# Juízes Online são suficientes ou precisamos de um *VAR*?

Alexandre de A. Barbosa, Evandro de Barros Costa, Patrick Henrique Brito
alexandre146@gmail.com,ebc.academico@gmail.com,patrick@ic.ufal.br
Universidade Federal de Alagoas (UFAL)

## RESUMO

Tradicionalmente em disciplinas introdutórias de programação, são adotadas atividades práticas de codificação. Nesses exercícios, o professor seleciona um conjunto de problemas e os discentes devem submeter algoritmos/códigos como solução. O professor deve avaliar as soluções propostas fornecendo *feedback* aos discentes. Uma estratégia amplamente utilizada para agilizar a avaliação das soluções são os juízes on-line. Um juiz on-line indica se um código é correto, ou não, com base em casos de teste. Nesta pesquisa, foi realizado um levantamento com o objetivo de investigar quais processos de avaliação são empregados em disciplinas introdutórias de programação, sendo observadas quais ações são realizadas e quais critérios são considerados por professores e monitores. Tais indagações visam responder a questão "juízes on-line são uma abordagem suficiente para o ensino e a aprendizagem de programação ou é necessário a adoção de um '*VAR*'?". A análise realizada sobre os dados sugere que juízes on-line são extremamente úteis para avaliação de códigos, mas não são suficientes para contemplar todos os critérios empregados por professores e monitores nas disciplinas.

## PALAVRAS-CHAVE

Ensino de programação, Juízes on-line, Avaliação

## 1 INTRODUÇÃO

Na área de Computação a habilidade de programar é uma das competências fundamentais. Esse conhecimento é basilar para a compreensão de diversos outros conceitos necessários tanto para a vida acadêmica, quanto para as atividades profissionais. Os períodos iniciais dos cursos na área de Computação, em geral, englobam diferentes disciplinas que possuem como foco o estudo de conceitos de algoritmos e a implementação de programas.

Apesar de tal importância, é bastante comum encontrar problemas em ambientes de ensino e da aprendizagem de programação. Nesse contexto, é frequente encontrar estudantes desmotivados e que não conseguem entender os conceitos necessários para a prática de programação. Muitos são os fatores descritos na literatura como relacionados ao cenário de dificuldades nas disciplinas de programação. Entre os diversos motivos relacionados ao cenário descrito, e que são apresentados na literatura [3, 8, 17, 19, 25], alguns dos mais comumente citados são (i) a dificuldade em relação ao entendimento de conceitos abstratos, (ii) os problemas de interpretação do que é solicitado nos enunciados dos exercícios, (iii) a falta de aplicação dos conceitos provenientes da matemática, (iv) o atraso em relação ao *feedback* e (v) as complexidades inerentes ao uso de uma linguagem de programação, tal como sua sintaxe e semântica associada.

Tradicionalmente em disciplinas de programação, nas avaliações e exercícios são adotadas atividades práticas de codificação. Nessas atividades o professor seleciona um conjunto de problemas (exercícios) e, para cada um destes, os discentes devem submeter códigos (programas) como solução para cada problema.

Após a submissão das soluções propostas pelos estudantes, é necessário proceder com a avaliação destas. Como descreve Lima et al. [18], "*No ensino de programação, é comum o uso de Ambientes de Correção Automática de Código (ACACs). Esses apresentam uma alta diversidade de exercícios de programação que requerem que o estudante elabore um código como solução.*".

Nestes ambientes de avaliação automática, uma estratégia amplamente utilizada para facilitar e agilizar as atividades de avaliação de código é o uso de juízes on-line. Um juiz on-line consiste em uma aplicação que indica se um código é correto, ou não, com base em um conjunto de casos de teste [21]. Um caso de teste corresponde a especificação de um conjunto de entradas e as respectivas saídas esperadas [7].

Dessa forma, se um código passa pelo conjunto de teste, ele é aceito como uma solução correta, em outro caso, o código é dito incorreto. Essa abordagem, contudo, não avalia outros critérios de qualidade ou pedagógicos. Um juiz on-line considera apenas se uma solução fornece o conjunto de saídas esperado para um conjunto de entrada fornecido. Uma vez fornecida a saída esperada, ela é considerada totalmente correta.

Para ilustrar a limitação existente na abordagem de juízes on-line, dado o enunciado de um problema de programação "Forneça o valor do fatorial para um número inteiro dado pelo usuário", e sejam observados os códigos exibidos em 1 e 2 como soluções propostas por discentes.

**Código Fonte 1: Solução 1 para o problema "Forneça o valor do fatorial para um número inteiro dado pelo usuário".**

```
1 import math
2 print(math.factorial(int(input("Digite um inteiro: "))))
```


*EduComp'23, Abril 24-29, 2023, Recife, Pernambuco, Brasil (Online)*

**Código Fonte 2: Solução 2 para o problema "Forneça o valor do fatorial para um número inteiro dado pelo usuário".**

```
1 def fatorial(n) :
2         fat = 1
3         for i in range(1, n+1) :
4                 fat = fat * i
5         return fat
6
7 n = int(input("Digite um inteiro:"))
8 print(fatorial(n))
```

As soluções exibidas no Código Fonte 1 e no Código Fonte 2 seriam avaliadas como totalmente corretas por um juiz on-line. Para o Código Fonte 2 pode-se argumentar que o discente empregou instruções de repetição e desenvolveu uma lógica para solução do problema proposto, enquanto no Código Fonte 1 a proposta de solução apenas realizada uma chamada a uma função da linguagem. Do ponto de vista pedagógico, estes código poderiam ser avaliados de forma distinta, dependendo do conteúdo trabalhado e dos critérios do avaliador. Sendo assim, pode ser necessário observar as soluções, mesmo quando essas são avaliadas como totalmente corretas por um juiz on-line.

No futebol, o juiz de campo, ou árbitro principal, toma decisões em relação a aplicação das regras do esporte. Nos últimos anos, o árbitro de vídeo ou *Video assistant referee* (*VAR*) surgiu como função auxiliar ao juiz de campo, de modo a analisar as decisões tomadas pelo árbitro principal com a utilização de imagens de vídeo. O uso de novos recursos de análise permite que erros sejam corrigidos ao longo de uma partida.

De forma similar ao "VAR" do futebol, a avaliação de um juiz on-line no contexto de programação, que seria equivalente à decisão do juiz de campo no futebol, pode necessitar de uma checagem através de outros meios, tal como a análises realizada pelo "VAR" no futebol.

Nesta pesquisa, buscou-se, como objetivo geral, investigar os processos de avaliação que vem sendo empregados por professores e monitores de disciplinas introdutórias de programação. Para estes processos, é observado se os juízes on-line são utilizados e se estes são suficientes para automatizar as avaliações de acordo com as ações e critérios considerados nestes processos, ou se é necessária a adoção de mecanismos auxiliares, um '*VAR*' para o ensino e a aprendizagem de programação.

O restante desta pesquisa está organizada da seguinte forma, na próxima seção são descritos os conceitos necessários para a compreensão da pesquisa. Na Seção 3 são apresentados os trabalhos que tratam do tema de avaliação em disciplinas introdutórias de programação. Na Seção 4 é apresentado um levantamento relacionado à avaliação de códigos em disciplinas de programação, também são exibidos e discutidos os resultados obtidos. Finalmente, na última seção são apresentadas as conclusões e trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, apresenta-se a fundamentação teórica que é dividida em duas partes, a descrição do qeu são os juízes on-line, apresentada a seguir e a descrição do que pode ser considerado como "VAR" no ensino de programação, exibido em 2.2.

## 2.1 Juízes on-line

Em [28], um juiz on-line é definido como um sistema on-line que executa ao menos um dos seguintes passos de avaliação: (a) coleta e compila/interpreta os códigos; (b) avalia soluções com base em um conjunto de testes; e (c) computa um *score* de avaliação.

Um juiz on-line utiliza uma abordagem baseada em um conjunto de casos de teste, para indicar se um código é correto. Um caso de teste corresponde à especificação de conjunto de entradas e as respectivas saídas esperadas [7]. Sendo assim, as aplicações do tipo juiz on-line consideram corretos os códigos que fornecem as saídas esperadas para cada entrada fornecida, em outro caso o código é dito incorreto.

A abordagem utilizada por juízes on-line não avalia critérios pedagógicos ou critérios de qualidade em um código, em geral, se uma solução fornece o conjunto de saídas esperado ela é considerada totalmente correta. Desta forma, diversas iniciativas buscam aprimorar a abordagem de juízes on-line.

Alguns exemplos de aprimoramentos propostos no contexto de juízes on-line são as aplicações que utilizam meios para julgar se um código é parcialmente correto, sendo empregado para isso o percentual de aprovação no conjunto de testes, tais como [20, 21]; e as propostas que complementam a indicação de sucesso, ou falha, fornecida pelos testes com dicas que auxiliem o aluno a identificar e corrigir os erros, como exemplo o ambiente *The Huxley* [21] e a ferramenta *feeper* [11]. As dicas apresentadas são associadas a um caso de teste com base na experiência do testador/professor em determinar a provável causa da falha.

## 2.2 "VAR" em ensino de programação

Diversos autores, tais como [5, 9, 12, 27, 28], apontam que a avaliação de códigos em disciplinas de programação exige muito tempo, dedicação e é uma ação sujeita à erros por parte do avaliador.

Dessa forma, muitas pesquisas, tais como [1, 13, 16, 23, 24], têm sido desenvolvidas com o intuito de propor métodos, abordagens ou ferramentas para facilitar o acompanhamento das atividades em disciplinas de programação. Conforme afirmam Porfirio, Pereira e Maschio [12] onde descrevem que a pesquisa e desenvolvimento de ferramentas de apoio para o ensino de programação é um tópico bastante difundido. O mapeamento sistemático apresentado em [4] corrobora com a afirmação anterior.

A sigla "VAR" do futebol é utilizada como referência aos auxiliares ao juiz principal, no ensino de programação não há uma categoria de ferramentas que esteja associada à sigla "VAR'. A referência ao termo "VAR" ocorre nesta pesquisa com o intuito de conduzir a comunidade de docentes e pesquisadores em Computação à reflexão sobre as práticas de avaliação adotadas.

Como citado anteriormente muitas pesquisas sobre avaliação já foram conduzidas. Contudo, os resultados apontados em mapeamentos e levantamentos anteriores [4, 12, 28], indicam que o foco da avaliação automática é em aspectos de corretude em relação às saídas esperadas. Por outro lado a grande quantidade de trabalhos relacionados ao tema de ensino de programação demonstra a preocupação da comunidade na busca de soluções de avaliação.

## 3 TRABALHOS RELACIONADOS

A principal estratégia identificada na literatura como meio para facilitar e agilizar a avaliação de códigos é o uso de juízes on-line [4, 12, 28]. Entre os aspectos analisados em [12] aqueles suportados por juízes on-line representam 66,67% dos trabalhos observados, sendo 42,31% sobre Corretude Funcional, 10,90% sobre erros de compilação, 7,79% erros sintáticos e 5,77% tratam de erros de execução.

Uma grande quantidade de trabalhos sobre juízes on-line pode ser encontrada na literatura. Alguns exemplos de ferramentas podem ser citados, o sistema BOCA (*BOCA Online Contest Administrator*) [6] que é utilizado na Maratona de Programação da SBC, o ambiente *The Huxley* [21], a ferramenta *feeper* [11], o software *URI Online Judge Blocks* [10] e *CodeBench* [9] são exemplos de trabalhos relacionados com propostas de sistemas juízes on-line. Existem ainda muitas outras ferramentas nesta categoria.

Outros trabalhos onde são analisadas a experiência e a aceitação dos estudantes em relação aos juízes on-line foram desenvolvidos.

Em [26] foram analisadas, entre outros aspectos, a frequência de submissões e o grau de similaridade entre as soluções propostas. O autor descreve que os estudantes realizam uma grande quantidade de submissões, o que é positivo pois a quantidade de exercícios realizado é um fator descrito como muito importante para o aprendizado. Contudo, um problema associado é o fato de que as submissões realizadas pelos discentes possuem um garu de médio à alto em relação as similaridades dos códigos, o que pode indicar plágio em relação às respostas.

Os autores de [2] analisam a aceitação dos discentes em relação ao uso de juízes on-line. Após coletarem dados a partir de informações fornecidas por 187 estudantes de graduação, alguns dos resultados positivos reportados são a melhoria na motivação, uma vez que os discentes passaram a utilizar o sistema com alta frequência; autoeficácia, pois os estudantes se tornaram mais independentes em relação a organização de suas sessões de estudo; e a influência social, pois os discentes ao observarem o desempenho de seus colegas tiveram seu comportamento influenciado. Um impacto negativo apontado pela pesquisa foi o aumento da ansiedade entre os discentes.

A pesquisa apresentada em [15] engloba dados de 162 estudantes de primeiro período através do uso de questionários. Os autores descrevem que os discentes concordaram que a ferramenta permite um *feedback* rápido e que é útil para eles realizarem correções em seus códigos. Muitos discentes descreveram que corrigiam suas soluções até que obtivessem um *feedback* de aceitação na ferramenta.

No trabalho de Hidalgo et al. [14] é descrito que juízes on-line são ferramentas úteis nso contextos de ensino, em competições de programação e como ferramentas de recrutamento. Os autores citam que a maioria dos trabalhos encontrados na literatura focam em aspectos importantes para o aprendiz e deixam de fora as avaliações e observações de docentes. Nessa pesquisa o objetivo é listar requisitos a partir da observação de diferentes ferramentas e avaliar o nível de satisfação dos docentes em relação aos requisitos identificados. Como resultado na pesquisa é reportado um baixo nível de satisfação em relação aos requisitos.

## 4 UM LEVANTAMENTO SOBRE AVALIAÇÃO EM DISCIPLINAS INTRODUTÓRIAS DE PROGRAMAÇÃO

### 4.1 Descrição dos dados e do perfil dos avaliadores

O objetivo da condução do levantamento foi compreender os processos relacionados com a avaliação de códigos em disciplinas introdutórias de programação. Dessa forma, o levantamento se relaciona com a identificação dos processos utilizados pelos avaliadores e de quais critérios são considerados por cada avaliador. Os questionários completos utilizados na condução do levantamento podem ser encontrados nos seguintes links https://forms.gle/tV7W3d12UmvvPwyg9, para os monitores e https://forms.gle/KCjACJ3LZ3ZKF8GR7, para os docentes. Foram criados e distribuídos dois tipos de questionários pois as perguntas relacionadas ao perfil do respondente eram distintas para o perfil de docente e monitor. No início de ambos os questionários foi apresentado um Termo de Consentimento Livre e Esclarecido (TCLE), também foram exibidas informações sobre a pesquisa e descrições sobre a anonimidade na divulgação dos resultados computados.

Uma vez compreendidos os processos de avaliação empregados por professores e monitores nessas disciplinas, buscou-se então verificar se juízes on-line são adequados e suficientes nesses processos.

Um total de 37 pessoas responderam ao questionário, sendo 15 professores e 22 monitores/ex-monitores de disciplinas introdutórias de programação. Os sujeitos que responderam ao questionário estavam vinculados às seguintes instituições: Instituto Federal de Alagoas (IFAL), Instituto Federal de Sergipe (IFSE), Universidade Federal de Alagoas (UFAL), Universidade Federal da Paraíba (UFPB), Universidade de São Paulo (USP), Universidade Federal de Minas Gerais (UFMG) e da Faculdade de Tecnologia (FATEC/AL).

Em relação ao nível de formação de cada categoria, 68.2% dos monitores possuía graduação concluída e 31.8% estavam com a graduação em andamento, já os professores, 66.7% possuía doutorado, 26.7% possuía mestrado e 6.7% eram graduados.

Entre as respostas fornecidas, a linguagem Python foi a mais utilizada, seja na condução da disciplina quando ministrada pelo professor, ou durante as atividades de monitoria. Entre os monitores, 65.4% desempenharam suas atividades em disciplinas onde a linguagem Python foi adotada. As linguagens Java (19,2%), C (11,5%) e C++ (3,8%) também foram utilizadas nas disciplinas de programação acompanhadas pelos monitores. Entre os professores, 37.5% ministraram as disciplinas adotando a linguagem Python em suas turmas. As linguagens C (33.,3%), Java (12.5%), Pascal (8,3%), C++ (4,2%) e Javascript (4,2%) também foram utilizadas nas disciplinas de programação ministradas pelos respondentes.

Na Figura 1 é exibido o gráfico que resume os intervalos representando quantos anos de experiência lecionando disciplinas de programação os docentes possuem.

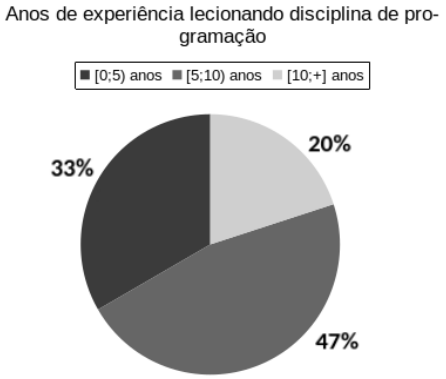


**Figura 1: Experiência (em anos) dos docentes ministrando disciplinas de programação.**

Todos os professores descreveram que foram adotadas atividades práticas nas disciplinas que ministraram, tendo a maioria adotado ferramentas de apoio (ex. ambientes de desenvolvimento integrado - *Integrated Development Environment* (*IDEs*), juízes on-line, outras) na condução da disciplina, apenas um docente informou que não adota nenhuma ferramenta da apoio. Somente um dos monitores descreveu que não foram utilizadas atividades práticas ou ferramentas de apoio na disciplina em que desempenhou as atividades de monitoria. Os demais monitores descreveram que foram adotadas atividades práticas e a maioria das disciplinas fez uso de ferramentas de apoio (ex. IDEs, juízes on-line, outras), apenas um monitor informou que não foi adotada nenhuma ferramenta da apoio. Na Figura 2 é exibido o gráfico que resume os percentuais de adoção das categorias de ferramentas informadas, observa-se que os juízes on-line são a categoria mais utilizada.

## 4.2 Resultados e discussão

Inicialmente, foi indagado aos professores e monitores quais ações são realizadas ao longo do processo de avaliação adotado. Algumas descrições do processo de avaliação são apresentadas a seguir:

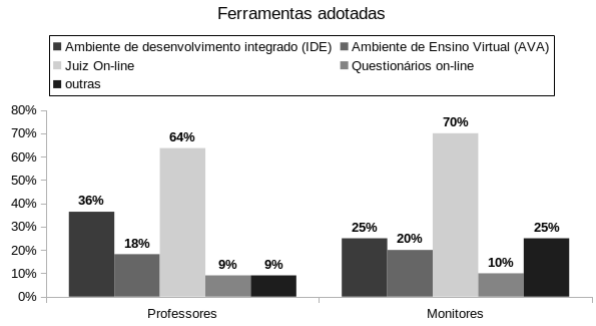


**Figura 2: Ferramentas adotadas nas disciplinas de programação observadas.**

**Professor 1)** "*1) Avaliação do fluxo 'normal' da execução (o que é esperado está sendo executado); 2) Avaliação do fluxo 'errado' da execução (o que não é esperado está sendo tratado); 3) Avaliação lógica do código (variáveis, condicionais, iterações, etc); 4) Avaliação geral do código (foi implementado com boas práticas de programação).*"

**Professor 2)** "*Em geral, gosto de resolver o problema, definindo pesos para as partes do algoritmo e vou comparando as soluções dos alunos com a solução que fiz. Coloco observações sobre eficiência de uso de memória e processamento, mas não penalizo as soluções sob esses aspectos ou sobre o uso de outras estruturas desde que o problema proposto possa ser resolvido.*"

**Professor 3)** "*Tiro pouca pontuação (alguns décimos) de erros sintáticos. Se perceber que a ideia do algoritmo está compatível com uma solução correta, embora contenha erros mínimos, pontuo consideravelmente, penalizando apenas para o aluno perceber o que errou. Geralmente, penalizo a utilização incorreta de if-else, for e while.*"

**Professor 4)** "*Primeiro verifico se o código apresenta o comportamento (saída) esperado. Depois verifico a aplicabilidade das estruturas e comandos utilizados para solucionar o problema apresentado.*"

**Professor 5)** "*Basicamente, eu verifico se o estudante utilizou corretamente o construto esperado para aquela solução (e.g., operador aritmético, estrutura de seleção ou estrutura de repetição). Para mim, esse é o ponto mais importante. Após isso, eu verifico outras características como, por exemplo, indentação, nomes de variáveis, tamanho do código etc.*"

Do mesmo modo, os monitores foram indagados sobre o processo de avaliação adotado. Algumas respostas do processo de avaliação empregados pelos monitores são apresentadas a seguir:

**Monitor 1)** "*Observo o código em busca de erros, depois executo o código utilizando alguns valores para testar os valores de saída.*"

**Monitor 2)**"*Identifico o problema e a solução de referência, logo verifico a solução, começando pela corretude sintática, depois a funcional, e então verifico os detalhes relacionados.*"

**Monitor 3)** "*Verifico a corretude funcional, utilizando diferentes entradas e verificando se a saída é a esperada, seguido pela leitura do código, atendando para a estrutura e buscando compreender a lógica utilizada para resolver o problema.*"

**Monitor 4)** "*Primeiramente eu respondo a questão, criando um código funcional com os conhecimentos utilizados durante aula. Depois, verifico se os resultados correspondem ao esperado. Posteriormente seriam feitas sugestões (podendo ou não descontar na pontuação).*"

**Monitor 5)** "*Verifico se para meus testes está dando a saída correta, depois verifico se há partes desnecessárias ou se não há maneiras mais fáceis de fazer o mesmo.*"

Observando as descrições fornecidas por professores e monitores, é possível notar que a execução do código, fornecendo entradas válidas e inválidas para observar as saídas produzidas é uma pratica comum. Essa execução pode ser automatizada pelos juízes on-line, assim como a observação se um valor fornecido como saída é correto para um conjunto de entrada fornecido. Contudo, podemos identificar que em várias descrições são observados critérios que não são capturados por um juiz on-line, características relacionadas com aspectos de qualidade das soluções são observadas pelos avaliadores.

Dando continuidade a realização das investigações, foi adotada uma categorização de problemas em três classes, sendo elas:

- **Problemas básicos**, são aqueles cujas soluções (códigos) necessitam apenas de um conjunto básico de construções (ex. cálculos matemáticos, entrada e saída, operações de comparação e atribuição);
- **Problemas de decisão**, são aqueles cujas soluções (códigos) necessitam de construções básicas e de estruturas de decisão/condicionais;
- **Problemas de repetição**, são aqueles cujas soluções (códigos) necessitam de construções básicas e de estruturas de repetição/iteração, podendo ou não necessitar de estruturas de decisão/condicionais.

A classificação descrita foi definida pois estes conceitos estão entre os mais abordados nas disciplinas de programação, conforme análise apresentada por Silva, Caceffo e Azevedo [22] e , em geral, são trabalhados como os conceitos iniciais nestas disciplinas.

Após apresentar estas descrições, foi indagado quais, dentre os itens listados no questionário, são os critérios observados nas avaliações. O respondente poderia ainda citar outros critérios.

A seguir, na Figura 3, é apresentado o gráfico relacionado com a indicação dos critérios de avaliação para problemas classificados como básicos. São exibidos apenas os critérios citados com maior frequência por professores ou monitores.

O critério de corretude funcional está relacionado ao programa implementado fornecer as saídas determinadas para um conjunto de entrada. O critério corretude sintática corresponde a adequação do código proposto ao conjunto de regras sintáticas da linguagem utilizada. O uso correto de operadores, para os problemas básicos, está relacionado com a utilização dos operadores matemáticos considerados adequados para resolução do problema. O critério sobre o uso correto de funções de entrada e saída se refere a adequação dos tipos de entrada esperados para a resolução do problema e sua aplicação em relação as funções da linguagem que permitem sua entrada, por exemplo "*print*" e "*input*" na linguagem Python. O tamanho do algoritmo corresponde à quantidade de linhas empregadas na implementação do algoritmo proposto como solução. Os avaliadores consideram que essa característica deve ser observada.


**Figura 3: Características observadas nos códigos ao avaliar problemas "Básicos".**

Conforme é exibido no gráfico exibido na Figura 3, para os professores "Corretude funcional" representa o principal critério ao avaliar problemas do tipo "Básico". Sendo "Corretude funcional" opção indicada por 93% professores. Os itens "Corretude sintática" e "Uso correto de operadores (matemáticos, relacionais, outros)" foram indicados como critérios utilizados por 73% professores, ficando em segundo lugar entre as indicações dos professores.

Do mesmo modo como ocorre com os professores, para os monitores "Corretude funcional" representa o principal critério ao avaliar problemas do tipo "Básico". Sendo "Corretude funcional" opção indicada por 91% monitores. O item "Uso correto de operadores (matemáticos, relacionais, outros)" foi indicado como critério utilizado por 82% monitores, ficando em segundo lugar entre as indicações dos monitores.

Observando o gráfico exibido na Figura 3 é possível afirmar que ocorre uma boa concordância em relação aos critérios adotados por professores e monitores, a exceção é o critério "Tamanho do algoritmo" que possui uma importância maior para os monitores e não é muito considerado pelos docentes.

Para os critérios mais frequentemente apontados para os problemas classificados como básicos, a corretude sintática pode ser avaliada por um juiz on-line, de modo que códigos sintaticamente incorretos não executam e são avaliados com o menor *score* possível, enquanto códigos sintaticamente corretos podem ser avaliados com o conjunto de testes definido. Do ponto de vista pedagógico, podem existir soluções sintaticamente incorretas muito próximas de uma solução adequada, quando, por exemplo, toda a lógica do código é

correta, mas o discente não incluiu um ";" ao final de uma instrução na linguagem C ou Java.

Considerando o critério de corretude funcional, um juiz on-line é adequado para realizar essa avaliação. Um conjunto de testes contendo, por exemplo, 10 testes pode ser definido e caso o código submetido passe por todos os testes é atribuída uma avaliação máxima. Algumas ferramentas permitem a avaliação parcial, sendo assim, quando o o código submetido passa apenas por metade dos casos de teste um *score* de 50% do valor máximo pode ser atribuído.

O uso correto de operadores, o qual no caso de problemas classificados como básicos corresponde aos operadores matemáticos e de atribuição, é parcialmente avaliado por um juiz on-line, pois o uso do operador matemático ou de atribuição incorreto pode impactar no valor de saída apresentado pela solução.

Os critérios "Uso correto de funções de entrada e saída" e "Tamanho do algoritmo" são ignorados por juízes on-line. Um avaliador pode considerar recomendações quando tais critérios não são cumpridos. Retornando às soluções exibidas no Código Fonte 1 e no Código Fonte 2, um avaliador poderia recomendar que instruções de entrada e saída de dados sejam utilizadas em linhas distintas, pois a leitura do código é facilitada. Essa observação pode se relacionar tanto em relação ao critério de "Uso correto de funções de entrada e saída" como também ao critério "Tamanho do algoritmo".

A seguir, na Figura 4, é apresentado o gráfico relacionado com a indicação dos critérios de avaliação para problemas classificados como condicionais. São exibidos apenas os critérios citados com maior frequência por professores ou monitores.

Alguns dos critérios descritos anteriormente, tais como corretude funcional e sintática, continuam sendo empregadas com alta frequência de acordo com os respondentes. O critério de aplicabilidade da estrutura de decisão e a aplicabilidade dos operadores lógicos corresponde à indagação se os respondentes consideram o emprego de uma ou outra estrutura de decisão ou operador lógico relevante quando observam uma solução proposta. O item relacionado com a presença/ausência de decisões aninhadas na resolução do problema se refere ao uso de uma estrutura condicional pertencendo ao bloco de outra estrutura condicional. O critério relacionado com a quantidade de decisões utilizadas na solução proposta se refere à indagação se os respondentes consideram a aplicação de uma menor quantidade de decisões em estruturas condicionais importantes na construção do código.

Conforme é exibido no gráfico exibido na Figura 4, a maioria dos professores observa "Corretude funcional" do código ao avaliar problemas do tipo "Condicionais". Sendo "Corretude funcional" indicada como critério utilizado por 87% professores. O critério "Aplicabilidade dos operadores lógicos usados", particular de problemas condicionais, foi indicado como critério utilizado por 73% professores. Sendo este, o segundo critério mais indicado entre os professores. Os critérios "Aplicabilidade da estrutura de decisão" e "Presença/ausência de decisões aninhadas", particulares de problemas condicionais, foram indicados como critérios utilizados por 67%


**Figura 4: Características observadas nos códigos ao avaliar problemas "Condicionais".**

professores cada um. Sendo estes, classificados empatados como os terceiros critérios mais indicados entre os professores.

Do mesmo modo como ocorre com os professores, para os monitores "Corretude funcional" representa o principal critério ao avaliar problemas do tipo "Condicionais". Sendo "Corretude funcional" indicada como critério utilizado por 91% monitores. O critério "Aplicabilidade dos operadores lógicos utilizados", particular de problemas condicionais, foi indicado como critério utilizado por 77% monitores. Sendo este, o segundo critério mais indicado entre os monitores. O critério "Aplicabilidade da estrutura de decisão", particular de problemas condicionais, foi indicado como critério utilizado por 73% monitores. Sendo este, o terceiro critério mais indicado entre os monitores.

Assim como ocorreu em relação aos problemas classificados como "Básico", o gráfico exibido na Figura 4 é possível afirmar que ocorre uma boa concordância em relação aos critérios adotados por professores e monitores quando consideram problemas classificados como "Condicionais". As exceções são os critérios "Presença/ausência de decisões aninhadas" e "Quantidade de decisões utilizadas" que invertem os níveis de importância quando considerados por professores e monitores.

A seguir, na Figura 5, é apresentado o gráfico relacionado com a indicação dos critérios de avaliação para problemas classificados como repetição. São exibidos apenas os critérios citados com maior frequência por professores ou monitores.

O critério de corretude funcional continua sendo empregado com alta frequência de acordo com os respondentes. Mesmo considerando problemas classificados na categoria de repetição, o critério "Quantidade de decisões utilizadas" foi bastante indicado, pois estruturas de repetição, ex. *while* e *do-while*, podem utilizar este conceito. O critério "Aplicabilidade da estrutura de repetição" corresponde a indagação se os respondentes consideram o emprego de uma ou outra estrutura de repetição relevante quando observam uma solução proposta. Os itens quantidade de estruturas de repetição, profundidade de iterações e presença/ausência de iterações

aninhadas se referem ao modo como as estruturas de repetição são utilizadas e impactam em questões de eficiência da solução proposta.

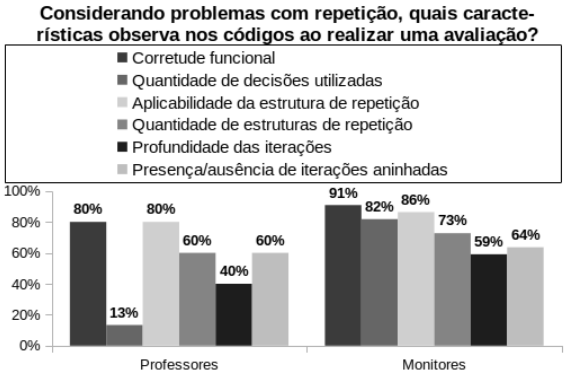


**Figura 5: Características observadas nos códigos ao avaliar problemas "Repetição".**

Conforme é exibido no gráfico exibido na Figura 5, a maioria dos professores observa os critérios "Corretude funcional" e "Aplicabilidade da estrutura de repetição" ao avaliar problemas do tipo "Repetição". Ambas as opções foram indicadas por 80% professores. Os critérios particulares de problemas de repetição, "Quantidade de estrutura de repetição" e "Presença/ausência de iterações aninhadas", foram indicados como critérios utilizados por 60% professores.

Assim como para as classificações anteriores, a maioria dos monitores observa o critério "Corretude funcional" ao avaliar problemas do tipo "Repetição". Esta opção foi indicada por 91% monitores. O critérios particular de problemas de repetição, "Aplicabilidade da estrutura de repetição" foi indicado por 86% dos monitores. O critério "Quantidade de decisões usadas", que envolve estruturas condicionais, foi indicado como critério utilizado por 82% monitores. Sendo este, o terceiro critério mais indicado entre os monitores.

Do mesmo modo como ocorreu em relação aos problemas classificados como "Básicos", no gráfico exibido na Figura 5 é possível afirmar que ocorre uma boa concordância em relação aos critérios adotados por professores e monitores quando consideram problemas classificados como "Repetição". Para um único critério ocorre exceção, sendo o critério "Quantidade de decisões aninhadas" que possui uma importância maior para os monitores e não é muito considerado pelos docentes.

De acordo com as descrições das ações apresentadas por professores e monitores é possível concluir que a abordagem de fornecer entradas e observar saídas esperadas para um código é bastante comum, sendo um juiz on-line suficiente para automatizar a avaliação dessa abordagem. As descrições fornecidas corroboram ainda com a indicação do critério "Corretude funcional" como o mais utilizado para todas as categorias de problemas.

Contudo, ao observar os dados do levantamento, é possível concluir que o critério "Corretude funcional", apesar de ser o mais frequentemente utilizado, não é o único a ser adotado ao longo do processo de avaliação.

Para problemas classificados como básicos, o critério de uso correto de operadores matemáticos, seja para professores ou monitores, possui uma frequência de indicações muito próxima ao critério de corretude funcional. Neste caso, pode ser argumentado que o uso incorreto dos operadores possivelmente gera saídas incorretas, o que é capturado por um juiz on-line.

De modo similar, para problemas classificados como decisão, o critério de aplicabilidade dos operadores lógicos usados são frequentemente apontados por professores e monitores, tendo este critério uma frequência de indicação próxima ao critério de corretude funcional. Considerando ainda os problemas classificados como repetição, o critério aplicabilidade da estrutura de repetição é indicado pelos professores tantas vezes quanto o critério de corretude funcional, e para os monitores os critérios tem uma indicação de frequência muito próximas.

Para ilustrar a limitação existente na abordagem de juízes on-line em relação aos problemas classificados como "Condicionais" ou "Repetição", pode-se adotar o seguinte enunciado "Forneça todos os números pares no intervalo [1;10]", e observados os Códigos exibidos em 3, 4 e 5 como soluções propostas por discentes.

**Código Fonte 3: ".]Solução 1 para o problema "Todos os pares no intervalo [1;10]".**

```
1 print(2)
2 print(4)
3 print(6)
4 print(8)
5 print(10)
```

**Código Fonte 4: ".]Solução 2 para o problema "Todos os pares no intervalo [1;10]".**

```
1 for i in range(2,11,2):
2     print(i)
```

**Código Fonte 5: ".]Solução 3 para o problema "Todos os pares no intervalo [1;10]".**

```
1 for i in range(1,11):
2     if (i % 2 == 0) :
3         print(i)
```

As soluções exibidas no Código Fonte 3, no Código Fonte 4 e no Código Fonte 5 seriam avaliadas como totalmente corretas por um juiz on-line. Contudo, dependendo dos critérios de um avaliador, estas soluções poderiam ser avaliadas de modo diferente.

A solução apresentada no Código Fonte 3 poderia ser penalizada pois um avaliador observaria a ausência de estruturas de decisões e de estruturas de repetição, ferindo os critérios de "quantidade de decisões utilizadas", "aplicabilidade da estrutura de repetição (ex. for e while)" e "quantidade de estruturas de repetição". Dessa forma, poderia ser avaliada com o mais baixo *score* entre as soluções apresentadas.

A solução apresentada no Código Fonte 4 utiliza estruturas de repetição e pode ser suficiente para solucionar o problema

solicitado de forma completa. Contudo, um avaliador pode argumentar que uma solução que emprega uma estrutura de decisão deveria ser utilizada, uma vez que o conteúdo relacionado já foi apresentado. Por sua vez, um avaliador distinto poderia descrever que o código é suficiente e é mais eficiente do que outros que utilize uma condição, pois emprega menos instruções e terá um processamento mais rápido.

De forma similar ao que foi descrito anteriormente, a solução apresentada no Código Fonte 5 poderia ser avaliada de formas distintas, dependendo do avaliador que realizaria a avaliação,

A abordagem de juízes on-line possui grande importância no contexto de ensino e da aprendizagem. Contudo, essa abordagem possui limitações. Sendo assim, podemos afirmar que para o universo estudado, o uso de juízes on-line não é suficiente para englobar todos os critérios utilizados pelos avaliadores, sejam estes os professores ou monitores nas disciplinas. A grande vantagem no uso de um juiz on-line se refere a velocidade de avaliação, provendo um rápido *feedback*.

Um docente que observe somente a saída correta gerada pelo programa, poderia ser auxiliado somente por um juiz on-line. Porém, conforme observado nas descrições apresentadas no *levantamento*, existem docentes e monitores que julgam ser importante considerar outros aspectos durante a avaliação, para estes a abordagem de juízes on-line necessita de um "VAR", um meio auxiliar para avaliar cada aspecto considerado na solução.

Então, para que se continue com essa vantagem, mas de modo a englobar outros critérios adotados por um avaliador, se torna necessária adoção de formas de avaliação adicionais que possam empregar outras verificações auxiliares aos juízes on-line. Dessa forma, é possível afirmar que é necessário um "VAR" para o ensino e a aprendizagem de programação.

### 4.3 Ameaças à validade

Os avaliadores que participaram do levantamento possuem diferentes contextos e experiências em relação à avaliação em disciplinas introdutórias de programação. Ocorre variação no tempo em que um avaliador lecionou ou foi monitor(a) em disciplinas, diferentes linguagens e ferramentas foram utilizadas, entre outras questões. Desta forma, se faz presente a ameaça à validade de constructo, neste caso em relação ao contexto e a experiência dos sujeitos participantes do levantamento. Com isso, pode se descrever que os resultados apresentados podem ser diferentes se tais variações não estiverem presentes.

Ameaças à validade externa também estão presentes neste levantamento, ou seja, os resultados exibidos podem não ser válidos para uma população mais geral. Uma vez que apenas 37 avaliadores participaram do levantamento, e que o universo de avaliadores é bastante vasto para o contexto de programação, é possível que se o mesmo levantamento for realizado para um conjunto diferente de avaliadores, os resultados não sejam semelhantes.

## 5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Nesta pesquisa, buscou-se investigar quais são as práticas relacionadas ao processo de avaliação em atividades práticas de disciplinas introdutórias de programação.

Como trabalho futuro considera-se a condução de uma investigação sobre o uso de juízes on-line e outras ferramentas de avaliação junto aos discentes. Dessa forma, espera-se analisar a efetividade destas ferramentas em relação ao aprendizado dos estudantes.

A estratégia de uso de juízes on-line é a alternativa mais popular quando se trata de avaliação de códigos como exercícios. Um juiz on-line permite que os discentes realizem muitos exercícios e recebam *feedback* de modo rápido.

Os critérios adotados por avaliadores (professores e monitores), em relação ao processo de avaliação de códigos, foram investigados. Com essa investigação é possível afirmar que embora a maioria utilize como critério a adequação do código a um conjunto de entrada e observação das saídas, o que é suportado pelas abordagens que empregam juízes on-line, outros critérios são apontados e também possuem um alto nível de utilização, embora pareçam ter um menor peso na avaliação.

Pode-se descrever que os juízes on-line possuem um grande valor, pois tais ferramentas disponibilizam uma grande quantidade de exercícios, permitem que discentes realizem a submissão de suas soluções e observem um *feedback*, a indicação se o código foi considerado correto ou incorreto, de maneira extremamente rápida. Conclui-se que juízes on-line são ferramentas auxiliares valiosas, estas contudo não são suficientes para realizar um processo de avaliação no contexto de ensino e da aprendizagem de programação de acordo com as observações realizadas por professores e monitores.

Desta forma, os resultados apresentados sugerem que é importante considerar outras abordagens de avaliação, além dos juízes on-line no contexto de ensino e da aprendizagem de programação. Outras abordagens que empregam as mais diversas técnicas vem sendo pesquisadas pela comunidade, estas podem auxiliar a abordagem de juízes on-line provendo um *feedback* mais adequado aos discentes.

Outras abordagens que empregam as mais diversas técnicas vem sendo pesquisadas pela comunidade, algumas empregam abordagens de Inteligência Artificial com o intuito de agilizar o *feedback* fornecido aos discentes.

## REFERÊNCIAS

[1] Luis Gustavo Araujo, Roberto Bittencourt, and Christina Chavez. 2021. Python Enhanced Error Feedback: Uma IDE Online de Apoio ao Processo de Ensino-Aprendizagem em Programação. In *Anais do Simpósio Brasileiro de Educação em Computação* (On-line). SBC, Porto Alegre, RS, Brasil, 326–333.

[2] Daramsenge Bilegjargal and Nien-Lin Hsueh. 2021. Understanding Students' Acceptance of Online Judge System in Programming Courses: A Structural Equation Modeling Approach. *IEEE Access* 9, 152606–152615.

[3] Rodrigo. L. B. L. Campos. 2010. Metodologia ERM2C: Para melhoria do processo de ensino aprendizagem de lógica de programação. In *Anais do XXX Congresso da Sociedade Brasileira de Computação, Workshop de Educação em Informática (CSBC 2010 - WEI 2010)*. Minas Gerais, Brasil.

[4] Alexandre de Andrade Barbosa, Allan Lucio Correia, and Evandro de Barros Costa. 2015. Um mapeamento sistemático sobre analisadores de código em disciplinas de programação. In *Anais do Simpósio Brasileiro de Informática na Educação*.

[5] Alexandre de Andrade Barbosa and Evandro de Barros Costa. 2015. Simulated Learners in Peers Assessment for Introductory Programming Courses. In *Proceedings of the Workshops at the 17th International Conference on Artificial Intelligence in Education, AIED 2015, Madrid, Spain, June 22 + 26, 2015*. http://ceur-ws.org/Vol-1432/sl_pap7.pdf

[6] C. E. DE CAMPOS, C. P. ; FERREIRA. 2004. BOCA: um sistema de apoio a competições de programação (BOCA: A Support System for Programming Contests).. In *Workshop de Educacao em Computacao (Brazilian Workshop on Education in Computing), Anais do Congresso da SBC*. Salvador, BA.

[7] M. Delamaro, M. Jino, and J. Maldonado. 2013. *Introdução Ao Teste De Software*. Elsevier Brasil. https://books.google.com.br/books?id=zqw4DwAAQBAJ

[8] Alexandre N. Duarte, Hugo Moreira, and Thiago Silva Mello. 2010. Competitividade como fator motivacional para o estudo de computação. In *Anais Simpósio Brasileiro de Informática na Educação*. João Pessoa, PB, Brasil.

[9] Leandro Galvão e David Fernandes e Bruno Gadelha. 2016. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 27, 1, 140.

[10] Jessica Dagostini e Marcos Lima e Jean Bez e Neilor Tonin. 2018. URI Online Judge Blocks: Construindo Soluções em uma Plataforma Online de Programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 29, 1, 168.

[11] Fábio P. Alves e Patricia Jaques. 2015. Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação* 3, 1, 51.

[12] Andres Porfirio e Roberto Pereira e Eleandro Maschio. 2021. A-Learn EvId: A Method for Identifying Evidence of Computer Programming Skills Through Automatic Source Code Assessment. *Revista Brasileira de Informática na Educação* 29, 0, 692–717.

[13] Matheus Gaudencio. 2019. *Repositório do pycomparecode*. https://github.com/matheusgr/pycomparecode Último acesso em fevereiro de 2017.

[14] Jeisson Hidalgo-Céspedes, Gabriela Marín-Raventós, and Marta Eunice Calderón-Campos. 2020. Online Judge Support for Programming Teaching. In *2020 XLVI Latin American Computing Conference (CLEI)*. 522–530.

[15] Rodziah Latih, Marini Abu Bakar, Norleyza Jailani, Noorazean Mohd. Ali, Syahanim Mohd. Salleh, and Abdullah Mohd. Zin. 2017. PC2 to support instant feedback and good programming practice. In *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*. 1–5.

[16] Sihan Li, Xusheng Xiao, Blake Bassett, Tao Xie, and Nikolai Tillmann. 2016. Measuring Code Behavioral Similarity for Programming and Software Engineering Education. In *Proceedings of the 38th International Conference on Software Engineering Companion* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery (ACM), New York, NY, USA, 501–510.

[17] Erivan Lima and Emanuel Coutinho. 2019. Uma Análise sobre o Desempenho de Alunos de Graduação em Disciplinas Iniciais de Programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 30, 1, 1876.

[18] Marcos Lima, Leandro Carvalho, Elaine Oliveira, David Oliveira, and Filipe Pereira. 2021. Uso de atributos de código para classificação da facilidade de questões de codificação. In *Anais do Simpósio Brasileiro de Educação em Computação* (On-line). SBC, Porto Alegre, RS, Brasil, 113–122.

[19] Marcelle Pereira Mota, Silvana R. de Brito, Mireille Pinheiro Moreira, and Eloi Luiz Favero. 2009. Ambiente integrado a plataforma moodle para apoio ao desenvolvimento das habilidades iniciais de programação. In *Anais do XX Simpósio Brasileiro de Informática na Educação (Simpósio Brasileiro de Informática na Educação 2009)*. Florianópolis, SC, Brasil.

[20] Joseph Oliveira, Elaine Oliveira, Leandro Carvalho, and David Fernandes. 2019. Mensagens estendidas de feedback em um juiz online para alunos de introdução à computação: resultados preliminares. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 30, 1, 329.

[21] Rodrigo B. Paes, Romero Malaquias, Marcio Guimaraes, and Hyggo Almeida. 2013. Ferramenta para a Avaliação de Aprendizado de Alunos em Programação de Computadores. In *Anais do II Congresso Brasileiro de Informática na Educação (Congresso Brasileiro de Informática na Educação 2013) Workshops (Workshops do Congresso Brasileiro de Informática na Educação 2013)*. Campinas, SP.

[22] Eryck Silva, Ricardo Caceffo, and Rodolfo Azevedo. 2022. Análise dos Tópicos Mais Abordados em Disciplinas de Introdução à Programação em Universidades Federais Brasileiras. In *Anais do II Simpósio Brasileiro de Educação em Computação* (Online). SBC, Porto Alegre, RS, Brasil, 29–39.

[23] Marina Silva and Ana Paula Ferreira. 2022. Linguagens visuais para o ensino de programação: uma revisão da literatura com foco em paradigmas de programação. In *Anais do II Simpósio Brasileiro de Educação em Computação* (Online). SBC, Porto Alegre, RS, Brasil, 18–28.

[24] Marlos Tacio Silva, Evandro De Barros Costa, Paulo Henrique Barbosa, and Juliana De Carvalho Cavalcante. 2014. Um Arcabouço para Construção de Mecanismos de Análise de Códigos de Programação Introdutória. In *Anais do Simpósio Brasileiro de Informática na Educação*. 1083–1092.

[25] Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. 2013. Automated Feedback Generation for Introductory Programming Assignments. In *Procceedings of the 34th Association for Computing Machinery SIGPLAN Conference on Programming Language Design and Implementation* (Seattle, Washington, USA) *(PLDI)*. New York, USA, 15–26.

[26] Allan A. Sioson. 2013. Experiences on the use of an automatic C++ solution grader system. In *IISA 2013*. 1–6.

[27] Zahid Ullah, Adidah Lajis, Mona Jamjoom, Abdulrahman Altalhi, Abdullah Al-Ghamdi, and Farrukh Saleem. 2018. The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. *Computer Applications in Engineering Education* 26.

[28] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. 2018. A Survey on Online Judge Systems and Their Applications. *Association for Computing Machinery Computer Survey* 51, 1, Article 3, 34 pages.