, o nível 0 indica que a ferramenta não fornece suporte para o critério ou permite o desenvolvimento da metacognição de forma

simples ou somativa, isto é em forma de notas, número de falhas/sucesso, medidas estatísticas ou até índices metacognitivos como os propostos por Tobias e Everson (2002). No nível 1, o critério suporta as estratégias formativas, isto é fornecendo gráficos da evolução e/ou informações para melhorar o ensino e a aprendizagem com dicas tanto de conteúdo e atividades quanto do processo cognitivo e metacognitivo. No nível 2, o suporte para essas habilidades metacognitivas é personalizado baseado no histórico dos estudantes e/ou exemplos de erros e sucessos relacionados. Por outro lado, para alguns critérios o suporte pode ser manual, quando não está diretamente disponível ou devem ser usadas outras ferramentas (nível 0), centrado no instrutor (nível 1) quando a configuração/ativação/customização da funcionalidade que mobiliza a habilidade metacognitiva é realizada pelo docente ou híbrido (nível 2) se além disso permite a participação do estudante. Os níveis também diferem no quesito *Diferentes conteúdos e atividades* no qual foi considerado como nível 0 quando a ferramenta usa um único tipo de conteúdo ou atividade (por exemplo, no caso de juízes online que somente incluem listas de exercícios). Os outros níveis indicam uma variedade de conteúdos (textos, vídeos, imagens, animações) e atividades (testes, *quizzes*, exercícios de programação) porém separados no nível 1 ou combinados num único objeto de aprendizagem com SLC (e.g. notebooks e ebook) no nível 2.

Os requisitos não funcionais foram incluídos pois permitem aos professores usar os mesmos ambientes em diversos contextos e disciplinas, contribuindo para aumentar o senso de autoeficácia e motivação dos estudantes. Além da interoperabilidade de Queirós e Leal (2012), a integração com outras ferramentas foi incluída como forma de promover a abordagem SLC. Nesse requisito os níveis 1 e 2 representam a forma de integração, ie. se a ferramenta permite integrar outras ferramentas ou ser integrada em outros ambientes (nível 1) e ambas opções (nível 2). A usabilidade apresentada na tabela diz respeito ao número de linguagens de programação (LP) que o ambiente suporta: uma única linguagem (nível 0); algumas linguagens populares, como C, Java e Python (nível 1); e uma ampla variedade de linguagens (nível 2) [Wasik et al. 2018].

4. Avaliando ambientes para o ensino de programação com metacognição

Para avaliar e comparar diferentes ferramentas de ensino de programação e ao mesmo tempo permitir a uma comparação entre elas, foram selecionadas algumas ferramentas e critérios com base nas revisões da literatura das seções 2 e 3. As ferramentas escolhidas foram as seguintes. Como ambiente de AA para educação (AAE) foi escolhido o Jutge.org por ser um sistema web que integra características de um juiz online com LMS, além de técnicas modernas de verificação, métodos formais, análise de código estatístico e mineração de dados [Petit et al. 2018]. Ele tem sido usado em cursos de programação introdutória, mas também de estruturas de dados, inteligência artificial e projeto de circuitos. Também foi selecionado TestMyCode (TMC) que permite baixar, testar localmente e submeter exercícios de programação diretamente desde um IDE (*Integrated Development Environment*). Como juízes online foram selecionados o BOCA por ser usado nas competições de programação promovidas pela Sociedade Brasileira de Computação e o URI Academic também usado para ensino em instituições brasileiras. A plataforma Moodle (integrada com o VPL) foi escolhida pelo amplo uso no Brasil sendo, conforme os dados oficiais, o quarto país com o maior número de sites registrados. Como plataforma de *ebooks* foi selecionada Runestone

Academy, em lugar de OpenDSA, pelo uso em universidades brasileiras e o acesso aos livros sem necessidade de registro.

A Tabela 2 apresenta, para cada ambiente escolhido, uma avaliação conforme os critérios e níveis definidos na seção anterior, em que níveis mais altos representam maior apoio. Como mostra a tabela, a integração do Moodle com VPL fornece um maior suporte (65%) ao desenvolvimento de estratégias metacognitivas. A inclusão dos recursos de gerenciamento de cursos e avaliação automática permitem apoiar (mesmo que não de forma completa e personalizada) a reflexão, o planejamento e monitoramento. A plataforma também tem a maior avaliação entre os requisitos não funcionais. A plataforma de *ebooks* é que permite um maior nível de reflexão ao permitir combinar e customizar diferentes conteúdos e atividades. Porém tem poucas funcionalidades para a comunicação entre instrutores e estudantes, interoperabilidade e integração com outras ferramentas. Dentre os AAE e os juízes online, o Jutge.org e o URI Academic tiveram a mesma avaliação apesar de terem características diferentes. O TMC e o BOCA são os ambientes com menor suporte para a metacognição.

Tabela 2: Comparação de ambientes baseada no apoio a metacognição.

Critérios	TMC	Jutge.org	URI Academic	BOCA	Moodle+VPL	Runestone
Diferentes conteúdos e atividades	0	0	0	0	1	2
Customização de SLC	0	0	0	0	0	2
Fóruns, chat, grupos específicos	0	0	2	1	2	0
Detecção de plágio	0	0	1	0	1	0
Agenda e lembretes de atividades	0	1	1	0	1	1
Monitoramento de desempenho	0	1	0	0	1	1
***Feedback* automático**	1	1	0	0	1	1
Interoperabilidade	2	2	0	1	2	0
Integração	1	0	1	1	2	0
Usabilidade	1	2	2	1	2	1
Total	**5**	**7**	**7**	**4**	**13**	**8**

5. Considerações finais e trabalhos futuros

Neste artigo foram avaliados seis ambientes usados para o ensino de programação no Brasil, estabelecendo uma comparação baseada em critérios que apoiam o desenvolvimento de estratégias de aprendizagem metacognitivas. O resultado da avaliação aponta que a escolha de um LMS com ferramentas de AA fornece um ambiente apropriado para a mobilização da metacognição. Para o professor o uso dessa abordagem possibilita a criação rápida dos cursos combinando materiais previamente

elaborados com atividades de correção e *feedback* automáticos. A criação de *ebooks* interativos é outra abordagem promissora, porém requer maior esforço de preparação, pois os recursos de simulação e avaliação precisam ser organizados e combinados de forma adequada com elementos tradicionais como textos, imagens e vídeos. No entanto, as plataformas de livros interativos constituem uma das ferramentas SLC que possibilitam maior motivação e engajamento. A possibilidade de integração das duas abordagens já é realidade, pois, por exemplo, os *ebooks* de OpenDSA podem ser configurados para funcionar dentro do Canvas, Moodle ou Blackboard.

Apesar de existirem diversas revisões sistemáticas sobre ferramentas de avaliação automática e geração de feedback, nenhuma permite comparar de forma quantitativa diferentes ambientes considerando seu apoio ao desenvolvimento da consciência metacognitiva com base em diversos critérios. Por isso, este trabalho pode servir de guia para os docentes avaliarem e escolherem os ambientes a serem usados no ensino de programação. Em particular, uma versão preliminar da comparação apresentada na seção 4 foi usada para decidir o ambiente computacional na oferta da disciplina Algoritmos e Estruturas de Dados I da UFABC [Venero e Chalco 2019].

A comparação mostrou que nas três classes de ferramentas consideradas ainda há necessidade de pesquisas que forneçam evidências da influência e eficácia desses ambientes nos processos metacognitivos. Para isso são necessárias funcionalidades que permitam a interação formativa e personalizada da ferramenta com o estudante bem como métricas para avaliar o desenvolvimento das habilidades metacognitivas. Como trabalho futuro, pretendemos realizar um mapeamento das ferramentas que têm sido usadas com efetividade para mobilizar habilidades metacognitivas. Com isso, pretendemos estender a comparação a outros critérios, níveis e ambientes de ensino (e.g. os sistemas tutores inteligentes) além de elencar as principais lacunas de pesquisa.

Referências

Bez, J. L., Tonin, N. A. & Rodegheri, P. R. (2014). URI Online Judge Academic: A tool for algorithms and programming classes. Int. Conf. on Computer Science & Education. 149-152.

Bloom, B., Furst, E., Hill, W. & Krathwohl, D.R. (1956). Taxonomy of Educational Objectives: Handbook I, The Cognitive Domain, Addison-Wesley.

Borges, R. P., Oliveira, P. R. F., Lima , R.G.R. & Lima, R.W. (2018). A Systematic Review of Literature on Methodologies, Practices, and Tools for Programming Teaching, In IEEE Latin America Transactions, vol. 16(5), 1468-1475.

Brusilovsky, P. et al. (2014). Increasing adoption of smart learning content for computer science education. In Proc. Working Group Reports of the ACM ITiCSE . 31–57.

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.

Coull, N. J. & Duncan, I.M.M. (2011). Emergent requirements for supporting introductory programming. Innovations in Teaching and Learning in Information and Computer Sciences 10(1), 78–85.

De Campos, C. P. & Ferreira, C. E. (2004). "BOCA: Um Sistema de Apoio para Competições de Programação.Workshop de Educação em Computação, Anais do Congresso da SBC.

Derry, S.J. & Hawkes, L.W. (1993). Local cognitive model of problem-solving behavior: An application of Fuzzy Theory. Computers as Cognitive Tools. Lawrence Erlbaum Associates.

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.

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

Fouh, E., Hamouda, S., Farghally, M.F., & Shaffer, C.A. (2016) Automating Learner Feedback in an eTextbook for Data Structures and Algorithms Courses. In Challenges in ICT Education: Formative Assessment, Learning Data Analytics and Gamification, Elsevier, 135-165.

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.

Hudesman, J., Crosby, S., Flugman, B., Issac, S., Everson, H. & Clay, D. B. (2013). Using Formative Assessment and Metacognition to Improve Student Achievement. Journal of Developmental Education, 37(1), 2-4,6-8,10,12-13.

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

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.

Korhonen, A., Naps, T. L., Boisvert, C. R., Crescenzi, P., Karavirta, V. , Mannila, L., Miller, B. N. , Morrison, B. B., Rodger, S. H., Ross, R. J., & Shaffer, C. A. (2013). Requirements and design strategies for open source interactive computer science eBooks. In Working Group Reports, ITiCSE, 53-72. ACM.

Lobb, R. & Harlow, J. (2016). Coderunner: a tool for assessing computer programming skills. ACM Inroads 7(1), 47-51.

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. ACM ITiCSE, 55-106.

Medeiros, R. P., Ramalho , G. L,. & Falcão, T. P. (2019). A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. IEEE Transactions on Education, 62(2),77-90.

Miller B. & Ranum D. (2014). Runestone interactive: tools for creating interactive course materials. In Pro. 1st ACM Conf on Learning @ scale conference, 213-214.

Ott, C., Robins, A. & Shephard, K. (2016). Translating Principles of Effective Feedback for Students into the CS1 Context. ACM Transactions Computing Education, 16(1).

Petit J. et al. (2018). Jutge.org: Characteristics and Experiences. In IEEE Transactions on Learning Technologies, 11(3), 321-333.

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

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

Pollari-Malmi, K., Guerra, J., Brusilovsky, P., Malmi, L. & Sirkiä, T. (2017). On the value of using an interactive electronic textbook in an introductory programming course. In Proc. Koli Calling Int. Conf. Computing Education Research, 168-172.

Queirós, R. & Leal, J. P. (2012) Programming Exercises Evaluation Systems - An Interoperability Survey, CSEDU, 1, pp. 83–90.

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. Int. Conf. e-Learning, e-Business, Enterprise Information System.

Rother, E. T. (2007). Revisão sistemática X revisão narrativa. Acta Paulista de Enfermagem, 20(2), v-vi.

Rößling, G., Joy, M., Moreno, A., Radenski, A., Malmi, L., Kerren, A., Naps, T., Ross, R. J., Clancy, M., Korhonen, A., Oechsle, R. & Iturbide, J. A. V. (2008). Enhancing learning management systems to better support computer science education. SIGCSE Bull., 40(4), 142–166.

Schoenfeld, A.H. (1987). What's all the fuss about metacognition ? In Schoenfeld, A.H. (ed.), Cognitive Science and Mathematics Education, chapter 8, 189-215.

Shute, V. J. (2008). Focus on Formative Feedback. Review of Educational Research, 78(1), 153-189.

Venero, M. L. F. & Chalco, J. M. (2019). Ensino de programação avançada incentivando a metacognição: uma experiência positiva usando Moodle+VPL. Anais do SBIE.

Vihavainen, A., Vikberg, T., Luukkainen, M., & Pärtel, M. (2013). Scaffolding students' learning using test my code. In Innovation and Technology in Computer Science Education. 117–122

Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2018). A survey on online judge systems and their applications. ACM Computing Surveys, 51(1), 3.

Zhigang, S., Xiaohong, S., Ning, Z., & Yanyu, C. 2012. Moodle Plugins for Highly Efficient Programming Courses. In 1st Moodle Research Conference. 157–163.