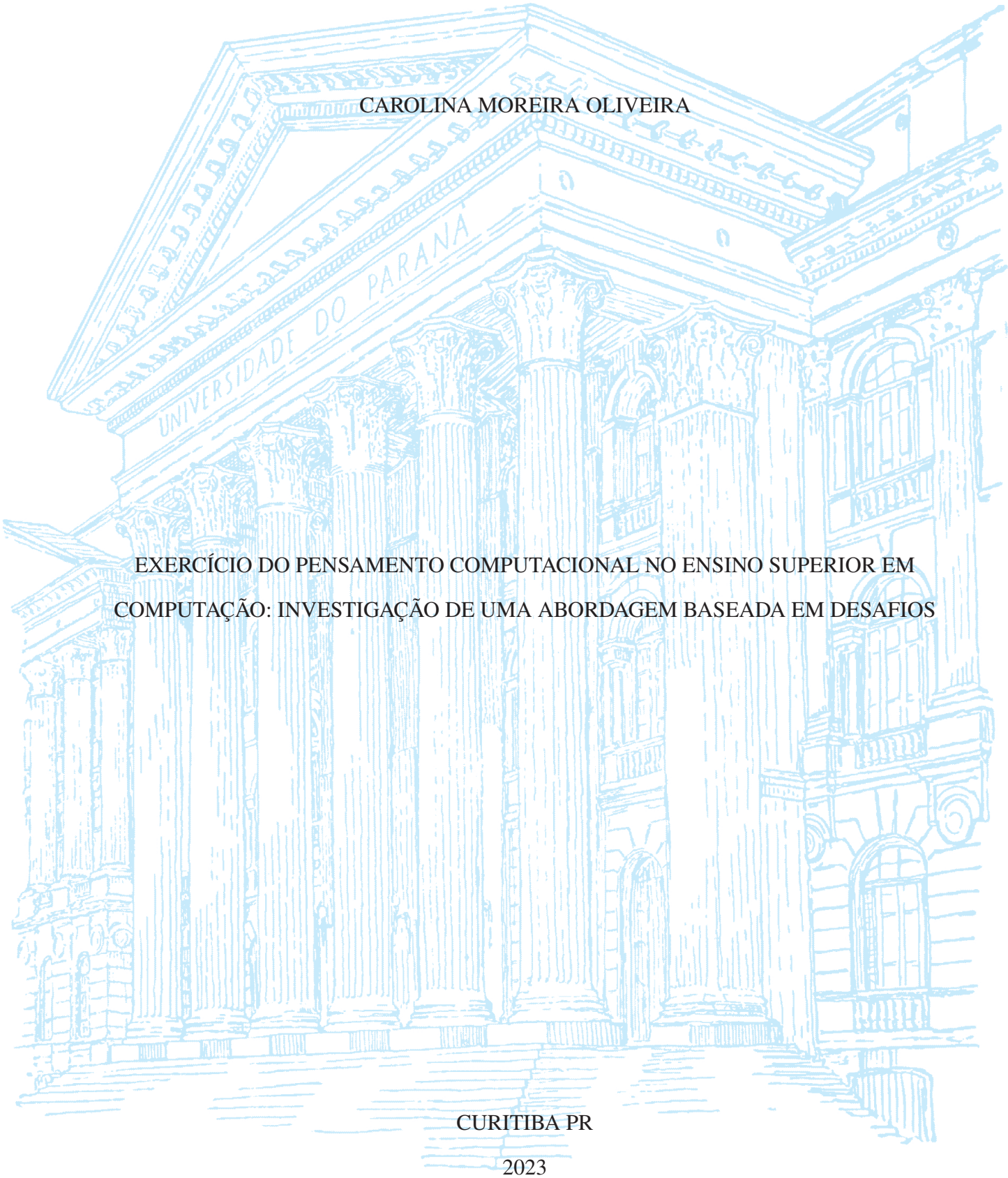
UNIVERSIDADE FEDERAL DO PARANÁ

CAROLINA MOREIRA OLIVEIRA



EXERCÍCIO DO PENSAMENTO COMPUTACIONAL NO ENSINO SUPERIOR EM COMPUTAÇÃO: INVESTIGAÇÃO DE UMA ABORDAGEM BASEADA EM DESAFIOS

CURITIBA PR

2023

CAROLINA MOREIRA OLIVEIRA

EXERCÍCIO DO PENSAMENTO COMPUTACIONAL NO ENSINO SUPERIOR EM COMPUTAÇÃO: INVESTIGAÇÃO DE UMA ABORDAGEM BASEADA EM DESAFIOS

Tese apresentada como requisito parcial à obtenção do grau de Doutora em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Dr. Roberto Pereira.

CURITIBA PR

2023

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Oliveira, Carolina Moreira
Exercício do pensamento computacional no ensino superior em computação: investigação de uma abordagem baseada em desafios / Carolina Moreira Oliveira. – Curitiba, 2023.
1 recurso on-line : PDF.

Tese (Doutorado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.

Orientador: Roberto Pereira

1. Informática – Estudo e ensino. 2. Escolas de informática – Ensino superior. 3. Pensamento computacional. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Informática. III. Pereira, Roberto. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **CAROLINA MOREIRA OLIVEIRA** intitulada: **EXERCÍCIO DO PENSAMENTO COMPUTACIONAL NO ENSINO SUPERIOR EM COMPUTAÇÃO: INVESTIGAÇÃO DE UMA ABORDAGEM BASEADA EM DESAFIOS**, sob orientação do Prof. Dr. ROBERTO PEREIRA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutora está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 11 de Abril de 2023.

Assinatura Eletrônica
13/04/2023 19:52:14.0
ROBERTO PEREIRA
Presidente da Banca Examinadora

Assinatura Eletrônica
18/04/2023 17:01:14.0
LETICIA MARA PERES
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
13/04/2023 19:09:12.0
ISMAR FRANGO SILVEIRA
Avaliador Externo (UNIVERSIDADE PRESBITERIANA MACKENZIE)

Assinatura Eletrônica
14/04/2023 08:47:49.0
RITA CRISTINA GALARRAGA BERARDI
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Chegou a hora de sonhar de novo!

RESUMO

Diretrizes curriculares da graduação em Computação recomendam que estudantes desenvolvam o Pensamento Computacional, mas não especificam quando e como essas habilidades devem ser desenvolvidas. A literatura na área também reconhece a importância e relevância do Pensamento Computacional, entretanto, ainda existem poucas evidências em torno de como apoiar o exercício dessas habilidades no ensino superior em Computação. Diante do exposto, esta pesquisa teve como objetivo investigar uma abordagem para apoiar o exercício do Pensamento Computacional no ensino superior em Computação. A abordagem Pense Computacionalmente foi criada com o propósito de apoiar professores e pesquisadores durante a concepção e condução de práticas educacionais que promovam o Pensamento Computacional por meio do processo de solução dos desafios que considera os quatro pilares do Pensamento Computacional. A abordagem tem como premissa que o Pensamento Computacional deve ser explicitamente exercitado no primeiro semestre da graduação de forma significativa, colaborativa, contextualizada, desplugada e baseada em desafios. Desenvolvemos a abordagem para ser utilizada ou adaptada para diferentes contextos educacionais, sendo composta por: um conjunto de orientações sobre como preparar e conduzir o exercício do Pensamento Computacional; um conjunto de sete práticas (desafios) para serem utilizadas ou inspirar a criação de novas; um método para guiar a condução da prática e instruir estudantes a proporem uma solução; e um artefato para apoiar a análise das soluções para coletar evidências do exercício do Pensamento Computacional. Para experimentarmos a abordagem e investigar as potencialidades e fragilidades, realizamos seis estudos de caso. Os estudos de caso foram conduzidos em ofertas diferentes da disciplina de Introdução à Ciência da Computação, dos cursos de Bacharelado em Ciência da Computação e Informática Biomédica da UFPR. Os resultados revelam que os desafios foram bem avaliados tanto pelos docentes quanto discentes, e foram muito bons em provocar o raciocínio e promover o interesse. Além disso, os resultados demonstram que estudantes exercitaram o Pensamento Computacional e desenvolveram a percepção do exercício dessas habilidades. Desse modo, atingimos o objetivo específico de investigar uma abordagem para apoiar o exercício do Pensamento Computacional no ensino superior em Computação. Como resultados desta pesquisa também podemos citar: contribuições conceituais: definição do Pensamento Computacional e suas habilidades no contexto do ensino superior em Computação; relação dos passos realizados durante a resolução de um desafio com as habilidades do Pensamento Computacional; e definição de como coletar evidências do exercício do Pensamento Computacional; contribuição metodológica: como conduzir atividades que foquem no exercício do Pensamento Computacional no ensino superior em Computação; e contribuição técnica: abordagem formatada em material didático.

Palavras-chave: Pensamento Computacional, Educação em Computação, Ensino Superior

ABSTRACT

The curriculum guidelines for Computing undergraduate courses point out that computational thinking should be a skill exercised by all students in those courses. However, they do not reveal when and how these skills should be developed. The literature in the area also recognizes the importance and relevance of Computational Thinking. There is still little evidence around how to support the exercise of these skills in higher education in Computing. Given the above, this research aimed to investigate an approach to support the exercise of Computational Thinking in higher education in Computing. The *Pense Computacionalmente* approach was created to support teachers and researchers during the design and conduct of educational practices that promote the exercise of computational thinking through the process of solving challenges considering the four pillars of computational thinking. The approach is based on the premise that Computational Thinking should be explicitly exercised in the first semester of undergraduate courses in a meaningful, collaborative, contextualized, unplugged, and challenge-based way. We developed the approach to be used or adapted for different educational contexts, comprising: a set of guidelines on how to prepare and conduct the exercise of Computational Thinking; a set of seven practices (challenges) to be used or to inspire the creation of new ones; a method to guide the conduct of the practice and instruct the students to propose a solution; and an artifact to support the analysis of solutions to collect evidence from the exercise of Computational Thinking. In order to try out the approach and investigate its strengths and weaknesses, we carried out six case studies. The case studies were conducted in different offerings of the subject Introduction to Computer Science, of the Bachelor's Degree in Computer Science and Biomedical Informatics at UFPR. The studies results reveal that the challenges were well evaluated by both teachers and students, and were very good at provoking reasoning and promoting students' interest. In addition, the results demonstrate that students exercised Computational Thinking and developed the perception of exercising these skills. In this way, we achieved the specific objective of investigating an approach to support the exercise of Computational Thinking in higher education in Computing. As results of this research, we can also mention: conceptual contributions: the definition of Computational Thinking and its abilities in the context of higher education in Computing; the relationship of the steps taken during the challenge resolution with Computational Thinking skills; and the characterization of how to collect evidence of the exercise of the Computational Thinking; methodological contribution: how to conduct activities that focus on the exercise of Computational Thinking in higher education in Computing; and technical contribution: approach formatted in didactic material.

Keywords: Computational Thinking, Computer Science Education, Higher Education

LISTA DE FIGURAS

2.1 Componentes básicos de resolução de problemas de acordo com Naur – tradução Sveinsdottir e Frøkjær (1988) . . . 18
2.2 Componentes fundamentais na resolução de problemas usando computadores de acordo com Naur – tradução Sveinsdottir e Frøkjær (1988) . . . 18

4.1 Habilidades exercitadas por desafio . . . 33
4.2 Folha do Teste dos Labs . . . 33
4.3 Método para guiar a resolução das práticas . . . 36
4.4 Relação entre as Etapas e Habilidades . . . 37
4.5 Planilha da explicação do artefato . . . 41
4.6 Planilha base para os desafios . . . 41

5.1 Personas criadas pelos estudantes de CI1003 2019/1 . . . 47
5.2 Percepção dos estudantes com relação aos desafios realizados na disciplina . . . 49
5.3 Percepção dos estudantes com relação aos desafios ao seu aprendizado de CI1055 50
5.4 Percepção dos estudantes com relação a realização de atividades em grupos . . . 51
5.5 Percepção dos estudantes com relação às atividades em grupo e individual . . . 52
5.6 Dispositivos utilizados para acompanhar as aulas e atividades da disciplina CI1003 56
5.7 Percepção dos estudantes com relação a qualidade da internet deles . . . 56
5.8 Percepção dos estudantes sobre os desafios considerando nível de dificuldade, qualidade da explicação, capacidade de provocar o raciocínio e capacidade de promover o interesse . . . 57
5.9 Percepção dos estudantes sobre o exercício das habilidades do Pensamento Computacional . . . 60
5.10 Percepção dos estudantes sobre as habilidades exercitadas em cada um dos desafios 60

A.1 Most 100 frequent words used in the title of 2,966 papers . . . 90
A.2 Search returned papers overview: publication venues and publications per year . 90
A.3 PRISMA Flow Diagram for the systematic mapping study . . . 91
A.4 Distribution of selected papers around the world . . . 92
A.5 Names of institutions with at least two papers . . . 92
A.6 Selected papers overview: publication venues and publications per year . . . 93
A.7 Overview of papers distribution by publication venues . . . 93
A.8 Overview of computational thinking definition . . . 95
A.9 Initiatives' Contents . . . 97
A.10 Tools used in the initiatives . . . 98

A.11 Skills and amount of papers that mention each skill (*n=39*) 99
A.12 Data collection methods. (*n=80*) 102

LISTA DE TABELAS

2.1 Habilidades do Pensamento Computacional Selecionadas 21

4.1 Descrição das soft skills 31
4.2 Relação das habilidades do Pensamento Computacional com os passos dos desafios. 38
4.3 Definição das habilidades e enumeração das evidências a serem coletadas 39

5.1 Detalhes técnicos dos estudos de caso 44
5.2 Resumo dos estudos de caso realizados. 46
5.3 Temas das respostas sobre percepção dos estudantes com relação dos desafios ao seu aprendizado de CI1055 50
5.4 Temas das respostas sobre motivos que gosta de realizar atividades em grupo . . 52
5.5 Temas das respostas sobre motivos que não gosta de realizar atividades em grupo 52
5.6 Sumarização das respostas das perguntas abertas 53
5.7 Temas das respostas sobre Percepção dos estudantes com relação dos desafios ao seu aprendizado de CI1055 58
5.8 Temas das respostas sobre percepção dos estudantes com relação dos desafios ao seu aprendizado de CI1055 61
5.9 Detalhes do experimento 63
5.10 Perguntas e respostas do questionário pós-estudo 66

A.1 Computational thinking definitions. 94
B.1 Data extraction template. 106
C.1 Included papers.. 108

LISTA DE ACRÔNIMOS

ACM	Association for Computing Machinery
ABP	Aprendizagem Baseada em Problemas
CEIE	Comissão Especial de Informática naEducação
CI1003	Introdução à Ciência da Computação
CI1005	Algoritmos e Estruturas de Dados (ALG 1)
CSTA	*Computer Science Teachers Association*
DInf	Departamento de Informática
DCN	Diretrizes Curriculares Nacionais
ERE	Ensino Remota Emergência
IEEE-CS	IEEE Computer Society
ISTE	International Society for Technology in Education
MEC	Ministério da Educação
MSL	Mapeamento Sistemático de Literatura
PBL	*Problem Based Learning*
SBC	Sociedade Brasileira de Computação
STEM	*Science, Technology, Engineering and Math*
UFPR	Universidade Federal do Paraná
5W1H	What, Who, Why, Where, When, How

SUMÁRIO

1 INTRODUÇÃO 12
1.1 PROBLEMA DE PESQUISA E OBJETIVOS 14
1.2 CONDUÇÃO DA PESQUISA 14
1.3 ORGANIZAÇÃO DA TESE 16
2 EDUCAÇÃO EM COMPUTAÇÃO & PENSAMENTO COMPUTACIONAL 17
2.1 PENSAMENTO COMPUTACIONAL 19
2.2 PENSAMENTO COMPUTACIONAL & GRADUAÇÃO EM COMPUTAÇÃO . 21
3 FUNDAMENTAÇÃO TEÓRICA 23
3.1 APRENDIZAGEM SIGNIFICATIVA 23
3.2 APRENDIZAGEM BASEADA EM PROBLEMA 24
3.3 APRENDIZAGEM BASEADA EM DESAFIOS 24
3.4 APRENDIZAGEM COLABORATIVA 26
3.5 COMPUTAÇÃO DESPLUGADA 26
3.6 CONSIDERAÇÕES FINAIS 27
4 PENSE COMPUTACIONALMENTE: UMA ABORDAGEM BASEADA EM DESAFIOS 30
4.1 PRÁTICAS DA ABORDAGEM 30
4.1.1 Catálogo de Práticas 32
4.1.2 Representação da Solução 34
4.2 MÉTODO 35
4.3 COMPUTACIONALIDADE: ARTEFATO PARA COLETA DE EVIDÊNCIAS DO EXERCÍCIO DO PENSAMENTO COMPUTACIONAL 37
4.4 UTILIZAÇÃO DA ABORDAGEM PENSE COMPUTACIONALMENTE 42
5 INVESTIGAÇÃO DA ABORDAGEM PENSE COMPUTACIONALMENTE: ESTUDOS DE CASO 43
5.1 ESTUDOS DE CASO 43
5.2 RESULTADOS E DISCUSSÕES SOBRE OS ESTUDOS DE CASO 45
5.2.1 Estudo de Caso Piloto 45
5.2.2 Desafios 2.0 53
5.2.3 Estudos de Caso Remoto 55
5.3 COLETA DE EVIDÊNCIAS DO EXERCÍCIO DO PENSAMENTO COMPUTACIONAL: EXPERIMENTO DO ARTEFATO 63
5.3.1 Resultados e Discussões sobre o Experimento 65
5.4 CONCEPÇÃO DA ABORDAGEM 68

6 CONCLUSÕES E TRABALHOS FUTUROS 70
REFERÊNCIAS 73
APÊNDICE A – AN OVERVIEW OF COMPUTATIONAL THINKING IN HIGHER EDUCATION: A SYSTEMATIC MAPPING STUDY. 87
A.1 INTRODUCTION 87
A.2 RESEARCH DESIGN 87
A.2.1 Review Protocol. 87
A.2.2 Search in Digital Libraries 88
A.2.3 Paper Selection 88
A.2.4 Data Extraction and Data Analysis 89
A.2.5 Process Overview 89
A.3 RESULTS AND DISCUSSION 91
A.3.1 Synthesis of Result 91
A.3.2 RQ01: Do the papers define the term computational thinking? What is the definition? 94
A.3.3 RQ02: What are the theoretical or methodological research bases? 95
A.3.4 RQ03: How is computational thinking being taught or developed? 97
A.3.5 RQ04: Why developing computational thinking? 99
A.3.6 RQ05: What are the benefits and challenges of developing computational thinking in higher education?. 100
A.3.7 RQ06: How is computational thinking being evaluated?. 102
A.3.8 Threats to validity 103
A.4 CONCLUSION 104
APÊNDICE B – DATA EXTRACTION TEMPLATE 106
APÊNDICE C – LIST OF INCLUDED PAPERS 108
APÊNDICE D – ABORDAGEM PENSE COMPUTACIONALMENTE . . 112

1 INTRODUÇÃO

Historicamente, a teoria por trás do Pensamento Computacional possui relação com as origens da Computação e com os pioneiros Alan Perlis, Donald Knuth e Peter Naur. Na década de 1950, Donald Knuth afirmava que os humanos não são suficientemente precisos e rápidos para realizar procedimentos mais elaborados e que poderiam se beneficiar do poder de processamento dos computadores para isso (Knuth, 1974). Assim, Knuth acreditava que os computadores eram meramente ferramentas e questionava que a ênfase era ensinar "como" usar um computador. Knuth confirmava que se uma pessoa possuísse os conhecimentos sobre a Computação ela estaria preparada para entender e colaborar com outras matérias, como química, linguística e música.

Voltando às origens da Computação, percebemos que as ideias por trás do Pensamento Computacional não são recentes. Entretanto, foi apenas em 1980 que o termo em si foi utilizado pela primeira vez por Seymour Papert (1980). Papert reconhecia no computador uma poderosa ferramenta educacional e aliada do processo de ensino e aprendizagem. A popularização do termo veio a partir da publicação do artigo da Cientista da Computação Jeannette Wing, em 2006, no qual ela definiu o Pensamento Computacional como um conjunto de habilidades baseadas nos conceitos fundamentais da Ciência da Computação que podem ser utilizados para resolver problemas, projetar sistemas e entender o comportamento humano.

Wing (2006) mostra que o Pensamento Computacional é uma habilidade fundamental para todos, não apenas para os cientistas da Computação – tratando-se de um conhecimento transversal a diferentes áreas e que se relaciona com a Computação enquanto Ciência. Além disso, Wing (2006) atribuiu ao Pensamento Computacional o mesmo grau de importância dado à alfabetização e ao desenvolvimento da lógica matemática. Ao fazer essa comparação, Wing (2006) indica a necessidade de desenvolver essa habilidade desde a infância. Tal afirmação pode ser um dos motivos por trás de um grande número de pesquisas concentrarem o seu público-alvo na educação básica. Pesquisas como Ortiz et al. (2019) e Avila et al. (2017) revelam que tanto no cenário nacional quanto no cenário internacional a preferência de contexto é na educação básica.

Ainda que sejam poucas as pesquisas que tenham como público-alvo o ensino superior e o ensino superior em Computação, Oliveira et al. (2022) expõe que o Pensamento Computacional pode apoiar a evolução de estudantes no que diz respeito à compreensão de conceitos computacionais, promovendo a segurança para criar e explorar novas soluções. Além disso, Avila et al. (2017) apresentam que estudantes de graduação também podem se beneficiar do Pensamento Computacional principalmente na área de Ciência da Computação e cursos afins. Gouws et al. (2013) asseguram que, por meio de uma boa base no Pensamento Computacional estudantes podem desenvolver modelos mentais mais fortes e, consequentemente, performarão melhor as atividades relacionadas a programação e a Ciência da Computação.

Com raízes derivadas do pensamento natural, o Pensamento Computacional envolve ideias como abstração, decomposição, reconhecimento de padrões e algoritmos. Pensar computacionalmente envolve uma variedade de ferramentas mentais que refletem a amplitude do campo da Ciência da Computação, preocupando-se desde a análise de problemas até a formulação de soluções, considerando como as soluções serão representadas para que elas sejam efetivamente executadas por um agente de processamento de informação (Wing, 2011; Nardelli, 2019).

Sendo uma habilidade humana, o Pensamento Computacional pode ser estimulado e corresponde a um conjunto de competências para solução de problemas que todas as pessoas deveriam exercitar (Curzon et al., 2019). Pensar computacionalmente ajuda no ensino e aprendizagem das mais diversas áreas e possibilita uma aprendizagem mais ampla, favorecendo

o desenvolvimento dos saberes necessários para apoiar o ensino e aprendizado dos estudantes. Quando alinhado a uma metodologia focada em soluções de problemas reais, é possível por meio desse meio, menos abstrato, estimular a construção do conhecimento e o engajamento de estudantes. Segundo Ortiz et al. (2018), é importante considerar o contexto dos estudantes para tornar a aprendizagem mais significativa, proporcionando que estudantes trabalhem em atividades que lhes são mais atrativas e, consequentemente, alcançar um maior aproveitamento do conteúdo e melhorar o comprometimento deles com o processo.

Giraffa e Müller (2017) adicionam que para trazer estudantes como principais responsáveis pelo aprendizado eles devem estar envolvidos no processo de construção do seu conhecimento, e que a metodologia utilizada possui impactos significativos em estudantes, de modo que as atividades devem ser diversificadas para permitir a aplicação dos conceitos trabalhados. A visão de Giraffa e Müller (2017) torna-se útil para contornar as deficiências de formação básica que estudantes ingressantes apresentam, relacionadas desde a escrita e interpretação de textos, hábitos de estudo até aos aspectos de conteúdo. As lacunas de formação apresentadas pelos estudantes ao ingressarem no ensino superior resultam em dificuldades, principalmente em disciplinas que necessitam de muita abstração e requerem competências no que diz respeito à capacidade de resolução de problemas (Lisbôa e Karling, 2019).

As dificuldades enfrentadas, resultante das lacunas de formação, tornam-se um dos grandes desafios nas instituições de ensino superior e um divisor de águas nos anos iniciais dos cursos. Para Lisbôa e Karling (2019), uma forma de remediar essa situação seria promovendo a aprendizagem significativa utilizando estratégias metodológicas mais atrativas, como atividades lúdicas e frente a um desafio proposto. Segundo Carvalho et al. (2019), as deficiências de formação ficam claras quando estudantes se deparam com disciplinas de cálculo e programação. Essa situação mostra a gravidade do problema, pois indica falhas durante o processo de formação e que resultam em custos financeiros e pessoais a todas as envolvidas. Além disso, Carvalho et al. (2019) apresentam a necessidade de políticas que apoiem estudantes ao longo do curso, tanto para ajudá-los na jornada acadêmica quanto em situações de vulnerabilidade social e socioeconômica.

Caeli e Yadav (2020) afirmam que para que estudantes compreendam as ideias e práticas dos conceitos de Computação, é recomendado começar com abordagens desplugadas. As atividades desplugadas estimulam o pensamento lógico e ajudam a desenvolver o Pensamento Computacional e as habilidades de resolução de problemas (Curzon et al., 2019; Bell et al., 2012). Portanto, atividades desplugadas podem ajudar a reduzir a alta carga cognitiva enfrentada pelos estudantes do primeiro ano. Por definição, desplugada significa usar algo de forma desconectada e analógica, e está associada a atividades que utilizam os conceitos fundamentais da Computação sem o uso de tecnologias computacionais (Bell et al., 2012). A aplicação de atividades com conteúdo computacional sem a utilização do computador para resolvê-las garante que estudantes separem a instrumentação da solução.

O Pensamento Computacional é reconhecido pelas diretrizes que regulamentam a graduação em Computação como uma habilidade fundamental que deve ser exercitada por todos os estudantes em formação na área (ACM e IEEE, 2014; Zorzo et al., 2017). As instituições argumentam que a Computação está se tornando uma das principais disciplinas da educação do Século XXI, reconhecendo que todas as pessoas em formação devem possuir algum nível de compreensão sobre essa área de conhecimento. As diretrizes não deixam claro quando e como essas habilidades devem ser exercitadas, porém, Barcelos e Silveira (2012) explicam que o Pensamento Computacional deve ser desenvolvido nos primeiros semestres dos cursos de ensino superior, pois há toda uma geração de estudantes que não tiveram a oportunidade de desenvolver explicitamente esse tipo de habilidade durante o ensino básico.

Mesmo que a literatura reconheça a importância e relevância do Pensamento Computacional, ainda existem poucas evidências em torno de como apoiar o exercício das habilidades do Pensamento Computacional no ensino superior. Segundo Angeli e Giannakos (2020), para criar atividades que foquem no exercício do Pensamento Computacional é necessário preocupações em como: (i) projetar essas atividades; (ii) ensinar; (iii) avaliar; e (iv) selecionar e utilizar tecnologias como ferramenta nessas atividades. Assim, esta tese partiu do pressuposto que o exercício do Pensamento Computacional deve ser realizado explicitamente (com atividades que tenham foco de exercitar essa habilidade), sem a utilização de uma linguagem de programação (maneira desplugada) e que sejam contextualizadas na realidade dos estudantes e das comunidades acadêmicas em que as atividades serão realizadas.

1.1 PROBLEMA DE PESQUISA E OBJETIVOS

Diante do exposto, esta tese busca lançar luzes sobre como apoiar o exercício do Pensamento Computacional no ensino superior em Computação contribuindo para as lacunas identificadas na literatura: (i) a necessidade de estudantes dos cursos de graduação em Computação exercitarem o Pensamento Computacional; (ii) o número limitado de pesquisas que investigam o exercício do Pensamento Computacional para além da programação. Assim, esta tese possui como problema de pesquisa *a falta de conhecimento sobre como apoiar o exercício do Pensamento Computacional no ensino superior em Computação*.

O objetivo geral desta pesquisa é *investigar uma abordagem para apoiar a concepção e condução de práticas educacionais*[1] *que promovam o exercício do Pensamento Computacional no ensino superior em Computação*. Desse modo, o objetivo geral é dividido nos seguintes objetivos específicos (OE):

OE1) Caracterizar as iniciativas da literatura que focam no exercício do Pensamento Computacional no ensino superior;

OE2) Propor uma abordagem para apoiar o exercício do Pensamento Computacional no ensino superior em Computação;

OE03) Investigar as potencialidades e fragilidades da abordagem com o público-alvo em diferentes contextos;

OE4) Investigar as potencialidades e fragilidades do uso de atividades desplugadas para exercitar o Pensamento Computacional; e

OE5) Investigar as potencialidades e fragilidades do uso de linguagem natural e fluxogramas como forma de representar soluções.

1.2 CONDUÇÃO DA PESQUISA

Esta pesquisa foi conduzida em diferentes etapas, e utilizando diferentes métodos, e a condução foi baseada tanto na investigação conceitual quanto na construtiva. Na investigação conceitual foi realizado um mapeamento da literatura para obter um panorama das iniciativas que promoviam o exercício do Pensamento Computacional no ensino superior e identificar potenciais lacunas de

[1] Um prática educacional é um conjunto de ações e estratégias realizadas por docentes para promover o aprendizado de estudantes em um determinado contexto educacional. Nesta pesquisa, as práticas educacionais são baseadas em desafios e serão consideradas sinônimos. Utilizaremos práticas educacionais quando formos nos referir de maneira mais genérica e desafios quando formos falar de maneira mais específica as atividades criadas.

pesquisa. Para isso, consideramos quatro bibliotecas digitais: ACM, IEEE, Scopus e Comissão Especial de Informática na Educação (CEIE). As bibliotecas digitais foram escolhidas pela relevância na área de pesquisa, por abordarem os cenários brasileiro e internacional, e por incluir artigos científicos revisados por pares escritos em inglês e português. A *string* utilizada foi o termo *Pensamento Computacional* e sua tradução *computational thinking*, decidimos usar apenas o termo devido à limitação na definição de sinônimos para "ensino superior".

A busca nas quatro bibliotecas digitais foi realizada em abril de 2019 e retornou 2.966 artigos, publicados entre 1987 e 2019. Não definimos uma janela de busca porque queríamos mapear todos os artigos com o termo Pensamento Computacional e selecionar manualmente quais trabalhos apresentam uma iniciativa para exercitar o Pensamento Computacional no ensino superior. O processo de filtragem ocorreu em duas etapas: 1º filtro e 2º filtro. No 1º filtro, foram aplicados os critérios de inclusão com base na leitura de títulos e resumos de artigos. Esse processo resultou na seleção de 228 trabalhos para a próxima etapa. O 2º filtro foi um processo de leitura de todos os artigos e extração de dados dos artigos selecionados. No final, selecionamos 94 artigos relevantes para nosso mapeamento sistemático.

Os resultados do mapeamento sistemático revelam que mesmo que esforços estejam sendo direcionados para explorar as potencialidades do Pensamento Computacional no ensino superior ainda existem lacunas de pesquisa. De acordo com o mapeamento, o Pensamento Computacional ainda está muito associado aos fundamentos de programação e pouco associado com a área como um todo. Portanto, identificamos que o Pensamento Computacional pode ser implementado em contextos amplos ao invés de focar apenas em soluções computacionais. Além disso, com o mapeamento sistemático identificamos lacunas para futuras pesquisas, por exemplo: (i) investigação e formalização de métodos de avaliações; e (ii) refinamento das relações entre o Pensamento Computacional, suas habilidades e o conteúdo. O mapeamento sistemático da literatura pode ser visto na íntegra no Apêndice A.

Fez parte da investigação construtiva a concepção de um conjunto de práticas educacionais para apoiar o exercício do Pensamento Computacional no ensino superior em Computação e da formatação de uma abordagem. Seis estudos de caso foram realizados para investigar as potencialidades e fragilidades da abordagem no contexto de uma disciplina introdutória dos cursos de Bacharelado em Ciência da Computação e Informática Biomédica da Universidade Federal do Paraná (UFPR) que tem como objetivo promover habilidades de entendimento e resolução de problemas exercitando habilidades de Pensamento Computacional. Foi a partir da experimentação em sala que observamos que existe uma relação entre a forma como as atividades são conduzidas e as habilidades exercitadas. Com base nessa observação, criamos um método para guiar a resolução das práticas em sala de aula. Da união das práticas educacionais com o método de condução em sala identificamos que estávamos criando uma abordagem para apoiar os professores e pesquisadores a criarem e conduzirem práticas educacionais que promovam o exercício do Pensamento Computacional. A abordagem Pense Computacionalmente é resultado das observações realizadas em sala, dos questionários de feedback que os estudantes responderam, da interação e entrevista realizada com docentes da disciplina, somado com a experiência acadêmica profissional da autora da tese. Detalhes da abordagem desenvolvida podem ser encontrados no Capítulo 4.

Os estudos de caso, além de fornecerem evidências das potencialidades e fragilidades das práticas educacionais conduzidas, apoiaram na evolução de todo o material produzido e em como a pesquisa foi construída. Nos estudos de caso, os métodos para a coleta de dados foram: a observação participante, registro em arquivos, entrevistas semiestruturadas, e questionário online e anônimo. Além disso, dados foram coletados a partir da instância da disciplina no Moodle. A análise de dados foi qualitativa, e o método de análise dos dados utilizados foi a Análise Temática

de Clarke et al. (2015). A descrição dos estudos de caso realizados, os resultados obtidos e detalhes da concepção da abordagem podem ser vistos no Capítulo 5.

Como contribuições desta tese podemos citar: (i) contribuições conceituais: como apoiar o exercício do Pensamento Computacional no ensino superior em Computação; relação dos passos realizados durante a resolução de um desafios com as habilidades do Pensamento Computacional; e a caracterização de como coletar evidências do exercício do Pensamento Computacional; (ii) contribuição metodológica: como conduzir atividades que foquem no exercício do Pensamento Computacional no ensino superior em Computação; e (iii) contribuição técnica: formalização da abordagem em material didático.

1.3 ORGANIZAÇÃO DA TESE

Depois desta Introdução, o Capítulo 2 apresenta a origem da Educação em Computação, define o Pensamento Computacional e explica a relação do Pensamento Computacional e a graduação em Computação. No Capítulo 3 encontra-se a fundamentação teórica da tese. O Capítulo 4 descreve a abordagem Pense Computacionalmente proposta para apoiar a concepção e condução de práticas que promovam o exercício do Pensamento Computacional no ensino superior em Computação. O Capítulo 5 relata os estudos de caso realizados para experimentação da abordagem Pense Computacionalmente, e o estudo de caso realizado para utilização e avaliação do artefato criado para apoiar a coleta de evidências do exercício do Pensamento Computacional. A conclusão e os trabalhos futuros são apresentados no Capítulo 6. Por fim, o Apêndice A relata o mapeamento sistemático da literatura, o Apêndice B apresenta o modelo de extração de dados do mapeamento sistemático, o Apêndice C lista os 94 artigos incluídos no mapeamento, o Apêndice D traz uma versão da abordagem Pense Computacionalmente.

2 EDUCAÇÃO EM COMPUTAÇÃO & PENSAMENTO COMPUTACIONAL

Em meados de 1950, o conteúdo de uma nova disciplina universitária chamada Ciência da Computação estava sendo discutida nos círculos universitários dos Estados Unidos. O rápido crescimento da área mostrou a necessidade de modificações educacionais nas universidades, o que levou a *Association for Computing Machinery* (ACM) instituir um comitê de currículo para criar as recomendações preliminares para um programa de graduação em Ciência da Computação (Conte et al., 1965). As recomendações foram criadas para oferecer orientação aos cursos sobre quais disciplinas deveriam ser incluídas, quais eram os conteúdos de apoio, e quais eram as necessidades dos estudantes. Na época, o currículo base da graduação era composto por cursos teóricos, algoritmos numéricos e modelos computacionais e suas aplicações.

No mesmo período, o comitê do *Colege of Engineering* da Universidade de Michigan se reuniu para discutir sobre os principais tópicos sobre o uso dos computadores na educação em Engenharia. Além disso, o comitê questionava como, quando e onde a programação deveria ser inserida nos currículos de graduação (Katz, 1960). Durante um encontro do comitê, Alan Perlis sugeriu que estudantes deveriam ser ensinados a compreender os computadores como ferramentas gerais para a solução de problemas, em vez de ferramentas específicas para resolver problemas específicos (Katz, 1960). Enquanto participantes do comitê unanimemente afirmavam que projetar soluções algorítmicas para resolver um problema envolve processos de pensamento básicos, Perlis reforçava que todas as pessoas deveriam aprender e desenvolver essas habilidades.

As discussões em torno do entendimento do que era ou não um algoritmo e sobre o que envolvia as soluções algorítmicas começaram a ganhar espaço com o avanço das discussões em volta dos currículos de graduação dos cursos de Ciência da Computação. Em 1962, Peter Elias sugeria que todas as pessoas deveriam entender sobre algoritmos, mas que não precisavam necessariamente envolver um computador, e que nem precisavam entender em detalhes como um computador funcionava (Greenberger, 1962). O computador era reconhecido por Peter Naur como um último ponto e não como o elemento mais importante da Ciência da Computação. Em um movimento para retirar o computador como foco da Ciência da Computação e apresentá-la como Ciência dos Dados e processamento dos dados, Naur começou a utilizar o termo datalogy – termo Dinamarquês para Ciência da Computação.

Peter Naur ainda afirmava que os conceitos básicos da Ciência da Computação são de natureza geral e reúnem um grande número de atividades e conceitos humanos, que de muitas maneiras inspiram ideias sobre as tarefas cotidianas e em diferentes áreas (Sveinsdottir e Frøkjær, 1988). Ele também afirmava que os conceitos fundamentais deveriam ser ilustrados por meio de exemplos concretos e que podem ser retirados do contexto em que os estudantes estão familiarizados. Naur ainda discutia ferramentas de software a partir de uma perspectiva em que pessoas, ferramentas e problemas se relacionam no todo. Para ele, as ferramentas disponíveis impactam diretamente na compreensão e a formulação da solução (Figura 2.1). Uma segunda perspectiva de Naur é a inclusão das linguagens de programação como ferramentas para resolução de problemas em diferentes áreas (Figura 2.2), o perigo desta visão é que elas tenderão a manter uma perspectiva do problema que é limitada a uma ferramenta – que pode se tornar em pouco tempo obsoleta. Assim, as características das ferramentas moldam o pensamento das pessoas e suas concepções de problema (Sveinsdottir e Frøkjær, 1988).

Embora a programação seja considerada uma ferramenta importante para resolução de problemas, principalmente na Ciência da Computação, alguns pesquisadores afirmam que a parte mais importante é a habilidade de projetar e expressar a solução como um algoritmo.

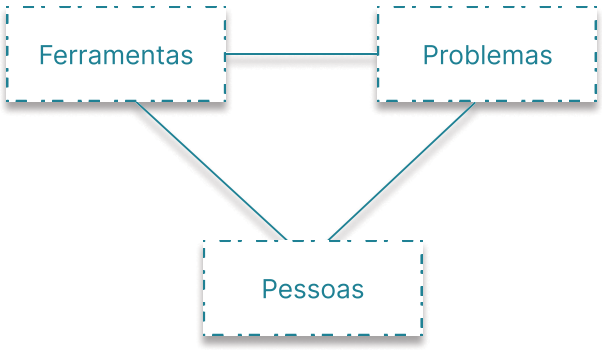


Figura 2.1: Componentes básicos de resolução de problemas de acordo com Naur – tradução Sveinsdottir e Frøkjær (1988)

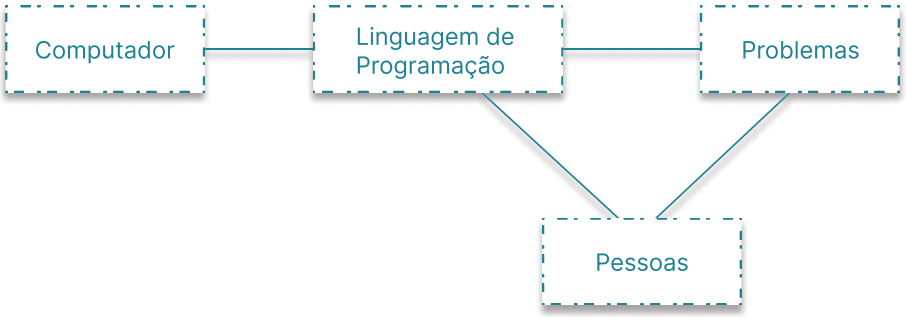


Figura 2.2: Componentes fundamentais na resolução de problemas usando computadores de acordo com Naur – tradução Sveinsdottir e Frøkjær (1988)

Nesta mesma linha, Alan Perlis afirmou que projetar algoritmos para resolver problemas envolvia processos de pensamento natural (Katz, 1960). Ainda que a tarefa mais significativa para a resolução de problemas seja a de elaborar e preparar o algoritmo, segundo Sveinsdottir e Frøkjær (1988) o computador apenas fornece poder de processamento para tarefas que levariam muito tempo para serem concluídas por um humano.

Ao reconhecerem o computador como uma ferramenta importante e poderosa começaram os questionamentos de como incluir o computador nas escolas, as possibilidades do uso do computador, e como ensinar a utilizá-lo (Pea e Kurland, 1984). Em 1980, Papert explorou a relação do computador com a inovação do sistema educacional e como ele poderia ser utilizado para estimular o pensamento e desenvolver o conhecimento. Dessa forma, o computador tornava-se um aliado na construção do conhecimento e influenciador do pensamento humano. Com influências de Jean Piaget, Papert cria a filosofia LOGO, que inverte o atual papel do computador nas escolas. Papert afirma que o aprendizado acontece por meio do processo da criança "programar" o computador, ao invés do computador ensinar a criança. Neste ponto, o computador assume o papel de ferramenta (máquina de ensinar) para suportar o conhecimento e de apoio à criança na construção do seu próprio pensamento.

A partir de então, a relação do computador com a inovação do sistema educacional começou a ser explorada. Hsu et al. (2018) revela que a ferramenta de ensino mais utilizada para ensinar Computação para crianças e jovens é a linguagem de programação, embora existissem alternativas e até mesmo alternativas desplugadas. Geralmente, a Computação é ensinada por meio da interação com uma das atividades mais exercidas pelos profissionais da área: programando. Entretanto, mesmo que seja esperado que o aprendizado ocorra por meio da prática de programação é uma maneira ineficaz de ensinar — ainda mais para iniciantes (Guzdial, 2015). Guzdial (2015) ainda revela que estudantes precisam da oportunidade de adquirir muito

conhecimento antes de programar, e mesmo que ensinar Computação a partir da perspectiva de uma linguagem de programação seja uma abordagem válida, ela não desenvolve a compreensão e habilidades de resolução de problemas.

Discussões sobre como a Computação deve ser ensinada acontecem desde a década de 1960, com alguns defendendo que todos deveriam aprender a programar e outros defendendo que o foco deveria ser os conceitos. Naur afirmava que a computação deveria ser ensinada como parte da educação geral, mas ele duvidava que esse objetivo seria alcançado nos próximos anos. Em 1966, Naur previa que seriam necessárias décadas para que mudanças fundamentais ocorressem no sistema educacional (Sveinsdottir e Frøkjær, 1988). Meio século se passou desde a afirmação de Naur, muitas mudanças ocorreram tanto na Computação quanto na forma como a Computação é ensinada, entretanto, muito ainda precisa ser feito. Ao longo dessas décadas, sociedades profissionais e científicas da Computação discutiram amplamente a educação em Computação e formularam diretrizes curriculares para regulamentar a formação em Computação. A ACM, IEEE-CS, Sociedade Brasileira de Computação (SBC) e o Ministério da Educação (MEC) são exemplos de organizações que mantêm pesquisas para definir e evoluir as diretrizes da educação em Computação, considerando o que é comum da área e o que é particular do local.

2.1 PENSAMENTO COMPUTACIONAL

O termo e a teoria que embasa o Pensamento Computacional possuem relação com os primórdios da Computação, e a definição mais moderna sucede a definição de Seymour Papert em 1980. Papert (1980) fez referência ao termo Pensamento Computacional no seu livro intitulado *"Mindstorms: Children, Computers, and Powerful Ideas"* e mostrava a relevância dele para vários aspectos da vida cotidiana, explorando a relação do computador com a inovação do sistema educacional e como ele poderia ser utilizado para estimular o pensamento e desenvolver o conhecimento. Dessa forma, o computador tornava-se um aliado na construção do conhecimento e influenciador do pensamento humano. Para Papert (1980) o Pensamento Computacional é fruto da sua abordagem construcionista da educação, em que as dimensões afetivas e sociais possuem a mesma importância quanto o conteúdo teórico e técnico.

Décadas depois, o Pensamento Computacional foi reapresentado por Wing (2006) como um conjunto de habilidades universalmente aplicáveis baseadas nos conceitos fundamentais da Computação que podem ser utilizadas para resolver problemas, projetar sistemas e entender o comportamento humano. Wing atribuiu a essa competência o mesmo grau de importância dado à alfabetização e ao desenvolvimento da lógica matemática. Ao reapresentar o termo, Wing (2006) deixou muitos tópicos em aberto, dando margem para diferentes interpretações sobre o que era e o que não era Pensamento Computacional.

Pensamento Computacional é definido como um processo de pensamento envolvido na formulação de problemas e suas soluções, preocupado em como as soluções são representadas para serem efetivamente executadas por um agente de processamento de informações (Wing, 2011; Nardelli, 2019). Considerado uma habilidade humana, o Pensamento Computacional é um conjunto de habilidades mentais e cognitivas desenvolvidas pelo estudo e a prática da Computação. Pensar computacionalmente é uma habilidade que deve ser exercitada por todos, indiferente da área, e pode ser aplicada nas mais diversas áreas de interesse humano (Gouws et al., 2013). O Pensamento Computacional está associado à maneira de pensar que é típica da Computação, porém, pode ajudar no ensino e aprendizagem nas mais diferentes áreas e inclui uma variedade de ferramentas mentais que refletem uma amplitude da Computação.

Pensar computacionalmente é visualizar um mesmo problema em diferentes níveis, e encontrar estratégias para reformulá-lo em partes menores e mais fáceis de serem resolvidas –

refletindo na solução de maneira lógica, algorítmica, abstrata, recursiva e paralela. Essa abordagem inclui explorar diferentes aspectos do problema, considerando a sua complexidade e projetando soluções com os recursos disponíveis (Wing, 2008). Ainda, pensar computacionalmente é ser capaz de pensar maneiras diferentes de resolver um problema por meio de uma perspectiva computacional, enfatizando a ideia do que é computável e como a Computação funciona.

O Pensamento Computacional está enraizado em abordagens humanas não digitais para a resolução de problemas considerando os conceitos computacionais, e fornece métodos para correlacionar esses conceitos com tecnologias para criar uma solução para o problema identificado (Caeli e Yadav, 2020). Sendo um processo fundamentalmente cognitivo, o Pensamento Computacional envolve o raciocínio lógico e abstrato. Para Agbo et al. (2019), o Pensamento Computacional é uma abordagem fundamental para entender, abstrair e modelar problemas com a intenção de produzir soluções, e envolve mais do que fornecer instruções para o computador por meio de uma linguagem de programação.

Mesmo que a palavra "computacional" remeta ao computador e permita uma analogia com "programação", eles não podem ser compreendidos como sinônimos. O Pensamento Computacional favorece o entendimento sobre os sistemas computacionais, como eles funcionam e como eles são programados (Barr e Stephenson, 2011). Desse modo, o Pensamento Computacional expressa mais do que apenas codificar e ser capaz de programar um computador. De acordo com Xu e Tu (2011), o pensar computacionalmente se assemelha ao pensamento lógico e matemático, sendo uma espécie de pensamento analítico sob as lentes computacionais.

A essência do Pensamento Computacional está relacionada às habilidades inerentes à resolução de problemas, que incluem quatro pilares: (i) abstração, (ii) decomposição, (iii) reconhecimento de padrões, e (iv) algoritmos. A abstração é o pilar responsável por focar nas partes do problema que são relevantes, enquanto a decomposição identifica quais partes do problema podem ser divididas em partes menores, mais fáceis de resolver e de gerenciar. O reconhecimento de padrões reconhece as similaridades nos problemas ou soluções para que seja possível replicar ou aperfeiçoar as soluções. O pilar de algoritmos, compreende a formalização da solução, por meio de instruções claras e objetivas.

Wing (2008) reconhece que a abstração é considerada o cerne do Pensamento Computacional, sendo uma habilidade de alto nível e o processo de pensamento mais importante. Consequentemente, é o processo de abstração que estabelece quais detalhes são importantes e quais podem ser ignorados, e permite identificar qual será a melhor ferramenta a ser utilizada dentre as disponíveis para resolver um problema específico. Barr e Stephenson (2011) apresentam que coleta de dados, análise de dados, representação de dados, decomposição de problemas, abstração, algoritmos e procedimentos, automação, paralelização, e simulação são habilidades do Pensamento Computacional. Além disso, as habilidades incluem: projetar soluções, testar, depurar, modelar, simular, comunicar (saber utilizar o vocabulário) e explorar.

CSTA e ISTE (2011) adicionam que também são habilidades do Pensamento Computacional: (i) entender como expressar as soluções de maneira que ferramentas auxiliam na resolução; (ii) organizar, interpretar e representar os dados; (iii) automatizar as soluções; (iv) identificar, analisar e implementar as soluções buscando a eficiência e a eficácia delas; e (v) generalizar o processo de resolução problemas para todas as áreas de interesse humano. Para eles, essas habilidades são suportadas por atitudes que vão desde a confiança ao lidar com a complexidade, passando pela persistência, tolerância, até a capacidade de comunicar e trabalhar com outras pessoas em busca de um objetivo em comum.

Os pilares do Pensamento Computacional podem ser vistos como habilidades de alto nível que se relacionam com um grupo de habilidades derivadas. Palts e Pedaste (2020) revelam que não existe um entendimento comum sobre o Pensamento Computacional e suas dimensões.

A partir de uma revisão sistemática, os autores produziram um agrupamento das dimensões e das habilidades do Pensamento Computacional. Assim, considerando as dimensões e habilidades produzidas por Palts e Pedaste (2020), as diretrizes curriculares dos cursos de graduação em Computação (Zorzo et al., 2017; ACM e IEEE, 2014), as habilidades encontradas no mapeamento sistemático da literatura conduzido pela autora Oliveira et al. (2022), selecionamos no conjunto das seguintes habilidades (Tabela 2.1).

Tabela 2.1: Habilidades do Pensamento Computacional Selecionadas

Habilidades	**Descrição**	**Ref**
Abstração	Interpretar e entender o problema e a atividade, identificando quais aspectos são relevantes e quais podem ser ignorados.	Palts e Pedaste (2020) ACM e IEEE (2014) Wing (2006) Pereira et al. (2021)
Decomposição	Identificar quais partes da atividade podem ser divididas em partes menores, mais fáceis de resolver e de gerenciar.	Wing (2006) Palts e Pedaste (2020) Pereira et al. (2021)
Reconhecimento de Padrões	Identificar os padrões da atividade, dos dados e da solução, reconhecer as similaridades entre as atividades e soluções para que seja possível reaproveitar ou aperfeiçoar as soluções existentes.	Zorzo et al. (2017) Pereira et al. (2021)
Coleta e Análise dos Dados	Identificar como os dados estão ordenados e organizados, e de fazer sentido dos dados coletados.	Palts e Pedaste (2020)
Algoritmo	Definir uma sequência ordenada de instruções finitas para a solução de um problema.	Palts e Pedaste (2020) Zorzo et al. (2017) Pereira et al. (2021)
Generalização	Propor uma solução que possa ser utilizada para diferentes quantidades de dados.	Palts e Pedaste (2020)
Representação da Solução	Organizar as informações e instruções da solução considerando com quem ou o que irá interpretar ou executar a solução.	Palts e Pedaste (2020) Pereira et al. (2021)
Simulação e Teste	Simular e testar a solução para a entrada de dados com o objetivo de identificar falhas e espaço para melhorias.	Palts e Pedaste (2020) Wing (2006)
Otimização	Corrigir, refinar e otimizar a solução existente.	Zorzo et al. (2017) Pereira et al. (2021)

2.2 PENSAMENTO COMPUTACIONAL & GRADUAÇÃO EM COMPUTAÇÃO

No Brasil, em 2016, o Ministério da Educação (MEC) instituiu as Diretrizes Curriculares Nacionais (DCN16) para os cursos de graduação em computação (MEC, 2016). Este documento determina como os cursos de graduação em computação devem conduzir a formação dos profissionais, e determina deve formar profissionais com conhecimento das questões sociais, entendimento do impacto das tecnologias na sociedade, e visão crítica e criativa para a resolução de problemas. Além disso, as diretrizes reconhecem a importância do Pensamento Computacional no cotidiano de estudo e trabalho na área e a sua aplicação em outros domínios.

A partir da DCN16, a Sociedade Brasileira de Computação (SBC) criou um documento com referenciais para os cursos de graduação em Computação (Zorzo et al., 2017). As competências e habilidades apresentadas pelas DCN16 foram agrupadas em eixos de formação, e cada eixo de formação corresponde a uma macro competência e relaciona-se com um grupo de competências derivadas. O primeiro eixo de formação apresentado é o de resolução de

problemas, no qual estudantes devem ser capazes de criar soluções para problemas complexos. Esse eixo possui uma competência específica (C.1.5) que reconhece a importância do Pensamento Computacional e sua aplicação em circunstâncias apropriadas e em domínios diversos.

No cenário internacional, a ACM e a IEEE Computer Society juntaram forças para criar as diretrizes curriculares para a graduação em Ciência da Computação (ACM e IEEE, 2014). As instituições argumentam que a Computação está se tornando uma das principais disciplinas da educação do Século XXI, reconhecendo que todos os indivíduos devem possuir algum nível de conhecimento sobre. Assumem também que, em um futuro próximo, todas pessoas em formação tenham conhecimentos sobre computação e que o Pensamento Computacional é uma das habilidades fundamentais desejadas para todos os graduandos.

Ainda existem poucas evidências em torno de como apoiar o exercício das habilidades do Pensamento Computacional no ensino superior. Pesquisas como Zanetti et al. (2016), Ortiz e Pereira (2018) e Silva et al. (2022) revelam que existe interesse em identificar como apoiar a inserção desses conteúdos na educação, entretanto, essas mesmas pesquisas ilustram que o foco ainda não é o ensino superior. Zanetti et al. (2016) realizaram um panorama sobre Pensamento Computacional no Brasil entre 2012 e 2015, e revelam que os estudos possuem foco para o ensino médio e ensino fundamental. O mesmo panorama é encontrado pela pesquisa de Ortiz e Pereira (2018), no qual apenas 8 trabalhos (17,4%) tinham como foco o exercício do Pensamento Computacional no ensino superior. Silva et al. (2022) afirmam que apesar de existir uma tendência crescente, poucos estudos focam no Pensamento Computacional no ensino superior, e indicam a necessidade de mais pesquisas neste contexto. A necessidade ainda é maior nos cursos de computação, uma vez que as diretrizes de formação reconhecem o Pensamento Computacional como uma habilidade necessária a ser desenvolvida pelas pessoas em formação.

Ainda que exista muitos espaços para pesquisas sobre Pensamento Computacional no ensino superior, pesquisas dão luzes do que devem ser considerado para promover o exercício dessas habilidades nesses contextos. Barcelos e Silveira (2012) explicam que o Pensamento Computacional deve ser desenvolvido nos primeiros semestres dos cursos de ensino superior, pois há toda uma geração de estudantes que não tiveram a oportunidade de desenvolver explicitamente esse tipo de habilidade durante o ensino básico. Para que estudantes captem e coloquem em prática conceitos computacionais, é necessário começar com uma abordagem desplugada pois elas estimulam o pensamento lógico e ajudam a desenvolver o Pensamento Computacional e as habilidades de resolução de problemas (Caeli e Yadav, 2020; Curzon et al., 2019).

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica da tese, composto por: Aprendizagem Significativa (3.1), Aprendizagem Baseada em Problemas (3.2), Aprendizagem Baseada em Desafios (3.3), Aprendizagem Colaborativa (3.4), Computação Desplugada (3.5).

3.1 APRENDIZAGEM SIGNIFICATIVA

A aprendizagem significativa é uma teoria da aprendizagem que considera o conhecimento prévio da pessoa como base para a construção de novos conhecimentos. Definida por Ausubel (1963), a Aprendizagem Significativa é um processo no qual novas informações interagem com a estrutura do conhecimento da pessoa. Para Ausubel o cérebro humano armazena informações de maneira organizada, hierárquica, e que conhecimentos mais específicos são ligados a conhecimentos mais gerais. Assim, Ausubel afirma que a Aprendizagem Significativa ocorre quando uma nova informação se ancora em conceitos relevantes na estrutura cognitiva de quem aprende.

Ausubel propôs a Aprendizagem Significativa a partir do ponto de vista cognitivista e define a aprendizagem como organização e interação do material na estrutura cognitiva, de maneira que novas ideias e informações podem ser aprendidas na medida que conceitos relevantes e inclusivos estejam adequadamente claros na estrutura cognitiva da pessoa (Moreira e Masini, 1982). A aprendizagem significativa é um processo por meio do qual uma nova informação relaciona-se, de maneira substantiva (não-literal) e não-arbitrária, a um aspecto relevante da estrutura de conhecimento do indivíduo. O conhecimento prévio ajuda a significar o conteúdo que será aprendido, esse processo é chamado de subsunçor ou ideia-âncora (Ausubel, 1963).

Em outras palavras, na Aprendizagem Significativa a aquisição de novos conhecimentos acontece por meio da ancoragem, conexão, entre o que a pessoa sabe com o que será aprendido. A forma como acontece o processo de ancoragem, a relação do conhecimento prévio com o novo conhecimento, atribui um significado ao conhecimento o que resulta em uma fixação mais duradoura (Ausubel, 1963). Para Ausubel deve existir uma sincronia entre as condições para aprendizagem ocorrer, que vai desde as estratégias e adoção de materiais por parte dos docentes até a predisposição dos estudantes de aprender. Além disso, a construção de novos sentidos depende também da interação social.

Na Aprendizagem Significativa professores devem considerar o contexto e história dos estudantes para propor novos problemas que favoreçam a aprendizagem. Em outras palavras, professores são responsáveis em organizar os conhecimentos de forma a integrá-los com os anteriores e existentes, desenvolvendo um sentido mais concreto para aprendizagem e proporcionar ambientes favoráveis para que o novo conhecimento tenha onde se ancorar. Quanto maior o número de relações que a pessoa conseguir criar durante a aprendizagem com o conhecimento que ela tem, mais consistente estará o conhecimento (Ausubel, 1963).

A Aprendizagem Significativa é estabelecida quando é atribuído o significado a um conhecimento a partir da interação com os conhecimentos prévios, independente se esses significados sejam aceitos ou não no contexto da pessoa. Entretanto, o conhecimento prévio nem sempre pode ser visto como uma variável que facilita a construção de novos conhecimentos. Preconceitos e conhecimento de senso comum podem dificultar o aprendizado escolar.

3.2 APRENDIZAGEM BASEADA EM PROBLEMA

A Aprendizagem Baseada em Problemas (ABP), do inglês *Problem Based Learning* (PBL), é uma metodologia de aprendizagem que utiliza a resolução de problemas para aquisição de conhecimento (Hmelo-Silver, 2004). A metodologia traz o estudante para o centro da aprendizagem, transformando-a em responsável pela obtenção do seu conhecimento. Assim, o estudante passa a ser ativo no processo de ensino e aprendizagem e os professores tornam-se facilitadores. A ABP promove aulas mais dinâmicas e divertidas, buscando resultar em uma maior participação e desempenho dos estudantes. A construção do conhecimento ocorre de forma colaborativa, pois os estudantes trabalham em grupos e concentram-se em investigar, explicar e resolver problemas reais e que não possuem apenas uma solução correta (Torp e Sage, 1998).

O processo da Aprendizagem Baseada em Problemas começa com a apresentação do problema, paralelamente, professores auxiliam os grupos de estudantes a reconhecer o cenário do problema e representá-lo. O próximo passo é a discussão entre estudantes de cada grupo, com o objetivo de identificar as ideias, os fatos, as metas e formular hipóteses. Em seguida, o momento é reservado para individualmente cada um dos estudantes pesquisar sobre os conteúdos necessários para responder às questões levantadas no passo anterior. Após esse momento, os estudantes se reúnem novamente em grupos para aplicação dos conhecimentos. Por fim, os estudantes precisam refletir sobre o conhecimento adquirido (Hmelo-Silver, 2004).

A Aprendizagem Baseada em Problemas favorece a construção social do conhecimento, em que à medida que estudantes trabalham em pequenos grupos para propor soluções de problemas do mundo real aplicam seus conhecimentos e refletem sobre o que aprenderam, permitindo que eles construam seus novos conhecimentos. A organização em grupos permite utilizar problemas que normalmente seriam difíceis de trabalhar individualmente. A reflexão proposta no fim de cada ciclo proporciona aos estudantes a relacionar o novo conhecimento com os anteriores, e como as novas estratégias podem ser misturadas e reutilizadas para propor novas soluções. Além disso, desenvolver a autonomia e o protagonismo dos estudantes favorece a aprendizagem ao longo da vida, sendo algumas das principais vantagens dessa metodologia.

3.3 APRENDIZAGEM BASEADA EM DESAFIOS

A Aprendizagem Baseada em Desafios, do inglês *Challenge Based Learning*, é uma estrutura de aprendizagem projetada para melhorar o aprendizado e o comportamento organizacional em torno de experiências de aprendizado que incentivam estudantes a utilizarem a tecnologia para resolução de um desafio. Os desafios podem variar em tamanho e escopo, são problemas do mundo real em suas casas, escolas ou comunidades. A Aprendizagem Baseada em Desafios tem como a premissa de que precisamos de uma estrutura eficaz e eficiente para pensar profundamente durante a criação de soluções sustentáveis para os desafios que enfrentamos em nossa sociedade, essa estrutura ajuda a remediar os erros e não negligenciar ideias inovadoras. Para Aprendizagem Baseada em Desafios é crucial apoiar o desenvolvimento de uma geração de estudantes que seja engajada e que tenha conhecimentos para identificar os desafios e desenvolver soluções inovadoras e sustentáveis (Apple, 2011).

A Aprendizagem Baseada em Desafios baseia-se na aprendizagem a partir do levantamento de uma série de desafios relacionados a um tema, em que os desafios envolvem a contribuição de soluções que podem beneficiar a sociedade (Nichols e Cator, 2008). A Aprendizagem Baseada em Desafios possui princípios na aprendizagem ativa, como a Aprendizagem Baseada em Problemas (Hmelo-Silver, 2004) e Aprendizagem Baseada em Projetos de (Krajcik e Blumenfeld, 2006). A Aprendizagem Baseada em Desafios procura guiar o processo de ensino

e aprendizagem de estudantes por meio da resolução de desafios reais, em que os estudantes devem propor soluções variadas para os desafios propostos. As soluções devem ser claras e fundamentadas, além de serem próximas ao contexto dos estudantes, seja a sala de aula, a universidade ou a comunidade. Assim, é possível associar um desafio com ações que possam ser convertidas de maneira prática pelos estudantes.

Ao utilizar desafios reais, os estudantes conseguem aplicar suas soluções no ambiente proposto e visualizar o comportamento da solução. Trazendo para o contexto de sala de aula, é possível utilizar os problemas enfrentados localmente pelos estudantes e beneficiar-se das soluções propostas por eles para remediar essas situações. Esse processo, de executar as soluções, possibilita que estudantes reflitam sobre o funcionamento da solução, verifique os impactos da sua solução e maneiras de melhorá-la (Oliveira et al., 2019). A Aprendizagem Baseada em Desafios fomenta o aprendizado colaborativo, aproximando professores e estudantes para trabalharem juntos para aprender sobre questões importantes, propor soluções para os desafios reais, e agir. A estrutura da Aprendizagem Baseada em Desafios incentiva que estudantes reflitam sobre o seu aprendizado e sobre os impactos das suas soluções, além de que incentiva que as soluções sejam publicadas para uma audiência mundial (Nichols et al., 2016).

Iniciada na Apple em 2008, a Aprendizagem Baseada em Desafios fez parte de um projeto colaborativo chamado Apple Classrooms of Tomorrow-Today (ACOT2) que tinha como objetivo identificar os princípios essenciais de design do ambiente de aprendizado do Século XXI com um foco no ensino médio. ACOT2 é uma continuação do programa anterior da Apple, Apple Classrooms of Tomorrow (ACOT), que investigou e colaborou com escolas e universidades durante os anos de 1985 e 1995 e que apresenta resultados até hoje. Uma das principais inquietações por trás da criação da Aprendizagem Baseado em Desafios foi o fato de a educação ser centrada em conteúdos que atendam os requisitos e padrões, mas falta um contexto do mundo real e oportunidades para estudantes terem participação ativa. Apple (2011); Nichols et al. (2016) ainda afirmam que muitas vezes não conseguir envolver estudantes no processo de aprendizagem pode resultar em abandono do estudo e resultar em um trabalho sem inspiração.

A Aprendizagem Baseada em Desafios é colaborativa e prática, em que os estudantes precisam colaborar com outros estudantes, professores, e especialistas para desenvolver um conhecimento mais profundo dos assuntos estudados e propor soluções que possam gerar impactos. Por meio do aprendizado baseado em desafios, estudantes e professores estão fazendo a diferença e provando que o aprendizado pode ser profundo, envolvente, significativo e proposital (Apple, 2011). Professores possuem a tarefa de identificar conteúdos multidisciplinares, relacionar com o contexto dos estudantes, conectar com o mundo atual, e preparar uma experiência na qual estudantes consigam propor soluções que sejam implementadas em suas comunidades.

A Aprendizagem Baseada em Desafios começa com a identificação de uma grande ideia e se desdobra para as seguintes etapas: a questão essencial; o desafio; questões norteadoras, atividades e recursos; definir e implementar a solução; avaliação da solução; publicação da solução; e reflexão da solução. A avaliação e a reflexão são partes importantes do processo em todas as etapas, pois reforçam o aprendizado e preparam estudantes para as próximas etapas do aprendizado. A natureza interdisciplinar, cooperativa e aplicada pela Aprendizagem Baseada em Desafios enfatiza a exploração de tópicos de vários ângulos e através das lentes de várias disciplinas, o que possibilita que estudantes percebam e construam conexões naturais entre as áreas de assunto que nem sempre são evidentes.

3.4 APRENDIZAGEM COLABORATIVA

A aprendizagem colaborativa foi apresentada como uma abordagem educacional na qual estudantes trabalham juntos em pequenos grupos em busca de um objetivo comum, tendo como objetivo permitir que estudantes construam o conhecimento por meio da colaboração entre eles (Dillenbourg, 1999). O foco da aprendizagem colaborativa é a interação de forma coletiva, no qual o grupo é responsável pela aprendizagem de cada participante. A troca ativa de ideias dentro de pequenos grupos promove o pensamento crítico e impactam o interesse entre participantes. Logo, o sucesso de um estudante ajuda outros estudantes a também terem sucesso e o todo passa a ser maior que a soma das partes individuais – de modo que aprender colaborativamente pode produzir ganhos superiores a aprendizagem individual (Barros, 1994).

Utilizar a aprendizagem colaborativa possibilita os estudantes a refletirem sobre suas ideias, defenderem o seu ponto de vista e amadurecer a solução sobre a perspectiva do colega (Gokhale, 1995). Assim, a abordagem permite o desenvolvimento da comunicação, responsabilidade, exposição e compreensão. Trabalhar em grupo adiciona variáveis diferentes no processo, uma vez que essa prática reuni pessoas com diversidades cognitivas (e.g., experiência, conhecimento, etc...) em prol do mesmo objetivo. Sabe-se também, que pessoas diferentes possuem abordagens diferentes de ver o problema e solucioná-lo. Portanto, podemos considerar que essas soluções tendem a ser mais amplas, inteligentes, diversas e inclusivas.

3.5 COMPUTAÇÃO DESPLUGADA

A Computação desplugada é uma abordagem baseada no construtivismo para ensinar os conceitos gerais da Computação de maneira desconectada (Curzon et al., 2019; Bell et al., 2012). Construtivismo é uma tese epistemológica de Piaget (2002) que busca explicar como o entendimento da perpetuação do conhecimento (possibilidades e limitações), como ele surge e se desenvolve. O Construtivismo preocupa-se em como uma pessoa passa de um conhecimento simples para um conhecimento mais complexo. As atividades desplugadas buscam comunicar os conteúdos sem a distração dos computadores, para isso, são utilizadas diferentes técnicas, como: dramatização, quebra-cabeças, jogos, desafios, entre outros. Essas atividades possuem o objetivo de comunicar criativamente as ideias computacionais para quem tiver interesse em ouvir.

Estudantes quando participam de uma atividade desplugada não precisam se preocupar com a sintaxe de uma linguagem de programação ou com a interface de desenvolvimento. Todos os esforços são destinados à resolução do problema proposto, permitindo uma maior interação entre estudantes e expondo-os a objetos reais. Essa interação com objetos do mundo real, possibilita ao estudante construir o seu próprio conhecimento por intermédio de alguma ferramenta, é o princípio central do Construcionismo de Papert (Papert, 1980). A Computação Desplugada proporciona um ambiente de exploração que permite o uso da criatividade para criar soluções e ajuda a compreender as teorias computacionais. Logo, é considerado um método de ensino atraente e que traz benefícios para os estudantes.

Na literatura localizamos diferentes trabalhos que apresentam os benefícios de utilizar uma abordagem desplugada, que variam desde o ensino infantil até o ensino profissional (Oliveira et al., 2019; Brackmann, 2017; Bell et al., 2012; Bulhões et al., 2019). Segundo pesquisas de Curzon et al. (2019) e Bell et al. (2012), às atividades desplugadas estimulam o raciocínio e ajudam a desenvolver o Pensamento Computacional, sendo utilizadas para exercitar as habilidades como decomposição, generalização e abstração. Pensar computacionalmente proporciona aos estudantes a compreenderem a Computação enquanto ciência, modificar a forma como lidam

com problemas e criam as suas soluções. Dessa forma, transfere o foco do computador para os conceitos e teorias e transforma estudantes de consumidores a criadores de soluções tecnológicas.

3.6 CONSIDERAÇÕES FINAIS

De acordo com o mapeamento sistemático realizado, um dos desafios do Pensamento Computacional no ensino superior está relacionado com o engajamento de estudantes. Os artigos que abordam o engajamento, afirmam que as atividades realizadas possuem relação direta com a motivação e engajamento dos estudantes. A atividade como um todo, a forma como é apresentada, os requisitos, e até o nível de dificuldade podem impactar no interesse e no envolvimento para realizá-la. Kim e Kim (2017) sinalizam que muitos estudantes não participaram de todas as aulas, consequentemente deixaram de realizar várias atividades, e perceberam que era comum a baixa motivação entre eles. Como uma estratégia para diversificar as oportunidades de aprendizagem, Gardner et al. (2011) incluíram inicialmente atividades opcionais, porém, perceberam que estudantes apenas se envolviam e participavam das atividades se elas valessem nota.

Se pensarmos que a experiência de aprendizagem em sala de aula é construída de maneira incremental, e que as atividades vão evoluindo em complexidade à medida que elas vão sendo conduzidas, deixar de realizar uma significativa porcentagem dessas atividades impactará na construção do conhecimento. Logo, a falta de conhecimento para realizar uma atividade, ou atividades muito complexas, pode resultar na desmotivação dos estudantes. Com a pesquisa de Gardner et al. (2011), identificamos as problemáticas dos estudantes assumirem um papel muito passivo em seus processos de aprendizagem e a cultura das notas existentes na educação. Quando falamos no ensino superior, além de ser uma etapa educacional, devemos ter a consciência de que ele ajuda na formação e capacitação de conhecimentos e habilidades.

Mesmo que o ensino superior esteja relacionado a dominar uma área específica, para futuramente exercer uma profissão, ele está ligado ao desenvolvimento crítico de pessoas e na evolução enquanto indivíduos. Com isso, devemos enquanto professores e pesquisadores desmistificar a cultura das notas e fomentar o aprendizado para a vida. Mostrando a necessidade dos estudantes adotarem uma postura aberta ao conhecimento, expondo que o que importa é o conhecimento desenvolvido ao longo da formação, e que nenhuma nota ou diploma, por mais requisitado que seja, pode encerrar a jornada de desenvolvimento.

Pelo ângulo de apresentar sentido para o processo de ensino e aprendizagem, a Aprendizagem Significativa tem foco em desenvolver soluções que são importantes para cada um dos estudantes e para a comunidade que eles estão inseridos. A educação deve fornecer experiências para mostrar que conceitos abstratos fazem sentido e são aplicáveis ao mundo cotidiano. A aprendizagem contextualizada possibilita o estudante a construir significado com base em suas próprias experiências. Segundo Freire (1997), é importante utilizar informações do contexto que os estudantes estão familiarizados, assim, o conhecimento prévio contribui significativamente para o aprendizado e evolução desses estudantes.

A junção da aprendizagem significativa e da aprendizagem contextualizada possibilita estudantes construírem os conhecimentos a partir de conhecimentos prévios e de sua experiência. Portanto, uma abordagem significativa e contextualizada busca utilizar-se de problemas reais e atuais para explicar os conteúdos que serão ensinados, proporcionando estudantes a se debruçar na criação de soluções que podem gerar impactos reais. Ainda, as atividades propostas devem ser ancoradas no contexto diverso dos estudantes, o que permite uma troca de saberes e incentiva os estudantes a aprenderem uns com os outros.

O processo de ensino e aprendizagem deve ser guiada por propósito, sempre em busca de algo maior (Tissenbaum et al., 2019). O propósito também será responsável por mostrar quando

os conceitos abstratos podem ser aplicados, esclarecendo alguns porquês e comos. Aproximar a sala de aula ao contexto dos estudantes, com problemas reais e atuais são formas de trazer propósito para a educação. Tissenbaum et al. (2019) revela que o engajamento traz o sentido de coletividade, direciona as pessoas a buscarem o mesmo objetivo e alinha esses objetivos em busca de uma solução em comum. O engajamento entre estudantes reduz as barreiras de comunicação, permitindo que eles se sintam mais confortáveis ao trabalhar em grupos - produzindo um sentimento de segurança e apoio.

Outro desafio de exercitar o Pensamento Computacional no ensino superior está relacionado às dificuldades dos estudantes expressarem e formalizarem as soluções, principalmente estudantes iniciantes. Pesquisas sinalizaram que estudantes apresentam problemas para expressar suas soluções no papel, não conseguiram criar nenhuma descrição formal de seus algoritmos, e apresentaram dificuldades em fazer representações computacionais (Cápay, 2014; Yeh et al., 2011). Considerando iniciativas que estavam relacionadas com conteúdos de programação, a principal dificuldade estava relacionada aos diferentes aspectos da programação que confundiam os estudantes (Barba e Chancellor, 2015; Bryant et al., 2011; Davies, 2008). Mesmo que Davies (2008) relate que aprender os conceitos de programação sem uma linguagem de programação não parece resultar em maior competência geral, Caeli e Yadav (2020) e Curzon et al. (2019) afirmam que é interessante iniciar a exercitar o Pensamento Computacional com uma abordagem desplugada pois ela estimula o pensamento lógico, ajuda a desenvolver o Pensamento Computacional e as habilidades de resolução de problemas.

Para além dos desafios existentes, alguns artigos selecionados no mapeamento sistemático sinalizam benefícios de exercitar o Pensamento Computacional no ensino superior. Os benefícios mencionados vão desde a evolução dos estudantes quanto à compreensão dos conceitos computacionais e desenvolvimento do Pensamento Computacional até a redução da evasão dos cursos. É interessante perceber que os benefícios estão diretamente relacionados com os desafios encontrados. Nas pesquisas de Miller et al. (2014), Gero e Levin (2019), e Buckingham e Hogan (2014) estudantes se sentiram mais à vontade por não precisarem aprender uma linguagem de programação específica, e apresentaram melhores resultados em atividades desplugadas. Além disso, os estudantes aprenderam a compreender os requisitos do problema e abstrair a essência para iniciar a solução (Kim e Kim, 2017; Vieira et al., 2019; Philip et al., 2013; Tang et al., 2018).

Nos cursos de graduação em Computação, exige-se que estudantes ingressantes desenvolvam as habilidades de resolução de problemas e aprendam pelo menos uma linguagem de programação para a formalização lógica e estrutural da solução. Esse processo envolve a aprendizagem da semântica, da sintaxe da linguagem de programação, do ambiente de desenvolvimento, e a conexão do pensamento algorítmico com o entendimento do processamento computacional para a produção do resultado. De acordo com Gouws et al. (2013), ao focar na semântica em vez da sintaxe de uma linguagem específica, estudantes desenvolvem primeiro habilidades de solucionadores de problemas e posteriormente de codificadores.

Nesse sentido, a utilização de atividades desplugadas pode auxiliar na redução da carga cognitiva elevada enfrentada pelos estudantes ingressantes. O conceito de desplugada, por definição, significa utilização de algo de maneira desconectada e analógica (Bell et al., 2012). Na Computação, o conceito de desplugada está associado a atividades que utilizam os conceitos fundamentais da Ciência da Computação, porém, não fazem uso de nenhuma tecnologia. Aplicar atividades com conteúdos computacionais sem estar na frente do computador garante que os estudantes separem o problema da instrumentalização da solução.

Metodologias mais flexíveis, resultados mais tangíveis, e trabalho em equipe são aspectos que impactam positivamente na motivação dos estudantes. Segundo Kazimoglu et al. (2012) e Almeida et al. (2018), trabalhar em grupo aumenta a motivação dos estudantes porque é

uma oportunidade de aprender coletivamente. A interação entre estudantes revelou que eles podem aprender uns com os outros, especialmente quando fornecem ou recebem feedback, o que mantém eles envolvidos ativamente na experiência de aprendizagem. A estratégia pedagógica em que estudantes trabalham em busca de objetivos comuns de aprendizagem é conhecida como Aprendizagem Colaborativa. Além de serem incentivadas a interagirem e compartilharem conhecimentos, a Aprendizagem Colaborativa procura envolver os estudantes a desenvolverem habilidades sociais, como comunicação, resolução de problemas, negociação e flexibilidade, além de melhorar a experiência do aprendizado.

4 PENSE COMPUTACIONALMENTE: UMA ABORDAGEM BASEADA EM DESAFIOS

Pense Computacionalmente é uma abordagem baseada em desafios que tem o objetivo apoiar o exercício do Pensamento Computacional em cursos de graduação em Computação. Entende-se como desafio um problema, um obstáculo, ou uma situação que pode ser melhorada. Os desafios não possuem uma única solução, e estão associados a problemas maiores que demandam discussões. A abordagem foi concebida com a intenção de apoiar professores e pesquisadores na concepção e condução de práticas educacionais que promovam o exercício do Pensamento Computacional por meio do processo de solução dos desafios considerando os quatro pilares do Pensamento Computacional: abstração, decomposição, reconhecimento de padrões e algoritmos.

A abordagem tem como premissa de que o Pensamento Computacional deve ser explicitamente exercitado desde o primeiro semestre dos cursos de graduação em Computação, de forma significativa, colaborativa, contextualizada, com uma abordagem desplugada e baseada em desafios. As atividades devem ser baseadas em problemas reais e ancoradas no contexto acadêmico onde serão realizadas, e as atividades devem fornecer experiências para mostrar que conceitos abstratos fazem sentido e são aplicáveis ao mundo cotidiano, proporcionando aos estudantes a construírem significado com base em suas próprias experiências.

A abordagem está fundamentada em: aprendizagem significativa, aprendizagem contextualizada, aprendizagem colaborativa, aprendizagem baseada em desafios, e computação desplugada. Os cinco pilares da abordagem dão luzes de como as práticas devem ser realizadas para apoiar o exercício do Pensamento Computacional. As práticas propostas devem ser ancoradas no contexto diverso dos estudantes, o que permite uma troca de saberes e incentiva eles a aprenderem uns com os outros (significativa). Ao utilizar problemas reais e contextualizados na realidade dos estudantes consegue-se gerar sentido e motivação para o aprendizado (contextualizada). A colaboração entre estudantes permite que eles construam o conhecimento por meio da colaboração entre eles (colaborativa). Uma prática deve ser criada baseado em problemas que possibilitem que estudantes apliquem as soluções, facilitando visualizar o comportamento da solução (baseada em desafios). Para que os conceitos computacionais sejam colocados em prática é necessário começar com uma abordagem desplugada pois ela estimula o pensamento lógico e ajuda a desenvolver o Pensamento Computacional e as habilidades de resolução de problemas (computação desplugada).

Desenvolvemos a abordagem para ser utilizada ou adaptada para diferentes contextos educacionais, e ela é composta por: (i) um conjunto de orientações de como preparar e conduzir o exercício do Pensamento Computacional; (ii) um conjunto de sete práticas para serem utilizadas ou inspirar a criação de novas; (iii) um método para guiar a condução da prática e instruir estudantes durante a proporem uma solução; e (iv) um artefato para apoiar a análise das soluções para coletar evidências do exercício do Pensamento Computacional.

4.1 PRÁTICAS DA ABORDAGEM

A abordagem Pense Computacionalmente conta com um conjunto de sete práticas, também chamados de desafios, que pode ser utilizado ou adaptado para outros contextos. A estrutura base de uma prática é composta por: (i) um desafio e a sua descrição; (ii) restrições a serem respeitadas durante a criação da solução; (iii) explicação sobre os objetivos da prática; e (iv) as habilidades a serem exercitadas. A descrição do desafio é responsável por detalhar o problema e contextualizá-lo, além de deixar explícito quais são os objetivos, os benefícios e as habilidades

que ele exercita. Para cada desafio uma solução deve ser proposta, e as soluções devem ser representadas de duas formas: sequência de instruções em linguagem natural[1] ou fluxograma[2].

Selecionamos um conjunto de 9 habilidades do Pensamento Computacional que serão exercitadas nas práticas, são elas: abstração, decomposição, reconhecimento de padrões, coleta e análise dos dados, algoritmos, generalização, representação da solução, simulação e teste e otimização – as definições podem ser encontradas na Tabela 2.1. As *soft skills*[3] exercitadas possuem relação direta com as habilidades que estudantes de Computação devem exercitar, e que são apontadas também pelas diretrizes curriculares dos cursos de graduação em Computação. Assim, chegamos ao conjunto de 9 *soft skills*, sendo: comunicação, criatividade e inovação, gestão de tempo, negociação e flexibilidade, pensamento analítico, pensamento crítico, resolução de problemas, responsabilidade e ética profissional, e trabalho em equipe. A Tabela 4.1 detalha as *soft skills*. Cada prática auxilia no exercício do Pensamento Computacional e *soft skills*.

Tabela 4.1: Descrição das soft skills

Soft Skill	**Descrição**
Comunicação	Habilidade de ouvir e se fazer entender de maneira clara e eficaz, levando em consideração o contexto e o público-alvo.
Criatividade e Inovação	Habilidade de combinar ideias novas ou existentes para gerar um resultado ou solução nova, considerando o seu conhecimento prévio, repertório e experiências.
Gestão de Tempo	Habilidade de definir prioridades e metas, gerenciando o tempo necessário para realizar uma determinada tarefa e propor uma solução.
Negociação e Flexibilidade	Habilidade de dialogar com pessoas e buscar uma solução que satisfaça os envolvidos, defendendo os seus interesses, mas também reconhecendo quando ceder.
Pensamento Analítico	Habilidade de analisar, resumir, quantificar e cruzar dados para chegar a uma conclusão.
Pensamento Crítico	Habilidade de refletir, criticar, julgar e cruzar informações, teorias e no conhecimento prévio para chegar em uma opinião própria.
Resolução de Problemas	Habilidade de analisar o cenário do problema, determinar as causas do problema e buscar soluções para ele, considerando os recursos existentes.
Responsabilidade e Ética Profissional	Habilidade de conduzir as atividades e relações profissionais norteando suas ações baseadas em valores individuais e coletivos, desenvolvendo soluções socialmente responsáveis que considere questões sociais, éticas, culturais.
Trabalho em Equipe	Habilidade de interagir e trabalhar com outras pessoas na mesma solução, sabendo ouvir, negociar e respeitar todas as pessoas que compõem a equipe.

O foco de exercitar essas habilidades é preparar os estudantes a atuarem na criação de soluções para problemas, de maneira criativa, colaborativamente e socialmente consciente. O exercício do Pensamento Computacional e das *soft skills* possuem relação direta com a condução

[1]Linguagem natural é a forma mais comum de comunicação, à qual utilizamos no dia a dia e estamos mais familiarizados, e pode ser compreendida como a forma utilizada pelas pessoas para falar, escrever e se comunicar, de acordo com a sua cultura e nacionalidade. Portanto, representar a solução em instruções em linguagem natural é criar uma sequência de passos que deve ser realizada para chegar no resultado esperado.

[2]Fluxograma é uma representação esquemática gráfica de um processo, que ilustra de forma descomplicada a sequência de passos do desenvolvimento de um processo. Um fluxograma geralmente é composto por figuras geométricas padronizadas e setas que indicam o fluxo e unem as figuras. Por meio de uma representação gráfica é possível compreender a sequência de passos a serem percorridos durante o processo.

[3]Soft skills é um termo em inglês utilizado para referir-se às habilidades interpessoais. De um modo geral, essas habilidades são transpostas para todos os níveis ao longo de uma vida, principalmente na caminhada educacional, e possuem relação direta com o contexto social e educacional.

das práticas, logo, ao retirar, adicionar, ou até mesmo focar em uma das etapas do método impactará no conjunto final de habilidades exercitadas pelos estudantes. Portanto, a condução de uma prática precisa ser guiada com base nas habilidades que se deseja exercitar.

4.1.1 Catálogo de Práticas

A abordagem Pense Computacionalmente conta com um catálogo de sete práticas, que serão apresentadas nas sequências. O principal objetivo desse catálogo é proporcionar material base para conduzir práticas para o exercício do Pensamento Computacional, seguindo os princípios apresentados pela abordagem. Além disso, durante todo o catálogo existem dicas de como adaptar as práticas, de forma a aproximá-las do contexto em que será aplicado. Assim, o catálogo de práticas é um norte para professores e pesquisadores utilizarem ou adaptarem os desafios existentes. Cada prática é composta por um desafio e sua descrição, as restrições a serem respeitadas pela solução, explicação sobre os objetivos da prática e as habilidades a serem exercitadas. A descrição do desafio é responsável por detalhar o problema e contextualizá-lo.

Cada desafio exercita um conjunto de habilidades do Pensamento Computacional. Essas habilidades possuem relação com o objetivo do desafio, a temática do desafio e os passos necessários para propor uma solução para o desafio. O conjunto dos desafios foi criado para exercitar de forma mais intensa pelo menos cada uma das 9 habilidades. A quantidade de habilidades e a intensidade em que essas habilidades serão exercitadas podem ser utilizadas como indícios para saber em qual ordem os desafios podem ser conduzidos. Quanto maior o número de habilidades, mais aspectos precisam ser considerados para propor uma solução, e maior também tende a ser a carga cognitiva esperada para resolver o desafio. Assim, podemos relacionar a dificuldade dos desafios com a quantidade de habilidades que se busca exercitar.

A relação de habilidades para cada desafio pode ser vista na Figura 4.1, e as cores utilizadas representam a intensidade que o desafio exercita de cada uma das habilidades. As habilidades possuem relação com os passos necessários para solucionar cada desafio. O conjunto de habilidades destacadas para cada desafio representa as habilidades em evidência, logo, outras habilidades também são exercitadas e reforçadas durante a resolução de um desafio.

Desafio. Mais velha[4] — o objetivo do desafio é chegar em uma resposta para o problema apresentado, sendo um problema de lógica em que duas amigas se encontram e uma delas desafia a outra a descobrir a idade de suas filhas com apenas 3 dicas. O ponto principal do desafio é a interpretação das dicas, e para chegar nas idades é necessário interpretar cada uma delas de maneira sequencial. Este desafio busca mostrar a importância de interpretação, entendimento e representação da ideia, principalmente antes de começar a propor uma solução. As discussões podem levar as práticas computacionais de teste e *debugging*, mostrando a relevância e a importância delas. Este desafio busca desenvolver a habilidade de interpretação do problema e a estruturação da solução, refletindo nos passos necessários para chegar no resultado. O desafio possui apenas uma resposta correta, e entender as dicas é essencial.

Desafio. Teste dos Labs[5] — o objetivo é desenvolver a capacidade de interpretação do problema e a estruturação da solução, refletindo nos passos necessários para chegar no resultado do desafio. O desafio consiste em caracterizar cada laboratório com base em dicas e nas relações entre as dicas apresentadas. Neste teste existem 5 laboratórios do Departamento de Informática (DInf) da UFPR, cada laboratório possui uma área de pesquisa, um docente representante, que recomenda um livro para estudos e que utiliza uma linguagem de programação. Este desafio foca

[4] Adaptado de E. Maschio. Projeto Imersa: O mais velho tem olhos azuis. Disponível em: https://imersa.blog/o-mais-velho-tem-olhos-azuis-6e23bbc91894

[5] O teste dos laboratórios foi criado por um estudante para uma atividade da disciplina de Introdução à Ciência da Computação do 2º semestre de 2019, e foi baseado no teste popularmente conhecido como *Teste de QI de Einstein.*

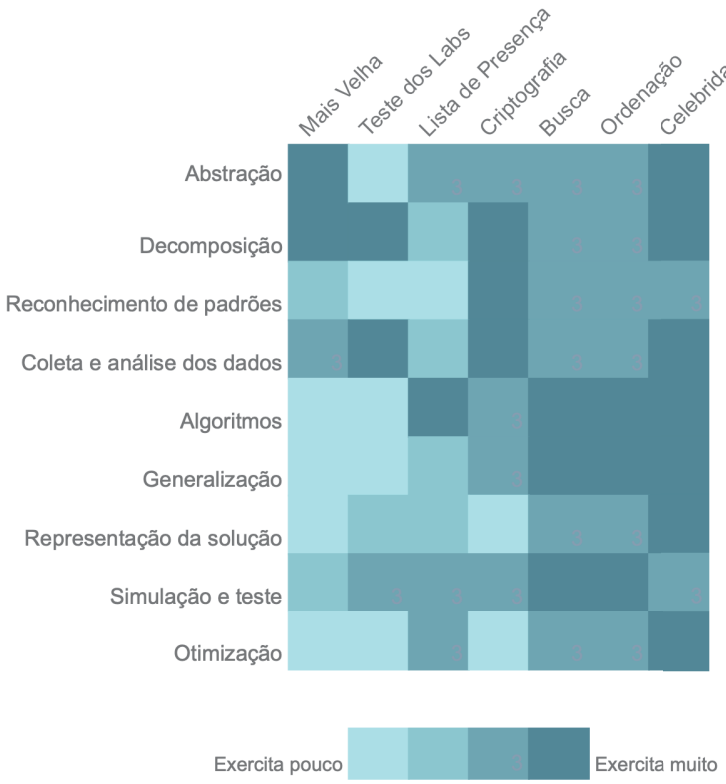


Figura 4.1: Habilidades exercitadas por desafio

em desenvolver a habilidade de interpretação do problema e a estruturação da solução, refletindo nos passos necessários para chegar no resultado do desafio. O desafio possui apenas uma resposta correta e entender as dicas é essencial para chegar na solução. A Figura 4.2 mostra a folha de resposta do desafio.

	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5
Nome do Lab					
Docente					
Área de Pesquisa					
Linguagem de Programação					
Livro Recomendado					

Figura 4.2: Folha do Teste dos Labs

Desafio. Lista de Presença — O objetivo é propor um modo estruturado e otimizado de passar a lista de presença da aula para que ela seja assinada por todos os estudantes presentes. Este desafio foi originário de um problema que ocorria em sala de aula: a demora para todos assinarem a lista de presença, muitas vezes a aula terminava e ainda havia pessoas que não conseguiam assinar a lista. A solução precisa respeitar a distribuição da sala, não deve restringir a escolha de lugares, e que a lista deve ser impressa e assinada por todos as pessoas matriculadas e presentes. Neste desafio, serão apresentados ao conceito de resolução de problemas e simulação

e testes da solução. Assim, ao observar os testes e a execução será possível fazer uma reflexão sobre a importância de planejar e testar soluções.

Desafio. Criptografia — O objetivo visa apresentar o conceito computacional de criptografia e segurança de dados. Neste desafio, trabalharemos também a codificação de informação, criptografia de mensagem e processo de descriptografia. O desafio apresenta uma frase criptografada e pede para descriptografar a mensagem e explicar como descriptografou a mensagem. O foco deste desafio é desenvolver habilidades de abstração, decomposição, coleta e análise dos e reconhecimento de padrões.

Desafio. Busca — O objetivo é exercitar os conceitos de busca computacional, ou também conhecido como pesquisa. O desafio busca verificar se um determinado objeto está presente no conjunto ou encontrar um objeto que atenda critérios pré-definidos. Para este desafio trabalharemos também com a visibilidade de mulheres notáveis da Computação, recomendamos a utilização do conjunto de cartas *Notable Women in Computing* que representa essas mulheres e está disponível para download. Assim, as soluções propostas para o desafio podem ser testadas em sala, e ao observar a execução será possível fazer uma reflexão sobre como os elementos, as informações e a organização dos elementos impactam na solução.

Desafio. Ordenação — O objetivo é exercitar os conceitos de ordenação computacional, ou também conhecido como *sorting*. Este desafio tem o objetivo organizar um conjunto de n cartões, inicialmente não ordenados, com base em algum critério pré-definido, podendo ser: nome, data de nascimento, identificador, entre outras informações. O desafio consiste em definir uma sequência de instruções para ordenar um conjunto de cartões considerando os critérios previamente definidos. A solução deve respeitar que cada cartão ocupa apenas uma posição na matriz unidimensional (vetor), e não pode ser colocada em cima de outra e nem ocupar duas posições ao mesmo tempo. Neste desafio também recomendamos a utilização do conjunto de cartas *Notable Women in Computing*. A simulação da solução pode ser feita com cartões reais, facilitando a reflexão sobre a solução e correção caso um problema seja encontrado.

Desafio. Celebridade [6] — O desafio busca determinar se existe uma celebridade em uma festa e nomeá-la, caso exista uma. Para uma pessoa ser uma celebridade, ela precisa ser conhecida por todas as pessoas presentes na festa, mas não pode conhecer ninguém. Para identificar quem é celebridade é necessário realizar perguntas a todas as pessoas presentes, como o número de dados é significativo é necessário tomar nota das respostas e controlar as perguntas. Como restrição, só pode fazer uma pergunta por vez e perguntas de resposta sim ou não. Não pode perguntar se conhece um grupo de pessoas e nem generalizar as perguntas. Exemplo possível de pergunta: Ada, você conhece a Margaret Hamilton? Um aspecto importante deste desafio é pensar na otimização, pois será necessário conversar com todas as pessoas presentes. A partir desse desafio é possível apresentar os conceitos de grafos e otimização.

4.1.2 Representação da Solução

A abordagem Pense Computacionalmente recomenda que as soluções sejam representadas como sequência de instruções em linguagem natural ou fluxograma. Linguagem natural é a forma mais comum de comunicação, a qual utilizamos no dia a dia e estamos mais familiarizados, e pode ser compreendida como a forma utilizada pelas pessoas para falar, escrever e se comunicar, de acordo com a sua cultura e nacionalidade. Portanto, representar a solução em instruções em linguagem natural é criar uma sequência de passos que devem ser realizados para chegar no resultado esperado. Fluxograma é uma representação esquemática gráfica de um processo, que

[6] Desafio inspirado em problema apresentado no livro Introduction to Algorithms: A Creative Approach (Udi Manber).

ilustra de forma descomplicada a sequência de passos do desenvolvimento de um processo. Um fluxograma geralmente é composto por figuras geométricas padronizadas e setas que indicam o fluxo e unem as figuras. Por meio de uma representação gráfica é possível compreender de forma rápida e fácil a sequência de passos a serem percorridos durante o processo.

4.2 MÉTODO

A abordagem Pense Computacionalmente reconhece que mesmo que as práticas sejam diferentes entre si elas possuem uma estrutura básica que deve ser respeitada durante o processo de criar a solução para o desafio. A estrutura básica foi dividida em 6 etapas, que podem envolver tanto trabalho individual quanto em grupos. Para representar a estrutura básica para guiar o processo de resolução de uma prática criamos um método. O método foi representado de maneira cíclica para representar a relação de dependência entre as etapas, e ilustrar o processo de amadurecimento da solução — existe uma relação de amadurecimento da solução à medida que as etapas vão sendo concluídas. As etapas do método foram pensadas para reservar momentos específicos para elaborar, testar, compartilhar, refinar e otimizar a solução.

O processo pode ser repetido quantas vezes desejar ou forem necessárias. A Figura 4.3 ilustra o processo a ser conduzido durante o momento de resolução das práticas. No fim de cada ciclo, espera-se que as soluções estejam superiores à solução inicial, ou do ciclo anterior (caso executado mais de um ciclo). As etapas do método foram pensadas para guiar o processo de criação da solução, por isso ele foi dividido em seis etapas, são elas:

Etapa 1) Caracterizar e representar o problema, identificando seus requisitos e restrições;

Etapa 2) Elaborar e representar uma solução inicial;

Etapa 3) Testar a solução inicial, a fim de verificar se ela resolve o problema caracterizado na etapa 1;

Etapa 4) Socializar as soluções, para receber comentários sobre a solução proposta, considerando a corretude e facilidade de execução da solução;

Etapa 5) Refinar a solução proposta; e

Etapa 6) Testar a solução e avaliar os resultados que ela produz.

No método foi previsto tanto momentos para trabalhar individualmente quanto em grupos. As etapas em grupo são destinadas para que os estudantes compartilhem entre si suas soluções, recebam e forneçam comentários sobre as soluções, apontando aspectos que podem ser corrigidos ou melhorados. Ainda, as etapas realizadas em grupo podem resultar em uma nova solução — considerando o melhor das soluções de cada integrante do grupo. A etapa de socialização representa um processo de amadurecimento de maneira colaborativa, uma vez que os estudantes avaliam, argumentam e apontam correções entre eles. A utilização do método e a condução do processo de resolução da atividade possui relação direta com as habilidades que serão exercitadas durante o processo.

Cada desafio exercita um sub-conjunto de habilidades do Pensamento Computacional, são elas: Abstração, Decomposição, Reconhecimento de Padrões, Coleta e Análise dos Dados, Algoritmo, Generalização, Representação da Solução, Simulação e Teste, e Otimização. Essas habilidades possuem relação com os passos necessários para propor uma solução para o desafio. Pelo fato de o exercício do Pensamento Computacional estar relacionado com a condução do



Figura 4.3: Método para guiar a resolução das práticas

desafio, é necessário conduzir a resolução das práticas com base nos objetivos explicitados em cada desafio. Assim, é possível relacionar as habilidades exercitadas em cada etapa do processo de resolução de um desafio. A Figura 4.4 ilustra a relação das habilidades exercitadas durante cada etapa da condução da prática prevista pelo método.

Durante a etapa (1), para identificar quais são as informações fornecidas, os resultados esperados, e quais são as restrições, são exercitadas as habilidades de abstração, decomposição e reconhecimento de padrões. Ao caracterizar o problema na etapa (1) o pensamento crítico, e habilidades de resolução de problemas e responsabilidade ética e profissional também são exercitadas. Na etapa (2), a elaboração e representação da solução desenvolvem a habilidade de algoritmos, generalização, coleta e análise dos dados, reconhecimento de padrões e representação da solução. Ao testar a solução, na etapa (3), é desenvolvido as habilidades de decomposição, coleta e análise dos dados, simulação e teste. Resolução de problemas, pensamento crítico e analítico são as *soft skills* exercitadas na etapa (3).

A etapa (4) é destinada para socialização da solução, assim, ao trabalhar de maneira colaborativa desenvolve-se as *soft skills* de trabalho em equipe, negociação e flexibilidade, responsabilidade e ética profissional e criatividade e inovação. Após a etapa de socialização, a etapa (5) foca em refinar a solução proposta, desenvolvendo as habilidades de algoritmos, otimização e generalização. Por meio da etapa (6), de teste e avaliação dos resultados, é exercitada as habilidades de coleta e análise dos dados, simulação/teste e otimização. A temática da prática também ajuda a desenvolver habilidades como pensamento crítico, responsabilidade ética e

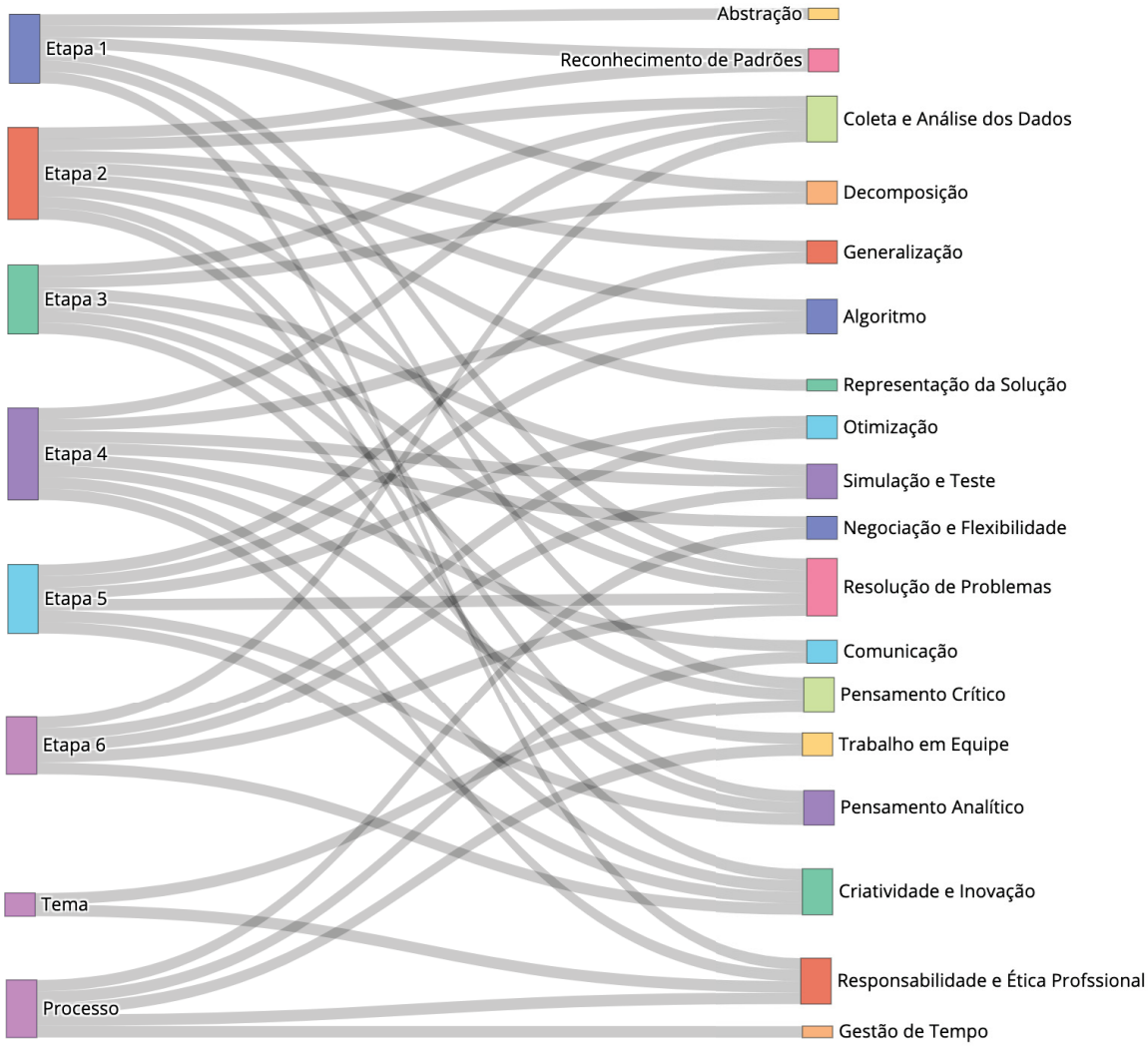


Figura 4.4: Relação entre as Etapas e Habilidades

profissional. Além disso, durante todo o processo de resolução dos desafios as habilidades de gestão de tempo, resiliência e flexibilidade são exercitadas.

A Tabela 4.2 apresenta a correlação das habilidades do Pensamento Computacional com os passos necessários para solucionar um desafio. Essa relação é feita considerando as seis etapas do processo de resolução, apresentadas na Figura 4.2.

4.3 COMPUTACIONALIDADE: ARTEFATO PARA COLETA DE EVIDÊNCIAS DO EXERCÍCIO DO PENSAMENTO COMPUTACIONAL

Computacionalidade é um artefato para apoiar professores e pesquisadores durante o processo de análise e avaliação das soluções de atividades que visam exercitar o Pensamento Computacional, e tem como objetivo fornecer uma estrutura e rigor para a condução das análises. O artefato é composto por: (i) descrição do conjunto de habilidades e suas definições; (ii) instruções sobre como e quais evidências coletar; e (iii) uma escala para mensurar as evidências do exercício das habilidades. O subconjunto de habilidades selecionadas está presente na Tabela 2.1.

Para cada uma das habilidades do conjunto, definimos como seria identificada a evidência do seu exercício, e enumeramos pelo menos uma evidência a ser coletada. A descrição de como

Tabela 4.2: Relação das habilidades do Pensamento Computacional com os passos dos desafios.

Habilidades do PC	**Passos dos Desafios**
Abstração	Realizar interpretação do desafio. Identificar quais são as entradas e as saídas do desafio. Eliminar os detalhes desnecessários.
Decomposição	Identificar as partes de um desafio e subdividi-lo quando necessário. Reaproveitar soluções desenvolvidas para resolver o mesmo desafio.
Reconhecimento de Padrões	Identificar os padrões do desafio e da solução. Reaproveitar as instruções por meio de repetições e condições. Transferir ideias entre desafios.
Coleta e análise dos dados	Identificar e coletar os dados necessários para a solução. Identificar como os dados estão representados e organizados. Utilizar os dados para solução do desafio.
Algoritmos	Formalizar a sequência de passos necessários para resolver o desafio. Refinar a solução.
Generalização	Propor soluções que possam ser utilizadas com diferentes quantidades de dados.
Representação da Solução	Representar a solução para ser interpretada por alguém ou alguma, considerando os recursos e limitações disponíveis.
Simulação e Teste	Simular as resoluções de acordo com o cenário do desafio. Executar a resolução do colega e propor correções. Comparar o desempenho de soluções semelhantes.
Otimização	Evoluir a solução considerando os pontos que podem reduzir a quantidade de interações. Propor melhorias na solução do colega.

verificamos as evidências do exercício das habilidades e a enumeração das evidências a serem coletadas pode ser vista na Tabela 4.3. Também desenvolvemos uma escala simplificada para verificar os indícios do exercício das habilidades do Pensamento Computacional nas soluções das atividades. A escala foi inspirada na escala utilizada em França e Silva (2020). Cada uma das habilidades exercitadas pelas atividades pode ser mensurada da seguinte forma:

[ ]: Não se aplica

0: Não existe evidência do exercício dessa habilidade

1: Existe pouca evidência do exercício dessa habilidade

2: Existe evidência do exercício dessa habilidade

3: Existem muitas evidências do exercício dessa habilidade

Tabela 4.3: Definição das habilidades e enumeração das evidências a serem coletadas

Habilidade	**Evidências**
Abstração	Temos evidências que houve o exercício da abstração quando o contexto do desafio é compreendido e utilizado para identificar quais informações são importantes e quais devem ser utilizadas para propor a solução. Abstrair garante que os diferentes aspectos do desafio estão sendo identificados e considerados durante a criação da solução, resultando em uma solução que atinja o esperado. Se a resposta estiver errada ou diferente do tipo previsto, temos evidências que houve problemas no processo de abstração. Observar se: 1. A solução apresenta a contextualização do desafio. 2. Foi identificado os requisitos essenciais para a solução. 3. Foi atingido o resultado esperado pelo desafio.
Decomposição	Temos evidências que houve o exercício da decomposição quando os diferentes subproblemas de um desafio são identificados e tratados. Dividir o desafio em partes menores favorece a simplificação do desafio em partes mais fáceis de serem gerenciáveis. Assim, utiliza-se dos sub-desafios de um desafio para compor a solução final. Observar se: 1. O desafio foi dividido em sub-desafios. 2. Utiliza-se da resolução dos sub-desafios para compor a solução final.
Reconhecimento de Padrões	Temos evidências que houve o exercício do reconhecimento de padrões quando são identificadas as similaridades (padrões) em um desafio ou solução. Logo, reconhecer soluções previamente conhecidas podem ser utilizadas novamente em problemas similares. Observar se: 1. Foi identificado os padrões do desafio. 2. Houve reaproveitamento de soluções. 3. Houve reaproveitamento de instruções por meio de repetição e condição.
Coleta e análise dos dados	Temos evidências que houve o exercício das habilidades de coleta e análise dos dados quando os dados são utilizados para auxiliar na resolução do desafio. Passando desde a fase de identificação, interpretação dos dados, quanto de como os dados atendem os requisitos do desafio e como eles serão utilizados para chegar na solução. Observar se: 1. As entradas de dados foram identificadas e entendidas. 2. As entradas de dados foram manipuladas respeitando as restrições e possibilidades do desafio.
Algoritmo	Temos evidências que houve o exercício da habilidade de algoritmo quando é criado uma sequência de passos para resolver o desafio. Observar se: 1. A solução fornece uma sequência de instruções para resolver o desafio. 2. As instruções estão detalhadas. 3. A solução está completa. 4. Reconhece a diferença entre as estruturas de condição e repetição. 5. Controla as repetições e condições. 6. Existe ponto de parada/fim.
Generalização	Temos evidências que houve o exercício da habilidade de generalização quando a solução para o desafio considera diferentes entradas de dados, não sendo restrito ao desafio em questão ou uma certa quantidade de dados. Observar se: 1. A solução funciona para um tamanho variável da entrada de dados *(n, n+1)*.

Representação da Solução	Temos evidências que houve o exercício da representação da solução quando a solução do desafio está expressada em uma sequência de passos finitos, sem ambiguidade e de forma que não seja necessário realizar inferências sobre a instrução. Ao final, a sequência de passos deve fornecer o resultado ou resolver o desafio proposto. Observar se: 1. As instruções estão organizadas e ordenadas de uma forma coerente. 2. A solução está representada considerando quem irá interpretar a solução. 3. As instruções estão escritas de forma clara e objetivas (i.e., não ambíguas).
Simulação e Teste	Temos evidências que houve o exercício da simulação/teste quando é realizada uma reprodução da solução com diferentes entradas de dados para identificar comportamentos diferentes da solução. É uma habilidade que se relaciona com a otimização, uma vez que para otimizar uma solução é interessante conhecer o comportamento dela. Observar se: 1. A solução é testável, i.e., oferece as condições necessárias para ser testada (e.g., instruções, entradas, saídas esperadas). 2. Há indícios de que a solução foi simulada/testada (e.g., a condição de parada é atendida). 3. Há indícios de tratamento de exceção ou de verificações de dados e condições. 4. A solução pode ser executada simultaneamente ou em cooperação com outras partes da solução.
Otimização	Temos evidências que houve o exercício da otimização quando uma solução considera as possibilidades de reduzir a quantidade de recursos necessários para atingir o resultado do desafio. Também, quando existe o processo de correção da solução existente ou refinamento de uma solução conhecida. Observar se: 1. Apresentou informações sobre correção e refinamento da solução existente. 2. Foi considerado na solução a redução de recursos necessário. 3. Foi reduzido o número de passos necessários para atingir o resultado. 4. Foi reduzido o número de passos utilizados para representar a solução.

O artefato foi planejado para avaliar diferentes aspectos que vão além de atingir o resultado esperado, pois entendemos que o processo de criar uma solução para uma atividade auxilia a exercitar, reforçar, e desenvolver conhecimentos. Esse é um dos principais motivos que o artefato detalha as habilidades e enumera evidências a serem coletadas, chamando a atenção para diferentes aspectos que devem ser observados durante a análise do exercício dessas habilidades. A escala serve para avaliar e mensurar em qual grau está sendo exercitada uma habilidade específica. A enumeração das evidências e as escalas ajudam a estruturar o processo de avaliação de uma atividade que busca exercitar o Pensamento Computacional.

Para sumarizar e representar o artefato, utilizamos o Google Sheets[7] como apoio, porém, qualquer outra ferramenta poderia ser utilizada. A escolha pelo Google Sheets se baseou nos recursos existentes, pela familiaridade que as pessoas possuem com os serviços Google, pela facilidade de uso, e por ser uma ferramenta web colaborativa e não necessitar de instalação. O artefato contém uma planilha para apresentar as explicações e uma planilha base para registrar as evidências a serem coletadas. A Figura 4.5 ilustra como a explicação do artefato foi organizada na planilha. A primeira página da planilha apresenta o conjunto de habilidades selecionadas e as suas definições, a lista de evidências a serem coletadas, e a escala de avaliação (ilustrada na Figura 4.5.a). A explicação está disponível na primeira planilha do artefato, ao lado das planilhas correspondentes a cada desafio, para estar próxima da utilização, acessível e ser consultada sempre que necessário (Figura 4.5.b).

[7]O Google Sheets é um programa de planilhas incluído como parte do pacote gratuito e online de Editores de Documentos Google oferecido pelo Google.

O artefato conta com uma planilha base, com todas as evidências que devem ser coletadas para o conjunto de habilidades. Na planilha base, as colunas representam as evidências que devem ser coletadas e cada linha representa um estudante. Cada célula (evidência x estudante) contém uma caixa de seleção com as opções da escala. O recurso de caixa de seleção foi utilizado para disponibilizar a escala e ser de fácil seleção. A Figura 4.6 ilustra a planilha base e como as evidências foram organizadas na planilha.

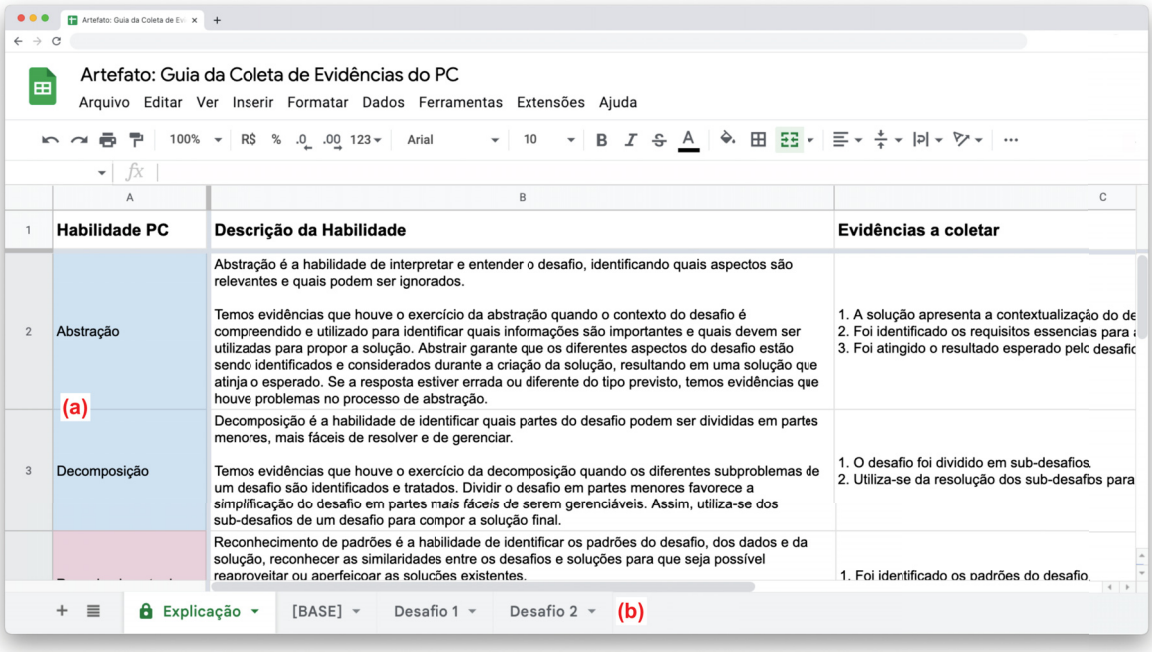


Figura 4.5: Planilha da explicação do artefato

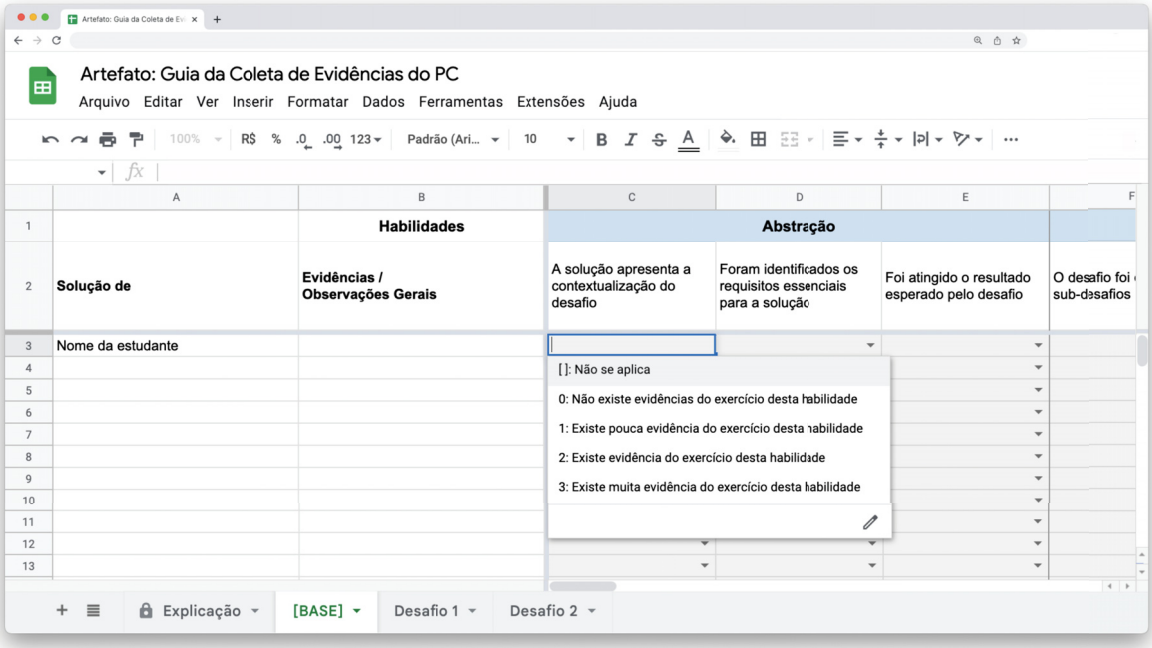


Figura 4.6: Planilha base para os desafios

4.4 UTILIZAÇÃO DA ABORDAGEM PENSE COMPUTACIONALMENTE

A abordagem Pense Computacionalmente foi desenvolvida para apoiar professores e pesquisadores durante a criação e condução de práticas educacionais que promovam o exercício do Pensamento Computacional no ensino Superior em Computação. A abordagem orienta como preparar e conduzir atividades para promover o exercício do Pensamento Computacional, fornece um conjunto de desafios que podem ser utilizados e um artefato para guiar a avaliação das soluções e coleta de evidências do exercício do Pensamento Computacional. A abordagem foi criada para ser utilizada ou adaptada para diferentes contextos educacionais, deixando dicas de como adaptar e personalizar para o contexto em que será utilizada. A abordagem Pense Computacionalmente está presente no Apêndice D, nele contém as orientações da abordagem, detalhes sobre o método de condução das práticas e dos desafios.

Para utilizar a abordagem é necessário primeiro identificar e definir os objetivos da prática educacional, isso ajudará na customização e personalização delas. É necessário também estabelecer o conjunto de habilidades do Pensamento Computacional que busca-se exercitar, uma vez que o exercício das habilidades possui relação direta com as atividades e como elas são conduzidas. Selecionar quais são as habilidades ajudará a identificar quais atividades podem ser utilizadas, ou revelar a necessidade de criação de novas. Além disso, enumerar as habilidades auxiliará na condução da resolução das práticas. Depois de definir os objetivos da prática educacional, é importante identificar quais desafios serão utilizados e, se necessário, personalizar os desafios selecionados para o contexto em que será utilizada. Para ajudar na adaptação, dicas foram deixadas ao longo da abordagem de como os ajustes poderiam ser feitos.

5 INVESTIGAÇÃO DA ABORDAGEM PENSE COMPUTACIONALMENTE: ESTUDOS DE CASO

Com o objetivo de compreender na prática sobre o exercício do Pensamento Computacional no ensino superior em Computação e investigar as potencialidades e fragilidades da abordagem desenvolvida foram realizados seis estudos de caso. Ao longo dos estudos de caso foram elaborados e aplicados um conjunto de práticas educacionais que focavam no exercício do Pensamento Computacional, a condução em sala de aula permitiu a investigação dentro do ambiente real. Os estudos foram realizados de maneira incremental, e no fim de cada estudo eram realizados diagnósticos sobre os resultados e lições aprendidas para aprimorar os materiais desenvolvidos. Durante o quinto estudo de caso, realizamos um experimento com professores e pós-graduandos colaboradores da disciplina, para verificar as potencialidades e fragilidades do artefato criado para coletar evidências do exercício do Pensamento Computacional. Os estudos de caso estão detalhados na Seção 5.1, o experimento realizado com o artefato está presente na Seção 5.3, e detalhes sobre a concepção da abordagem podem ser visto na Seção 5.4.

5.1 ESTUDOS DE CASO

Estudo de caso é uma estratégia de pesquisa científica que pode ser utilizada em diferentes situações para contribuir no entendimento de fenômenos complexos individuais, organizacionais, sociais, ou políticos (Yin, 2014). Estudo de caso é um estudo aprofundado de uma instância específica dentro de um contexto real específico, que possibilita uma observação profunda sobre o fenômeno investigado. No estudo de caso a coleta de dados é qualitativa e a análise dos dados é realizada para gerar novos *insights*, teorias e hipóteses que possam auxiliar na compreensão de um fenômeno no mundo real (Lazar et al., 2017). Segundo Yin (2014), mesmo que os estudos de caso sejam pequenos eles podem envolver mais de um caso, e essa estratégia é recomendada para aumentar a confiabilidade dos resultados.

Seis estudos de caso foram realizados com o objetivo de identificar as potencialidades e as fragilidades do uso de atividades baseadas em desafios para apoiar o exercício do Pensamento Computacional em cursos de graduação em Computação. A partir dos estudos de caso e da natureza construtiva da pesquisa, busca-se avançar o conhecimento sobre como exercitar e coletar evidências do exercício do Pensamento Computacional. Os estudos foram conduzidos no contexto da disciplina de Introdução à Ciência da Computação (CI1003) dos cursos de Bacharelado em Ciência da Computação e de Informática Biomédica, ambos da UFPR.

A disciplina CI1003 foi criada para promover o exercício de habilidades consideradas importantes para estudantes no primeiro semestre do curso de Computação da UFPR. O objetivo geral da disciplina é favorecer o Pensamento Computacional, crítico e sistêmico de estudantes sobre a Ciência da Computação e suas possibilidades de atuação. A disciplina possui carga horária semestral de 60 horas aula, é ofertada no 1º semestre da graduação, e é dividida em 3 blocos: Bloco 1. Introdução à Ciência da Computação; Bloco 2. Pensamento Computacional; e Bloco 3. Pensamento Sistêmico e Socialmente Consciente. A disciplina é detalhada no artigo de Pereira et al. (2021), no qual relatam sobre as ofertas que ocorrem desde 2019.

Os estudos de caso concentraram ações no Bloco 2, que tinha como objetivo promover habilidades de entendimento e resolução de problemas por meio do exercício do Pensamento Computacional. Cada estudo de caso foi conduzido em uma oferta diferente da disciplina e com um grupo diferente de estudantes. Dos seis estudos de caso realizados, os dois primeiros estudos

foram realizados de maneira presencial enquanto os 4 últimos foram realizados na modalidade Ensino Remoto Emergencial (ERE). A adaptação para o contexto remoto aconteceu porque a disciplina passou a ser ofertada de maneira remota. Detalhes técnicos dos estudos de caso realizados podem ser vistos na Tabela 5.1.

Tabela 5.1: Detalhes técnicos dos estudos de caso

Objetivo?	Identificar as potencialidades e as fragilidades do uso de atividades baseadas em desafios para apoiar o exercício do Pensamento Computacional em cursos de graduação em Computação.
Onde?	Disciplina de Introdução à Ciência da Computação (CI1003) dos cursos de Bacharelado em Ciência da Computação e curso de Informática Biomédica da UFPR.
Quantos estudos de caso foram realizados?	6
Quando os estudos de caso aconteceram?	Em seis ofertas diferentes da disciplina de CI1003, a primeira foi em 2019 e a última em 2021.
Quantos estudantes participaram?	358 estudantes matriculados na disciplina.
Quais foram os outros participantes?	3 professores da disciplina e 10 pós-graduandos colaboradores.
Quais métodos de avaliação foram utilizados?	(i) Observação participante, classificada como individual – quando apenas a pesquisadora está envolvida na observação sobre a perspectiva da ferramenta 5W1H; (ii) Registro em arquivo, no qual foram coletadas as soluções dos desafios para verificar as evidências do exercício das habilidades do Pensamento Computacional; (iii) Instância da disciplina no Moodle; (iv) Entrevistas semiestruturadas com os professores e colaboradores da disciplina; e (v) Questionário online e anônimo, para coletar a percepção dos estudantes.

Os participantes dos estudos foram pessoas envolvidas na disciplina, são elas: professores, pós-graduandos colaboradores e estudantes. Ao todo estiveram envolvidos 3 professores da disciplina, 10 pós-graduandos colaboradores, a autora da pesquisa e 358 estudantes matriculados na disciplina. A seleção dos participantes foi realizada de maneira não-probabilística de amostragem por conveniência (Wohlin et al., 2012), uma vez que participantes são do mesmo grupo de pesquisa da autora e pessoas envolvidas com a disciplina onde realizamos os estudos.

A análise dos dados foi qualitativa, em que o estudo visa tirar conclusões dos dados e manter uma clara cadeia de evidências (Wohlin et al., 2012). O método de análise dos dados utilizado foi a Análise Temática, pois permite identificar, analisar, interpretar e relacionar os padrões (temas) a partir de dados qualitativos com uma abordagem indutiva, que se baseia nos dados e não possui categorias ou temas preestabelecidos (Braun e Clarke, 2006). Como métodos para coleta de dados foram utilizados:

> (i) Observação participante, classificada como individual – quando apenas a pesquisadora está envolvida na observação sobre a perspectiva da ferramenta 5W1H;
>
> (ii) Registro em arquivo, no qual foram coletadas as soluções dos desafios para verificar as evidências do exercício das habilidades do Pensamento Computacional;
>
> (iii) Instância da disciplina no Moodle;

(iv) Entrevistas semiestruturadas com os professores e colaboradores da disciplina; e

(v) Questionário online e anônimo, para coletar a percepção dos estudantes.

A disciplina teve apoio da plataforma Moodle[1] para disponibilizar todo o material da disciplina, junto com explicações e instruções. O Moodle serviu como repositório de materiais, gerenciador das atividades, ambiente de entrega das atividades, e ambiente mediador do contato entre discentes, docentes e colaboradores. Nas ofertas remotas as atividades síncronas foram conduzidas com dinâmica de sala de aula invertida[2], tendo o foco na conversação entre discentes, docentes e colaboradores sobre os tópicos da disciplina. Para os encontros síncronos online a plataforma BBB[3] foi utilizada como ferramenta de apoio.

5.2 RESULTADOS E DISCUSSÕES SOBRE OS ESTUDOS DE CASO

Os resultados e discussões sobre os estudos de caso foram divididos em três partes: (i) o estudo piloto; (ii) os desafios 2.0; e (iii) os estudos realizados na modalidade Ensino Remoto Emergencial. Essa divisão aconteceu pelas grandes diferenças entre esses estudos. A partir de cada estudo, foi possível identificar e coletar evidências sobre o exercício do Pensamento Computacional. Além disso, de maneira interativa, no fim de cada estudo, foram identificados os problemas e estratégias eram traçadas para resolvê-los para a próxima interação. Nas próximas subseções serão apresentados os dados coletados durante os estudos e as discussões originárias desses dados. A sumarização das informações de cada estudo de caso pode ser visto na Tabela 5.2.

5.2.1 Estudo de Caso Piloto

O estudo de caso piloto foi realizado na oferta da disciplina de 2019/1, no período entre fevereiro e julho de 2019. A disciplina possuía duas turmas (A e B), com 49 estudantes matriculados em cada uma delas, totalizando 98 estudantes (91 homens e 7 mulheres), e todos os inscritos eram ingressantes do curso de Bacharelado em Ciência da Computação. A disciplina foi acompanhada por dois docentes e quatro colaboradores pós-graduandos. O objetivo do estudo piloto era elaborar e aplicar atividades que estimulem e exercitem o raciocínio lógico e promovam o exercício do Pensamento Computacional durante o processo de resolução das atividades. Com esse objetivo, foram elaboradas quatro práticas educacionais: Desafio. Lista de Presença, Desafio. Ordenação, Desafio. Busca, e Desafio. Celebridade.

Os desafios foram realizados presencialmente em sala de aula, e foram divididos em duas partes. Durante a primeira parte do desafio, os estudantes precisavam propor individualmente uma solução respeitando os requisitos e restrições. Na sequência, a segunda parte era destinada para socialização e amadurecimento da solução. Durante a socialização da solução, cada estudante deveria executar a solução de um colega, discutir, comparar e fornecer *feedback* sobre a solução – considerando a corretude e facilidade de execução. A socialização da solução tem como o objetivo instigar estudantes a melhorarem as suas soluções, ou criar uma nova solução utilizando o melhor de cada uma das soluções iniciais.

[1] O Moodle é um sistema de gestão da aprendizagem que serviu como repositório de materiais, gerenciador das atividades, ambiente de entrega das atividades, e ambiente mediador do contato entre discentes, docentes e colaboradores.

[2] A sala de aula invertida é um modelo pedagógico em que os estudantes recebem o conteúdo antes da aula e usam o tempo em sala de aula para atividades práticas e colaborativas.

[3] BigBlueButton, ou BBB, é um software de web-conferência que oferece recursos de interação focados em educação a distância.

Tabela 5.2: Resumo dos estudos de caso realizados

Oferta	**Objetivo**	**Período da Oferta**	**Modalidade da Oferta**	**Quantidade de Estudantes**	**Desafios conduzidos**	**Resultados**
Estudo Piloto 2019/1	Elaborar e aplicar atividades que estimulem o raciocínio lógico e o exercício do Pensamento Computacional durante o processo de resolução das atividades	19/02/2019 a 02/07/2019	Presencial	98 estudantes	Desafio. Lista de Presença Desafio. Ordenação Desafio. Busca Desafio. Celebridade	Foram elaborados quatro desafios que estimulem o raciocínio lógico e o exercício do Pensamento Computacional durante o processo de resolução das atividades. Os desafios foram utilizados em sala de aula, e a observação resultou em: um conjunto de lições aprendidas sobre como conduzir atividades para o exercício do Pensamento Computacional em sala de aula; e um artigo publicado no SBIE (Oliveira et al., 2019).
Desafios 2.0 2019/2	Desenvolver/evoluir atividades que estimulam o raciocínio lógico e o exercício do Pensamento Computacional durante o processo de resolução das atividades e criar orientações sobre a condução dos desafios.	12/08/2019 a 09/12/2019	Presencial	11 estudantes	Desafio. Celebridade Desafio. Busca Desafio. Mais Velha Desafio. Ordenação	Refinamento dos quatro desafios existentes, criação de 3 novos desafios, criação de orientações de como conduzir os desafios em sala de aula, formatação dos desafios em formato de plano de aula e folhas de atividades, e desenvolvimento de questionário de autoavaliação.
2020/ERE-1	Identificar a aplicabilidades dos desafios criados para que estimulam o raciocínio lógico e o exercício do Pensamento Computacional durante o processo de resolução um contexto remoto.	03/08/2020 a 14/09/2020	Remota	54 estudantes	Desafio. Teste dos Labs Desafio. Mais Velha Desafio. Criptografia Desafio. Ordenação Desafio. Celebridade	Adaptação dos desafios para serem conduzidos em um contexto remoto.
2020/ERE-2	Identificar as potencialidades e fragilidades dos desafios criados para que estimulam o raciocínio lógico e o exercício do Pensamento Computacional durante o processo de resolução.	03/11/2020 a 21/12/2020	Remota	46 estudantes	Desafio. Teste dos Labs Desafio. Mais Velha Desafio. Criptografia Desafio. Ordenação Desafio. Celebridade	Criação de orientações para conduzir os desafios em sala de aula, e coleta da percepção de professores e colaboradores por meio de entrevista.
2021/ERE-3	Aprofundar o conhecimento sobre as habilidades relacionadas com o Pensamento Computacional e como coletar evidências do seu exercício no conjunto de desafios utilizados.	17/05/2021 a 12/07/2021	Remota	27 estudantes	Desafio. Teste dos Labs Desafio. Mais Velha Desafio. Criptografia Desafio. Ordenação Desafio. Celebridade	Criação de orientações para coletar evidências do exercício do Pensamento Computacional.
2021/ERE-4	Aprofundar o conhecimento sobre as habilidades relacionadas com o Pensamento Computacional e como coletar evidências do seu desenvolvimento.	18/10/2021 a 10/11/2021	Remota	122 estudantes	Desafio. Teste dos Labs Desafio. Mais Velha Desafio. Criptografia Desafio. Ordenação Desafio. Celebridade	Aperfeiçoamento de todo o material desenvolvido.

A formalização dos desafios deveria ser feita em linguagem natural ou fluxograma. Como os desafios Lista de Presença e Ordenação são mais iniciais, a solução poderia ser representada em linguagem natural. Enquanto os desafios Busca e Celebridade por serem maiores e mais complexos deveriam ser representados com um fluxograma. Para a condução dos desafios Busca e Ordenação foi utilizado um conjunto de cartões físicos que representam personalidades da Computação. Esse material foi criado pela turma em uma atividade anterior da disciplina, que estava relacionada à história da Computação. As personas foram representadas na forma de cartão com as seguintes informações: foto, nome, ano de nascimento e falecimento (se for o caso), cidade e país de origem, formação acadêmica, instituição de atuação, área da Computação de maior atuação, principais feitos e contribuições para a Computação e curiosidades. A Figura 5.1 apresenta 4 exemplos de personas criadas pela turma de CI1003 em 2019/1.

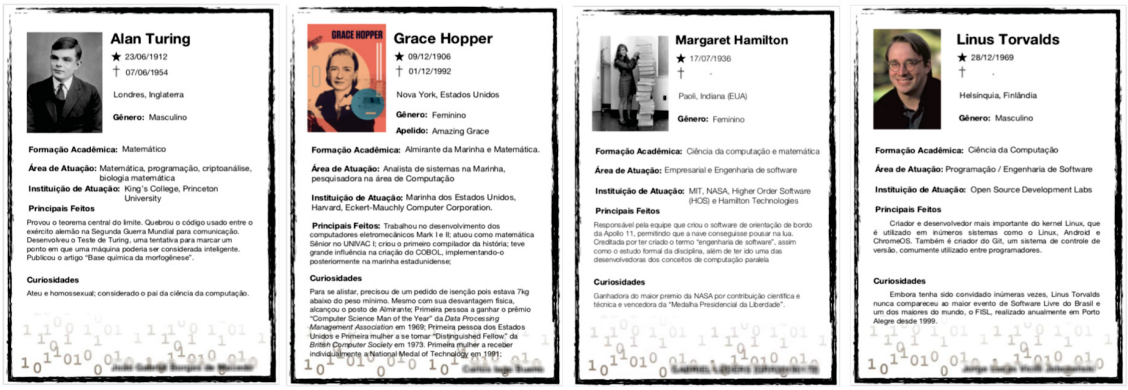


Figura 5.1: Personas criadas pelos estudantes de CI1003 2019/1

O estudo piloto teve como resultados a elaboração de quatro desafios, e aplicação e observação dos desafios em sala de aula. A observação dos desafios resultou em um conjunto de lições aprendidas sobre como conduzir atividades para o exercício do Pensamento Computacional em sala de aula, e um artigo publicado no SBIE 2019 (Oliveira et al., 2019).

5.2.1.1 Observações da Oferta

Da observação da condução dos desafios em sala de aula foram tiradas algumas lições aprendidas. *Lição 1. Deixar as aulas mais dinâmicas e participativas.* O primeiro desafio foi realizado no início do semestre, estudantes tinham apenas um mês de aula, se conheciam há pouco tempo, tinham vergonha de conversar com os colegas e expor seus apontamentos em sala. Com o passar do tempo, percebemos que os estudantes começaram a ganhar confiança e estavam participando mais das aulas e mais engajadas tanto na execução dos desafios quanto com os colegas no momento de criação da solução. Durante a socialização, percebemos que as discussões começaram a ser mais longas e profundas. Assim, a partir da nossa observação, aprendemos que as aulas precisam ser mais dinâmicas e participativas.

Lição 2. Formar grupos de até 6 pessoas. Como o *Desafio. Lista de Presença* era um problema real da disciplina, planejamos executar pelo menos duas soluções para observar a sua execução. Assim, planejamos mais rodadas de socialização em grupos e a formação dos grupos seria incremental. Na primeira rodada os estudantes criaram as suas soluções individualmente, depois em duplas, em grupos de 4 pessoas, 8 e 16 pessoas. A cada rodada de socialização as soluções eram discutidas e melhoradas, e a quantidade de soluções na turma diminuía. No fim, duas soluções seriam executadas em sala, para observar a execução e comparar o comportamento de cada uma delas. Observamos que o tamanho dos grupos influencia no engajamento dos

integrantes do grupo, à medida que o grupo aumentava menos estudantes participavam e uma quantidade pequena deles interagia. O engajamento individual dos integrantes de um grupo era proporcionalmente inverso ao tamanho do grupo: quanto maior o grupo menor era a interação entre os participantes. A partir da nossa observação, um tamanho bom para os grupos é de 6 integrantes. Com essa quantidade integrantes conseguem ouvir e ser ouvido por todos, com espaço para criar a solução final do grupo.

Lição 3. Cuidar a quantidade de entregas. Os desafios faziam parte da disciplina e eram contabilizados como parte da nota da disciplina. Para isso, os estudantes precisavam entregar as suas soluções via Moodle. Como os desafios possuíam duas partes, e quase sempre não era possível finalizá-los em sala, os estudantes precisavam investir tempo extraclasse tanto para representar a solução quanto para formalizar a entrega. Neste sentido, a lição aprendida foi com relação a quantidade de entregas e motivar os estudantes a aproveitarem o tempo em sala de aula. Percebemos que ao solicitarmos muitas entregas os estudantes ficam desmotivados para realizar ou formalizaram a entrega de qualquer maneira apenas para registrar que entregou.

Lição 4. Existe relação entre o nível de atenção com o local na sala. Notamos que alguns estudantes se distraíam com facilidade, principalmente os que sentavam mais ao fundo da sala. Isso pode ter ocorrido pelo fato de que a sala utilizada precisava ser muito grande e alguns deles ficavam muito distantes dos professores, mesmo que os professores estivessem constantemente circulando pela sala no momento da realização dos desafios. Observamos que existe uma relação do nível de atenção com o local escolhido em sala de aula, e que na maioria das vezes os estudantes escolhem o lugar na turma também baseado no seu nível de interesse.

Lição 5. Problemas reais e contextualizados promovem o engajamento e aprendizagem significativa. Os desafios foram pensados para aproximar os estudantes dos desafios. O desafio da lista de chamada foi um problema real enfrentado em sala, os desafios de ordenação e busca utilizaram como material as personas criadas pelos estudantes, e o desafio da celebridade trouxe a própria turma como contexto do problema (exemplo, uma festa dos calouros). Percebemos durante as aulas que utilizar problemas reais e contextualizados aumentou o engajamento dos estudantes, mais interessados com o processo de solução, revelando uma aprendizagem significativa. Percebemos durante as aulas que ao utilizar problemas reais e contextualizados aumentou o engajamento dos estudantes. No Desafio. Lista de Presença observamos durante as interações em sala que eles estavam motivados a criar uma solução que pudesse ajudar no "problema" que tínhamos em sala. O envolvimento e a motivação deles revela a importância de atividades contextualizadas para ter uma aprendizagem significativa.

Lição 6. Socialização ajuda a melhorar a solução. A segunda parte dos desafios era destinada à socialização da solução com colegas. A socialização das soluções garantia um espaço de tempo específico para que estudantes compartilhassem as suas soluções e refletissem sobre os problemas em sua solução. A socialização também ajudou os estudantes a perceberem aspectos com relação a representação da solução, como as soluções iriam ser executadas pelos colegas a solução precisavam estar clara e objetiva, sem duplo sentido ou ambiguidades. Reservar um espaço da atividade para que os estudantes avaliassem a solução de um colega e fornecesse um diagnóstico sobre pontos que precisava ser esclarecido ou corrigido fez com que os estudantes desenvolvessem um olhar crítico sobre o problema, a solução, e a relação problema solução. Além disso, auxiliou o exercício de habilidades como argumentação e comunicação, resultando também na autoavaliação da sua solução pelos estudantes.

Lição 7. Aumento no engajamento e melhor gestão do tempo em sala. No Desafio. Celebridade, o último desafio realizado em sala, foi possível perceber que os estudantes estavam mais engajados, mais familiarizados com o processo de identificar os requisitos do desafio e organizando melhores os próximos passos para criar a solução. Observamos também uma postura

mais madura dos estudantes, perguntando e comentando de maneira mais segura, e aproveitando melhor o tempo em sala. Com as atividades realizadas foi possível exercitar a habilidades de trabalho em equipe, utilizando de forma explícita as habilidades do Pensamento Computacional: dividindo o problema em problemas menores, mais fáceis de serem resolvidos; simplificando e identificando as reais necessidades à medida que as soluções eram desenvolvidas; negociando desde o momento de dividir as tarefas até ao mesclar as soluções ou defender as próprias ideias; e na construção da solução, trabalhando de maneira colaborativa para representar a solução.

5.2.1.2 Questionário de Feedback

No fim da disciplina foi aplicado um questionário online e anônimo para coletar a percepção dos estudantes sobre a disciplina, conteúdos, recursos e os desafios realizados. O questionário foi composto por 23 perguntas, sendo 11 delas perguntas descritivas. Responder o questionário era opcional e não estava atrelado a nenhuma nota na disciplina. Dos 91 estudantes concluintes da disciplina, 33 responderam o questionário (36,3% dos matriculados). Sobre os desafios realizados, adicionamos uma pergunta para avaliarem os desafios considerando a escala de 1 (detestei os desafios) a 5 (gostei muito dos desafios), 11 indicaram que gostaram dos desafios e 1 que gostou muito. Outras 9 pessoas sinalizaram que os desafios foram indiferentes, e 12 não gostaram dos desafios – conforme Figura 5.2. Essa avaliação dos desafios mostra um equilíbrio entre as respostas, sendo que 12 pessoas avaliaram que gostaram dos desafios, 12 que não gostaram dos desafios e 9 avaliaram eles como indiferentes (nem gostaram nem não gostaram).

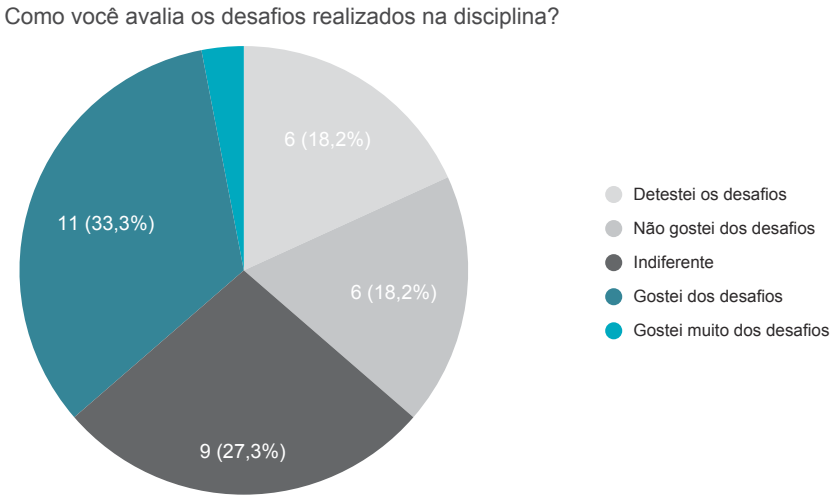


Figura 5.2: Percepção dos estudantes com relação aos desafios realizados na disciplina

O conteúdo da disciplina CI1003 possui uma relação direta com conteúdos da disciplina de CI1055 – Algoritmos e Estruturas de Dados (ALG 1), em que os conteúdos de uma apoiam o desenvolvimento do conteúdo da outra. Para investigar melhor essa relação, perguntamos a percepção dos estudantes sobre a relação dos desafios conduzidos e o aprendizado na disciplina CI1055. Os estudantes responderam que os desafios facilitaram (14 - 42,4%) e facilitaram muito (4 - 12,1%) o aprendizado em ALG 1. Entretanto, 15 estudantes afirmam que os desafios não facilitaram e nem dificultaram em nada – conforme Figura 5.3.

Na sequência deixamos um espaço para que as respostas fossem justificadas, recebemos 23 respostas e a sumarização das respostas pode ser encontrada na Tabela 5.3. Das 15 pessoas que avaliaram que realizar os desafios não facilitaram e nem dificultaram o aprendizado em Alg

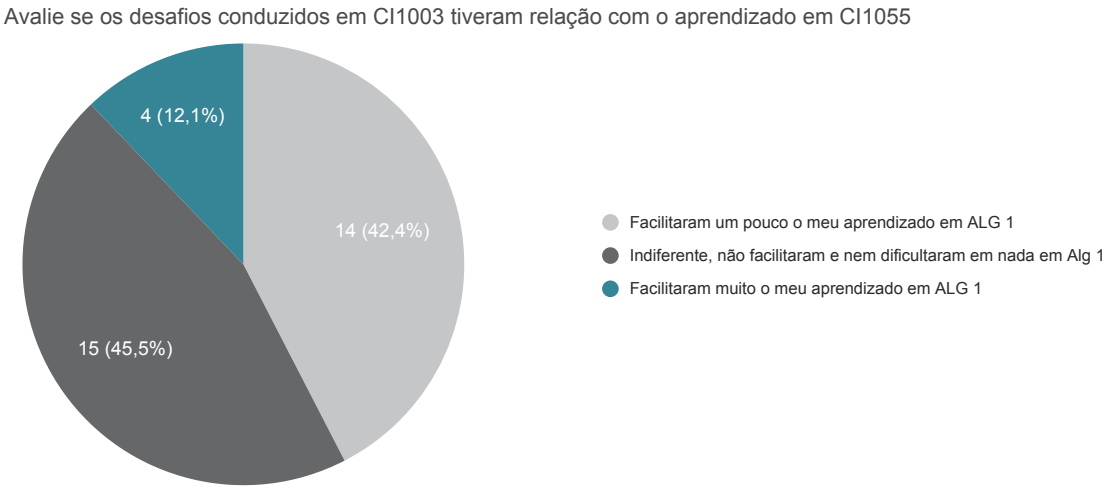


Figura 5.3: Percepção dos estudantes com relação aos desafios ao seu aprendizado de CI1055

1, 6 não explicaram o porquê, 5 reforçaram que os desafios não tiveram nenhuma relação ao aprendizado em CI1055 e 4 sinalizaram que tinham conhecimento prévio dos conteúdos abordados. Considerando que as duas disciplinas não estavam 100% sincronizadas, o conhecimento prévio pode ser por dois diferentes motivos: (i) a pessoa cursou um ensino médio técnico ou algum curso sobre o tema; ou (ii) quando os desafios foram conduzidos atividades com o mesmo tema havia sido conduzido na disciplina CI1055. O fato dessas pessoas possuírem algum tipo de conhecimento prévio sobre algoritmos revela como os desafios podem ser considerados por eles muito triviais em um primeiro momento, e muitas vezes ainda mais usual por não requerer a formalização da solução em uma linguagem de programação.

Tabela 5.3: Temas das respostas sobre percepção dos estudantes com relação dos desafios ao seu aprendizado de CI1055

Por que os desafios facilitaram ou não o aprendizado em ALG 1? *(n = 23)*	
Tema	Nº de Respostas
Não facilitaram e nem dificultaram em nada em Alg 1	10
Tinha conhecimento prévio	4
Facilitou	16
Compreensão de exercícios de programação em Alg 1	6
Oportunidade de visualizar os problemas com mais calma e tempo	4
Noção de vetores e auxiliou a ficar mais atento aos problemas imperceptíveis	3
Esquematização do Pensamento Computacional	1
Importância de abstrair e criar uma sequência de passos para resolver o problema	1
Fluxograma ajudou a entendimento de Alg 1	1

Além disso, uma pessoa que avaliou que os desafios facilitaram um pouco o aprendizado em Alg 1 sinalizou em sua resposta discursiva que os desafios não facilitaram e nem dificultaram

em nada em Alg 1. Do restante, 13 pessoas sinalizaram que os desafios facilitaram o aprendizado, e os principais motivos foram desde a compreensão de exercícios de programação até a oportunidade de visualizar os problemas com mais calma e tempo. Ao fornecer oportunidades de visualizar os problemas com mais calma e tempo conseguimos fomentar a consciência da importância de abstrair e ter um bom conhecimento sobre o problema para então criar uma sequência de passos para resolvê-lo. Além disso, observamos a importância de destinar um tempo para dedicação do entendimento do problema para propor uma solução mais próxima da melhor solução.

Quando perguntamos aos estudantes com relação à opinião deles sobre realizar atividades em grupo, 29 responderam que gostam de realizar atividades em grupo e 10 deles afirmam que aprendem melhor com atividades em grupos. Apenas uma pessoa reconhece que mesmo que ele goste de realizar atividades em grupo ele aprende melhor em atividades individuais. Outras 18 pessoas não percebem diferença no aprendizado se a atividade for em grupo ou individual. Figura 5.4 e 5.5 sumarizam as respostas desta pergunta. Na sequência deixamos uma pergunta discursiva sobre os motivos de gostar ou não de realizar atividades em grupo – Tabelas 5.4 e 5.5. Recebemos 23 respostas, 11 pessoas mencionaram o exercício de habilidades como comunicação, empatia e trabalho em equipe. Exposição a outros pensamentos e formas de resolver problemas (5), interação com outras pessoas (5), transformação em atividades mais divertidas e motivadoras (4) e aprendizagem coletiva (2) foram elencados pelos estudantes como principais motivos para gostar de realizar atividades em grupo.

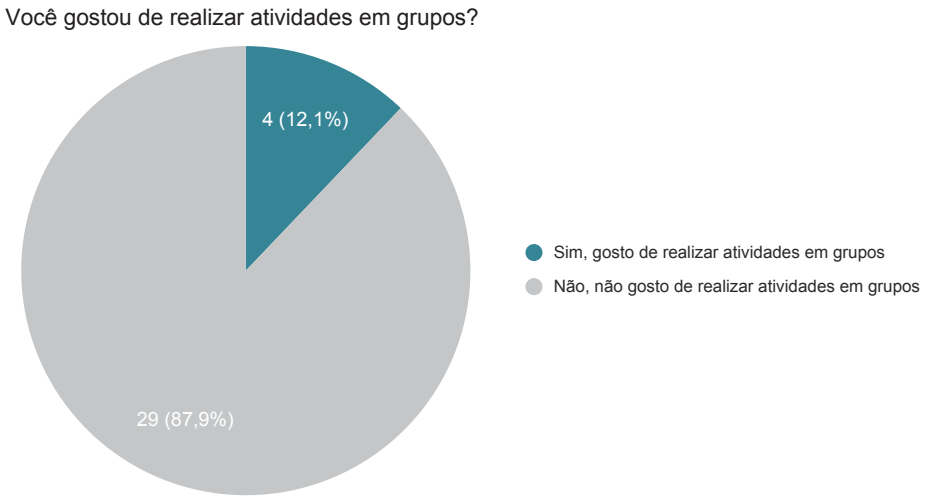


Figura 5.4: Percepção dos estudantes com relação a realização de atividades em grupos

Apenas 3 pessoas compartilharam porque não gostaram de realizar atividades em grupo, o principal motivo mencionado nas respostas foi o fato de que não estão acostumados a interagir com outras pessoas (2). Essa opinião sobre não gostar das atividades em grupo por não ter costume de interagir com outras pessoas pode ser um motivo que surgiu por serem estudantes do primeiro semestre do curso, e naquele momento a turma ainda não tinha proximidade e não estavam acostumadas a interagir entre si. Assim, as respostas com relação aos motivos que gostaram de realizar atividades em grupo da luzes a importância de existir um momento específico durante as práticas para socialização, além de deixar a atividade mais dinâmica e motivadoras auxilia no exercício de habilidades como trabalho em equipe, a aprendizagem se torna colaborativa e favorece o sentimento de união e parceria entre as pessoas envolvidas.

Para entender melhor a percepção dos estudantes com relação aos desafios, foram analisadas as respostas de 4 perguntas abertas e não obrigatórias que foram destinadas para

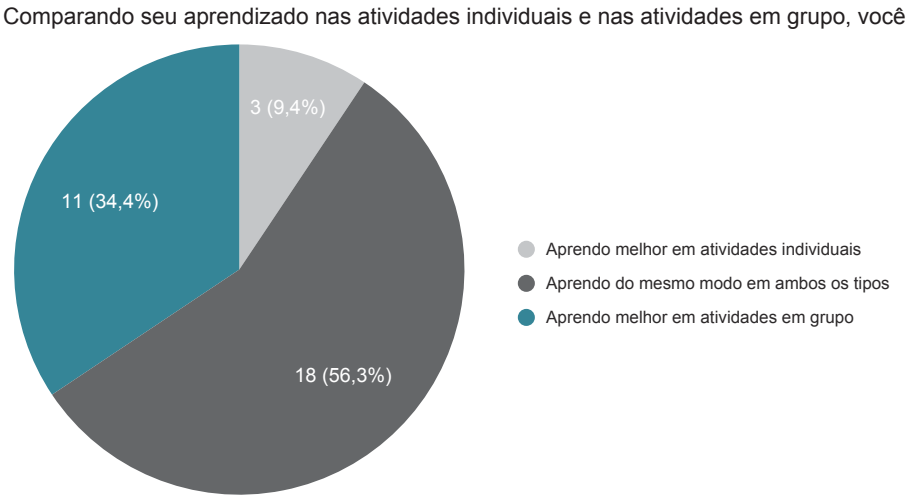


Figura 5.5: Percepção dos estudantes com relação às atividades em grupo e individual

Tabela 5.4: Temas das respostas sobre motivos que gosta de realizar atividades em grupo

Por quais motivos você *gostou* de realizar atividades em grupo? *(n = 23)*	
Tema	Nº de Respostas
Exercita habilidades como comunicação, parceria, empatia	6
Exercitar a habilidade de trabalho em equipe	5
Exposição a outros pensamentos e forma de resolver problemas	5
Interagir com outras pessoas	5
Atividades ficavam mais divertidas e motivadoras	4
Aprendizagem coletiva	2
Gostos pessoais	1

Tabela 5.5: Temas das respostas sobre motivos que não gosta de realizar atividades em grupo

Por quais motivos você *não gostou* de realizar atividades em grupo? *(n = 3)*	
Tema	Nº de Respostas
Não tem costume de interagir com outras pessoas	2
Atividades muito repetitivas	1
Dificuldade para desenvolver atividades criativas em grupo	1

fornecerem a opinião sobre: (i) o que faltou na disciplina; (ii) quais aspectos mudaria na disciplina; (iii) quais aspectos que mais gostou na disciplina; e (iv) comentários gerais. Ao todo, foram recebidas 99 respostas para as 4 perguntas, sendo que 31 pessoas responderam pelo menos uma das perguntas. As respostas foram sumarizadas, e as 12 respostas que mencionaram os desafios foram compiladas e organizadas em 3 grupos de resposta: (i) o que faltou na disciplina;

(ii) quais aspectos mudaria na disciplina; e (iii) quais aspectos que mais gostou na disciplina. A sumarização das respostas e dos temas gerados podem ser vistos na Tabela 5.6.

Tabela 5.6: Sumarização das respostas das perguntas abertas

O que faltou na disciplina	**Quais aspectos mudaria na disciplina**	**Quais aspectos que mais gostou na disciplina**
Trabalhar de forma mais aprofundada os desafios	Definição melhor dos prazos de entrega dos desafios (3)	Os desafios propostos (2)
Espaço para discutir mais as soluções dos desafios	Propor os desafios para utilizar linguagem de programação	O estímulo do pensamento computacional
	Fornecer mais exercícios sobre Pensamento Computacional	Os desafios forneceram uma base para conceitos computacionais
	Ajustar o ritmo dos desafios para que eles não sejam tão longos	
	Adotar as soluções dos desafios em sala de aula	
	Ajustar quando apresentar os conteúdos para terem melhor sincronia com Alg 1	

As respostas sinalizaram que faltou na disciplina trabalhar de forma mais aprofundada os desafios e espaço para discutir com profundidade as soluções. Com relação aos aspectos que mudariam na disciplina, foram mencionados por três pessoas que era necessário melhorar os prazos das entregas dos desafios, isso corrobora com a quantidade de entregas sinalizada pela Lição 3. Além disso, foi apontada a necessidade de ajustar o ritmo dos desafios para que eles não sejam tão longos, e ajustar quando apresentar os desafios para terem melhor sincronia com a disciplina de Alg 1. Uma pessoa mencionou que mudariam os desafios para que eles passassem a utilizar uma linguagem de programação para formalizar a solução. Essa mesma pessoa avaliou que não gostou dos desafios e que eles foram indiferentes para o aprendizado em CI1055. Em contrapartida, uma pessoa sinalizou que forneceria mais exercícios sobre Pensamento Computacional, essa mesma pessoa avaliou que os desafios facilitaram muito o aprendizado dela em Alg 1.

Com relação aos aspectos que mais gostaram na disciplina, quatro respostas pontuaram os desafios, sendo que uma reforça que mais gostou do estímulo do Pensamento Computacional e outra que os desafios forneceram uma base para conceitos computacionais. Por serem perguntas que eram abrangentes a toda a disciplina, e que não direcionavam para os desafios, entendemos que existe uma significância ainda maior nessas respostas e que os desafios foram capazes de despertar o interesse dos estudantes e estimular o Pensamento Computacional. A percepção sobre como os desafios forneceram uma base para conceitos computacionais reforça a importância de realizar os desafios com estudantes do primeiro período do curso de graduação em Computação, pois, esse conhecimento base apoiará o aprendizado ao longo de toda a formação.

5.2.2 Desafios 2.0

Este estudo de caso foi realizado na oferta da disciplina CI1003 de 2019/2, no período entre agosto e dezembro de 2019. Os encontros da disciplina nesta oferta ocorreram presencialmente, uma vez por semana com duração de 4 horas cada. Nesta oferta, eram 11 estudantes matriculados.

A disciplina foi acompanhada por um docente e pela autora da pesquisa. O objetivo deste estudo era refinar os desafios existentes e criar novos desafios que estimulem o raciocínio lógico e o exercício do Pensamento Computacional dos estudantes. Nesta oferta foram elaboradas três novas práticas educacionais: Desafio. Mais Velha, Desafio. Ordenação e Desafio Criptografia.

Os desafios foram conduzidos em sala de aula, e foram divididos em duas partes. Durante a primeira parte do desafio, os estudantes precisavam propor individualmente uma solução e a segunda parte foi destinada para socialização e amadurecimento da solução. Como a turma era pequena a socialização foi realizada de maneira conjunta, com toda a sala, e anotações iam sendo feitas utilizando um quadro e giz. Enquanto os estudantes apresentavam suas soluções e argumentavam sobre as soluções dos colegas, o docente e a colaboradora mediavam a conversa, questionavam os estudantes sobre suas soluções e representavam as soluções no quadro. Essa adaptação da socialização permitiu que todos os presentes conseguissem interagir mais, tirando dúvidas, refletindo sobre as outras soluções e no fim esboçar uma solução conjunta para o desafio.

5.2.2.1 Observação da Oferta

Da observação da condução dos desafios em sala de aula foram tiradas algumas lições aprendidas. *Lição 8. O tamanho da turma impacta diretamente na interação.* Como esta turma era pequena, ocorreu uma menor interação em sala de aula, apenas 4 estudantes frequentavam as aulas e eles não interagiam quase nada. Por outro lado, eles tinham maior liberdade para perguntar e interagir durante o momento destinado à resolução do problema. Quando eles estavam motivados e engajados conseguimos realizar discussões acerca das dúvidas apresentada por eles, corrigir e remediar os erros com maior facilidade ao longo da resolução e não apenas no fim de cada ciclo.

Lição 9. É difícil promover o engajamento em turmas pequenas. Estudantes matriculados nesta oferta eram de turmas diferentes, e o contato que eles tinham umas com os outros ocorriam uma vez por semana durante as aulas da disciplina. Como as aulas ficaram concentradas em apenas um dia da semana, eles se encontravam presencialmente a cada semana. Caso um estudante faltasse eles ficavam pelo menos 2 semanas sem contato. O pouco contato entre os estudantes dificultava a construção de um relacionamento e uma convivência com a turma. O engajamento entre os estudantes pode impactar na segurança de cada um, principalmente para eles compartilharem suas dúvidas, opiniões e sugestões. Por ser uma turma pequena era necessário realizar mais questionamentos durante a resolução dos desafios, e nem sempre eles respondiam, foi difícil obter retorno dos estudantes presentes.

Lição 10. Observar o comportamento da solução ajuda a evolui-la. No dia que o desafio ordenando a biblioteca foi realizado levamos para a sala de aula alguns livros físicos. A intenção era que os livros fossem utilizados para testar as soluções com o próprio objeto do desafio, e assim observar o comportamento de suas soluções. Ao conduzir o teste em sala de aula, os estudantes percebiam que algumas de suas instruções não estavam tão claras e objetivas, favorecendo a correção logo no início da etapa de solução do desafio. Além disso, o professor demonstrou algumas estratégias de ordenação em sala e conduziu uma conversa sobre o tema.

Lição 11. É importante desassociar o Pensamento Computacional de linguagens de programação. Alguns estudantes da turma tinham conhecimento prévio sobre linguagens de programação e sabiam programar. Durante a condução dos desafios percebemos que as dúvidas desses estudantes estavam relacionadas com a linguagem de programação, como representar a solução, como manipular dados nesta linguagem, ou como realizar um determinado comando. Percebemos que os estudantes estavam muito mais preocupados em como representar a solução com uma linguagem de programação do que realmente pensar em como criar uma solução para o problema. Quando isso acontecia, direcionávamos o estudante a pensar mais sobre a solução em si. Entretanto, esses estudantes apresentavam uma resistência, e não compreendiam como

essas dúvidas seriam secundárias. Portanto, é necessário apoiar o estudante a pensar em um primeiro momento em estratégias para solucionar o problema e posteriormente em representar essa solução, para não correr o risco de esbarrar em problemas e limitações da linguagem.

5.2.3 Estudos de Caso Remoto

Os estudos de caso remoto foram realizados com atividades síncronas e assíncronas, em quatro ofertas remotas diferente da disciplina CI1003, foram elas:

CI1003 2020/Especial – entre agosto e setembro de 2020;

CI1003 2020/ERE-2 – entre novembro e dezembro de 2020;

CI1003 2021/ERE-3 – entre maio e julho de 2021;

CI1003 2021/ERE-4 – entre outubro e novembro de 2021.

Ao todo, nas quatro ofertas, 249 estudantes estavam matriculados na disciplina e todos eram ingressantes dos cursos. A disciplina foi acompanhada por três docentes e sete colaboradores pós-graduandos. O objetivo do estudo de caso foi observar o comportamento dos desafios em quatro ofertas remotas da disciplina, identificar as potencialidades e fragilidades da abordagem Pense Computacionalmente, e refinar todo material existente. Para isso, em um primeiro momento, foi trabalhado em cima de como adaptar todo o material para o formato digital e de maneira totalmente textual via Moodle. Além disso, ocorreu a criação de um novo desafio (Desafio. Criptografia) e a incorporação do Desafio. Teste dos Labs.

Nas ofertas remotas foram conduzidos 5 desafios: Desafio. Mais Velha, Desafio. Teste dos Labs, Desafio. Criptografia, Desafio. Ordenação, e Desafio. Celebridade. Para conduzir os desafios na modalidade remota precisamos realizar algumas adaptações, principalmente com relação à etapa de socialização. A socialização dos desafios ocorre ao disponibilizar as soluções no Moodle de maneira que qualquer integrante possa consultar outras soluções. Mesmo que as soluções estejam disponibilizadas, os estudantes foram incentivados a realizar uma versão da sua solução antes de verificar as soluções dos colegas. Assim, dando destaque ao processo de pensar sobre o problema e propor uma solução, e não apenas atingir a "resposta correta". Principalmente, porque as avaliações dos desafios não focam em avaliar se um estudante resolveu corretamente ou não, mas sim se ele propôs uma solução que atenda o que foi pedido no enunciado do desafio.

5.2.3.1 Questionário de Feedback

Ao final de cada oferta da disciplina foi disponibilizado um questionário de feedback online e anônimo para coletar a percepção dos estudantes. O questionário foi composto por 21 questões, sendo 6 delas abertas, e 5 das questões foram para coletar a percepção dos estudantes sobre os desafios conduzidos. O questionário revela a percepção dos estudantes sobre a disciplina CI1003 como um todo, e também sobre a realização dos desafios. Responder o questionário não era um requisito obrigatório, respondia quem tinha mesmo a intenção de fornecer os comentários.

Das 249 pessoas matriculadas na disciplina, 181 estudantes finalizaram a disciplina com aprovação, e desses 68 (37,6%) responderam o questionário de *feedback*. Das 68 respostas, identificamos uma pessoa que respondeu tudo no extremo negativo e, infelizmente, não forneceu informações sobre o motivo da sua frustração. Portanto, julgamos essa resposta como um *outlier* e consideramos 67 respostas válidas (37% dos estudantes concluintes). A idade média dos estudantes foi de 20 anos, variando entre 17 e 33 anos de idade, utilizaram computadores desktop,

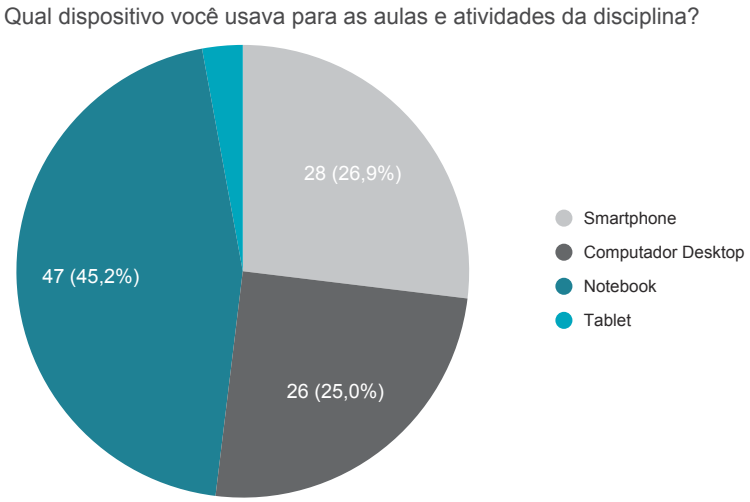


Figura 5.6: Dispositivos utilizados para acompanhar as aulas e atividades da disciplina CI1003

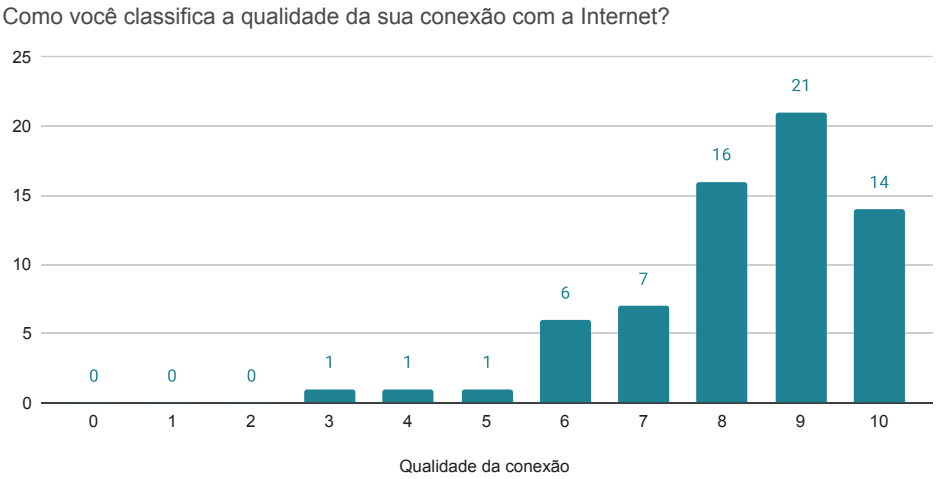


Figura 5.7: Percepção dos estudantes com relação a qualidade da internet deles

notebooks, smartphones e tablets para acessar a disciplina e julgam que as suas conexões de internet eram boas — as informações estão ilustradas nas Figuras 5.6 e 5.7

Para avaliar os desafios, perguntamos no questionário a percepção dos estudantes sobre os desafios considerando 4 aspectos: i) o nível de dificuldades, ii) a qualidade da explicação, iii) a capacidade de provocar o raciocínio, e iv) a capacidade de promover o interesse. Os quatro elementos foram avaliados com uma escala de 5 pontos que foi de muito ruim a muito bom. De maneira geral, os quatro aspectos foram avaliados com uma tendência de percepção positiva – mesmo se tratando de aspectos subjetivos de avaliação. A Figura 5.8 ilustra como os desafios foram avaliados em cada um dos aspectos, e as respostas nos revelam que os desafios foram muito bons em provocar o raciocínio e bom na capacidade de promover o interesse dos estudantes. Mostrando que os desafios foram capazes de exercitar o Pensamento Computacional.

Mesmo que bem avaliados, é possível perceber que ainda existem espaços para ajustes nos desafios, tanto com relação ao nível de dificuldades quanto na qualidade das explicações. O nível de dificuldade é sempre um fator que pode impactar os outros aspectos, principalmente com relação a capacidade de promover o interesse, uma vez que desafios muito fáceis podem ser

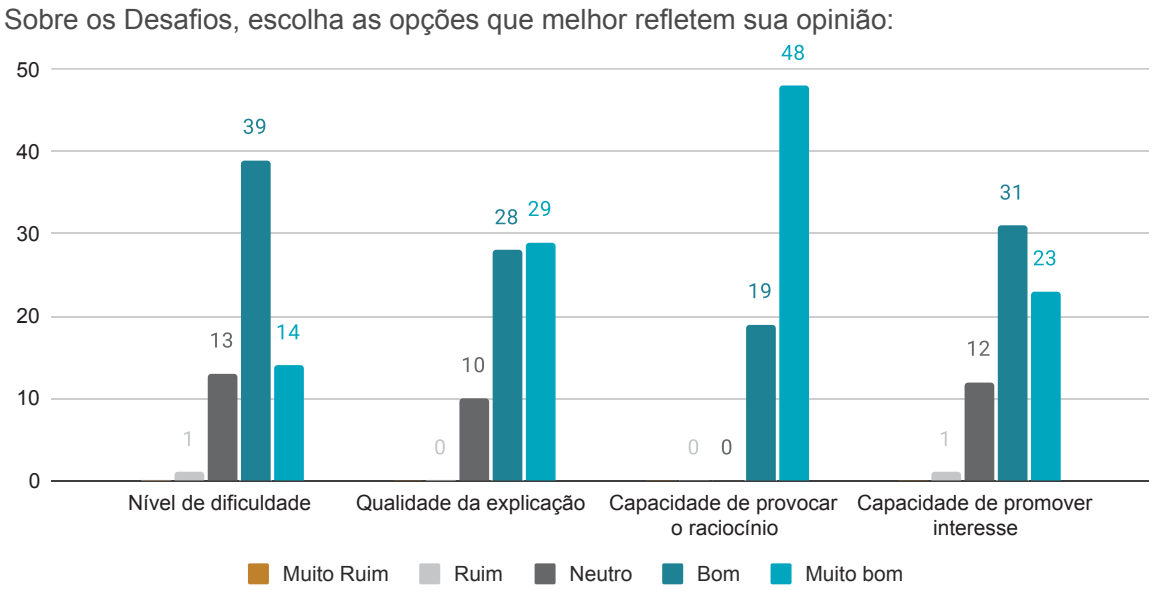


Figura 5.8: Percepção dos estudantes sobre os desafios considerando nível de dificuldade, qualidade da explicação, capacidade de provocar o raciocínio e capacidade de promover o interesse

entediantes e o contrário pode gerar desmotivação. Neste caso, ambos podem gerar o abandono do desafio proposto e não promover o interesse. Além disso, o nível de dificuldade pode impactar diretamente na percepção dos estudantes com relação a capacidade de provocar o raciocínio.

A explicação dos desafios foi disponibilizada de maneira textual no Moodle, e instruções foram fornecidas no encontro síncrono. O fato da explicação ser apenas textual pode ter impactado na opinião quanto à qualidade da explicação e nível de dificuldade. Escrita e interpretação de texto são capacidades humanas e subjetivas, e infelizmente, raramente conseguimos produzir um texto perfeito e que seja entendido por todos. Por outro lado, essas duas habilidades são importantes de serem desenvolvidas quando pensamos no perfil do profissional da área. Ainda com relação à interpretação textual, reparamos que os estudantes não entregavam tudo que o desafio pedia. Cogitamos que os estudantes estão com problemas de interpretação de texto, e que isso pode ser um problema dessa geração que está acostumada a consumir textos cursos.

Para entender quais aspectos dos desafios poderiam ser melhorados pedimos para os estudantes pontuarem o que eles tinham gostado mais e menos nos desafios. Quatro respostas foram desconsideradas porque não tinham informações suficientes, e não foi possível entender se eram pontos que a pessoa tinha gostado ou não nos desafios. Consideramos 28 respostas, e os temas que surgiram dessa pergunta podem ser vistos na Tabela 5.7. As respostas corroboram com a percepção dos estudantes sobre os desafios considerando os 4 aspectos, no qual eles possuem consciência que com os desafios estão exercitando o raciocínio e o Pensamento Computacional.

Os estudantes gostaram dos desafios tanto de maneira geral (4) quanto ao fato de que os desafios incentivaram testar diferentes caminhos para solução (3) e até a forma como os desafios foram conduzidos (3). De acordo com um estudante os desafios estavam *"balanceados, e capazes de realmente promover o pensamento lógico e instigar os estudantes a encontrarem suas soluções"*. Além disso, os desafios foram considerados interessantes pois *"fizeram pensar fora da caixa"*. Quando a pessoa afirmou que o que ela mais gostou *"foi sair formulando caminhos que poderiam ser a resposta e vendo que eu estava errado"* e a outra menciona que *"não sentia medo de fazer os desafios"* entendemos que incentivar os estudantes a focarem na

solução indiferente da resposta final contribui para o desenvolvimento da segurança, aceitação e entendimento ao erro, e aumento da motivação para realização dos desafios.

Tabela 5.7: Temas das respostas sobre Percepção dos estudantes com relação dos desafios ao seu aprendizado de CI1055

Pontue o que você mais gostou e o que menos gostou nos desafios *(n = 28)*	
Tema	Nº de Respostas
Gostou	37
Exercitar o raciocínio	7
Desafios de maneira geral	4
Incentivou a testar diferentes caminhos	3
Como os desafios foram conduzidos	3
Possibilidade de ver as soluções dos colegas	2
Socialização dos desafios	2
Desafio 1.2. Teste dos Labs	2
Desafio 2. Criptografia	2
Desafio 4. Celebridade	2
Desafio 3. Ordenação	1
Desafios de Lógica	1
Desafios mais detalhados	1
Espaço para errar e aprender	1
Exercitar as habilidades de programação	1
Exercitar o Pensamento Computacional de maneira desplugada	1
Forma como o problema foi abordado	1
Criação de Fluxograma	1
Tema dos desafios	1
Apoiou aprender programação	1
Não Gostou	14
Desafio 1.1. O mais velho tem olhos azuis	3
Pouco tempo para resolver os desafios	3
Representar a solução em fluxograma	1
Desafio Raciocínio e Criptografia	1
Como foram conduzidos remotamente	1
Desafio 3. Ordenação	1
Explicação do desafio	1
Desafios pouco detalhado	1
Ordem de apresentação dos desafios	1
Desafio 1.2. Teste dos Labs	1

Uma das pessoas que avaliou como neutra a capacidade dos desafios de promover o interesse citou que *"todos os desafios abordaram áreas diferentes do raciocínio, porém na*

situação atual de período especial não se mostraram uma atividade muito atraente vide outras matérias bem apressadas ocupando a maior parte do tempo do aluno". Esse comentário revela que mesmo que os desafios tenham promovido raciocínio e o desenvolvimento das habilidades do Pensamento Computacional, ainda existe espaço para melhorar o ritmo em que os desafios são apresentados e conduzido em aulas neste contexto remoto. Um estudante afirmou que *"um dos problemas não teve a explicação completa tão clara, mas isso foi resolvido em uma das aulas síncronas antes do encerramento da atividade"*, essa afirmação dá luzes à importância dos encontros síncronos para remediação de erros, dúvidas e guiar os estudantes durante a resolução do problema. Ainda precisamos deixar mais explícito durante as aulas os motivos das atividades serem do jeito que são e estarem sendo conduzidos dessa forma. Além de reforçar os benefícios de estarmos utilizando linguagem natural e fluxograma para representar a solução. Ter conhecimento sobre os motivos ajuda a desenvolver nos estudantes uma consciência sobre o seu aprendizado, e somando para uma aprendizagem mais significativa.

Entre os principais pontos que os estudantes relataram que não gostaram estão: um desafio específico (6) e o pouco tempo para resolver os desafios (3). Um estudante mencionou que não gostou de alguns desafios pois *"não eram exercícios muito algorítmicos, era mais uma charada."*, essa resposta indica como os estudantes possuem uma visão de que as atividades precisam estar relacionadas com a resolução de um problema computacional, ou que remeta ao computador. Em contrapartida, representar a solução em fluxograma foi mencionada também como algo que um outro estudante não gostou pois *"criar um fluxograma, escrever um passo-a-passo me lembra muito mais de programação"*. Essa resposta revela uma fragilidade de utilizar o fluxograma como forma de representar as soluções, pois ele é muito associado a programação o que pode se transformar em uma barreira para os estudantes.

Ao mesmo tempo que um fluxograma ajuda representar a solução de uma maneira mais organizada e padronizada, para isso conta com os recursos dos elementos gráficos, ele pode somar na camada de complexidade para os estudantes ingressantes que precisam se preocupar com quais aspectos gráficos eles devem utilizar para cada tipo de instrução. Mesmo que o fluxograma adicione uma complexidade, as representações visuais podem ser utilizadas para explicar e ilustrar conceitos como fluxo de execução, repetição, e tomadas de decisões (condicionais). Além disso, o fluxograma pode ser uma alternativa intermediária entre representar as soluções em linguagem natural e linguagem de programação.

Pedimos para os estudantes avaliarem o quanto eles achavam que tinham exercitado as habilidades do Pensamento Computacional com os desafios propostos, em uma escala de 1 (não exercitou nada) a 10 (exercitou muito), 67 (100%) respostas avaliam que acreditam ter exercitado acima da média (nota acima de 6). Quase metade dos estudantes (43,3%) acreditam que exercitaram muito o Pensamento Computacional com os desafios propostos. A Figura 5.9, sumariza a percepção dos estudantes com relação a quanto eles avaliam que exercitaram as habilidades do Pensamento Computacional com os desafios propostos. Esse resultado nos mostra que os estudantes desenvolveram uma consciência sobre o exercício das habilidades com relação aos desafios conduzidos. Por se tratar de estudantes ingressantes, desenvolver um entendimento sobre o exercício do Pensamento Computacional é importante pois ajudará a estabelecerem relação com o que eles sabem e como utilizar essas habilidades. Além disso, desenvolver a percepção do próprio aprendizado pode influenciar na motivação e engajamento.

Com relação às habilidades do Pensamento Computacional, foi pedido para os estudantes selecionarem as habilidades que eles julgam ter exercitado em cada um dos desafios realizados. As habilidades foram representadas em termos dos quatro pilares do Pensamento Computacional: abstração, decomposição, reconhecimento de padrões e algoritmos. A Figura 5.10 sumariza a percepção dos estudantes para cada desafio. Essa questão foi utilizada para verificar se

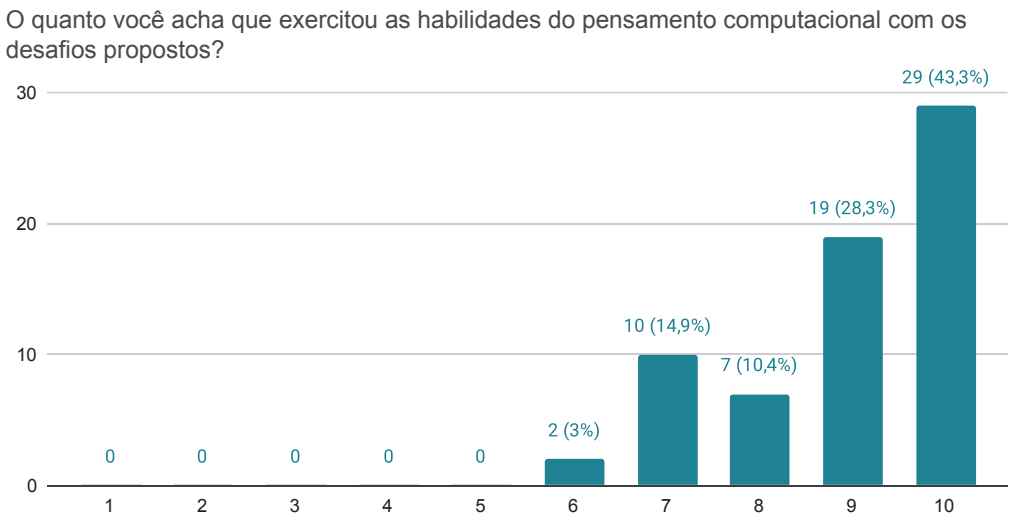


Figura 5.9: Percepção dos estudantes sobre o exercício das habilidades do Pensamento Computacional

os estudantes reconhecem e sabem diferenciar essas habilidades e conseguem identificar o exercício dessas habilidades no conjunto de desafios. As respostas mostram que os estudantes desenvolveram um entendimento sobre o Pensamento Computacional e suas habilidades, pois eles conseguem elencar quais habilidades estavam sendo mais exercitadas em cada um dos desafios. Mesmo que esse resultado não nos permita dizer o quanto foi o entendimento sobre o Pensamento Computacional ele nos revela indícios de que os estudantes conseguem diferenciar as habilidades entre si e verificar que cada desafio tem foco de desenvolver em um conjunto de habilidades específicas. Desenvolver a consciência sobre o que está sendo exercitado em cada desafio é importante para envolver os estudantes no processo de construção do seu conhecimento. É necessário que os estudantes tenham conhecimento sobre o que eles estão exercitando, pois existe uma chance maior deles evoluírem para um entendimento quando eles conseguem visualizar o conhecimento e seu desenvolvido, do que quando eles não estão enxergando.

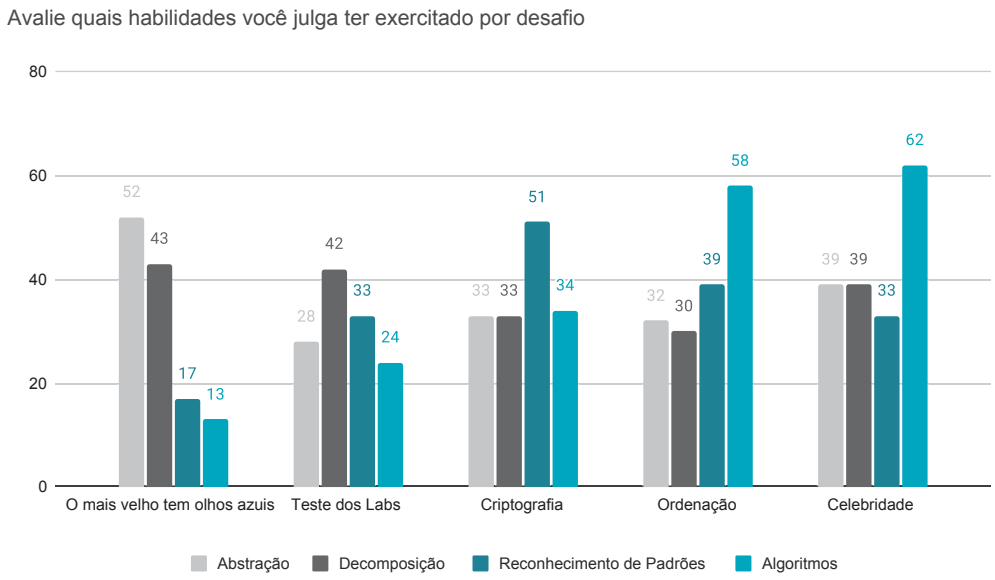


Figura 5.10: Percepção dos estudantes sobre as habilidades exercitadas em cada um dos desafios

Na abordagem Pense Computacionalmente, uma das etapas da resolução de um desafio é a socialização da solução. Nesta etapa os estudantes trocam a solução entre si com o objetivo de executar, avaliar e fornecer comentários e sugestões para o colega melhorar a sua solução. Na versão remota, a socialização ocorreu ao deixar todas as soluções disponíveis no Moodle. Cada estudante formalizou a entrega da solução criando um tópico no Moodle e adicionando a sua solução neste tópico. Esses tópicos eram livres e abertos, todos os estudantes tinham acesso e poderiam consultar, comentar e fazer perguntas. Para entendermos a percepção dos estudantes quanto a esse processo, perguntamos para eles se foi útil ou não ter acesso às soluções dos colegas. Foi unânime a percepção dos estudantes sobre ser útil ter acesso à solução dos colegas, pois ajudou a visualizar o problema sobre uma outra perspectiva e a aprender a partir de uma outra solução para melhorar a sua própria solução. A utilidade percebida também estava relacionada com a otimização das soluções, e foi citado que *"...após passar bastante tempo refletindo sobre um problema e sua solução, a mente pode ficar fechada e perder alguns detalhes. Nesse sentido, ter contato com outras soluções ajuda a perceber melhor possíveis fraquezas do que desenvolvemos."*. A sumarização das respostas está detalhada na Tabela 5.8.

Tabela 5.8: Temas das respostas sobre percepção dos estudantes com relação dos desafios ao seu aprendizado de CI1055

Olhar para as soluções que estavam postadas no Moodle foi útil para você analisar, corrigir e amadurecer a sua solução? Comente *(n = 33)*	
Tema	Nº de Respostas
Sim	33
Visualizar o problema por outra perspectiva	9
Aprender como otimizar a solução	5
Comparar soluções	5
Tirar dúvidas	3
Amadurecer a resposta	2
Aprender novos métodos	2
Exercitar o pensamento crítico	1
Pensar em outras formas de resolver o problema	1
Não	0

No fim do questionário, os estudantes tinham um espaço para fornecer *feedback* qualitativos considerando todos os aspectos da disciplina e das atividades conduzidas. Na pergunta com relação às dificuldades enfrentadas na disciplina em formato remoto, 3 respostas de 50 mencionaram diretamente os desafios. As respostas sinalizaram que as dificuldades estavam relacionadas aos desafios utilizarem uma metodologia nova e ter diferentes formas para chegar em uma solução. O ponto de ter diferentes formas de resolver um problema foi constantemente abordado pelos estudantes durante as aulas síncronas, assim, conduzimos as interações sempre para focar em como diminuir o número de possibilidades de resposta e sempre buscando a otimização da solução. Entre as 59 respostas recebidas sobre o que mais gostou na disciplina, 13 mencionaram os desafios e o exercício do Pensamento Computacional. Quando perguntamos o que menos gostaram na disciplina, das 48 respostas apenas 4 mencionaram os desafios, e 3 especificam um desafio e outra resposta menciona que não gostou de precisar analisar uma solução e fornecer feedback. Uma outra resposta lamentou que *"o bloco 2 foi muito curto"*.

Essas perguntas sobre a disciplina foram consideradas justamente porque gostaríamos de verificar se os estudantes mencionaram os desafios. As respostas são consideradas ainda mais importantes porque mesmo não sendo focada nos desafios, os estudantes sinalizaram os desafios como sendo o ponto que eles mais gostaram na disciplina. Isso revela o impacto que os desafios tiveram sobre os estudantes e na disciplina, corroborando com o avanço do exercício do Pensamento Computacional e os objetivos da disciplina.

Com as observações realizadas nos estudos exploratórios é possível afirmar que os desafios realizados auxiliam no exercício das habilidades do Pensamento Computacional. As lições aprendidas revelam que exercitar o Pensamento Computacional de maneira desplugada traz diferentes benefícios para estudantes de Computação, indicando que eles precisam desenvolver desde o primeiro momento as habilidades do pensar computacionalmente. Ainda, a estratégia desplugada contribuiu para facilitar o entendimento de conteúdos tão abstratos.

Com os estudos realizados foi possível perceber a evolução dos estudantes entre o primeiro e o último desafio realizado, principalmente na postura de cada um deles durante o trabalho em equipes e a responsabilidade com o tempo em sala destinado às tarefas. Ainda foi reconhecido que as soluções possuíam um amadurecimento significativo entre as entregas. As lições aprendidas demonstram positivamente que o método aplicado auxilia o estudante a desenvolver o hábito de pensar computacionalmente durante a resolução dos problemas. Em relação às dificuldades durante o processo de solução dos desafios, reconhecemos a importância da socialização e simulação para direcionar os estudantes ao refinamento e correção das soluções.

Mesmo que as soluções estejam disponibilizadas, os estudantes foram incentivados a realizar uma versão da sua solução antes de verificar as soluções dos colegas. Assim, dando importância ao processo de pensar sobre o problema e propor uma solução e não apenas atingir a "resposta certa". Principalmente, porque as avaliações dos desafios não focam em avaliar se um estudante resolveu corretamente ou não, mas sim se foi proposta uma solução que atenda o que foi pedido no enunciado do desafio. Um problema de deixar as soluções abertas é que em um determinado momento ocorreria convergência das soluções, e todas as soluções posteriores estavam muito próximas das anteriores. Ainda, as soluções com interpretações diferentes poderiam induzir os colegas ao erro.

Fez parte também da socialização uma avaliação por pares, cada um dos estudantes precisava avaliar e fornecer um comentário escrito sobre a solução de um colega. Esse processo de avaliação por pares pode ser visto como construção social do conhecimento, e com um impacto mais significativo, pois se trata da visão de um colega de mesmo nível e não de um professor. Quando a turma estava mais confortável em realizar comentários, percebemos que as conversas entre os estudantes extrapolavam a sala de aula e encontros síncronos. No contexto remoto, o Moodle ajudou a fomentar ainda mais essa conversa entre os estudantes.

A forma como os desafios foram conduzidos possuem relação com a prática docente, e pode influenciar diretamente no interesse e motivação dos estudantes. A prática docente também pode influenciar no conjunto de habilidades que será exercitado, então, ao retirar, adicionar, ou até mesmo focar em uma etapa do processo de resolução da solução impactará no conjunto final de habilidades exercitadas pelos estudantes. Portanto, os objetivos de uma prática devem ser claros e a condução precisa ser guiada com base nas habilidades que se deseja exercitar.

No contexto remoto é mais difícil perceber a necessidade de intervenção, e em quais aspectos os estudantes precisam de mais apoio. O contexto remoto requer uma atenção maior à qualidade de explicação, principalmente por a interação ser menor quando comparado ao período comum de aulas. Reconhecemos que a qualidade da explicação pode melhorar a percepção dos estudantes sobre a dificuldade do desafio, e com a dificuldade perceptível ajustada pode ter impactos positivos na capacidade de promover o interesse.

Algumas habilidades só serão desenvolvidas de acordo com a metodologia realizada, ou a temática da prática. Aspectos como responsabilidade e ética profissional é um processo de reflexão muito grande, e que demanda tempo e envolve questões subjetivas. Com os desafios buscamos provocar o raciocínio e o exercício do Pensamento Computacional, a percepção dos estudantes revela que eles tanto exercitaram o Pensamento Computacional quanto desenvolveram consciência desse exercício. A autopercepção individual do próprio aprendizado é fundamental quando falamos de estudantes ingressantes, porque ela tem efeitos positivos sobre a motivação e interesse dos estudantes.

Fluxograma utiliza elementos gráficos para representar uma solução, e isso pode adicionar uma camada de complexidade com relação aos elementos que pode ser comparada com a complexidade existente em aprender a utilizar uma linguagem de programação. Além de que por ter uma relação grande com programação pode ser visto como uma barreira pelos estudantes. Para os estudantes, representar a solução em forma de fluxograma foi uma tarefa difícil, e poucas soluções foram representadas em fluxograma de maneira correta, considerando um fluxograma bem estruturado, com início, meio fim, entrada, saída, condições e repetições. Os estudantes também tinham dúvidas de onde poderiam colocar uma estrutura de repetição ou condicional para não precisar fazer tudo sequencial.

5.3 COLETA DE EVIDÊNCIAS DO EXERCÍCIO DO PENSAMENTO COMPUTACIONAL: EXPERIMENTO DO ARTEFATO

Durante o estudo de caso #4 os professores e colaboradores da disciplina relataram a necessidade de algum apoio para conduzir a análise e avaliação das soluções dos desafios realizados pelos estudantes. A partir dessa demanda, propomos um artefato para verificar o exercício do Pensamento Computacional nos desafios da abordagem Pense Computacionalmente por meio da coleta de evidências. O artefato Computacionalidade fornece o conjunto de habilidades e suas definições, enumera as evidências a serem analisadas, e para mensurar fornece uma escala simplificada. Com o objetivo de verificar as potencialidades e fragilidades do artefato desenvolvido realizamos um experimento com os professores e colaboradores da disciplina de CI1003. O experimento foi realizado neste contexto porque foi nesta disciplina que estávamos investigando as atividades que criamos para exercitar o Pensamento Computacional, e foram os docentes desta disciplina que relataram a necessidade de apoio para analisar e avaliar as soluções das atividades. A Tabela 5.9 apresenta a ficha técnica do experimento realizado.

Tabela 5.9: Detalhes do experimento

Onde?	Disciplina de Introdução à Ciência da Computação, dos Cursos de Bacharelado em Ciência da Computação e Informática Biomédica da Universidade Federal do Paraná (UFPR).
Quem participou?	7 docentes: 3 professores da disciplina e 4 doutorandos colaboradores.
Qual foi a dinâmica?	2 encontros síncronos: utilização do artefato e avaliação do artefato.
Quando aconteceram as dinâmicas?	1º encontro síncrono: 25 de junho de 2021. Utilização do artefato: entre 25 de junho a 06 de julho de 2021. 2º encontro síncrono: 13 de julho de 2021.

Participaram do experimento 7 docentes, sendo os 3 professores da disciplina e 4 doutorandos colaboradores da disciplina em que o estudo de caso #4 ocorreu. A turma era composta por 25 estudantes. A seleção dos participantes foi realizada de maneira não-probabilística de amostragem por conveniência (Wohlin et al., 2012), uma vez que os participantes são do mesmo grupo de pesquisa da autora e pessoas envolvidas com a disciplina onde escolhemos realizar os estudos de caso. Como métodos para coleta de dados foram utilizados:

- Registros em arquivo, para coletar as soluções dos 5 desafios para verificar as evidências do exercício das habilidades do Pensamento Computacional;
- Questionário pós-estudo, para coletar a percepção das pessoas participantes sobre o artefato produzido;
- Grupo focal, para coletar mais informações relacionadas ao questionário de pós-estudo, a fim de ampliar a discussão com as pessoas que responderam o questionário pós-estudo sobre o que eles responderam.

A análise dos dados foi qualitativa, no qual o estudo visa tirar conclusões dos dados e manter uma clara cadeia de evidências (Wohlin et al., 2012). O método de análise dos dados utilizado foi a Análise Temática, pois permite identificar, analisar, interpretar e relacionar os padrões (temas) a partir de dados qualitativos com uma abordagem indutiva, que se baseia nos dados e não possui categorias ou temas preestabelecidos (Braun e Clarke, 2006).

O experimento foi dividido em quatro momentos: (i) encontro síncrono de explicação e alinhamento; (ii) utilização do artefato para analisar as soluções dos estudantes, e coletar evidências; (iii) resposta do questionário de pós-estudo; (iv) encontro síncrono para realizar o grupo focal pós-estudo. O primeiro encontro síncrono teve como objetivo apresentar o artefato e explicar a sua utilização. Cada desafio tinha uma planilha específica, em que cada linha da planilha representava um estudante. A utilização do artefato e a coleta de evidências consistiram em verificar cada uma das soluções e registrar a evidência na planilha do desafio. Ao todo, 83 soluções foram armazenadas e analisadas. A quantidade de soluções analisadas é menor do que a quantidade esperada porque nem todos os estudantes entregaram soluções para todos os desafios. O número de soluções foi dividido pelos 8 participantes, e cada participante avaliou entre 10 e 11 soluções e revisou outras 10 ou 11 soluções avaliadas pelos demais participantes, de modo a reduzir o viés da análise. Nos casos em que havia discordância, os participantes discutiam e chegavam a um consenso utilizando os recursos de comentários da ferramenta.

Após o término do experimento, os participantes preencheram o questionário pós-estudo. O questionário pós-estudo foi baseado no TAM (*Technology Acceptance Model 3*) proposto por Venkatesh e Bala (2008), e teve como objetivo capturar as impressões dos participantes quanto à utilidade, facilidade percebida e intenção de uso do artefato produzido. O questionário era composto por 16 questões obrigatórias de múltipla-escolha, subdivididas em 3 seções, somado a 3 questões discursivas optativas. As questões de múltipla-escolha foram avaliadas em uma escala de 5 pontos, baseada na escala Likert (1932). As questões discursivas são para coletar comentários gerais sobre utilidade, a facilidade, ou a intenção de uso do artefato. O grupo focal foi realizado em um terceiro momento do experimento para aprofundar as discussões sobre a experiência e sobre os resultados do questionário pós-estudo que os participantes responderam. O grupo focal foi realizado no dia 16 de julho de 2021 com 7 participantes (uma pessoa não pôde participar), teve duração de 60 minutos e foi dividido em 3 etapas: (i) introdução; (ii) apresentação da síntese dos resultados do questionário de avaliação; e (iii) uma conversa entre os participantes, a partir da sumarização realizada pela autora.

5.3.1 Resultados e Discussões sobre o Experimento

A partir do questionário pós-estudo, aplicado via Google Forms, foi possível obter as impressões dos participantes referentes à utilidade percebida, facilidade percebida e intenção de uso. As perguntas e respostas das 16 questões de múltipla-escolha obrigatórias deste questionário estão sumarizadas na Tabela 5.10. A partir das respostas, percebemos que os resultados obtidos nos três aspectos avaliados foram positivos. O artefato atinge o objetivo esperado de apoiar a coleta de evidências do exercício do Pensamento Computacional sendo útil para guiar esse processo. Os participantes sinalizaram que é muito útil ter conhecimento sobre as dimensões e habilidades do Pensamento Computacional durante a coleta de evidências. A escala para mensurar a dimensão do exercício do Pensamento Computacional também foi considerada útil. Além de facilitar o processo de coleta de evidências, disponibilizar as habilidades e as evidências que devem ser coletadas ajuda a reduzir a dúvida e o viés durante o processo de coleta. A escala, por sua vez, ajuda a padronizar a avaliação e a partir dela conseguimos ter uma visão ampla para entender como e em quais níveis as habilidades estão sendo exercitadas.

Com relação à utilidade percebida, 5 participantes sinalizaram que foi fácil utilizar o artefato para coletar evidências do exercício do Pensamento Computacional. O artefato aumenta (1) e muito (4) a qualidade da coleta de evidências, pois *"ele é especialmente útil por lembrar em quais pontos prestar atenção ao analisar uma solução"* de acordo com um participante. Por ser um artefato que contempla 9 habilidades e 25 evidências a serem coletadas, e ser a primeira vez que os participantes estavam utilizando, o artefato exigiu esforço e tempo considerável para analisar todas as soluções. Por se tratar de um número significativo de evidências a serem observadas, os participantes mencionaram que isso pode ser um dos fatores que inviabilize a utilização do artefato, mesmo que ele facilite o entendimento geral do que se busca analisar em cada uma das habilidades. Além disso, ter uma grande quantidade de questões passa a ser um obstáculo para a utilização do artefato quando a turma é grande. Nessa mesma linha, verificamos que, com relação à facilidade de uso, todos os participantes sinalizaram que utilizar o artefato exige esforço mental. A carga cognitiva necessária, principalmente na primeira utilização, pode ter sido alta pois eram muitas informações novas e aspectos que deveriam ser considerados durante a execução da análise das soluções.

Considerando a facilidade de uso, o artefato facilita o entendimento de como coletar evidências do exercício do Pensamento Computacional e facilita a visualização e compreensão de quais habilidades estão sendo exercitadas em cada um dos desafios. Entretanto, um participante sinalizou que o artefato não facilita a visualização e compreensão da relação das habilidades nos desafios. O formato utilizado é um ponto recorrentemente mencionado pelos participantes: *"a estrutura da planilha dificulta o registro, e mesmo que muito fácil de preencher perde-se a visão do todo"*. Um dos participantes sinalizou que talvez a usabilidade do artefato em dispositivos móveis seja comprometida pelo formato que ele se encontra hoje, principalmente por quê *"a planilha acaba tirando muito espaço da tela pela grande quantidade de barras e ícones que possui"*. Então, um formato mais acessível e de fácil navegação deve ser repensado. Mesmo que o artefato tenha sido representado em uma planilha do Google Sheets, a intenção era de facilitar o entendimento sobre as habilidades, as evidências e estruturar a coleta de evidências do Pensamento Computacional. Ao utilizar uma planilha online tivemos como objetivo aproveitar os recursos existentes na ferramenta que poderiam facilitar a interação como ocultar linhas e colunas; congelar linhas e colunas; e filtrar informações. Entretanto, o aproveitamento desses recursos ficou restrito a quem tinha conhecimento sobre eles e a quem lembrava de usá-los.

Quando perguntamos aos participantes sobre a intenção de uso, todos avaliaram que provavelmente utilizariam o artefato outras vezes e provavelmente recomendariam para outras pessoas utilizarem também. *"O artefato facilita compreensão sobre aquilo que é importante*

Tabela 5.10: Perguntas e respostas do questionário pós-estudo

	Perguntas	Alternativas e Quantidade				
Utilidade Percebida	Quão difícil é coletar evidências do exercício do Pensamento Computacional utilizando o artefato?	É muito fácil 5	É fácil 2	Neutro 0	É difícil 0	É difícil 0
	O artefato apoia a coleta de evidências do exercício do Pensamento Computacional?	Sim, o artefato apoia muito 5	Sim, o artefato apoia 2	Neutro 0	Não, o artefato não apoia 0	Não, o artefato prejudica 0
	O artefato é útil para guiar a coleta de evidências do exercício do PC?	Sim, o artefato é muito útil 5	Sim, o artefato é útil 2	Neutro 0	Não, o artefato não é útil 0	Não, o artefato prejudica 0
	Com o artefato você consegue coletar evidências do Pensamento Computacional que precisa coletar?	Sim, consigo coletar muito fácil 0	Sim, consigo coletar fácil 6	Neutro 1	Não, não consigo coletar 0	Não, não consigo coletar nada 0
	Quão útil é ter uma escala para mensurar as habilidades exercitadas pelo desafio?	A escala foi muito útil 4	A escala foi útil 3	Neutro 0	A escala não foi útil 0	A escala atrapalhou 0
	O quão útil é ter conhecimento sobre as dimensões do Pensamento Computacional para a coleta de evidências?	É muito útil 7	É útil 0	Neutro 0	Não é útil 0	Prejudica 0
	O quão útil é ter conhecimento das habilidades do Pensamento Computacional durante coleta de evidências do exercício do Pensamento Computacional?	É muito útil 7	É útil 0	Neutro 0	Não é útil 0	Prejudica 0
	O quão útil é ter conhecimento sobre quais pontos devem ser observados de cada habilidade durante a coleta de evidências do exercício do Pensamento Computacional?	É muito útil 6	É útil 1	Neutro 0	Não é útil 0	Prejudica 0
	O quanto o artefato aumenta a qualidade da coleta de evidências do exercício Pensamento Computacional?	O artefato aumentou muito a qualidade 4	O artefato aumentou a qualidade 1	Neutro 2	O artefato não aumenta a qualidade 0	O artefato prejudica a qualidade 0
Facilidade de Usos	O quanto o artefato exige do meu esforço mental?	Exige muito esforço mental 1	Exige esforço mental 5	Neutro 0	Exige pouco esforço mental 1	Exige nenhum esforço mental 0
	É fácil utilizar o artefato para fazer o que eu quero que ele faça (coletar evidências do exercício do Pensamento Computacional)?	Sim, muito fácil 0	Sim, fácil 5	Neutro 1	Não, foi pouco fácil 1	Não, foi difícil 0
	A minha interação com o artefato é clara e compreensível?	Sim, muito clara e compreensível 1	Sim, clara e compreensível 4	Neutro 1	Não, foi pouco clara e compreensível 1	Não, não foi clara e compreensível 0
	O artefato facilita a visualização e compreensão de quais habilidades do Pensamento Computacional estão sendo trabalhadas nos desafios?	Sim, facilita muito a visualização e compreensão 0	Sim, facilita a visualização e compreensão 5	Neutro 1	Não, ele não facilita a visualização e compreensão 1	Não, ele não facilita nada a visualização e compreensão 0
	O artefato facilita o meu entendimento de como coletar evidências do exercício do Pensamento Computacional?	Sim, facilita muito o entendimento 2	Sim, facilita o entendimento 5	Neutro 0	Não, ele não facilita o entendimento 0	Não, ele não facilita nada o entendimento 0
Intenção de Usos	Supondo que eu precise coletar evidências do desenvolvimento do Pensamento Computacional e tenha acesso ao artefato, eu pretendo utilizá-lo outras vezes?	Muito provavelmente pretendo utilizar o artefato outras vezes 2	Provavelmente pretendo utilizar o artefato outras vezes 5	Neutro 0	Provavelmente eu não pretendo utilizar o artefato outras vezes 0	Não pretendo o artefato outras vezes 0
	Eu recomendaria o uso do artefato para outras pessoas que têm interesse em coletar evidências do exercício do Pensamento Computacional?	Muito provavelmente eu recomendaria o uso do artefato 5	Provavelmente eu recomendaria o uso do artefato 2	Neutro 0	Provavelmente eu não recomendaria o uso do artefato 0	Não recomendaria o uso do artefato 0

de ser percebido e coletado nos desafios e auxilia a guiar a análise", disse um participante. O artefato pode ser visto como um guia do que minimamente precisa ser observado para concluir que a pessoa exercita uma habilidade com uma determinada atividade. O artefato também pode ser utilizado para apoiar a criação de novos desafios, mencionou um participante durante o grupo focal. Logo, o artefato serve como uma base para estruturar a análise de maneira mais ampla e abrangente, reduzindo as chances de que evidências relevantes sejam negligenciadas.

O grupo focal foi conduzido a partir da sumarização dos dados resultantes do questionário de avaliação do artefato. Um dos grandes problemas mencionados pelos participantes foi a quantidade de perguntas e informações existentes no artefato, e utilizamos o grupo focal para refletir sobre esses aspectos e pensar em estratégias para reduzir essa quantidade de informações. O grupo debateu diferentes estratégias para a redução da quantidade de perguntas, e os participantes sempre adicionavam pontos e questões para amadurecer as ideias. Uma das estratégias pensadas para isso foi o artefato servir como um guia de quais habilidades podem ser exercitadas e como coletar evidências de seu exercício para, a partir disso, a pessoa que for utilizar o artefato selecionar o conjunto de habilidades que deseja verificar. A seleção das habilidades a serem coletadas também pode ser ajustada de acordo com cada atividade.

Um segundo tema que aprofundamos no grupo focal foi como compilar as evidências coletadas, que tipo de informações os docentes gostariam ter disponível ao final da avaliação, e de que forma elas seriam representadas. Os participantes compartilharam que ajudaria ter representações visuais para sumarizar o que foi coletado. Neste momento, a conversa foi conduzida para que os participantes compartilhassem que tipo de representação visual eles conheciam e que eles viam que poderia ser utilizada, principalmente para que docentes tivessem uma perspectiva do estudante, do estudante com relação à turma, e da turma como um todo. Na mesma linha de sumarizar a coleta de evidências, um dos participantes sinalizou que, durante o preenchimento do artefato, ele sentiu que estava fazendo duas coisas e que elas poderiam ser integradas no futuro. Os docentes avaliaram as soluções dos estudantes utilizando o artefato como apoio, porém, como fazia parte de uma disciplina eles precisavam dar um retorno por escrito para cada um dos estudantes. O artefato conta com um espaço para que o docente anote observações sobre uma determinada solução. Logo, o participante teve a impressão de que estava avaliando a solução (coletando evidências com o apoio do artefato) e anotando as observações de cada solução. Assim, ele percebeu que a coleta de evidências somada com as observações anotadas poderia ser utilizada para compor o *feedback* qualitativo para os estudantes. A conversa entre os participantes identificou que esse recurso somaria na sumarização da coleta de evidências.

O artefato Computacionalidade foi desenvolvido considerando um conjunto de 9 habilidades que estão relacionadas com os pilares do Pensamento Computacional: (i) abstração, (ii) decomposição, (iii) reconhecimento de padrões e (iv) algoritmos. A partir da definição de cada uma das habilidades, enumeramos pelo menos uma evidência a ser observada para cada habilidade, totalizando 25 evidências. Definir as evidências e uma escala para mensurar as evidências são os pontos substanciais e úteis do artefato, independente do formato utilizado. O artefato serve também como um *template* para que professores e pesquisadores se inspirem para adaptar e instanciar as suas próprias versões e utilizá-las em seus contextos. Ainda, o artefato fornece uma visão holística das habilidades e das evidências, ampliando a abrangência multidimensional da análise. Sendo uma inspiração ou ponto de partida para que outras pessoas ajustem, adaptem e reproduzam para representar as informações que professores e pesquisadores têm intenção de verificar.

O artefato apoia a coleta das evidências do exercício dessas habilidades, porém, ele ainda não conta com nenhum recurso que compile ou transforme a análise feita em algo qualitativo ou quantitativo. A sumarização das evidências coletadas pode ser transformada em um valor

que quantifique o exercício dessas habilidades quanto ser convertida em um comentário que qualifique a solução. Para converter a coleta em um valor final é necessário definir e atribuir uma relação de pesos entre as evidências para cada atividade, além da necessidade de criar uma fórmula para resumir a análise em um único valor final. Enquanto para traduzir a coleta em um parecer escrito das soluções (qualitativo) é necessário pensar em soluções de processamento de texto. Definir pesos, a fórmula matemática, e formalizar o processo de tradução das evidências coletadas para gerar um parecer textual são visualizados como trabalhos futuros.

5.4 CONCEPÇÃO DA ABORDAGEM

A partir do problema de pesquisa em identificar como apoiar o exercício do Pensamento Computacional no ensino superior em Computação buscamos investigar uma abordagem para apoiar a concepção e condução de práticas educacionais que promovam o exercício do Pensamento Computacional no ensino superior em Computação. A criação da abordagem deu início em 2019 com o desenvolvimento de um conjunto de quatro desafios, e foram conduzidos na disciplina de Introdução à Ciência da Computação dos cursos de Bacharelado em Ciência da Computação e Informática Biomédica da Universidade Federal do Paraná. Como foi a primeira dinâmica em sala de aula, a criação e a utilização dos desafios foram acontecendo paralelamente. Inicialmente, os desafios tinham o objetivo de provocar estudantes a pensarem no processo de resolução de problemas considerando as habilidades do Pensamento Computacional. Foram criados quatro desafios: a lista de chamada, ordenação, busca e celebridade.

Desde o início, a premissa era que o Pensamento Computacional deveria ser explicitamente exercitado no primeiro semestre dos cursos de graduação em Computação de forma contextualizada, desplugada e baseada em desafios. Logo, pensamos em criar desafios que não requeriam a utilização de nenhuma ferramenta computacional para propor a solução. As soluções deveriam ser representadas em sequências de instruções simples em linguagem natural, uma vez que ocorreria a interpretação literal das instruções ao executá-las. Por estarmos trabalhando em uma disciplina com estudantes recém ingressantes no curso, percebemos que eles não se conheciam e ainda estavam tímidos em sala, logo pensamos que trariam benefícios para o engajamento entre eles se tivéssemos uma etapa de construção colaborativa.

Uma habilidade importante para graduandos em Computação é a habilidade de simular e testar soluções, equivalente aos testes de mesa. Reconhecemos que ter uma etapa para testar a solução e verificar a execução dela auxilia no entendimento e reforço dos conceitos computacionais. Com isso, identificamos que durante a condução em sala deveríamos ter uma etapa para testar as soluções. Com a intenção de deixar os desafios mais dinâmicos, reconhecemos que precisava ter uma etapa de socialização entre os estudantes. A etapa de socialização não poderia começar do zero, ela deveria auxiliar o amadurecimento da solução. Observamos em sala que os estudantes começaram a se engajar mais tanto durante a resolução dos desafios quanto com os colegas no momento de criação da solução. Além disso, durante o momento da socialização foi possível perceber que as discussões começaram a ser mais profundas e duravam mais tempo.

No segundo estudo de caso, como tinham poucos estudantes, aproveitamos para observar mais de perto quais seriam as dúvidas apresentadas ao longo da condução dos desafios. Durante o estudo de caso, além de refinar os quatro desafios existentes, criamos três novos desafios: o Mais Velho, Ordenando a Biblioteca, e Criptografia. O Teste dos Labs foi criado por um estudante para uma atividade da disciplina deste estudo de caso, o teste foi incorporado como um desafio da abordagem. No fim deste estudo de caso, identificamos a necessidade de organizar todas as informações sobre os desafios e orientações de como conduzi-los em sala pensando na

divulgação e replicação do material. Com isso, formatamos os desafios no estilo de plano de aula, fornecendo informações de materiais, métodos, e folhas de resolução.

Para o terceiro estudo de caso, com o objetivo de identificar a aplicabilidade dos desafios em um contexto remoto, foi necessário adaptar todo o material criado para serem conduzidos no contexto remoto. Selecionamos cinco desafios para serem conduzidos neste estudo de caso, foram: Desafio. O mais velho tem olhos azuis; Desafio. Desafio Teste dos Labs; Desafio. Criptografia; Desafio. Ordenação; e Desafio. Celebridade. Neste estudo de caso também aperfeiçoamos o questionário online para coletar a percepção dos estudantes. No quarto estudo de caso realizado buscamos identificar as potencialidades e fragilidades dos desafios criados para que estimularem o exercício do Pensamento Computacional. A partir deste estudo, e da necessidade identificada, criamos as orientações para conduzir os desafios em sala de aula, e coletamos a percepção dos professores e colaboradores da disciplina nesta oferta por meio de entrevista.

Durante o quarto estudo de caso, professores e colaboradores da disciplina relataram a necessidade de um apoio para conduzir a análise e avaliação das soluções dos desafios realizados pelos estudantes. A partir dessa demanda, durante o quinto estudo de caso propomos o artefato Computacionalidade para verificar o exercício do Pensamento Computacional nos desafios da abordagem Pense Computacionalmente por meio da coleta de evidências. Além de continuar identificando as potencialidades e fragilidades da abordagem, o quinto e sexto estudo de caso foram para aprofundar o conhecimento sobre as habilidades relacionadas com o Pensamento Computacional e como coletar evidências do seu desenvolvimento. O artefato para coleta de evidências foi incorporado na abordagem.

Durante o quinto e sexto estudo de caso o material da abordagem foi diagramado, buscamos utilizar na abordagem uma linguagem mais inclusiva. Além disso, temáticas de desafios foram alteradas com o objetivo de levantar questionamentos com relação ao gênero e éticas sociais. O processo de criação da abordagem aconteceu de maneira incremental, de uma maneira que no fim de cada estudo de caso foram identificados os problemas e traçando estratégias para resolvê-los. Para o próximo estudo, as melhorias foram incorporadas na abordagem.

6 CONCLUSÕES E TRABALHOS FUTUROS

O Pensamento Computacional é um processo de pensamento envolvido na formulação de problemas e suas soluções, e preocupa-se em como as soluções devem ser representadas para que possam ser efetivamente executadas por um agente de processamento de informações. Pensar computacionalmente significa visualizar um mesmo problema em diferentes níveis, e encontrar estratégias para reformular um problema difícil em problemas menores e mais fáceis de serem resolvidos – refletindo na solução de maneira lógica, algorítmica, e abstrata. Mesmo que o Pensamento Computacional favoreça o desenvolvimento das habilidades de codificação, ele é muito mais que apenas programação e envolve mais conteúdos. Desse modo, o Pensamento Computacional deve ser desenvolvido às luzes das habilidades mais gerais de resolução de problemas e o seu desenvolvimento guiará o aprendizado dos conteúdos Computacionais.

O Pensamento Computacional proporciona uma aprendizagem mais ampla e apoia a aprendizagem de conceitos centrais da Computação. Além disso, o Pensamento Computacional contribui para apoiar o processo de aprendizagem dos estudantes e pode ser uma possibilidade de mitigar as lacunas de formação dos estudantes ingressantes nos cursos de graduação em Computação. Embora a literatura reconheça a importância e a relevância do Pensamento Computacional, ainda há poucas evidências sobre quais aspectos devem ser considerados durante a criação de práticas educacionais que favoreçam o exercício dessas habilidades.

Segundo o MEC, a SBC, a ACM e a IEEE o Pensamento Computacional é uma das habilidades fundamentais desejadas para todos os graduandos, principalmente nas áreas da Computação. As diretrizes curriculares para a graduação em Computação apontam que o Pensamento Computacional deve ser exercitado pelos estudantes, mas não discutem quando e como o Pensamento Computacional deve ser exercido. Para avançar no problema da falta de conhecimento em como apoiar o exercício do Pensamento Computacional no ensino superior em Computação investigamos uma abordagem baseada em desafios.

OE1) Caracterizar as iniciativas da literatura que focam no exercício do Pensamento Computacional no ensino superior. Com o objetivo de mapear sistematicamente a literatura de Pensamento Computacional para obter uma visão geral das iniciativas existentes realizadas no ensino superior e aprofundar o conhecimento das pesquisas na área, realizamos um mapeamento sistemático de literatura. Em abril de 2019 mapeamos 4 bibliotecas digitais que nos retornou 2.966, após o processo de filtragem selecionamos 94 artigos relevantes para nosso mapeamento sistemático. De acordo com o mapeamento, o Pensamento Computacional é investigado a partir de diferentes perspectivas e com diferentes públicos. O mapeamento revelou que o Pensamento Computacional está intimamente associado aos fundamentos de programação e pouco associado com a área da Computação em si.

OE2) Propor uma abordagem para apoiar o exercício do Pensamento Computacional no ensino superior em Computação. A criação da abordagem deu início em 2019 com o desenvolvimento de um conjunto de quatro desafios, e foram conduzidos na disciplina de Introdução à Ciência da Computação do curso de Bacharelado em Ciência da Computação e Informática Biomédica da Universidade Federal do Paraná (UFPR). Desde 2019, os desafios foram conduzidos em ofertas diferentes da disciplina com um grupo diferente de estudantes. O processo de criação da abordagem aconteceu de maneira interativa, no fim de cada estudo foram identificados os problemas e traçados estratégias para resolvê-los. Para a próxima oferta, as melhorias eram incorporadas na abordagem.

OE3) Investigar as potencialidades e fragilidades da abordagem com o público-alvo em diferentes contextos. De 2019 a 2021 a disciplina foi ofertada 6 vezes, e em cada uma delas realizamos um estudo de caso para experimentar a abordagem em diferentes contextos.

OE4) Investigar as potencialidades e fragilidades do uso de atividades desplugadas para exercitar o Pensamento Computacional, e OE5) Investigar as potencialidades e fragilidades do uso de linguagem natural e fluxogramas como forma de representar soluções. Por meio da observação participante realizada durante as ofertas da disciplina, questionário online e anônimo para coletar a percepção dos estudantes, e colaboradores da disciplina, investigamos os impactos da abordagem, as fragilidades e potencialidades. A abordagem tem como propósito apoiar professores e pesquisadores na concepção e condução de práticas educacionais para promover o exercício do Pensamento Computacional por meio da solução dos desafios.

A abordagem tem como premissa de que o Pensamento Computacional deve ser explicitamente exercitado desde o primeiro semestre dos cursos de graduação de forma significativa, colaborativa, contextualizada e com uma abordagem desplugada e baseada em desafios. Desenvolvemos a abordagem para ser utilizada ou adaptada para diferentes contextos educacionais, e ela é composta por: (i) um conjunto de orientações de como preparar e conduzir o exercício do Pensamento Computacional; (ii) um conjunto de sete práticas para serem utilizadas ou inspirar a criação de novas; (iii) um método para guiar a condução da prática e instruir os estudantes durante a proporem uma solução; e (iv) um artefato para apoiar a análise das soluções para coletar evidências do exercício do Pensamento Computacional.

Os resultados revelam que os desafios foram bem avaliados tanto pelos docentes quanto discentes, e foram muito bons em provocar o raciocínio e bom na capacidade de promover o interesse dos estudantes. Com isso, atingimos o objetivo esperado de exercitar o Pensamento Computacional por meio da abordagem Pense Computacionalmente e os desafios propostos. Mesmo que os desafios foram bem avaliados, é possível perceber que ainda existem espaços para ajustes, tanto com relação ao nível de dificuldades quanto na qualidade das explicações. A qualidade de explicação é ainda mais delicada em contextos remotos, na qual a interação foi reduzida à expressão textual. Além disso, os estudantes sinalizaram que gostaram dos desafios tanto da maneira geral quanto ao fato de que os desafios incentivaram testar diferentes caminhos para solução e até a forma como os desafios foram conduzidos. Os desafios foram considerados interessantes porque instigaram estudantes a pensarem em maneiras diferentes de resolver o problema, e isso teve impacto na segurança, aceitação e entendimento ao erro, e aumento da motivação para realizar os desafios.

Embora a abordagem tenha sido criada e testada com estudantes de graduação em Computação, vemos que ela tem um grande potencial para ser adaptada e utilizada em outros cursos de graduação, e até mesmo com estudantes de outras faixas etárias. As atividades devem fornecer experiências para mostrar que conceitos abstratos fazem sentido e são aplicáveis ao mundo cotidiano, proporcionando aos estudantes a construírem significado com base em suas próprias experiências. Considerando isso, para utilização da abordagem em outros contextos e público-alvo basta ajustar os desafios para considerar o contexto em que será aplicado, e regular a profundidade de cada desafio.

Como resultados desta tese, podemos citar contribuições conceituais: como criar e conduzir práticas educacionais que exercitem o Pensamento Computacional no ensino superior em Computação; a relação habilidade por desafio e com as etapas de resolução de um desafio; e a caracterização de como ter evidências do exercício das habilidades. Uma contribuição metodológica que o que considerar e como conduzir o exercício do Pensamento Computacional no ensino superior em Computação. A formalização da abordagem como um material didático autocontido como uma contribuição técnica.

Como trabalhos futuros identificamos a possibilidade de aplicar a abordagem em outros cursos de graduação, e com outras faixas etárias. Para isso, identificamos a necessidade de adaptar a abordagem para que ela esteja mais próxima do contexto em que será aplicada. Considerando a evolução da abordagem, vemos que uma próxima etapa seja investigar como as questões de gênero influenciam na educação em Computação e na abordagem. A ampliação da abordagem com a incorporação de novos desafios, que considerem outros conteúdos computacionais e outras temáticas que apoiem o exercício das habilidades de responsabilidade e ética profissional. Com relação à avaliação e evidências do exercício do Pensamento Computacional podemos citar a possibilidade de desenvolvimento de uma solução para avaliar as soluções automaticamente e identificar os exercícios das habilidades com base no artefato criado.

REFERÊNCIAS

ACM, A. f. C. M. e IEEE, C. S. (2014). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Relatório técnico.

Agbo, F. J., Oyelere, S. S., Suhonen, J. e Adewumi, S. (2019). A systematic review of computational thinking approach for programming education in higher education institutions. Em *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, Koli Calling '19, New York, NY, USA. Association for Computing Machinery.

Almeida, R. e Pessoa, T. (2017). Scratch Software in Higher Education: pedagogical experience in educational science. Em *2017 International Symposium on Computers in Education (SIIE)*, páginas 1–5.

Almeida, R., Pessoa, T. e Gomes, A. (2018). Learning to Think Like a Trainer: Bringing Scratch for Educational Sciences Professional's Formation. Em *2018 IEEE Frontiers in Education Conference (FIE)*, páginas 1–7.

Ambrósio, A. P., Xavier, C. e Georges, F. (2014). Digital Ink for Cognitive Assessment of Computational Thinking. Em *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, páginas 1–7.

Angeli, C. e Giannakos, M. (2020). Computational Thinking Education: Issues and Challenges. *Computers in Human Behavior*, 105:106185.

Apple (2011). Challenge Based Learning: a classroom guide. Relatório técnico.

Ausubel, D. P. (1963). *Psychology of Meaningful Verbal Learning: An Introduction to School Learning*. Grune & Stratton.

Avila, C. M. O., Bordini, A., Cavalheiro, S. A. d. C., Reiser, R. H. S., Weisshahn, Y., Aguiar, M. S., da Cunha, M. M. e Foss, L. (2017). Computação na Educação Básica no Brasil: o estado da arte. *Revista de Informática Teórica e Aplicada*, 23(2):210.

Babbitt, T., Schooler, C. e King, K. (2019). Punch Cards to Python: A Case Study of a CS0 Core Course. Em *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, página 811–817, New York, NY, USA. Association for Computing Machinery.

Barba, E. e Chancellor, S. (2015). Tangible Media Approaches to Introductory Computer Science. Em *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, página 207–212, New York, NY, USA. Association for Computing Machinery.

Barcelos, T. S. e Silveira, I. F. (2012). Teaching Computational Thinking in Initial Series an Analysis of the Confluence Among Mathematics and Computer Sciences in Elementary Education and its Implications for Higher Education. Em *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, páginas 1–8.

Barr, V. e Stephenson, C. (2011). Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1):48–54.

Barros, L. A. (1994). *Suporte a Ambientes Distribuídos para Aprendizagem Cooperativa*. Tese de doutorado, Universidade Federal do Rio de Janeiro.

Batista, E., Jr., A., Cantero, S., Bogarim, C. e Larrea, A. (2016). Uso do Scratch no Ensino de Programação em Ponta Porã: das séries inicias ao ensino superior. *Anais do Workshop de Informática na Escola*, 22(1):565.

Behnke, K. A., Kos, B. A. e Bennett, J. K. (2016). Computer Science Principles: Impacting Student Motivation & Learning Within and Beyond the Classroom. Em *Proceedings of the 2016 ACM Conference on International Computing Education Research*, ICER '16, página 171–180, New York, NY, USA. Association for Computing Machinery.

Bell, T., Rosamond, F. e Casey, N. (2012). *Computer Science Unplugged and Related Projects in Math and Computer Science Popularization*, páginas 398–456. Springer Berlin Heidelberg, Berlin, Heidelberg.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M. e Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational psychologist*, 26(3-4):369–398.

Boechler, P., Artym, C., Dejong, E., Carbonaro, M. e Stroulia, E. (2014). Computational Thinking, Code Complexity, and Prior Experience in a Videogame-Building Assignment. Em *2014 IEEE 14th International Conference on Advanced Learning Technologies*, páginas 396–398.

Borges, K., Menezes, C. e Fagundes, L. (2016). Projetos Maker Como Forma de Promover o Desenvolvimento do Raciocínio Formal. *Anais do Workshop de Informática na Escola*, 22(1):515.

Brackmann, C., Barone, D., Casali, A. e Muñoz-Hernandez, R. B. S. (2016). Computational Thinking: Panorama of the Americas. Em *2016 International Symposium on Computers in Education (SIIE)*, páginas 1–6.

Brackmann, C. P. (2017). *Desenvolvimento do Pensamento Computacional Através de Atividades Desplugadas na Educação Básica*. Tese de doutorado, Universidade Federal do Rio Grande do Sul.

Brackmann, C. P., Barone, D. A. C., Boucinha, R. M. e reichert, J. (2019). Development of computational thinking in brazilian schools with social and economic vulnerability: How to teach computer science without machines. *International Journal for Innovation Education and Research*, 7(4):79–96.

Braun, V. e Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101.

Bremgartner, V., de Magalhães Netto, J. F. e de Menezes, C. S. (2017). Conceptual Framework of Educational Resources Adaptation for Improve Collaborative Learning in Virtual Learning Environments. Em *2017 IEEE Frontiers in Education Conference (FIE)*, páginas 1–9.

Bryant, R., Weiss, R., Orr, G. e Yerion, K. (2011). Using the Context of Algorithmic Art to Change Attitudes in Introductory Programming. *J. Comput. Sci. Coll.*, 27(1):112–119.

Buckingham, L. e Hogan, J. (2014). Computational Science for Undergraduate Biologists via QUT.Bio.excel. volume 29, páginas 1403–1412.

Bui, L. D., Kim, Y. G., Ho, W., Ho, H. T. T. e Pham, N. K. (2018). Developing WebQuest 2.0 Model for Promoting Computational Thinking Skill. *International Journal of Engineering & Technology*, 7(2.29):140–144.

Bulhões, D., Barbosa, F., Viana, R., Santos, C. e Villela, M. (2019). O uso da Computação Desplugada no Processo de Ensino-aprendizagem de Alunos do Curso Técnico em Informática. *Simpósio Brasileiro de Informática na Educação (SBIE)*, 30(1):932.

Caeli, E. N. e Yadav, A. (2020). Unplugged Approaches to Computational Thinking: a Historical Perspective. *TechTrends*, 64(1):29–36.

Carnegie Mellon University, C. (2010). Center for Computational Thinking. Last accessed 24 October 2020.

Carvalho, L., Santos, A., Nakamura, F. e Oliveira, E. (2019). Detecção Precoce de Evasão em Cursos de Graduação Presencial em Computação: um estudo preliminar. Em *Anais do XXVII Workshop sobre Educação em Computação*, páginas 233–243, Porto Alegre, RS, Brasil. SBC.

Chang, C.-K. (2014). Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction. *Journal of Educational Computing Research*, 51(2):185–204.

Chilana, P. K., Alcock, C., Dembla, S., Ho, A., Hurst, A., Armstrong, B. e Guo, P. J. (2015). Perceptions of Non-CS Majors in intro Programming: the rise of the conversational programmer. Em *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, páginas 251–259.

Chowdhury, B., Bart, A. C. e Kafura, D. (2018). Analysis of Collaborative Learning in a Computational Thinking Class. Em *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, página 143–148, New York, NY, USA. Association for Computing Machinery.

Clarke, V., Braun, V. e Hayfield, N. (2015). *Thematic analysis. Qualitative psychology: A practical guide to research methods*, página 222–248.

Conte, S. D., Hamblen, J. W., Kehl, W. B., Navarro, S. O., Rheinboldt, W. C., Young, D. M. e Atchinson, W. F. (1965). An undergraduate program in computer science—preliminary recommendations. *Commun. ACM*, 8(9):543–552.

Cordova, D. I. e Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of educational psychology*, 88(4):715.

Cortina, T. J. (2007). An Introduction to Computer Science for Non-Majors Using Principles of Computation. *SIGCSE Bull.*, 39(1):218–222.

Council, N. R. et al. (2010). *Committee for the Workshops on Computational Thinking, National Research Council, Report of a Workshop on the Scope and Nature of Computational Thinking*. National Academies Press.

Cuny, J., Snyder, L. e Wing, J. M. (2010). Demystifying Computational Thinking for non-Computer Scientists. *Unpublished manuscript in progress, referenced in http://www.cs.cmu.edu/˜CompThink/resources/TheLinkWing.pdf*.

Curzon, P., Bell, T., Waite, J. e Dorling, M. (2019). *Computational Thinking*, página 513–546. Cambridge Handbooks in Psychology. Cambridge University Press.

Cápay, M. (2014). Algorithmic Thinking Observation: how students of applied informatics break the mystery of black box applications. Em *2014 International Conference on Interactive Collaborative Learning (ICL)*, páginas 535–540.

Davies, S. (2008). The Effects of Emphasizing Computational Thinking in an Introductory Programming Course. Em *2008 38th Annual Frontiers in Education Conference*, páginas T2C–3–T2C–8.

Dillenbourg, P. (1999). What do you mean by collaborative learning? *Collaborative- learning: Cognitive and Computational Approaches, Oxford: Elsevier*, páginas 1–19.

Epstein, R. (1990). *Generativity Theory and Creativity*. Sage Publications, Inc.

Epstein, R. (1996). *Cognition, Creativity, and Behavior: Selected Essays*. Praeger Publishers/-Greenwood Publishing Group.

Figueiredo, J., Gomes, N. e García-Peñalvo, F. J. (2016). Ne-Course for Learning Programming. Em *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '16, página 549–553, New York, NY, USA. Association for Computing Machinery.

Flanigan, A. E., Peteranetz, M. S., Shell, D. F. e Soh, L.-K. (2015). Exploring Changes in Computer Science Students' Implicit Theories of Intelligence Across the Semester. Em *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, página 161–168, New York, NY, USA. Association for Computing Machinery.

França, C. e Silva, C. (2020). Identificação de critérios para avaliação do pensamento computacional aplicado. Em *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, páginas 1493–1502, Porto Alegre, RS, Brasil. SBC.

Freire, P. (1997). *Pedagogia da Autonomia: Saberes Necessários à Prática Educativa*. Paz e Terra.

Freire, P. (2014). *Educação como Prática da Liberdade*. Paz e Terra.

García-Peñalvo, F. J. (2016). What Computational Thinking Is. *Journal of Information Technology Research*, 9(3).

Gardner, D., Demuynck, M.-A. e Zhang, J. (2011). Engaging Non-Majors in Computer Literacy Courses. Em *Proceedings of the 49th Annual Southeast Regional Conference*, ACM-SE '11, página 36–41, New York, NY, USA. Association for Computing Machinery.

Gero, A. e Levin, I. (2019). Computational Thinking and Constructionism: creating difference equations in spreadsheets. *International Journal of Mathematical Education in Science and Technology*, 50(5):779–787.

Ginat, D. (2010). The Baffling CS Notions of "as-If"and “Don’t-Care”. Em *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE ’10, página 385–389, New York, NY, USA. Association for Computing Machinery.

Giraffa, L. M. M. e Müller, L. (2017). Metodologia Baseada em Sala de Aula Invertida e Resolução de Problemas Relacionado ao Cotidiano dos Estudantes: uma proposta para ensinar programação para iniciantes. *International Journal on Computational Thinking (IJCThink)*, 1(1):52.

Gokhale, A. A. (1995). *Collaborative Learning Enhances Critical Thinking*, volume 7. Digital Library and Archives of the Virginia Tech University Libraries, Journal of Technology Education.

Goodman, A. L. e Dekhtyar, A. (2014). Teaching Bioinformatics in Concert. *PLOS Computational Biology*, 10(11):1–5.

Gossen, F., Kühn, D., Margaria, T. e Lamprecht, A. (2018). Computational Thinking: Learning by Doing with the Cinco Adventure Game Tool. Em *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, páginas 990–999.

Gouws, L. A., Bradshaw, K. e Wentworth, P. (2013). Computational Thinking in Educational Activities: an evaluation of the educational game Light-bot. páginas 10–15.

Greenberger, M. (1962). *Management and the Computer of the Future*. MIT Press, London.

Guzdial, M. (2015). What’s the best way to teach computer science to beginners? *Commun. ACM*, 58(2):12–13.

Haddad, R. J. e Kalaani, Y. (2015). Can Computational Thinking Predict Academic Performance? Em *2015 IEEE Integrated STEM Education Conference*, páginas 225–229.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M. e Hosking, A. L. (2009). A Multidisciplinary Approach Towards Computational Thinking for Science Majors. Em *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE ’09, página 183–187, New York, NY, USA. Association for Computing Machinery.

Hmelo-Silver, C. E. (2004). Problem-based Learning: What and how do students learn? *Educational Psychology Review*, 16(3):235–266.

Hsu, T.-C., Chang, S.-C. e Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126:296–310.

Hurson, A. R. e Sedigh, S. (2010). Transforming the Instruction of Introductory Computing to Engineering Students. Em *2010 IEEE Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments*, páginas 1–19.

International Society for Technology in Education (ISTE) and Computer Science Teachers Associatio (CSTA (2011). Operational Definition of Computational Thinking: for k-12 education. Relatório técnico.

ISTE, I. S. f. T. i. E. e CSTA, C. S. T. A. (2011). Operational Definition of Computational Thinking: for k-12 education. Relatório técnico.

Jung, A., Park, J., Ahn, A. e Yun, M. (2017). CS for ALL: Introducing Computational Thinking with Hands-On Experience in College. Em *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, páginas 1073–1078.

Kafura, D., Bart, A. C. e Chowdhury, B. (2015). Design and Preliminary Results From a Computational Thinking Course. Em *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, página 63–68, New York, NY, USA. Association for Computing Machinery.

Kafura, D., Bart, A. C. e Chowdhury, B. (2018). A Computational Thinking Course Accessible to Non-Stem Majors. *J. Comput. Sci. Coll.*, 34(2):157–163.

Kafura, D. e Tatar, D. (2011). Initial Experience with a Computational Thinking Course for Computer Science Students. Em *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, página 251–256, New York, NY, USA. Association for Computing Machinery.

Katz, D. L. (1960). Conference report on the use of computers in engineering classroom instruction. *Commun. ACM*, 3(10):522–527.

Kazimoglu, C., Kiernan, M., Bacon, L. e MacKinnon, L. (2012). Learning Programming at the Computational Thinking Level via Digital Game-Play. *Procedia Computer Science*, 9:522 – 531. Proceedings of the International Conference on Computational Science, ICCS 2012.

Kim, B., Kim, T. e Kim, J. (2014). Paper-and-pencil Programming Strategy Toward Computational Thinking for non-majors: design your solution. *Journal of Educational Computing Research*, 49(4):437–459. cited By 6.

Kim, J. A. e Kim, H. J. (2017). Flipped Learning of Scratch Programming with Code.Org. Em *Proceedings of the 2017 9th International Conference on Education Technology and Computers*, ICETC 2017, página 68–72, New York, NY, USA. Association for Computing Machinery.

Knobelsdorf, M., Otto, J. e Sprenger, S. (2017). A Computing Education Approach for Geography Students in Context of GIS. Em *2017 IEEE Global Engineering Education Conference (EDUCON)*, páginas 1790–1796.

Knuth, D. E. (1974). Computer science and its relation to mathematics. *The American Mathematical Monthly*, 81(4):323–343.

Krajcik, J. S. e Blumenfeld, P. C. (2006). *Project-based learning*. na.

Krugel, J. e Hubwieser, P. (2017). Computational Thinking as Springboard for Learning Object-Oriented Programming in an Interactive MOOC. Em *2017 IEEE Global Engineering Education Conference (EDUCON)*, páginas 1709–1712.

Kurniawan, O., Lee, N. T. S., Datta, S., Sockalingam, N. e Leong, P. K. (2018). Effectiveness of Physical Robot Versus Robot Simulator in Teaching Introductory Programming. Em *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, páginas 486–493.

Lai, Y.-H., Chen, S.-Y., Lai, C.-F., Chang, Y.-C. e Su, Y.-S. (2019). Study on enhancing AIoT computational thinking skills by plot image-based VR. *Interactive Learning Environments*.

Larkins, D. B. e Harvey, W. (2010). Introductory Computational Science Using MATLAB and Image Processing. *Procedia Computer Science*, 1(1):913 – 919. ICCS 2010.

Lazar, J., Feng, J. H. e Hochheiser, H. (2017). Critical acclaim for research methods in human computer interaction, second edition. Em *Research Methods in Human Computer Interaction (Second Edition)*, páginas v–viii. Morgan Kaufmann, Boston, second edition edition.

Lester, C. Y. (2008). An Innovative Approach to Teaching an Undergraduate Software Engineering Course. Em *2008 The Third International Conference on Software Engineering Advances*, páginas 301–306.

Li, F., Li, D., Zheng, J. e Zhao, S. (2015). Virtual Experiments for Introduction of Computing: using virtual reality technology. Em *2015 IEEE Frontiers in Education Conference (FIE)*, páginas 1–5.

Liao, L. e Liang, J. (2017). An Empirical Study on Blended Learning to Promote the Development of Computational Thinking Ability of College Students. Em *Proceedings - 2017 International Symposium on Educational Technology, ISET 2017*, páginas 256–260.

Libeskind-Hadas, R. e Bush, E. (2013). A First Course in Computing with Applications to Biology. *Briefings in Bioinformatics*, 14(5):610–617.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Lima, A., Carvalho, T., Bailo, D., Rodrigues, J., Aquino, W. e Acosta, P. (2018). Uma Oficina para Ensino de Algoritmos Paralelos por Meio de Computação Desplugada. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, 7(1):619.

Lin, C., Chao, P., Lin, E. e Tzeng, H. (2018). Exploring the Role of Visual Programming Activities in Computational Thinking. Em *2018 1st International Cognitive Cities Conference (IC3)*, páginas 135–138.

Lisbôa, E. S. e Karling, D. A. (2019). Desenvolvimento do Pensamento Computacional no Ensino Superior: um estudo realizado com a ferramenta App Inventor. *Revista de Ensino da Escola de Educação Básica da Universidade Federal de Uberlândia*, 21(1):58 – 69.

López, A. R. e García-Peñalvo, F. J. (2016). Relationship of Knowledge to Learn in Programming Methodology and Evaluation of Computational Thinking. Em *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '16, página 73–77, New York, NY, USA. Association for Computing Machinery.

L'Heureux, J., Boisvert, D., Cohen, R. e Sanghera, K. (2012). IT Problem Solving: An Implementation of Computational Thinking in Information Technology. Em *Proceedings of the 13th Annual Conference on Information Technology Education*, SIGITE '12, página 183–188, New York, NY, USA. Association for Computing Machinery.

Martin, F. e Sherman, M. (2015). A Dual-Major Course Emphasizing Computer Science and Graphic Design. *J. Comput. Sci. Coll.*, 30(6):24–31.

Mason, D., Khan, I. e Farafontov, V. (2016). Computational Thinking as a Liberal Study. Em *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, página 24–29, New York, NY, USA. Association for Computing Machinery.

MEC, M. d. E. (2016). Diretrizes Curriculares Nacionais para os cursos de Graduação em Computação (DCN16). *Resolução CNE/CES nº 5, de 16 de novembro de 2016*, 2016:9.

Miller, C. S., Perkoviundefined, L. e Settle, A. (2010). File References, Trees, and Computational Thinking. Em *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, página 132–136, New York, NY, USA. Association for Computing Machinery.

Miller, L. D., Soh, L., Chiriacescu, V., Ingraham, E., Shell, D. F., Ramsay, S. e Hazley, M. P. (2013). Improving Learning of Computational Thinking Using Creative Thinking Exercises in CS-1 Computer Science Courses. Em *2013 IEEE Frontiers in Education Conference (FIE)*, páginas 1426–1432.

Miller, L. D., Soh, L.-K., Chiriacescu, V., Ingraham, E., Shell, D. F. e Hazley, M. P. (2014). Integrating Computational and Creative Thinking to Improve Learning and Performance in CS1. Em *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, página 475–480, New York, NY, USA. Association for Computing Machinery.

Moreira, M. A. e Masini, E. F. S. (1982). *Aprendizagem significativa: a teoria de David Ausubel*. Editora Moraes, São Paulo.

Nardelli, E. (2019). Do We Really Need Computational Thinking? *Communications of the ACM*, 62(2):32–35.

Ng, P. H. F. (2017). Introducing the Practices for Adopting the Constructivist Teaching in Game Engineering. Em *2017 IEEE Global Engineering Education Conference (EDUCON)*, páginas 1636–1643.

Nichols, M., Cator, K. e Torres, M. (2016). CBL Guide. Relatório técnico.

Nichols, M. H. e Cator, K. (2008). Challenge Based Learning. Relatório técnico.

Oliveira, C. M., Pereira, R., Galvão, L., Peres, L. e Schultz, E. (2019). Utilização de Desafios para o Desenvolvimento do Pensamento Computacional no Ensino Superior: um Relato de experiência. *Simpósio Brasileiro de Informática na Educação (SBIE)*, página 2005.

Oliveira, C. M., Pereira, R., Gasparini, I. e Boscarioli, C. (2022). An Overview of Computational Thinking in Higher Education: a technical report of a systematic mapping study. Relatório técnico.

Ortiz, J. e Pereira, R. (2018). Um mapeamento sistemático sobre as iniciativas para promover o pensamento computacional. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 29(1):1093.

Ortiz, J. S. B., Oliveira, C. M. e Pereira, R. (2018). Aspectos do Contexto Sociocultural dos Alunos estão Presentes nas Pesquisas para Ensinar Pensamento Computacional? Em *Anais dos Workshops do Congresso Brasileiro de Informática na Educação (CBIE 2018)*, volume 7, páginas 520 – 529.

Ortiz, J. S. B., Oliveira, C. M. e Pereira, R. (2019). Teaching Computational Thinking: are we considering students' socio-cultural context? *Journal on Computational Thinking (IJCThink)*, 3(1):3–18.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., McGuinness, L. A., Stewart, L. A., Thomas, J., Tricco, A. C., Welch, V. A., Whiting, P. e Moher, D. (2021). The prisma 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, 372.

Palts, T. e Pedaste, M. (2020). A model for developing computational thinking skills. *Informatics in Education*, 19(1):113–128.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., USA.

Pea, R. D. e Kurland, D. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, 2(2):137–168.

Pereira, R., Peres, L. e Silva, F. (2021). Hello world: 17 habilidades para exercitar desde o início da graduação em computação. Em *Anais do Simpósio Brasileiro de Educação em Computação*, páginas 193–203. SBC.

Peteranetz, M. S., Flanigan, A. E., Shell, D. F. e Soh, L. (2018). Helping Engineering Students Learn in Introductory Computer Science (CS1) Using Computational Creativity Exercises (CCEs). *IEEE Transactions on Education*, 61(3):195–203.

Peteranetz, M. S., Soh, L.-K. e Ingraham, E. (2019). Building Computational Creativity in an Online Course for Non-Majors. Em *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, página 442–448, New York, NY, USA. Association for Computing Machinery.

Petersen, K., Vakkalanka, S. e Kuzniarz, L. (2015). Guidelines for Conducting Systematic Mapping Studies in Software Engineering: an update. *Information and Software Technology*, 64:1 – 18.

Philip, M., Renumol, V. G. e Gopeekrishnan, R. (2013). A Pragmatic Approach to Develop Computational Thinking Skills in Novices in Computing Education. Em *2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, páginas 199–204.

Piaget, J. (1972). Intellectual Evolution from Adolescence to Adulthood. *Human development*, 15(1):1–12.

Piaget, J. (2002). *Epistemologia Genética*. WMF Martins Fontes.

Pollock, L., Mouza, C., Guidry, K. R. e Pusecker, K. (2019). Infusing Computational Thinking Across Disciplines: Reflections & Lessons Learned. Em *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, página 435–441, New York, NY, USA. Association for Computing Machinery.

Prensky, M. (2003). Digital Game-Based Learning. *Computers in Entertainment (CIE)*, 1(1):21–21.

Qin, H. (2009). Teaching Computational Thinking through Bioinformatics to Biology Students. *SIGCSE Bull.*, 41(1):188–191.

Qualls, J. A., Grant, M. M. e Sherrell, L. B. (2011). CS1 Students' Understanding of Computational Thinking Concepts. *J. Comput. Sci. Coll.*, 26(5):62–71.

Qualls, J. A. e Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. 25(5):66–71.

Rahman, M. M. e Paudel, R. (2018). Visual Programming and Interactive Learning Based Dynamic Instructional Approaches to Teach an Introductory Programming Course. Em *2018 IEEE Frontiers in Education Conference (FIE)*, páginas 1–6.

Resnick, M., Rusk, N. e Cooke, S. (1998). The computer clubhouse: Technological fluency in the inner city. *High Technology and Low-Income Communities*, páginas 1–17.

Rezende, C. M. C. e Bispo, E. L. (2018). Comparison Between the Use of Pseudocode and Visual Programming in Programming Teaching: an evaluation from Scratch Tool. Em *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–5.

Ritz, A. (2018). Programming the Central Dogma: An Integrated Unit on Computer Science and Molecular Biology Concepts. Em *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, página 239–244, New York, NY, USA. Association for Computing Machinery.

Rodriguez, B., Kennicutt, S., Rader, C. e Camp, T. (2017). Assessing computational thinking in cs unplugged activities. Em *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, página 501–506, New York, NY, USA. Association for Computing Machinery.

Rojas-López, A. e García-Peñalvo, F. J. (2018). Learning Scenarios for the Subject Methodology of Programming From Evaluating the Computational Thinking of New Students. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 13(1):30–36.

Romero, M., Lepage, A. e Lille, B. (2017). Computational Thinking Development Through Creative Programming in Higher Education. *International Journal of Educational Technology in Higher Education*, 14(1).

Román-González, M., Pérez-González, J.-C. e Jiménez-Fernández, C. (2017). Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. *Computers in Human Behavior*, 72:678–691.

Royal Society, G. B. (2012). *Shut Down or Restart?: the way forward for computing in UK schools*. Royal Society.

Rubinstein, A. e Chor, B. (2014). Computational Thinking in Life Science Education. *PLOS Computational Biology*, 10:1–5.

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P. e Saulters, C. (2010). Teaching Computational Thinking through Musical Live Coding in Scratch. Em *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, página 351–355, New York, NY, USA. Association for Computing Machinery.

Scott, A. e Barlowe, S. (2016). How Software Works: Computational Thinking and Ethics Before CS1. Em *2016 IEEE Frontiers in Education Conference (FIE)*, páginas 1–9.

Se, S., Ashwini B, Chandran, A. e Soman K.P (2015). Computational Thinking Leads to Computational Learning: Flipped Class Room Experiments in Linear Algebra. Em *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, páginas 1–6.

Senske, N. (2017). Evaluation and Impact of a Required Computational Thinking Course for Architecture Students. Em *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, página 525–530, New York, NY, USA. Association for Computing Machinery.

Shell, D. F., Hazley, M. P., Soh, L., Dee Miller, L., Chiriacescu, V. e Ingraham, E. (2014). Improving Learning of Computational Thinking using Computational Creativity Exercises in a College CSI Computer Science Course for Engineers. Em *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, páginas 1–7.

Shell, D. F., Hazley, M. P., Soh, L., Ingraham, E. e Ramsay, S. (2013). Associations of Students' Creativity, Motivation, and Self-regulation with Learning and Achievement in College Computer Science Courses. Em *2013 IEEE Frontiers in Education Conference (FIE)*, páginas 1637–1643.

Shell, D. F., Soh, L.-K., Flanigan, A. E., Peteranetz, M. S. e Ingraham, E. (2017). Improving Students' Learning and Achievement in CS Classrooms through Computational Creativity Exercises That Integrate Computational and Creative Thinking. Em *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, página 543–548, New York, NY, USA. Association for Computing Machinery.

Sherman, M. e Martin, F. (2015). The Assessment of Mobile Computational Thinking. *J. Comput. Sci. Coll.*, 30(6):53–59.

Shuiyan He, Yongmin Hang e Yi Ding (2014). Teaching Method Based on Computational Thinking a Case Research. Em *2014 9th International Conference on Computer Science Education*, páginas 817–820.

Shyamala, C. K., Velayutham, C. S. e Parameswaran, L. (2017). Teaching Computational Thinking to Entry-level Undergraduate Engineering Students at Amrita University. Em *2017 IEEE Global Engineering Education Conference (EDUCON)*, páginas 1731–1734.

Silva, I., Júnior, J. A. e Falcão, T. P. (2022). Panorama sobre iniciativas para promover o pensamento computacional no ensino superior brasileiro. Em *Anais do II Simpósio Brasileiro de Educação em Computação*, páginas 88–98, Porto Alegre, RS, Brasil. SBC.

Siva, S., Im, T., McKlin, T., Freeman, J. e Magerko, B. (2018). Using Music to Engage Students in an Introductory Undergraduate Programming Course for Non-Majors. Em *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, página 975–980, New York, NY, USA. Association for Computing Machinery.

Souza, A. A., Barcelos, T. S., Munoz, R., Silveira, I. F., Omar, N. e Silva, L. A. (2017). Self-organizing Maps to Find Computational Thinking Features in a Game Building Workshop. Em *2017 XLIII Latin American Computer Conference (CLEI)*, páginas 1–8.

Strauss, A. e Corbin, J. M. (1997). *Grounded Theory in Practice*. Sage.

Sveinsdottir, E. e Frøkjær, E. (1988). Datalogy - the copenhagen tradition of computer science. *BIT*, 28:450–472.

Tang, Y., Chen, W., Li, Z., Shi, L. e Wu, Y. (2018). Blended Learning of "Data Structure"Course Based on Computational Thinking. Em *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, páginas 403–406.

Taylor, K. (2013). Can Utilizing Social Media and Visual Programming Increase Retention of Minorities in Programming Classes? Em *2013 IEEE Frontiers in Education Conference (FIE)*, páginas 1046–1048.

Tedre, M. e Denning, P. J. (2016). The Long Quest for Computational Thinking. Em *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, páginas 120–129.

Tissenbaum, M., Sheldon, J. e Abelson, H. (2019). From Computational Thinking to Computational Action. *Communications of the ACM*, 62(3):34–36.

Torp, L. e Sage, S. (1998). *Problems as Possibilities: Problem-based Learning for K-12 Education*. Association for Supervision and Curriculum Development (Ascd).

Tsai, C.-W., Shen, P.-D., Tsai, M.-C. e Chen, W.-Y. (2017). Exploring the Effects of Web-mediated Computational Thinking on Developing Students' Computing Skills in a Ubiquitous Learning Environment. *Interactive Learning Environments*, 25(6):762–777. cited By 6.

Tsai, M.-C. e Tsai, C.-W. (2018). Applying Online Externally-Facilitated Regulated Learning and Computational Thinking to Improve Students' Learning. *Univers. Access Inf. Soc.*, 17(4):811–820.

Tseng, T.-H., Tai, Y., Tsai, S.-P. e Ting, Y.-L. (2018). Students' Self-authoring Mobile App for Integrative Learning of STEM. *The International Journal of Electrical Engineering & Education*, 0(0):0020720918800438.

Tucker, B. (2012). The Flipped Classroom. *Education next*, 12(1):82–83.

Turbak, F., Sandu, S., Kotsopoulos, O., Erdman, E., Davis, E. e Chadha, K. (2012). Blocks Languages for Creating Tangible Artifacts. Em *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, páginas 137–144.

Van Dyne, M. e Braun, J. (2014). Effectiveness of a Computational Thinking (CS0) Course on Student Analytical Skills. Em *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, página 133–138, New York, NY, USA. Association for Computing Machinery.

Venkatesh, V. e Bala, H. (2008). Technology acceptance model 3 and a research agenda on interventions. *Decision Sciences*, 39(2):273–315.

Vieira, C., Magana, A., Roy, A. e Falk, M. (2019). Student Explanations in the Context of Computational Science and Engineering Education. *Cognition and Instruction*, 37(2):201–231. cited By 3.

Villalba-Condori, K. O., Cuba-Sayco, S. E. C., Chávez, E. P. G., Deco, C. e Bender, C. (2018). Approaches of Learning and Computational Thinking in Students That Get into the Computer Sciences Career. Em *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM'18, página 36–40, New York, NY, USA. Association for Computing Machinery.

Vygotsky, L. (1978). Mind and Society: the development of higher mental processes.

Walden, J., Doyle, M., Garns, R. e Hart, Z. (2013). An Informatics Perspective on Computational Thinking. Em *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, página 4–9, New York, NY, USA. Association for Computing Machinery.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. e Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, 25(1):127–147.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3):33–35.

Wing, J. M. (2008). Computational Thinking and Thinking About Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3717–3725.

Wing, J. M. (2011). Research Notebook: Computational Thinking—What and Why? *The Link Magazine*, (March):June 23, 2015.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. e Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Wu, B., Hu, Y., Ruis, A. e Wang, M. (2019). Analysing Computational Thinking in Collaborative Programming: a quantitative ethnography approach. *Journal of Computer Assisted Learning*, 35(3):421–434.

Xu, Z. W. e Tu, D. D. (2011). Three New Concepts of Future Computer Science. *Journal of Computer Science and Technology*, 26(4):616–624.

Yang, H., Martin, P., Satterfield, D., Babbitt, R., Wong, J., Shelley, M. e Chang, C. K. (2011). A Novel Interdisciplinary Course in Gerontechnology for Disseminating Computational Thinking. Em *2011 Frontiers in Education Conference (FIE)*, páginas T3H–1–T3H–6.

Yeh, K., Xie, Y. e Ke, F. (2011). Teaching Computational Thinking to non-Computing Majors Using Spreadsheet Functions. Em *2011 Frontiers in Education Conference (FIE)*, páginas F3J–1–F3J–5.

Yin, R. (2014). *Case Study Research*. Applied social research methods series. SAGE Publications.

Yinnan, Z. e Chaosheng, L. (2012). Training for Computational Thinking Capability on Programming Language Teaching. Em *2012 7th International Conference on Computer Science Education (ICCSE)*, páginas 1804–1809.

Yuen, T. T. e Robbins, K. A. (2014). A Qualitative Study of Students' Computational Thinking Skills in a Data-Driven Computing Class. *ACM Trans. Comput. Educ.*, 14(4).

Zanetti, H., Borges, M. e Ricarte, I. (2016). Pensamento computacional no ensino de programação: Uma revisão sistemática da literatura brasileira. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 27(1):21.

Zorzo, A. F., Nunes, D., Ecivaldo, S. e Martins, S. (2017). *Referenciais de Formação para os Cursos de Graduação em Computação 2017*. Sociedade Brasileira de Computação (SBC.

APÊNDICE A – AN OVERVIEW OF COMPUTATIONAL THINKING IN HIGHER EDUCATION: A SYSTEMATIC MAPPING STUDY

Na literatura, inúmeras pesquisas investigam o Pensamento Computacional a partir de diferentes perspectivas e com diferentes públicos. Mesmo que o público-alvo da pesquisa concentrem-se no ensino fundamental e médio, há evidências de que estudantes de graduação em Computação também poderiam se beneficiar do exercício do Pensamento Computacional. Nesse sentido, realizamos um mapeamento sistemático da literatura para obter um panorama das iniciativas de Pensamento Computacional existentes no ensino superior. O conteúdo deste capítulo deriva do relatório técnico do mapeamento sistemático da literatura (Oliveira et al., 2022) e foi mantido no idioma Inglês, adotado para o relatório.

A.1 INTRODUCTION

This technical report presents a systematic mapping study to answer the following research question: *what is the research scenario in computational thinking in higher education?* Based on data extracted from 94 selected papers, the literature was categorized and mapped according to the definition adopted for the term computational thinking, the theoretical and methodological foundation, evaluation strategies and other relevant aspects that answered the main research question. Results from this systematic mapping show that computational thinking can support students' evolution regarding the understanding of computational concepts to reduce the courses dropout rate and support students to be secure to create and explore new solutions. The remaining of this technical report is organized as follows: In Section 2, the research design is presented, including the search in digital libraries, paper selection process, data extraction, and synthesis of the results. Then, results are discussed in Section 4 to answer our research question. Finally, Section 5 presents the conclusion and draw an agenda for future research.

A.2 RESEARCH DESIGN

The purpose of our systematic mapping was to investigate research on computational thinking to obtain an overview and characterize research initiatives in higher education. The systematic mapping was based on Petersen et al. (2015)'s guidelines and was carried out through five stages: (1) development of review protocol, (2) search in digital libraries, (3) paper selection, (4) data extraction, and (5) synthesis of results. The following sections will describe each stage.

A.2.1 Review Protocol

The first step in a systematic mapping study is to define a study protocol, which explains how the search will be conducted and guarantees that the same process will be applied to all the retrieved papers, reducing the researcher bias. In this stage, we structured the research protocol with research questions, data sources, search string, inclusion and exclusion criteria, and data extraction. The review protocol also has information about how to search into digital libraries, how to select the studies, and how to extract data from the selected studies.

To guide this systematic mapping study and answer our main question (*what is the research scenario in computational thinking in higher education?*), we defined six research questions that focus on identifying the research foundations, the benefits and difficulties reported,

the evaluation methods, tools, and techniques. The following research questions formed the basis of this systematic mapping:

> RQ01: Do studies define the term computational thinking? What are the adopted definitions?
>
> RQ02: What are the theoretical or methodological research bases?
>
> RQ03: How is computational thinking being taught or developed?
>
> RQ04: Why developing computational thinking?
>
> RQ05: What are the benefits and challenges of developing computational thinking in higher education?
>
> RQ06: How is computational thinking being evaluated?

A.2.2 Search in Digital Libraries

The search string for this systematic mapping was "*computational thinking*". We decided to use only the term due to the limitation in defining synonyms for "higher education". The search was realized in four digital libraries: ACM[1], IEEE Xplore[2], Scopus[3], and the library of the Brazilian Special Committee of Computing in Education (CEIE)[4]. We chose these digital libraries because of their relevance in our research area, well established in both areas of computer science and computer science education, and for addressing Brazilian and international scenarios. Our search covered peer-reviewed scientific papers written in English and Portuguese.

A.2.3 Paper Selection

The paper selection was conducted in two phases: 1st filter and 2nd filter. For the 1st filter, inclusion criteria were applied based on reading papers' titles and abstracts. For the 2nd filter, exclusion criteria were applied based on the full-text reading. A paper would be automatically included in the 2nd filter when a conclusion based only on the title and abstract reading was not possible. A researcher evaluated each paper and checked if the paper should be included or not, based on inclusion criteria. A second researcher was consulted to verify whether the paper should be included or excluded in case of doubts. The following inclusion (IC) and exclusion criteria (EC) were applied:

> IC01. The paper presents an initiative (i.e., activities, workshops, classes, dynamics, exercises) to teach or develop computational thinking in higher education.
>
> EC01. The paper has not been peer-reviewed, such as calls, books, abstracts, releases, editorials, manuscripts, and gray literature.
>
> EC02. The paper is not available.
>
> EC03. Duplicate papers (when the same paper is in more than one digital library).

[1]Available on: https://dl.acm.org/
[2]Available on: https://ieeexplore.ieee.org/
[3]Available on: https://www.scopus.com/
[4]Available on: https://br-ie.org/pub/

EC04. The paper presents only questionnaire or surveys analysis, exploratory studies or mappings, and systematic reviews.

EC05. The paper is not in English or Portuguese.

EC06. The paper did not attend the inclusion criteria.

A.2.4 Data Extraction and Data Analysis

A data extraction template was developed to guide data extraction, organize the information, and help the data analysis. The data extraction template has 21 extraction fields, and each of these has an expected value that helps answer a research question. Besides the data to answer the research questions, we collected basic information such as publication year, publication venue, and researcher institutions to have an overview of the selected papers. We reproduced the data extraction template into a spreadsheet to systematically extract information from each selected paper to store and analyze all data. The data extraction template is presented in Appendix B.

Before starting the data analysis, all data had to be normalized because each paper presented information differently. After the normalization, we separated the answers to each research question into different sheets for detailed qualitative coding analysis. The qualitative analysis was in three distinct phases. First, each of the three research questions (RQ03, RQ04, RQ05) was coded individually by me, one of the four authors. Through an iterative process, themes with few occurrences (small themes) were merged and collapsed into related themes. Finally, we find generic themes to discuss in-depth in the discussion section. The qualitative analysis to analyze the extracted data was based on two methods: a technique based on the Grounded Theory by Strauss e Corbin (1997), and Thematic Analysis by Clarke et al. (2015), and both methodologies are used to identify, analyze and interpret qualitative data.

A.2.5 Process Overview

The search in four digital libraries was performed in April 2019, and the search has returned 2,966 papers, published between 1987 and 2019. We did not define a search window because we wanted to map all the papers with the term computational thinking and manually select which papers present an initiative to teach or develop computational thinking in higher education. To overview the retrieved research on computational thinking, we analyzed the most frequent words used in the title of the 2,966 papers from the initial search: computational thinking, programming, learning, computer, and students were the six most common words. Figure A.1 shows the other 94 most common words, these words provide insight into which themes computational thinking is associated with.

Regarding the publication venues, Figure A.2(a) shows, computational thinking papers has been published mainly in conferences (67.4%) when compared to journals (32.6%). Figure A.2(b) shows the number of publications of papers on computational thinking, being possible to observe two interesting periods. Between 1987 and 1995, the first one reflects the origin and theory behind computational thinking: Seymour Papert used the term computational thinking for the first time in 1980, recognizing the computer as a powerful educational tool and an ally in the teaching and learning experience. The second one, after 2006, reflects the popularization of the term computational thinking. In 2006, Jeannette Wing published the ACM Viewpoint paper entitled *Computational Thinking*, giving huge visibility to the topic. Further, her viewpoint can be considered one of the main reasons for the increasing popularity of the topic and, consequently, in research growth. The low number in 2019 may be related to the fact that several conference editions have not happened yet when we searched in digital libraries (April/2019).

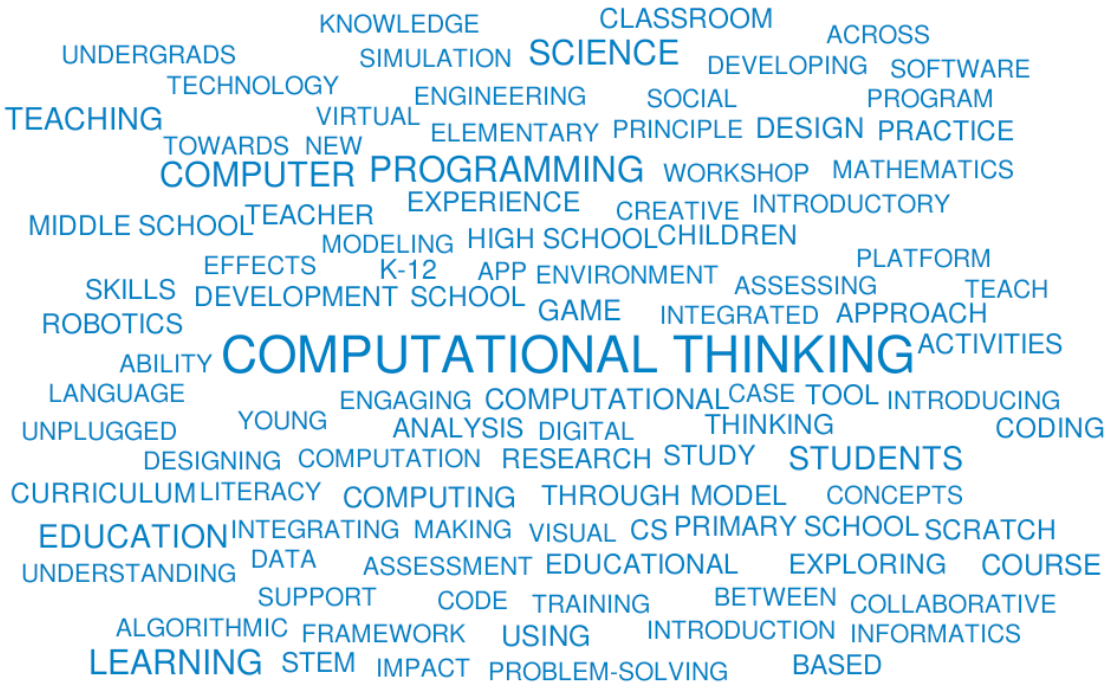
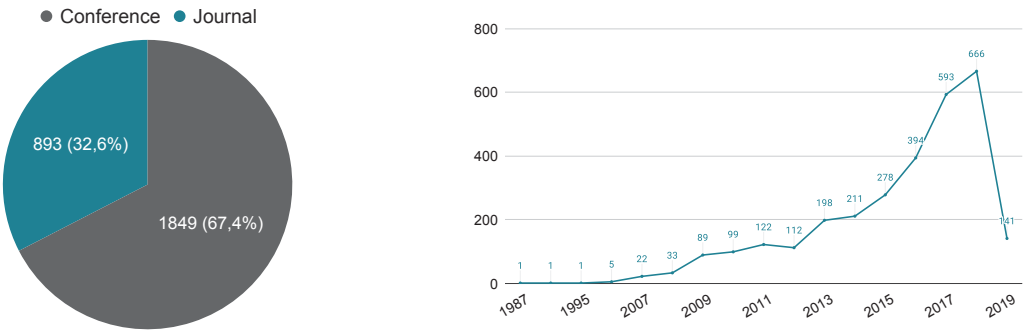


Figura A.1: Most 100 frequent words used in the title of 2,966 papers

The initial search results were 2,966 papers. Since it was a considerable number of papers, we implemented a two-stage filtering process: 1st filter and 2nd filter. In the 1st filter, inclusion criteria were applied based on reading papers titles and abstracts. This process resulted in 228 papers being selected for the next stage. The 2nd filter was a process of reading through all papers and extracting data from the selected papers. At the end of the selection process, we have 94 relevant papers for our systematic mapping. The papers duplication check across the digital libraries was done using title-based sorting in a spreadsheet. In case of duplication, we considered just one of them. Figure A.3 shows the PRISMA Flow Diagram (Page et al., 2021) for the systematic mapping study, which illustrates the overview of the paper selection process.



(a) Publication venues

(b) Publications per year

Figura A.2: Search returned papers overview: publication venues and publications per year

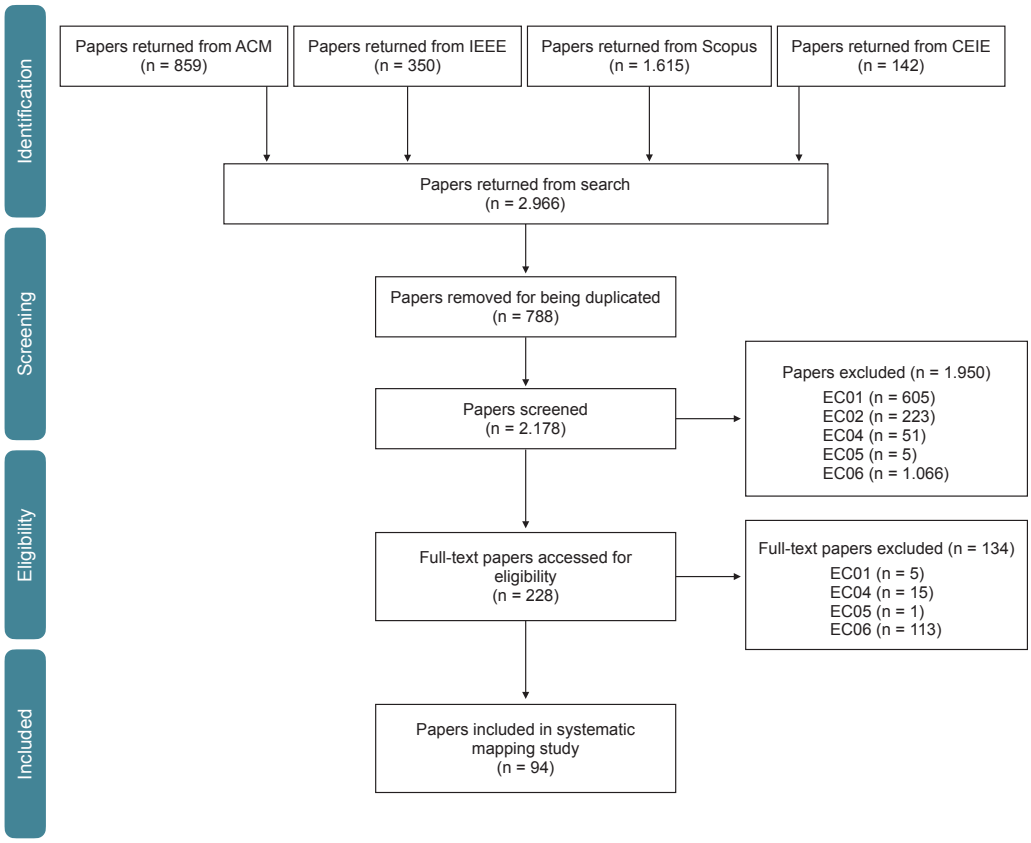


Figura A.3: PRISMA Flow Diagram for the systematic mapping study

A.3 RESULTS AND DISCUSSION

This section presents the general results from the systematic mapping study, and the results are organized according to the five research questions, as presented in Section 2.1.

A.3.1 Synthesis of Result

A total of 94 peer-reviewed papers written in English or Portuguese were selected for our systematic mapping study. The research scenario about computational thinking in higher education involved 302 researchers, from 110 universities in 24 countries. Some papers were co-authored by researchers from different universities in different countries, so in this case, we counted the paper for all involved countries. Figure A.4 displays the research distribution around the world. As indicated, United States (US) is the country that has the highest amount of published papers on computational thinking, 50 papers (53.1%). Consequently, US is the country that has involved the largest number of authors and institutions: 154 authors and 48 institutions. China and Brazil are the second countries with the highest number of published papers in this systematic mapping, with 7 papers each. Among these Brazilian papers, 5 were written in English and 2 in Portuguese.

The university with the largest number of papers about computational thinking in higher education is the University of Nebraska-Lincoln, with 8 papers and 40 researchers. The papers investigate the combination of computational thinking with creative learning and the use of creative exercises Miller et al. (2014); Shell et al. (2017); Peteranetz et al. (2019); Flanigan

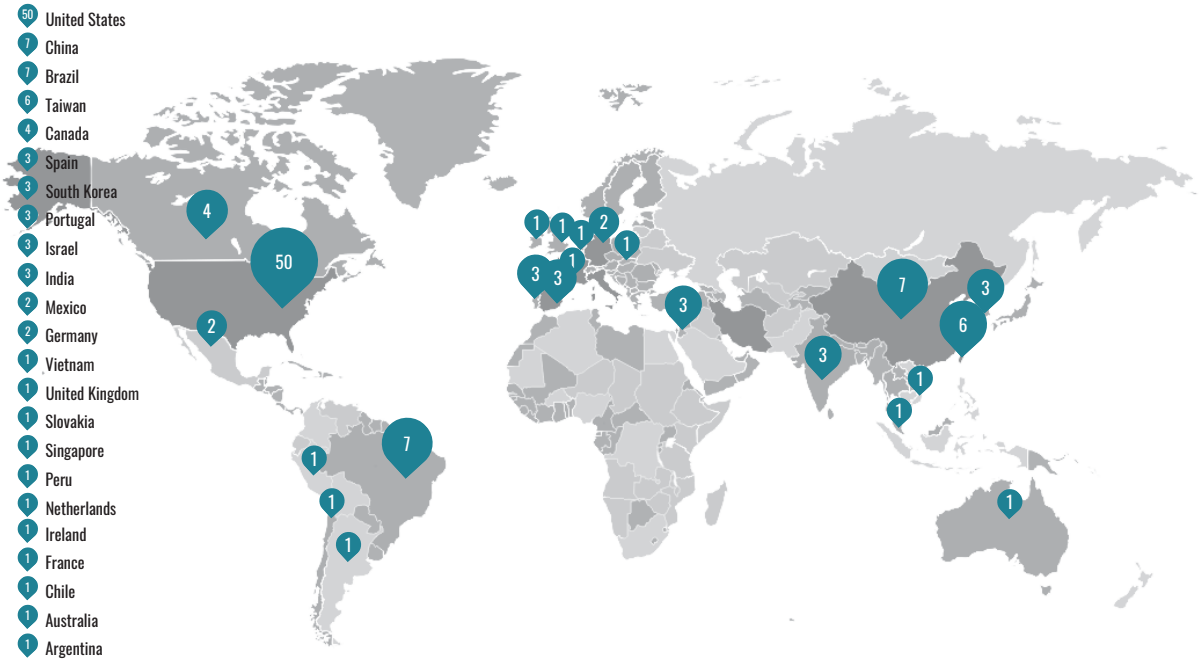


Figura A.4: Distribution of selected papers around the world

et al. (2015); Miller et al. (2013); Shell et al. (2014, 2013); Peteranetz et al. (2018) . The main purpose of these papers is to investigate how this combination can motivate students and help them develop emotional intelligence. To test computational thinking, they use creative exercises in the Introductory Computer Science course (CS1). The papers reported that 1,944 students were exposed to the researchers' activities, and all of them were undergraduate students in STEM. Figure A.5 presents the institutions with at least two papers.

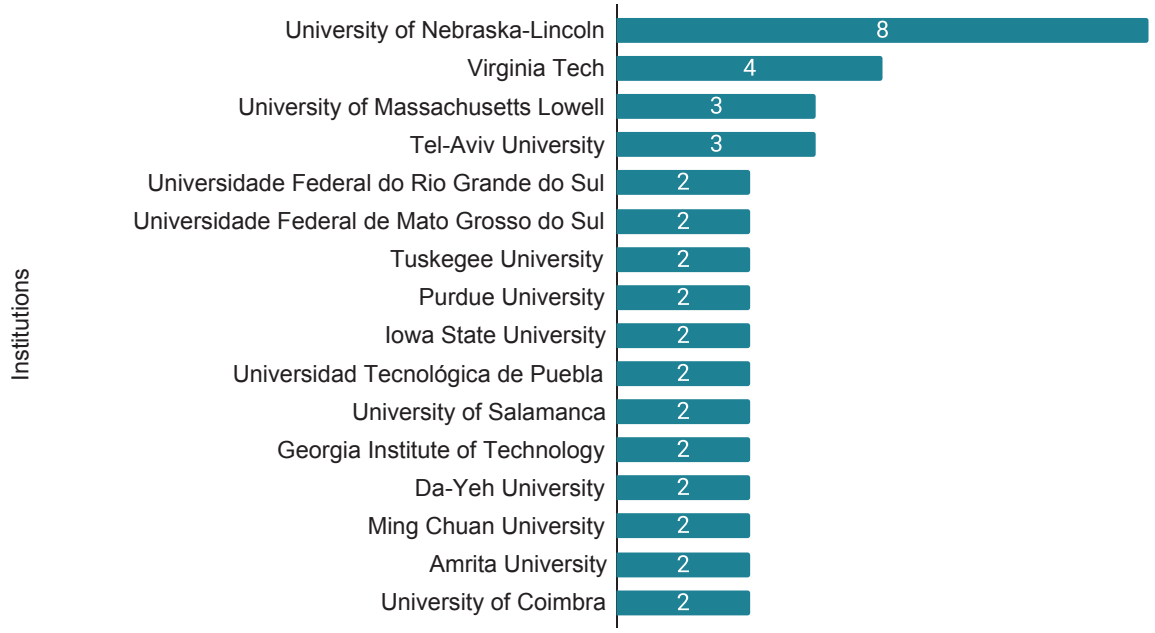
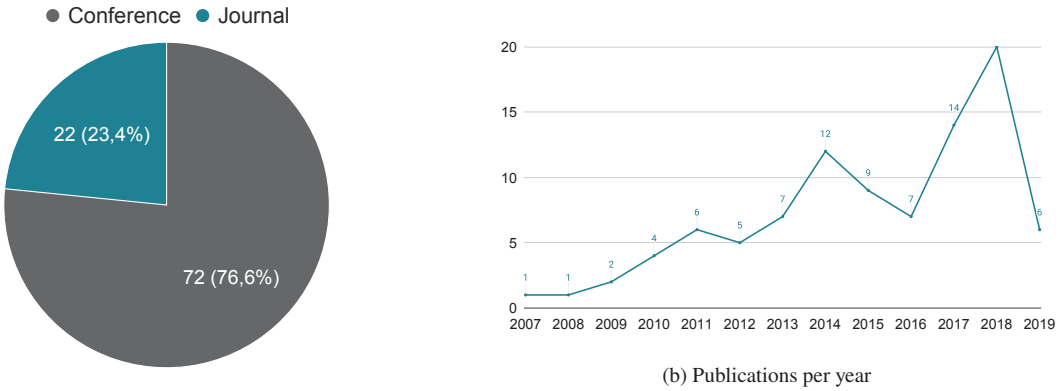


Figura A.5: Names of institutions with at least two papers



(a) Publication venues

(b) Publications per year

Figura A.6: Selected papers overview: publication venues and publications per year

Regarding publication venues, the selected papers originated from 45 different publication venues – 30 conferences and 15 journals. Considering the 94 selected papers, 72 (76.6%) were published in conferences, whereas 22 (23.4%) in journals (Figure A.6(a)). Figure A.7 provides an overview of papers distribution by publication venues. The order in Figure A.7 is based on the number of published papers, considering venues with at least 2 publications. The selected papers were published between 2007 and 2019. The annual quantity of published papers about computational thinking increased over the years, and also increased research about computational thinking in higher education (Figure A.2(b)). Although, in 2015 and 2016, the papers about the theme decreased. A list of the selected papers is documented in Appendix C.

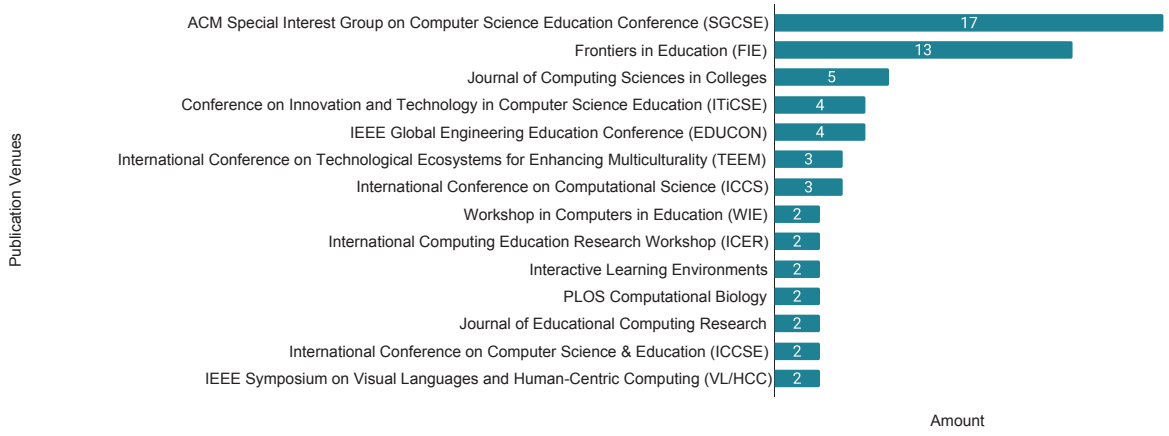


Figura A.7: Overview of papers distribution by publication venues

Although the search returned papers published between 1987 and 2019, in our systematic mapping study the computational thinking's first reach in higher education was published in 2007 Cortina (2007). A North American researcher from Carnegie Mellon University subscribed to Wing (2006)'s computational thinking definition and used unplugged activities to teach students to think computationally. At this time, both researchers were working at the same university, which may have influenced the research. Ambrósio et al. (2014) was the first Brazilian paper in the theme, appeared only in 2014, reporting a cognitive assessment test for computational thinking based on the navigation of a simple map.

A.3.2 RQ01: Do the papers define the term computational thinking? What is the definition?

From the 94 selected papers, 18 papers (19.1%) did not present any definition for computational thinking. The primary reference for computational thinking definition was Wing's research (2006; 2008; 2011): 75 papers (79.7%) cited Wing's definition, 64 (68%) of which used Wing's viewpoint paper published in 2006. Other seven papers mixed Wing's definition with another definition, and only three papers used a different author to define computational thinking. When a definition was presented, computational thinking was defined based on at least one of these references: Tedre e Denning (2016), Weintrop et al. (2016), García-Peñalvo (2016), Royal Society (2012), Barr e Stephenson (2011), ISTE e CSTA (2011), Wing (2011), Cuny et al. (2010), Council et al. (2010), Qualls e Sherrell (2010), Carnegie Mellon University (2010), Wing (2008, 2006), Resnick et al. (1998).

To understand how computational thinking has been understood in the literature, we analyzed the definitions presented in the papers. At the end of the coding process, it was perceived that computational thinking can be considered a process, a mental tool, or a skill associated with problem-solving systems design and understanding human behavior. Table A.1 shows the twelve definitions of computational thinking presented in each paper, the references for each definition, and the paper id that used the definition.

Tabela A.1: Computational thinking definitions.

Definitions	**References**	**Papers**
Computational thinking is the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.	Cuny et al. (2010) Wing (2011)	Papers [1], [8], [39], [46], [54], [58], [62].
Computational thinking is the study of an effective approach used by people to solve problems, and involves the systematic evaluation of information and is a crucial piece of problem-solving.	Wing (2008)	Papers [7], [22], [32], [42], [48], [50], [79].
Computational thinking is an intellectual skill rooted in the ability to conceive of meaningful information-based representations that can be effectively manipulated using an automated agent (e.g., a computer).	Wing (2006) ISTE e CSTA (2011)	Paper [16].
Computational thinking is a fundamental skill not just for computer students but for everyone and should be used in all areas, a commonplace.	García-Peñalvo (2016) Wing (2011)	Paper [30].
Computational thinking involves solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.	Wing (2006)	Papers [4], [5], [6], [9], [11], [12], [13], [14], [15], [17], [19], [21], [23], [24], [26], [27], [28], [29], [33], [34], [37], [38], [40], [42], [44], [45], [46], [48], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [64], [66], [67], [68], [70], [71], [73], [74], [75], [76], [77], [80], [83], [85], [87], [88], [89], [90], [91], [92].
Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science. To flourish in today's world, computational thinking has to be a fundamental part of the way people think and understand the world. Computational thinking means creating and making use of different levels of abstraction to understand and solve problems more effectively. Computational thinking means thinking algorithmically and with the ability to apply mathematical concepts such as induction to develop more efficient, fair, and secure solutions. Computational thinking means understanding the consequences of scale, not only for reasons of efficiency but also for economic and social reasons.	Carnegie Mellon University (2010)	Paper [20].

Definitions	References	Papers
Computational thinking is a fundamental competence of the 21st century, such as writing or reading. It refers to a series of capacities and skills that can be stimulated, facilitating the teaching-learning process in several areas, such as mathematics.	Resnick et al. (1998)	Paper [86].
Computational thinking is a way that allows us to create solutions to problems through the use of skills such as abstraction, decomposition, generalization, evaluation, and algorithmic design.	Wing (2006) Barr e Stephenson (2011) García-Peñalvo (2016) Royal Society (2012)	Paper [25].
Computational thinking is the complementary term that relates to the thought processes involved in computational research, including designing computational models and using computational instruments. Computational thinking is not only amounts to the ability to write code but also comprises the skills to analyze a problem and solve it effectively.	Wing (2006) Tedre e Denning (2016) Council et al. (2010)	Paper [47].
Computational thinking is an approach to problem-solving that consolidates logic skills with core computer science concepts.	Qualls e Sherrell (2010)	Paper [55].
Computational thinking focuses on the ability to manage complex situations by generating abstractions and maintaining the relationships among them. To generate abstractions, we need to have formal representations that reflect our cognitive processes and structures (discerning aspects of the situation). This characteristic of CT will enable us to generate hypotheses and search for a plausible solution systematically. During problem-solving, we consistently evaluate the current situation against our previous experience or our prediction until the best solution is reached.	Council et al. (2010)	Paper [65].

According to the mapped definitions, computational thinking is a human skill that can be stimulated and helps in teaching and learning in different areas, including a variety of mental tools that reflect the breadth of the Computer Science field. This skill includes everything from analyzing the problem to formalizing the solution, using abstraction, decomposition, generalization, evaluation, and algorithmic design skills. The first step in formalizing a solution is to identify which information processing agent will execute it, then represent the solution in a way that an agent can execute it. An information processing agent can be either a machine or a human. Therefore, the problem-solving process does not necessarily need to involve computational technology, which is entirely associated with performing abstractions at different levels to understand and solve problems more efficiently. Figure A.8 presents an overview of computational thinking definition, references, and the paper id that used each reference.

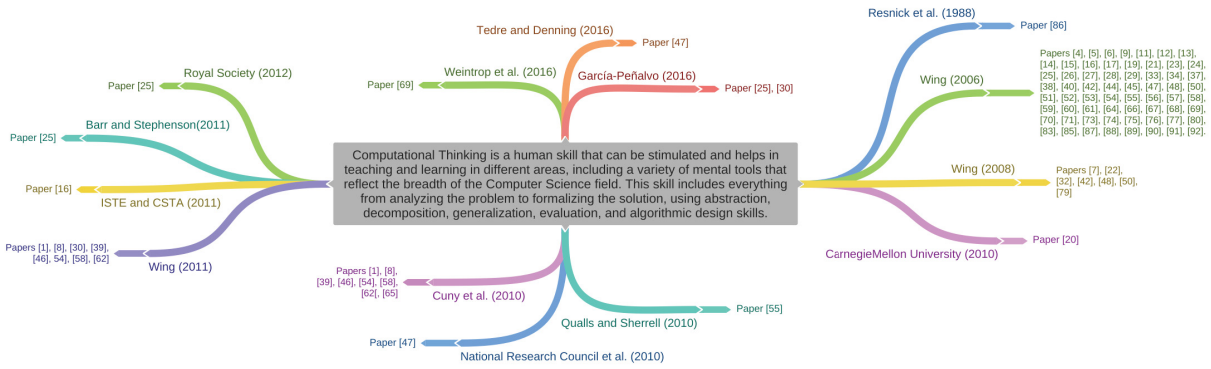


Figura A.8: Overview of computational thinking definition

A.3.3 RQ02: What are the theoretical or methodological research bases?

Vygotsky (1978), Piaget (1972, 2002) Freire (1997, 2014), Papert (1980), and Epstein (1990, 1996) were the theoretical bases pointed out by the papers. Piaget's genetic epistemology explains

the order of succession in which different cognitive capacities are built, caring for the study of how knowledge originates and develops. Piaget states that a person has an active role in creating and updating their knowledge, which he names Constructivism. Piaget also affirms that this construction of knowledge occurs through investigation. Papert (1980) coined the term Constructionism, which explains that knowledge can be built with a tool's aid, in this case, the computer. Papert recognized the computer as a stimulator of ideas and envisioned using it to innovate the educational system. Therefore, Papert's constructionism can be considered an approach to Piaget's Constructivism.

Vygotsky was a pioneer in arguing that children's intellectual development occurs due to social interactions and living conditions. For him, teachers are responsible for creating the possibilities for learning to occur. The exchange of experience between students and educators is the main point of the pedagogical process. For Vygotsky, the educational process needs to be dynamic and related to life, and life needs to be part of the school (Vygotsky, 1978). Freire was a great Brazilian educator, and he shows the importance of using information from the context that students are already familiar with it. Thus, prior knowledge contributes significantly to the learning of these students Freire (1997). Freire believed that education could not be disassociated from politics as soon as teaching and learning are considered political acts in themselves. Furthermore, Freire claimed for a liberating education and defended the transformation of reality through education Freire (2014). In this sense, Freirean pedagogy states that both sides, who teach and learn, will learn together in the classroom. Another theoretical basis cited was Epstein's Theory of Generativity, a formal and predictive theory of people's creative behavior. Epstein (1996) stated that new ideas result from the connection between the old ones and that interaction occurs dynamically between previous behaviors. The theory also suggests that anyone can learn to be creative, and there are strategies to enhance that creativity.

For methodological bases, Project-Based Learning (Blumenfeld et al., 1991), Collaborative Learning (Gokhale, 1995), Contextualized Learning (Cordova e Lepper, 1996), Game-Based Learning (Prensky, 2003), Problem-Based Learning (Hmelo-Silver, 2004), Flipped Classroom (Tucker, 2012), and Unplugged Learning (Bell et al., 2012) were the most common ones. Collaborative learning was presented as a method in which students work together in small groups in search of a common goal to allow students to build knowledge through collaboration between them (Dillenbourg, 1999). The focus of collaborative learning is to interact collectively, in which the group is responsible for each participant's learning.

The Flipped Classroom is a teaching methodology that inverts the traditional classroom model, where teachers stop being positioned in front of the class to explain the matter to the students. Based on this inversion, students previously study the content that will be worked in the classroom and use the face-to-face moment to answer questions and do activities (Tucker, 2012). Game-based Learning is a methodology that allows the creation and use of games for educational purposes, and it provides a different teaching experience to students. Incorporating the elements of games into the classroom makes it easier for students to learn content in a more interesting way (Prensky, 2003).

Problem-Based Learning is a learning methodology that uses problem-solving to acquire knowledge, bringing students to the center of learning, making them responsible for obtaining their knowledge (Hmelo-Silver, 2004). Thus, students become active in their learning experience, and teachers become facilitators, provoking, challenging, motivating. Unplugged learning is very much associated with computing and can also be known as unplugged computing. The Unplugged approach is a methodology that seeks to teach the general concepts of Computer Science in a disconnected way (Bell et al., 2012). Unplugged activities seek to communicate the

contents without the distraction of computers and expect the use of different techniques, such as dramatization, puzzles, games, drawings, and challenges.

Contextualized Learning enables students to build meaning for their learning based on their own experiences, focusing on teaching and learning through concrete applications in a specific context that is of interest to students. One way to generate meaning and motivation for learning is to use real problems close to the student's reality (Cordova e Lepper, 1996). Project-Based Learning is a teaching method designed to involve students in investigating authentic and complete problems. Contents proposed to be taught are related to the project, and during the development, contents are presented and worked on (Blumenfeld et al., 1991).

A.3.4 RQ03: How is computational thinking being taught or developed?

To understand how computational thinking is being taught, we extracted data about the initiatives reported in the selected papers regarding their content and theme, where they took place, target audience, duration, tools, and computational thinking skills they were intended to promote. As these are academic initiatives, most activities (76 - 80.8%) took place in person in classrooms or laboratories at universities. Ten initiatives were entirely online, and eight did not report the location. Regarding the initiatives' duration, 53 (56.3%) occurred during one semester, while 16 initiatives were of short duration (up to 50 hours), and 25 (29.5%) did not report the duration.

When verifying the contents or topic explored in the mapped initiatives, computational thinking appeared significantly associated with computer programming: from the 94 papers, 50 (53.1%) papers reported fundamentals of programming as the content explored in their initiatives. Other contents were mentioned, such as foundations of Computer Science (10 papers), creating solutions for the health area (2 papers), or infrastructure and hardware (3 papers). However, even if the term "computational" refers to the computer and allows an analogy with "programming", they cannot be understood as synonyms Oliveira et al. (2019). Computational thinking is for programming in the same way as programming is for Computer Science, both are only part of the field and do not correspond to the whole area. Figure A.9 shows the contents of the initiatives and the number of papers reporting them.

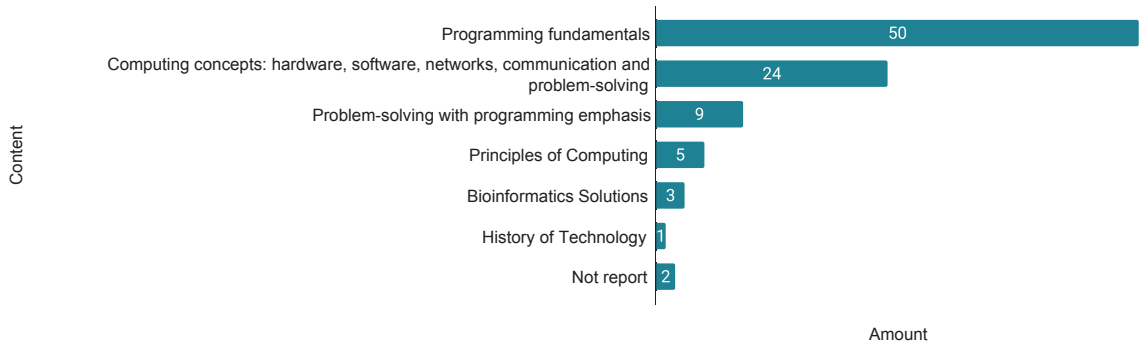


Figura A.9: Initiatives' Contents

As expected, tools used in initiatives have a strong relationship with their contents. Of the total, 81 (86.1%) papers mention a tool or a set of tools used, and 13 (13.8%) papers did not mention the use of any tool. Figure A.10 shows the list of tools used, and it is possible to see that the most used tools are programming languages and visual programming languages (block-based). The most used tools may be directly related to the fact that programming fundamentals were the most recurring content among the initiatives. Even for initiatives conducted in person, a

virtual learning environment (i.e. Moodle) was used to support the course and make the content available online, as well, to allow students to interact with each other outside the course.

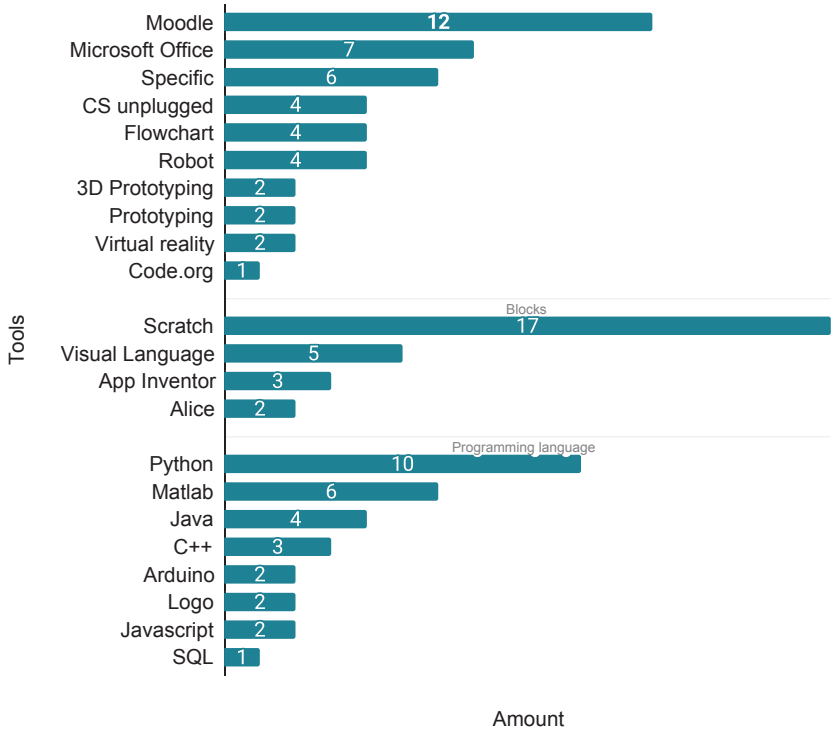


Figura A.10: Tools used in the initiatives

As presented in RQ01, there is no unanimity on computational thinking definition, and there is also no relationship between what skills it exercises. According to (Brackmann et al., 2016), abstraction, decomposition, pattern recognition, and algorithmic design are the four pillars of computational thinking. These pillars are high-level skills that guide the development of computational thinking, and, in the literature, they still have a lack of references about low-level computational thinking skills. To identify the computational thinking skills, we extracted all the skills explicitly mentioned in each selected paper. Unfortunately, only 39 (41.4%) papers report the exercised skills. Among the papers that report the exercised skills, abstraction is present in 31 papers. This significant amount of papers mentioning abstraction may be due to the fact that abstraction is considered the essence of computational thinking. Figure A.11 shows the skills mapped and the number of papers mentioning their initiatives exercised each of them.

Regarding the target audience of the reported initiatives, papers usually characterize the participants very briefly. The characterization of initiatives is usually limited to the courses' names and the number of people who have participated. While 74 (78.7%) papers show the number of participants and 91 (96.8%) show the name of the course where the initiative happened, only six papers show the participants' age, and 30 (31.9%) papers show their gender. In only five papers the female audience was larger than the male audience Peteranetz et al. (2019); Libeskind-Hadas e Bush (2013); Lin et al. (2018); Almeida e Pessoa (2017); Almeida et al. (2018). Analyzing the relationship between gender and course, the initiatives that have higher female participation occurred in courses in education, biology, and communication, while those with a

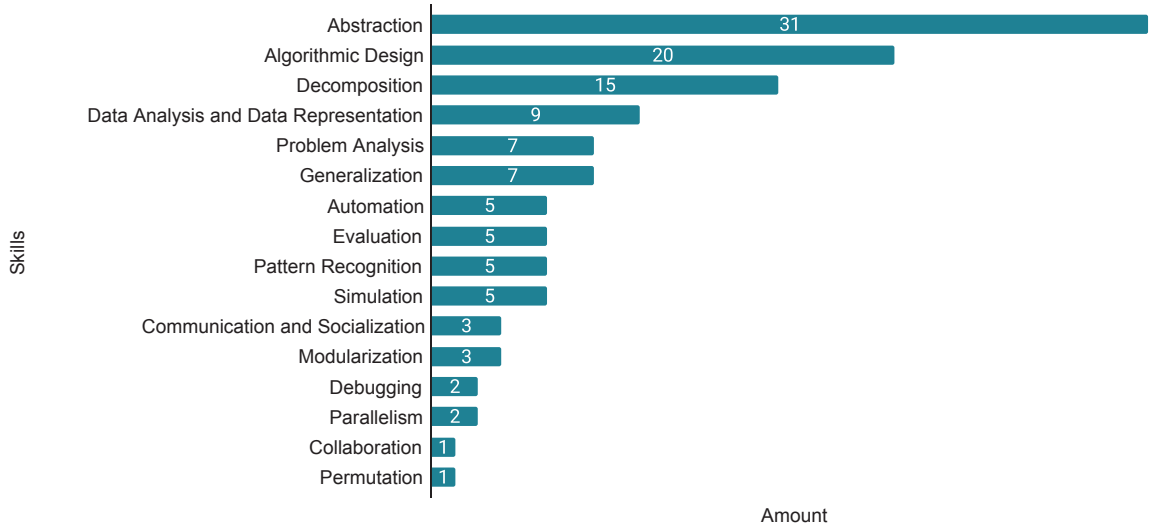


Figura A.11: Skills and amount of papers that mention each skill (*n=39*)

majority of male participants occurred in STEM[5] courses. In these papers, the gender is still limited to male and female, and only one paper presents the number of non-binary participants.

The low number of women in initiatives taken in STEM courses exposes the necessity to support women's entry and permanence in these areas. This low quantity also reveals how research in Computer Education should be developed through the eyes of inclusion and equity. The insufficient characterization of initiatives and participants reveals how the authors are not concerned with detailing as much as possible when they report the research. Providing a detailed description of both the initiatives and the participants helps to understand the initiative better, justifying the choices, methods, and activities. We must remember that research involving people has some particularities, and these make each research unique. The more detailed the research, it will be easier to understand, and there will have more possibilities for it to be continued and reproduced by other researchers.

A.3.5 RQ04: Why developing computational thinking?

Unfortunately, not all papers explicitly mention the purpose of their studies. Some of them let their purpose be understood, while others specify the goal and still explain why. From the data analysis, we discovered that the main goal in developing computational thinking is to learn how to convert a real problem into a computational solution and this corroborates with the definition of computational thinking presented in RQ01. However, when it mentions "computer solutions" it is associated with the computer, but it must be interpreted as an information processing agent – it can be technological or human. Even though computational thinking is related to Computer Science, it can be associated with any area and course in a wide range of topics and disciplines. Therefore, developing computational thinking skills is associated with providing students with a better understanding of Computer Science topics and concepts to support them when building a solution, regardless of area and content.

Some initiatives used computational thinking to teach students how to convert a real problem into a computational solution Larkins e Harvey (2010); Libeskind-Hadas e Bush (2013);

[5]STEM is the abbreviation for Science, Technology, Engineering, and Math.

Goodman e Dekhtyar (2014); Buckingham e Hogan (2014); Tang et al. (2018); Lester (2008); Hurson e Sedigh (2010), others presented computational thinking for Science courses Kafura e Tatar (2011); Hambrusch et al. (2009); Ritz (2018); Senske (2017); Libeskind-Hadas e Bush (2013); Goodman e Dekhtyar (2014); Buckingham e Hogan (2014). In addition to developing computational thinking, six initiatives explored the direct benefits of developing these skills to support programming learning Babbitt et al. (2019); Kazimoglu et al. (2012); Tang et al. (2018); Krugel e Hubwieser (2017); Kurniawan et al. (2018); Yinnan e Chaosheng (2012). Another four initiatives seek to understand the relationship between computational thinking and programming learning, the challenges and benefits of developing computational thinking to support computer programming learning by beginning students Babbitt et al. (2019); Villalba-Condori et al. (2018); López e García-Peñalvo (2016); Yuen e Robbins (2014); Wu et al. (2019). Five initiatives used computational thinking as a basis for activities that sought to support and engage students [Batista et al. (2016); Chowdhury et al. (2018); Wu et al. (2019); Shell et al. (2014, 2013). Eight initiatives aimed to provide students with a better understanding of Computer Science topics and concepts Kafura e Tatar (2011); Gardner et al. (2011); Kafura et al. (2018); Siva et al. (2018); Mason et al. (2016); Rubinstein e Chor (2014); Kim et al. (2014); Li et al. (2015).

A.3.6 RQ05: What are the benefits and challenges of developing computational thinking in higher education?

Of the 94 selected papers in this systematic mapping, 78 (82.9%) papers mentioned benefits, 34 (36.1%) mentioned difficulties, and only 30 (31.9%) papers mentioned both benefits and difficulties faced to develop computational thinking in their reported activities. Descriptions of initiatives' difficulties and benefits are related to how the research was conducted and to the students' experience. Analyzing the data extracted to answer this RQ revealed that papers are more focused on reporting the results obtained than the process. This observation corroborates with RQ03, where we described that papers do not provide many details about the method applied to conduct the activities. The next subsections will detail the challenges and benefits of developing computational thinking in higher education.

A.3.6.1 Challenges to develop computational thinking in higher education

Challenge 1 - Generalizing results. Whereas the number of students who were in the initiatives (sample size) is too small to justify the generalization of the results. Thus, it is not easy to have enough information to support the extension and evolution of the activities they used. Miller et al. (2014) reported difficulties generalizing results because although the study involved four courses, it happened in just one semester. As for Yuen e Robbins (2014), Figueiredo et al. (2016), Wu et al. (2019), and Tseng et al. (2018), the main difficulty was the classes' size, too small, limiting the generalization.

Challenge 2 - The interdisciplinary relationship of computational thinking. Besides, Miller et al. (2010), Flanigan et al. (2015), Tsai e Tsai (2018), Miller et al. (2013) and Rojas-López e García-Peñalvo (2018) reported that students from other majors do not have basic knowledge of computing, limiting the understanding of the concepts used as a basis for computational thinking. Qin (2009) reported that the computer is technically challenging because the students are uncomfortable during one-to-one interaction with a computer during exercises. Also, students often have problems with spelling, and many errors occur due to mistype commands and parameters. To address the issue that many students in life sciences are often weak in computing skills and tend to avoid computing-orient courses, they incorporated computational thinking into

a bioinformatics course for life science majors. Other problems, such as lab setup and hardware problems were mentioned by Buckingham e Hogan (2014) and Kurniawan et al. (2018).

Challenge 3 - Evaluating and formalizing Computational Thinking Solutions. Vieira et al. (2019) reveals a lack of a valid instrument to assess students' understanding, while Romero et al. (2017) adverts that automatic analysis tools are limited and human interference is necessary to evaluate the development of computational thinking. *Challenge 4 - Expressing and formalizing solutions*. Cápay (2014) described that some students had problems expressing their solutions on paper, and they could not create any formal description of their algorithms because they tried to write the final solution without documenting their previous attempts. Also, they were not accustomed to thinking about exceptions or restrictions in data inputs (Cápay, 2014). As well, Yeh et al. (2011) reported that students presented difficulties in making computational representations. While for initiatives with programming content, the main difficulty was that different programming aspects confused students (Barba e Chancellor, 2015; Bryant et al., 2011; Davies, 2008). Davies (2008) reported the dependence with the programming language, and learning the concepts of programming without a programming language does not seem to result in greater general competence.

Challenge 5 - Student engagement. Twelve papers reveal that student engagement is a challenge, and they also claim that activities carried out have a direct relationship with students' motivation and engagement. In the paper Larkins e Harvey (2010), students pointed out that they needed more context on the subject before carrying out the tasks, while for Almeida et al. (2018), they assessed the time of the activities as insufficient. Engagement among students is low when they do not know each other, least of all with those who do not attend all classes (Kim e Kim, 2017). There are also difficulties in understanding the relationship between motivation and grade, where students who are not motivated with the discipline scored poorly and showed little success (Shell et al., 2017). Some students only felt motivated for activities that would give a grade (Gardner et al., 2011).

A.3.6.2 Benefits of developing computational thinking in higher education

The benefits presented by the papers range from the evolution of students regarding the understanding of computational concepts to the reduction in the courses dropout rate. Also, some papers state that the methodology used directly impacts students' learning. *Benefit 1 - Students' confidence and evolution*. As pointed for Miller et al. (2014), Gero e Levin (2019), and Buckingham e Hogan (2014), students felt more comfortable because they did not need to learn a specific programming language. These students had better results in unplugged activities. In those papers, they used computational thinking as an approach to programming fundamentals and to introducing CS. Thus, students learned to understand the problem's requirements and identify its essence (Kim e Kim, 2017; Vieira et al., 2019; Philip et al., 2013; Tang et al., 2018). Furthermore, papers report the students' evolution regarding understanding Computer concepts.

Benefit 2 - Students' motivation. Another benefit highlighted was students' motivation and self-regulation. Papers reporting studies with a more flexible methodology mentioned that students developed security to create and explore new solutions, mostly because they had space for that (Sherman e Martin, 2015; Figueiredo et al., 2016). Thirteen papers showed that activities carried out significantly increased engagement, enthusiasm, motivation, intention to persist, and progression. Two initiatives revealed that students were motivated when they got more tangible results Turbak et al. (2012); Ng (2017), and 7 of them claimed that teamwork kept students motivated (Peteranetz et al., 2019; Siva et al., 2018; Jung et al., 2017; Gossen et al., 2018; Scott e Barlowe, 2016; Rahman e Paudel, 2018; Cápay, 2014). According to Kazimoglu et al. (2012) and Almeida et al. (2018), working in groups increases students' motivation because it is an

opportunity to learn together. The interaction between students revealed that they could learn from each other, especially when they provide or receive feedback from colleagues, which keeps students actively involved in the learning experience.

Benefit 3 - Promote Students' critical thinking and problem-solving skills. In addition to supporting computational thinking development, six papers managed to promote students' critical thinking and problem-solving (Walden et al., 2013; Yuen e Robbins, 2014; Bui et al., 2018; Peteranetz et al., 2019; Lai et al., 2019; Shell et al., 2014). The reported activities helped students to improve their computing skills and understand their applicability (Chowdhury et al., 2018; Qualls et al., 2011). Making students aware that programming is related to problem-solving and showing that knowledge and solution can be applied outside the classroom has brought meaning to students' learning (Chilana et al., 2015). Other papers reported that developing computational thinking helps students to expand their knowledge of programming (Shell et al., 2017; Chowdhury et al., 2018; Haddad e Kalaani, 2015; Miller et al., 2013; Rahman e Paudel, 2018). Also, Kafura e Tatar (2011) and Gardner et al. (2011) revealed that students developed a greater understanding of the covered concepts, expanded the vocabulary, and communicated better. Another significant benefit is that computational thinking has the potential to help students during the course Miller et al. (2010); Van Dyne e Braun (2014); Kim et al. (2014); Davies (2008): the four papers reported a significant improvement in the students' logic. Moreover, activities to develop computational thinking helped increase the retention rate and decrease the course failure rate. Shell et al. (2017), Haddad e Kalaani (2015), Miller et al. (2013) and Rahman e Paudel (2018) observed an increase in the courses' average grades after the initiative.

A.3.7 RQ06: How is computational thinking being evaluated?

The papers included in this systematic mapping reported different data collection methods to assess computational thinking development. Among the 94 selected papers, 14 (14.8%) of them do not describe whether there was an evaluation of computational thinking development. Of the remainder, 54 (57.4%) papers mention using only one data collection resource, and 26 (27.6%) associate more than one method. Tests were the most used data collection method (45 papers - 47.8%) since the research conducted a pre-test and a post-test to compare the students' evolution. The combination of tests and questionnaires was the most used assessment method (12 papers - 12.7%). Figure A.12 shows the primary data collection methods used.

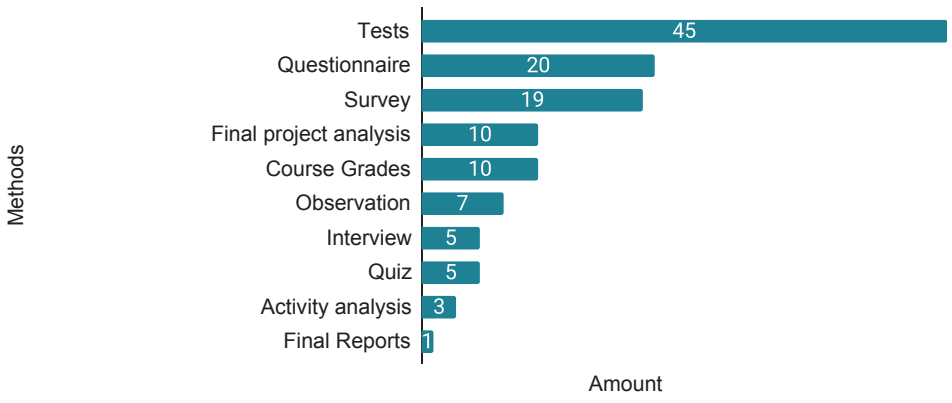


Figura A.12: Data collection methods. (*n=80*)

The nature of the initiative is directly related to the type of data collection method. For example, when an initiative has Project-Based Learning as a methodological basis, the final

project produced by the students will be used to verify the development of computational thinking. In addition to the data collection methods used to assess computational thinking, we found some specific assessment methods. Villalba-Condori et al. (2018) used the Computational Thinking Test (CTt) by Román-González et al. (2017), which evaluates the loops, directions, conditional and simple functions. While (Peteranetz et al., 2019) developed a Computational Thinking Test, an 18-item test to measure computational thinking skills. A member of the research team wrote test items, and the items measured abstraction, evaluation, generalization, decomposition, pattern recognition, and algorithms. The test must be applied twice, first at the beginning of the semester and second at the end. This dual application is to check the students progress during the semester.

To test the effectiveness of incorporating Computational Thinking into pre-curriculum, Van Dyne e Braun (2014) used the pre-test and post-test of the Whimbey Analytical Skills Inventory (WASI). The WASI pre-test consists of 38 problems that test mathematical and logical thinking in various ways, the post-test consists of 37 problems of composition similar to the pre-test. Students do not know when the pre-test and the post-test WASI will happen. The pre-test is on the first day of the course, the post-test on the last day, and the WASI test contents are taught during the semester. The WASI test investigates more profound logical concepts, such as the development of algorithms, recursion, and structured logic.

Kim et al. (2014) used the Group Assessment of Logical Thinking (GALT), a pencil and paper test designed to measure logical reasoning skills. The test consists of 21 questions divided into six sub-elements: conservation, proportional reasoning, control variables, probabilistic reasoning, correlational reasoning, and combinatorial reasoning. Pre and post-GALT are held in the first and last class, respectively. The comparison between the pre-test result with the post-test is made to verify the evolution of the students' logical reasoning.

Faculty of Computer Science for Engineers at the University of Nebraska-Lincoln developed a knowledge test to measure computational thinking skills and Computer knowledge. The test has 13 conceptual and problem-solving questions focused on computational thinking. The paper (Shell et al., 2013) associates the test with the Student Perceptions of Classroom Knowledge Building (SPOCK) instrument that assesses participant planning, goal setting, monitoring, and evaluation of study and learning.

Tests and questionnaires were the most used methods to measure computational thinking. Traditional tests, with questions and answers, were used to evaluate computational thinking and evaluate the effects of the initiative. These tests were used in two moments, at the beginning and at the end of the initiative, which allowed researchers to evaluate students' evolution through the comparison between pre and post-test. Tests and questionnaires were applied to collect students' perceptions regarding the initiatives, providing inputs for researchers to evolve their initiatives. In these surveys, researchers also collected students' motivation about computational thinking and the initiative they took part in. Another form of evaluation was the analysis of final projects, where students were able to understand the applicability of computational thinking and produce a contextualized solution in the real world. When analyzing projects, researchers provide personalized feedback to students and helped them with their main difficulties. The combination of different assessment methods and data collection can be used to provide a more comprehensive assessment of students' learning.

A.3.8 Threats to validity

The systematic mapping was based on the Petersen et al. (2015) guidelines. Even if minimal, the risk of bias in systematic mapping remains and knowing that we had reduced the bias using different strategies. One threat to validity may be the search string. The search string for this systematic mapping was "computational thinking": we decided to use only the to avoid excluding

potential papers that characterized their public and context using other terms (e.g., undergraduate students). Using only the term "computational thinking", the search returned a large number of papers and the paper selection was conducted in two stages: 1st and 2nd filter. Several papers were left out because they were available only with payment – 223 papers.

According to (Petersen et al., 2015), another threat to validity may be the potential researcher bias in the selection of papers (theoretical validity). To mitigate this threat, a second researcher was always consulted when there were doubts about the inclusion or exclusion of a paper, and two other researchers reviewed the entire systematic mapping study. A different threat to validity is losing relevant papers that are not indexed in digital libraries used for the search. However, we selected relevant digital libraries according to the research topic and area. Another threat to validity is interpretation when not having the necessary data or misunderstanding the initiatives and results. Few papers report the initiative and result in detail, and some papers leave the information implied, then the omission of details and information makes it difficult to understand the initiative.

A.4 CONCLUSION

This technical report detailed a systematic mapping study to obtain an overview of computational thinking in higher education. The search from four digital libraries resulted in 2,966 papers published between 1987 and 2019. After applying the 1st and 2nd filters, a total of 94 peer-reviewed papers written in English or Portuguese were selected for our systematic mapping study. Based on the extracted data, we answered each research question to produce an overview of the research scenario in computational thinking in higher education.

After analyzing the set of 2,966 papers returned from the initial search, we noticed that computational thinking has been receiving attention in the scientific community. The number of published papers in the theme increases year after year, revealing a considerable growth in interest in incorporate computational thinking into school and academic curriculum. Even though the area is growing, the vast majority of papers focus on initiatives, activities and theoretical studies to support the development of computational thinking in k-12. However, due to technological advances and the influence of technology on society, we must consider supporting the development of computational thinking with different audiences. Furthermore, even if the number of papers that aims to support the development of computational thinking in higher education is less frequent, the 94 selected papers in this systematic mapping indicate benefits to exercising computational thinking in higher education.

Research on computational thinking has two well-marked phases. The first phase was between 1987 and 1995, illustrating the origin and the theory behind computational thinking. Seymour Papert research marks the first phase, he used the term for the first time in the literature to show how the computer could stimulate thinking and develop knowledge. The second phase started in 2006 when Jeannette Wing published a viewpoint of computational thinking at Communications of the ACM. Since then, the term started to gain popularity, and researchers worldwide began to invest efforts to understand what this "new" way of thinking, how to develop and explore the benefits in different contexts and purposes. According to Wing (2006), in addition to learning to read, write, and arithmetics, we should add computational thinking to the analytical skills of each child. This statement may have been one of the main motivations for researchers to have children as their main research audience focus.

In the 94 selected papers, the initiatives focused on formal educational contexts, such as the classroom and laboratories. No papers were found that applied activities in informal education contexts. The length of the initiatives varies over a semester, and the workload is up to

50 hours. The selected papers provide few details about the initiatives undertaken, revealed that some papers lacked rigor in how they describe the study and the methodology. Some papers did not provide a complete description of the initiative, and only six papers show the age of participants. Lack of information, such as the age, gender, number of participants, and the initiative's length, can prevent the replication and comparison between initiatives. Besides, this insufficient characterization reveals that we must improve the characterization when reporting the initiative mainly because the results obtained have a direct relationship with the participants of the initiative and how it was conducted. Therefore, providing research details as much as possible will support teachers to replicate, adapt and implement their computational thinking initiatives.

According to our systematic mapping study, there is still space for research on the assessment and evaluation of computational thinking. Evaluating computational thinking can be related to understanding what it is, what skills are associated, and how they can be developed. Therefore, more assessments must be designed to identify the relationship between computational thinking skills and domain knowledge. Developing more assessments will establish a more significant relationship between computational thinking, its skills, and how to incorporate them into initiatives. As revealed by the definitions, computational thinking includes a variety of mental tools that reflect the breadth of Computer Science, being considered a fundamental skill for the 21st century (Wing, 2011). Therefore, computational thinking has the potential to help students deepen their understanding of different subject domains and solve problems in everyday life (ISTE e CSTA, 2011).

There is still a conflict of understanding about what computational thinking is, what programming is, and the relationship between them in the literature. The contents of the localized initiatives are closely related to the programming fundamentals, solving computational problems, and principles of computing. Therefore, the most used tools in the initiatives are the computer, the programming languages, and the visual programming languages. As suggested in the review of Hsu et al. (2018), computational thinking can be implemented in broad contexts instead of focusing only on computational solutions. Further, Brackmann et al. (2019) suggested that computational thinking could be introduced without computers, and Rodriguez et al. (2017) found that unplugging has good results.

The results show that effort is being directed but reveal that there is still space to discuss future research. Computational thinking can be associated with any subject domain and age group, so we must explore efficient teaching approaches to work with computational thinking. Therefore, our systematic mapping opens up several other avenues for further research, for example: (i) more formal assessment methods should be developed; (ii) a relationship between computational thinking, its skills, and the content should be established; (iii) each paper should define what the computational thinking skills are and how to evaluate each of them; and (iv) research should be detailed as much as possible.

APÊNDICE B – DATA EXTRACTION TEMPLATE

The data extraction template is presented in B.1.

Tabela B.1: Data extraction template.

Item	Expected Value	Research Question(s)
Id	A unique integer for each paper.	
Title	Paper title	
Authors	Full name of each author.	
Publication Year	Year the paper was published.	RQ07
Publication language	Language the paper was written.	RQ07
Published in	Name of the publication venues where the paper was published.	RQ07
Digital Library	Name of the digital library the paper is available.	RQ07
Author's institution	Institution name of each author.	RQ08
Institution country	Country of each institution.	RQ08
Study Purpose	Study purpose presented in the paper.	RQ04
Study Benefits	Study benefits reported in the text.	RQ05
Study Difficulties	Study difficulties reported in the text.	RQ05
Number of Students	The number of students who participated in the study.	RQ03
Students' age	The average age of students who participated in the study.	RQ03
Students' gender	Gender of students who participated in the study.	RQ03
Undergraduate Course Period or Year	Year and university period of the students who participated in the study.	RQ03
Undergraduate Course Name	Name of the undergraduate course of the students who participated in the course.	RQ03
Initiative Name	Initiative Name	SQ03
Place of initiative	The place where the initiative has happened - Possible answers: classroom, laboratory, room + lab, or online.	RQ03
Duration of the initiative	Duration of the initiative in minutes.	SQ03
Initiative Theme	The theme of the initiative.	RQ03
Initiative Content	List of content taught in the initiative.	RQ03
Initiative Tools	List of tools used in the initiative	RQ03
Computational thinking skills	List of computational thinking skills developed in the Initiative	RQ03

Item	Expected Value	Research Question(s)
Computational thinking definition	Computational thinking definition used in the paper.	RQ01 RQ02
Research Basis	Is the paper based on any theoretical or methodological basis? - Possible answers: Yes, which ones? or not.	RQ02
Evaluation methods	Is computational thinking being evaluated? - Possible answers: Yes, how? or not.	RQ06
Evaluation Tools	Were tools used to evaluate computational thinking? - Possible Answers: Yes, which ones? or not.	RQ06
Skills Evaluation	How were computational thinking skills evaluated? - Possible answers: together or separately	RQ06
Socialization	Have student results been socialized? - Possible answers: Yes, how? or No.	RQ06

APÊNDICE C – LIST OF INCLUDED PAPERS

The papers included in our systematic mapping are compiled in Table C.1.

Tabela C.1: Included papers.

#	Authors	Title	Year
1	Borges et al.	Projetos Maker como Forma de Promover o Desenvolvimento do Raciocínio Formal *(only in Portuguese)*	2016
2	Batista et al.	Uso do Scratch no Ensino de Programação em Ponta Porã: das Séries Inicias ao Ensino Superior *(only in Portuguese)*	2016
3	Lima et al.	Uma Oficina para Ensino de Algoritmos Paralelos por meio de Computação Desplugada *(only in Portuguese)*	2018
4	Walden et al.	An Informatics Perspective on Computational Thinking	2013
5	Qin	Teaching Computational Thinking Through Bioinformatics to Biology Students	2009
6	Miller et al.	Integrating Computational and Creative Thinking to Improve Learning and Performance in CS1	2014
7	Kafura e Tatar	Initial Experience with a Computational Thinking Course for Computer Science Students	2011
8	Shell et al.	Improving Students' Learning and Achievement in CS Classrooms Through Computational Creativity Exercises that Integrate Computational and Creative Thinking	2017
9	Chowdhury et al.	Analysis of Collaborative Learning in Acomputational Thinking Class	2018
10	Babbitt et al.	Punch Cards to Python: A Case Study of a CS0 Core Course	2019
11	Miller et al.	File References, Trees, and Computational Thinking	2010
12	Villalba-Condori et al.	Approaches of Learning and Computational Thinking in Students that Get Into the Computer Sciences Career	2018
13	Kim e Kim	Flipped Learning of Scratch Programming with Code.org	2017
14	Cortina	An Introduction to Computer Science for non-majors Using Principles of Computation	2007
15	Hambrusch et al.	A Multidisciplinary Approach Towards Computational Thinking for Science Majors	2009
16	Kafura et al.	Design and Preliminary Results from a Computational Thinking Course	2015
17	Peteranetz et al.	Building Computational Creativity in an Online Course for non-major	2019
18	Ritz	Programming the Central Dogma: an Integrated Unit on Computer Science and Molecular Biology Concepts	2018
19	Pollock et al.	Infusing Computational Thinking Across Disciplines: Reflections Lessons Learned	2019
20	Gardner et al.	Engaging non-majors in Computer Literacy Courses	2011
21	Senske	Evaluation and Impact of a Required Computational Thinking Course for Architecture Students	2017
22	Van Dyne e Braun	Effectiveness of a Computational Thinking (CS0) Courseon Student Analytical Skills	2014

#	Authors	Title	Year
23	Behnke et al.	Computer Science Principles: Impacting Student Motivation Kearning within and Beyond the Classroom	2016
24	Sherman e Martin	The Assessment of Mobile Computational Thinking	2015
25	López e García-Peñalvo	Relationship of Knowledge to Learn in Programming Methodology and Evaluation of Computational Thinking	2016
26	Kafura et al.	A Computational Thinking Course Accessible to non-stem majors	2018
27	Qualls et al.	CS1 Students' Understanding of Computational Thinking Concepts	2011
28	Yuen e Robbins	A Qualitative Study of Students' Computational Thinking Skills in a Data-driven Computing Class	2014
29	Ginat	The Baffling CS Notions of "As-if" And "Don't-care"	2010
30	Figueiredo et al.	Ne-course For Learning Programming	2016
31	Barba e Chancellor	Tangible Media Approaches to Introductory Computer Science	2015
32	Siva et al.	Using Music to Engage Students in an Introductory Undergraduate Programming Course for non-majors	2018
33	Martin e Sherman	A Dual-major Course Emphasizing Computer Science and Graphic Design	2015
34	Bryant et al.	Using the Context of Algorithmic Art to Change Attitudes in Introductory Programming	2011
35	Flanigan et al.	Exploring Changes in Computer Science Students' Implicit Theories of Intelligence Across the Semester	2015
36	Ruthmann et al.	Teaching Computational Thinking Through Musical Live Coding in Scratch	2010
37	Mason et al.	Computational Thinking as a Liberal Study	2016
38	L'Heureux et al.	It Problem Solving: An Implementation of Computational Thinking in Information Technology	2012
39	Lai et al.	Study on Enhancing a IoT Computational Thinking Skills by Plot Image-based Vr	2019
40	Wu et al.	Analysing Computational Thinking in Collaborative Programming: A Quantitative Ethnography Approach	2019
41	Larkins e Harvey	Introductory Computational Science Using Matlab and Image Processing	2010
42	Kazimoglu et al.	Learning Programming at The Computational Thinking Level Via Digital Game-play	2012
43	Libeskind-Hadas e Bush	A First Course in Computing with Applications to Biology	2013
44	Rubinstein e Chor	Computational Thinking in Life Science Education	2014
45	Goodman e Dekhtyar	Teaching Bioinformatics in Concert	2014
46	Tsai e Tsai	Applying Online Externally-facilitated Regulated Learning and Computational Thinking to Improve Students' Learning	2018
47	Gero e Levin	Computational Thinking and Constructionism: Creating Difference Equations in Spreadsheets	2019
48	Bui et al.	Developing Webquest 2.0 Model For Promoting Computational Thinking Skill	2018
49	Tseng et al.	Students' Self-authoring Mobile App for Integrative Learning of Stem	2018
50	Romero et al.	Computational Thinking Development Through Creative Programming in Higher Education	2017

#	Authors	Title	Year
51	Chang	Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction	2014
52	Buckingham e Hogan	Computational Science for Undergraduate Biologists via Qut.Bio.Excel	2014
53	Kim et al.	Paper-and-pencil Programming Strategy Toward Computational Thinking for non-majors: Design Your Solution	2014
54	Vieira et al.	Student Explanations in the Context of Computational Science and Engineering Education	2019
55	Tsai et al.	Exploring The Effects of Web-mediated Computational Thinking on Developing Students' Computing Skills in a Ubiquitous Learning Environment	2017
56	Liao e Liang	An Empirical Study on Blended Learning to Promote the Development of Computational Thinking Ability of College Students	2017
57	Haddad e Kalaani	Can Computational Thinking Predict Academic Performance?	2015
58	Philip et al.	A Pragmatic Approach to Develop Computational Thinking Skills in Novices in Computing Education	2013
59	Shuiyan He et al.	Teaching Method Based on Computational Thinking a Case Research	2014
60	Shyamala et al.	Teaching Computational Thinking to Entry-level Undergraduate Engineering Students at Amrita University	2017
61	Se et al.	Computational Thinking Leads to Computational Learning: Flipped Class Room Experiments in Linear Algebra	2015
62	Ambrósio et al.	Digital Ink for Cognitive Assessment of Computational Thinking	2014
63	Lin et al.	Exploring the Role of Visual Programming Activities in Computational Thinking	2018
64	Tang et al.	Blended Learning of "Data Structure"Course Based on Computational Thinking	2018
65	Yeh et al.	Teaching Computational Thinking to non-computing Majors Using Spreadsheet Functions	2011
66	Miller et al.	Improving Learning of Computational Thinking Using Creative Thinking Exercises in CS-1 Computer Science Courses	2013
67	Rojas-López e García-Peñalvo	Learning Scenarios for the Subject Methodology of Programming from Evaluating the Computational Thinking of New Students	2018
68	Souza et al.	Self-organizing Maps to Find Computational Thinking Features in a Game Building Workshop	2017
69	Jung et al.	CS for All: Introducing Computational Thinking with Hands-on Experience in College	2017
70	Gossen et al.	Computational Thinking: Learning by Doing with the Cinco Adventure Game Tool	2018
71	Shell et al.	Improving Learning of Computational Thinking Using Computational Creativity Exercises in a College CS1 Computer Science Course for Engineers	2014
72	Yang et al.	A Novel Interdisciplinary Course in Gerontechnology for Disseminating Computational Thinking	2011
73	Krugel e Hubwieser	Computational Thinking as Springboard for Learning Object- Oriented Programming in an Interactive Mooc	2017
74	Scott e Barlowe	How Software Works: Computational Thinking and Ethics Before CS1	2016

#	Authors	Title	Year
75	Rahman e Paudel	Visual Programming and Interactive Learning Based Dynamic Instructional Approaches to Teach an Introductory Programming Course	2018
76	Davies	The Effects of Emphasizing Computational Thinking in an Introductory Programming Course	2008
77	Rezende e Bispo	Comparison Between the Use of Pseudocode and Visual Programming in Programming Teaching an Evaluation from Scratch Tool	2018
78	Taylor	Can Utilizing Social Media and Visual Programming Increase Retention of Minorities in Programming Classes?	2013
79	Almeida e Pessoa	Scratch Software in Higher Education Pedagogical Experience in Educational Science	2017
80	Lester	An Innovative Approach to Teaching an Undergraduate Software Engineering Course	2008
81	Knobelsdorf et al.	A Computing Education Approach for Geography Students in Context of Gis	2017
82	Shell et al.	Associations Of Students' Creativity, Motivation, and Self-regulation with Learning and Achievement in College Computer Science Courses	2013
83	Peteranetz et al.	Helping Engineering Students Learn in Introductory Computer Science (CS1) Using Computational Creativity Exercises (Cces)	2018
84	Chilana et al.	Perceptions of non-CS majors in Intro Programming: the Rise of the Conversational Programmer	2015
85	Kurniawan et al.	Effectiveness of Physical Robot Versus Robot Simulator in Teaching Introductory Programming	2018
86	Almeida et al.	Learning to Think Like a Trainer: Bringing Scratch for Educational Sciences Professional's Formation	2018
87	Li et al.	Virtual Experiments for Introduction of Computing: Using Virtual Reality Technology	2015
88	Hurson e Sedigh	Transforming the Instruction of Introductory Computing to Engineering Students	2010
89	Boechler et al.	Computational Thinking, Code Complexity and Prior Experience in a Videogame-building Assignment	2014
90	Yinnan e Chaosheng	Training for Computational Thinking Capability on Programming Language Teaching	2012
91	Bremgartner et al.	Conceptual Framework of Educational Resources Adaptation for Improve Collaborative Learning in Virtual Learning Environments	2017
92	Turbak et al.	Blocks Languages for Creating Tangible Artifacts	2012
93	Cápay	Algorithmic Thinking Observation: How Students of Applied Informatics Break the Mystery of Black Box Applications	2014
94	Ng	Introducing The Practices for Adopting the Constructivist Teaching in Game Engineering	2017

APÊNDICE D – ABORDAGEM PENSE COMPUTACIONALMENTE

A abordagem Pense Computacionalmente foi criada e compilada em um material que será apresentado na sequência.

Pense Computacionalmente



Uma abordagem para apoiar o exercício do pensamento computacional

Carolina Moreira Oliveira

Roberto Pereira

Este livro reúne todas as informações necessárias para inspirar ou ser utilizada por professores e pesquisadores para promover o exercício do pensamento computacional no ensino superior em Computação. A abordagem pode ser adaptada para ser utilizada em turmas do ensino médio, ensino médio técnico ou com diferentes cursos de graduação. Dúvidas e sugestões podem ser encaminhadas para carolmoliveiraa@gmail.com.

Para acessar todas as informações da abordagem atualizada acesse https://github.com/carollm/pense-computacionalmente

Versão 1.0

Autores: Carolina Moreira Oliveira e Roberto Pereira

Esta obra está licenciada com uma Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional (CC BY-NC-SA 4.0), quais dúvidas quanto as permissões consulte o link.



Agradecimentos a Ludmilla Galvão pela ilustração da logo da abordagem.

Sumário

1. Introdução 04

2. Orientações 06

3. Práticas 07

4. Método 10

5. Método de Condução das Práticas 12

6. Representação da Solução 14

7. Evidências 16

8. Catálogo de Práticas 21

8.1. Desafio Mais Velha 22

8.2. Teste dos Labs 24

8.3. Desafio Lista de Presença 27

8.4. Desafio Criptografia 29

8.5. Desafio Busca 32

8.6. Desafio Ordenação 34

8.7. Desafio Celebridade 36

9. Referências 38

1. Introdução

Devido à influência da tecnologia na sociedade, a Computação está se tornando uma das principais disciplinas da educação do Século XXI. Desse modo, os conceitos relacionados a Computação e o pensamento computacional começaram a receber mais atenção tanto no âmbito educacional quanto empresarial.

O pensamento computacional é definido como um processo de pensamento envolvido na formulação de problemas e de suas soluções, preocupando-se em como as soluções são representadas para que elas sejam efetivamente executadas por um agente de processamento de informação[1, 2]. Com raízes derivadas do pensamento natural, o pensamento computacional envolve ideias como abstração, decomposição e representação dos dados.

A popularização do Pensamento Computacional veio a partir da publicação da Wing em 2006[3], onde a autora definiu o termo como um conjunto de habilidades baseadas nos conceitos fundamentais da Ciência da Computação que podem ser utilizadas para resolver problemas, projetar sistemas e entender o comportamento humano. Sendo uma habilidade humana, o pensamento computacional promove uma aprendizagem mais ampla que auxilia no desenvolvimento dos saberes necessários para apoiar o ensino e aprendizado dos estudantes em diferentes áreas.

Desde 2006, a quantidade de pesquisas sobre o tema tem aumentado ano após ano, revelando um crescimento considerável no interesse à incorporação do pensamento computacional nos currículos. O Ministério da Educação (MEC)[4], a Sociedade Brasileira de Computação (SBC)[5], a Association for Computing Machinery (ACM) e a IEEE Computer Society[6] reconhecem o pensamento computacional é reconhecido como uma das habilidades fundamentais desejadas para todos os graduandos, principalmente em Computação.

Diretrizes curriculares das graduações em Computação reconhecem o pensamento computacional como uma habilidade que deve ser exercitada por todos os estudantes em formação na área. Entretanto, as diretrizes não deixam claro quando e como essas habilidades devem ser exercitadas.

Com objetivo de apoiar pesquisadores e professores na concepção e condução de práticas[a] que promovam o exercício do pensamento computacional desenvolvemos a abordagem Pense Computacionalmente.

A abordagem tem como premissa que o pensamento computacional deve ser exercitado explicitamente desde o primeiro semestre dos cursos de graduação, de maneira contextualizada, significativa, colaborativa e com uma abordagem desplugada[b] e baseada em desafios.

A abordagem é composta por um conjunto de **orientações** de como preparar e conduzir o exercício do pensamento computacional, um **método** para guiar estudantes durante a resolução das práticas, e um conjunto de sete **práticas** para serem utilizadas ou inspirar a criação de novas.



Composição da abordagem

[a] Práticas podem ser compreendidas como um conjunto de ações que facilitam e promovem a aprendizagem dos estudantes, levando em conta seu contexto social e cultural, os objetivos, entre outros critérios.

[b] Desplugada aqui podem ser compreendidas como sem a utilização de uma linguagem de programação.

2. Orientações

O pensamento computacional deve ser exercitado explicitamente desde o 1º semestre dos cursos de graduação de forma significativa, colaborativa, contextualizada e desplugada. As atividades devem ser baseadas em problemas reais e ancoradas no contexto acadêmico onde serão realizadas, levando em consideração a comunidade e estudantes, e assim possibilitando a troca de conhecimento e incentivando os estudantes a aprenderem uns com os outros. Ao utilizar problemas reais e contextualizados consegue-se somar ao propósito e motivação para o aprendizado. Além disso, é importante reservar um espaço para os estudantes interagirem entre si e socializarem as suas soluções — possibilitando que os estudantes construam o conhecimento por meio da colaboração entre eles.

As atividades devem fornecer experiências para mostrar que conceitos abstratos fazem sentido e são aplicáveis ao mundo cotidiano, proporcionando aos estudantes a construírem significado com base em suas próprias experiências.

Assim, os cinco pilares da abordagem dão luzes de como as práticas devem ser realizadas para apoiar o exercício do pensamento computacional.



Pilares da Abordagem

3. Práticas

A abordagem Pense Computacionalmente conta com um conjunto de sete práticas, que pode ser utilizado ou adaptado para o contexto onde será realizado. A estrutura base de uma prática é composta por: (i) um desafio e a sua descrição; (ii) restrições a serem respeitadas durante a criação da solução; (iii) explicação sobre os objetivos da prática; e (iv) as habilidades a serem exercitadas. A descrição do desafio é responsável por detalhar o problema e contextualizá-lo, além de deixar explícito quais são os objetivos, os benefícios e as habilidades que ele exercita.

Selecionamos um conjunto de 9 habilidades do pensamento computacional que serão exercitadas nas práticas, são elas: abstração, decomposição, reconhecimento de padrões, coleta e análise dos dados, algoritmos, generalização, representação da solução, simulação e teste e otimização – as definições podem ser encontradas na Tabela 1.

O conjunto de sete desafios pode ser consultado na página 19.

3.1. Habilidades

As práticas da abordagem Pense Computacionalmente auxilia no exercício do pensamento computacional e *soft skills*. O conjunto de habilidades foi definido a partir de uma revisão sistemática, da literatura, e das diretrizes curriculares dos cursos de graduação em Computação.

Habilidades Pensamento Computacional

A essência do pensamento computacional está relacionada com as habilidades inerentes à resolução de problemas, que incluem quatro pilares: i. abstração, ii. decomposição, iii. reconhecimento de padrões e iv. algoritmos. Os pilares do pensamento computacional podem ser vistos como habilidades de alto nível que se relacionam com um grupo de habilidades derivadas.

Para as práticas selecionamos um conjunto de 9 habilidades do pensamento computacional são elas:

- Abstração
- Decomposição
- Reconhecimento de padrões
- Coleta e análise dos dados
- Algoritmos
- Generalização
- Representação da solução
- Simulação e Teste
- Otimização

As habilidades estão detalhadas na Tabela 1.

Tabela 1. Descrição das habilidades

Habilidades	Descrição
Abstração	Habilidade de interpretar e entender o desafio, identificando quais aspectos são relevantes e quais podem ser ignorados.
Decomposição	Habilidade de identificar quais partes do desafio podem ser divididas em partes menores, mais fáceis de resolver e de gerenciar.
Reconhecimento de Padrões	Habilidade de identificar os padrões do desafio, dos dados e da solução, reconhecer as similaridades entre os desafios e soluções para que seja possível reaproveitar ou aperfeiçoar as soluções existentes.
Coleta e análise dos dados	Habilidade de identificar como os dados estão ordenados e organizados, e de fazer sentido dos dados coletados.
Algoritmo	Habilidade de definir uma sequência ordenada de instruções finitas para a solução de um problema.
Generalização	Habilidade de propor uma solução que possa ser utilizada para diferentes quantidades de dados.
Representação da Solução	Habilidade de organizar as informações e instruções do algoritmo considerando com quem ou o que irá interpretar a solução.
Simulação e Teste	Habilidade de simular e testar a solução para a entrada de dados do desafio com o objetivo de identificar falhas e espaço para melhorias.
Otimização	Habilidade de corrigir, refinar e otimizar a solução existente.

Soft Skills

Soft skills é um termo em Inglês utilizado para referir-se às habilidades interpessoais. De um modo geral, essas habilidades são transpostas para todos os níveis ao longo de uma vida, principalmente na caminhada educacional, e possuem relação direta com o contexto social e educacional.

As *soft skills* exercitadas pela abordagem Pense Computacionalmente possuem relação direta com as habilidades que estudantes de Computação devem exercitar, e que são apontadas também pelas diretrizes curriculares dos cursos de graduação em Computação. Assim, chegamos ao conjunto de 9 soft skills, são elas:

- Comunicação
- Criatividade e Inovação
- Gestão de Tempo
- Negociação e Flexibilidade
- Pensamento Analítico
- Pensamento Crítico
- Resolução de Problemas
- Responsabilidade e Ética Profissional
- Trabalho em Equipe

As soft skills estão detalhadas na Tabela 2.

Tabela 2. Descrição das soft skills

Soft Skill	Descrição
Comunicação	Habilidade de ouvir e se fazer entender de maneira clara e eficaz, levando em consideração o contexto e o público-alvo.
Criatividade e Inovação	Habilidade de combinar ideias novas ou já existentes para gerar um resultado ou solução nova, considerando o seu conhecimento prévio, repertório e experiências.
Gestão de Tempo	Habilidade de definir prioridades e metas, gerenciando o tempo necessário para realizar uma determina tarefa e propor uma solução.
Negociação e Flexibilidade	Habilidade de dialogar com pessoas e buscar uma solução que satisfaça os envolvidos, defendendo os seus interesses, mas também reconhecendo quando ceder.
Pensamento Analítico	Habilidade de analisar, resumir, quantificar e cruzar dados para chegar a uma conclusão.
Pensamento Crítico	Habilidade de refletir, criticar, julgar e cruzar informações, teorias e no conhecimento prévio para chegar em uma opinião própria.
Resolução de Problemas	Habilidade de analisar o cenário do problema, determinar as causas do problema e buscar soluções para ele, considerando os recursos existentes.
Responsabilidade e Ética Profissional	Habilidade de conduzir as atividades e relações profissionais norteando suas ações baseadas em valores individuais e coletivos, desenvolvendo soluções socialmente responsáveis que considere questões sociais, éticas, culturais.
Trabalho em Equipe	Habilidade de interagir e trabalhar com outras pessoas na mesma solução, sabendo ouvir, negociar e respeitar todas as pessoas que compõem a equipe.

4. Método

Mesmo que as práticas sejam diferentes entre si, elas possuem uma estrutura básica que deve ser respeitada durante o processo de criar a solução para o problema. A estrutura básica foi dividida em 6 etapas, que pode envolver tanto trabalho individual quanto em grupos. Para representar a estrutura básica para guiar o processo de resolução de uma prática criamos um método.

O método foi representado de maneira cíclica para representar a relação de dependência entre as etapas, e ilustrar o processo de amadurecimento da solução — existe uma relação de amadurecimento da solução à medida que as etapas vão sendo concluídas. As etapas do método foram pensadas para reservar momentos específicos para elaborar, testar, compartilhar, refinar e otimizar a solução.

O processo pode ser repetido quantas vezes desejar ou forem necessárias. No fim de cada ciclo, espera-se que as soluções estejam superiores à solução inicial, ou do ciclo anterior (caso executado mais de um ciclo).



Método para guiar a resolução das práticas

As etapas do método foram pensadas para guiar o processo de criação da solução, por isso ele foi dividido em seis etapas, são elas:

1. Caracterizar e representar o problema, identificando seus requisitos e restrições;
2. Elaborar e representar uma solução inicial;
3. Testar a solução inicial, a fim de verificar se ela resolve o problema caracterizado na etapa 1;
4. Socializar as soluções, para receber comentários sobre a solução proposta, considerando a corretude e facilidade de execução da solução;
5. Refinar a solução proposta; e
6. Testar a solução e avaliar os resultados que ela produz.

No método foi previsto tanto momentos para para trabalhar individualmente quanto em grupos. As etapas em grupo são destinadas para que os estudantes compartilhem entre si suas soluções, recebam e forneçam comentários sobre as soluções, apontando aspectos que podem ser corrigidos ou melhorados. Ainda, as etapas realizadas em grupo podem resultar em uma nova solução — considerando o melhor das soluções de cada integrante do grupo.

A etapa de socialização representa um processo de amadurecimento de maneira colaborativa, uma vez que os estudantes avaliam, argumentam e apontam correções entre eles.

A utilização do método e a condução do processo de resolução da atividade possui relação direta com as habilidades que serão exercitadas durante o processo. Consulte a página 10 para entender melhor sobre as habilidades exercitadas.

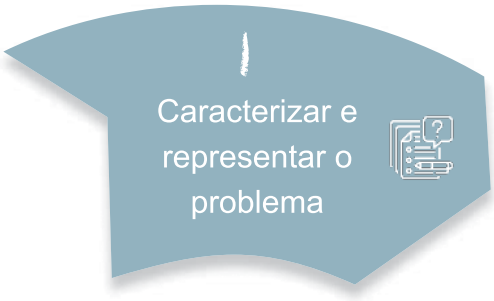


Dica: a etapa de socialização pode ser ajustada de acordo com a quantidade de estudantes presentes na sala e o tempo disponível.

5. Método de Condução das Práticas

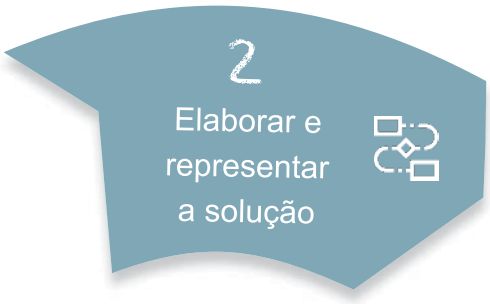
O foco de exercitar essas habilidades é preparar os estudantes a atuarem na criação de soluções para problemas, de maneira criativa, colaborativamente e socialmente consciente. O exercício do pensamento computacional e das *soft skills* possuem relação direta com a condução das práticas, logo, ao retirar, adicionar, ou até mesmo focar em uma das etapas do método impactará no conjunto final de habilidades exercitadas pelos estudantes. Portanto, a condução de uma prática precisa ser guiada com base nas habilidades que deseja-se exercitar.

Relação de habilidades exercitadas durante cada etapa da condução da prática prevista pelo método:



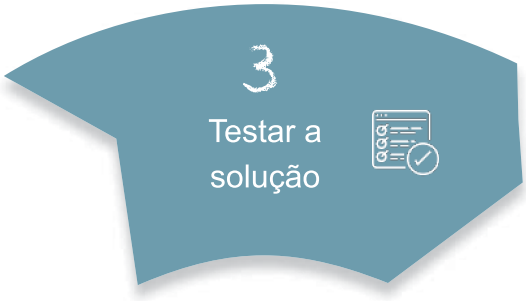
Abstração
Decomposição
Reconhecimento de Padrões

Pensamento Crítico
Resolução de Problemas
Responsabilidade e Ética Profissional



Algoritmos
Generalização
Coleta e Análise dos Dados
Reconhecimento de Padrões
Representação da Solução

Criatividade e Inovação
Resolução de Problemas
Pensamento Analítico



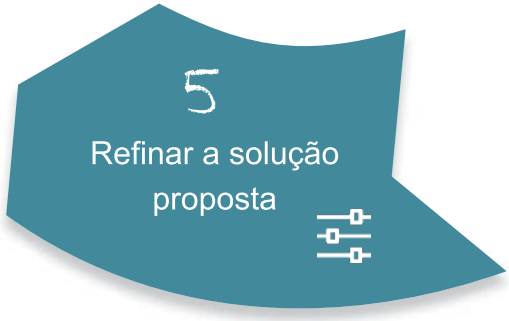
Decomposição
Coleta e Análise dos Dados
Simulação e teste

Resolução de Problemas
Pensamento Crítico
Pensamento Analítico



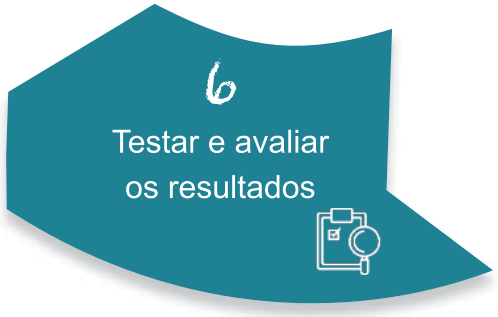
Algoritmo
Coleta e Análise de dados
Simulação e Teste

Criatividade e Inovação
Trabalho em Equipe
Negociação e Flexibilidade
Responsabilidade e Ética Profissional



Algoritmos
Generalização
Otimização

Criatividade e Inovação
Pensamento Analítico
Resolução de Problemas



Coleta e Análise dos Dados
Simulação / teste
Otimização

Criatividade e Inovação
Resolução de Problemas

A temática da prática também ajuda a desenvolver habilidades como pensamento crítico, responsabilidade ética e profissional. Além disso, durante todo o processo de resolução dos desafios as habilidades de gestão de tempo, resiliência e flexibilidade são exercitadas.

6. Representação da Solução

Um dos pilares da abordagem Pense Computacionalmente é a Computação Desplugada, uma abordagem baseada no construtivismo para ensinar os conceitos gerais da Ciência da Computação de maneira desconectada. As atividades desplugadas buscam comunicar os conteúdos sem a distração dos computadores, essas atividades possuem o objetivo de utilizar criativamente as ideias computacionais para solucionar problemas.

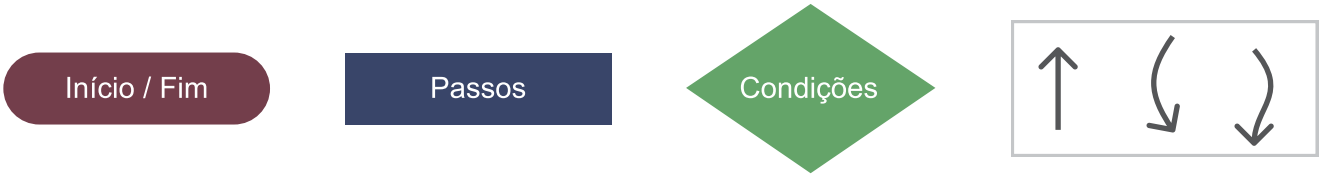
Os estudantes quando participam de uma atividade desplugada não precisam se preocupar com a sintaxe de uma linguagem de programação ou com uma interface de desenvolvimento, e todos os esforços são destinados a resolução do problema proposto. A Computação Desplugada proporciona um ambiente de exploração que permite o uso da criatividade para criar soluções e ajuda a compreender as teorias computacionais.

Assim, a abordagem Pense Computacionalmente recomenda que as soluções das práticas sejam representadas de duas formas: (i) sequência de instruções em linguagem natural; ou (ii) fluxograma.

Linguagem natural é a forma mais comum de comunicação, à qual utilizamos no dia-a-dia e estamos mais familiarizados, e pode ser compreendida como a forma utilizada pelas pessoas para falar, escrever e se comunicar, de acordo com a sua cultura e nacionalidade. Portanto, representar a solução em instruções em linguagem natural é criar uma sequência de passos que deve ser realizadas para chegar no resultado esperado.

Fluxograma é uma representação esquemática gráfica de um processo, que ilustra de forma descomplicada a sequência de passos do desenvolvimento de um processo. Um fluxograma geralmente é composto por figuras geométricas padronizadas e setas que indicam o fluxo e unem as figuras. Através da gráfica é possível compreender de forma rápida e fácil a sequência de passos a serem percorridos durante o processo.

Elementos básicos que podem ser utilizados no fluxograma para representar as soluções:



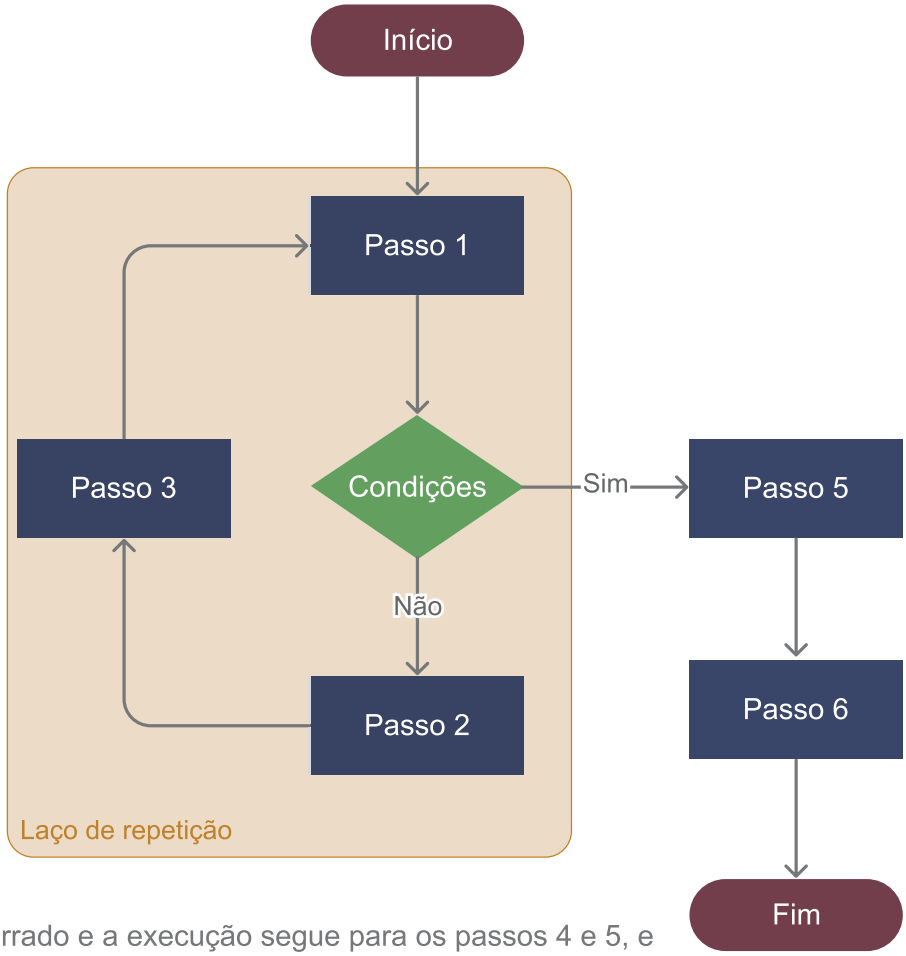
Toda a solução em um fluxograma precisa ter apenas um **início** e pelo menos um **fim**. Os **passos** representam as instruções a serem executadas, o passo-a-passo que resolverá o problema. As **setas** representam a sequência de execução e indicam a direção que devemos seguir para executar os próximos passos. As **condições** controlam a sequência de execução, direcionando a execução do resultado do teste de condição seja verdadeiro/sim ou falso/não. Soluções muitas vezes possuem sequência de passos que se repetem enquanto ou até que determinada condição ocorre, conhecemos isso como **laços**.

No exemplo ao lado, há um estado de início e um de fim. Entre eles, há uma sequência de passos e um condicional que forma um laço de repetição.

A condição é testada logo após a execução do passo 1. Se a resposta for *SIM*, então a execução segue para os passos 4 e 5, ignorando os passos 2 e 3.

Se a resposta for *NÃO*, então a execução segue para os passos 2 e 3, retornando ao passo 1, e então a condição é testada novamente, formando um laço.

O laço de repetição das instruções 2, 3 e 1 será executado até que o resultado do teste da condição seja SIM.



Quando o resultado for SIM, o laço é encerrado e a execução segue para os passos 4 e 5, e então para o Fim. Perceba que se a condição nunca mudar, nunca passaremos pelas instruções 4 e 5, e nunca chegaremos ao Fim: nossa solução ficará em loop eterno.

Por isso, você deve garantir que nos passos 2, 3 ou 1, algo seja feito para que o resultado da condição mude e o laço seja encerrado.

7. Evidências

Considerando a coleta de evidências, a abordagem Pense Computacionalmente conta com um guia para orientar a coleta de evidências do exercício do pensamento computacional. O guia é composto pelo conjunto de habilidades que será observado, indicações de evidências a serem coletadas, e uma escala simplificada para mensurar.

O guia descreve o que é para ser observado de cada uma das habilidades, e descreve pelo menos um ponto por habilidade a ser coletado nas soluções. A escala criada ajuda a mensurar a evidência coletada, a escala varia entre 0 e 3:

0: Não existe evidências do exercício desta habilidade

1: Existe pouca evidência do exercício desta habilidade

2: Existe evidência do exercício desta habilidade

3: Existe muitas evidência do exercício desta habilidade

Para sumarizar e representar o guia estamos utilizando uma tabela como apoio, porém, poderia ser utilizado qualquer outra ferramenta para isso. A tabela está disponível no site da abordagem.

A coleta de evidências é observada dos seguintes pontos:

Abstração

Temos evidências que houve o exercício da abstração quando o contexto do desafio é apresentado e o resultado esperado é atingido. Atingir o resultado esperado garante que os diferentes aspectos do desafio foram identificados e considerados durante a criação da solução. Se a resposta estiver errada ou diferente do tipo esperado, temos evidências que houve problemas no processo de abstração.

Observar se:
- A solução apresenta a contextualização do desafio.
- Foi identificado os requisitos essenciais para a solução.
- Foi atingido o resultado esperado pelo desafio.

Decomposição

Temos evidências que houve o exercício da decomposição quando os diferentes subproblemas de um desafio são identificados e tratados. Dividir o desafio em partes menores favorece a simplificação do desafio em partes mais fáceis de serem gerenciáveis. Assim, utiliza-se dos sub-desafios de um desafio para compor a solução final.

Observar se:
- O desafio foi dividido em sub-desafios.
- Utiliza-se da resolução dos sub-desafios para compor a solução final.

Reconhecimento de Padrões

Temos evidências que houve o exercício do reconhecimento de padrões quando são identificadas as similaridades (padrões) em um desafio ou solução. Logo, reconhecer soluções previamente conhecidas podem ser utilizadas novamente em problemas similares.

Observar se:
- Foi identificado os padrões do desafio.
- Houve reaproveitamento de soluções.
- Houve reaproveitamento de instruções por meio de repetição e condição.

Coleta e análise dos dados

Temos evidências que houve o exercício das habilidades de coleta e análise dos dados quando os dados são utilizados para auxiliar na resolução do desafio. Passando desde a fase de identificação, interpretação dos dados, quanto de como os dados atendem os requisitos do desafio e como eles serão utilizados para chegar na solução.

Observar se:
- As entradas de dados foram identificadas e entendidas.
- As entradas de dados foram manipuladas respeitando as restrições e possibilidades do desafio.

Algoritmo

Temos evidências que houve o exercício da habilidade de algoritmo quando é criado uma sequência de passos para resolver o desafio.

Observar se:
- A solução fornece uma sequência de instruções para resolver o desafio.
- As instruções estão detalhadas.
- A solução está completa.
- Reconhece a diferença entre as estruturas de condição e repetição,
- Controla as repetições e condições,
- Existe ponto de parada/fim.

Generalização

Temos evidências que houve o exercício da habilidade de generalização quando a solução para o desafio considera diferentes entradas de dados, não sendo restrito ao desafio em questão ou uma certa quantidade de dados.

Observar se:
- A solução funciona para um tamanho variável da entrada de dados (n, n+1).

Representação da Solução

Temos evidências que houve o exercício da representação da solução quando a solução do desafio está expressada em uma sequência de passos finitos, sem ambiguidade e de forma que não seja necessário realizar inferências sobre a instrução. Ao final, a sequência de passos deve fornecer o resultado ou resolver o desafio proposto.

Observar se:
- As instruções estão organizadas e ordenadas de uma forma coerente,
- A solução está representada considerando quem irá interpretar a solução,
- As instruções estão escritas de forma clara e objetivas (i.e., não ambíguas).

Simulação e Teste

Temos evidências que houve o exercício da simulação/teste quando é realizada uma reprodução da solução com diferentes entradas de dados para identificar comportamentos diferentes da solução. É uma habilidade que se relaciona com a otimização, uma vez que para otimizar uma solução é interessante conhecer o comportamento dela.

Observar se:

- A solução é testável, i.e., oferece as condições necessárias para ser testada (e.g., instruções, entradas, saídas esperadas).
- Há indícios de que a solução foi simulada/testada (e.g., a condição de parada é atendida),
- Há indícios de tratamento de exceção ou de verificações de dados e condições,
- A solução pode ser executada simultaneamente ou em cooperação com outras partes da solução.

Otimização

Temos evidências que houve o exercício da otimização quando uma solução considera as possibilidades de reduzir a quantidade de recursos necessários para atingir o resultado do desafio. Também, quando existe o processo de correção da solução existente ou refinamento de uma solução já conhecida.

Observar se:

- Apresentou informações sobre correção e refinamento da solução existente,
- Foi considerado na solução a redução de recursos necessário,
- Foi reduzido o número de passos necessários para atingir o resultado.
- Foi reduzido o número de passos utilizados para representar a solução.

O artefato foi planejado para avaliar diferentes aspectos que vão além de atingir o resultado final esperado, pois entendemos que o processo de criar uma solução para uma atividade auxilia a exercitar, reforçar, e desenvolver conhecimentos.

Para sumarizar e representar o artefato, utilizamos o Google Sheets[c] como apoio, porém, qualquer outra ferramenta poderia ser utilizada. A escolha pelo Google Sheets se baseou nos recursos existentes, pela familiaridade que as pessoas possuem com os serviços Google, pela facilidade de uso, e por ser uma ferramenta web colaborativa e não necessitar de instalação.

Qualquer outra ferramenta pode ser utilizadas para representar o artefato.

[c] O Google Sheets é um programa de planilhas incluído como parte do pacote gratuito e online de Editores de Documentos Google oferecido pelo Google.

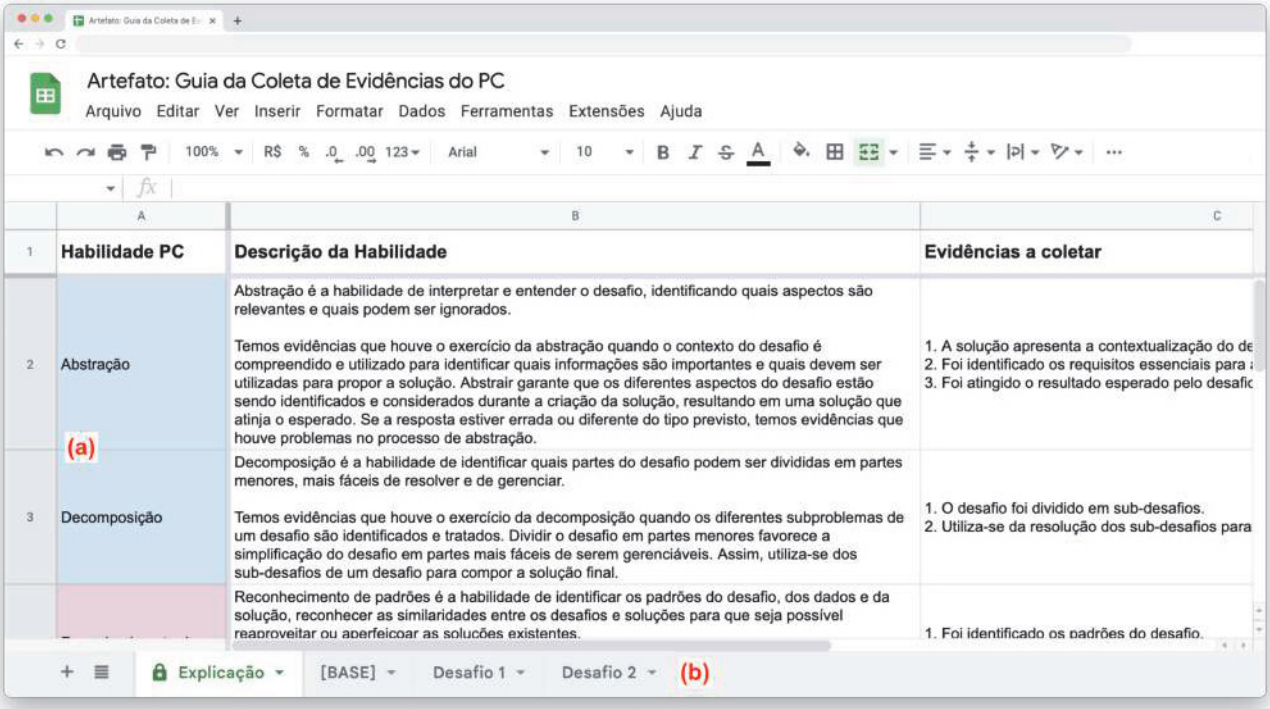


Figura 1. Planilha da explicação do artefato

O artefato contém uma planilha para apresentar as explicações e uma planilha base para registrar as evidências a serem coletadas. A Figura 1 ilustra como a explicação do artefato foi organizada na planilha. A primeira página da planilha apresenta o conjunto de habilidades selecionadas e as suas definições, a lista de evidências a serem coletadas, e a escala de avaliação (ilustrada na Figura 1.a). A explicação está disponível na primeira planilha do artefato, ao lado das planilhas correspondentes a cada desafio, para estar próxima da utilização, acessível e ser consultada sempre que necessário (Figura 1.b).

O artefato conta com uma planilha base, com todas as evidências que devem ser coletadas para o conjunto de habilidades. Na planilha base, as colunas representam as evidências que devem ser coletadas e cada linha representa um estudante. Cada célula (evidência x estudante) contém uma caixa de seleção com as opções da escala. O recurso de caixa de seleção foi utilizado para disponibilizar a escala e ser de fácil seleção. A Figura 2 ilustra a planilha base e como as como as evidências foram organizadas na planilha.

[c] O Google Sheets é um programa de planilhas incluído como parte do pacote gratuito e online de Editores de Documentos Google oferecido pelo Google.

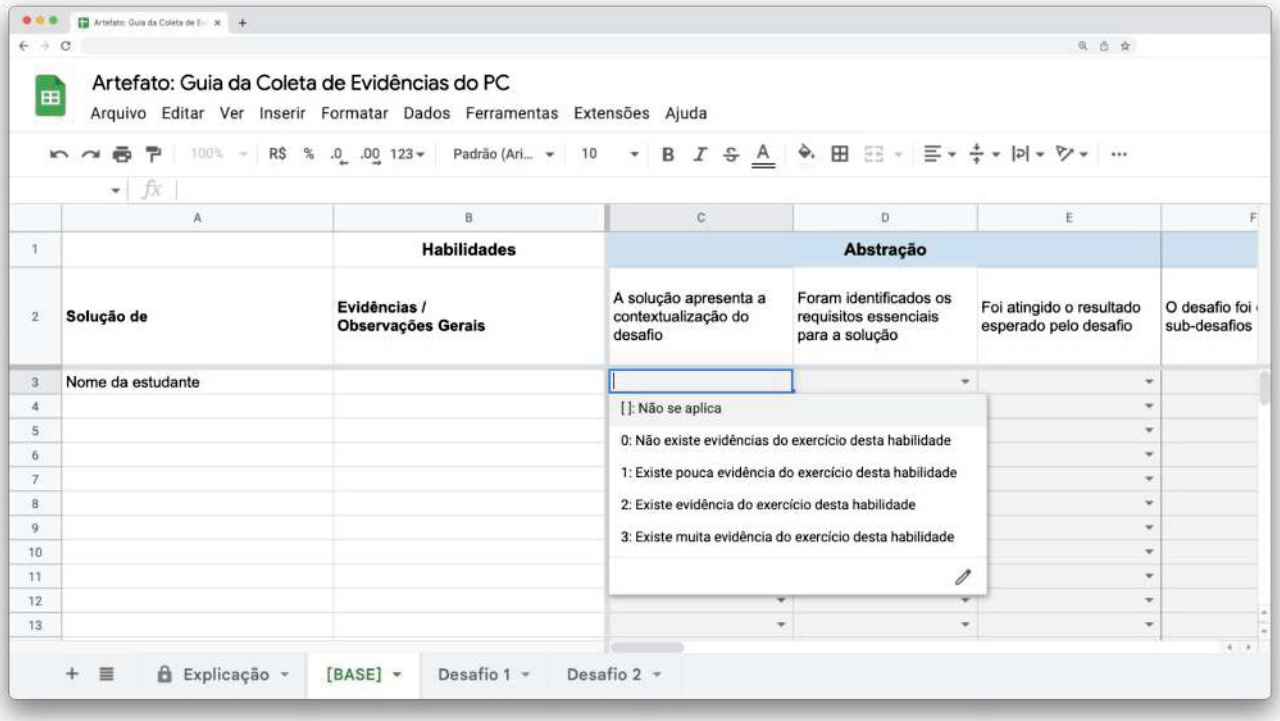


Figura 2. Planilha base para os desafios

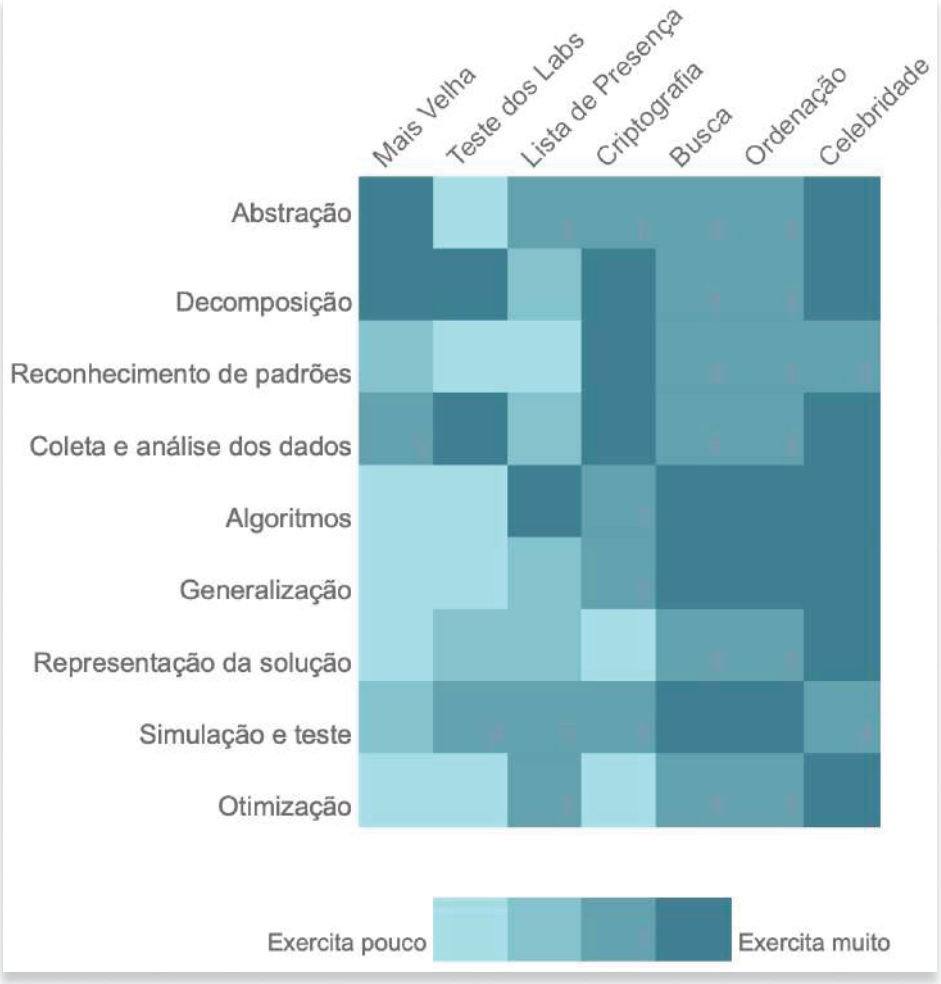
[c] O Google Sheets é um programa de planilhas incluído como parte do pacote gratuito e online de Editores de Documentos Google oferecido pelo Google.

8. Catálogo de Práticas

A abordagem Pense Computacionalmente conta com um catálogo de sete práticas, que serão apresentadas nas próximas páginas. O principal objetivo desse catálogo é proporcionar material base para conduzir práticas para o desenvolvimento do pensamento computacional, seguindo os princípios apresentados pela abordagem. Além disso, durante todo o catálogo existem dicas de como adaptar as práticas, de forma a aproximá-las do contexto em que será aplicado. Assim, o catálogo de práticas é um norte para professores e pesquisadores utilizarem ou adaptarem os desafios existentes. Cada prática é composta por um desafio e sua descrição, as restrições a serem respeitadas pela solução, explicação sobre os objetivos da prática e as habilidades a serem exercitadas. A descrição do desafio é responsável por detalhar o problema e contextualizá-lo.

Cada desafio exercita um conjunto de habilidades do pensamento computacional, essas habilidades possuem relação com o objetivo do desafio, a temática do desafio e os passos necessários para propor uma solução para o desafio.

A relação de habilidades do pensamento computacional para cada desafio pode ser vista na figura ao lado, e as cores utilizadas representam a intensidade que o desafio exercita de cada uma das habilidades.





Desafio Mais Velha

Habilidades: Abstração, Decomposição, Coleta e Análise dos Dados

O objetivo do Desafio Primeira Filha é chegar em uma resposta para o problema apresentado. Este desafio busca desenvolver a capacidade de interpretação do problema e a estruturação da solução, refletindo nos passos necessários para chegar no resultado. O desafio possui apenas uma resposta correta, e o entender as dicas é essencial.

Contexto do Desafio

Duas amigas, Cientistas da Computação, se encontraram na rua depois de longa data. Uma das amigas estava toda agitada que suas três filhas estavam de aniversário naquele dia, e começou:

— Hoje, minhas três filhas comemoram seus aniversários. Duvido que você adivinhe a idade de cada um delas!

— Pouco provável, ao menos que você me dê uma dica…

— O produto das suas idades é 36.

— Essa dica não basta. Preciso de outra.

— A soma das suas idades é igual ao número de janelas daquele prédio — respondeu enquanto apontava para um edifício próximo.

— Ainda não consigo afirmar com certeza e você sabe disso. Qual é a última dica?

— A minha filha mais velha tem olhos azuis.

— Agora, sim! Com certeza, digo que as idades dos suas filhas são…

O desafio é assumir o papel da amiga e descobrir a idade das três filhas. Após resolver este desafio é necessário explicar como chegou na solução: use um conjunto de passos, infográfico ou fluxograma mostrando como chegou na solução.

Resposta do Desafio

A primeira dica diz: o produto das idades é 36. Com essa primeira dica, a amiga considerou todas as hipóteses de três números inteiros positivos cuja multiplicação era 36. Ela pediu uma segunda dica quando percebeu que existem 8 possibilidades, são elas:

	Filha 1	Filha 2	Filha 3
1	1	1	36
2	1	2	18
3	1	3	12
4	1	4	9
5	1	6	6
6	2	2	9
7	2	3	6
8	3	3	4

Filha 1	Filha 2	Filha 3		Soma	
1	1	36	=	38	
1	2	18	=	21	
1	3	12	=	16	
1	4	9	=	14	
1	6	6	=	13	←
2	2	9	=	13	←
2	3	6	=	11	
3	3	4	=	10	

A segunda dica diz: a soma das suas idades é igual ao número de janelas daquele prédio — respondeu enquanto apontava para um edifício próximo. A amiga sabia quantas janelas tinha no prédio, e que o número era igual à soma das idades. Logo, ela conseguiria responder ao desafio, exceto se a quantidade de janelas fosse 13, pois existem duas combinações possíveis. Portanto, se o prédio tivesse 38, 21, 16, 14, 11 ou 10 janelas, ela conseguiria adivinhar as idades com a segunda dica. Mesmo que não seja informada a quantidade de janelas é possível inferir que o prédio tinha 13, visto que ela pediu mais uma dica.

A terceira dica diz: a minha filha mais velha tem olhos azuis. Essa dica mostra uma característica que só a filha mais velha tem, descartando a possibilidade das filhas mais velhas serem gêmeas.

Filha 1	Filha 2	Filha 3		Soma	
1	6	6	=	13	←
2	2	9	=	13	←

Logo, as idades são 2, 2 e 9.



Dica: você pode caracterizar melhor as personagens do desafio, porém, cuidado para não perder alterar as 3 dicas que são importantes para resolver o desafio.

* Esse desafio é uma adaptação do enunciado ”O mais velho tem olhos azuis”.



Teste dos Labs

Habilidades: Abstração, Decomposição. Coleta e Análise dos Dados, Simulação e Testes

O objetivo do Desafio Teste dos Labs é desenvolver a capacidade de interpretação do problema e a estruturação da solução, refletindo nos passos necessários para chegar no resultado do desafio. O desafio possui apenas uma resposta correta e entender as dicas é essencial para chegar na solução.

Contexto do Desafio

O Desafio Teste dos Labs consiste em caracterizar cada laboratório com base em dicas e nas relações entre as dicas apresentadas. Neste teste, há 5 Laboratórios do Departamento de Informática (DInf) da Universidade Federal do Paraná, e cada um deles tem: uma área de pesquisa, docente representante, linguagem de programação, e um livro recomendado para estudo (as opções aparecem na planilha). O desafio é resolver o teste com base nas dicas e nas relações entre elas, e nenhuma opção é utilizada mais que uma vez e todas devem ser devidamente alocadas.

	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5
Nome do Lab					
Docente					
Área de Pesquisa					
Linguagem de Programação					
Livro Recomendado					

** O Teste dos Labs foi baseado no teste popularmente chamado “Teste de QI de Einstein” e adaptado de um teste criado por Gustavo Roberto Tobias Tomaz, discente de CI1003 no 2º Semestre de 2019.

As opções para os labs, docentes, áreas de Pesquisa, linguagens de programação e livros estão indicadas abaixo. Nenhuma opção é utilizada mais que uma vez e todas devem ser devidamente alocadas, são possibilidades:

1. **Nome do Lab:** C3SL, FAES, HiPES, LIAMF, VRI
2. **Docente:** Castilho, Fabiano, Letícia, Todt, Zanata
3. **Área de Pesquisa:** Arquitetura de Computadores, Computação Científica, Engenharia de Software, Inteligência Artificial, Robótica
4. **Linguagem:** C, Go, Java, Pascal, Python
5. **Livro recomendado:** Algoritmos: teoria e prática, Artificial Intelligence: a modern approach, Informática em Saúde, Organização e Arquitetura de Computadores, The Art of Computer Systems Performance Analysis

Evite adotar estratégias de tentativa e erro, e preste atenção nas dicas abaixo:

1. Letícia trabalha no 1º Lab;
2. Fabiano trabalha no LIAMFI;
3. Castilho recomenda o livro “Algoritmos: Teoria e Prática”;
4. Zanata trabalha com pesquisa na área de Arquitetura de Computadores;
5. VRI fica à esquerda do C3SL;
6. Quem fica no VRI trabalha com pesquisa na área da Robótica;
7. Quem usa Python recomenda o livro “Artificial Intelligence: A Modern Approach”;
8. Quem fica no FAES usa Java;
9. Quem fica no Lab do centro pesquisa na área de Inteligência Artificial;
10. Quem usa Go fica ao lado de quem recomenda o livro “Artificial Intelligence: A Modern Approach”;
11. Quem recomenda “The Art of Computer Systems Performance Analysys” fica ao lado de quem usa Java;
12. Quem usa Pascal pesquisa na área da Computação Científica;
13. Todt usa C;
14. Castilho fica ao lado do VRI;
15. Quem usa Go é vizinho de quem pesquisa na área de Engenharia de Software; e
16. Quem recomenda o livro "Organização e Arquitetura de Computadores" fica ao lado do C3SL.



Dica: você pode caracterizar o desafio com as informações do departamento e laboratórios onde o desafio será realizado.

Resposta do Desafio

	Lab 1	Lab 2	Lab 3	Lab 4	Lab 5
Nome do Lab	FAES	HIPES	LIANF	VRI	C3SL
Docente	Letícia	Zanata	Fabiano	Todt	Castilho
Área de Pesquisa	Engenharia de Software	Arquitetura de Comp	Inteligência Artificial	Robótica	Computação Científica
Linguagem de Programação	Java	Go	Python	C	Pascal
Livro Recomendado	Informática em Saúde	The Art of Computer ...	Artificial Intelligence	Org e Arq de Computadores	Algoritmos



Desafio Lista de Presença

Habilidades: Abstração, Algortimos e Simulação/Teste

O objetivo do Desafio Lista de Chamada é propor um modo estruturado e otimizado de passar a lista de presença da aula para que ela seja assinada por todos os estudantes presentes. Neste desafio, os estudantes serão apresentados ao conceito de resolução de problemas e simulação e testes da solução. Assim, ao observar os testes e a execução será possível fazer uma reflexão sobre a importância de planejar e testar soluções.

Contexto do Desafio

Nas aulas de Introdução à Ciência da Computação na Universidade Federal do Paraná a frequência é registrada via assinatura na lista de chamada impressa. A lista de chamada é dividida em duas colunas, uma com o nome de todas as pessoas matriculadas na disciplina e outra para assinatura. Geralmente, a lista é liberada para assinatura após a metade da aula, e cada estudante presente deve encontrar o seu nome, assinar na linha correspondente, e passar a lista adiante.

Na sala, normalmente há um número variável de estudantes, dispostos em carteiras que estão distribuídas em um número variável de fileiras. A lista é entregue aleatoriamente, de maneira não definida, a uma estudante na primeira carteira de uma das fileiras. A lista é passada de um estudante para outro, e quem decide para onde a lista vai é quem está com a lista em mãos. Geralmente, por não haver uma definição sobre como passar a lista de chamada, pode ocorrer algumas situações que atrasem o processo de assinatura, como: (i) passar duas vezes pela mesma fileira; (ii) não passar por uma fileira ou estudante; (iii) percorrer o caminho mais longo.

Os estudante que não conseguiram assinar a lista assinam no fim da aula, porém, isso geralmente gerar atraso para a próxima turma que precisa usar a sala, ou resultar em uma falta para quem não conseguiu assinar a lista. Assim, o processo de percorrimento da lista de presença precisa ser otimizado para que no fim da aula todos tenham assinado a lista.

O desafio é propor uma maneira alternativa e mais rápida de assinatura da lista de chamada de maneira otimizada. São restrições do desafio: (i) a configuração da sala deve ser mantida, e as pessoas sentam onde elas quiserem; e (ii) a lista deve ser impressa e assinada pelas pessoas presentes e matriculadas na disciplina.



Dica: o nome do curso, da univerisidade, a quantidade de estudantes presentes e a quantidade de listas podem ser adaptadas para a realidade de onde o desafio será realizado.

Desafio Criptografia

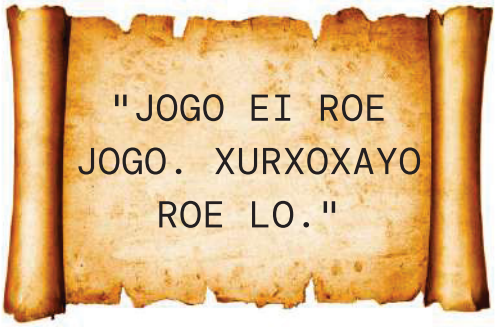
Habilidades: Abstração, Decomposição e Reconhecimento de Padrões

O objetivo do Desafio Criptografia visa apresentar o conceito Computacional de criptografia e segurança de dados.Neste desafio, trabalharemos também a codificação de informação, criptografia de mensagem e processo de descriptografia. O desafio apresenta uma frase criptografada e pede para que o estudante descriptografar a mensagem e explicar como descriptografou a mensagem.

Criptografia é uma técnica utilizada para codificar dados e informações para que apenas o destinatário consiga ler. A técnica de criptografar uma informação é feita a partir de um algoritmo que utiliza um padrão para codificar a mensagem, e o mesmo padrão deve ser utilizado para decodificar a mensagem. A criptografia é utilizada para proteger os dados do cartão, senhas, informações pessoais, entre outras.

Contexto do Desafio

Você recebeu uma mensagem criptografada e não consegue entender o seu conteúdo. A mensagem recebida foi:





O desafio é descriptografar a mensagem para entender seu conteúdo. Para isso, você precisa primeiro identificar como a mensagem foi criptografada. Após descriptografar a mensagem você deve explicar por meio de um conjunto de passos, infográfico ou fluxograma como descriptografar a mensagem.

Resposta do Desafio

A primeira etapa é tentar entender a lógica por trás da criptografia. Por se tratar de uma frase, podemos imaginar que ela resultará em uma outra frase que possui um sentido e que a frase será em Português.

Existe uma relação entre as palavras e as letras, mantendo-se uma relação a ordem das letras e seus tipos, é possível afirmar a descoberta de um padrão na mensagem. Os locais onde há vogais é realmente ocupado por vogais e assim também com as consoantes que substituem outras consoantes que formam a mensagem em Português e de forma compreensível. Assim, separamos as vogais e as consoantes em dois diferentes conjuntos:

Vogais: a, e, i, o, u

Consoantes: b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t ,v, w, x, y, z

A partir disso, diferentes estratégias podem ser seguidas para identificar qual o padrão da troca. Porém, como o número de vogais é menor, seria mais prático começar a descriptografia por elas. Na frase “JOGA EI ROE JOGO. XURXOXAYO ROE LO” contém um palavra de com apenas duas vogais, começaremos por ela: “EI”.

Um primeira estratégia possível é trocar por uma outra vogal, diferente dela mesma, que poderia resultar nas seguintes possibilidades: io, iu, ia, ie, ou, oa, oe, uo, ua, ue, ao, au, ae. Levando em conta a língua portuguesa, é possível apenas as seguintes opções: ae, ao, au, ia, ou.

Vogal	Possibilidades
E	I, O, U, A
I	O, U, A, E

As possibilidades de trocas resultam em 3 possibilidades de regras de trocas de vogais:

1. Volta 1 letra;
2. Avança 1 letra; e
3. Avança 2 letras.

Com isso, percebemos que existe um padrão de troca, que é repetido para o conjunto de vogais e consoantes. Agora só precisamos brincar com os padrões e verificar qual descriptografa a mensagem recebida.

Para esta mensagem o padrão de trocas entre vogais e consoantes são diferentes, ficando:

Consoante: volta 3 letras

Vogais: avança 2 letras

Aplicando esse padrão de troca na mensagem recebida você chega na mensagem original:
“FAÇA OU NÃO FAÇA. TENTATIVA NÃO HÁ.”

A frase é um dizer do Mestre Yoda, personagem da saga Star Wars.



Dica: pode-se alterar a mensagem criptografada para alguma frase temática, regional, ou interna do contexto onde o desafio será aplicado.

Plus: Achou que o desafio está muito fácil? Você pode pedir para criarem suas próprias técnicas de criptografar/descriptografar e acrescente uma nova etapa do desafio. Também, estudantes podem desafias uns aos outros para resolver a nova criptografia.



Desafio Busca

Habilidades: Reconhecimento de Padrões, Algoritmos, Generalização e Simulação e Teste

O objetivo do Desafio Busca é exercitar os conceitos de busca computacional, ou também conhecido como pesquisa. Neste desafio, trabalharemos também com a visibilidade de mulheres notáveis da Computação. O conjunto de cartas que representam mulheres notáveis da Computação está disponível para download, e podem ser impressas para serem utilizadas como material didático em sala. Assim, as soluções propostas para o desafio podem ser testadas em sala, e ao observar a execução será possível fazer uma reflexão sobre como os elementos, as informações e a organização dos elementos impactam na solução.

A busca computacional, também conhecida por pesquisa computacional, tem como princípio localizar um elemento específico dentro de um conjunto de elementos. Buscar um elemento pode ser considerado uma tarefa trabalhosa, principalmente quando o conjunto de elementos é grande e não ordenado. Em contrapartida, quando os elementos estão ordenados é possível encontrar meios mais eficientes de realizar a busca de um elementos específicos, realizar contagens, etc. Para realizar uma busca é necessário definir qual informação será considerada durante o processo, essa informação deve representar uma propriedade ou característica comum no conjunto de dados.

Contexto do Desafio

Considerando o conjunto de cartas que representam mulheres notáveis da Computação, cada carta apresenta uma mulher relevante para a história da Computação. Esses cartões podem ser ordenados de acordo com diferentes critérios, como: nome, data de nascimento, área, etc. O desafio é propor uma forma de ordenar um conjunto de n cartões, por nome ou por data de nascimento.

Neste desafios são restrições: (i) a sua solução deve funcionar para 1, 2, 3, ou *n* cartas (sendo *n* > *0*); e (ii) cada carta ocupa apenas uma posição, logo, duas cartas não podem ocupar uma mesma posição, e uma carta não pode ocupar duas posições ao mesmo tempo.

O desafio é criar uma forma de ordenar o conjunto de cartas e representar, na forma de passos ou fluxograma, a sequência de ações para ordenar as cartas.

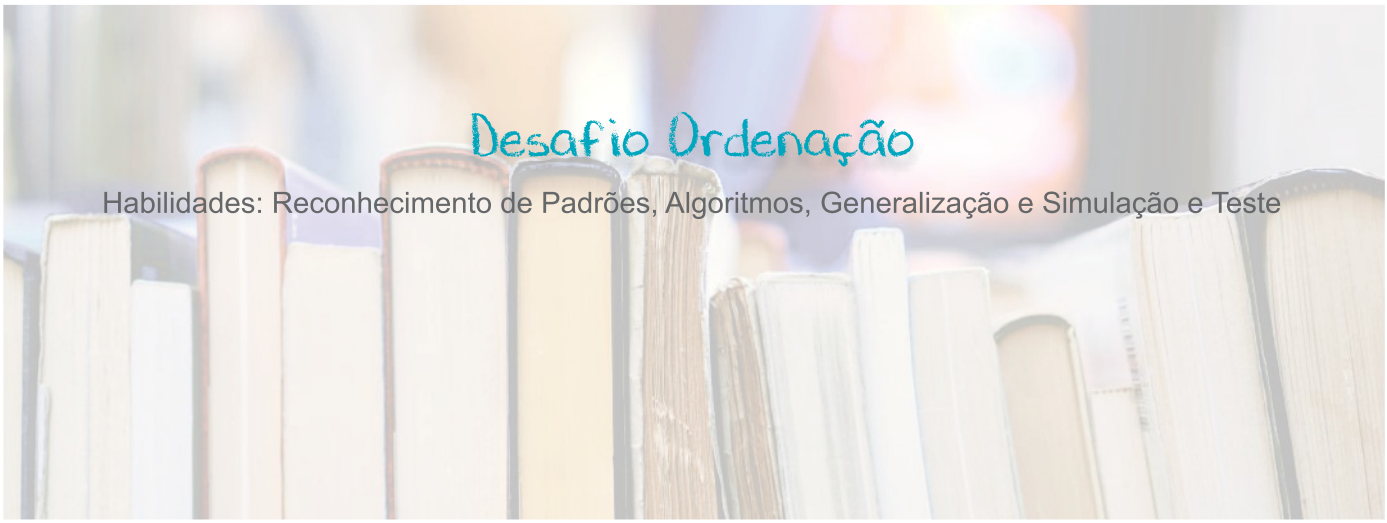
Dicas:

1. Exercite a ordenação com cartas físicas (imprima as cartas que representam mulheres notáveis da Computação, crie os seus, use livros etc.). Isso te ajudará a entender os passos (ações) necessários para a ordenação e a sua ordem.
2. Anote os passos e veja se é possível fazer de forma mais eficiente (por ex, com menos passos).
3. Teste sua solução com um conjunto pequeno de cartões (elementos), e depois vá aumentando a quantidade para ser se ela continua funcionando corretamente.
4. Mostre sua solução a alguém, e veja se essa pessoa consegue ordenar as cartas seguindo os passos da sua solução. Isso vai te ajudar a ver se a solução está fácil de entender.

ATENÇÃO: Sua solução deve ser representada de maneira clara e objetiva, a fim de que outra pessoa possa testar a sua solução sem ter dúvidas sobre as ações que deve executar.



Dica: o conjunto de elementos a ser ordenados deve possuir características comuns entre si, e que possam ser comparadas. O conjunto de cartas pode ser trocado por: dominós, carta de baralho, figurinhas.



Desafio Ordenação

Habilidades: Reconhecimento de Padrões, Algoritmos, Generalização e Simulação e Teste

O objetivo do Desafio Ordenação é exercitar os conceitos de ordenação computacional, ou também conhecido como *sorting*. Neste desafio, é utilizado o contexto da biblioteca da universidade, assim, é possível os estudantes visualizarem melhor o problema. A simulação da execução das soluções pode ser feita com livros reais, permitindo que eles reflitam sobre o comportamento da solução e remediem caso encontrem um problema.

Ordenação é o ato de colocar um conjunto de elementos em uma ordem predefinida. O termo técnico em inglês para ordenação é *sorting*, cuja tradução literal é "classificação". As principais razões para ser ordenar elementos/dados são:

- Fazer pesquisa ou busca eficiente,
- Permitir a fusão eficiente de sequências de dados, e
- Possibilitar o processamento de dados em uma ordem definida.

Ordenar um conjunto de dados pode ser uma tarefa muito trabalhosa, especialmente quando existem grandes quantidades de dados. É preciso, então, encontrar meios eficientes de se ordenar usando a menor quantidade de recursos possível. Não temos como garantir que todos os conjuntos de elementos estejam sempre ordenados do modo que precisamos. Então, é importante exercitar e conhecer os conceitos de ordenação.



Dica: o conjunto de elementos a ser ordenados deve possuir características comuns entre si, e que possam ser comparadas. O conjunto de cartas pode ser trocado por: dominós, carta de baralho, figurinhas.



Contexto do Desafio

Imagine que a partir de agora você trabalha na biblioteca da universidade e hoje é seu primeiro dia de trabalho. Você ainda está se familiarizando com o trabalho e, de repente, bem no meio da tarde tranquila, chega um carregamento de 1.280 livros diferentes.

Os entregadores deixaram os livros em uma linha reta, porém, estão todos desordenados. Para piorar as coisas, as aulas começam amanhã. O que significa que logo de manhã os estudantes vão procurar por esses livros. Como podemos organizá-los a tempo?

O desafio é desenvolver uma sequência de passos para ordenar os livros, na forma de fluxograma. Pense também na otimização da solução, visando diminuir a quantidade de trocas e comparações.

Neste desafio é possível apresentar diferentes estratégias de ordenação, podendo ser:

- Ordenação por Flutuação/Troca (Bubble Sort)
- Ordenação por Inserção (Insertion Sort)
- Ordenação por Seleção (Selection Sort)
- Ordenação por Pivô (Quick Sort)



Desafio Celebridade

Habilidades: Algoritmos, Generalização, Simulação e Testes e Otimização

O objetivo do Desafio Celebridade busca determinar se existe uma celebridade em uma festa e nomeá-la, caso exista uma. Para uma pessoa ser uma celebridade, ela precisa ser conhecida por todas as pessoas presentes na festa, mas não pode conhecer ninguém. A partir desse desafio é possível apresentar os conceitos de grafos e otimização.

Contexto do Desafio

Você agora é Jornalista do DInf News e deverá cobrir a Festa de Recepção de Estudantes que conta com muitas pessoas convidadas. Nesse ano, a tradicional Festa será híbrida e isso demanda uma interação diferente entre as pessoas da festa. O seu trabalho, como jornalista, é identificar se há uma celebridade na festa e informar quem é essa celebridade (caso exista uma).

Para ser uma celebridade, uma pessoa deve ser conhecida por TODAS as outras pessoas, mas não deve conhecer NENHUMA delas. Para identificar se há uma celebridade e quem é ela, você deve entrevistar as pessoas que estão na festa. Você só pode abordar uma pessoa por vez, e só pode fazer a seguinte pergunta: "Com licença, você conhece essa outra pessoa?" (apontando para outra pessoa presente na festa). A pessoa responderá Sim ou Não. Assuma que todas as pessoas (até mesmo a eventual celebridade) são educadas e responderão a pergunta com a resposta correta (Sim ou Não).

Para esse desafio, você precisa propor uma forma de identificar se existe ou não uma celebridade na festa. Se existir, deve nomeá-la; Se não existir, deve informar que não há.

Neste desafios são restrições:

- Sua solução deve funcionar para uma festa com n pessoas (sendo n >= 2);
- Você sabe com antecedência quantas pessoas estão presentes na festa,
- Você pode fazer apenas uma pergunta por vez, e a pessoa responderá "Sim" ou "Não" (e.g. "Ada, você conhece Grace Hopper?")
- Ao final, sua solução deve indicar quem é a celebridade, ou então informar que não há nenhuma.

Dicas:

- Simule a resolução do desafio (em papel, ou em uma planilha), pois isso te ajudará a entender os passos (ações) necessários e a sua ordem;
- Teste sua solução com um conjunto pequeno de pessoas (ex: n = 5) e depois vá aumentando a quantidade (n+1) para ver se ela continua funcionando corretamente;
- Analise a quantidade de perguntas necessárias para se obter a resposta e veja se é possível fazer de forma mais eficiente (com menos perguntas).
- Mostre sua solução a alguém, e veja se essa pessoa consegue encontrar a celebridade (ou dizer que não há) seguindo os passos da sua solução. Isso vai te ajudar a ver se a solução está fácil de entender.



Dica: O nome da festa, o curso, o ano, a quantidade de estudantes presentes podem ser alterados para a realidade de onde o desafio será realizado.

9. Referências

1. Wing, J. M. (2011). Research Notebook: Computational Thinking—What and Why? The Link Magazine, (March):June 23, 2015.
2. Nardelli, E. (2019). Do We Really Need Computational Thinking? Communications of the ACM, 62(2):32–35.
3. Wing, J. M. (2006). Computational Thinking. Communications of the ACM, 49(3):33–35.
4. Ministério da Educação (MEC) (2016). Diretrizes Curriculares Nacionais para os cursos de Graduação em Computação (DCN16). Resolução CNE/CES no 5, de 16 de novembro de 2016, 2016:9.
5. Zorzo, A. F., Nunes, D., Ecivaldo, S. e Martins, S. (2017). Referenciais de Formação para os Cursos de Graduação em Computação 2017. Sociedade Brasileira de Computação (SBC).
6. ACM, e IEEE. (2014). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Relatório técnico.