Estudo sobre o surgimento e evolução de *misconceptions* em uma disciplina de introdução à programação

Airton Nascimento Silveira Filho, Victor Araújo Passos, Leandro Silva Galvão de Carvalho, Elaine H. T. Oliveira, David Fernandes, Fabíola Guerra Nakamura

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. Gen. Rodrigo Octávio, 6200, Coroado I – 69080-900 – Manaus – AM – Brasil

{airton.filho,victor.passos,galvao,elaine,david,fabiola}@icomp.ufam.edu.br

Resumo. *Juízes online, amplamente usados em disciplinas introdutórias de programação, costumam focar na conformidade com casos de teste, pouco contribuindo para identificar compreensões imperfeitas (*misconceptions*) do aprendiz sobre o assunto. Neste trabalho, levantou-se a evolução dos misconceptions exibidos pelos alunos ao longo de um período letivo em uma disciplina desse tipo, a partir dos códigos submetidos a um juiz online. Os resultados indicam que* misconceptions *relacionados a estruturas de decisão são os mais persistentes ao longo do tempo. Além disso, verificou-se que alguns misconceptions frequentemente surgem em conjunto, como os ligados a atribuições e estruturas redundantes, aplicadas desnecessariamente na codificação, enquanto outras categorias aparecem pontualmente. A contribuição desta análise não se limita a catalogar os* misconceptions *dos alunos, possibilitando que pesquisas futuras abordem estratégias de mitigar os mais frequentes e os mais recorrentes.*

Abstract. *Online judges, widely used in introductory programming courses, focus on compliance with test cases, contributing little to the identification of students'* misconceptions *on the subject. This study examined the evolution of misconceptions exhibited by students over an academic period in such a course, based on codes submitted to an online judge. The results indicate that* misconceptions *related to decision structures are the most persistent over time. Furthermore, some* misconceptions *were often observed together, such as those related to attributions and redundant structures unnecessarily applied in coding, while other categories appeared only occasionally. This analysis contributes not only by cataloging students' misconceptions but also by enabling future research to address strategies to mitigate the most frequent and recurrent ones.*

1. Introdução

Misconceptions, ou compreensões imperfeitas, são equívocos que surgem durante o processo de aprendizagem dos estudantes, e que entram em conflito com conceitos comumente aceitos na área de estudo [Qian and Lehman 2017]. Na computação, esses entendimentos são inadequados para contextos práticos de programação [Sorva 2013], mas alguns autores apresentam esses *misconceptions* como elementos-chave durante o desenvolvimento dos estudos, retirando parte do teor negativo do termo [Gomes 2010].

Esses equívocos podem ocorrer quando o aluno conhece os recursos disponíveis para criar um código, mas, devido a conceitos aprendidos de forma parcial, acaba aplicando estruturas de maneira desnecessária ou redundante. Diferentemente de confusões casuais, ou *mistakes*, que geralmente envolvem problemas de escrita e resultam em erros de sintaxe, gerando erros de compilação ou execução [McCall and Kölling 2014]. *Misconceptions* estão relacionadas à compreensão inadequada dos módulos fundamentais do conteúdo abordado, como na resolução de atividades, onde essas compreensões imperfeitas são frequentemente encontradas. Um exemplo é na implementação de estruturas de decisão usadas corretamente em questões que necessitam de múltiplos casos, entretanto com verificações inconsistentes criadas pelo estudante que podem deixar a estrutura maior do que o necessário.

A análise desses *misconceptions* é fundamental para identificar rapidamente seu surgimento durante o estudo de módulos iniciais em disciplinas introdutórias de programação. Se não forem corrigidas, essas dificuldades de compreensão podem se perpetuar, impactando negativamente o entendimento de módulos mais avançados e comprometendo a assimilação de novas estruturas. Esse cenário pode criar obstáculos significativos no progresso dos estudantes em programação introdutória, cuja reversão pode se tornar cada vez mais difícil para programadores iniciantes [Teague and Lister 2014].

Trabalhos como o de [Sorva 2012] catalogaram *misconceptions* gerais e específicos na linguagem Java, enquanto [Caceffo et al. 2017] e [Gama et al. 2018] os exploraram em C e Python. Em outra vertente, [Araujo et al. 2021] utilizaram dados de juízes online para ampliar essas análises de códigos em Python e [Silva et al. 2023a] aplicaram aprendizado de máquina para identificar compreensões imperfeitas a partir de códigos corretos, criando o catálogo mais amplo até o momento. Entretanto, esses estudos de catalogação representam um passo inicial na compreensão dos *misconceptions*. O presente trabalho se situa um passo adiante, na tentativa de explorar uma oportunidade de pesquisa nessa área, que é entender o surgimento, a evolução e a remediação dos *misconceptions* ao longo do tempo.

O presente trabalho tem o objetivo de analisar o surgimento e a recorrência de *misconceptions* cometidos por estudantes de uma disciplina introdutória de programação. A pesquisa busca identificar categorias recorrentes de *misconceptions* ao longo da disciplina, estabelecendo conexões entre as áreas afetadas e os momentos em que esses problemas emergem. Além disso, o estudo explora como essas compreensões imperfeitas se manifestam, persistem em diferentes conteúdos da disciplina e eventualmente desaparecem ao longo da prática de codificação dos alunos. Para essa análise, foram verificados códigos submetidos por estudantes para avaliações de aprendizado em um juiz online. Para nortear este trabalho, foram levantadas as seguintes questões de pesquisa:

QP1: Quais são os principais *misconceptions* identificados em cada módulo da disciplina?

QP2: As categorias utilizadas para rotular os *misconceptions* foram suficientes para abranger todos os problemas encontrados?

QP3: A permanência desses *misconceptions* ao longo da disciplina gerou variações ou evoluções dessas compreensões imperfeitas?

2. Trabalhos relacionados

Estudos sobre *misconceptions* em disciplinas introdutórias de programação têm sido realizados em diferentes contextos acadêmicos, focando principalmente em como esses entendimentos incompletos afetam o desempenho dos alunos ao longo do tempo. A relevância desse tipo de estudo se destaca na possibilidade de desenvolver estratégias pedagógicas que possam reduzir as taxas de reprovação e melhorar o aprendizado de programação.

[Sorva 2012], em sua análise sobre *misconceptions* na linguagem Java, desenvolveu um catálogo de 162 *misconceptions* que vai além de conceitos práticos, abrangendo também aspectos teóricos, como a crença de que "o computador sabe as intenções do programa ou de um pedaço do código, e age de acordo". O estudo levantou 86 *misconceptions* específicos da linguagem Java, com foco em programação orientada a objetos.

[Caceffo et al. 2017] identificaram *misconceptions* em disciplinas introdutórias de programação na linguagem C, por meio da análise de provas e entrevistas com alunos, destacando problemas recorrentes relacionados ao uso de variáveis, parâmetros de função e escopo, recursão, expressões booleanas, iteração, ponteiros e estruturas. Com esses dados, eles elaboraram um questionário de múltipla escolha para identificar entendimentos incompletos de forma mais ampla. Expandindo essa abordagem, [Gama et al. 2018] adaptaram a pesquisa para a linguagem Python, excluindo aspectos específicos do C, como ponteiros e estruturas, e identificando novos problemas relacionados ao uso e implementação de classes e objetos. O estudo resultou em um catálogo com 28 *misconceptions* específicos dessa linguagem, complementando a análise inicial de [Caceffo et al. 2017].

O trabalho de [Araujo et al. 2021] teve como diferencial a abordagem de Juízes Online como base de dados, criando um novo catálogo de *misconceptions* baseado no trabalho de [Gama et al. 2018] e aplicando em um grupo amostral diferente da pesquisa anterior, também utilizando Python como linguagem. Assim, foi criado um catálogo através de uma análise empírica contendo 27 *misconceptions*, sendo que 19 eram comuns entre os dois levantamentos e os novos incluíam problemas de sintaxe e lógica.

Por fim, [Silva et al. 2023a] propuseram um novo catálogo, dessa vez analisando somente códigos em Python considerados corretos, a partir de questões resolvidas em juiz online. Eles utilizaram aprendizado de máquina para otimizar a identificação, resultando em um catálogo de 45 *misconceptions* no total. Esse catálogo foi utilizado de base para esta pesquisa, pois é o mais recente e completo até o início do trabalho.

Além disso, este trabalho contribui ao acompanhar a evolução do surgimento e permanência de *misconceptions* ao longo de um semestre letivo de uma disciplina introdutória de programação, ministrada na linguagem Python. Assim, obteve-se um panorama dinâmico das turmas estudadas, extrapolando o registro estático que os catálogos, apesar de úteis, acabam fornecendo.

3. Metodologia

Nesta pesquisa, foi utilizada a base de dados anonimizada de um juiz online, contendo registros de interação dos alunos matriculados em uma disciplina introdutória de programação, realizada no primeiro semestre letivo de 2023 na Universidade Federal do Amazonas. Essa disciplina foi ministrada em seis módulos temáticos. Cada módulo

compreende uma aula teórico-prática de abertura, listas de exercícios formativos e a uma avaliação somativa parcial. Os temas são explicitados na Tabela 1. Foram selecionadas turmas de dois cursos distintos, fora da área de Computação: Licenciatura em Matemática e Engenharia Mecânica, pois possuem discrepâncias tanto na nota de corte para ingresso, como na nota média nessa disciplina. O estudo focou apenas nas questões utilizadas nas seis avaliações parciais, que foram resolvidas presencialmente em um ambiente fiscalizado, minimizando o risco de plágio e, consequentemente, de que as respostas registradas em um ID, na verdade, fossem provenientes de outra pessoa.

Tabela 1. Conteúdos ensinados ao longo dos seis módulos da disciplina

Módulo	Conteúdo
M1	Programação Sequencial
M2	Decisão Simples
M3	Decisão aninhada
M4	Repetição por condição
M5	Vetores e Strings
M6	Repetição por contagem

3.1. Coleta e Tratamento de dados

Para coletar os códigos de resposta dos alunos, foi utilizado e adaptado um *script* Python previamente desenvolvido por [Lima et al. 2021]. Os dados coletados com o auxílio do extrator estavam anonimizados pelo juiz online, sendo apenas possível identificar os alunos através de IDs. Para a filtragem dos cursos a serem analisados, foi utilizado um processo manual, verificando individualmente as informações de cada arquivo gerado e, caso pertencesse ao curso desejado, registrando o ID para a análise posterior. Ao fim desse tratamento, chegou-se ao total de 83 estudantes, sendo 41 pertencentes à Engenharia Mecânica e 42 à Licenciatura em Matemática.

3.2. Análise dos códigos de solução dos alunos

Ao todo, os 83 estudantes analisados geraram 498 códigos distintos de solução às questões de avaliação. Para as questões em que os códigos foram submetidos e avaliados pelo juiz online, a análise considerou a última versão submetida. Já para as questões em que o estudante realizou apenas testes com seu código, sem o submeter para avaliação, foi analisada a última versão de código registrada.

Antes de iniciar a avaliação de cada código e estudante, executou-se um procedimento de calibração de critérios entre os dois primeiros autores deste trabalho, a fim de prover maior confiabilidade na detecção de *misconceptions*. Para evitar inconsistências nas interpretações dos pesquisadores, cada um analisou individualmente 10 códigos em comum, e posteriormente compartilharam e discutiram os resultados, alinhando as metodologias de avaliação e as interpretações. Após essa calibração, cada pesquisador ficou responsável pelos estudantes de um dos dois cursos selecionados neste estudo.

Para realizar uma análise abrangente e identificar todos os possíveis momentos de surgimento de *misconceptions* durante a disciplina, foram consideradas tanto as questões que passaram em todos os casos de teste (consideradas corretas), como as questões que

não passaram em pelo menos um caso de teste (consideradas incorretas). Essa abordagem foi adotada porque problemas conceituais não se limitam a questões avaliadas como erradas. É possível que uma solução seja aceita pelo juiz online, mas apresente lógica ou estrutura de código inadequada, ou ineficaz. Dessa forma, a análise detalhada de todos os códigos foi essencial para garantir que nenhum *misconception* passasse despercebido.

3.3. Classificação dos *misconceptions*

Os *misconceptions* foram identificados e organizados com base na classificação proposta por [Silva et al. 2023a], que estruturaram os *misconceptions* em grupos associados a conteúdos modulares e tópicos específicos abordados em cada etapa da disciplina. Esses agrupamentos incluíam problemas relacionados a estruturas de decisão, repetição, entre outros, facilitando a análise dos resultados.

Para questões incorretas, foi necessário identificar o erro e determinar se ele era resultado de uma casualidade ou de um *misconception*. Já as questões marcadas como corretas exigiram uma análise mais detalhada da lógica do código, para verificar a existência de um *misconception* que interferiu na eficácia da questão. Inicialmente, buscou-se analisar se o problema encontrado poderia ser classificado em uma das categorias propostas por [Silva et al. 2023a] ou se seria necessária a criação de uma nova categoria para representá-lo com maior precisão.

4. Resultados obtidos

A metodologia gerou achados significativos, e nesta seção serão apresentados os resultados obtidos, guiados por questões de pesquisa levantadas a partir das análises realizadas.

4.1. Abordagem analítica dos registros

Nesta pesquisa, os *misconceptions* foram identificados nos seguintes contextos:

- **Submissões corretas:** códigos que passaram em todos os casos de teste do juiz online, mas que apresentaram falhas na lógica ou na estrutura, indicando algum *misconception*.
- **Submissões incorretas:** códigos que não passaram nos casos de testes do juiz online devido a erros relacionados a compreensões imperfeitas ou erros de implementação.
- **Códigos não submetidos:** códigos que não foram submetidos pelo estudante para avaliação no juiz online, mas que foram utilizados em testes e apresentaram sinais de *misconceptions*.

Além disso, alguns códigos não puderam ser incluídos na análise devido à falta de informações ou características que os qualificassem para o estudo, como:

- **Códigos incompletos:** rascunhos de código que não apresentavam uma quantidade suficiente de linhas para uma execução efetiva, como registros que continham apenas declarações de variáveis em seu corpo.
- **Compreensões imperfeitas do enunciado**: casos em que os erros observados foram atribuídos a falhas de lógica, sem relação direta com *misconceptions* de programação, como equívocos na interpretação do enunciado da questão.

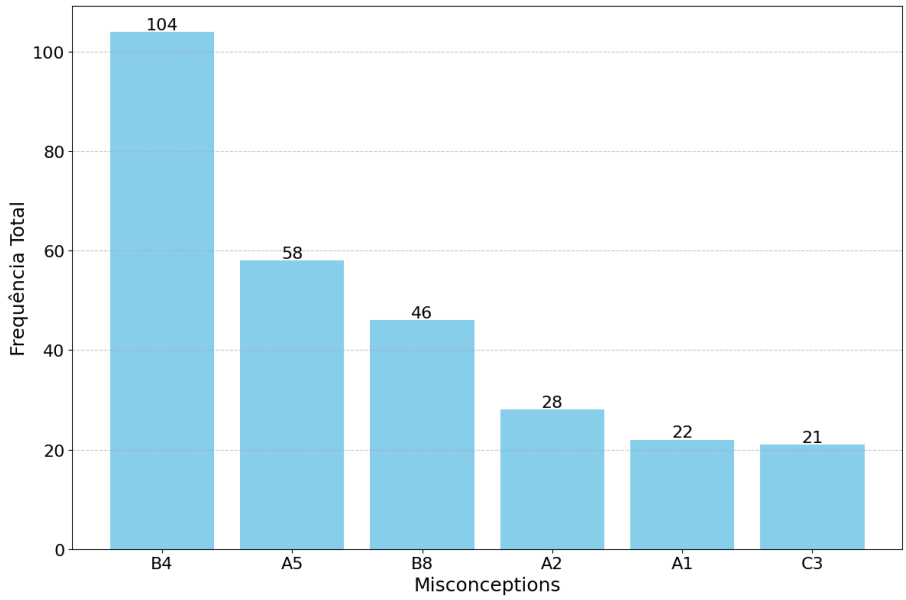

Figura 1. Recorte dos seis *misconceptions* mais frequentes na resolução de avaliações nos cursos de Engenharia Mecânica e Licenciatura em Matemática

Durante a análise, ao se deparar com um *misconception*, os pesquisadores tentavam enquadrá-lo em uma das 45 categorias rotuladas por [Silva et al. 2023a]. As mais frequentes estão listadas na Tabela 2. A listagem completa pode ser consultada em [Silva et al. 2023b] (p.125).

Tabela 2. Descrição dos *misconceptions* mais frequentes, conforme nomenclatura de [Silva et al. 2023a]

ID	Nome
A1	Variável não utilizada
A2	Variável atribuída a si mesma
A5	Importação não utilizada
B4	Comandos repetidos dentro de blocos if-elif-else
B8	Não utilização da declaração elif/else
C3	Operações redundantes dentro do loop

Com base nessa análise, foi possível organizar graficamente os *misconceptions* mais frequentes em ambos os cursos (Figura 1). Destacaram-se problemas de concepção relacionados às estruturas de decisão, sendo a categoria com maior ocorrência associada ao uso repetitivo de comandos em blocos *if-else* em situações onde uma abordagem mais eficiente poderia ter sido adotada durante a escrita do código, rotulada como B4. Além disso, a importação de bibliotecas não utilizadas, rotulada como A5, também foi uma falha recorrente nas atividades, indicando um possível problema de redundância e repetição desnecessária de linhas no código nos testes avaliados.

4.2. QP1: Quais são os principais *misconceptions* identificados em cada módulo da disciplina?

Ao organizar cada registro de *misconceptions*, rotulados com as categorias criadas por [Silva et al. 2023a], em grupos divididos pelos 6 módulos da disciplina, foi possível montar uma tabela comparativa com a quantidade de rótulos obtidos em cada registro avaliado.

Tabela 3. Cinco principais *misconceptions* identificados em cada módulo da disciplina, nos dois cursos analisados.

Módulos	Engenharia Mecânica	Licenciatura em Matemática
M1	A1 (3), A3 (6), A5 (17), H1 (2), H2 (3)	A1 (2), A3 (1), A5 (2), B8 (1), H3 (1)
M2	B10 (2), B11 (2), B4 (22), B8 (5), X2 (3)	A1 (3), A3 (2), B11 (2), B4 (5), X2 (4)
M3	B11 (3), B4 (52), B8 (4), B9 (3), E1 (3)	B1 (3), B10 (2), B11 (3), B4 (15), B8 (14)
M4	A2 (3), B4 (10), B8 (3), C2 (5), G4 (5)	A1 (1), A2 (1), B8 (1), X2 (2), X3 (2)
M5	A2 (12), A5 (15), B11 (10), B8 (14), C3 (11)	A1 (1), A5 (3), A6 (1), E1 (1), X1 (2)
M6	A2 (11), A5 (17), B10 (13), C3 (10), E1 (7)	A1 (2), A5 (4), A6 (5), H1 (4), X2 (3)

Ao analisar a Tabela 3, é perceptível que o surgimento de *misconceptions* no primeiro módulo é reduzido. Isso pode ser atribuído à baixa complexidade do conteúdo abordado e das avaliações propostas, com registros limitados a importações e variáveis não utilizadas. Nos módulos 2 e 3, que tratam de estruturas de decisão simples e aninhada, respectivamente, observa-se uma crescente de *misconceptions*, frequentemente associados a ausência ou redundância das estruturas estudadas.

Nos módulos finais, que abordam estruturas de repetição por condição e por contagem, e uso de vetores, há mudanças no padrão de surgimento de *misconceptions*. Essa reflete na diferente complexidade do conteúdo proposto. Ainda assim, as categorias de problemas detectadas nessas questões mostram semelhanças entre si.

4.3. QP2: As categorias utilizadas para rotular os *misconceptions* foram suficientes para abranger todos os problemas encontrados?

As categorias desenvolvidas por [Silva et al. 2023a] abrangeram grande parte dos registros de código analisados. No entanto, durante a pesquisa, algumas questões foram identificadas como *misconceptions*, pois o estudante tentou utilizar de conceitos válidos para a resolução de problemas mas com falhas na sua concepção, que não se enquadravam em nenhum dos rótulos existentes. Após análise e alinhamento entre os pesquisadores, constatou-se a necessidade de criar novos rótulos para representar adequadamente esses casos específicos. Assim, foram criadas as categorias descritas na Tabela 4.

Diferente de trabalhos como de [Araujo et al. 2021], que baseou seus resultados em um catálogo existente de pesquisas anteriores sem modificações marcantes, o presente trabalho buscou alinhar as categorias de [Silva et al. 2023a] com os novos rótulos de acordo com a necessidade de classificação, e assim possibilitar que nenhuma lacuna de aprendizado fique despercebida na busca por *misconceptions*. Além disso, expandiu os achados na área de estudo com essas categorias, que podem ser utilizadas em trabalhos futuros para classificar resultados similares.

Tabela 4. Misconceptions identificados neste trabalho, não catalogados por [Silva et al. 2023b]

ID	Nome	Descrição do *misconception*
X1	Uso incorreto de estrutura	Aplicação de uma estrutura (ex.: *if*) para realizar uma tarefa que seria mais apropriada para outra estrutura (ex.: *for*), comprometendo a clareza ou eficiência do código.
X2	Operações inválidas com variáveis de tipos diferentes	Uso de operações entre tipos incompatíveis de variáveis que a linguagem permite, mas que poderiam ser realizadas mais adequadamente.
X3	Controle de prioridade de operações indevido	Realização da operação correta solicitada, mas com uso inadequado de operadores ou ordem de precedência, gerando resultados inesperados.
X4	Função aplicada como método ou vice-versa	Uso da função ou método certo para a resolução da questão, entretanto de maneira indevida, tratando-os de maneira contrária.

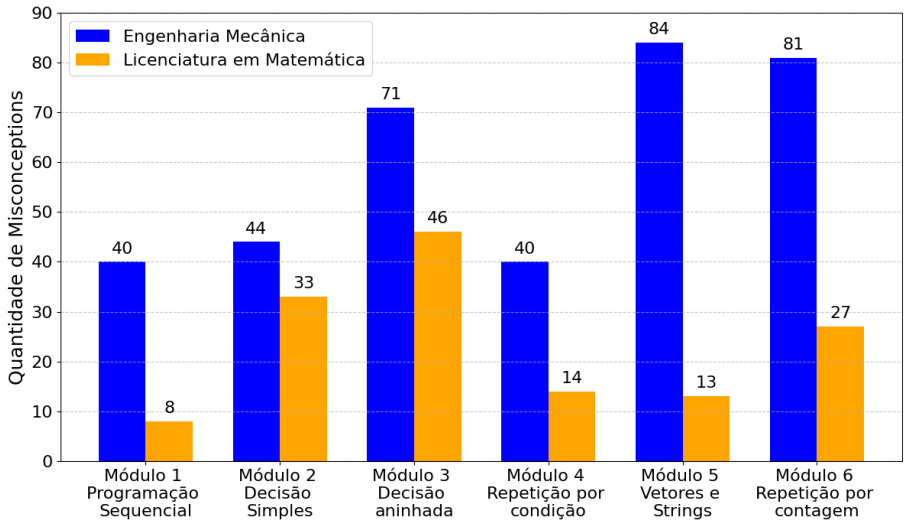

Figura 2. Quantidade de *misconceptions* por módulo da disciplina

4.4. QP3: A permanência desses *misconceptions* ao longo da disciplina gerou variações ou evoluções dessas compreensões imperfeitas?

Durante a pesquisa, foi perceptível que, ao longo dos módulos, alguns *misconceptions* foram perpetuados e estenderam-se para outros testes avaliativos, aumentando gradativamente seus registros, como mostrado na Figura 2. Ao verificar as categorias dessas compreensões imperfeitas, constatou-se que eles correspondiam a grupos de rótulos relacionados ao conteúdo principal dos módulos. Essa correspondência é coerente com seu surgimento e, em módulos posteriores, com conteúdos similares, mas de maior complexidade, o aparecimento desses *misconceptions* foi acompanhado do surgimento de novos rótulos pertencentes ao mesmo grupo.

No segundo módulo da disciplina, que aborda estruturas de decisão simples, o *misconception* predominante foi catalogado como B4. Esse rótulo corresponde a repetições desnecessárias de operações dentro de blocos *if-else*, como a duplicação de uma mesma

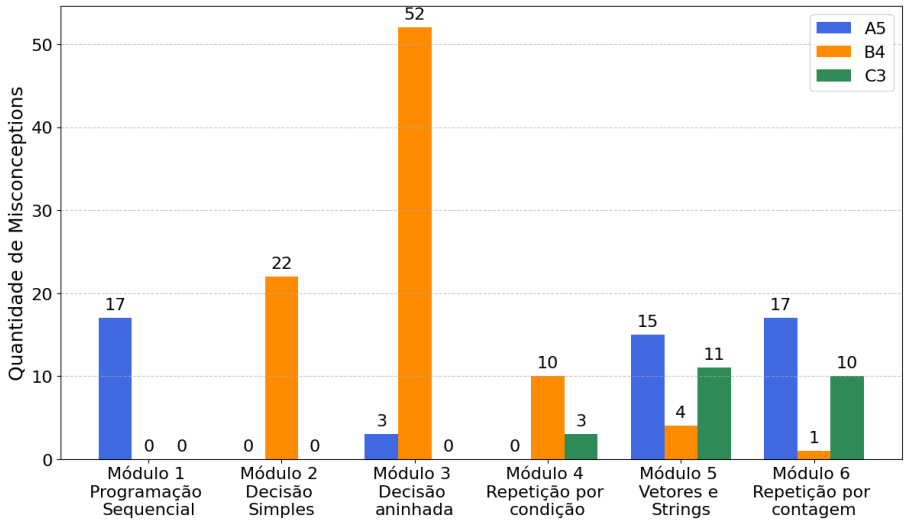

Figura 3. Surgimento e persistência de três exemplos de *misconceptions* ao longo da disciplina, na turma de Engenharia Mecânica

linha de código em ambos os blocos, situação que poderia ser evitada por meio de uma abordagem mais eficiente e estratégica. No terceiro módulo, ainda relacionado a estrutura de decisão, mas de forma aninhada, evidenciando maior complexidade, o número de ocorrências desse tipo de *misconception* aumentou, evidenciando uma persistência dessa problemática. Além disso, foram identificados outros problemas de concepção relacionados a estruturas de decisão, como a repetição desnecessária dessas estruturas devido à falta de uso adequado de operadores lógicos nos códigos testados ou submetidos.

Adiante na pesquisa, foi possível observar que, no quarto módulo, houve uma redução nos *misconceptions* detectados. Essa redução pode ser atribuída à mudança de conteúdo, que passou a abordar estruturas de repetição por condição, como o *while*, alterando também as categorias de erros cometidos pelos estudantes. Nesse módulo, os *misconceptions* passaram a estar mais relacionados à redundância nos *loops*, enquanto os problemas ligados a blocos *if-else* se tornaram menos frequentes. No entanto, nos módulos seguintes, especialmente no sexto e último, os problemas de concepção relacionados às categorias de repetição aumentaram significativamente.

Outro comportamento observado foi o retorno de problemas de compreensão associados às categorias mais básicas, como a declaração de variáveis e importações não utilizadas. Isso ocorreu porque, nos dois últimos módulos, novas bibliotecas da linguagem de programação foram introduzidas, e os problemas iniciais não foram completamente resolvidos. Como resultado, surgiram novamente esses *misconceptions*, agora em conjunto com aqueles relacionados às estruturas de repetição abordadas nesses módulos. Essa combinação gerou *misconceptions* detectados simultaneamente, devido a uma série de estruturas e operações redundantes herdadas de módulos anteriores que permaneceram sem solução. Esse comportamento observado ao longo da pesquisa possibilitou a elaboração do gráfico apresentado na Figura 3. O gráfico facilita a análise da evolução dos *misconceptions* em relação ao conteúdo abordado nos módulos, evidenciando que o surgimento de compreensões imperfeitas de conceitos básicos tendem a se intensificar à medida que não são solucionadas e módulos de maior complexidade são introduzidos.

5. Limitações

Durante o trabalho, foram identificadas algumas limitações que podem ter influenciado os resultados. Uma delas são as questões deixadas em branco ou incompletas pelos alunos, impossibilitando mapear os *misconceptions* desses estudantes. A maioria dos alunos que desistem das avaliações são de Licenciatura em Matemática, o que trouxe a falsa impressão que houve menos *misconceptions* nesse curso do que em Engenharia Mecânica. É possível assumir que este fato tenha relação com a identificação dos alunos com o curso de Licenciatura, dado que sua nota de corte é mais baixa.

Outra limitação foi a remoção de um sétimo módulo da disciplina, cujo tema era manipulação de matrizes, e que não foi abordado em 2023 devido à redução do calendário acadêmico da universidade estudada, em virtude da pandemia de Covid-19. A inclusão deste módulo poderia enriquecer a pesquisa com a inserção de mais um ponto de medição temporal e o estudo da relação desse módulo com o M5, que abordou vetores e strings.

Por fim, outra limitação relevante foi a falta de rotulação para certos *misconceptions* não encontrados no trabalho de [Silva et al. 2023a], mas que foram identificados neste, tornando necessária a criação de rótulos adicionais para prosseguir com a pesquisa, como citado na Seção 4.4. Um motivo provável para o ocorrido é a diferença de grupos amostrais e questões analisadas durante as duas pesquisas, pois no estudo atual também houve itens do catálogo original que não foram encontrados.

6. Considerações finais

Após analisar os *misconceptions* presentes em 498 códigos de 83 estudantes de programação introdutória ao longo de seis avaliações parciais, reuniram-se importantes observações de natureza longitudinal.

Os estudantes de Engenharia Mecânica apresentaram maior taxa de aprovação na disciplina, mas também geraram mais *misconceptions*. Esse fato pode estar relacionado à maior quantidade de testes e submissões realizadas durante as avaliações, bem como na maior taxa de persistência na disciplina. Em contraste, os estudantes de Licenciatura em Matemática registraram maiores taxas de reprovação, mas tiveram um número menor de *misconceptions* detectados, o que pode ser explicado pela menor quantidade de códigos submetidos para avaliação.

Por fim, a análise temporal dos *misconceptions* revelou certos padrões, como a permanência e o agravamento de compreensões imperfeitas da mesma categoria. Conforme discutido na Seção 4.4, a ausência de correções de dúvidas em módulos básicos tende a intensificar o surgimento e a evolução de *misconceptions* em módulos mais avançados de conteúdo similar. Isso ocorre devido à introdução de maior complexidade, que contribui para a persistência de dificuldades anteriores e o aparecimento de novas. Na Figura 2, é possível verificar que, no terceiro módulo, em que a complexidade das estruturas de decisão aumenta, o número de *misconceptions* detectados também cresce.

Outro comportamento detectado está relacionado ao surgimento de *misconceptions* em grupos nos módulos mais complexos da disciplina. Durante a análise do módulo final, foi perceptível o registro de compreensões imperfeitas de categorias diferentes, agrupadas em momentos similares nas questões, como em aplicações redundantes de estruturas de decisão dentro de estruturas de repetição, conteúdo principal do módulo, que

também apresentavam falhas na aplicação da lógica. Isso mostra que, com o passar do tempo, *misconceptions* não solucionados de determinado conteúdo podem surgir aplicados em questões de conteúdos diferentes, o que pode dificultar a correção de dúvidas principais daquele módulo estudado.

Agradecimentos

Esta pesquisa, realizada no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER), de acordo com o Artigo 39 do Decreto n°10.521/2020, foi financiada pela Samsung Eletrônica da Amazônia Ltda, nos termos da Lei Federal n°8.387/1991, através do convênio 001/2020 firmado com a UFAM e FAEPI, Brasil. Também recebeu apoio do Conselho Nacional de Desenvolvimento Cientifico e Tecnologico - CNPq (Processo 303443/2023-5).

Referências

Araujo, A., Zordan Filho, D. L., Oliveira, E. H. T., Carvalho, L. S. G. C., Pereira, F. D., and Oliveira, D. B. F. (2021). Mapeamento e análise empírica de misconceptions comuns em avaliações de introdução à programação. In *Simpósio Brasileiro de Educação em Computação (EduComp)*, pages 123–131. SBC.

Caceffo, R., de França, B., Gama, G., Benatti, R., Aparecida, T., Caldas, T., and Azevedo, R. (2017). An antipattern documentation about misconceptions related to an introductory programming course in c. In *Technical Report 17-15*, page 42. Institute of Computing, University of Campinas.

Gama, G., Caceffo, R., Souza, R., Bennati, R., Aparecida, T., Garcia, I., and Azevedo, R. (2018). An antipattern documentation about misconceptions related to an introductory programming course in Python. *Institute of Computing, University of Campinas, Tech. Rep. IC-18-19*, page 106.

Gomes, J. A. (2010). *Dificuldades de aprendizagem de programação de computadores: contributos para a sua compreensão e resolução*. PhD thesis, Universidade de Coimbra (Portugal).

Lima, M. A., Carvalho, L. S., Oliveira, E. H., Oliveira, D. B., and Pereira, F. D. (2021). Uso de atributos de código para classificar a dificuldade de questões de programação em juízes online. *Revista Brasileira de Informática na Educação*, 29:1137–1157.

McCall, D. and Kölling, M. (2014). Meaningful categorisation of novice programmer errors. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. IEEE.

Qian, Y. and Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–24.

Silva, E. P., Caceffo, R., and Azevedo, R. (2023a). When test cases are not enough: Identification, assessment, and rationale of misconceptions in correct code (MC³). *Revista Brasileira de Informática na Educação*, 31:1165–1199.

Silva, E. P., Caceffo, R. E., and Azevedo, R. (2023b). Passar nos casos de teste é suficiente? identificação e análise de problemas de compreensão em códigos corretos.

In *Simpósio Brasileiro de Educação em Computação (EDUCOMP)*, pages 119–129. SBC.

Sorva, J. (2012). *Visual program simulation in introductory programming education*. PhD thesis.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Trans. Comput. Educ*, 13(2):1–31.

Teague, D. and Lister, R. (2014). Programming: reading, writing and reversing. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 285–290.