Juiz online focado no ensino de programação: uma análise de usabilidade de ferramenta autoral

Leandro L. M. da Silva[1], **Vivyann F. Cedraz**[1], **Jeferson A. do Patrocínio**[1], **Saul Delabrida**[1], **Reinaldo S. Fortes**[1]

[1]Universidade Federal de Ouro Preto (UFOP)
Campus Morro do Cruzeiro – Ouro Preto, MG – Brasil.

{leandro.liberio, vivyann.cedraz}@aluno.ufop.edu.br,

{jefersonap, saul.delabrida, reifortes}@ufop.edu.br,

Abstract. *Professors have been using online judgment tools to support programming education. Although many tools are available, most are geared toward competitive programming, which can negatively affect student engagement in the early stages. This paper introduces a new tool and evaluates its usability for students of various undergraduate courses. The focus was to evaluate the students' perception of the introduction to programming in the initial periods. With 118 students from 4 courses participating in the evaluations, the results show that, despite good usability, the tool can impact students from non-computer-related courses. The paper discusses learned lessons and future development opportunities.*

Resumo. *Docentes têm utilizado ferramentas de julgamento online para apoiar o ensino de programação. Embora muitas ferramentas estejam disponíveis, a maioria é voltada para a programação competitiva, o que pode afetar negativamente o engajamento dos discentes nos estágios iniciais. Este artigo apresenta uma nova ferramenta e avalia sua usabilidade para discentes de variados cursos de graduação. O foco foi avaliar a percepção dos discentes de introdução à programação em períodos iniciais. Com 118 discentes de 4 cursos participando das avaliações, os resultados mostram que, apesar da boa usabilidade, a ferramenta pode impactar discentes de cursos não relacionados à computação. O artigo discute lições aprendidas e oportunidades de desenvolvimento futuro.*

1. Introdução

A presença da tecnologia nas mais diversas organizações e segmentos da sociedade é inevitável diante de tantos avanços nas últimas décadas. Na educação não tem sido diferente, tanto com o aumento do número de matrículas em Educação a Distância (EaD), quanto o aumento expressivo da utilização de ferramentas para suporte ao ensino. Entretanto, o uso de tais ferramentas exige cuidados, para que elas não se tornem mais um obstáculo ao invés de um mecanismo facilitador. Para isso, uma ferramenta educacional *online* deve oferecer uma interface eficiente e que conduza o discente ao seu objetivo principal, de estudar e compreender o conteúdo pedagógico, exigindo esforço e tempo mínimos.

A introdução à programação de computadores no ensino superior é um grande desafio devido a variados fatores, como a falta de base em *pensamento computacional*,

DOI: 10.5753/sbie.2024.242421

ainda pouco difundido na educação básica do Brasil, e o excesso de discentes nas turmas. A falta de motivação é um desafio adicional, observável em disciplinas introdutórias de programação, normalmente obrigatórias em cursos de ciências exatas que não envolvem a Ciência da Computação e afins (como engenharias, matemática e física). Sendo assim, nada mais natural do que recorrer ao uso de ferramentas computacionais para promover o ensino de programação, motivando os discentes e contribuindo com o seu aprendizado.

Outro requisito importante é a disponibilidade *online* da ferramenta, para permitir que os discentes acessem a qualquer momento e recebam rapidamente o resultado de correção. São inúmeras as ferramentas disponíveis na *Web* com essa finalidade. Exemplo disso são os sistemas de correção automática de exercícios de programação, conhecidos como os *juízes online*, como *Beecrowd*[1], *SPOJ*[2] e *Codeforces*[3]. Este tipo de ferramenta disponibiliza desafios de programação e corrige as respostas submetidas instantaneamente. Entretanto, elas são mais focadas em treinamento para programação competitiva do que para o ensino de programação propriamente dito.

Embora a programação competitiva seja um mecanismo eficiente para engajar discentes à prática de programação e, portanto, contribuir para o ensino-aprendizagem, do ponto de vista didático-pedagógico ela pode não ser muito eficiente, principalmente para iniciantes e discentes de outros cursos que não os de Ciência da Computação e afins.

Neste contexto, surge o *opCoders Judge*, um *juiz online* utilizado para o ensino de programação introdutória na Universidade Federal de Ouro Preto (UFOP). Optou-se pelo desenvolvimento de uma ferramenta autoral visando: (a) construir uma ferramenta simples e objetiva, de fácil utilização pelos discentes; (b) aspectos pedagógicos, focando mais no ensino da programação do que no treinamento para programação competitiva; (c) flexibilidade e facilidade para criação de novos recursos e funcionalidades.

Este trabalho visa avaliar quais impactos sobre a perspectiva de usabilidade no uso de um juiz *online* nas disciplinas de programação nos cursos que acontecem nos primeiros períodos. Desta maneira, estima-se que o *opCoders Judge* possa contribuir para o ensino de programação de computadores e pensamento computacional, como uma ferramenta de auxílio ao ensino-aprendizagem. Sendo assim, neste trabalho, avaliações de usabilidade com 118 voluntários de variados cursos que ensinam programação de computadores foram realizadas. O projeto foi aprovado pelo Comitê de Ética em Pesquisa (CEP). Observou-se, primeiramente, a utilização do *opCoders Judge* em aulas práticas ao longo dos últimos quatro semestres letivos e depois, a partir de avaliações de usabilidade preliminares, um protótipo operacional com foco no usuário foi elaborado pelos voluntários. Docentes perceberam que muitos discentes tinham dificuldades no uso de ferramentas computacionais, incluindo não apenas ambientes integrados de programação, mas também o próprio *opCoders Judge*. Observou-se também que discentes de cursos mais envolvidos com tecnologias computacionais, como Ciência da Computação e Controle e Automação, se mostravam mais aptos ao uso das ferramentas do que discentes de outros cursos. Estimamos que dificuldades na utilização destas ferramentas desviam o foco do discente para o aprendizado do conteúdo pedagógico, com o risco de se as tornar obstáculos ao invés de facilitadores.

[1] `https://www.beecrowd.com.br/`
[2] `https://www.spoj.com/`
[3] `https://codeforces.com/`

Portanto, como principais contribuições deste artigo destaca-se: (a) apresentação do *opCoders Judge*, um juiz *online* desenvolvido com foco no ensino de programação de computadores para uso em disciplinas de graduação; (b) a realização de avaliações de usabilidade para aprimoramentos na ferramenta e análise de resultados segmentados por cursos; (c) a discussão acerca das experiências e lições aprendidas pelos autores deste artigo na utilização da ferramenta, que direcionam para variadas expectativas de desenvolvimentos futuros. Tais contribuições permitem um melhor entendimento do desenvolvimento e utilização da ferramenta e evidenciam a necessidade de atenção especial à usabilidade, contribuindo com a evolução do *opCoders Judge* e ferramentas correlatas.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma revisão bibliográfica. O *opCoders Judge* é descrito na Seção 3. Resultados das avaliações de usabilidade, lições aprendidas e direcionamentos para trabalhos futuros são discutidos nas Seções 4 e 5. A Seção 6 conclui o artigo.

2. Revisão Bibliográfica

Esta seção apresenta fundamentos teóricos sobre *Juízes online*, na Subseção 2.1, métodos de avaliação de usabilidade na Subseção 2.2 e trabalhos relacionados na Subseção 2.3.

2.1. Juízes online

Um *Juiz online* é uma ferramenta computacional desenvolvida para realizar correção automática de exercícios práticos de programação. A correção automática pode ser baseada em dois tipos de análises: **dinâmica** e **estática** [Ball 1999, Louridas 2006]. Na análise *dinâmica* são executados testes de *caixa-preta*, onde o programa é executado e compara-se as saídas obtidas com as saídas esperadas. Elementos de desempenho também podem ser avaliados, como tempo de execução e consumo de memória. Na análise *estática* o programa não é executado. O código é examinado a partir de alguma representação abstrata gerada a partir dele (e.g. *Árvore de Sintaxe Abstrata*, *Grafo de Fluxo de Controle* e *Grafo de Dependências*), identificando erros e não-conformidades com o especificado para o problema. Pode-se considerar que as duas análises sejam complementares, possibilitando avaliar os programas sob perspectivas diferentes e amplificando aspectos pedagógicos.

A maioria dos *Juízes online* realiza apenas a análise dinâmica e são focados em treinamento para programação competitiva[4], limitando-os do ponto de vista pedagógico [Brito and Fortes 2019]. Os enunciados dos exercícios definem padrões de entrada e saída sem mensagens amigáveis para contextualizá-los, e um teste é considerado válido apenas se a saída do programa é exatamente igual à saída esperada. Para iniciantes em programação, o uso de mensagens simples como “Informe o valor de X” ou “O resultado é Y” pode fazer diferença para a compreensão do problema e sua solução. Além disso, a segmentação de elementos da saída esperada, como mensagens textuais estáticas e valores processados, diferenciando seus pesos na nota atribuída pela correção automática, podem contribuir no aspecto pedagógico. Neste contexto, conforme defendido por [Brito and Fortes 2019], algumas discrepâncias na saída, como diferenças entre caixa das letras, falta de acentuação e espaçamentos extras, não devem gerar penalidades na nota, com o risco de desmotivar os discentes pela punição de erros de baixa criticidade.

[4]Competições de programação envolvem desafios práticos de programação, ganha o competidor (ou time) que resolve corretamente o maior número de exercícios em um determinado tempo.

Embora a programação competitiva desempenhe um papel relevante no ensino e difusão da prática de programação, o *opCoders Judge* observa os cenários descritos anteriormente, visando se tornar uma ferramenta mais voltada para os aspectos pedagógicos do ensino de programação do que para o treinamento para programação competitiva.

2.2. Avaliação de Usabilidade

Usabilidade é um conceito na área de Interação Humano-Computador (IHC) relacionada à qualidade de uma interface [Winckler and Pimenta 2002]. Conforme norma ISO [9241-11 1998], usabilidade pode ser definida como "*a medida na qual um produto pode ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso*".

Como relatado anteriormente, existe uma grande variedade de cursos de graduação atendidos pelas disciplinas introdutórias de programação e os docentes, ao utilizar o *opCoders Judge*, começaram a observar certas dificuldades na sua utilização, embora sua interface seja simples e composta de poucas funcionalidades. Com isso, torna-se relevante realizar avaliações de usabilidade visando descobrir potenciais problemas na interface e identificar perfis de usuários que eventualmente demandam maior atenção e cuidados. Este é um dos principais objetivos do presente trabalho.

Há diversos métodos na literatura capazes de avaliar a usabilidade de uma interface. Neste trabalho foram aplicadas avaliações por investigação, que envolvem a participação dos usuários, possibilitando examinar suas opiniões, comportamentos, pontos de vista e expectativas em relação ao sistema. Os dois métodos aplicados neste trabalho, SUS e SAM, foram escolhidos por sua forte aderência aos objetivos do estudo, além de sua facilidade e rapidez de aplicação e análise de resultados.

O *System Usability Scale* (SUS) é um modelo de questionário proposto por [Brooke et al. 1996], que visa avaliar a usabilidade de uma interface por meio de uma escala numérica, avaliando três critérios: *efetividade*, *eficiência* e *satisfação*. O questionário contém 10 afirmações com respostas na escala *Likert*, variando entre 1 ("*Discordo totalmente*") e 5 ("*Concordo totalmente*"). Afirmações ímpares mapeiam características positivas, enquanto as pares mapeiam características negativas. Espera-se notas altas para características positivas e notas baixas para as negativas. A pontuação geral da interface é definida a partir de um cálculo descrito pelo autor, e cujo valor final corresponde à usabilidade geral do sistema, variando entre 0 e 100 pontos. Para [Sauro 2011], a pontuação média considerada aceitável para uma interface é 68 pontos.

Proposto por [Lang 1980], o *Self-Assessment Manikin* (SAM) é um modelo de questionário que utiliza pictogramas para avaliar a qualidade de um sistema a partir de três dimensões: *satisfação*, *motivação* e *controle*. O questionário utiliza 5 pictogramas, representando as alterações afetivas em cada dimensão, e o usuário escolhe aquele que corresponde ao seu sentimento após utilizar o sistema. A pontuação mínima é 1 e a pontuação máxima é 9, para cada uma das dimensões. Por utilizar pictogramas, a escala SAM é facilmente compreendida e se torna fácil aplicá-la em diversos cenários [Aguirre et al. 2019]. Além disso, os pictogramas conseguem ativar estruturas cerebrais, de modo que a resposta do usuário seja muito similar aos estímulos verdadeiros [Lang and Bradley 2007].

2.3. Trabalhos relacionados

Buscou-se, por trabalhos que tenham relatado avaliações com usuários para identificar problemas de usabilidade em juízes *online* para o ensino de programação introdutória.

[Pelz et al. 2012] apresentam um mecanismo para correção automática de tarefas de programação que considera análises estáticas e dinâmicas segmentadas em: (1) verificação erros sintáticos; (2) conformidade com comandos obrigatórios; (3) similaridade com códigos gabaritos; (4) comparação entre saídas esperadas e obtidas. Experimentos foram realizados com 93 discentes de Ciência da Computação e algumas limitações foram apontadas pelos autores: (a) possibilidades de burlar o mecanismo de correção; (b) ser restrito a problemas pequenos.

[Freitas et al. 2016] analisaram a usabilidade do módulo Laboratório Virtual de Programação do Moodle (VPL) utilizado na Universidade Federal de Santa Catarina. A análise foi feita com 37 discentes do curso de Bacharelado em Tecnologias de Informação e Comunicação. Os autores realizaram a avaliação heurística de [BASTIEN and SCAPIN 1993] e verificaram a usabilidade por meio do questionário de satisfação do *Inventário de Medição de Usabilidade de Software* (SUMI). Os resultados mostraram inúmeros problemas, e todos os usuários manifestaram insatisfação com a interface através do questionário. Alguns deles precisaram repetir a tarefa várias vezes e esclarecer dúvidas sobre algumas funcionalidades.

[Souza 2019] desenvolveu o protótipo de um software educacional, o *Loop Academic*, para suporte no ensino-aprendizagem de programação introdutória. O protótipo foi concebido a partir das heurísticas de usabilidade propostas por [Nielsen 1994] e das heurísticas reformuladas por [Benyon 2011]. A avaliação de usabilidade foi aplicada a 7 discentes de Ciência da Computação com base na estratégia de [Valentim et al. 2017], com aplicação da Escala SAM [Lang 1980]. Como resultado, os autores concluem que o protótipo se adéqua à necessidade dos discentes, alcançando quase todas as máximas nas avaliações de *Satisfação*, *Controle* e *Motivação* da Escala SAM.

Percebe-se que, a despeito da importância da avaliação de usabilidade de *Juízes online*, poucos trabalhos no Brasil têm uma preocupação específica quanto a isso. Nos trabalhos anteriores foram relatados sucesso quando às avaliações de usabilidade para discentes de Ciência da Computação, mas resultados insatisfatórios para discentes de Tecnologia da Informação e Comunicação. Sendo assim, além de apresentar o desenvolvimento do *opCoders Judge*, o foco deste trabalho também está em avaliar a sua interface com discentes de diferentes cursos e, a partir disso, contribuir com o desenvolvimento futuro de ferramentas de domínios relacionados. Pelo nosso conhecimento, este tipo de avaliação ainda não foi reportado na literatura a respeito de *juízes online* no Brasil.

3. O *opCoders Judge*

O *opCoders Judge* é uma ferramenta utilizada para apoio ao ensino de programação introdutória em vários cursos de graduação da Universidade Federal de Ouro Preto (UFOP). Seu desenvolvimento foi realizado por meio uma sequência de estágios incrementais.

O desenvolvimento do *opCoders Judge* se iniciou com a realização de correção por meio de acesso *offline*. A gestão das tarefas é feita pelo *Moodle*, permitindo *download* do enunciado em arquivo PDF e *upload* dos arquivos de solução. Após o prazo de entrega,

os arquivos são baixados pelo professor e corrigidos pela execução de um *script* Python que gera os resultados de correção em arquivos PDF e os envia aos discentes por *e-mail*. Para atender aos requisitos didático-pedagógicos, são feitas análises dinâmica e estática, seguindo as definições de [Brito and Fortes 2019]. A análise estática: (a) é ponderada pelo grau de similaridade textual da saída obtida com a saída esperada; (b) não penaliza certas diferenças, como espaços extras, acentuação de caracteres, sinais de pontuação e diferenças de caixas alta e baixa; (c) é segmentada, aplicando pesos menores a mensagens textuais estáticas e pesos maiores a saídas variáveis resultantes de processamentos. A análise estática avalia somente a definição de funções solicitadas no exercício, verificando apenas a existência de função com o mesmo nome e quantidade de argumentos de entrada e saída. As principais limitações desta versão são: (a) seu funcionamento *offline*; e (b) a fragilidade na avaliação de funções.

Em um segundo estágio, implementou-se uma versão de acesso *online*. O objetivo é permitir que os discentes submetam suas soluções e obtenham as correções prontamente e a qualquer momento. Buscou-se uma interface simples, objetiva e com funcionalidades básicas para o discente: (a) listagem de tarefas; (b) descrição e enunciados dos exercícios que compõem cada tarefa; (c) recurso para entrega de soluções para cada exercício; (d) detalhes de correção, onde se destaca as diferenças entre a saída esperada e a saída obtida em cada caso de teste; (e) consulta a dados pessoais e alteração de senha. A análise estática foi aprimorada, agora são realizados testes unitários para cada função, através da comparação dos valores retornados em chamadas das funções implementadas pelos discentes e os valores de retorno esperados. O resultado foi satisfatório e a ferramenta é atualmente utilizada efetivamente pelos discentes de diversas turmas de introdução à programação envolvendo variados cursos de graduação. A Figura 1 apresenta o mapa do site para acesso às funcionalidades para os discentes.

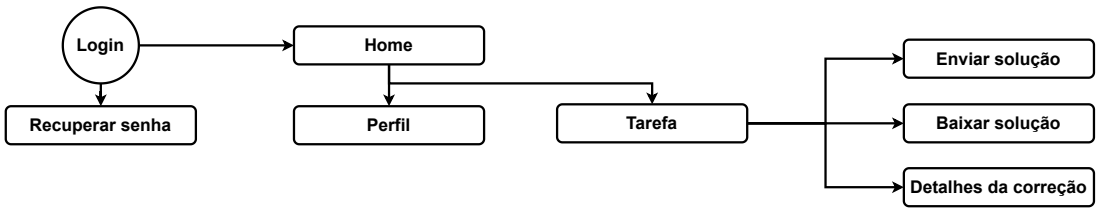


Figura 1. Mapa do site do *opCoders Judge* na visão do discente.

Apesar de serem poucas páginas, com funcionalidades simples, os docentes observaram que alguns discentes apresentavam dificuldades na utilização da ferramenta. Assim, em um terceiro estágio focou-se totalmente na experiência do discente. O objetivo foi identificar problemas de usabilidade e elaborar um protótipo de interface para implementação futura. Primeiramente, realizou-se uma avaliação por inspeção da versão *online* do *opCoders Judge* implementada anteriormente, seguindo as 10 Heurísticas de Nielsen [Nielsen 1994]. A partir das falhas observadas, desenvolveu-se o protótipo de uma nova interface e conduziram-se avaliações por investigação, seguindo os questionários SUS e SAM, com discentes de Ciência da Computação.

Em seguida, em um quarto estágio, a partir das avaliações de usabilidade realizadas, buscou-se aprimorar o protótipo anterior. Adicionalmente, considerando mais uma vez a experiência dos docentes com a utilização do *opCoders Judge* em sala de aula,

observou-se que discentes de diferentes cursos apresentavam variados graus de dificuldade com a interface. A hipótese levantada é que discentes de cursos mais próximos à Ciência da Computação apresentam menos dificuldades em termos de usabilidade do que discentes de outros cursos. Sendo assim, o novo protótipo foi avaliado com a aplicação dos questionários SUS e SAM com discentes de variados cursos de ciências exatas. Com os resultados destas avaliações, deu-se início à implementação de uma nova versão *online* para o *opCoders Judge*, que ainda não foi concluída.

4. Análise de usabilidade do *opCoders Judge*

[Bastien 2010] destaca a importância das avaliações de usabilidade, argumentando que muitas vezes os desenvolvedores não conseguem identificar alguns aspectos negativos na interface. Em sistemas onde o perfil dos usuários é muito variado, conhecê-los é também essencial para a implementação de um sistema com boa usabilidade. O *opCoders Judge* é disponibilizado para discentes de quase 30 cursos de ciências exatas (e.g. engenharias, matemática, química e física). Sendo assim, este trabalho se concentra em apresentar o resultado de avaliações de usabilidade com foco na comparação de diferentes grupos.

Materiais e métodos são apresentados na Subseção 4.1. Na Subseção 4.2 é apresentada uma análise de perfil dos voluntários. Na Subseção 4.3 são apresentados resultados dos questionários SUS e SAM. E na Seção 4.4 são avaliadas opiniões dos voluntários.

4.1. Materiais e Métodos

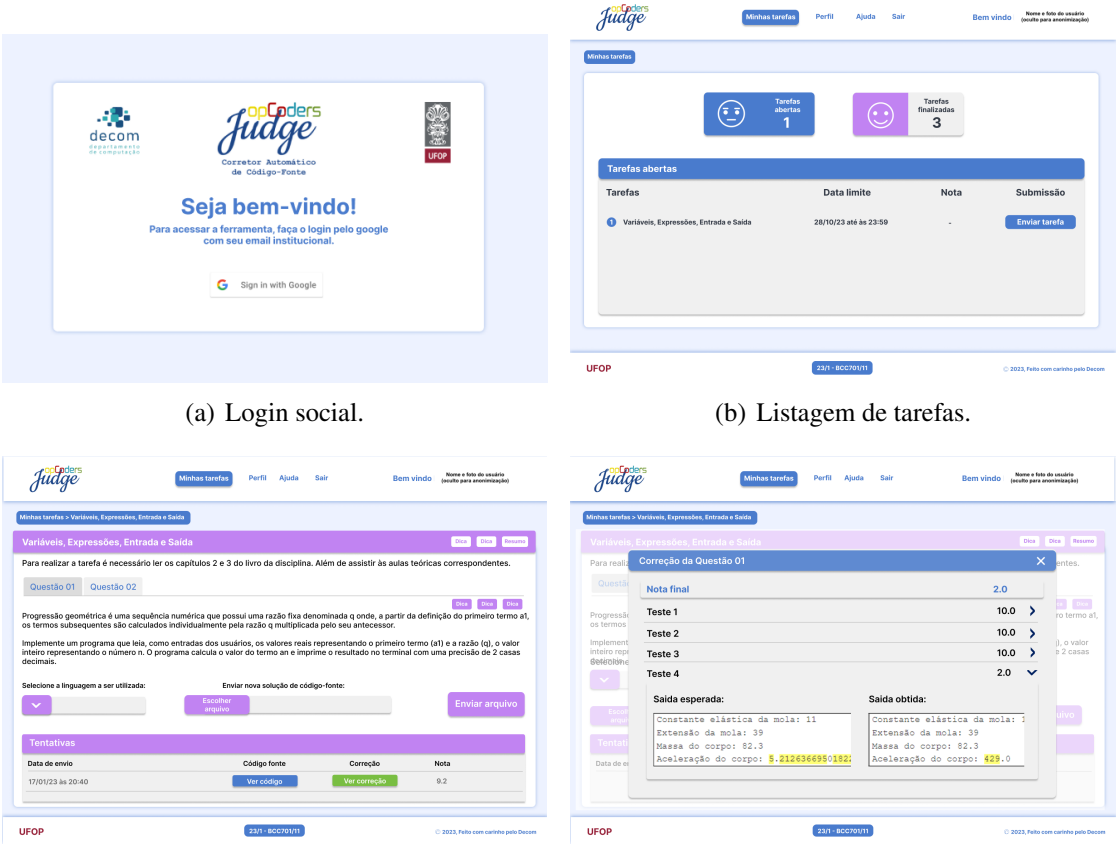
Os voluntários foram submetidos a uma tarefa utilizando um protótipo interativo implementado no **Figma**, devido ao baixo custo de prototipação em comparação ao custo do desenvolvimento final, bem como o tempo gasto no desenvolvimento. A tarefa consistia em simular a submissão da solução de um exercício prático e a verificação do resultado fictício da correção desta entrega através do protótipo interativo. Os voluntários foram submetidos a 3 questionários: **Perfil**, **SUS**, **SAM**. A avaliação de usabilidade foi aprovado pelo Comitê de Ética em Pesquisa (CEP) e pode ser identificado através do Certificado de Apresentação de Apreciação Ética (CAAE) de número 63182722.9.0000.5150.

Para a condução das avaliações optou-se por selecionar voluntários matriculados em disciplina de introdução à programação do primeiro período dos cursos: (a) Ciência da Computação (COM); (b) Eng. de Controle e Automação (CAU); (c) Eng. Mecânica (MEC); (d) Arquitetura e Urbanismo (AUR). Estima-se que esta ordem apresente uma crescente de dificuldade com a interface, ou seja, que discentes COM tenham mais facilidade, seguidos por CAU e MEC, enquanto discentes AUR tenham a maior dificuldade.

As avaliações aconteceram no horário da aula de programação, foi primeiramente explicado como e porque a avaliação estava sendo realizada e esclarecida a natureza voluntária de sua participação. Ao final, as avaliações envolveram a participação de 118 voluntários, sendo: (a) 31 de COM; (b) 26 de CAU; (c) 27 de MEC; (d) 34 de AUR.

As principais interfaces do protótipo são apresentadas na Figura 2. O fluxo de utilização é bem simples. Primeiro é exigida a autenticação através do login social Google (Figura 2(a)). Logo após a autenticação, a listagem de tarefas disponíveis para o discente é apresentada (Figura 2(b)). O discente clica em uma tarefa disponível para ter acesso ao seu conteúdo, composto de dicas e enunciado dos exercícios práticos (Figura 2(c)). Além disso, esta página lista as entregas realizadas para cada exercício. O discente

tem opção de baixar um código-fonte entregue ou visualizar os detalhes da correção (Figura 2(d)). O protótipo também disponibiliza páginas para informações de perfil e acesso a ajuda, porém, para a realização das avaliações não é esperada a necessidade de acesso a elas. Optou-se pela utilização do protótipo tendo em vista que a versão *online* do *opCoders Judge* já havia passado por avaliações por inspeção e investigação e problemas já haviam sido identificados. O protótipo visava solucionar os problemas identificados e novas avaliações foram realizados para validar a interface proposta antes de sua implementação.



(a) Login social.

(b) Listagem de tarefas.

(c) Conteúdo da tarefa (dicas e questões).

(d) Detalhes de correção de uma solução.

Figura 2. Principais páginas do protótipo.

4.2. Perfil dos participantes

O questionário de perfil tem a intenção de identificar o grau de conhecimento e familiaridade do discente nos estudos de programação, envolvendo os seguintes questionamentos: (a) *Se está cursando o primeiro período do curso*; (b) *Se já usou algum corretor de exercícios de programação*; (c) *Se gosta de programar*; (d) *Se já programou*. A Figura 3 sintetiza os percentuais de respostas obtidas para cada pergunta.

Observa-se que a grande maioria dos voluntários está *cursando o primeiro período do curso*, com percentuais muito próximos entre os cursos, com destaque para um menor percentual para discentes MEC, com 74%. Discentes cursando o primeiro período ainda não possuem muitas informações sobre a disciplina e ferramentas utilizadas, sendo menos propícios a eventuais influências que pudessem enviesar suas respostas, os percentuais observados garantem maior confiabilidade nos resultados das avaliações. Poucos

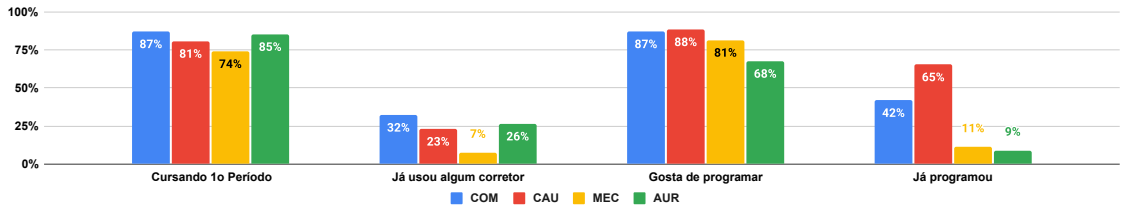


Figura 3. Resultado da avaliação de perfil dos voluntários.

discentes *já tiveram contato com um corretor* de exercícios de programação, sendo que o maior percentual é de 32% para discentes COM. Entretanto, destaca-se o baixo percentual de discentes MEC, apenas 7%, e muito inferior aos demais. Com relação a *gostar de programar*, há novamente pouca diferença entre os percentuais dos cursos, com apenas um percentual um pouco mais afastado, mostrando que discentes AUR de fato gostam menos do objeto de estudos da disciplina de programação. Por fim, para os discentes que *já programaram*, uma grande variedade de percentuais, com destaque para o percentual de discentes CAU, com o maior valor (65%), e discentes com pouca experiência em programação para os cursos MEC (11%) e AUR (9%).

Estes resultados, em conjunto com o número de discentes voluntários para cada curso (veja Seção 4.1), demonstram que a amostragem de discentes é equilibrada e compatível com os objetivos do estudo. A quantidade de discentes novatos é grande, assim, a maioria dos voluntários não teve oportunidade de trocar informações ou conhecer a ferramenta avaliada e há poucos discentes com experiência no uso de ferramentas correlatas.

4.3. Resultados SUS e SAM

As Tabelas 1 e 2 apresentam os resultados sumarizados dos questionários SUS e SAM, respectivamente. Como discutido na Subseção 2.2 o SUS atribui um *valor final de pontuação* entre 0 e 100, por meio de perguntas *positivas* e *negativas* e o SAM permite avaliar as dimensões de *satisfação*, *motivação* e *controle*. Estes fatores encontram-se representados nas referidas tabelas. A pontuação final SUS segue o cálculo padrão definido pelo modelo e os valores das questões positivas e negativas são definidos pela média das respostas dos voluntários, assim como ocorre para os valores das três dimensões do SAM.

Na Tabela 1, observa-se que discentes COM, conforme esperado, apresentam os melhores valores para todas as métricas. Estimava-se que os discentes AUR apresentariam os piores resultados, porém, quem os obteve foram os discentes MEC. Isto pode ser em consequência da discrepância entre os percentuais de voluntários que já tinham utilizado corretor (apenas 7%, veja Subseção 4.2). Conforme esperado, discentes CAU ficaram mais próximos de discentes COM, enquanto discentes AUR ficaram mais próximos de discentes MEC. Embora os resultados de pontuação final sejam bons (bem acima do mínimo de 68, veja Seção 2.2), eles apontam a possibilidade de melhorias na interface, e a discrepância esperada entre diferentes perfis de usuários.

Na Tabela 2 observa-se resultados próximos entre discentes COM e CAU, estando estes entre os melhores resultados. Discentes COM se mostraram mais satisfeitos com a interface, porém menos motivados e com menor sensação de controle do que os discentes

	COM	CAU	MEC	AUR
Pontuação final	**87,42 ▲**	83,75	*78,61 ▽*	80,15
Questões positivas	**4,42 ▲**	4,26	*4,06 ▽*	4,20
Questões negativas	**1,43 ▲**	1,58	*1,88 ▽*	1,78

Tabela 1. Resultados de avaliação pelo questionário SUS.

	COM	CAU	MEC	AUR
Satisfação	**8,0 ▲**	7,7	*6,6 ▽*	7,0
Motivação	6,6	**7,4 ▲**	*5,8 ▽*	7,2
Controle	7,3	**7,6 ▲**	*6,0 ▽*	6,9

Tabela 2. Resultados de avaliação pelo questionário SAM.

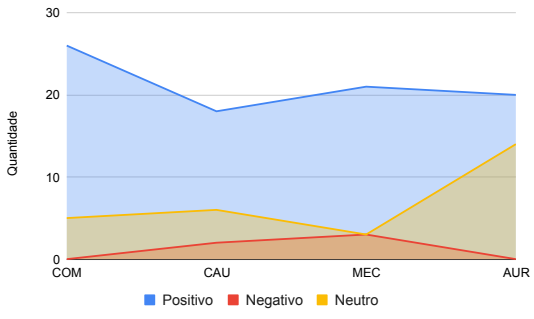
CAU. Os piores resultados novamente ficaram para os discentes MEC, provavelmente pelo mesmo motivo anterior, a menor taxa de discentes que já conheciam algum corretor.

Embora os resultados apresentados nas Tabelas 1 e 2 sejam bons do ponto de vista da usabilidade, eles também demonstram que os discentes de áreas sem forte relação com Ciência da Computação possuem uma percepção de usabilidade menor. Isso, pode requerer que os docentes tenham que demandar uma atenção maior sobre o impacto do uso desta ferramenta nos períodos iniciais e de que o desenvolvimento deste tipo de ferramenta deve também levar este aspecto em consideração.

Vale ressaltar que as avaliações foram feitas em um protótipo interativo implementado a partir de avaliações de usabilidade anteriores. Sendo assim, bons resultados em termos de usabilidade já eram esperados. Embora haja espaço para melhorias de interface, os resultados gerais das avaliações demostram que o objetivo de uma interface com elevado grau de usabilidade foi alcançado.

4.4. Opiniões dos próprios voluntários

Os voluntários expressaram sua opinião por meio de um campo de texto livre. Classificamos as respostas como: “positivas”, “negativas” ou “neutras”. Dentre os 118 voluntários, apenas 6 deixaram este campo vazio, sendo classificadas como neutras. Respostas que apontavam tanto aspectos positivos quanto negativos foram também classificadas como neutras. Também avaliamos palavras-chave nas respostas. A Figura 4 ilustra resultados.



(a) Classificação das opiniões.

(b) Nuvem de palavras.

Figura 4. Análise de opiniões dos usuários.

Na Figura 4(a), observa-se a predominância de aspectos positivos para todos os cursos. Chama a atenção a quantidade de aspectos neutros de discentes AUR, embora aspectos positivos continue expressivo. Na Figura 4(b), observa-se uma predominância ainda mais absoluta para aspectos positivos, como '*fácil*', '*intuitiva*', '*prática*' e '*simples*', enquanto aspectos negativos aparecem raramente, como '*confusa*' e '*complicada*'.

5. Lições Aprendidas e direcionamentos para desenvolvimentos futuros

O *opCoders Judge* vem sendo efetivamente utilizado em sala de aula ao longo dos últimos quatro semestres letivos. Observações dos docentes em conjunto com os resultados das avaliações de usabilidade constituem uma relevante fonte de conhecimento para direcionar desenvolvimentos futuros, descritos resumidamente a seguir.

Avaliações de usabilidade: Por mais simples que os recursos possam parecer para os desenvolvedores, alguns discentes costumam ter mais dificuldade no uso de certas funcionalidades. Esta situação pode ser agravada por uma certa resistência e desmotivação de muitos discentes, ainda mais quando são utilizadas outras ferramentas que os discentes recém-chegados à universidade ainda não possuem familiaridade, como *Moodle*, videoaulas, sistema acadêmico da instituição, dentre outras. Embora a relevância das avaliações de usabilidade seja comprovada cientificamente, ainda é possível identificar sua ausência em muitos projetos de desenvolvimento de ferramentas computacionais aplicadas a este contexto. Para implementações futuras planejamos replicar as avaliações de usabilidade, explorar outros métodos de avaliação e aprofundar as análises destes resultados.

Desenvolvimento focado em diferentes perfis de usuários: Após elaborar um protótipo de alta fidelidade, realizamos avaliações de usabilidade que apontaram bons resultados, mas que ainda pode ser melhorado. Isto demonstra a importância de se desenvolver a ferramenta tendo sempre em vista o foco na experiência do usuário e da realização de avaliações para validação das propostas, antes mesmo da implementação de uma ferramenta operacional. No escopo deste trabalho destaca-se a grande diversidade de cursos e, consequentemente, diversidade de perfis de discentes. Por um lado, discentes que possuem mais facilidade de uso, podem demostrar menos motivação, enquanto discentes com mais dificuldade podem perder motivação ao longo do tempo. Um olhar atento para as nuances de cada perfil de discente é essencial para o sucesso da ferramenta.

A importância de limitar o conteúdo das páginas: Observamos que parte da falta de motivação dos discentes pode estar na necessidade de ler e absorver muitas informações. A primeira versão *online* do *opCoders Judge* oferece uma descrição da tarefa contendo resumo da matéria e dicas práticas. Na maioria das vezes este recurso possui muito conteúdo textual. Observou-se que, apesar das informações e dicas serem relevantes para a resolução dos exercícios da tarefa, é importante que ela seja exibida de forma segmentada e apenas quando o discente sentir necessidade de informações adicionais.

A importância de prover informações de ajuda: Outro recurso não observado inicialmente pela equipe de desenvolvimento foi um sistema de ajuda para os usuários em relação aos recursos oferecidos pela ferramenta. A primeira versão *online* do *opCoders Judge* não oferece recurso de ajuda, mas o protótipo proposto leva isso em consideração, disponibilizando recurso específico de ajuda para as funcionalidades. Muitos dos participantes, principalmente alguns que demandaram mais tempo de avaliação, acabaram encontrando respostas para suas dificuldades acessando o mecanismo de ajuda do protótipo.

A introdução de mecanismos motivacionais: Apesar das avaliações com os discentes apresentarem bons resultados de usabilidade, observamos uma limitação do ponto de vista da motivação. Concluímos que, apesar de todo o esforço em desenvolver um protótipo centrado no usuário, ainda é necessário aumentar a capacidade da ferramenta em motivar os discentes a realizarem as tarefas. Acredita-se que o caminho para isso possa ser introduzir elementos de gamificação e outros mecanismos de interação como realidade virtual ou aumentada para apresentação de resultados e conteúdo pedagógico envolvido à resolução dos problemas, tornando o uso da ferramenta mais agradável e engajador.

Geração de um banco de questões dinâmicas: Uma demanda essencial para um juiz *online* é a oferta de um variado conjunto de exercícios práticos. Além disso, cada exercício pode ser elaborado com a inserção de elementos de valores variáveis, permitindo que um exercício seja apresentado de diferentes maneiras por meio de componentes dinâmicos, escolhidos a partir de regras e critérios de aleatoriedade, preservando a essência do problema. Além disso, ela pode representar uma eficaz medida preventiva contra compartilhamento de soluções entre colegas de turma, promovendo a integridade acadêmica.

Geração automática de questões: O preenchimento do banco de questões exige muito esforço dos docentes. Neste contexto, explorar diferentes *chatbots* baseados em *Large Language Models* (LLM's) pode proporcionar a criação automática de questões a partir de instruções passadas ao *prompt*. Porém, incorporar um *prompt* ao *opCoders Judge*, especializado para a criação e inserção de questões diretamente através de sua interface, seria essencial para otimização de tempo e esforço para a geração de um banco de questões variadas e abrangentes.

Personalização e tutoria inteligente: Com um conjunto abrangente de questões e incorporação de conteúdo pedagógico, é possível agregar modelos de inteligência artificial para conduzir atividades atreladas às capacidades, dificuldades e características individuais dos discentes baseados no histórico de utilização do *opCoders Judge*.

6. Conclusão

Este artigo apresenta o *opCoders Judge*, um juiz *online* autoral para ensino de programação e a uma avaliação de usabilidade. O objetivo foi avaliar os possíveis impactos que esta ferramenta pode trazer se aplicada em cursos que possuem disciplinas de programação introdutória nos semestres iniciais.

Avaliações de usabilidade foram realizadas com discentes de quatro cursos distintos e os resultados, através dos métodos SUS e SAM, demonstram que a interface proposta possui uma boa usabilidade. Observou-se também que uma grande variedade de perfis de discentes exige um cuidado ainda mais especial com relação à usabilidade. Por fim, discutiu-se uma lista de lições aprendidas durante o desenvolvimento e análises de usabilidade do *opCoders Judge*, podendo assim, contribuir diretamente com o desenvolvimento ou aprimoramento de ferramentas correlatas.

Como trabalhos futuros, pretende-se continuar o desenvolvimento do *opCoders Judge* com foco no usuário e em avaliações abrangentes de usabilidade, agregando novos recursos e funcionalidades como: (a) a criação de um banco de questões dinâmicas por meio de mecanismos de geração automática de exercícios; (b) inserção de mecanismos motivacionais por meio de elementos de gamificação e realidade estendida; (c) capacidades de personalização e tutoria inteligente por meio de modelos de inteligência artificial.

Agradecimentos

Os autores gostariam de agradecer à Universidade Federal de Ouro Preto (UFOP), ao Departamento de Computação da UFOP (DECOM/UFOP), à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), Código do Financiamento APQ-03665-22, e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Código de Financiamento 306101/2021-1.

Referências

[9241-11 1998] 9241-11, I. (1998). Ergonomic requirements for office work with visual display terminals (vdts) — part 11: Guidance on usability. Disponível em: `https://www.iso.org/standard/16883.html` Acesso em: 08 outubro 2022.

[Aguirre et al. 2019] Aguirre, A. R., da Cunha, S. M., Deluchi, M., Gonçalves, R., and Bizarro, L. (2019). Aplicação da escala SAM na seleção de imagens de alimentos saudáveis e não saudáveis para utilização em tarefas experimentais. *Ciências & Cognição*, 24(2):245–264.

[Ball 1999] Ball, T. (1999). The concept of dynamic analysis. *ACM SIGSOFT Software Engineering Notes*, 24(6):216–234.

[BASTIEN and SCAPIN 1993] BASTIEN, J. C. and SCAPIN, D. (1993). Ergonomics criteria for the evaluation of human-computer interfaces: Relatório de pesquisa nº 156. *INRIA-Institut National de Recherche en Informatique et en Automatique*.

[Bastien 2010] Bastien, J. M. C. (2010). Usability testing: a review of some methodological and technical aspects of the method. *Elsevier*, 1:6.

[Benyon 2011] Benyon, D. (2011). Interação humano-computador. *Traduçao de Heloısa Coimbra de Souza. 2a. ed. Sao Paulo: Person Prentice Hall*, page 464.

[Brito and Fortes 2019] Brito, P. and Fortes, R. (2019). O uso de corretores automáticos para o ensino de programação de computadores para alunos de engenharia. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, page 449.

[Brooke et al. 1996] Brooke, J. et al. (1996). SUS - A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

[Freitas et al. 2016] Freitas, L. M. d. et al. (2016). *Análise de Usabilidade do Módulo Laboratório Virtual de Programação do Moodle*. Monografia (graduação em tecnologia da informação e comunicação), Universidade Federal de Santa Catarina.

[Lang 1980] Lang, P. (1980). Behavioral treatment and bio-behavioral assessment: Computer applications. *Technology in mental health care delivery systems*, pages 119–137.

[Lang and Bradley 2007] Lang, P. and Bradley, M. M. (2007). The international affective picture system (iaps) in the study of emotion and attention. *Handbook of emotion elicitation and assessment*, 29:70–73.

[Louridas 2006] Louridas, P. (2006). Static code analysis. *Ieee Software*, 23(4):58–61.

[Nielsen 1994] Nielsen, J. (1994). Usability inspection methods. In *Conference companion on Human factors in computing systems*, pages 413–414.

[Pelz et al. 2012] Pelz, F. D., Jesus, E. A. d., and Raabe, A. L. A. (2012). Um Mecanismo para Correção Automática de Exercícios Práticos de Programação Introdutória. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 23(1). Number: 1.

[Sauro 2011] Sauro, J. (2011). Measuring usability with the system usability scale (SUS). Disponível em: `https://measuringu.com/sus/` Acesso em: 08 outubro 2022.

[Souza 2019] Souza, D. M. O. (2019). *Desenvolvimento e avaliação do protótipo do loop academic: um software educacional para o auxílio no processo de ensino-aprendizagem de programação introdutória*. Monografia (graduação em tecnologia da informação), Universidade Federal Rural do Semi-Árido.

[Valentim et al. 2017] Valentim, N. M. C. et al. (2017). *Antecipando a usabilidade nas fases iniciais do processo de desenvolvimento de software*. Tese (doutorado em informática), Universidade Federal do Amazonas.

[Winckler and Pimenta 2002] Winckler, M. and Pimenta, M. S. (2002). Avaliação de usabilidade de sites web. *Escola de Informática da SBC Sul (ERI 2002). Porto Alegre*, 1:85–137.