Mensuração da carga cognitiva em um curso de programação introdutória para crianças

Esteic Janaina Santos Batista

PÓS-GRADUAÇÃO FACOM UFMS

Data de Depósito:

Assinatura:_____

Mensuração da carga cognitiva em um curso de programação introdutória para crianças

Esteic Janaina Santos Batista

Orientador: *Prof. Dr Amaury Antônio de Castro Jr.*
Coorientador: *Prof. Dr Anderson Corrêa de Lima*

Dissertação apresentada à Faculdade de Computação (FACOM) da Universidade Federal de Mato Grosso do Sul (UFMS) como parte dos requisitos necessários à obtenção para do título em Mestre em Computação Aplicada.

UFMS - Campo Grande
junho/2023

À minha filha
Mayara,

Aos meus pais,
Elcio e Enedina,

Às minhas irmãs,
Joyce e Rosângela,

Ao meu companheiro,
Felippi.

Agradecimentos

Gostaria de dedicar este capítulo de agradecimentos às pessoas que estiveram ao meu lado durante toda a jornada da minha dissertação, oferecendo apoio, incentivo e compreensão. Sem o apoio dessas pessoas especiais, não teria sido possível chegar até aqui e realizar o sonho de iniciar o mestrado.

Primeiramente, quero expressar minha profunda gratidão aos meus pais, Enedina e Elcio. Vocês sempre estiveram ao meu lado, fornecendo amor, apoio incondicional e encorajamento em todos os momentos. Agradeço por compreenderem e apoiarem minha decisão de mudar de cidade, abraçando uma nova jornada que inclui um novo emprego e o início do mestrado. Seu amor e suporte constantes foram fundamentais para que eu persistisse diante dos desafios e dificuldades que surgiram ao longo do caminho.

Agradeço ao meu companheiro, Felippi, por sua paciência, incentivo e apoio desde o momento em que eu estava cursando disciplinas do mestrado. Sua presença ao meu lado, ajudando-me a superar dúvidas, incentivando-me a nunca desistir, mesmo em meio a tantos desafios, e cuidando da nossa princesa Mayara em tantas tardes de final de semana, permitindo-me concentrar nos estudos, foi crucial para meu progresso. Sua dedicação e apoio constantes foram um pilar fundamental ao longo dessa jornada.

Gostaria de dedicar um agradecimento especial à minha amada filha, Mayara. Seu sorriso, sua energia contagiante e seu amor incondicional foram uma fonte constante de motivação durante toda a jornada do mestrado. Mesmo nos momentos em que precisava me dedicar aos estudos, você compreendeu e demonstrou compaixão, tornando-se minha maior aliada nessa busca pelo conhecimento. Sou profundamente grata por ter você como minha filha e por todo o amor e alegria que você trouxe à minha vida. Seu apoio e compreensão foram inestimáveis e fizeram toda a diferença para o meu sucesso.

Gostaria de expressar minha sincera gratidão ao meu orientador, professor Amaury. Tenho o prazer de chamá-lo de meu pai acadêmico, pois ele tem sido meu mentor desde o início da graduação, atuando como meu tutor no

Programa de Educação Tutorial (PET) Fronteira. Compartilhamos uma paixão mútua pelo NERDS (Núcleo Educacional de Robótica e Desenvolvimento de Software), e foi através dele que descobri meu amor pela área de Educação em Computação. Sua orientação, apoio e incentivo foram fundamentais para que eu não desistisse durante essa jornada. Além de ser um excelente profissional, você também se mostrou um amigo e um ser humano excepcional, e isso fez toda a diferença.

Agradeço também ao meu coorientador, Anderson, por seu incentivo constante e por acreditar que este dia estaria próximo. Sua presença trouxe leveza a todo o processo, e sou grata por compartilhar não apenas uma relação profissional, mas também uma amizade durante toda essa jornada.

Gostaria de estender meus agradecimentos aos vários amigos pessoais e de trabalho que me apoiaram ao longo desse período desafiador. Suas palavras de encorajamento, apoio e compreensão foram inestimáveis, especialmente nos momentos em que precisei me ausentar para coletar dados para minha pesquisa. Entre esses amigos especiais, gostaria de agradecer a Quésia, parceira, amiga e colega de mestrado. Compartilhamos momentos bons e ruins ao longo dessa jornada, e sua amizade e companheirismo foram inestimáveis.

Agradeço à Escola Estadual Enrique Cyrillo em nome do diretor Fabiano Francisco Soares por abrir suas portas e permitir que eu aplicasse o curso para os alunos da escola, o que possibilitou a coleta de dados para minha pesquisa. A colaboração e apoio da instituição foram fundamentais para o sucesso deste trabalho, e sou grata pela oportunidade.

Por fim, gostaria de expressar minha gratidão aos professores da banca Dra. Graziela Santos de Araújo (FACOM/UFMS) e Dr. Ismar Frango Silveira (Mackenzie/Cruzeiro do Sul) por dedicarem seu tempo e conhecimento para avaliar meu trabalho e contribuírem para o aprimoramento desta dissertação. Seus comentários e sugestões foram valiosos para o desenvolvimento deste estudo.

A todas que mencionei aqui e àquelas que, porventura, não foram citadas, mas que contribuíram de alguma forma para minha jornada acadêmica, meu mais sincero obrigado. Seu apoio e encorajamento foram essenciais para que eu alcançasse essa etapa importante em minha vida. Estou verdadeiramente grata e honrada por ter vocês ao meu lado.

Abstract

The ability of a student to learn a concept is directly related to how much cognitive load is used to comprehend the material. The Cognitive Load Theory (CLT) explains how a person's learning is hindered when the limited capacity of working memory is exceeded during the process. Therefore, the use of instruments to measure cognitive load in the classroom is essential. This study presents and discusses an introductory programming course planned based on the pedagogical recommendations of CLT and Scaffolding Learning. Additionally, a previously developed instrument for measuring cognitive load was adapted to meet the updates of CLT and the age range of the course's target audience. The results show that the assessment of cognitive load is a valuable tool for understanding the perceived difficulty of students in introductory programming activities. The proposed learning path proved to be effective in reducing the perceived cognitive load of students, resulting in a better understanding of the concepts covered in the course.

Resumo

A capacidade de aprendizado de um aluno está diretamente relacionada à quantidade de carga cognitiva utilizada para compreender o material. A Teoria da Carga Cognitiva (TCC) explica como o aprendizado de uma pessoa é prejudicado quando a capacidade limitada da memória de trabalho é excedida durante o processo. Portanto, o uso de instrumentos para medir a carga cognitiva em sala de aula é essencial. Este estudo apresenta e discute um curso de programação introdutória planejado com base nas recomendações pedagógicas da TCC e Aprendizagem por Andaimes (*Scaffolding Learning*). Além disso, um instrumento desenvolvido anteriormente para medir a carga cognitiva foi adaptado para atender às atualizações da TCC e à faixa etária do público-alvo do curso. Os resultados mostram que a avaliação da carga cognitiva é uma ferramenta valiosa para entender a dificuldade percebida pelos alunos em atividades de programação introdutória. A trilha de aprendizagem proposta se mostrou eficaz na redução da carga cognitiva percebida pelos alunos, resultando em uma melhor compreensão dos conceitos abordados no curso.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BNCC Base Nacional Comum Curricular

PC Pensamento Computacional

SBC Sociedade Brasileira de Computação

TCC Teoria da Carga Cognitiva

CAPÍTULO

1

Introdução

Com o rápido avanço da tecnologia, profissionais que dominam a programação serão cada vez mais necessários, seja para atuar na área, para entender o mundo ao nosso redor, ser cidadãos conscientes em um mundo cada vez mais digital ou para enriquecer outras áreas que usam cada vez mais esse conhecimento. Por esse motivo, disciplinas de programação, que antes eram específicas para cursos de computação, estão sendo introduzidas gradualmente no ensino básico.

A programação introdutória é um curso para iniciantes que envolve habilidades de resolução de problemas, conceitos básicos de programação, a sintaxe e a semântica de uma linguagem de programação, além de ensinar como formular soluções com o seu uso.

Nos últimos anos, tem havido um aumento no número de publicações e discussões sobre a programação introdutória. Isso se deve, em grande parte, às dificuldades que tanto os professores quanto os alunos enfrentam ao ensinar e aprender programação para iniciantes. Essas dificuldades são frequentemente citadas como um dos principais fatores que contribuem para altas taxas de evasão em cursos de computação, não apenas no Brasil, mas em todo o mundo. Conforme de Holanda et al. (2019) e Hermans (2020) essas dificuldades envolvem:

- compreender e aplicar os conceitos de programação em algoritmos para resolver problemas concretos;
- desenvolver habilidades e competências do pensamento computacional;
- compreender a sintaxe das linguagens de programação.

Hermans (2020) afirma que aprender uma linguagem de programação tem muitas semelhanças com a aprendizagem de uma língua natural, uma vez que os alunos precisam aprender sua sintaxe e semântica. Portanto, a educação em programação pode ser aprimorada pelo uso de estratégias instrucionais comuns no ensino de linguagem natural. Isso pode incluir o ensino gradual da linguagem de programação, começando com as letras maiúsculas e minúsculas, a pontuação e outros aspectos semelhantes.

Segundo Duran et al. (2022), a Teoria da Carga Cognitiva tem se tornado cada vez mais popular na comunidade de Educação em Computação, pois explica como o aprendizado pode ser afetado pelo gargalo da memória de trabalho humana e como o ensino pode superar essa limitação.

A estratégia de ensino gradual apresentada no trabalho de Hermans (2020) pode ser aplicada ao ensino de programação, pois a carga cognitiva dos alunos durante a programação pode ser sobrecarregada pelo ensino simultâneo de conceitos de programação, sintaxe e resolução de problemas. De fato, o processo de aprendizagem de programação pode ser semelhante ao processo de ensino de crianças mais jovens a escrever. Com base nas recomendações da Teoria da Carga Cognitiva, a autora propõe a ferramenta Hedy para ensinar programação de forma gradual. Nessa ferramenta, a sintaxe e os conceitos de programação são apresentados gradualmente em uma trilha de aprendizagem com desafios divididos por níveis. Cada nível incrementa uma nova regra na sintaxe e/ou conceito de programação até que, ao final, o aluno esteja programando em uma linguagem muito próxima ao Python.

Uma outra teoria que pode auxiliar neste processo de ensino é do *Scaffolding Learning* (Aprendizagem por Andaimes) que consiste em um suporte temporário de um tutor durante o processo de aprendizagem na resolução de problemas em que os estudantes inicialmente não conseguiriam realizar de forma desassistida. Ela já foi utilizada em contextos de ensino de linguagens, como apresentado no trabalho de Gibbons (2002). Considerando a proximidade do ensino de uma linguagem natural e que o ensino de programação envolve a habilidade de resolução de problemas, o *Scaffolding Learning* aprensenta-se como suporte ainda pouco explorado no processo de ensino de linguagem de programação.

Este trabalho apresenta uma ferramenta adaptada para mensurar a carga cognitiva dos estudantes durante um curso de programação introdutória. Para isso, o curso foi desenvolvido levando em consideração o ensino gradual de linguagem de programação defendido por Hermans (2020) e as recomendações da Teoria da Carga Cognitiva e Aprendizagem por Andaimes, que serão detalhadas ao longo deste trabalho.

O objetivo deste trabalho é avaliar a eficácia da ferramenta proposta na

mensuração da carga cognitiva de estudantes de 6º ao 9º ano do Ensino Fundamental durante o curso de programação introdutória. Para tanto, utilizamos a ferramenta Hedy e a linguagem de programação Python com o Google Colab. Os cursos foram planejados com base nas recomendações da Teoria da Carga Cognitiva e Aprendizagem por Andaimes, que serão discutidas em detalhes ao longo deste trabalho.

1.1 Motivação

A motivação para este trabalho surge da crescente importância da programação no contexto atual. Sendo assim, profissionais com habilidades em programação são cada vez mais demandados. Além disso, a programação também oferece oportunidades para o desenvolvimento do pensamento lógico, da criatividade e da resolução de problemas.

Diante desses desafios, é necessário buscar abordagens e estratégias pedagógicas eficazes para o ensino de programação introdutória. A abordagem gradual de ensino, que introduz os conceitos e a sintaxe da linguagem de programação de forma progressiva, tem se mostrado promissora. Além disso, teorias como a Teoria da Carga Cognitiva e a Aprendizagem por Andaimes oferecem insights valiosos sobre como melhorar o processo de ensino e aprendizagem.

A motivação principal deste estudo é a busca por estratégias e ferramentas que possam tornar o ensino de programação introdutória mais eficaz, acessível e motivador. Através da mensuração da carga cognitiva dos alunos ao longo do curso, pretendemos validar a eficácia da trilha de aprendizagem proposta e contribuir para a melhoria da qualidade do ensino de programação no Ensino Fundamental 2.

1.2 Objetivos

O objetivo geral deste trabalho é validar um instrumento adaptado para mensurar a carga cognitiva dos alunos durante um curso de programação introdutória. Para alcançar esse objetivo, foram delimitados os seguintes objetivos específicos:

- Propor uma trilha de aprendizagem para iniciantes em cursos de programação introdutória, com base nas teorias da Aprendizagem por Andaimes e da Carga Cognitiva.

- Definir recomendações pedagógicas para abordar cada conjunto de conceitos de programação em cursos introdutórios.

- Analisar a aplicação das ferramentas Hedy e da linguagem Python no Ensino Fundamental 2.

1.3 Principais Contribuições

As principais contribuições deste trabalho são:

- Uma trilha de aprendizagem adaptada para o ensino de programação introdutória, seguindo a abordagem de ensino gradual, com base nas recomendações da Teoria da Carga Cognitiva e da Aprendizagem por Andaimes.
- Recomendações pedagógicas para abordar cada conjunto de conceitos de programação de forma eficaz e acessível a estudantes iniciantes na aprendizagem de programação.
- Avaliação da eficácia da ferramenta Hedy e da linguagem Python no ensino de programação introdutória no Ensino Fundamental 2.

1.4 Organização

O restante deste trabalho está organizado da seguinte forma: no Capítulo 2, apresentaremos o referencial teórico, abordando as principais dificuldades no ensino e aprendizagem de programação introdutória, a Teoria da Carga Cognitiva, a Aprendizagem por Andaimes e as ferramentas utilizadas neste estudo. No Capítulo 3, descreveremos a metodologia empregada, incluindo os objetivos do estudo, os materiais e métodos utilizados, a escolha das áreas de estudo, os participantes e os procedimentos para coleta e análise de dados.

Os resultados obtidos serão apresentados no Capítulo 4, destacando a aplicação da ferramenta Hedy e da linguagem Python no ensino de programação introdutória no Ensino Fundamental 2. No Capítulo 5, realizaremos uma discussão dos resultados à luz da literatura revisada, analisando as implicações para o ensino de programação introdutória e apresentando sugestões para pesquisas futuras. Por fim, no Capítulo 6, faremos uma síntese dos principais achados do estudo, destacando suas contribuições e possíveis limitações, além de apresentar uma visão geral das principais conclusões obtidas.

CAPÍTULO

2

Revisão da literatura

Essa seção apresenta uma revisão sobre diversos aspectos relacionados ao ensino de programação, com o objetivo de embasar a pesquisa realizada pela autora. Inicialmente, são abordadas as dificuldades encontradas no processo de ensino e aprendizagem de programação, como a falta de motivação dos alunos, a complexidade dos conceitos e a falta de habilidades prévias necessárias para a compreensão do conteúdo. Em seguida, é apresentada a teoria da carga cognitiva e sua evolução, que busca entender como o cérebro humano processa informações e como isso pode ser aplicado no ensino de programação.

Além disso, a seção também aborda recomendações pedagógicas na teoria da carga cognitiva, que visam reduzir a sobrecarga cognitiva dos alunos durante o processo de aprendizagem. São apresentadas estratégias como a divisão do conteúdo em partes menores, a utilização de exemplos concretos e a redução da quantidade de informações apresentadas de uma só vez. Por fim, são discutidas ferramentas de programação introdutória, com destaque para a linguagem Python, que tem se mostrado uma opção interessante para o ensino de programação para crianças e jovens.

2.1 Ensino de Programação

De acordo com França and Tedesco (2015) e Schoeffel et al. (2015) alguns países têm adotado o ensino de computação nas escolas com o objetivo de desenvolver a capacidade dos estudantes de resolver problemas, melhorar suas habilidades e criar soluções tecnológicas em benefício da sociedade. A programação surge como uma nova possibilidade de utilização de computadores,

tablets e outros recursos em aula, permitindo que os estudantes aprendam de forma diferente, criando jogos, aplicativos, histórias, animações ou resolvendo desafios com o uso de linguagens de programação.

No Brasil, o ensino de programação para crianças e jovens já faz parte de diversas iniciativas com o objetivo de desenvolver habilidades como raciocínio lógico, autonomia, pensamento crítico, colaboração, trabalho em equipe, empatia e capacidade de resolver problemas complexos que possam surgir em suas vidas.

A Sociedade Brasileira de Computação (SBC) recentemente aprovou normas sobre Computação na Educação Básica – Complemento à Base Nacional Comum Curricular (BNCC). Essas normas apresentam competências e premissas específicas da computação na BNCC para cada fase escolar. O documento, apresentado à comunidade em 2019 [Sociedade Brasileira de Computação (2019)], concebe a Computação para Educação Básica, formada por três eixos fundamentais: Mundo Digital, Cultura Digital e Pensamento Computacional (PC), que está intrinsecamente relacionado ao ensino de conceitos de programação, conforme Ministério da Educação (2022b) e Ministério da Educação (2022a).

O Pensamento Computacional é uma habilidade fundamental para a resolução de problemas, o desenvolvimento de sistemas e a compreensão do comportamento humano. Ele combina conceitos da computação com o uso de ferramentas e recursos computacionais para resolver problemas, empregando o pensamento crítico e os fundamentos da computação na criação de estratégias e artefatos para lidar com a complexidade e gerar soluções inovadoras. Além disso, o Pensamento Computacional ajuda os estudantes a resolver problemas de forma autônoma e a compreender o contexto tecnológico atual, segundo Wing (2006).

De acordo com Brackmann (2017), o Pensamento Computacional baseia-se em quatro pilares interdependentes para atingir seu objetivo principal de resolução de problemas, sendo: decomposição, reconhecimento de padrões, abstração e algoritmo. A decomposição consiste em dividir um problema em partes menores para analisá-las separadamente e tornar a solução mais fácil. O reconhecimento de padrões envolve identificar padrões para resolver problemas de forma rápida e eficiente, fazendo uso de soluções previamente definidas em outros problemas e com base em experiências anteriores. A abstração é o processo de separar detalhes irrelevantes para se concentrar nas informações importantes. O algoritmo é um conjunto de instruções claras, necessárias e organizadas para solucionar um problema específico .

Portanto, é essencial que os cursos introdutórios de programação tenham como premissa o desenvolvimento do Pensamento Computacional, utilizando

linguagens de programação apropriadas para o público-alvo. Isso permitirá que os estudantes compreendam e apliquem efetivamente os quatro pilares do Pensamento Computacional, adquirindo habilidades valiosas para a resolução de problemas e o desenvolvimento de soluções inovadoras.

2.2 Dificuldades no Ensino e Aprendizagem de Programação

O ensino e aprendizagem de programação apresentam desafios significativos que foram objeto de discussão em várias pesquisas nas últimas décadas. O estudo secundário realizado por Medeiros et al. (2018) discute as principais dificuldades enfrentadas pelos alunos, que são apresentadas a seguir.

Um dos principais desafios enfrentados pelos alunos é a resolução de problemas, que foi citada como um desafio de aprendizagem. Outras dificuldades incluem motivação, engajamento e dificuldades em aprender a sintaxe de linguagens de programação. Para muitos alunos, aprender a sintaxe e a semântica de uma linguagem ao mesmo tempo pode ser sobrecarregante. Combinar conhecimentos novos e anteriores e desenvolver habilidades gerais de resolução de problemas também é um desafio envolvendo: habilidades matemáticas (compreender o contexto de um problema e identificar as informações-chave para elaborar uma solução), pensamento crítico e capacidade de argumentação, criatividade e conhecimento de inglês.

Os alunos enfrentam dificuldades com os erros de sintaxe, que são comuns em programadores iniciantes. Corrigir esses erros pode ser um processo demorado e muitas vezes leva a um comportamento de depuração aleatório, influenciado pelo fato de que os alunos muitas vezes não entendem as mensagens do compilador. Além disso, os alunos enfrentam dificuldades com os conceitos de programação, como estruturas de controle e estruturas de dados na escrita do código.

As dificuldades comportamentais também são um desafio para os alunos, incluindo aspectos sociais e emocionais que podem ter impacto na aprendizagem, como motivação, envolvimento e confiança (incluindo a percepção de que a programação é difícil), bem como hábitos de estudo e gerenciamento de tempo.

Por outro lado, os docentes enfrentam desafios no ensino de programação introdutória devido à falta de métodos e ferramentas adequadas, bem como à heterogeneidade de conhecimento dos alunos, o que dificulta o dimensionamento e o ensino personalizado.

Portanto, o ensino de programação introdutória deve superar esses desafios para garantir uma experiência de aprendizado bem-sucedida para os alunos.

Isso pode incluir o desenvolvimento de estratégias para aprimorar as habilidades de resolução de problemas, o uso de métodos de ensino personalizados para atender às necessidades individuais dos alunos e a adoção de ferramentas de ensino adequadas.

2.3 Teoria da Carga Cognitiva e sua evolução

A Teoria da Carga Cognitiva (TCC) foi proposta pelo psicólogo educacional australiano John Sweller em 1988, Sweller (1988). Essa teoria se baseia no conhecimento sobre a cognição humana para produzir diretrizes e técnicas embasadas em dezenas de estudos e pesquisas experimentais que comprovam que a aprendizagem ocorre de maneira mais eficaz em condições alinhadas com a arquitetura cognitiva humana, implicando, por exemplo, em ambientes de aprendizagem eficientes.

A carga cognitiva está relacionada com a quantidade de informação que a memória de trabalho pode armazenar simultaneamente. Ela parte do pressuposto de que todo processamento consciente de informação ocorre na Memória de Trabalho, a qual possui limitações tanto na duração quanto na capacidade. Portanto, os métodos de instrução devem evitar sobrecarregar a memória de trabalho com atividades adicionais que não contribuam diretamente para a aprendizagem.

Segundo Sweller (1988) todo material instrucional impõe uma carga cognitiva, e essa pode ser dividida em duas categorias independentes - intrínseca e estranha. A carga cognitiva intrínseca e estranha são aditivas e, juntas, determinam a carga cognitiva total. Se essa carga cognitiva exceder a capacidade da memória de trabalho, o processamento da informação, incluindo o aprendizado, será comprometido.

O conceito-chave da TCC é a interatividade do elemento, que se refere a quantos elementos uma situação exige que o aluno processe simultaneamente na memória de trabalho. Durante uma aula, isso reflete no grau em que esses elementos estão interconectados no sentido de que eles devem ser comparados, contrastados, integrados ou processados conscientemente juntos, a partir dos quais surge a carga cognitiva. Ou seja, muita interatividade de elementos significa sobrecarga cognitiva e aprendizagem malsucedida.

A carga intrínseca, de acordo com Sweller (2010), refere-se à complexidade natural das informações que precisam ser compreendidas e do material que deve ser aprendido para alcançar um determinado objetivo de aprendizagem. Em outras palavras, é a quantidade de esforço mental necessário para assimilar o conteúdo em questão. Quando nos propomos a aprender algo novo, essa carga intrínseca é inevitável, pois faz parte do processo de aquisição de

conhecimento.

Entretanto, o nível de conhecimento prévio que os alunos possuem pode afetar a carga intrínseca percebida. Quando os estudantes já possuem algum conhecimento relacionado ao tópico em questão, isso pode reduzir a complexidade percebida, tornando o aprendizado mais fácil e menos exigente. Isso ocorre porque o conhecimento prévio permite que os alunos organizem as informações de forma mais eficiente, podendo agrupar conceitos relacionados ou reconhecer padrões, o que facilita a compreensão e a retenção de novas informações.

A carga estranha, ao contrário, envolve o processamento cognitivo que não é necessário para a aprendizagem, como conteúdo desnecessário em materiais de aprendizagem, conteúdo que não é fácil de acessar durante a resolução de um problema, mas precisa ser mantido em mente, instruções redundantes que precisam ser verificadas em busca de novos conteúdos, instruções que introduzem muitos tópicos novos desnecessariamente de uma vez.

Vários efeitos da carga cognitiva podem ser explicados em termos de carga estranha. Por exemplo, materiais que requerem que o aluno integre múltiplas fontes de informação causam um efeito de atenção dividida, uma vez que lidar com as fontes na memória de trabalho constitui uma carga estranha. Da mesma forma, materiais que repetem as mesmas informações em vários formatos causam um efeito de redundância, pois o processamento externo é necessário para comparar as alimentações de informações.

A Teoria da Carga Cognitiva (TCC), assim como outras teorias educacionais que surgiram décadas atrás, passou por revisão e atualização, que foi publicada em Sweller (2010). No recente trabalho de Duran et al. (2022), foram discutidas essas diferenças e sua ocorrência na área de Educação em Computação.

Uma das grandes mudanças nessa atualização foi referente a uma terceira fonte de carga cognitiva que era definida pela Teoria, denominada carga pertinente, que surgia do envolvimento com a atividade de aprendizagem. Junto com a carga intrínseca e estranha, a carga pertinente somava-se à carga cognitiva geral.

A carga pertinente seria uma carga cognitiva "boa": quanto mais, melhor. Se houver muita carga intrínseca e/ou estranha, não haverá "espaço"para a carga pertinente e nenhum aprendizado ocorrerá. Por outro lado, mesmo que a carga intrínseca e estranha sejam baixas o suficiente, a atividade de aprendizagem pode não envolver o aluno o suficiente para processar a informação.

Porém, na revisão publicada em Sweller (2010), o autor descarta a ideia de carga pertinente como uma terceira fonte de carga adicional. Sendo assim, a carga cognitiva é dividida em duas categorias: a carga intrínseca que

refere-se à complexidade inerente da informação que está sendo aprendida, e a carga estranha que diz respeito aos processamentos cognitivos desnecessários para a aprendizagem, como informações redundantes ou distrações. Qualquer carga cognitiva que não seja estranha é intrínseca, e o aprendizado resulta da forma como lidamos com ela, e elas passam a ser definidas estritamente em termos de interatividade do elemento, ou seja, a quantidade de elementos interconectados que devem ser processados conscientemente juntos.

A palavra "pertinente"não desapareceu na atualização da Teoria da Carga Cognitiva; entretanto, refere-se agora a recursos pertinentes ou processamento pertinente, que consiste em recursos de memória de trabalho usados para lidar com a interatividade do elemento associado à carga cognitiva intrínseca, desempenhando assim um papel reduzido na nova TCC, uma vez que não é uma carga que precisamos aumentar. Em outras palavras, o termo “pertinente” é usado para se referir a elementos ou informações relevantes para a tarefa de aprendizagem em questão.

Dentro dessa estrutura, a pertinência é uma propriedade dos elementos ou informações que são considerados relevantes e necessários para a tarefa/objetivo de aprendizagem, fornecendo informações essenciais ou facilitando a compreensão e a transferência do conhecimento.

Em resumo, na nova TCC, o termo “pertinente” refere-se à relevância e importância dos elementos ou informações em relação à tarefa de aprendizagem específica, mas não representa uma fonte separada de carga cognitiva.

Os recursos pertinentes são os recursos de memória de trabalho utilizados para lidar com a interatividade dos elementos associada à carga cognitiva intrínseca. Esses recursos desempenham um papel fundamental na facilitação da aprendizagem eficaz.São exemplos de recursos pertinentes no contexto da TCC:

- Organizadores gráficos: Diagramas, gráficos e mapas conceituais podem ser usados para representar visualmente informações complexas e facilitar a compreensão e a organização do conteúdo. Esses recursos ajudam os alunos a identificar relações entre conceitos e a estruturar o conhecimento de forma mais clara.

- Modelos e exemplos: Fornecer modelos e exemplos práticos pode ajudar os alunos a compreender conceitos abstratos e aplicar o conhecimento de forma mais efetiva. Esses recursos permitem que os alunos visualizem como aplicar o conhecimento em situações reais e forneçam uma estrutura de referência para a resolução de problemas.

- Dicas e pistas: Dicas e pistas fornecem informações adicionais que au-

Figura 2.1: Esquema da Teoria da Carga Cognitiva (atualizada). Fonte: A autora, 2023.

xiliam os alunos na resolução de problemas e na compreensão de conceitos. Esses recursos podem fornecer orientações específicas, destacar informações relevantes e ajudar os alunos a identificar estratégias de resolução de problemas.

- Feedback imediato: O feedback imediato durante a aprendizagem fornece aos alunos informações sobre seu desempenho e ajuda a corrigir erros e aprimorar o entendimento. Esse recurso permite que os alunos ajustem suas abordagens de aprendizagem e melhorem seu desempenho.

- Passos sequenciais: Dividir tarefas complexas em etapas sequenciais ajuda os alunos a lidar com a carga intrínseca, fornecendo uma estrutura clara e orientação passo a passo. Essa abordagem permite que os alunos processem as informações de maneira mais gerenciável, sem sobrecarregar sua memória de trabalho.

Os recursos pertinentes são os recursos de memória de trabalho utilizados para lidar com a interatividade dos elementos associada à carga cognitiva intrínseca, podendo minimizar a carga estranha, facilitando a compreensão e o processamento da informação relevante. Eles fornecem suporte adicional aos alunos, auxiliando-os na resolução de problemas e na compreensão dos

conceitos-chave, atuam como ferramentas de suporte que auxiliam os alunos a lidar de maneira mais eficaz com a carga cognitiva intrínseca, promovendo uma aprendizagem mais eficiente e significativa.

Por exemplo, ao utilizar organizadores gráficos, como mapas conceituais, os alunos podem visualizar a estrutura do conhecimento e identificar as relações entre os conceitos, evitando a carga estranha de ter que organizar mentalmente as informações de forma desordenada. Da mesma forma, fornecer dicas e pistas relevantes pode ajudar os alunos a filtrar informações desnecessárias e focar no que é essencial para a aprendizagem.

Na nova Teoria da Carga Cognitiva, a motivação é considerada um fator importante para a aprendizagem, porém, ela não é considerada uma carga adicional na mente do aluno. Em outras palavras, a motivação é vista como um aspecto externo à teoria em si.

Quando se fala em engajamento motivado, isso significa que antes de aplicar a teoria e considerar a carga cognitiva envolvida na aprendizagem de determinado conteúdo, é esperado que o aluno esteja motivado e engajado para aprender. A motivação é vista como uma condição prévia para que a teoria da carga cognitiva seja aplicada.

Isso significa que, embora a motivação seja relevante para a aprendizagem, ela não é considerada uma carga cognitiva adicional que precisa ser gerenciada ou controlada durante o processo de ensino e aprendizagem. Em vez disso, ela é vista como uma pré-condição para que a teoria da carga cognitiva seja aplicável e eficaz.

Essa perspectiva destaca a importância de criar um ambiente de aprendizagem motivador, estimulante e engajador, que desperte o interesse e a motivação dos alunos, antes de aplicar estratégias para gerenciar a carga cognitiva durante o processo de aprendizagem. Assim, a motivação é considerada um fator relevante, porém externo à teoria da carga cognitiva em si.

2.3.1 Recomendações Pedagógicas na Teoria da Carga Cognitiva

A pesquisa sobre carga cognitiva resultou em diversas recomendações pedagógicas que são frequentemente discutidas em termos de "efeitos". Esses efeitos descrevem princípios específicos que podem ser aplicados para otimizar o processo de aprendizagem com base na Teoria da Carga Cognitiva. Nesta seção, apresentaremos uma explicação mais detalhada sobre cada um desses efeitos.

Efeito do exemplo trabalhado

O efeito do exemplo trabalhado recomenda que os alunos estudem exemplos antes de tentar resolver problemas, pois essa abordagem facilita a compreensão dos conceitos. Os exemplos trabalhados, que apresentam uma solução pronta para os alunos estudarem, são considerados excelentes atividades de aprendizagem baseadas na Teoria da Carga Cognitiva. Ao analisar exemplos bem estruturados, os alunos podem compreender melhor a estratégia de resolução de problemas e, assim, aplicar esse conhecimento em situações semelhantes.

Efeito de atenção dividida

O efeito de atenção dividida recomenda que informações relacionadas sejam apresentadas de forma integrada, evitando sobrecarregar a memória de trabalho do aluno ao tentar conectar informações dispersas. Ao organizar o conteúdo de maneira coesa e lógica, os educadores podem facilitar a compreensão e a retenção de informações por parte dos alunos. A apresentação integrada de informações também ajuda os alunos a perceberem as relações entre os conceitos, promovendo uma aprendizagem mais significativa.

Efeito de redundância

O efeito de redundância enfatiza que a mesma informação não deve ser apresentada em vários formatos autocontidos. Quando informações essencialmente idênticas são apresentadas de maneira repetitiva, isso pode sobrecarregar a memória de trabalho do aluno, prejudicando a aprendizagem. Ao evitar redundâncias desnecessárias, os educadores podem direcionar a atenção dos alunos para os aspectos mais importantes e relevantes do conteúdo, melhorando a eficiência do processo de aprendizagem.

Efeito dos elementos isolados

O efeito dos elementos isolados recomenda que informações muito complexas sejam aprendidas isoladamente antes de serem integradas em um contexto mais amplo. Ao decompor conceitos complexos em partes mais simples e compreensíveis, os alunos podem desenvolver uma compreensão mais sólida e gradualmente construir um conhecimento mais abrangente. Essa abordagem incremental permite que os alunos se concentrem em aspectos específicos do conteúdo, evitando a sobrecarga cognitiva resultante de uma compreensão superficial e fragmentada.

Efeito de autoexplicação

O efeito de autoexplicação recomenda que os alunos gerem explicações para si mesmos sobre como raciocinam em relação a conceitos e procedimentos. Ao verbalizar ou escrever suas próprias explicações, os alunos são incentivados a refletir sobre o material de estudo, identificar lacunas em seu entendimento e estabelecer conexões significativas entre os conceitos. Essa abordagem promove a metacognição e o desenvolvimento de uma compreensão mais profunda e duradoura.

Efeitos compostos

Os efeitos compostos são aqueles que interagem com outros efeitos da carga cognitiva. Eles incluem o efeito de interatividade do elemento, que sugere que esses efeitos só se aplicam quando há informações suficientemente complexas envolvidas. Portanto, a interatividade entre os elementos do conteúdo pode afetar a carga cognitiva experimentada pelos alunos. Além disso, o efeito de reversão de experiência explica por que esses efeitos podem desaparecer ou até mesmo se reverter para alunos com conhecimento prévio do tópico. Por exemplo, alunos com um alto nível de conhecimento prévio podem se beneficiar mais da prática de resolução de problemas do que da simples exposição a exemplos trabalhados.

Neste trabalho, além dos efeitos mencionados, adaptamos essas recomendações pedagógicas para o contexto específico do ensino de programação em aplicações práticas. Nas seções seguintes, exploraremos cada um desses efeitos em maior profundidade, discutindo sua aplicação e relevância no ensino de programação.

2.3.2 Mensuração da Carga Cognitiva

Para projetar a instrução de forma eficaz, é essencial medir os componentes de carga cognitiva específicos para cada intervenção pedagógica. Em 2013, Leppink et al. (2013) desenvolveram um instrumento específico para medir diferentes tipos de carga cognitiva em uma pesquisa subjetiva composta por dez questões, aplicada em um curso de Estatística. Os resultados comprovaram que a ferramenta foi consistente em relação à medição das cargas cognitivas da TCC.

Posteriormente Leppink et al. (2014) estenderam seu trabalho de 2013, adaptando o instrumento de pesquisa para outro domínio, o de aprendizagem de línguas, e replicou suas análises. Esses novos dados reforçaram o forte suporte para a medição da carga intrínseca e estranha, mas encontraram menos suporte para a medição direta da carga pertinente.

Outro estudo de Morrison et al. (2014) adaptou o instrumento desenvolvido por Leppink para medir a carga cognitiva no contexto de uma nova disciplina: a introdução à ciência da computação em algumas aulas específicas. Essa adaptação envolveu apenas a troca de alguns termos no questionário para o contexto do ensino de programação. Ao final, o questionário foi composto pelas questões apresentadas a seguir.

Instruções: Todas as questões a seguir referem-se à palestra que acabou de terminar. Por favor, responda a cada uma das perguntas na seguinte escala, circulando o número apropriado (0 concordo totalmente e 10 discordo totalmente):

1. Os temas abordados na atividade eram muito complexos.
2. A atividade cobriu o código do programa que percebi como muito complexo.
3. A atividade abrangeu conceitos e de noções que percebi como muito complexos.
4. As instruções e/ou explicações durante a atividade não foram muito claras.
5. As instruções e/ou explicações foram, em termos de aprendizagem, muito ineficazes.
6. As instruções e/ou explicações estavam cheias de linguagem pouco clara.
7. A atividade realmente melhorou minha compreensão do(s) tópico(s) abordado(s).
8. A atividade realmente melhorou meu conhecimento e compreensão de computação/programação.
9. A atividade realmente melhorou minha compreensão do código do programa abordado.
10. A atividade realmente melhorou minha compreensão dos conceitos e definições.

O questionário é composto por 3 itens relacionados à carga intrínseca, 3 itens medindo carga estranha e 4 itens medindo carga pertinente. Os participantes respondem a cada item usando uma escala de 11 pontos de 0 a 10, onde 0 significa “concordo totalmente” e 10 significa “discordo totalmente”.

No entanto, tanto o instrumento original desenvolvido por Leppink et al. (2014) quanto o adaptado por Morrison et al. (2014), não incluem a atualização

da Teoria da Carga Cognitiva, pois ambos consideram a carga pertinente como a terceira componente da carga cognitiva.

Diante disso, propomos neste trabalho a adaptação do instrumento de medição de carga cognitiva para torná-lo acessível ao público de estudantes do Ensino Fundamental 2 em diante e para abranger a atualização da Teoria da Carga Cognitiva.

2.4 Aprendizagem por Andaimes

A Aprendizagem por Andaimes, ou *Scaffolding Learning*, como mais comumente conhecida proposta por Wood, Bruner e Ross Wood et al. (1976) trata-se de um suporte ou tutoria temporária, mas essencial, de um "mentor" no apoio aos alunos para realizar as tarefas com sucesso, reduzindo os graus de liberdade durante a solução da mesma, para que o aprendiz possa se concentrar no processo de aquisição de novos conhecimentos, aumentando a complexidade dos desafios para que possam resolvê-los de forma cada vez mais independente.

A Aprendizagem por Andaimes está diretamente relacionada com a teoria da Zona de Desenvolvimento Proximal de Vygotsky denominada no trabalho de Davis and Oliveira (1994), que é definida pela distância entre o nível de desenvolvimento real, ou seja, os problemas ou desafios que um aprendiz consegue resolver de forma independente, e o nível de desenvolvimento potencial, que se refere ao que ele consegue resolver sob a orientação de colegas ou um tutor. O Andaime, portanto, é o local onde esses aprendizes trabalham juntos para resolver um problema, com os mais avançados ajudando os menos avançados. A Figura 2.2 apresenta um esquema de Andaime na Aprendizagem.

Essa teoria é relevante para o contexto deste trabalho, pois o suporte fornecido aos alunos envolvidos em abordagens instrucionais centradas no problema, como é comum em aulas de ensino de programação, se beneficia ao incorporar andaimes. Os andaimes são definidos como um suporte interativo que aproveita o que os estudantes já sabem para ajudá-los a participar de forma significativa e adquirir habilidades em tarefas gradualmente mais complexas. Esse suporte é importante, pois aproveita o que os alunos já podem fazer para ajudá-los a realizar coisas que não seriam capazes de fazer de outra forma, como resolver o problema central, projetar um artefato para resolver o problema ou concluir um projeto.

No trabalho de Belland (2017) os Andaimes são caracterizados por:

- O Andaime deve fornecer suporte ao desempenho atual do estudante, mas também levar à capacidade de realizar a habilidade alvo de forma independente no futuro. Por exemplo, uma calculadora não se qualificaria

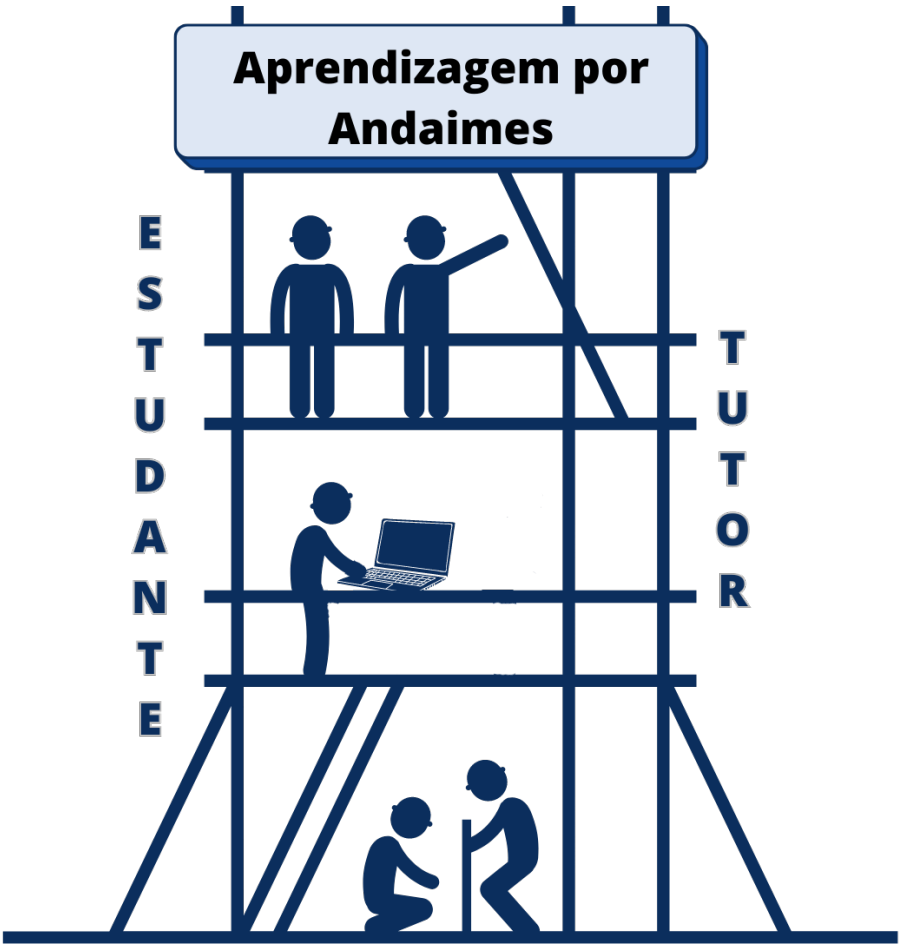

Figura 2.2: Esquema de Andaimes na Aprendizagem. Fonte: A autora, 2023.

como Andaime, pois não oferece suporte para que o estudante aprenda a realizar cálculos matemáticos, mas simplesmente realiza as operações por ele.

- O Andaime é usado enquanto os estudantes se envolvem com um problema autêntico. Ministrar palestras para os estudantes ou instruir sobre estratégias ou conteúdo antes do envolvimento deles com os problemas não se qualifica como Andaime.

- O andaime precisa (a) construir o conhecimento/habilidade a partir do que os alunos já sabem e (b) estar vinculado à avaliação contínua das habilidades dos alunos. Dessa forma, simplesmente dizer aos alunos o que fazer ou como fazer não se qualifica como Andaime, pois não leva em consideração as necessidades individuais dos alunos e não extrai e constrói a partir do que eles já sabem.

- O Andaime precisa simplificar alguns elementos da tarefa, mas também reter e destacar a complexidade de seus outros elementos. Isso permite que os estudantes reconheçam a complexidade da tarefa e sejam levados à autonomia em sua resolução no futuro.

Os andaimes podem ser classificados como:

- Andaime individual: é aquele em que o professor trabalha individualmente com o aluno para avaliar dinamicamente seu nível e fornecer a quantidade certa de suporte para que o aluno adquira as habilidades necessárias. Nesse tipo de andaime, o suporte é personalizado conforme as necessidades do aluno, permitindo que ele assuma a propriedade do aprendizado quando o andaime puder ser totalmente removido.

- Andaime de pares: o apoio é fornecido por colegas que possuem habilidades semelhantes ou superiores, aumentando a autoestima e engajando os alunos. Ele pode servir como uma segunda opção para obter apoio individualizado, mas requer uma estrutura que oriente o andaime sobre como e quando usá-la. No entanto, é importante destacar que este tipo de andaime deve ser utilizado em conjunto com outros, uma vez que os colegas com habilidades semelhantes não têm o conteúdo ou a experiência pedagógica para se envolver na avaliação dinâmica e personalização que é característica dos outros andaimes.

- Andaime de computador: o papel do tutor é cumprido por uma ferramenta tecnológica incluída no planejamento do assunto. As vantagens deste tipo de andaime é que ele pode ser usado individualmente e o desafio é torná-lo dinâmico e interativo. Ele auxilia os alunos na geração de soluções para problemas complexos e mal estruturados, o que significa que a ferramenta ajuda a estender as capacidades dos alunos, de forma que eles sejam capazes de desempenhar em um nível mais alto do que teriam de outra forma.

A avaliação dinâmica e a customização do Andaime estão intrinsecamente ligadas na definição original do Andaime Wood et al. (1976), porém, diferem em objetivos e métodos da avaliação tradicional, uma vez que não visam apenas determinar o nível atual de desempenho, mas também melhorá-lo, informando práticas instrucionais apropriadas, em vez de simplesmente classificar, e concentram-se nos níveis atuais e potenciais de desempenho dos alunos.

Quando a avaliação dinâmica é utilizada como base para o fornecimento de andaimes pelos professores, estes podem fazer perguntas e observar o desempenho dos alunos para determinar o nível de apoio necessário e, em seguida, fornecer o apoio adequado. A avaliação dinâmica também pode ser utilizada para ajustar o andaime que já está sendo fornecido, permitindo que os professores determinem até que ponto a habilidade do aluno está melhorando para levar ao sucesso sem andaimes ou com menos andaimes, possibilitando ajustes em tempo real.

No entanto, quando o resultado das avaliações dinâmicas é usado como base para a introdução, remoção ou ajuste de andaimes baseados em computador, os alunos geralmente precisam responder a perguntas de múltipla escolha. A veracidade das respostas ou a falta delas é então alimentada no rastreamento do modelo no sistema tutor inteligente, e o nível de suporte é aumentado ou reduzido. No entanto, o ajuste do andaime baseado em computador geralmente não é realizado com base na avaliação dinâmica, mas sim com base na auto-seleção ou em um cronograma fixo, o que resulta em dificuldades em programar ferramentas de computador para avaliar dinamicamente o desempenho dos alunos na resolução de problemas mal-estruturados que possuem inúmeros caminhos igualmente corretos que podem ser tomados.

Por exemplo, a avaliação dinâmica pode envolver o fornecimento de uma série de entradas, cada uma fornecendo diferentes níveis de suporte, permitindo que o professor determine o nível de habilidade atual do aluno com base no nível de apoio necessário para permitir um desempenho adequado.

A Aprendizagem por Andaimes está relacionada à Teoria da Carga Cognitiva, pois ambos buscam otimizar a aquisição de conhecimento, considerando a capacidade limitada da memória de trabalho. De acordo com Wood et al. (1976), ao fornecer suporte estruturado e direcionado, os andaimes ajudam a reduzir a carga cognitiva desnecessária, permitindo que os alunos concentrem-se na aprendizagem.

Por meio dos andaimes, os instrutores podem oferecer suporte temporário aos alunos, orientando-os na resolução de tarefas de forma gradual e adaptada ao seu nível de conhecimento. Isso inclui fornecer dicas, exemplos e orientações para ativar conhecimentos prévios relevantes, desenvolver estratégias de resolução de problemas e promover a transferência de habilidades.

A avaliação dinâmica desempenha um papel fundamental na Aprendizagem por Andaimes, permitindo que os instrutores avaliem continuamente o desempenho dos alunos e ajustem o suporte fornecido de acordo. Isso possibilita a personalização do andaime com base nas necessidades individuais dos alunos, garantindo que recebam o suporte adequado para progredir em sua aprendizagem.

Ao combinar a Teoria da Carga Cognitiva e a Aprendizagem por Andaimes, os instrutores podem projetar abordagens instrucionais mais eficazes, considerando as limitações da memória de trabalho e fornecendo suporte estruturado. Isso contribui para reduzir a carga cognitiva excessiva e promover a aprendizagem mais eficiente e profunda dos alunos.

O trabalho de Wood et al. (1976) apresenta formas de fornecer esse suporte, que podem ser estendidas ao ensino de programação e aplicadas juntamente aos efeitos da Teoria da Carga Cognitiva discutidos neste trabalho,

tais como trabalhar de forma colaborativa, engajar o estudante na resolução do problema, simplificar a tarefa por meio da solução parcialmente resolvida, demonstração de soluções seguida de um desafio equivalente ou a partir de uma resolução realizada anteriormente pelo estudante.

Portanto, o suporte da Aprendizagem por Andaimes pode ser considerado um recurso "pertinente" da Carga Cognitiva, pois contribui para a gestão eficiente da carga cognitiva dos alunos, permitindo-lhes direcionar seus recursos cognitivos para a aprendizagem significativa e promovendo uma melhor compreensão e retenção do conteúdo.

2.5 Ferramentas de Programação Introdutória

A introdução à programação é um desafio para muitos estudantes e professores. Com o aumento da demanda por habilidades em computação, muitas instituições educacionais estão buscando formas de ensinar programação de forma mais eficaz e acessível. Nesse sentido, as ferramentas de programação introdutória têm sido desenvolvidas para auxiliar nesse processo.

Essas ferramentas visam fornecer uma interface amigável para que os estudantes possam escrever e executar programas, sem a necessidade de se preocupar com a complexidade da linguagem de programação subjacente. Além disso, essas ferramentas podem fornecer feedback instantâneo e recursos de suporte, como sugestões de código, exemplos e tutoriais interativos, para ajudar os estudantes a superarem desafios e avançarem em seus projetos.

Essa seção apresenta as ferramentas que foram utilizadas neste estudo, incluindo suas características e vantagens.

2.5.1 Python

Python é uma linguagem de programação que combina os paradigmas procedural, funcional e orientado a objetos, e é amplamente reconhecida por sua facilidade de uso e aprendizado. Desenvolvida com o objetivo de ser uma linguagem acessível para leigos em computação, ela apresenta uma sintaxe simples e bibliotecas que apoiam o estudante no desenvolvimento dos códigos, além do reforço à indentação, ser interpretada e ter tipagem dinâmica de variáveis, dentre outras vantagens, segundo van Rossum and de Boer (1991).

O uso de Python como linguagem de programação introdutória tem sido amplamente estudado e defendido por educadores, pois permite que os alunos se concentrem nos conceitos fundamentais de programação sem serem distraídos por sintaxe complexa ou por preocupações com gerenciamento de memória,

Devido a essas características, Python tem se tornado cada vez mais popular e é amplamente utilizado em diversas áreas da computação por programadores profissionais. Além disso, seu uso como primeira linguagem em cursos introdutórios de programação tem se mostrado uma alternativa viável em comparação às linguagens tradicionais, como C, C++ e Java, conforme Júnior and Bogea (2020). Nesse contexto, a adoção de Python como primeira linguagem de programação pode trazer benefícios significativos para o processo de ensino-aprendizagem, fornecendo aos estudantes uma base sólida em programação e preparando-os para enfrentar desafios mais avançados em suas carreiras.

2.5.2 A Ferramenta Hedy

Hedy é uma ferramenta desenvolvida por pesquisadores da Universidade de Leiden, na Holanda, com o propósito de ensinar programação para iniciantes, segundo Hermans (2020). A ferramenta baseia seu design instrucional em estratégias comuns no ensino da língua materna, uma vez que a programação é uma forma de linguagem que tem o mesmo objetivo que qualquer outra: a comunicação, além de conter elementos correspondentes a outras linguagens. Sendo assim, a ferramenta se apoia na Teoria da Carga Cognitiva em seu desenvolvimento, buscando reduzir a carga cognitiva durante o aprendizado de programação em uma linguagem textual. Introduz os conceitos de programação e a sintaxe de forma lenta e gradual, até que os aprendizes estejam programando em Python, Hermans (2020).

A ferramenta é direcionada a crianças com cerca de 11 anos de idade, pois espera-se que nesta faixa etária elas exerçam uma quantidade limitada de carga cognitiva na leitura da linguagem natural e tenham maturidade suficiente para raciocinar de forma lógica e sistemática. Hedy possui 13 níveis que apresentam sequencialmente os conceitos de entrada e saída de dados, variáveis, tipos de dados, estrutura condicional, laço de repetição, operadores aritméticos e procedimentos com sintaxe simples e gradual. Nos demais níveis, a sintaxe é incorporada de forma mais próxima do Python, aplicando os mesmos conceitos de programação dos níveis anteriores. A Figura 2.3 mostra a interface da ferramenta e a Figura 2.4 apresenta o diagrama de funcionamento.

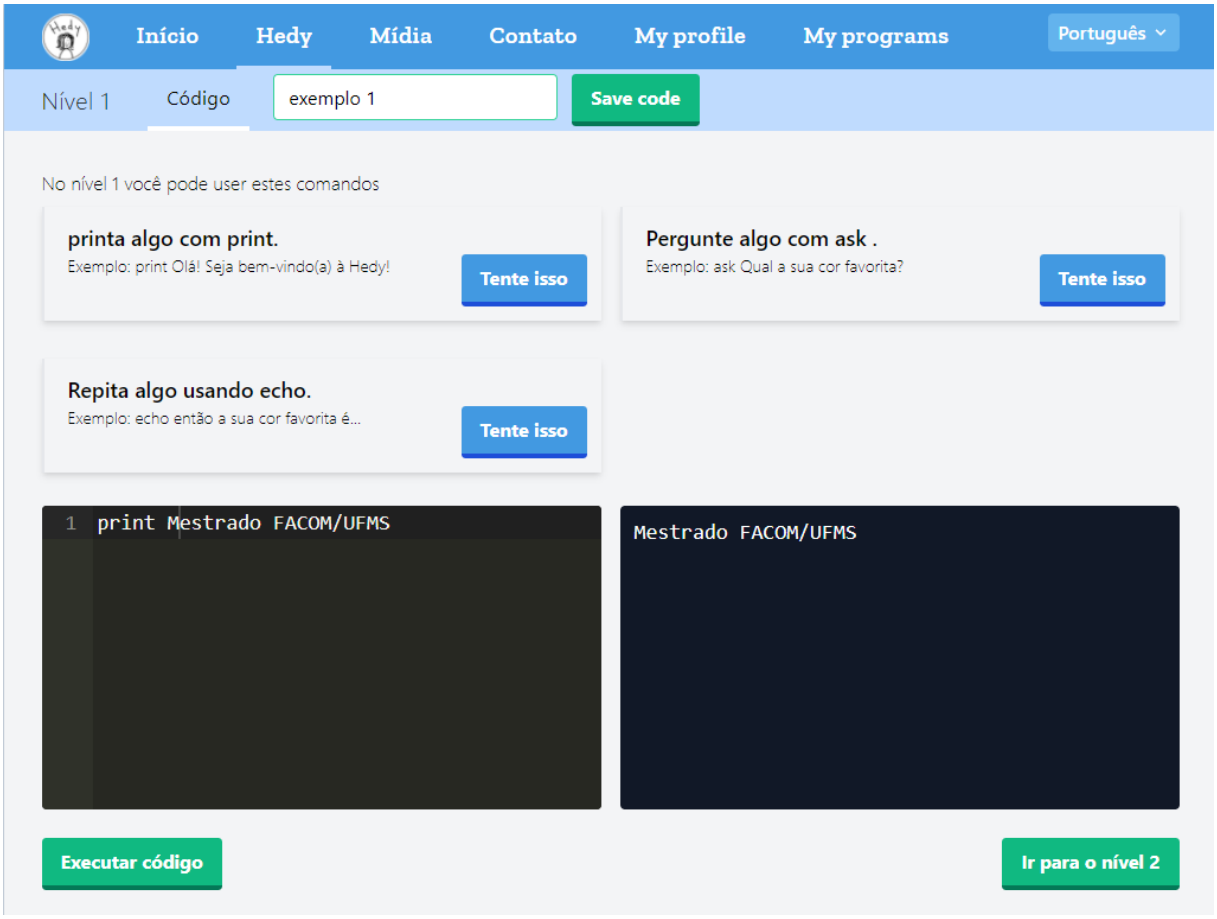

Figura 2.3: Interface de Hedy: primeira atividade do Nível 1

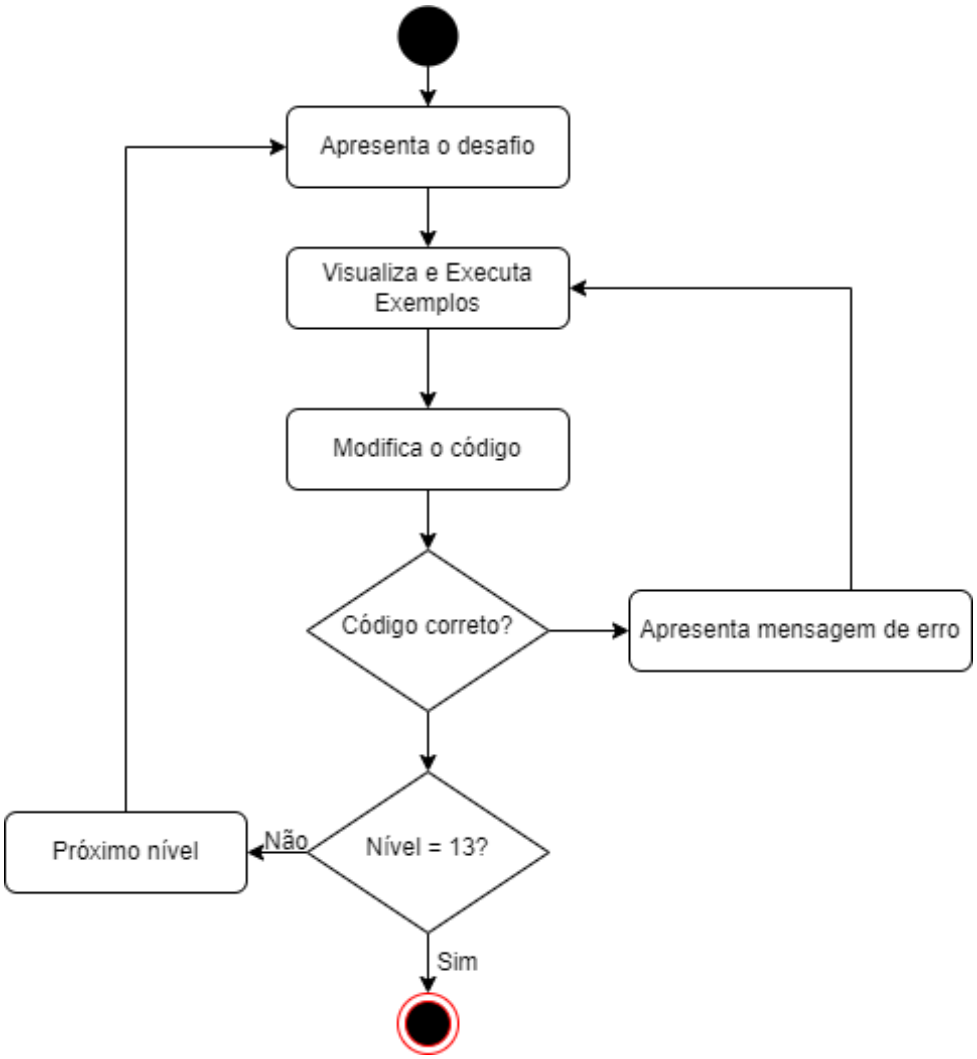

Figura 2.4: Diagrama representando funcionamento do Hedy. Fonte: A autora, 2023.

CAPÍTULO

3

Design do Estudo

Esta seção descreve como o estudo foi conduzido para atingir os objetivos estabelecidos para o trabalho. O estudo foi realizado em três etapas, conforme descrito abaixo:

- Adaptação de um instrumento para medir a Carga Cognitiva de estudantes em cursos de programação introdutória, baseado em Leppink et al. (2014).
- Criação de uma sequência didática para um curso de programação introdutória, fundamentada nas recomendações pedagógicas da Teoria da Carga Cognitiva e da Aprendizagem por Andaimes.
- Avaliação do curso de programação introdutória, que foi aplicado em duas turmas de ensino fundamental do 6º ao 9º ano. O curso foi ministrado utilizando a ferramenta Hedy e Google Colab.

3.1 Materiais e Métodos

Para adaptar o instrumento para medição da carga cognitiva de Leppink Leppink et al. (2013), foram levadas em consideração as adaptações realizadas por Morrison et al. (2014). Em seguida, foi realizado um estudo das perguntas e elas foram ajustadas para atender as atualizações da Teoria da Carga Cognitiva, bem como para se adequar ao contexto da aplicação, que foi um curso de programação introdutória destinado a jovens a partir do ensino fundamental 2. A escala também foi ajustada para facilitar o entendimento desse público jovem.

Para o design instrucional do curso de programação introdutória, foi realizada uma análise das recomendações pedagógicas da Teoria da Carga Cognitiva, agregando com os andaimes da Aprendizagem por Andaimes para definir como apresentar e trabalhar os conteúdos que comumente fazem parte dos cursos de programação introdutória.

O curso teve duração total de 30 horas, com cada aula tendo a duração de 90 minutos para cada turma. Para a aplicação da trilha do curso de programação introdutória nas duas turmas do ensino fundamental 2, utilizou-se a ferramenta Hedy para as turmas do 6º e 7º ano e o Google Colab para as turmas do 8º e 9º ano. As aulas aconteceram no formato híbrido (presencial e online), sendo que as aulas online contaram com o apoio de videoaulas gravadas para o curso em formato autoinstrucional.

A escolha do Hedy na primeira turma, composta pelos alunos do 6º e 7º ano, deu-se pelo fato de a ferramenta incorporar a Teoria da Carga Cognitiva em seu design instrucional e ser destinada especificamente para programação introdutória. A IDE do Google Colab foi escolhida devido ao local de aplicação da pesquisa, uma escola da rede pública, e aos computadores de seu laboratório tecnológico não suportarem a instalação de outras IDEs de programação em Python.

A linguagem de programação Python foi escolhida para a segunda turma, para que pudesse manter a linguagem mais próxima possível da segunda turma, uma vez que os desafios evoluem nos níveis de Hedy até o uso de uma linguagem muito próxima ao Pyhton. Além disso, um dos objetivos transversais deste trabalho é comparar o Hedy com outras IDEs, para saber se a incorporação das recomendações da Teoria da Carga Cognitiva, que também implica no uso da Aprendizagem por Andaimes, tem ganhos de desempenho dos alunos. Dessa forma, embora a segunda turma, composta pelos alunos do 8º e 9º ano, tenha utilizado a IDE do Google Colab, que não incorpora essas teorias em seu design, elas foram utilizadas para o projeto de todo o curso, mantendo a linguagem Python.

O formulário eletrônico para mensurar a carga cognitiva foi aplicado após cada aula do curso de programação introdutória, tanto para a turma que utilizou o Hedy quanto para a que usou o Google Colab.

3.2 Participantes

A escola onde o curso de programação introdutória foi aplicado é de tempo integral da Rede Estadual de Ensino de Mato Grosso do Sul. O curso foi oferecido em horário extracurricular, após o término das aulas no período da tarde.

A aplicação do curso contou com a participação de 10 alunos do 6º e 7º ano, que utilizaram a ferramenta Hedy, e 6 alunos do 8º e 9º ano, que utilizaram a ferramenta Google Colab. A participação no curso foi opcional e aberta somente aos alunos que demonstraram interesse, uma vez que a motivação é uma premissa importante para a aplicação da Teoria da Carga Cognitiva.

Todas as aulas seguiram a seguinte sequência: a) revisão do conceito apresentado na aula anterior e correção dos desafios propostos; b) apresentação de um novo conceito ou incremento da sintaxe; c) proposta de desafios relacionados ao novo conceito/sintaxe apresentado; d) aplicação de um formulário para mensuração da carga cognitiva referente à aula do dia.

Durante as aulas utilizou-se o instrumento de diário de bordo para registrar as observações da aula, principalmente quando alguns alunos apresentavam maior facilidade e dificuldade. Isso permitiu uma análise com a resposta que eles marcaram no formulário de mensuração da TCC, ou seja, serviu como balizador para saber o quanto essa dificuldade ou facilidade refletia no instrumento de mensuração, para conseguirmos validar o mesmo neste estudo em nível de confiabilidade.

3.3 Análise dos dados

A análise dos dados coletados foi baseada em técnicas estatísticas. Como o estudo contou com poucos participantes, optou-se por analisar individualmente os componentes da carga intrínseca e estranha dos questionários aplicados em cada aula, utilizando medidas como média, variância e desvio padrão.

Essa abordagem permitiu uma compreensão mais detalhada do comportamento da carga cognitiva ao longo do curso. Além disso, uma análise geral foi realizada considerando a média dos fatores para obter uma visão panorâmica do desempenho dos alunos durante o curso.

CAPÍTULO

4

Resultados

Nesta seção apresentaremos os resultados obtidos com a aplicação da metodologia descrita na seção anterior. Iniciaremos descrevendo os resultados referentes à validação do instrumento de mensuração da carga cognitiva utilizado. Em seguida, apresentaremos os resultados da análise da carga cognitiva dos participantes ao longo do curso, incluindo a comparação entre os diferentes níveis de desafios propostos. Por fim, discutiremos os principais achados do estudo e suas implicações para o ensino de programação.

4.1 Adaptação do instrumento para mensuração da carga cognitiva

O primeiro objetivo deste estudo foi adaptar o instrumento de mensuração da carga cognitiva, considerando a atualização da Teoria da Carga Cognitiva em 2010. Nessa adaptação, as questões referentes à carga pertinente foram retiradas, uma vez que, como visto nas seções anteriores, na atual TCC, ela se refere apenas aos recursos utilizados e não pode ser mensurada diretamente.

As demais questões foram adaptadas para atender: a) ao contexto do ensino de programação, de forma similar à adaptação realizada por Morrison et al. (2014); b) à apresentação das frases de forma positiva, por exemplo, o primeiro item foi alterado de “Os temas abordados na atividade eram muito complexos” e “Os conceitos abordados na atividade eram muito fáceis”; e c) à compreensão do texto e da escala pelo público de estudantes mais jovens, como no caso deste estudo que envolveu crianças de 11 a 14 anos.

O instrumento adaptado abrangeu seis perguntas, conforme apresentado

a seguir: os três primeiros itens abordam a carga intrínseca e os outros três itens abordam a carga estranha.

1. Os conceitos abordados na atividade eram muito fáceis.
2. A atividade abrangeu um código do programa que achei muito fácil.
3. Exigiu-se pouco esforço para conectar os vários conceitos de programação aprendidos nesta aula e/ou aulas anteriores para o desenvolvimento da atividade.
4. As instruções e/ou explicações para realizar a tarefa eram fáceis de entender
5. Foi de fácil compreensão o código do programa abordado na tarefa.
6. Considerei a linguagem (textos, imagens, vídeos) utilizada na explicação fácil de compreender.

Os participantes responderam a cada item usando uma escala de 10 pontos, de 1 a 10, em que 1 representa "discordo totalmente" e 10 representa "concordo totalmente". Em termos de aprendizagem, quanto maior a resposta dos itens 1 a 3 (carga intrínseca) e dos itens 4 a 6 (carga estranha), menor carga cognitiva estaria sendo gerada durante a interação dos elementos, e o aprendizado ocorrendo de forma mais eficaz, de acordo com a Teoria da Carga Cognitiva. É importante lembrar que, no instrumento original de Leppink et al. (2013) e na adaptação de Morrison et al. (2014), os itens 4 a 6 eram apresentados de forma negativa, e esperava-se que estivessem relacionados inversamente. Ou seja, quanto maior a carga intrínseca, menor a carga estranha.

Este questionário foi aplicado aos alunos das duas turmas do curso, sempre ao término da aula que apresentava a introdução de um novo conceito e/ou sintaxe.

4.2 Design do Curso de Programação Introdutória

Essa seção apresenta o Design do Curso de Programação Introdutória apresenta a metodologia utilizada para o desenvolvimento do curso de programação introdutória, incluindo a análise das recomendações pedagógicas da Teoria da Carga Cognitiva e a utilização dos andaimes da Aprendizagem por Andaimes para definir como apresentar e trabalhar os conteúdos que comumente fazem parte dos cursos de programação introdutória. Além disso, a seção descreve a duração do curso, a estrutura das aulas e as ferramentas utilizadas para a aplicação da trilha do curso de programação introdutória nas duas turmas do ensino fundamental 2.

4.2.1 Conhecimentos do Curso

O curso de Programação Introdutória abordou os seguintes conhecimentos:

1. Entrada e saída de dados: os alunos aprenderam como interagir com o usuário por meio da entrada e saída de dados. Isso inclui solicitar informações ao usuário, como texto ou números, e exibir resultados na tela.

2. Variáveis e atribuições: Os alunos foram introduzidos ao conceito de variáveis, que são espaços na memória utilizados para armazenar dados. Eles aprenderam a declarar variáveis, atribuir valores a elas e utilizar essas variáveis em cálculos e manipulações.

3. Estruturação de dados com listas: as listas são estruturas de dados que permitem armazenar uma coleção de elementos. Os alunos aprenderam a criar, acessar e modificar elementos em listas, além de entender conceitos como índices e iteração.

4. Números randômicos e números decimais: Foi ensinado como gerar números aleatórios para criar uma dinâmica ou para realizar simulações. Além disso, os alunos foram introduzidos aos números decimais e como trabalhar com eles em cálculos.

5. Operadores lógicos, matemáticos, aritméticos e de relação: os operadores são símbolos utilizados para realizar operações em programação. Os alunos aprenderam sobre operadores lógicos (como AND, OR e NOT), operadores matemáticos (como adição, subtração, multiplicação e divisão), operadores aritméticos (como resto da divisão inteira) e operadores de relação (como igualdade, maior que, menor que).

6. Estrutura condicional: a estrutura condicional if-else permite que os alunos criem condições em seu programa para tomar decisões. Eles aprenderam a avaliar uma condição e executar diferentes blocos de código com base nessa avaliação.

7. Estruturas de repetição: as estruturas de repetição permitem que os alunos executem um bloco de código repetidamente enquanto uma condição for verdadeira. Eles foram introduzidos às estruturas de repetição que são usadas para automatizar tarefas repetitivas ou percorrer elementos em uma lista.

4.2.2 Recomendações da Teoria da Carga Cognitiva para o ensino de programação introdutória

As recomendações da Teoria da Carga Cognitiva apresentadas na sequência foram adaptadas para o contexto do ensino de programação, com o objetivo de oferecer suporte à Aprendizagem por Andaimes necessária para a resolução de problemas algorítmicos durante o processo de aprendizagem.

Intercalar exemplos de problemas algorítmicos resolvidos com exercícios

Durante o ensino de programação, é recomendado intercalar exemplos resolvidos com exercícios a serem resolvidos pelos alunos. Por exemplo, após explicar os conceitos básicos de estruturas de controle de fluxo em Python, como loops e condicionais, os alunos podem ser apresentados a um exemplo resolvido de um programa que utiliza essas estruturas para resolver um problema específico. Em seguida, os alunos devem tentar resolver exercícios semelhantes, aplicando os conceitos aprendidos. Esse processo gradual permite que os alunos adquiram conhecimentos e habilidades progressivamente, construindo seu entendimento por meio da prática.

Muitas IDEs de programação já incorporam exemplos resolvidos, especialmente aquelas que utilizam sub-linguagens para facilitar o processo de aprendizagem, como Hedy e aquelas que usam Portugol. Esses exemplos resolvidos podem ser vistos como andaimes, no qual, aos poucos, os exemplos resolvidos tornam-se mais incompletos, permitindo que os estudantes ganhem autonomia à medida que compreendem o conceito apresentado.

Apresentar conceitos de forma isolada antes de integrá-los

Ao ensinar programação, é recomendado apresentar conceitos de forma isolada antes de integrá-los em um contexto mais amplo. Por exemplo, ao ensinar funções em Python, é melhor começar explicando a estrutura básica de uma função, seus parâmetros e como retornar valores. Em seguida, os alunos podem praticar escrevendo e chamando funções simples antes de avançar para exemplos mais complexos que envolvam o uso de funções em diferentes partes de um programa. Essa abordagem gradual ajuda os alunos a compreenderem os conceitos antes de lidar com a complexidade adicional da integração entre eles.

Introduzir conceitos de forma simultânea, como por exemplo, variáveis juntamente com entrada e saída de dados pode forçar uma alta interatividade dos elementos, principalmente quando a linguagem requer que a sintaxe seja introduzida ao mesmo tempo.

Para iniciantes, o uso de uma linguagem com poucas regras de sintaxe

pode ser uma boa alternativa. Quando a mesma permite a resolução de problemas com menos sintaxe, deve-se optar por ela, como é o caso das estruturas condicionais, para evitar a sobrecarga de memória de trabalho, aumentando assim a carga cognitiva durante o aprendizado. Dessa forma, ajuda-se a minimizar a interatividade dos elementos apresentados aos estudantes.

Verificar o conhecimento prévio dos alunos

Antes de iniciar um curso de programação, é importante verificar o conhecimento prévio dos alunos para adequar a abordagem e o nível de suporte necessário com o Andaime

Isso pode ser feito por meio de uma avaliação inicial ou de uma atividade diagnóstica que permita identificar o conhecimento prévio dos alunos sobre programação ou conceitos relacionados. Com base nessa avaliação, os educadores podem ajustar o ritmo e a profundidade do ensino, adaptando os exemplos resolvidos e os exercícios para atender às necessidades individuais dos alunos.

Incentivar a autoexplicação

Durante o processo de aprendizagem da programação, é recomendado que os alunos pratiquem a autoexplicação. Isso envolve pedir aos alunos que expliquem para si mesmos os conceitos e procedimentos necessários para resolver um problema algorítmico. Por exemplo, após a resolução de um exemplo incompleto, os alunos podem ser solicitados a explicar em voz alta ou por escrito como eles completaram o código e qual a lógica por trás de suas escolhas. Essa prática de autoexplicação promove a reflexão, a metacognição e o desenvolvimento de uma compreensão mais profunda.

Utilizar linguagens com poucas regras de sintaxe

No contexto de programação introdutória, é recomendado utilizar linguagens de programação com poucas regras de sintaxe (ou ainda nenhuma) para evitar sobrecarga cognitiva desnecessária. Python é um bom exemplo de linguagem com uma sintaxe clara e concisa, o que a torna uma opção popular para ensinar programação introdutória. Ao utilizar uma linguagem com uma sintaxe mais simples, os alunos podem se concentrar nos conceitos e na lógica da programação, em vez de se preocuparem com a complexidade da sintaxe.

Essas recomendações foram utilizadas no design instrucional do curso proposto, contribuindo para a melhoria do processo de aprendizagem em programação.

4.2.3 Avaliação Diagnóstica e Acompanhamento do Curso

Na primeira aula presencial, foi aplicada uma avaliação diagnóstica com vinte questões de conceitos de programação retiradas da prova da OBR (Olimpíada Brasileira de Robótica). O objetivo desse teste foi verificar o conhecimento prévio dos alunos e analisar os tipos de exercícios e nível de suporte inicial necessários. Como resultado, a Turma 1 (6º e 7º ano) obteve uma média de 20% de acertos (4 questões), e a Turma 2 (8º e 9º ano) obteve uma média de 35% de acertos (7 questões). Isso demonstra um nível baixo de conhecimento prévio em programação, considerando que as questões da OBR abordam conteúdos regulares estudados na escola com um nível de complexidade adaptado para iniciantes em robótica e programação.

A Tabela 4.1 apresenta uma tabela com os conteúdos abordados em cada aula para cada turma. As aulas que abordavam novos conceitos eram presenciais e estão indicadas na tabela em fonte normal. No início de cada aula presencial, os conceitos da aula anterior eram retomados, e a correção dos desafios realizados pelos alunos na aula online anterior era feita previamente para que a quantidade de suporte adequasse às necessidades da turma. As soluções dos desafios também eram utilizadas como exemplos para os desafios propostos na aula presencial.

Durante as aulas online, foram abordados desafios de programação utilizando os conceitos apresentados na aula anterior, com exceção da aula 11 da turma 1 e da aula 2 da turma 2, que apresentaram novos conceitos. Para auxiliar na resolução desses desafios, os alunos tinham acesso a um roteiro de estudo com explicações e videoaulas gravadas para cada turma, retomando os conceitos necessários.

Durante a apresentação de novos conceitos ou regras de sintaxe, foram exibidos códigos incompletos para que a turma pudesse auxiliar no raciocínio e completar o código faltante, com o suporte do tutor. Esse suporte foi reduzido ou retirado quando a turma avançava para a próxima etapa da aula, que consistia na realização de um ou mais desafios relacionados ao assunto abordado.

Ao final de cada aula, foi aplicado um formulário com uma ferramenta de mensuração da carga cognitiva referente à aula do dia, e uma explicação sobre as perguntas foi fornecida na introdução do formulário. Nas primeiras aulas presenciais do curso, o docente orientou os alunos sobre como responder ao formulário, lembrando-os de que o objetivo não era avaliar o professor, para que pudessem responder com sinceridade. Cada questão foi lida em voz alta para a turma, explicando como marcar a resposta corretamente. Nas aulas online, o formulário foi aplicado apenas em algumas aulas da turma 2. Embora tenha havido a intenção de aplicá-lo também na turma 1, houve baixa

Tabela 4.1: Conteúdos das aulas ministradas (em itálico aqueles executados em aula online)

	Turma 1 - Hedy - 6 e 7 ano	Turma 2 - Google Colab - 8 e 9 ano
Aula 1	Avaliação Diagnóstica	Avaliação Diagnóstica
Aula 2	Conceitos iniciais - Entrada e saída de Dados	*Conceitos iniciais - Entrada e saída de Dados*
Aula 3	*Desafios - Nível 1 (print, ask e echo)*	Variáveis (=, input, print)
Aula 4	Variáveis - Nível 2 (is)	*Exercícios variáveis*
Aula 5	*Exercícios Nível 2*	Listas (append, insert, remove, del)
Aula 6	Listas - Nível 3 (at random, add, remove)	*Exercícios Listas*
Aula 7	*Exercícios Nível 3*	Operadores no Python
Aula 8	Aspas Simples - Nível 4	*Exercícios operadores*
Aula 9	*Condicional - Nível 5 (if..else)*	Condicionais Python (if, if..else)
Aula 10	Condicional com listas - Nível 5 (if...else e in)	*Exercícios condicionais*
Aula 11	*Operadores Artiméticos - Nível 6 (+ - *)*	Conversão de Dados
Aula 12	Laço de repetição - Nível 7 (repeat sem identação)	*Exercícios com conversão de dados*
Aula 13	Laço de repetição - Nível 8 (repeat com identação)	Laço de Repetição While
Aula 14	Laço de repetição com condicional - Nível 8 (repeat e if)	*Exercícios laços de Repetiçao*
Aula 15	*Todos os comandos - Nível 9 (exercícios)*	Laço de Repetição for
Aula 16	Todos os comandos - Nível 9 (exercícios)	*Exercícios laços de Repetiçao*
Aula 17	Laço de repetição com condicional - Nível 10 (for e if)	Laço de Repetição com condicional
Aula 18	*Exercícios nível 10*	*Exercícios laços de Repetiçao*
Aula 19	Desafios finais - Programação em pares	Desafios Finais - Programação em pares
Aula 20	*Correção e Finalização*	*Correção e Finalização*

adesão dos alunos.

4.3 Discussão dos Resultados

Os dados obtidos durante as aulas no formulário com a ferramenta de mensuração da carga cognitiva foram analisados a partir da média e dispersão dos dados com a variância e desvio padrão, que nos diz o quanto eles podem variar dentro de seu conjunto.

A variância nos informa o quão distantes os valores estão da média, ou seja, quanto maior for a variância, mais distantes da média estarão os valores, e quanto menor for a variância, mais próximos os valores estarão da média. Observamos também o desvio padrão, que é o resultado positivo da raiz quadrada da variância, que nos indica a dispersão dos dados dentro de uma amostra com relação à média. Assim, quando se calcula o desvio padrão juntamente com a média de diferentes grupos, obtém-se mais informações para avaliar e diferenciar seus comportamentos. Quanto menor o desvio padrão, mais homogênea é a amostra.

A tabela 4.2 é o resultado do formulário para mensuração da carga cognitiva com Hedy em cada aula presencial em que foi aplicado para cada um dos itens e a tabela 4.3 apresenta a média dos dados destes itens para cada aula.

Podemos observar que a Aula 2 foi a que obteve menor valor médio (μ) na carga intrínseca e carga geral. Esses valores foram influenciados pelos itens 2 e 5, que se referem à facilidade de compreensão do código do programa, e pelo item 3, que diz respeito à conexão dos novos conceitos iniciais de programação

Tabela 4.2: Resultado da Mensuração da Carga Cognitiva nas aulas presenciais da turma 1 com o uso do Hedy para cada item da carga cognitiva

	Carga Intrínseca									Carga Estranha								
	Item 1			Item 2			Item 3			Item 4			Item 5			Item 6		
	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ
Aula 2	6,25	8,19	2,86	6,13	5,86	2,42	4,00	3,25	1,80	7,00	12,50	3,54	4,13	4,11	2,03	6,75	8,69	2,95
Aula 4	6,50	9,05	3,01	7,30	3,61	1,90	6,30	9,21	3,03	7,50	6,65	2,58	6,80	7,76	2,79	7,30	7,81	2,79
Aula 6	7,75	4,44	2,11	8,75	2,94	1,71	7,50	3,50	1,87	7,88	3,86	1,96	7,63	2,48	1,58	7,50	2,25	1,50
Aula 8	5,83	7,14	2,67	6,17	6,14	2,48	6,50	10,25	3,20	6,50	5,58	2,36	5,83	4,14	2,03	4,83	4,81	2,19
Aula 10	7,67	4,56	2,13	7,50	5,58	2,36	4,00	7,67	2,77	8,67	1,89	1,37	8,50	1,92	1,38	4,83	7,81	2,79
Aula 12	6,43	7,67	2,77	7,71	4,20	2,05	5,00	9,43	3,07	8,71	2,78	1,67	6,57	8,53	2,92	6,14	7,84	2,80
Aula 13	6,00	4,67	2,16	6,67	3,56	1,89	6,00	2,67	1,63	5,67	2,89	1,70	5,00	0,67	0,82	5,00	2,00	1,41
Aula 14	7,36	3,87	1,97	7,45	3,34	1,83	7,55	4,43	2,1	8,45	1,70	1,3	7,73	2,38	1,54	8,36	1,14	1,07
Aula 16	7,33	1,56	1,25	7,44	1,14	1,07	6,89	1,65	1,29	8,00	1,33	1,15	7,67	1,56	1,25	7,89	0,54	0,74
Aula 17	7,44	2,47	1,57	7,22	2,62	1,62	6,67	2,67	1,63	8,11	1,88	1,37	7,78	2,84	1,69	8,25	0,44	0,66
Aula 19	7,00	5,00	2,24	8,00	2,00	1,41	6,50	2,75	1,66	6,75	5,19	2,28	6,00	4,50	2,12	5,75	9,69	3,11
μ	**6,87**	**5,33**	**2,25**	**7,30**	**3,73**	**1,89**	**6,08**	**5,22**	**2,19**	**7,57**	**4,20**	**1,93**	**6,69**	**3,72**	**1,83**	**6,60**	**4,82**	**2,00**

apresentados.

Essa dificuldade logo na primeira aula pode ser explicada pelos graus de conhecimento prévio em programação que os alunos da turma apresentaram na avaliação diagnóstica realizada. A análise das medidas de dispersão, como a variância (σ^2) e o desvio padrão (σ) nas tabelas 4.2 e 4.3, mostra que a turma é heterogênea ao iniciar o curso. Isso significa que alguns alunos apresentaram baixa carga cognitiva, indicando maior dificuldade, enquanto outros tiveram alta carga cognitiva, indicando facilidade frente aos conceitos apresentados.

Essa disparidade entre os alunos pode ser vista na aula 2, onde alguns tiveram mais dificuldade para compreender os conceitos de entrada e saída de dados e as primeiras linhas de programação no nível 1 de Hedy, enquanto outros apresentaram facilidade. Esse contraste pode explicar a baixa média de carga intrínseca e geral nessa aula.

Essa dispersão diminui ao longo do curso como podemos observar na tabela 4.3 em que os valores de desvio padrão geral nas aulas diminui no decorrer do curso, no entanto, ainda existe, mantendo-se ainda distante do ideal, zero. Uma das hipóteses para esse comportamento é que ao decorrer do curso com a inserção gradual de sintaxe e conceitos, essa dispersão diminui, fazendo que com os alunos fiquem mais nivelados entre si em termos de conhecimento.

Na aula 8, o item 1 referente a carga intrínseca e item 6 referente a carga estranha, apresentaram os menores valores, estes dizem respeito respectivamente aos conceitos abordados na aula e à facilidade na linguagem (textos, imagens, vídeos) utilizadas na explicação. Nessa aula, foi abordado a incrementação de sintaxe no nível 8 de Hedy, com o uso de aspas simples no print e ask. O resultado foi de contraponto a nossa hipótese inicial, de que na inserção gradual da sintaxe haveria uma redução da carga cognitiva. A aula 13 apresentou comportamento similar ao incrementar a sintaxe para a identação dentro do laço de repetição *repeat*.

A aula 10 que aconteceu com o nível 5 de Hedy chama também a atenção

Tabela 4.3: Resultado da Mensuração da Carga Cognitiva nas aulas presenciais da turma 1 com o uso do Hedy agrupado por carga cognitiva, carga estranha e geral

	Carga Intrínseca			Carga Estranha			Geral		
	μ			μ			μ		
	μ	σ²	σ	μ	σ²	σ	μ	σ²	σ
Aula 2	5,46	5,77	2,36	5,96	8,43	2,84	5,71	7,10	2,60
Aula 4	6,70	7,29	2,65	7,20	7,41	2,72	6,95	7,35	2,68
Aula 6	8,00	3,63	1,90	7,67	2,86	1,68	7,83	3,24	1,79
Aula 8	6,17	7,84	2,78	5,72	4,84	2,20	5,94	6,34	2,49
Aula 10	6,39	5,94	2,42	7,33	3,87	1,85	6,86	4,90	2,14
Aula 12	6,38	7,10	2,63	7,14	6,38	2,46	6,76	6,74	2,55
Aula 13	6,22	3,63	1,89	5,22	1,85	1,31	5,72	2,74	1,60
Aula 14	7,45	3,88	1,97	8,18	1,74	1,30	7,82	2,81	1,64
Aula 16	7,22	1,45	1,20	7,85	1,14	1,05	7,54	1,30	1,13
Aula 17	7,11	2,59	1,61	8,05	1,72	1,24	7,58	2,15	1,42
Aula 19	7,17	3,25	1,77	6,17	6,46	2,50	6,67	4,85	2,14
μ	6,75	4,76	2,11	6,95	4,25	1,92	6,85	4,50	2,02

Tabela 4.4: Resultado da Mensuração da Carga Cognitiva nas aulas presenciais da turma 2 com o uso do Python no Google Colab para cada item da carga cognitiva

	Carga Intrínseca									Carga Estranha								
	Item 1			Item 2			Item 3			Item 4			Item 5			Item 6		
	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ
Aula 3	6,50	7,25	2,69	7,25	5,19	2,28	6,00	9,50	3,08	7,25	3,69	1,92	7,25	3,69	1,92	7,75	3,69	1,92
Aula 5	7,67	0,22	0,47	8	0,67	0,82	9	0,67	0,82	9,67	0,22	0,47	8,33	0,89	0,94	9,33	0,22	0,47
Aula 7	8,4	1,04	1,02	8	0,67	0,82	9,00	0,67	0,82	9,00	2,00	1,41	8,33	0,89	0,94	9,33	0,22	0,47
Aula 9	7,40	1,84	1,36	8,33	1,56	1,25	9,00	0,67	0,82	8,67	1,56	1,25	8,00	0,67	0,82	9,33	0,22	0,47
Aula 11	8,40	1,44	1,2	8,6	1,44	1,2	8,8	0,56	0,75	8,8	1,36	1,17	8,8	1,36	1,17	9	0,80	0,89
Aula 13	8,00	3,20	1,79	8,4	2,24	1,5	8,6	1,04	1,02	9	0,80	0,89	8,80	1,36	1,17	9,00	0,80	0,89
Aula 15	7,80	1,36	1,17	8,40	1,04	1,02	8,80	0,96	0,98	8,80	0,56	0,75	8,60	1,04	1,02	8,40	1,04	1,02
Aula 17	7,40	1,04	1,02	7,60	1,04	1,02	8,20	1,36	1,17	8,80	0,56	0,75	8,40	1,04	1,02	8,20	0,56	0,75
Aula 19	9,00	1,20	1,1	8,20	0,96	0,98	8,8	0,56	0,75	8,80	0,56	0,75	8,80	0,56	0,75	8,60	0,64	0,80
μ	**7,84**	**2,07**	**1,31**	**8,09**	**1,64**	**1,21**	**8,47**	**1,78**	**1,13**	**8,75**	**1,26**	**1,04**	**8,37**	**1,28**	**1,08**	**8,77**	**0,91**	**0,85**

pelo fato de na aula anterior online, os alunos terem contato com a condicional *if..else* no mesmo nível, sendo que na presente aula foi reforçado sobre o conceito, e posteriormente conectado com o conceito com listas ao trabalhar o uso da estrutura condicional para verificar se um item existe dentro da lista com o comando *in*.

A tabela da Figura 4.4 apresenta os resultados da mensuração da carga cognitiva para cada aula da turma 2 que trabalhou diretamente com o Python no Google Colab. A primeira aula assim como aconteceu na Turma 1 apresentou valores menores na mensuração da carga cognitiva de cada um dos itens nas aulas; e também no geral ao passo que a dispersão dos dados, representados pelos valores altos de desvio padrão e variância como podemos observar na tabela da Figura **??**, que pode ser explicado pelos da mesma forma pelos

	Carga Intrínseca			Carga Estranha			Geral		
	μ			μ			μ		
	μ	σ²	σ	μ	σ²	σ	μ	σ²	σ
Aula 3	6,58	7,31	2,68	7,42	3,69	1,92	7,00	5,50	2,30
Aula 5	8,22	0,52	0,70	9,11	0,44	0,63	8,67	0,48	0,67
Aula 7	8,47	0,79	0,89	8,89	1,04	0,94	8,68	0,91	0,91
Aula 9	8,24	1,36	1,14	8,67	0,82	0,85	8,46	1,09	1,00
Aula 11	8,60	1,15	1,05	8,87	1,17	1,08	8,73	1,16	1,06
Aula 13	8,33	2,16	1,44	8,93	0,99	0,98	8,63	1,57	1,21
Aula 15	8,33	1,12	1,06	8,60	0,88	0,93	8,47	1,00	0,99
Aula 17	7,73	1,15	1,07	8,47	0,72	0,84	8,10	0,93	0,96
Aula 19	8,67	0,91	0,94	8,73	0,59	0,77	8,70	0,75	0,86
μ	8,13	1,83	1,22	8,63	1,15	0,99	8,38	1,49	1,11

Tabela 4.5: Resultado da Mensuração da Carga Cognitiva nas aulas presenciais da turma 2 com o uso do Python no Google Colab agrupado por carga cognitiva, carga estranha e geral

conhecimentos desnivelados no curso no início do curso, reflexo da avaliação diagnóstica da primeira aula.

	Carga Intrínseca									Carga Estranha								
	Item 1			Item 2			Item 3			Item 4			Item 5			Item 6		
	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ	μ	σ2	σ
Aula 2	6,75	8,19	2,86	7,75	5,19	2,28	6,00	12,50	3,54	7,75	2,19	1,48	7,00	13,50	3,67	7,75	3,69	1,92
Aula 4	6,00	4,00	2	6,00	4,00	2,00	6,00	4,00	2,00	8,00	4,00	2,00	6,00	4,00	2,00	6,00	9,00	3,00
Aula 6	7,00	9,50	3,08	7,50	4,25	2,06	6,25	0,19	0,43	7,00	12,50	3,54	6,75	12,19	3,49	8,25	3,19	1,79
Aula 8	8,00	4,00	2	8,00	4,00	2,00	6,00	1,00	1,00	8,50	2,25	1,50	8,00	4,00	2,00	8,00	4,00	2,00
μ	6,94	6,42	2,49	7,31	4,36	2,08	6,06	4,42	1,74	7,81	5,23	2,13	6,94	8,42	2,79	7,50	4,97	2,18

Tabela 4.6: Resultado da Mensuração da Carga Cognitiva nas aulas online da turma 2 com o uso do Python no Google Colab para cada item da carga cognitiva

As demais aulas presenciais da turma 2 apresentaram uma média para a carga intrínseca e estranha altas, que representam uma baixa carga cognitiva nas aulas e consequentemente melhor aprendizado, e também nessas aulas observa-se a turma mais homogênea em conhecimento. Ressalta-se que nessas aulas foram aplicadas as recomendações pedagógicas referente aos efeitos da Teoria da Carga Cognitiva e Aprendizagem por Andaimes, e por mais que as ferramentas utilizadas não tinham a inserção gradual de conceitos e sintaxe em atividades já definidas como no Hedy utilizado na turma 1, o curso foi desenhado para que isso acontecesse, sendo que somente a sintaxe não permitiu que fosse realizada dessa forma. No entanto, simplificamos a mesma em algumas aulas para que posteriormente fosse apresentada a forma completa de algumas estruturas. Um exemplo ocorreu com o conceito da condicional *if..else*, em que a primeira sintaxe apresentada foi a ternária e posteriormente

incluímos com exemplos similares a sintaxe com a identação, conforme exemplificado na Figura 4.1.

```
idade = 14

#Condicional com o if ternário sem identação
print("Você pode votar.") if idade >= 16 else print("Você não pode votar.")

#Condicional normal do if com identação
if(idade>=16):
  print("Você pode votar.")
else:
  print("Você não pode votar.")
```

Figura 4.1: Exemplos de utilização do *if* ternário com sintaxe simplificada e *if* normal com identação

Os alunos apontaram maior carga cognitiva nas aulas online em comparação com as aulas presenciais da turma 2 como podemos observar na tabela da Figura **??**. Nas aulas foi disponibilizado um roteiro de estudo com explicação com textos e vídeos, para que o aluno tivesse acesso à explicação em mais de um formato diferente.

Nossa hipótese inicial era que um roteiro de estudos em um horário flexível permitiria que o aluno estudasse no seu ritmo, com apoio de vídeos gravados exclusivamente para a turma, favoreceria uma redução da carga cognitiva geral. No entanto, observamos um média maior da carga cognitiva dos alunos nas aulas e também uma dispersão maior, mostrando que a turma estava menos homogênea em conhecimento nestes momentos.

	Carga Intrínseca			Carga Estranha			Geral		
	μ			μ			μ		
	μ	σ²	σ	μ	σ²	σ	μ	σ²	σ
Aula 2	6,83	8,63	2,89	7,50	6,46	2,36	7,17	7,54	2,62
Aula 4	6,00	4,00	2,00	6,67	5,67	2,33	6,33	4,83	2,17
Aula 6	6,92	4,65	1,86	7,33	9,29	2,94	7,13	6,97	2,40
Aula 8	7,33	3,00	1,67	8,17	3,42	1,83	7,75	3,21	1,75
μ	6,77	5,07	2,10	7,42	6,21	2,37	7,09	5,64	2,23

Tabela 4.7: Resultado da Mensuração da Carga Cognitiva nas aulas online da turma 2 com o uso do Python no Google Colab agrupado por carga cognitiva, carga estranha e geral.

De acordo com os resultados obtidos nas duas turmas, observamos que houve uma baixa carga cognitiva apontados pelos alunos na ferramenta de mensuração aplicada durante as aulas. Nos cursos de programação introdutória com Hedy na Turma 1 os alunos apresentaram uma carga cognitiva nas

aulas presenciais que manteve-se consistente na maioria das aulas do curso. Tinha-se como hipótese inicial que a Turma 1 apresentaria menor carga cognitiva em relação a Turma 2, pelo fato de utilizar o Hedy que ao incorporar a Teoria da Carga Cognitiva introduz os conceitos e sintaxe de programação de forma gradual.

Na prática, observamos que apesar da ferramenta ter o intuito dos alunos utilizarem sozinhos, eles necessitam do apoio de um docente/tutor durante a jornada de aprendizado, tal fato evidenciou-se quando eles apresentavam dificuldades em realizar alguns desafios do nível em que encontravam-se nas aulas online, mesmo com os vídeos de apoio.

Os níveis de Hedy que tinham como objetivo a inserção de sintaxe apenas, observou-se dificuldade de compreensão dos alunos, principalmente quando alguns comandos deixavam de existir de um nível para o outro. Como exemplo, o nível 1 apresenta três comandos, sendo: *print* (para saída de dados), *ask* (para entrada de dados) e *echo* que armazena a entrada de *ask* e quando executado, imprime o texto a sua frente e logo após a informação guardada na entrada de *ask*, conforme exemplificado na Figura 4.2. No nível 2, o comando *echo* é retirado para introduzir as variáveis (conforme apresentado na Figura 4.3, o que gerou confusão nos alunos, pois mesmo com a orientação da docente, por diversas vezes no nível 2 eles tentavam usar o comando *echo*.

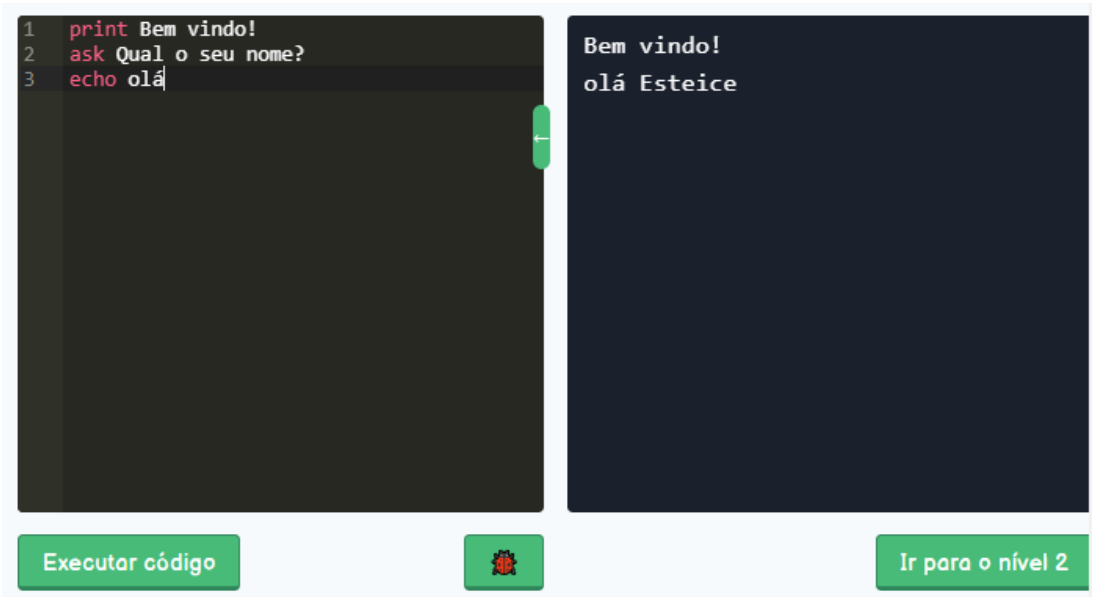

Figura 4.2: Exemplo dos comandos utilizados no Nível 1 de Hedy. A autora, 2023.

Com base nas tabelas apresentadas anteriormente, é possível observar que, em comparação com as aulas presenciais de Python, as aulas online na Turma 2 apresentaram resultados ligeiramente inferiores em termos de taxa de sucesso dos alunos.

A carga intrínseca refere-se à complexidade do conteúdo em si, enquanto

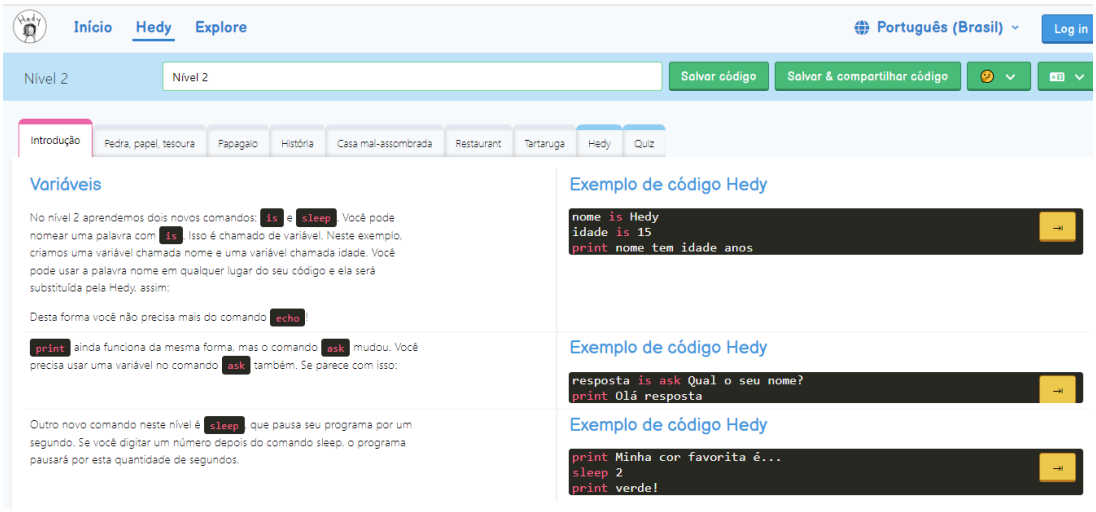

Figura 4.3: Comandos utilizados no Nível 2 de Hedy

a carga estranha refere-se às demandas adicionais colocadas sobre o aluno durante a aprendizagem. De acordo com a Teoria da Carga Cognitiva, uma menor carga cognitiva indica um aprendizado mais eficaz, onde os elementos de aprendizagem interagem de forma mais fluente.

No caso das aulas online de Python, se houve uma maior carga intrínseca, isso sugere que o conteúdo apresentado nas aulas online era mais complexo em comparação com as aulas presenciais. Isso pode ser atribuído a vários fatores, como a necessidade de os alunos navegarem por roteiros de estudos, textos e vídeos de apoio de forma independente. A autonomia exigida nesse formato pode aumentar a complexidade do processo de aprendizagem, exigindo dos alunos uma maior habilidade de autorregulação e organização.

Além disso, se houve uma maior carga estranha nas aulas online de Python, isso indica que os alunos enfrentaram demandas adicionais durante a aprendizagem. Embora recursos como exemplos práticos, exercícios e suporte adicional tenham sido fornecidos para ajudar na assimilação do conteúdo, a falta de interações diretas com o instrutor pode ter aumentado a carga estranha.

Nesse sentido, as diferenças no resultado das aulas online em relação às aulas presenciais foram mínimas, o que sugere que as estratégias implementadas no curso, como os exemplos práticos, exercícios e recursos adicionais, bem como as videoaulas de tira-dúvidas, foram efetivas em lidar com a carga cognitiva e fornecer suporte aos alunos.

É importante reconhecer que ainda existem áreas de melhoria. Com base na análise considerando a Teoria da Carga Cognitiva e o Scafollding Learning, podemos identificar algumas sugestões para aprimorar ainda mais as aulas online em Python:

- Aprimorar a clareza e organização do material didático: Certificar-se de que o conteúdo seja apresentado de forma clara, com instruções passo

a passo e exemplos bem estruturados. Isso ajudará a reduzir a carga cognitiva intrínseca e facilitar a assimilação do conteúdo.

- Ampliar as interações entre alunos e instrutor: Incentivar a participação ativa dos alunos por meio de fóruns de discussão, grupos de estudo ou sessões de perguntas e respostas ao vivo. Isso promoverá um ambiente de aprendizagem mais colaborativo, permitindo que os alunos compartilhem dúvidas, ideias e soluções.

- Oferecer suporte individualizado: Além das videoaulas de tira-dúvidas, considerar a possibilidade de oferecer sessões de tutoria individual ou feedback personalizado para os alunos. Isso ajudará a atender as necessidades individuais dos alunos, fornecendo um suporte mais direcionado e facilitando a superação de obstáculos cognitivos.

- Avaliar e ajustar a carga cognitiva: Regularmente avaliar a carga cognitiva exigida pelos materiais e atividades do curso. Se necessário, fazer ajustes para otimizar a distribuição de informações e tarefas ao longo do tempo, evitando sobrecarregar os alunos.

No entanto, é importante destacar que as diferenças nos resultados podem ter sido influenciadas por vários fatores, como o nível de engajamento dos alunos, sua familiaridade com o formato online de aprendizagem e suas preferências individuais de estudo. Portanto, é necessário analisar esses resultados com cautela e considerar que outros fatores além do formato de ensino podem ter influenciado os resultados finais.

4.4 Limitações do estudo

Este estudo teve algumas limitações importantes que devem ser consideradas ao interpretar os resultados referentes à mensuração da carga cognitiva durante o curso de programação nas duas turmas analisadas neste estudo.

Em primeiro lugar, a amostra foi relativamente pequena, o que pode ter afetado a precisão dos resultados. Além disso, o estudo foi realizado apenas em uma única escola em um curto período de tempo, o que limita a generalização dos resultados para outras áreas geográficas e escolas da mesma região como de Educação Profissional, particulares e de outros períodos, visto que foi aplicado pós-aula para alunos que estudavam na escola em período integral; além de não capturar mudanças ao longo do tempo.

No entanto, acreditamos que os resultados fornecem uma compreensão valiosa do tema em questão, principalmente por trazer recomendações pedagógicas que são pouco exploradas de forma conjunta no contexto de ensino

de programação, servindo como base para futuros estudos com uma amostra mais ampla e diversificada e em uma variedade de locais por um período de tempo mais longo.

CAPÍTULO

5

Considerações Finais

Neste estudo, validamos um instrumento adaptado para mensuração de carga cognitiva dos alunos durante um curso de programação introdutória para duas turmas da segunda fase do Ensino Fundamental.

Ao estruturar o curso, foi proposta uma trilha de aprendizagem em cursos de programação introdutória para iniciantes com base na Aprendizagem por Andaimes e Teoria da Carga Cognitiva, o que gerou recomendações pedagógicas para trabalhar os conceitos de programação introdutória que podem ser aplicadas em outros estudos.

Para analisar o instrumento de mensuração da carga cognitiva foram utilizadas ferramentas distintas de programação introdutória em cada uma das turmas. A primeira turma utilizou Hedy que nativamente já incorpora a Teoria da Carga Cognitiva em sua trilha de aprendizagem que é dividida por níveis, sendo que os conceitos e sintaxe de programação introdutória são inseridas gradualmente até que o aluno chegue ao final programando em uma linguagem muito próxima ao Pyhton. Já a segunda turma utilizou a IDE Google Colab com Python e incorporamos na estruturação do curso as recomendações pedagógicas da Aprendizagem por Andaimes e Teoria da Carga Cognitiva. As duas turmas aconteceram em formato híbrido com aulas presenciais e online.

A média da carga cognitiva geral mensurada através do instrumento proposto nesse estudo apontou uma dispersão mais alta nas aulas online que presenciais, ao contrário de nossa hipótese inicial. Além disso, algumas aulas que apresentaram novos conceitos ou incrementavam sintaxe apresentaram um valor maior da carga cognitiva geral na turma que utilizou Hedy em comparação à segunda que utilizou o Pyhton com o Google Colab.

5.1 Recomendações para o futuro

Durante essa pesquisa identificamos as seguintes lacunas, desafios e oportunidades para futuras pesquisas sobre o tema, que serão apresentados nas subseções a seguir.

5.1.1 Sistemas de Aprendizagem Adaptativos

Desenvolver sistemas adaptativos de ensino para o ensino de programação, incorporando as recomendações da Teoria da Carga Cognitiva e Aprendizagem por Andaimes (andaimes por computador). Durante nossa pesquisa, identificamos uma grande dispersão da carga cognitiva entre os alunos, o que ressalta a necessidade de um sistema que apoie o docente na adaptação. Cada aluno possui um estilo de aprendizagem diferente - alguns são aprendizes auditivos, outros visuais e outros cinestésicos. Portanto, é importante fornecer estímulos de aprendizagem adequados para cada estilo.

5.1.2 Análise da aplicação de Linguagem Textual e Linguagem de Programação em Blocos

Linguagens de programação em blocos têm sido amplamente utilizadas em cursos de programação para iniciantes, especialmente com estudantes crianças e adolescentes. Essas linguagens oferecem uma vantagem significativa, pois os alunos não precisam se preocupar com a sintaxe da linguagem, que é estruturada com formatos e cores para evitar erros conceituais. Isso permite que os alunos se concentrem no aprendizado dos conceitos de programação em vez de se preocupar com detalhes sintáticos.

Nesse contexto, é essencial mensurar a carga cognitiva dos alunos usando as ferramentas propostas neste trabalho, de forma a comparar turmas com o mesmo perfil de alunos, mas que utilizam abordagens diferentes no ensino da programação: programação em blocos, pseudocódigos com Portugol e uma linguagem de programação de alto nível como Hedy. Ao incorporar a Teoria da Carga Cognitiva e a Aprendizagem por Andaimes em uma sequência pedagógica, espera-se obter um volume maior de dados com um público mais uniforme, permitindo analisar até que ponto a escolha da ferramenta de ensino de linguagem de programação impacta nos resultados e como esses resultados se correlacionam. Essa análise é fundamental para melhorar a eficácia do ensino de programação e promover uma aprendizagem mais eficiente e profunda entre os alunos.

5.1.3 Análise cruzada com outras ferramentas

Uma outra lacuna de pesquisa é a aplicação de formulários que avaliem outros aspectos além da Teoria da Carga Cognitiva em turmas de programação introdutória. Uma possibilidade é analisar a aceitação da tecnologia pelo público-alvo utilizando ferramentas como o Modelo de Aceitação de Tecnologia (TAM). Com o TAM, é possível verificar como os alunos percebem e adotam as ferramentas utilizadas no ensino de programação, o que pode fornecer insights valiosos sobre a eficácia e a aceitação das abordagens de ensino.

Além disso, a aplicação de heurísticas de avaliação no uso das ferramentas pode ser uma abordagem útil para avaliar aspectos como divertimento e usabilidade das mesmas. Heurísticas populares, como as de Nielsen [Nielsen and Molich (1990)] e [Malone Malone (1982)] têm sido aplicadas em estudos para analisar diferentes ambientes de programação em blocos para crianças Batista (2017). Essa análise é importante porque, ao considerar a dimensão da experiência do aluno com a ferramenta, podemos inferir que quando o aluno se diverte e encontra a ferramenta fácil de usar, ele provavelmente se engajará mais nas atividades propostas. Essa premissa é essencial para aplicar com sucesso as recomendações da Teoria da Carga Cognitiva e garantir uma aprendizagem mais efetiva e envolvente no ensino de programação.

Referências Bibliográficas

Batista, E. J. S. (2017). Uma análise de ambientes de programaçao em blocos com base em recomendaçoes de interaçao criança-computador. *Trabalho de Conclusao de Curso-UFMS*. Citado na página 47.

Belland, B. R. (2017). *Instructional scaffolding in STEM education: Strategies and efficacy evidence*. Springer Nature. Citado na página 16.

Brackmann, C. P. (2017). Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica. Citado na página 6.

Davis, C. e Oliveira, Z. d. (1994). *Psicologia na educação*. Cortez. Citado na página 16.

de Holanda, W. D., de Paiva Freire, L., e da Silva Coutinho, J. C. (2019). Estratégias de ensino-aprendizagem de programação introdutória no ensino superior: uma revisão sistemática da literatura. *RENOTE-Revista Novas Tecnologias na Educação*, 17(1):527–536. Citado na página 1.

Duran, R., Zavgorodniaia, A., e Sorva, J. (2022). Cognitive load theory in computing education research: A review. *ACM Transactions on Computing Education (TOCE)*. Citado nas páginas 2 e 9.

França, R. e Tedesco, P. (2015). Desafios e oportunidades ao ensino do pensamento computacional na educação básica no brasil. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, pagina 1464. Citado na página 5.

Gibbons, P. (2002). *Scaffolding language, scaffolding learning*. Heinemann Portsmouth, NH. Citado na página 2.

Hermans, F. (2020). Hedy: A gradual language for programming education. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, páginas 259–270. Citado nas páginas 1, 2, e 21.

Júnior, D. L. G. e Bogea, D. T. R. (2020). Ensino de programação: Uma revisão sistemática e as aplicações ao ensino profissional. *Cadernos da FUCAMP*, 19(41). Citado na página 21.

Leppink, J., Paas, F., Van der Vleuten, C. P., Van Gog, T., e Van Merriënboer, J. J. (2013). Development of an instrument for measuring different types of cognitive load. *Behavior research methods*, 45(4):1058–1072. Citado nas páginas 14, 25, e 30.

Leppink, J., Paas, F., Van Gog, T., van Der Vleuten, C. P., e Van Merrienboer, J. J. (2014). Effects of pairs of problems and examples on task performance and different types of cognitive load. *Learning and instruction*, 30:32–42. Citado nas páginas 14, 15, e 25.

Malone, T. W. (1982). Heuristics for designing enjoyable user interfaces: Lessons from computer games. In *Proceedings of the 1982 conference on Human factors in computing systems*, páginas 63–68. ACM. Citado na página 47.

Medeiros, R. P., Ramalho, G. L., e Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90. Citado na página 7.

Ministério da Educação, M. (2022a). Documentos consolidados da computação na bncc. Disponível em: http://portal.mec.gov.br/index.php?option=com$_docmanview = downloadalias = 233371 - documentos - consolidados - comp - bncc - xlsxcategory_slug = janeiro - 2022 - pdfItemid = 30192. Citadonapgina$ 6.

Ministério da Educação, M. (2022b). Normas sobre computação na educação básica – complemento à bncc. Disponível em: http://portal.mec.gov.br/index.php?option=com$_docmanview = downloadalias = 182481 - texto - referencia - normas - sobre - computacao - na - educacao - basicacategory_slug = abril - 2021 - pdfItemid = 30192. Citadonapgina$ 6.

Morrison, B. B., Dorn, B., e Guzdial, M. (2014). Measuring cognitive load in introductory cs: adaptation of an instrument. In *Proceedings of the tenth annual conference on International computing education research*, páginas 131–138. Citado nas páginas 15, 25, 29, e 30.

Nielsen, J. e Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, páginas 249–256. ACM. Citado na página 47.

Schoeffel, P., Moser, P., Varela, G., Durigon, L., de Albuquerque, G. C., e Niquelatti, M. (2015). Uma experiência no ensino de pensamento computacional

para alunos do ensino fundamental. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, pagina 1474. Citado na página 5.

Sociedade Brasileira de Computação, S. (2019). Diretrizes para ensino de computação na educação básica. Disponível em: https://www.sbc.org.br/documentos-da-sbc/send/203-educacao-basica/1220-bncc-em-itinerario-informativo-computacao-2. Citado na página 6.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285. Citado na página 8.

Sweller, J. (2010). Cognitive load theory: Recent theoretical advances. Citado nas páginas 8 e 9.

van Rossum, G. e de Boer, J. (1991). Interactively testing remote servers using the python programming language. *CWI quarterly*, 4(4):283–303. Citado na página 20.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35. Citado na página 6.

Wood, D., Bruner, J. S., e Ross, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100. Citado nas páginas 16, 18, e 19.

Serviço Público Federal
Ministério da Educação
Fundação Universidade Federal de Mato Grosso do Sul

ATA DE DEFESA DE DISSERTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

MESTRADO

Aos vinte e dois dias do mês de junho do ano de dois mil e vinte e três, às nove horas, na Fundação Universidade Federal de Mato Grosso do Sul, reuniu-se a Banca Examinadora composta pelos membros Amaury Antônio de Castro Júnior (UFMS), Anderson Corrêa de Lima (UFMS), Graziela Santos de Araújo (UFMS) e Ismar Frango SIlveira (Mackenzie), sob a presidência do primeiro, para julgar o trabalho da aluna **ESTEIC JANAÍNA SANTOS BATISTA**, CPF 040.977.861-30, do Programa de Pós-Graduação em Computação Aplicada, Curso de Mestrado, da Fundação Universidade Federal de Mato Grosso do Sul, apresentado sob o título **"Mensuração da carga cognitiva em um curso de programação introdutória para crianças"** e orientação de Amaury Antônio de Castro Júnior. O presidente da Banca Examinadora declarou abertos os trabalhos e agradeceu a presença de todos os Membros. A seguir, concedeu a palavra à aluna que expôs sua Dissertação. Terminada a exposição, os senhores membros da Banca Examinadora iniciaram as arguições. Terminadas as arguições, o presidente da Banca Examinadora fez suas considerações. A seguir, a Banca Examinadora reuniu-se para avaliação, e após, emitiu parecer expresso conforme segue:

EXAMINADOR	AVALIAÇÃO
Dr. Amaury Antônio de Castro Júnior (Facom/UFMS - orientador)	APROVADO
Dr. Anderson Corrêa de Lima (CPPP/UFMS - coorientador)	APROVADO
Dra. Graziela Santos de Araújo (FACOM/UFMS - membro externo)	APROVADO
Dr. Ismar Frango SIlveira (Mackenzie - membro externo)	APROVADO
Dra. Luciana Montera Cheung (FACOM/UFMS - membro externo suplente)	—
Dr. Cláudio Zarate Sanavria (IFMS - membro externo suplente)	—

Resultado final: [X] Aprovação [] Reprovação

Observações: Sem observações.

Este é o parecer.

Documento assinado eletronicamente por **Amaury Antonio de Castro Junior**, **Professor do Magisterio Superior**, em 22/06/2023, às 11:24, conforme horário oficial de Mato Grosso do Sul, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Anderson Correa de Lima**, **Professor do Magisterio Superior**, em 22/06/2023, às 12:23, conforme horário oficial de Mato Grosso do Sul, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Graziela Santos de Araujo**, **Professora do Magistério Superior**, em 22/06/2023, às 13:26, conforme horário oficial de Mato Grosso do Sul, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Ismar Frango Silveira**, **Usuário Externo**, em 04/07/2023, às 18:33, conforme horário oficial de Mato Grosso do Sul, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufms.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4138035** e o código CRC **E8FF9082**.

COLEGIADO DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

Av Costa e Silva, s/nº - Cidade Universitária

Fone: (67)3345-7456

CEP 79070-900 - Campo Grande - MS

Referência: Processo nº 23104.020125/2023-98 SEI nº 4138035