Brazilian Journal of Development

# O Uso de Tutores de Programação Inteligentes na Produção de Feedback para Estudantes em Tarefas de Programação: Uma Revisão Sistemática da Literatura

# The Use of Intelligent Programming Tutors in Producing Feedback for Students in Programming Tasks: A Systematic Literature Review

DOI:10.34117/bjdv6n5-413

Recebimento dos originais: 20/04/2020
Aceitação para publicação: 20/05/2020

**Remyson Rodrigues Costa**
Bacharel em Tecnologia da Informação pela Universidade Federal Rural do Semi-Árido
Endereço: Departamento de Engenharias e Tecnologia. Universidade Federal Rural do Semi-Árido (UFERSA), Rodovia BR-226, KM 405, s/n - São Geraldo, Pau dos Ferros – RN, Brasil, 59900-000.
E-mail: remysonrodrigues@gmail.com

**Reudismam Rolim de Sousa**
Doutor em Ciência da Computação pela Universidade Federal de Campina Grande (UFCG)
Endereço: Departamento de Engenharias e Tecnologia. Universidade Federal Rural do Semi-Árido (UFERSA), Rodovia BR-226, KM 405, s/n - São Geraldo, Pau dos Ferros – RN, Brasil, 59900-000.
E-mail: reudismam.sousa@ufersa.edu.br

**RESUMO**
Estudantes encontram muitas dificuldades em componentes curriculares de introdução à programação. Muitas delas relacionadas à aprendizagem aos conceitos abstratos e lógicos na programação, além de fatores externos e internos ao aluno. Isso vem implicando na alta média de reprovação e evasão nesses cursos. Para tentar reparar esse problema, as universidades adotam o uso de Sistemas de Tutoria Inteligentes (STIs) para aprendizagem de programação, que podem desenvolver uma interação mais próxima dos estudantes com o aprendizado de programação. Neste trabalho, propõe-se uma revisão sistemática de como os tutores de programação auxiliam na produção de feedback para estudantes em tarefas de programação. Como resultado, foram identificados 36 trabalhos, em que elencou-se diferentes características, tais como as abordagens utilizadas pelos tutores de programação.

**Palavras-chave:** Algoritmos, Educação, Computação, Feedback, Sistemas de Tutoria Inteligentes

Brazilian Journal of Development

**ABSTRACT**
Students from the computing area find difficulties in Introduction to Programming classes, regarding on how to learning abstract and logical concepts in programming, besides their external and internal factors. This has led to a high level of fails and dropouts in these courses. To try to fix this problem, universities have been adopting Intelligent Tutoring Systems to teach programming, which can lead to a closer interaction of students with the task of programming learning. In this work, we propose a systematic review of how programming tutors assist in producing feedback for students in programming tasks. As a result, 36 works were identified, in which different characteristics were listed, such as the approaches used by the programming tutors.

**keywords:** Algorithms, Education, Computing, Feedback, Intelligent Tutor Systems

## 1 INTRODUÇÃO

Disciplinas introdutórias de programação estão presentes no currículo de vários cursos na área de engenharias, tecnologias, computação e em alguns países fazem parte também do ensino básico [Lazar et al. 2018]. Por isso, aprender a programar é uma tarefa essencial para os estudantes, sendo uma atividade que requer uma boa abordagem teórica e prática, em busca de desenvolver conhecimentos técnicos em programação e seus conceitos [Gross and Pinkwart 2015].

No entanto, aprender a programar nem sempre é uma tarefa simples, principalmente, para os alunos que possuem um primeiro contato com uma linguagem de programação [Gross and Pinkwart 2015]. Além da dificuldade de aprender a lógica da programação, pode-se considerar outros fatores externos e internos ao aluno, por exemplo, o suporte oferecido pela instituição, a metodologia adotada pelo professor e a vida social do aluno [Costa 2013].

Embora a programação seja uma área em grande evidência aos dias atuais, a literatura mostra que a aprendizagem de programação é um tema carente, quando relacionado à pesquisa com ênfase em técnicas, teorias e modelos de aprendizagem [Costa 2013].Todas essas implicações resultam na contribuição para a elevação no grau de reprovação e evasões em disciplinas da área [Giraffa and da Costa Mora 2013]. Nesse sentido, a programação se torna uma tarefa desafiadora para muitos estudantes, tanto pela dificuldade de desenvolver a lógica para resolver o problema quanto pelo desconhecimento da própria sintaxe da linguagem de programação. Para ajudar os estudantes a obterem um melhor desempenho nessa disciplina, muitas iniciativas foram criadas com o objetivo de facilitar o aprendizado de programação, fazendo com que esse conhecimento esteja disponível a todos [Lazar et al. 2018].

Uma prática comum nessas iniciativas de apoio aos estudantes é o uso de ferramentas que podem desenvolver uma interação mais estimulante dos estudantes com o aprendizado de programação. Segundo Thinakaran e Ali (2015), educadores apontam que a melhor forma de ensinar programação é por meio da interação individual entre professor/educador e aluno. No entanto, essa interação individual é muitas vezes dificultada pela grande quantidade de alunos que precisam receber *feedback* (dica ou correção) sobre o seu desempenho, o que torna essa interação individual muitas vezes inviável [Thinakaran and Ali 2015].

Para suprir essa demanda por um ensino de programação mais próximo dos alunos, os Tutores de Programação vêm ganhando importância. O objetivo da tutoria de programação é maximizar o processo para introdução e melhoria no ciclo de aprendizagem da programação [Chrysafiadi and Virvou 2014]. Tutores de Programação criam um processo de aprendizagem, para resolver exercícios e problemas de programação, com orientações de *feedback*, similar a um processo de tutoria individual, entre o aluno e o professor [Thinakaran and Ali 2015]. Dessa forma, o tutor, como um *proxy* (substituto temporário) para o professor, desempenha grande importância. Notadamente, o *feedback* produzido durante o processo de aprendizagem permite que os alunos identifiquem os seus defeitos e sejam capazes de aprender com os erros para resolver problemas mais rapidamente, evitando também que os mesmos erros sejam cometidos no futuro [Altuna Castillo and Guibert Estrada 2015]. No tocante a essa prática, pesquisas mostram que o uso de Tutores de Programação traz impactos positivos nos cursos introdutórios de programação [Cassel and Reis 2003].

Nesse sentido, um *feedback* rápido no determinado momento que o problema ocorre é indispensável para acelerar o aprendizado dos alunos. Para resolver esse problema, na introdução à programação vêm sendo adotando ferramentas automáticas para geração de *feedback* para os alunos, denominadas de ferramentas de tutoria inteligente [Fisher et al. 2016]. Nesse tipo de abordagem, um estudante que precisa de ajuda para resolver um determinado problema no código pode submeter a sua solução para o sistema de tutoria inteligente e receber *feedback* instantaneamente sobre o problema. Esse *feedback* é criado a partir dos dados de estudante que foram capazes de resolver o problema ou pelo *feedback* produzido pelo professor para estudantes que tiveram problemas similares no passado.

Desta forma, neste trabalho, propõe-se uma revisão sistemática de como os tutores de programação auxiliam na produção de *feedback* para estudantes em tarefas de programação. Como resultado, descreve-se um panorama geral desses sistemas de tutoria inteligente, incluindo, o tipo de abordagens empregadas nesses trabalhos. Este estudo dentre outras

contribuições, ajudará os projetistas de ferramentas a proporem melhores abordagens e pesquisados em geral a adquirirem um maior embasamento na área.

## 2 METODOLOGIA

Nesta seção, apresenta-se uma Revisão Sistemática da Literatura (RSL) de como os tutores de programação inteligentes auxiliam na produção de *feedback* para estudantes em tarefas de programação. Para conduzir a revisão sistemática, seguiu-se os procedimentos descritos por Kitchenham (2004). Uma RSL é o meio de identificar, avaliar e interpretar toda a pesquisa relevante para um determinado assunto [Kitchenham 2004]. Esta revisão tem como objetivo a análise de técnicas e abordagem existentes na literatura, para ajudar estudantes na aprendizagem de programação. Uma das principais atividades em uma RSL é o estabelecimento de questões de pesquisa [Kitchenham 2004]. A questão de pesquisa deste trabalho foi formulada de forma a focar o estudo no objetivo central do trabalho, resultando na seguinte pergunta:

**RQ1** – Como as abordagens de tutores de programação inteligente auxiliam na produção de *feedback* para estudantes em tarefas de programação?

### 2.1 SELEÇÃO DOS ESTUDOS

Para responder à questão de pesquisa foram identificados a intervenção, população e saída a partir da questão de pesquisa, uma abordagem que é comumente aplicada na literatura [Kitchenham 2004]. A partir desses termos foram derivados os termos usados para encontrar os trabalhos apresentados nessa revisão. Para cada um desses termos, identificou-se sinônimos, além desses foram procurados para palavras-chaves de artigos disponíveis, de forma a identificar quais os termos que são comumente utilizados na literatura. Após isso, construiu-se a *string* de busca para ser utilizada nas bases de dados. Na Tabela 1, sumariza-se os principais termos usados para realizar a revisão sistemática da literatura.

Brazilian Journal of Development

Tabela 1: Termos utilizados para construir a *string* de busca.

| Característica | Valor (inglês) | sinônimos (em inglês) |
|---|---|---|
| População | *estudantes (students)* | *programmer, programming students* |
| Intervenção | tutores de programação (*programming tutors*) | *programming education* |
| Saída | sistemas de tutoria (*tutoring systems*) | *feedback, hints, fix* |

Fonte: Autoria própria

Na Tabela 1, apresenta-se os termos elencados nas características intervenção, população e saída foram extraídos da questão de pesquisa. A categoria intervenção define qual tema será investigada na revisão sistemática, a população define os termos referentes ao público alvo incluído na pesquisa e a característica saída refere-se aos resultados esperados dos estudos. Para encontrar palavras-chaves, a seguir, baseou-se em termos focados no campo apresentado na problemática. Com os termos encontrados, criou-se uma *string* de busca booleana que incluem e relaciona os termos mais comuns na área de técnicas e abordagem para introdução à programação:

> ("programming students" or "students" or "programmer") and ("programming education" or "programming tutors") and ("feedback" or "hints" or "fix" or "tutoring systems" or "feedbacks generation" or "automatic hint generation")

Os critérios para seleção nas bases de dados, encontram-se sumarizados na Tabela 2.

Tabela 2: Critério de seleção de fontes

| Linguagem | Inglês |
|---|---|
| Método | Usando engenho de pesquisa da lista de base de dados |
| Base de dados | IEEE Explore, ACM Digital Library |
| Revisão | Autores |

Fonte: Autoria própria

A *string* de busca básica pode variar dependendo da sintaxe de empregada pela base de dados. De forma a encontrar os termos usados na *string* de busca, empregou-se termos comumente utilizados na literatura. Foram utilizadas duas bases de dados digitais para realizar a pesquisa, a *Associaton for Computing Machinery* (ACM) e *Institute of Electical and*

*Electronics Engineers* (IEEE), duas plataformas que são comumente aplicadas na condução de revisões sistemáticas da literatura. A ACM retornou a maior quantidade de trabalhos na área. O número de trabalhos retornados por cada base são apresentados na Tabela 3.

O processo de pesquisa foi feito de forma manual em bases de dados digitais, considerando trabalhos como artigos, periódicos, resumos estendidos, relato de experiência, estudos empíricos ou aplicação de alguma técnica/abordagem voltada a área, publicados entre 2014 e 2018. Para isso, empregou-se uma *string* de busca – a consulta que será realizada nas bases de dados digitais.

Tabela 3: Bases de dados e resultados da pesquisa

| Base de busca digital | Quantidade |
|---|---|
| ACM | 42 |
| IEEE | 16 |

Fonte: Autoria própria

A partir do processo de pesquisa da subseção anterior, os trabalhos retornados foram submetidos a uma segunda etapa, a critérios que vão determinar a inclusão ou exclusão. Os critérios de inclusão e exclusão foram elaborados para extrair apenas os dados de trabalhos que são relevantes para o objetivo do estudo. Logo, para determinar se os trabalhos atendem aos critérios de inclusão ou exclusão, os resumos foram lidos. Para trabalhos semelhantes que apresentavam a evolução de uma mesma pesquisa e que continha os mesmos autores, foi selecionada a versão mais completa. Os trabalhos selecionados, com base nos critérios de inclusão e exclusão, tiveram seus dados extraídos e sintetizados para fornecerem a análise e apresentação dos resultados. Na Tabela 4, são apresentados os critérios de inclusão e exclusão elaborados.

Tabela 4: Critérios de inclusão e exclusão.

| Critérios de Inclusão | Critérios de Exclusão |
|---|---|
| 1. Trabalhos completos sobre técnicas ou abordagens para ajudar estudantes a resolver problemas em tarefas de programação;<br>2. Trabalhos voltados para área de ensino-aprendizagem de programação; | 1. Trabalhos duplicado (artigos derivados da mesma pesquisa);<br>2. Trabalhos que estão fora dos critérios de inclusão. |

| | |
|---|---|
| 3. | Trabalhos publicados entre 2014 e 2018. |
| 4. | Trabalhos em português ou inglês |

Fonte: Autoria própria

Uma vez identificados os trabalhos como incluídos, o texto completo foi lido. Em seguida, extraiu-se os seguintes dados: título, ano, autores, características e detalhes da publicação. Destes trabalhos, foram extraídos também dados que pudessem ajudar a responder à questão de pesquisa, tais como a abordagem utilizada, linguagem de programação, tipo de saída/retorno da abordagem, nível de escolaridade, resumo, limitações, benefícios, respostas de suas questões de pesquisas e sugestões de pesquisas futuras.

## 3 RESULTADOS

Esta seção resume os resultados alcançados pelo estudo. O processo de pesquisa retornou um total de 58 trabalhos. Com base nos critérios de inclusão e exclusão, foram excluídos 22 trabalhos, ficando assim 36 trabalhos relevantes, selecionados para extração dos dados. A Tabela 5 apresenta os resultados gerais dos processos de pré-seleção e inclusão.

Na exclusão de trabalhos pré-selecionados, o principal motivo identificado foi o de o trabalho não estar relacionado a uma técnica ou abordagem na área de ensino-aprendizagem de programação, se tornando falsos positivos. Outros trabalhos excluídos não se relacionavam à programação.

Tabela 5: Resultado geral do processo de pesquisa.

| Base de Dados | Artigos Pré-Selecionados | Artigos Incluídos |
|---|---|---|
| ACM | 42 | 26 |
| IEEE | 16 | 10 |
| Total | 58 | 36 |

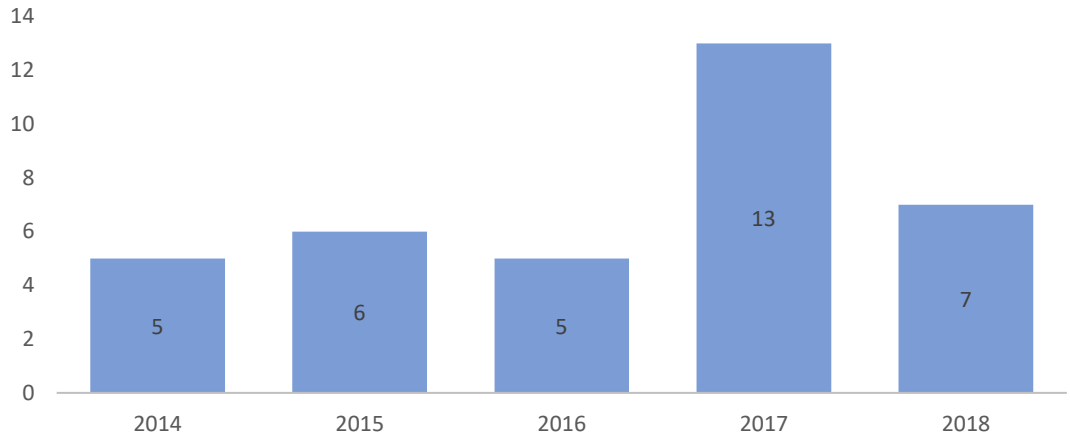
Fonte: Autoria própria

Na Figura 1 é apresentada a distribuição temporal dos artigos. Os dados mostram que as pesquisas na área vêm crescendo nos últimos anos, como é ilustrado na figura. Pode-se notar que mais de 50% dos artigos selecionados/incluídos nesse estudo foram publicados nos últimos 2 anos, com um aumento significativo no ano de 2017. Também se notou um aumento no

Brazilian Journal of Development

interesse de pesquisas nessa área, com o desenvolvimento de técnicas e abordagens no ensino-aprendizagem de programação.

Figura 1: Distribuição anual dos artigos.




Fonte: Autoria própria

Para apresentar um quadro geral das abordagens encontradas na literatura, dividiu-se os trabalhos em categorias. Na Figura 2, apresenta-se os tipos de abordagens propostas. Nos dados apresentados, 36% dos artigos propõem ferramentas de tutoria inteligente (Kekeshita e Ohta (2016), Yan et al. (2017), Fuchs e Wolff (2016), Karavirta et al. (2015), Jeuring et al. (2014), Head et al. (2017), Keuning et al. (2014), Glassman et al. (2015), Quinson e Oster (2015), Makihara (2015), Canou et al. (2017), Arends et l. (2017), Fu et al. (2017)), 25% propõem estudos sobre abordagens (Suzuki et al. (2017), Miller et al. (2014), Crow et al. (2018), Albrecht et al. (2018), Almeida et al. (2018), Keuning et al. (2017), Hermans e Aivaloglou (2017), Brown et al. (2014), Pollari-Malmi et al. (2017)), 17% propõem a utilização de metodologias (Lazar et al. (2018), Yoshizawa e Watanobe (2018), Hicks et al. (2014), Haden et al. (2017), Ohshima et al. (2016), Kaila et al. (2018a)), 14% propõem a aplicação de técnicas pedagógicas (Schiling (2015), Kaila et al. (2018), Krugel and Ubwieser (2017), Haden et al. (2016), Machuca e Solarte Pabón (2016)) e 8% propõem outros tipos de abordagens (Matsuzawa et al. (2015), Gulwani et al. (2018), Wang et al. (2017)).

No tocante ao desenvolvimento de ferramenta de tutoria inteligente, Kekeshita e Ohta (2016) e Keuning et al. (2014) desenvolveram ferramentas de apoio à programação, que tem como objetivo facilitar o processo de aprendizagem de programação introdutória. Em contrapartida, Jeuring et al. (2014), Glassman et al. (2015) e Quinson e Oster (2015) mostram

sistemas utilizados em cursos online de programação, que simulam professores interativo capazes de fornecer *feedback* sobre soluções e indicar se o aluno está indo ou não pelo caminho certo. Por sua vez, Fu et al. (2017) propõe um sistema para o ensino e aprendizagem na linguagem C, através de um painel que captura o comportamento dos alunos em sala de aula e identifica as diferentes dificuldades enfrentadas por distintos alunos que procuram diferentes conhecimentos na programação. De outro modo, Head et al. (2017) introduz uma abordagem que combina a experiência do professor com síntese de programas orientada por dados.

Com base nos dados da figura, nota-se uma preferência por abordagens de ferramentas de tutoria inteligente, ferramentas de auxílio à programação que objetiva ajudar alunos com dificuldade em resolver problemas de programação, orientando-os em como corrigir falhas e evitar possíveis erros. Essas ferramentas auxiliam os professores no processo de ensino, focando principalmente no fornecimento de *feedback*, geração de dicas/pistas e correções de problemas de programação. Alguns dos trabalhos categorizados como estudos, metodologias e técnicas pedagógicas apresentam a experiência e resultado de se utilizar uma ferramenta de tutoria para auxiliar curso de introdução à programação (Lazar et al. (2018), Hicks et al. (2014), Keuning et al. (2017)).

Entre os trabalhos categorizados como abordagens, Suzuki et al. (2017) e Miller et al. (2014) apresentam abordagens que tem como objetivo gerar dicas de programação automática, pedagogicamente úteis. Por sua vez, Albrecht et al. (2018) traz uma abordagem para cursos de programação introdutória, relacionados a como lidar com problemas de tarefas de programação. Diferentemente, Almeida et al. (2018) utiliza a programação de jogos como abordagem para o ensino da programação introdutória.

A categorização de metodologia atribuída a trabalhos, diz respeitos a trabalhos que apresentam métodos como resolução do seu estudo. Dentre eles, Lazar et al. (2018), propõe regras de rescritas de programa como uma formalização de linguagem de programação, em termos de edições de códigos. Por sua vez, Yoshizawa e Watanobe (2018) e Haden et al. (2017) propõem métodos para detecção de erros em tarefas de programação e coleta de dados de programas que mais afetam os alunos.

Algumas técnicas pedagógicas encontradas nos estudos, apresentam a experiência de lecionar em disciplinas de introdução à programação, adotando sistemas e metodologias durante o curso, para o auxílio em sala de aula (Schiling (2015), Kaila et al. (2018), Krugel and Ubwieser (2017), Haden et al. (2016)).

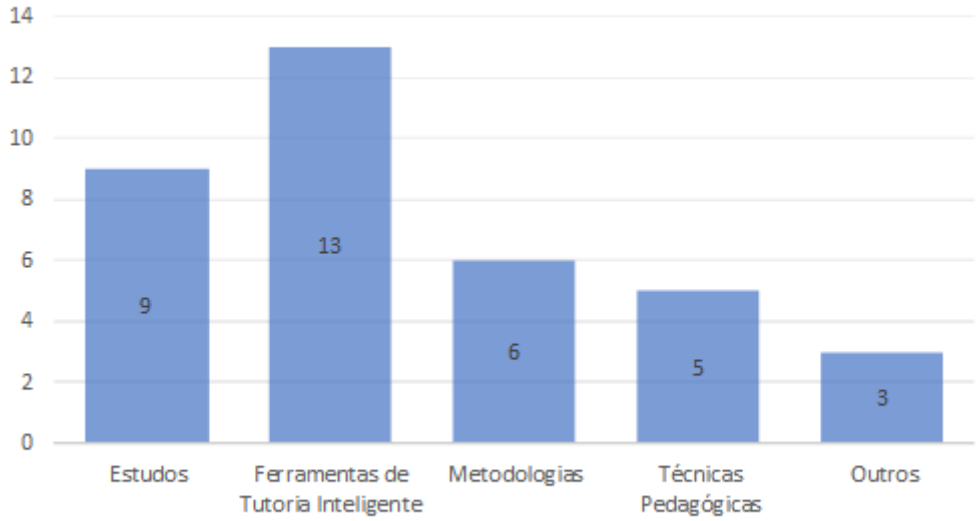
Brazilian Journal of Development

Entre os estudos na categoria outros tipos de estudos, Matsuzawa et al. (2015) descreve um sistema de tradução de linguagem de programação. De outro modo, Lazar et al. (2018) apresenta um algoritmo de reparo de programa, para atribuições de programação introdutória e Makihara (2015) um sistema para análise de dados de alunos, referente a problemas na programação.

## 4 TRABALHOS RELACIONADOS

No tocante ao desenvolvimento de iniciativas para o ensino de programação, várias abordagens foram propostas. Dentre elas, Agbo et al. (2019) realiza uma revisão sistemática da literatura para identificar como o pensamento computacional pode ser utilizado para o ensino de programação. O estudo busca identificar as abordagens que utilizam pensamento computacional para o ensino de programação e realiza uma classificação das abordagens utilizadas na área. Ao contrário de Agbo et al. (2019), este trabalho aborda o uso de tutores inteligentes para o ensino de programação.

Figura 2: Abordagens propostas nos artigos.




Fonte: Autoria própria

De outra forma, Crow et al. (2018) apresentam uma revisão sistemática do uso de tutores inteligentes na programação. O trabalho identifica dentre outros fatores, como recursos suplementares podem ser utilizados no sistema e mostra implicações do uso dos tutores inteligentes e como esses podem ser aprimorados para receber uma gama ampla de recursos suplementares. Diferentemente de Crow et al. (2018), este trabalho foca em como os tutores

inteligentes são utilizados para fornecer *feedback* aos estudantes de programação, de forma a melhorem o desempenho dos discentes.

Além desses, Sharid et al. (2019) apresentam uma revisão da literatura sobre o desenvolvimento de jogos relacionados a programação. O trabalho elenca várias características nesse tema, dentre elas a caracterização dos tópicos de programação que são tipicamente utilizados no projeto de jogos para programação, quais são as linguagens de programação comumente utilizadas e os *frameworks* utilizados.

No tocante à técnicas, Rolim et al. (2020) cria uma disciplina, Pré-Algoritmos, que utilizados *online* e linguagens de blocos para o ensino de programação. No tocante a estudos sobre problemas de programação, Queiroz et al. (2018) estudaram os fatores que motivam ou desmotivam os estudantes para o aprendizado e programação e Moreira et al. (2018) estudaram os desafios apresentados pelos estudantes para o aprendizado de programação.

### 4.1 AMEAÇAS À VALIDADE E LIMITAÇÕES

Assim como toda RSL, essa revisão pode possuir limitações com respeito a sua validade, tais como o processo de condução, a inserção de erros em algumas das etapas, por exemplo na pré-seleção dos artigos, na inclusão e exclusão de artigos e na extração dos dados. Para mitigar essa ameaça, elaborou-se um protocolo para guiar essa RSL. Adicionalmente, alguns estudos relevantes podem estar ausentes devido a se encontrarem em outra base de dados. Para superar essa limitação, selecionou-se bases de dados digitais comumente utilizados em RSL na área de computação. Outras duas possíveis limitações são número de artigos analisados durante a pré-seleção e a inclusão de estudos irrelevantes. Para superar essa limitação, esse estudo foi acompanho por mais de um pesquisador, que analisaram e acompanharam as etapas dessa RSL.

## 5 CONCLUSÃO

Neste trabalho foi apresentada uma revisão sistemática de como os tutores de programação auxiliam na produção de *feedback* para estudantes em tarefas de programação. A revisão sistemática da literatura retornou trabalhos válidos, que se enquadram nos critérios de exclusão e inclusão, para responder à questão de pesquisa utilizada. Os trabalhos elencados possuem como objetivo auxiliar os projetos de sistemas de tutoria inteligente, no sentido de auxiliar os estudantes que apresentam dificuldades na resolução de tarefas de programação, que costumam apresentar dificuldades no aprendizado de atividades que envolvam conceitos abstratos e lógica de programação.

Brazilian Journal of Development

**AGRADECIMENTOS**

O trabalho foi financiado pela Universidade Federal Rural do Semi-Árido -- UFERSA por meio da Pró-Reitoria de Pesquisa e Pós-Graduação (PROPPG) através do Edital PROPPG Nº 39/2019 de Apoio a Grupos de Pesquisa, concedida ao PIH00022-2019 do grupo de pesquisa Laboratório de Inovações em Software (LIS) do líder Reudismam Rolim de Sousa.

## REFERÊNCIAS

[Agbo et al. 2019] Agbo, F. J., Oyelere, S. S., Suhonen, J., and Adewumi, S. (2019). A systematic review of computational thinking approach for programming education in higher education institutions. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. Koli Calling'19, New York, NY, USA. Association for Computing Machinery.

[Albrecht et al. 2018] Albrecht, E., Gumz, E, and Grabowski, J. (2018). Experiences in introducing blended learning in an introductory programming course. *In Proceedings of the 3rd European Conference of Software Engineering Education*, ECSEE'18, page 93-101, New York, NY, USA. Association for Computing Machinery.

[Almeida et al. 2018] Almeida, J. B., Cunha, A., Macedo, N., Pacheco, H., and Proenga, J. (2018). Teaching how to program using automated assessment and functional glossy games (experience report). *Proc. ACM Program. Lang.*, 2(ICFP).

[Altuna Castillo and Guibert Estrada 2015] Altuna Castillo, E. J. and Guibert Estrada, L. (2015). Domain knowledge representation for programming teaching. *IEEE Latin America Transactions*, 13(5):1528-1533.

[Arends et al. 2017] Arends, H., Keuning, H., Heeren, B., and Jeuring, J. (2017). An intelligent tutor to learn the evaluation of microcontroller i/o programming expressions. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. Koli Calling '17, page 2-9, New York, NY, USA. Association for Computing Machinery.

[Brown et al. 2014] Brown, N. C. C., Kolling, M., McCall, D., and Utting, I. (2014). Blackbox: A large scale repository of novice programmers? activity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 223-228, New York, NY, USA. Association for Computing Machinery.

[Canou et al. 2017] Canou, B., Di Cosmo, R., and Henry, G. (2017). Scaling up functional programming education: Under the hood of the ocaml mooc. *Proc. ACM Program. Lang.*, l(ICFP).

[Cassel and Reis 2003] Cassel, L. and Reis, R. A. (2003). *Erratum to: Informatics Curricula and Teaching Methods*, pages E1-E1. Springer US, Boston, MA.

[Chrysafiadi and Virvou 2014] Chrysafiadi, K. and Virvou, M. (2014). K.e.m.cs: A set of student's characteristics for modeling in adaptive programming tutoring systems. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, pages 106-110.

[Costa 2013] Costa, T. H. (2013). Análise dos problemas enfrentados por alunos de programação. Trabalho de Conclusão de Curso. UEPB.

[Crow et al. 2018] Crow, T., Luxton-Reilly, A., and Wuensche, B. (2018). Intelligent tutoring systems for programming education: A systematic review. In *Proceedings of the 20th Australasian Computing Education Conference,* ACE '18, page 53-62, New York, NY, USA. Association for Computing Machinery.

[Fisher et al. 2016] Fisher, W., Rader, C., and Camp, T. (2016). Online programming tutors or paper study guides? In *2016 IEEE Frontiers in Education Conference (FIE),* pages 1-6.

[Fu et al. 2017] Fu, X., Shimada, A., Ogata, H., Taniguchi, Y., and Suehiro, D. (2017). Real-time learning analytics for C programming language courses. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, LAK '17, page 280-288, New York, NY, USA. Association for Computing Machinery.

[Fuchs and Wolff 2016] Fuchs, M. and Wolff, C. (2016). Improving programming education through gameful, formative feedback. In *2016 IEEE Global Engineering Education Conference (EDUCON),* pages 860-867.

[Giraffa and da Costa Mora 2013] Giraffa, L. M. M. and da Costa Mora, M. (2013). Evasão na disciplina de algoritmo e programação: Um estudo a partir dos fatores intervenientes na perspectiva do aluno. In *Tercera Conferencia sobre el Abandono en la Education Superior*, CLABES '13, pages 1-10. CLABES.

[Glassman et al. 2015] Glassman, E. L., Scott, J., Singh, R., Guo, P. J., and Miller, R. C. (2015). Overcode: Visualizing variation in student solutions to programming problems at scale. *ACM Trans. Comput.-Hum. Interact.*, 22(2).

[Gross and Pinkwart 2015] Gross, S. and Pinkwart, N. (2015). Towards an integrative learning environment for java programming. In *2015 IEEE 15th International Conference on Advanced Learning Technologies,* pages 24-28.

[Gulwani et al. 2018] Gulwani, S., Radicek, I., and Zuleger, F. (2018). Automated clustering and program repair for introductory programming assignments. *SIGPLAN Not.*, 53(4):465-480.

[Haden et al. 2016] Haden, P., Gasson, J., Wood, K., and Parsons, D. (2016). Can you learn to teach programming in two days? In *Proceedings of the Australasian Computer Science Week Multiconference,* ACSW '16, New York, NY, USA. Association for Computing Machinery.

[Haden et al. 2017] Haden, P" Parsons, D., Wood, K., and Gasson, J. (2017). Student affect in cs1: Insights from an easy data collection tool. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research,* Koli Calling '17, page 40-49, New York, NY, USA. Association for Computing Machinery.

Brazilian Journal of Development

[Head et al. 2017] Head, A., Glassman, E., Soares, G., Suzuki, R., Figueredo, L., D'Antoni, L., and Hartmann, B. (2017). Writing reusable code feedback at scale with mixed- initiative program synthesis. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S '17, page 89-98, New York, NY, USA. Association for Computing Machinery.

[Hermans and Aivaloglou 2017] Hermans, F. and Aivaloglou, E. (2017). To scratch or not to scratch? a controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, page 49-56, New York, NY, USA. Association for Computing Machinery.

[Hicks et al. 2014] Hicks, A., Peddycord II, B., Rindos, I., and Simmons, C. (2014). A comparison of two approaches for hint generation in programming tutors (abstract only). In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 718-719, New York, NY, USA. Association for Computing Machinery.

[Jeuring et al. 2014] Jeuring, J., van Binsbergen, L. T., Gerdes, A., and Heeren, B. (2014). Model solutions and properties for diagnosing student programs in ask-elle. In *Proceedings of the Computer Science Education Research Conference*, CSERC '14, page 31-40, New York, NY, USA. Association for Computing Machinery.

[Kaila et al. 2018a] Kaila, E., Laakso, M. ., Rajala, T., Makelainen, A., and Lokkila, E. (2018a). Technology-enhanced programming courses for upper secondary school students. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0683-0688.

[Kaila et al. 2018b] Kaila, E., Laakso, M., and Kurvinen, E. (2018b). Teaching future teachers to code — programming and computational thinking for teacher students. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0677-0682.

[Kakeshita and Ohta 2016] Kakeshita, T. and Ohta, K. (2016). Student feedback function for web-based programming education support tool pgtracer. In *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 322-327.

[Karavirta et al. 2015] Karavirta, V., Haavisto, R., Kaila, E., Laakso, M., Rajala, T., and Salakoski, T. (2015). Interactive learning content for introductory computer science course using the ville exercise framework. In *2015 International Conference on Learning and Teaching in Computing and Engineering*, pages 9-16.

[Keuning et al. 2014] Keuning, H., Heeren, B., and Jeuring, J. (2014). Strategy-based feedback in a programming tutor. In *Proceedings of the Computer Science Education Research Conference*, CSERC '14, page 43-54, New York, NY, USA. Association for Computing Machinery.

[Keuning et al. 2017] Keuning, H., Heeren, B., and Jeuring, J. (2017). Code quality issues in student programs. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE'17, page 110-115, New York, NY, USA. Association for Computing Machinery.

[Kitchenham 2004] Kitchenham, B. (2004). Procedures for performing systematic reviews.

[Krugel and Hubwieser 2017] Krugel, J. and Hubwieser, P. (2017). Computational thinking as springboard for learning object-oriented programming in an interactive mooc. In 2*017 IEEE Global Engineering Education Conference* (EDUCON), pages 1709-1712.

[Lazar et al. 2018] Lazar, T., Sadikov, A., and Bratko, I. (2018). Rewrite rules for debugging student programs in programming tutors. *IEEE Transactions on Learning Technologies*, 11(4):429440.

[Machuca and Solarte Pabon 2016] Machuca, L. and Solarte Pabon, O. (2016). Improving student performance in a first programming course. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '16, page 367, New York, NY, USA. Association for Computing Machinery.

[Makihara 2015] Makihara, E. (2015). Pockets: A tool to support exploratory programming for novices and educators. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 1066-1068, New York, NY, USA. Association for Computing Machinery.

[Matsuzawa et al. 2015] Matsuzawa, Y., Ohata, T., Sugiura, M., and Sakai, S. (2015). Language migration in non-cs introductory programming through mutual language translation environment. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE ’15, page 185-190, New York, NY, USA. Association for Computing Machinery.

[Miller et al. 2014] Miller, H., Haller, P., Rytz, L., and Odersky, M. (2014). Functional programming for all! scaling a mooc for students and professionals alike. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, page 256-263, New York, NY, USA. Association for Computing Machinery.

[Moreira et al. 2018] Moreira, G. L., Holanda1, W., da S. Coutinho, J. C., and Chagas, F. S. (2018). Desafios na aprendizagem de programação introdutória em cursos de ti da ufersa, campus pau dos ferros: um estudo exploratório. In *Proceedings of the III Encontro do Oeste Potiguar*, ECOP ’18, pages 90-96. ECOP.

[Ohshima et al. 2016] Ohshima, Y., Warth, A., Freudenberg, B., Lunzer, A., and Kay, A. (2016). Towards making a computer tutor for children of all ages: A memo. In *Proceedings of the Programming Experience 2016 (PX/16) Workshop*, PX/16, page 21-25, New York, NY, USA. Association for Computing Machinery.

[Pollari-Malmi et al. 2017] Pollari-Malmi, K., Guerra, J., Brusilovsky, P., Malmi, L., and Sirkia, T. (2017). On the value of using an interactive electronic textbook in an introductory programming course. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, Koli Calling '17, page 168-172, New York, NY, USA. Association for Computing Machinery.

[Queiroz et al. 2018] Queiroz, J. V., Rodrigues, L. M., and Coutinho, J. (2018). Um relato dos fatores motivacionais na aprendizagem de programação na perspectiva de alunosiniciantes em programagao da universidade federal rural do semi-árido campus pau dos ferros-rn. In

*Proceedings of the HI Encontro do Oeste Potiguar*, ECOP '18, pages 90-96. ECOP.

[Quinson and Oster 2015] Quinson, M. and Oster, G. (2015). A teaching system to learn programming: The programmer's learning machine. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, page 260-265, New York, NY, USA. Association for Computing Machinery.

[Rolim et al. 2020] Rolim, R., Leite, F. T., de Oliveira Guimarães, A., and de Oliveira, A. R. (2020). Pré-algoritmos - ações de apoio a melhoria do ensino de graduação. *Brazilian Journal of Development*, 6(3):12625-12635.

[Schilling 2015] Schilling, W. W. (2015). Analyzing the impact of asynchronous multimedia feedback on novice computer programmers. In *Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE)*, FIE '15, page 1-8, USA. IEEE Computer Society.

[Shahid et al. 2019] Shahid, M., Wajid, A., Haq, K. U., Saleem, I., and Shujja, A. H. (2019). A review of gamification for learning programming fundamental. In *2019 International Conference on Innovative Computing (ICIC)*, pages 1-8.

[Suzuki et al. 2017] Suzuki, R., Soares, G., Glassman, E., Head, A., D'Antoni, L., and Hartmann, B. (2017). Exploring the design space of automatically synthesized hints for introductory programming assignments. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, page 2951-2958, New York, NY, USA. Association for Computing Machinery.

[Thinakaran and Ali 2015] Thinakaran, R. and Ali, R. (2015). Work in progress: An initial review in programming tutoring tools. In *2015 IEEE International Conference on Teaching, Assessment, and Learningfor Engineering (TALE)*, pages 1-4.

[Wang et al. 2017] Wang, Y., White, W. M., and Andersen, E. (2017). Pathviewer: Visualizing pathways through student data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 960-964, New York, NY, USA. Association for Computing Machinery.

[Yan et al. 2017] Yan, Y., Hara, K., Kazuma, T., and He, A. (2017). A method for personalized C programming learning contents recommendation to enhance traditional instruction. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 320-327.

[Yoshizawa and Watanobe 2018] Yoshizawa, Y. and Watanobe, Y. (2018). Logic error detection algorithm for novice programmers based on structure pattern and error degree. In *2018 9th International Conference on Awareness Science and Technology (iCAST)*, pages 297-301.