Abordagens para Prover *Feedback* Automático em Atividades de Programação: Uma Revisão da Literatura

José Raul de Brito Andrade

ECIT Presidente João Goulart
raul.andrade@dce.ufpb.br

Resumo: Este trabalho apresenta uma revisão da literatura na qual identificamos quais as principais técnicas utilizadas nas ferramentas para prover feedback automático em programação. Nosso objetivo foi identificar as características dessas técnicas e estratégias usadas para auxiliar e mensurar a aprendizagem. Dentre os resultados, identificamos que a maioria dos estudos concentram-se nas aulas de graduação. Também identificamos que o uso de testes automáticos e comparação de códigos são estratégias populares. Os pesquisadores descreveram que os estudantes podem se beneficiar do uso dessas ferramentas, uma vez que melhoram seu desempenho e motivação. No entanto, embora essas abordagens sejam populares, não encontramos uma prática comum para avaliá-las.

Palavras-chave: *Revisão da literatura; Feedback em programação; Educação e computação.*

1. Introdução

As disciplinas de programação são essenciais na formação dos estudantes em cursos de Computação, devido à exigência desse conhecimento para vagas profissionais na área. No entanto, o processo de ensino e aprendizagem nessas disciplinas, principalmente nas introdutórias, apresenta diversos desafios. Dentre eles, destaca-se o elevado número de alunos por turma [13].

Na prática, tipicamente, essas disciplinas envolvem uma grande quantidade de atividades, o que dificulta no acompanhamento e avaliação dos alunos durante o semestre letivo. Com intuito de abordar esse problema, várias ferramentas e técnicas estão sendo propostas e aplicadas no ensino de programação para prover *feedback* automático para os alunos.

A área de avaliação de trabalhos de alunos foi amplamente explorada em muitos artigos de revisão bem conhecidos, nos quais esses sistemas são descritos à luz de possíveis aplicações, como casos de uso de ensino [1, 7]. Desse modo, o objetivo desta pesquisa é identificar as principais abordagens utilizadas para prover *feedback* automático para atividades de programação. Para isso, realizamos uma revisão da literatura, baseada no trabalho de Kitchenham [9], para identificar estudos relevantes em três dos principais repositórios da área: ACM, IEEE e ScienceDirect.

Dentre os 464 estudos identificados, selecionamos 203 para uma análise mais detalhada. Verificamos aspectos essenciais das abordagens de prover *feedback* para atividades de programação e, como resultado, analisamos 10 artigos nos quais verificamos que a maioria se concentra em turmas de graduação e na avaliação da experiência do usuário. Também identificamos que a comparação de código e o uso de testes automáticos programados previamente por professores ou monitores são estratégias populares. Com este estudo, pretendemos auxiliar os pesquisadores e desenvolvedores, envolvidos em projetos de ferramentas de *feedback* automático, na escolha de qual abordagem usar ou se basear, considerando a aceitação e resultados obtidos.

Este artigo está organizado da seguinte forma: na Seção 2 descrevemos a metodologia que adotamos para realizar esta revisão da literatura. Nós apresentamos e discutimos os resultados na Seção 3. Finalmente, abordamos as conclusões do estudo com as instruções para trabalhos futuros na Seção 4.

2. Metodologia de Pesquisa

Segundo Kitchenham [9], o processo metodológico de uma revisão da literatura consiste em um conjunto de perguntas e critérios para inclusão e exclusão de artigos, disponíveis e comprovados cientificamente. Posteriormente, os inclusos serão mapeados para responder às questões iniciais da pesquisa. Na revisão da literatura conduzida, adotamos quatro passos essenciais sugeridos pelo autor: (1) definição das questões de pesquisa; (2) condução da pesquisa; (3) triagem dos documentos; e (4) a extração de dados e mapeamento. Apresentamos no Quadro 1 os resultados obtidos na conclusão de cada etapa.

Quadro 1. Etapas da revisão da literatura.

	Etapas do processo	**Resultados**
1	Definição das questões de pesquisa	Escopo da pesquisa
2	Condução da pesquisa	Todos os artigos
3	Triagem dos documentos	Artigos relevantes
4	Extração de dados e mapeamento	Mapeamento da literatura

2.1. Questão de Pesquisa

Este estudo tem por objetivo sistematizar as buscas por produções científicas referentes às abordagens usadas para prover *feedback* automático em atividades de programação, de modo a descobrir as principais pesquisas da área, as técnicas mais utilizadas e como estão distribuídos os trabalhos desse tema. Entendendo que as questões de pesquisa representam o objetivo do estudo, definimos a seguinte questão:

Q1: Quais são as técnicas mais usadas em ferramentas para prover feedback automático em atividades de programação?

O objetivo foi verificar quais as principais abordagens para prover *feedback* em programação, de qual modo e em quais cenários elas são utilizadas.

DOI: 10.22478/ufpb.2595-0622.2019v3n1.46721

2.2. Condução da Pesquisa

Após definirmos a questão de pesquisa, projetamos a atividade seguinte para selecionar os artigos relacionados com o tema deste estudo. Selecionamos os termos chave e repositórios de pesquisa. As etapas dessa atividade consistiram em:

(1) Escolha de sinônimos para "*automated feedback*" no contexto de ensino de programação;
(2) Teste de cada sinônimo de "*automated feedback*" e análise dos resultados em cada repositório digital;
(3) Uso do *Boolean* OR para conectar os sinônimos selecionados;
(4) Uso do *Boolean* AND para conectar os termos não sinônimos; e
(5) Execução de testes piloto para avaliar a sequência de pesquisa.

Como resultado, obtivemos a seguinte *string* de busca:

((("automated" OR "automatic") AND "feedback") AND "programming" AND "education")

Para realizar as buscas, escolhemos três dos principais repositórios digitais, são eles: ACM, IEEE e ScienceDirect (Elsevier). Selecionamos esses por serem os principais veículos de publicação da área. Realizamos a pesquisa através de busca manual utilizando a *string* de busca. Inicialmente, foram retornados 464 trabalhos, os quais incluíam monografias, artigos e livros. Os estudos foram coletados com base em uma pesquisa de título e resumo. Assim, nesta etapa (etapa 2, de acordo com o Quadro 1), o resultado foi todos os artigos da busca inicial.

Em uma primeira triagem, excluímos os trabalhos que: (i) não fossem artigos completos; (ii) não possuíssem no seu texto o termo utilizado na busca; (iii) não abordassem aspectos técnicos e (iv) publicados há mais de 5 anos. Neste momento, foram selecionados 203 artigos. O Quadro 2 apresenta o resumo da pesquisa retornada em cada repositório.

Quadro 2. Resumo dos resultados da revisão da literatura.

Repositório	Resultados	1ª Triagem
ACM	286	119
IEEE	133	63
ScienceDirect	45	21

2.3. Triagem dos Documentos

Na segunda triagem, mais minuciosa, foram aplicados para cada artigo selecionado os **critérios de inclusão**: (a) responder à questão de pesquisa (Q1); (b) ser acessível eletronicamente; (c) tratar sobre ensino em programação independente de contexto; e (d) pertencer à literatura em Ciência da Computação; e os **critérios de exclusão**: (a) estudo não referente à Engenharia de Software; (b) estudo escrito em outro idioma que não inglês; e (c) ser relatório técnico, livro, documento disponível na forma de resumo, artigo resumido ou apresentação e pesquisa secundária.

A maioria dos estudos foi excluída porque abordava apenas aspectos educacionais, com pouco ou nenhum foco na parte técnica. Assim, nesse passo, foram selecionados 10 estudos como mais relevantes para nossos objetivos. Após essa seleção, todos os artigos foram lidos na íntegra e extraímos os dados importantes para este estudo.

3. Resultados e Discussão

Nesta seção detalhamos o resultado da revisão da literatura. Ela foi realizada entre os meses de maio e julho de 2019. Primeiramente, apresentamos os resultados gerais obtidos dos artigos selecionados, como países onde a pesquisa foi realizada, categorização dos trabalhos e distribuição de estudos por ano de publicação. Desse modo, a seção restante responde à questão de pesquisa definida anteriormente.

3.1 Resultados Gerais

Os 10 trabalhos que compõem a seleção realizada na segunda triagem de documentos foram lidos na íntegra e nessa leitura identificamos termos e características que nos permitiu categorizá-los. Identificamos cinco trabalhos de validação (E02, E05, E07, E08 e E10) e seis de avaliação e relatos de experiência (E01, E03, E06, E07, E09 e E10). Devido aos objetivos da pesquisa, estas categorias podem se sobrepor, ou seja, os artigos podem ser classificados em uma ou mais categoria.

Os trabalhos selecionados estão concentrados principalmente nos anos de 2016 e 2017, tendo apenas um de outro ano. Quanto à distribuição de trabalhos por países, há cinco trabalhos dos Estados Unidos, sendo dois deles em parceria com pesquisadores do Brasil. A Austrália, Bélgica, França, Índia e Marrocos tem um trabalho cada. Todos os trabalhos selecionados tiveram como público-alvo alunos de graduação. Apresentamos no Quadro 3 informações gerais dos trabalhos selecionados.

Quadro 3. Informações gerais dos trabalhos selecionados.

ID	Conferência/ Repositório	Ano	Trabalho
E01	CHESE/ACM	2016	Combéfis, S; Schils, A. [2]
E02	UMAP/ACM	2017	Chow, S. *et al.* [3]
E03	ItiCSE/ACM	2016	Gao, J; Pang, B; Lumetta, S. [4]
E04	L@S/ACM	2017	Head, A. *et al.* [5]
E05	ICETC/ACM	2018	Hossein, S; Peter, Y. [6]
E06	SIGPLAN/ACM	2016	Kim, D. *et al.* [8]
E07	ItiCSE/ACM	2017	Parihar, S. *et al.* [10]
E08	ICSE/IEEE	2017	Rolim, R. *et al.* [11]
E09	L@S/ACM	2017	Wang, K. *et al.* [12]
E10	ICEIT/IEEE	2016	Zougari, S. *et al.* [14]

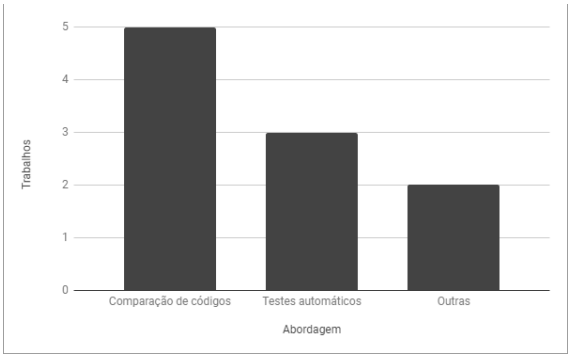
DOI: 10.22478/ufpb.2595-0622.2019v3n1.46721

Respondendo à Q1, de acordo com os estudos selecionados, as principais abordagens utilizadas são:

i) comparação de códigos,
ii) testes automáticos implementados previamente nas ferramentas,
iii) síntese de programas e programação por exemplos (PBE – *programming by example*).

A Figura 1 apresenta uma visão geral dos resultados.

Figura 1. Resumo dos resultados.



Dentre as abordagens citadas, a mais presente nos trabalhos analisados foi a de comparação de códigos. Nos estudos E05 e E06, essa abordagem é adotada de modo que o código do aluno é analisado e comparado com uma solução correta, definida previamente pelo professor. Então, é usado um algoritmo iterativo para calcular correspondências que mapeiam uma instância de instrução para alguma instância na outra versão.

No estudo E01 é explorado como identificar automaticamente erros utilizando ferramentas de detecção de plágio de código para medir a semelhança entre os códigos e, no estudo E09, é apresentado o IGrader, uma ferramenta baseada em dados que fornece *feedback* personalizado para atividades de programação. Tendo em vista uma submissão incorreta, o iGrader primeiro encontra uma submissão estreitamente relacionada (sintaticamente e semanticamente) para calcular discrepâncias de expressão correspondentes e, em seguida, um conjunto de reparações a partir das discrepâncias, para corrigir o programa. No estudo E02, os autores utilizaram uma variedade de técnicas, como filtragem, agrupamento e mineração de padrões para gerar *feedback*.

No estudo E08 é utilizada a abordagem de programação por exemplos, na qual foi apresentada a REFAZER, uma técnica para sintetizar transformações de programas. Esse trabalho avançou o desenvolvimento de PBE totalmente sem supervisão, uma vez que automatiza a extração de exemplos de entrada e de saída dos conjuntos de dados utilizados.

Outra técnica utilizada foi a síntese de programas. Ela consiste em sintetizar um programa em uma linguagem de programação subjacente para alcançar um objetivo de determinada especificação [5]. O trabalho E04 faz uso dessa abordagem, quando os autores apresentaram duas ferramentas: *Mistakebrowser* e *Fixpropagator*. A *Mistakebrowser* identifica erros nos códigos submetidos pelos alunos e permite ao professor deixar comentários. Já na *Fixpropagator*, além de possibilitar que o professor comente, ela "aprende" o *feedback* para que seja fornecido em erros semelhantes em novas submissões de outros alunos.

Usando a abordagem de testes automáticos, identificamos os estudos E03, E07 e E10. Essa técnica é uma das mais comuns, mas um diferencial do estudo E03 em relação à técnica é fornecer *feedback* do código analisado. A qualidade do *feedback* gerado é avaliada com base na capacidade de encontrar defeitos dentro dos códigos dos alunos. Os resultados mostraram que a estrutura automatizada pode descobrir mais erros dentro de submissões de alunos e pode fornecer *feedback* oportuno e útil.

De acordo com os dados analisados, as ferramentas/técnicas que fazem comparação entre códigos apresentam resultados mais consistentes de seu uso em relação à aplicação no ambiente de sala de aula. A abordagem de testes automáticos também é bastante utilizada. No entanto, não é trivial definir todos os cenários possíveis. Assim sendo, essa técnica tem sido complementada com outras para prover *feedback* com maior qualidade. As técnicas de síntese de programas e PBE, vêm se mostrando promissoras, com destaque para a primeira que está cada vez mais presente nas pesquisas (incluindo em trabalhos que não foram selecionados para este estudo).

4. Conclusão

Neste estudo, verificamos que a comparação entre códigos é a abordagem mais presente nas ferramentas analisadas e que elas apresentam resultados mais consistentes de seu uso em relação à sua aplicação no contexto de sala de aula. Verificamos também que a abordagem de testes automáticos é bastante adotada para prover *feedback* ao aluno. No entanto, não é trivial definir casos de testes para todos os cenários possíveis. Assim sendo, essa técnica tem sido complementada com outras para prover *feedback* com maior confiança. Um exemplo é o trabalho E02, que combina diferentes técnicas para prover *feedback* com mais qualidade.

As técnicas de síntese de programas e PBE, vêm se mostrando promissoras, com destaque para a primeira que tem apresentando uma quantidade cada vez maior de pesquisas. Notamos que embora essas abordagens sejam populares, não encontramos uma prática comum para avaliá-las ou mensurar a aprendizagem. Notamos também que há mais esforços de pesquisa no sentido do *feedback* funcional. Contudo, também é importante analisar aspectos qualitativos dos códigos.

Nosso trabalho corrobora com outras revisões que indicam que a comparação entre códigos é uma abordagem popular e contribui apresentando outras técnicas que não foram mencionadas em trabalhos anteriores, além do foco exclusivo nessas técnicas que possibilita uma visão mais detalhada do tema. Apesar de apresentar uma visão geral sobre o tema, é esperado que este estudo auxilie a comunidade de pesquisadores da área, assim como desenvolvedores envolvidos em projetos de ferramentas desse tipo, a selecionar qual técnica usar ou se basear, considerando a aceitação e resultados das abordagens apresentadas neste trabalho. É pretendido como trabalhos futuros aprofundar esse estudo incluindo mais trabalhos e explorar outras etapas da produção de ferramentas de *feedback* automático para atividades de programação.

Bibliografia

[1] Caiza, C; Ramiro, J. M. A. (2013) Programming assignments automatic grading: Review of tools and implementations. Proc. 7th International Technology, Education and Development Conference (INTED2013). (http://oa.upm.es/25765/)

[2] Combéfis, S; Schils, A. (2016) Automatic programming error class identification with code plagiarism-based clustering. Proc. 2nd International Code Hunt Workshop on Educational Software Engineering, p.16. ACM. DOI: 10.1145/2993270.2993271.

[3] Chow, S. et al. (2017) Automated data-driven hints for computer programming students. Proc. 25th Conference on User Modeling, Adaptation and Personalization, p. 5-10. ACM. DOI: 10.1145/3099023.3099065.

[4] Gao, J; Pang, B; Lumetta, S. (2016) Automated feedback framework for introductory programming courses. Proc. 2016 ACM Conference on Innovation and Technology in Computer Science Education, p. 53–58. ACM. DOI: 10.1145/2899415.2899440.

[5] Head, A. et al. (2017) Writing reusable code feedback at scale with mixed-initiative program synthesis. In Proceedings of the Fourth, p. 89-98. ACM. DOI: 10.1145/3051457.3051467.

[6] Hossein, S; Peter, Y. (2018) Evaluation of algorithms to support novice programmer. In: Proceedings of the 10th International Conference on Education Technology and Computers, p. 383-387. ACM. DOI: 10.1145/3290511.3290529.

[7] Keuning, H. et al. (2016) Towards a systematic review of automated feedback generation for programming exercises. Proc. 2016 ACM Conference on Innovation and Technology in Computer Science Education, p. 41-46. ACM. DOI: 10.1145/2899415.2899422.

[8] Kim, D. et al. (2016) Apex: Automatic programming assignment error explanation. In: ACM SIGPLAN Notices, p. 311-327. ACM. DOI: 10.1145/2983990.2984031.

[9] Kitchenham, B. et al. (2010) Systematic literature reviews in software engineering–a tertiary study. Information and software technology 52(8): 792-805. DOI: 10.1016/j.infsof.2010.03.006.

[10] Parihar, S. et al. (2017) Automatic grading and feedback using program repair for introductory programming courses. Proc. 2017 ACM Conference on Innovation and Technology in Computer Science Education, p. 92-97. ACM. DOI: 10.1145/3059009.3059026.

[11] Rolim, R. et al. (2017) Learning syntactic program transformations from examples. Proc. 39th International Conference on Software Engineering (ICSE ’17), p. 404-415. IEEE. DOI: 10.1109/ICSE.2017.44.

[12] Wang, K. et al. (2017) Data-driven feedback generator for online programing courses. Proc. ACM Conference on Learning @ Scale, L@S ’17, p. 257-260. ACM. DOI: 10.1145/3051457.3053999.

[13] Wilcox, C. (2015) The role of automation in undergraduate computer science education. Proc. 46th ACM Technical Symp. on Computer Science Education, p. 90-95. ACM. DOI: 10.1145/2676723.2677226.

[14] Zougari, S. et al. (2016) Towards an automatic assessment system in introductory programming courses. Proc. International Conference on Electrical and Information Technologies (ICEIT), p. 496-499. IEEE. DOI: 10.1109/EITech.2016.7519649.