A Aplicabilidade de uma Ferramenta de Correção Automatizada em Exercícios de Pensamento Computacional

Heitor Ugarte Calvet da Silveira
Universidade Regional de Blumenau
Blumenau, Santa Catarina
hucs@furb.br

Mauro M. Mattos
Universidade Regional de Blumenau
Blumenau, Santa Catarina
mattos@furb.br

Luciana P. de Araújo Kohler
Universidade Regional de Blumenau
Blumenau, Santa Catarina
lpa@furb.br

Leonardo Fronza
Universidade Regional de Blumenau
Blumenau, Santa Catarina
leofronza@furb.br

Jorge Kohn
Universidade Regional de Blumenau
Blumenau, Santa Catarina
jkohn@furb.br

Guilherme Fibrantz
Universidade Regional de Blumenau
Blumenau, Santa Catarina
gfibrantz@furb.br

ABSTRACT

The computer area is elementary to many knowledge areas. In this way, the computational thinking was included in Common Base National Curriculum to teach the children since the elementary school. The Computational Thinking (CT) is associate to development of habilities that involves abstraction capabilities of a context, allow to express to a generic solution, it supports the analysis and evaluation of a solution proposed. With the CT in the schools it is elementary that the teacher understand the basic concepts about programming to teach the students and manage to correct and evaluate the exercices developed by them. In this context, many tools were developed to support the teachers in the evaluate exercices process. In this way, this paper presents a tool to set right the exercices as a automated way. The purpose of this paper is identify the impact caused by tool in relation to the manual evaluation process, besides validate its applicability in a classroom.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

1 INTRODUÇÃO

Sabe-se que a área da programação é elementar para diversas áreas de conhecimento [6]. Dessa forma, é importante que se tenha experiência com a lógica de programação, a qual está sendo introduzida em algumas escolas de ensino fundamental assim como em diversos cursos técnicos [7] [6]. Esta disciplina exige um esforço e tempo significativo do docente, pois ele necessita prestar suporte a cada estudante individualmente ao mesmo tempo em que ele precisa corrigir os exercícios desenvolvidos e apresentar uma nota ou *feedback* ao aluno. Além disso, cabe ao professor estudar as linguagens de programação que serão ensinadas e utilizadas em sala de aula, fazendo-se necessário investir ainda mais tempo para a preparação das aulas.

Existem estudos que apontam o desenvolvimento de novas ferramentas para auxiliar no ensino de habilidades de programação, algumas destas sendo plataformas lúdicas e com aspectos de gamificação [5] [8]. Algumas destas habilidades são desenvolvidas com a resolução de diversos exercícios de programação. Acontece que, em muitas destas ferramentas, não há um *feedback* adequado para os estudantes e professores. Os estudantes, à medida em que resolvem os exercícios, devem receber um *feedback* imediato sobre o exercício desenvolvido anteriormente para aprimorar gradualmente suas soluções. Além disso, o *feedback* instantâneo permite que o aluno resolva os exercícios com uma dependência menor do docente, fazendo com que o professor atenda somente demandas e dúvidas específicas, tornando tanto o tempo do aluno quanto do professor mais eficiente [4]. Em contraste a isto, estas ferramentas não permitem que o docente regule as métricas de correção dos seus exercícios [2]. Por exemplo, o docente não pode estipular o tempo de resolução do exercício ou indicar um número tolerável de erros no código. Ainda, na maioria destas ferramentas lúdicas, não é possível que o professor desenvolva exercícios e atividades personalizadas.

Dessa forma, viu-se a necessidade de desenvolver uma ferramenta de correção automatizada personalizada que auxilia o docente no processo de avaliação de exercícios de lógica de programação. Com isso, realizou-se testes para identificar o impacto causado pela ferramenta desenvolvida em relação ao processo manual de correção. Ainda, outro objetivo deste artigo é validar a aplicabilidade da ferramenta e, assim, identificar se a ferramenta atende aos docentes na correção dos exercícios desenvolvidos pelos seus alunos.

Para otimizar a pesquisa em prol do segundo objetivo deste artigo, a revisão sistemática desenvolvida por Caiza e Álamo Ramiro foi utilizada como base [2], através da qual é identificada a ausência da possibilidade do docente estipular as métricas de avaliação de seu exercício.

Portanto, este artigo está organizado da seguinte forma. A seção 2 apresenta e discute a existência de ferramentas de correção automatizada que foram utilizadas anteriormente a esta pesquisa. A seção 3 trata sobre a plataforma utilizada para realizar os exercícios, assim como a ferramenta desenvolvida para a correção automática. A seção 4 descreve a metodologia utilizada nesta pesquisa, mostrando os participantes, procedimentos e as métricas consideradas. Por fim, a seção 5 demonstra os resultados obtidos com este estudo, discutindo os objetivos atingidos e os possíveis trabalhos futuros.

2 FERRAMENTAS DE CORREÇÃO AUTOMATIZADA

A correção automática de programas foi referenciada pela primeira vez por Fortythe [3] em 1965. Sabe-se que mesmo com o tempo e os estudos feitos, ainda existe uma carência de tecnologias para correção automatizada de exercícios de programação, principalmente para iniciantes.

A pesquisa feita por Caiza e Álamo Ramiro sobre o estado da arte resultou em onze ferramentas de correção e concluíram que existe ainda uma carência de ferramentas que permitam que os docentes estipulem as métricas que consideram importantes no

processo de avaliação de seus exercícios à fim de proporcionar uma nota justa. De acordo com Caiza e Álamo Ramiro [2], os estudos de 2010 mostram que a forma principal de correção é avaliar se o código-fonte produzido é idêntico a algum gabarito. Em outro ponto exibido por eles, o *feedback* proporcionado ao estudante pode desengatilhar a prática de tentativa e erro, sendo que este pode ser um ponto negativo das ferramentas. Uma das sugestões é que o *feedback* apresentado demonstre o que deve ser corrigido e como, no momento em que o erro acontecer.

Em termos de medidas de comparação, entre as ferramentas analisadas os autores usaram os seguintes itens: linguagem de programação suportada pela ferramenta de correção; linguagem de programação para desenvolver a ferramenta; arquitetura lógica; arquitetura de implantação; modo de trabalho (*plugin* ou *stand-alone*); métricas de correção disponibilizadas pela ferramenta e tecnologias utilizadas por ela.

O número de linguagens de programação suportadas pela ferramenta de correção indica se é possível utilizá-la em diferentes contextos de programação. A linguagem utilizada para desenvolver a ferramenta, assim como a arquitetura lógica indicam a facilidade com que se fornece suporte à ferramenta, ou seja, a necessidade de manutenção. A arquitetura de implantação indica a capacidade de hardware necessária para executar a ferramenta. O modo de trabalho é utilizado para indicar se a ferramenta pode ou não trabalhar sozinha ou se é um *plugin*. Em relação às métricas, elas identificam quando e como a ferramenta estipula uma nota ou como ela classifica um problema como certo e errado. Por fim, as tecnologias utilizadas são utilizadas quando se é considerado implementar ou desenvolver uma nova ferramenta, de forma que os padrões de compatibilidade possam ser mantidos.

Este estudo foi utilizado como base para esta pesquisa e para o desenvolvimento da ferramenta que automatiza o processo de correção, com foco em exercícios introdutórios de programação. Os exercícios corrigidos pela ferramenta desenvolvida, por serem introdutórios, devem ser escritos em um único arquivo de código-fonte.

3 A FERRAMENTA DE CORREÇÃO AUTOMATIZADA

Diante da necessidade de prover ferramentas de autocorreção com flexibilidade e que podem ser utilizadas como guia para o docente em sala de aula, excluindo a necessidade do docente ter grande conhecimento sobre programação, uma nova ferramenta foi desenvolvida. Esta ferramenta será apresentada nas próximas subseções.

3.1 Informações Gerais da Ferramenta

Esta ferramenta apresenta diversos *feedbacks* além da correção do exercício. É necessário que o documento que contém o código-fonte do aluno seja comparado com o gabarito que contém os seguintes dados: a hora de início da resolução, a hora do término, quantidade de erros de compilação e a quantidade de erros em tempo de execução. Ao passo em que a ferramenta compara os arquivos, essas informações devem estar presentes no gabarito para que seja feita a correção. Dessa forma, a ferramenta utilizada para resolver os exercícios deve providenciar estas informações no arquivo que será utilizado para comparação com o gabarito para que possam ser mensuradas. Caso a plataforma utilizada não apresente estas informações, a ferramenta de correção automatizada irá somente apresentar as métricas relacionadas à correção do código escrito.

A ferramenta de correção automatizada permite a análise de mais de um exercício simultaneamente, corrigindo um ou mais alunos. Neste caso, é necessário carregar todos os arquivos contendo os exercícios dos alunos com o padrão de título "nomeDoAluno-nomeDoExercicio" e todos os possíveis gabaritos no padrão "nomeDo Exercicio-numero". Dessa forma, o analisador irá verificar através do nome do arquivo quais gabaritos devem ser utilizados na comparação. Ainda, é possível fazer a correção dos exercícios de todos os alunos simultaneamente. De forma geral, a correção dos exercícios pela ferramenta não demora mais que 5 segundos, sendo um tempo bem eficiente em relação a uma correção manual.

A ferramenta de correção automatizada não apresenta uma nota final, sendo esta uma responsabilidade do docente indicar uma nota baseada nas métricas providenciadas pela ferramenta. Portanto, embora esta ferramenta não permita que o professor personalize as métricas, eles podem definir quais delas levar em consideração para estipular a nota final.

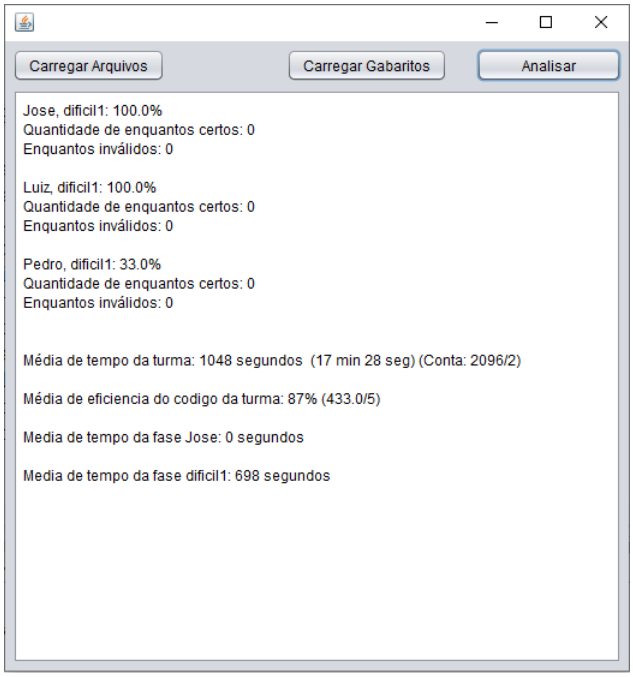

Figure 1: Interface da ferramenta

A Figura 1 apresenta a interface da ferramenta com um exemplo de correção. Pode-se notar que a interface é simples, sendo composta de uma área de texto e três botões. O primeiro botão "Carregar Arquivos" abre um diálogo para carregar os arquivos dos alunos a serem corrigidos, permitindo carregar quantos arquivos o professor desejar. Nesse caso, foram carregados três arquivos, sendo dos alunos: "Jose", "Luiz" e "Pedro". Em seguida, o botão "Carregar Gabaritos", permite carregar um ou mais gabaritos, conforme a necessidade do exercício. Neste caso, carregou-se somente um

gabarito. O botão "Analisar" faz a análise dos exercícios, comparando cada um dos arquivos carregados com o gabarito em questão e então, apresenta o resultado na área de texto.

Com essas informações, o professor pode analisar e fazer sua própria avaliação a fim de fornecer a nota para cada um dos alunos.

3.2 Classificação da ferramenta

Para comparar a ferramenta desenvolvida com as ferramentas estudadas por Caiza e Álamo Ramiro [2], esta ferramenta foi classificada com as mesmas métricas sugeridas pelos autores. Desta forma:

- linguagens de programação suportadas: esta ferramenta realiza correção sobre qualquer linguagem de programação, desde que o código-fonte fornecido para a correção possua comandos restritos (sem o uso de variáveis ou controles que tornam a geração de gabaritos muito trabalhosa) seja submetido para compara-lo à solução dos alunos;
- linguagem de programação utilizada para desenvolver a ferramenta: a linguagem Java foi utilizada por ser *open source* e multiplataforma;
- arquitetura lógica: a ferramenta não é modular, sendo difícil para adicionar novas funcionalidades ou flexibilizar as métricas de correção no momento. Entretanto, a comparação dos exercícios é feita utilizando documentos de texto, permitindo a correção de exercícios sobre qualquer linguagem;
- arquitetura de implantação: é somente necessário um computador com um sistema operacional suportado pelo Java, sendo independente de conexão à internet;
- modo de trabalho: independente, isto é, não é um *plugin* para outra ferramenta de desenvolvimento. Por isso, os exercícios a serem corrigidos, assim como o gabarito podem ser carregados pela ferramenta através de um seletor de arquivos. Sabe-se que em linguagens de programação pode ter mais de uma solução correta para um problema. Por isto, a ferramenta permite mais de um gabarito para comparação;
- métricas: a ferramenta disponibiliza, como resultado da análise automatizada, métricas individuais e médias da classe. Como métricas individuais apresenta-se: a porcentagem de acertos considerando todas as soluções submetidas; o tempo de resolução do exercício; a quantidade de laços de repetição utilizados de forma correta ou incorreta (em cenários em que isto é possível). É considerado um laço de repetição errôneo aquele em que o comando não está presente em nenhum gabarito. Como média de turma apresenta-se: a média geral do tempo de resolução de todos os exercícios; a média da eficiência do código-fonte por exercício (isto é, a média de respostas corretas na classe); e a média de tempo utilizado para desenvolver cada exercício da classe;
- tecnologias utilizadas: somente a Java Virtual Machine.

3.3 Cálculo das métricas disponibilizadas pela ferramenta

Para calcular as métricas providas pela ferramenta de correção automatizada, alguns cálculos básicos foram feitos. Estes serão descritos em seguida.

Para a primeira métrica, a porcentagem de acertos no exercício, a ferramenta usa como base o gabarito carregado para aquele exercício e o exercício a ser corrigido. Para a correção, o algoritmo caminha passo a passo as linhas do código do primeiro gabarito e faz a comparação com o código do exercício desenvolvido pelo aluno. Para cada linha do algoritmo desenvolvido pelo aluno que for igual ao gabarito em questão, é adicionado um em uma variável. Ao final, tem-se como resultado o total de linhas corretas produzidas pelo aluno. No caso de existir mais de um gabarito, o processo é executado novamente. Ao final, o algoritmo compara qual dos gabaritos resultou em uma maior quantidade de linhas corretas. Dessa forma, tem-se a quantidade de linhas corretas e o total de linhas que o aluno deveria ter desenvolvido, representado por `correctLines` e `studentLines` respectivamente na equação 1. Nessa quantidade total de linhas desenvolvidas pelo aluno, também pode-se identificar as linhas com erros de compilação (caso o algoritmo providenciar) ou simplesmente linhas diferentes da resolução correta. Após a contagem é aplicada uma regra de três de acordo com a equação 1 para se obter o percentual de linhas de código-fonte corretas.

$$\alpha = \frac{correctLines * 100}{studentLines} \tag{1}$$

Em relação ao tempo para a resolução dos exercícios é feita a diferença entre o tempo de início e o tempo final. Lembra-se que estas informações de horários devem ser providenciadas pelo arquivo que contém a solução do código-fonte do aluno.

Para que a quantidade de laços de repetição seja calculada como válida ou inválida, no momento em que o código-fonte do gabarito é processado, caso seja encontrado um laço de repetição e o mesmo não foi contabilizado, o comando é adicionado em um vetor de "laços corretos". O mesmo acontece ao avaliar o código-fonte do aluno. Caso encontrar um laço de repetição e o mesmo não foi contabilizado ainda, adiciona-se o comando em um vetor de "laços do aluno". Em seguida, os dois vetores são comparados. Considera-se que se os vetores forem iguais, o aluno desenvolveu todos os laços de repetição de forma correta. Caso haja alguma divergência (para mais ou para menos), ou a condição do laço utilizado pelo aluno for outra, o algoritmo de comparação adiciona o laço como um laço utilizado de forma incorreta. Lembra-se que para esta validação, todos os gabaritos são considerados.

Para a média da turma, uma média aritmética simples é realizada, como representada na equação 2. Na equação, soma-se a média de tempo geral da resolução dos exercícios, sendo que `x` representa o tempo total da resolução de cada um dos exercícios corrigidos (podendo conter a resolução de mais de um exercício e mais de um estudante) e `n` representa a quantidade total de exercícios corrigidos.

$$\mu = \frac{\sum^{x}}{\eta} \tag{2}$$

Após isso, para a média da eficiência do código-fonte por exercício, tem-se que `x` representa o percentual de respostas corretas para cada aluno por exercício e `n` representa o total de estudantes. Nesse caso, é considerado o nome do arquivo para apresentar a média da eficiência por exercício carregado na ferramenta.

4 METODOLOGIA DE PESQUISA

Para estruturar a pesquisa realizada com o objetivo de validar a ferramenta de correção automatizada, a metodologia de pesquisa foi dividida em participantes, procedimentos e métricas, conforme apresentado nas subseções seguintes.

4.1 Participantes

Os participantes da pesquisa foram 130 alunos do ensino fundamental (anos iniciais), entre 7 e 12 anos de idade. Esses alunos utilizaram a plataforma gamificada durante o período de 8 meses com o propósito de desenvolver seu raciocínio lógico, utilizando técnicas do pensamento computacional. Os estudos relacionados à aplicação desta plataforma gamificada durante esse período estão relacionados [1].

Assim, para testar a ferramenta de correção automatizada, foram utilizados os códigos elaborados por esses alunos, pois já tinham contato com a ferramenta há 8 meses, sendo que já estavam familiarizados com os exercícios introdutórios de programação.

4.2 Procedimentos

Para permitir o teste da ferramenta, foi feita uma alteração na plataforma de gamificação. Essa plataforma existe há mais de 10 anos e foi usada em um projeto de extensão na Universidade com crianças do ensino fundamental nos últimos dois anos [1]. Essa plataforma fornece ao aluno um conjunto de ferramentas de exercícios introdutórios que devem ser respondidos com setas do teclado ou comandos de programação (andarAcima(), andarAbaixo(), andarDireita() e andarEsquerda()), além do *loop while* com algumas condições, de acordo com o nível em que o aluno está [1]. A modificação realizada foi relacionada ao fornecimento de um arquivo de texto com o código-fonte desenvolvido pelo aluno ao final de cada fase, além das informações necessárias para o funcionamento total da ferramenta de correção automatizada.

Assim, para que o teste fosse aplicado com dados reais, foi realizada uma atividade de competição no último dia de aula com os participantes envolvidos. Esta atividade teve como objetivo verificar quantos exercícios cada um dos estudantes realizou em um determinado período de tempo, validando também a quantidade de erros de compilação ou execução que ocorreram até a solução final. Lembra-se que a plataforma gamificada permite ao aluno passar para o próximo exercício somente quando ele tiver concluído o exercício atual. Dessa forma, a própria plataforma fornece um *feedback* sobre os erros cometidos pelo aluno. No entanto, cada um dos erros que ocorreram, compilação ou execução, são salvos para que o professor tenha acesso ao avaliar seu desempenho.

Foram fornecidos 40 minutos para que cada aluno pudesse realizar até nove exercícios de programação. Os nove exercícios foram divididos igualmente entre os níveis fácil, médio e difícil. Lembra-se que no nível fácil o aluno podia utilizar apenas setas do teclado. No nível médio, o aluno poderia usar apenas os comandos de programação para caminhar em alguma direção sequencialmente. E, no nível difícil, o aluno também poderia usar os *loops* com condições, mas sem obrigatoriedade.

Para cada um desses exercícios, um ou mais gabaritos foram gerados. Os gabaritos serviriam posteriormente para que a ferramenta pudesse comparar a solução do aluno com a solução correta. Em alguns casos, mais de um gabarito foi fornecido devido ao fato de que mais de uma solução correta seria possível. Além disso, para cada exercício, foi indicado o número de *loops* possíveis com o objetivo de verificar se o aluno os utilizou ou não da forma correta para otimizar sua solução.

Os resultados dessa atividade são apresentados em outro estudo, mas os dados coletados com esses exercícios foram utilizados para o teste do instrumento de correção automatizada.

Tabela 1: Avaliação dos exercícios de uma amostra de 5 alunos

Métricas individuais					
Aluno	**Exercício**	**% de acerto**	**Laços +**	**Laços -**	**T (s)**
Anna	Difícil 1	88%	2	2	220
Gabriel	Difícil 1	77%	1	1	250
Maria	Difícil 1	100%	2	0	150
Matheus	Difícil 1	100%	2	0	120
Peter	Difícil 1	95%	2	0	230
Métricas da turma					
Média do tempo geral (s)					194
Média da eficiência dos exercícios Difícil 1 (%)					92%
Média de tempo dos exercícios Difícil 1 (s)					194

4.3 Métricas

Para analisar o primeiro objetivo de identificar o impacto causado pela relação da ferramenta de correção automatizada com o processo de avaliação manual, foi utilizada a ferramenta para avaliar os exercícios. Como medidas de avaliação para enfrentar o segundo objetivo, a ferramenta de correção automatizada fornece medidas individuais e medidas de classe, conforme relacionado na seção 3, item de medidas.

5 RESULTADOS E DISCUSSÕES

Após a aplicação do método de pesquisa conforme especificado na seção anterior, foi utilizada a ferramenta de correção automatizada para a correção dos exercícios realizados pelos alunos. Como resultados, obteve-se para cada um dos nove exercícios dos 130 alunos as métricas informadas, totalizando 1170 exercícios corrigidos. A tabela 1 demonstra um exemplo de resultado obtido com a correção de apenas um exercício de uma amostra de 5 alunos usando a ferramenta. Por outro lado, a tabela 2 demonstra um exemplo de resultado obtido ao corrigir 3 exercícios de uma amostra de 3 alunos usando a ferramenta. Utilizou-se amostras menores para representar as medidas dos alunos e da turma, porque se a amostra de 140 alunos fosse usada, a tabela seria muito extensa para este artigo. Além disso, os nomes dos alunos foram modificados para manter sua privacidade. Também, para as duas tabelas, foram utilizados os exercícios que permitem *loops*. Pode-se verificar que existem métricas individuais e métricas da turma em cada uma das tabelas.

Na Tabela 1, pode ser observada a avaliação do exercício denominado “Difícil1” de cinco alunos diferentes . Para cada um desses alunos, a ferramenta fornece as métricas individuais. Em seguida, na parte inferior da tabela, há as métricas da turma que consideram os exercícios a serem avaliados naquele momento. Como houve

Tabela 2: Avaliação de três exercícios de uma amostra de três alunos

Métricas individuais					
Aluno	**Exercício**	**% de acerti**	**Laços +**	**Laços -**	**T (s)**
Anna	Fácil1	70%	-	-	80
Gabriel	Fácil1	90%	-	-	150
Maria	Fácil1	85%	-	-	120
Anna	Médio1	100%	-	-	200
Gabriel	Médio1	95%	-	-	180
Maria	Médio1	66%	-	-	210
Anna	Difícil1	88%	2	1	220
Gabriel	Difícil1	77%	1	1	250
Maria	Difícil1	100%	2	0	150
Métricas da turma					
Média do tempo geral (s)				173.77	
Média da eficiência dos exercícios Fácil1 (s)				81.66%	
Média da eficiência dos exercícios Médio1 (s)				87%	
Média da eficiência dos exercícios Difícil1 (s)				88.33%	
Média de tempo dos exercícios Fácil1(s)				116.66	
Média de tempo dos exercícios Médio1 (s)				196.66	
Média de tempo dos exercícios Difícil1				206.66	

apenas um exercício avaliado, a média de tempo geral é a mesma que a média de exercício.

Na Tabela 2, pode-se observar a avaliação dos exercícios denominados "Fácil1", "Médio1" e "Difícil1" de três alunos diferentes. Para cada aluno e cada exercício, a ferramenta possui suas métricas individuais. Em seguida, na parte inferior da tabela, há as métricas da turma que consideram apenas os exercícios a serem avaliados naquele momento. Nesta tabela, há a eficiência e o tempo médio para cada um dos três exercícios avaliados. Todo o cálculo foi feito com base na especificação da subseção 3.3.

Após a análise gerada pela ferramenta de correção automatizada, percebeu-se que a ferramenta fornece recursos para que o professor possa avaliar a turma como um todo, além de cada aluno por exercício. Com esses dados, é possível que o professor forneça *feedback* ou uma nota com base nas medidas desejadas por ele.

Assim, em relação ao primeiro objetivo deste artigo, que é identificar o impacto causado pela ferramenta de correção automatizada em relação ao processo de avaliação manual, pode-se afirmar que a ferramenta de correção automatizada torna o processo de avaliação e a análise do código-fonte mais simples, bem como para professores que não tem habilidades em programação. Ainda, é mais eficiente corrigir com a ferramenta, visto que levam segundos para apresentar os resultados com as métricas de cada exercício. Isso pode ser afirmado porque após a atividade de competição e o uso da ferramenta, foi identificado que a ferramenta fornece a partir do modelo a porcentagem correta e outras métricas indicadas ao longo deste artigo. A partir dessas informações, torna-se possível para o professor avaliar e dar uma nota ou *feedback* ao aluno, pois ele terá as informações básicas para isso.

Com relação ao segundo objetivo deste artigo de verificar a aplicabilidade da ferramenta, identifica-se que a ferramenta é aplicável para apoiar o professor no processo de avaliação de exercícios de seus alunos. Isso pode ser afirmado com base nas pesquisas feitas por Caiza e Álamo Ramiro [2] e nas lacunas encontradas. No entanto, a ferramenta deixa lacunas no recurso de personalização executado pelo professor, uma vez que as métricas não podem ser selecionadas de acordo com o que o professor deseja ao definir a nota. Por outro lado, como a ferramenta não fornece a nota final, o professor pode usar as medidas desejadas para executar essa ação. Outro ponto a ser aprimorado é a arquitetura da ferramenta, a fim de permitir uma maneira mais simples na modularização da mesma.

Desse modo, deixa-se registrado neste artigo que, em relação ao *feedback* excessivo proporcionado ao aluno que algumas ferramentas de correção automatizada fornecem e podem levar a este a prática por tentativa e erro, a ferramenta de correção automatizada não se aplica, pois é utilizada pelo professor e não pelo aluno que está respondendo ao seu exercício. Portanto, essa é uma análise que deve ser feita pela ferramenta a ser utilizada para a resolução do exercício. Contudo, como a ferramenta de correção automatizada considera o código enviado a ela para a comparação com o gabarito, o certo é enviar o código contendo as tentativas de erro para que a ferramenta de correção automatizada as considere. Caso este código não seja enviado, a ferramenta não terá como identificar as ações de tentativa e erro realizadas pelo usuário.

Como trabalhos futuros, espera-se melhorar a ferramenta de correção automatizada para que seja possível utilizá-la em problemas mais complexos com mais de um arquivo de código-fonte, além de apresentar outras métricas para o professor. Por fim, espera-se adicionar um recurso para fornecer a nota final do exercício de modo que o professor configure as métricas e a fórmula para o cálculo da nota.

REFERENCES

[1] bindreview. 2018. blindreview. (2018).
[2] Julio C. Caiza and José María del Álamo Ramiro. 2013. Programming assignments automatic grading: review of tools and implementations. In *7th International Technology, Education and Development Conference (INTED2013)*. 5691–5700. http://oa.upm.es/25765/
[3] George E. Forsythe and Niklaus Wirth. 1965. Automatic Grading Programs. *Commun. ACM* 8, 5 (May 1965), 275–278. https://doi.org/10.1145/364914.364937
[4] Colin A. Higgins, Geoffrey Gray, Pavlos Symeonidis, and Athanasios Tsintsifas. 2005. Automated Assessment and Experiences of Teaching Programming. *J. Educ. Resour. Comput.* 5, 3, Article 5 (Sept. 2005). https://doi.org/10.1145/1163405.1163410
[5] S. A. Nikou and A. A. Economides. 2014. Measuring Student Motivation during "The Hour of CodeTM" Activities. In *2014 IEEE 14th International Conference on Advanced Learning Technologies*. 744–745. https://doi.org/10.1109/ICALT.2014.218
[6] Peter Rich and Charles Hodges. 2017. *Emerging Research, Practice, and Policy on Computational Thinking (Educational Communications and Technology: Issues and Innovations)*. Springer. https://www.amazon.com/Emerging-Computational-Educational-Communications-Technology-ebook/dp/B071H7MF21?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B071H7MF21
[7] SBC. 2018. Ensino de Computação na Educação Básica. In *Diretrizes para ensino de Computação na Educação Básica*, SBC (Ed.). 1–20.
[8] Danny Yaroslavski. 2014. How does Lightbot teach programming?. In *LightBot*. 1–5.