Curso de Graduação em Engenharia Da Computação

ESTUDO DO PENSAMENTO COMPUTACIONAL PARA RESOLUÇÃO DE PROBLEMAS COMPLEXOS

Aluno: Andrey Mateus Francisco Da Rocha

RGM: 28510356

Prof. Orientador: Gabriel Baião

Engenheiro Paulo De Frontin - RJ, Brasil

SUMÁRIO

1 RESUMO

Neste presente trabalho será abordado, o estudo da técnica de algoritmia conhecida como pensamento computacional e sua aplicação, utilizada para a resolução de problemas complexos, o estudo desta técnica e de seus pilares fundamentais tem com base, o leitor aprender a estrutura base e mais utilizada da técnica, que são seus pilares fundamentais e aplicação da técnica em problemas complexos, para isso teremos um total de **cinco capítulos**, começaremos no **resumo** que é aqui onde estamos, aqui te darei uma breve informação sobre oque você verá abaixo e se bate com as suas expectativas, seguindo da **introdução** falando sobre o nosso Objetivo e nossa Justificava que nos levou a tal trabalho, seguiremos com a **fundamentação teórica**, onde daremos uma base contextual sobre o tema, de onde surgiu, como surgiu, e o por que surgiu, também nos aprofundaremos nos pilaremos do tema e como funciona a sua estrutura e aplicação, seguindo adiante teremos a conclusão e **trabalhos futuros**, onde direi quais são as minhas propostas ou expectativas para este trabalha no futuro, e para finalizar teremos as **referências bibliográficas**, que dirão quais são as fontes que os textos da monografia são baseados ou inspirados.

2 INTRODUÇÃO

2.1 JUSTIFICATIVA

Em toda a vida do ser humano, o maior uso do seu tempo em vida sempre foi utilizando o seu cérebro para a resolução de problemas, desde a época da Pré-história, onde os maiores problemas do “ homem ” eram a sua comida, seu abrigo, construção de equipamentos de caça e como abater suas presas, fomos evoluindo gradativamente e juntamente disso o ambiente ao seu redor e os problemas gerados pelo homem também, chegamos ao período da idade Média, onde os maiores problemas do ser humano eram a Economia, Política, Sociedade e a Crise, saltamos então para o período da idade Moderna, sendo então os principais problemas, a produção capitalista, mundialização do comércio europeu e o colonialismo, e hoje onde vivemos no atual presente, o famoso período da idade Contemporânea, sendo conhecido pelos seus principais problemas, sendo eles: revoluções, guerras, avanço da ciência e da tecnologia e suas nuances, sendo uma dessas nuances o “ pensamento computacional”, tivemos a oportunidade de ler que cada período teve seus ambientes diferentes, e consecutivamente por seus ambientes serem diferentes, proporcionaram problemas diferentes, sendo o problema definido como um objetivo ou dificuldade que necessitou ou necessita ser sanada, concluso ou solucionado, e por mais que alguns desses problemas dos períodos antecessores possam parecer simples para alguns dos residentes de nossa época atual, acredito que todo problema seja complexo e tenha um nível próprio de complexidade para cada ser vivo que vive ou viveu em sua época, assim como creio que nos períodos futuros os nossos problemas atuais serão de baixa complexidade para os nossos descendentes, porém também salientarei que mesmo os indivíduos de uma mesma espécie e de um mesmo período, podem ter percepções distintas do que é um problema com alto e baixo nível e ou grau de complexidade.

Sabendo-se que todo indivíduo como um ser humano, pode ter as suas diferentes nuances, técnicas e dificuldades para solucionar um problema, proponho a utilização de uma técnica de algoritmia chamada de “pensamento computacional”, para a resolução de problemas complexos, onde temos em vista que todo problema é complexo, porém com níveis diferentes de complexidade para diferentes indivíduos, mesmo tendo em mente que essa técnica foi criada e destinada para “processadores de máquinas” realizarem cálculos complexos, é interessante saber que nosso cérebro tem a capacidade de processamento muito superior a maioria dos processadores de algumas máquinas atuais, e sabendo disso, podemos entender que conseguiremos utilizar dessa técnica para obtermos do mesmo resultado que os

processadores buscam, Eficiência e Alta performance, que vem por causa da organização, ordenação e outros pilares utilizados pela técnica, fazendo com que o nosso custo de tempo para resolução de um problema seja inferior ao que se tinha quando não se utilizava a técnica, seja o problema qual for, mas tenha em mente que abusar da técnica em problemas complexos que possuam grau ou nível inferior ao nível baixo, pode consecutivamente transformar o seu problema em um problema mais que complexo, um exemplo disto seria você aplicar a técnica em algo como, pegar uma fruta em sua geladeira, fritar um ovo ou coisas similares, por mais que utilizaremos de algoritmia para dividir o nosso problema em partes, é necessário analisar onde tal técnica surtirá mais efeito positivo do que negativo, pois imagina o tempo que você perderá aplicando a técnica para que você apenas precisasse ir em sua geladeira e pegasse uma fruta, entende como o tempo que você perderá pensando e analisando não compensará o tempo que a aplicação da técnica deveria reduzir, agora imagine que você precise construir uma casa, criar um software, limpar uma piscina, organizar suas finanças do mês e até mesmo ir as compras no mercado.

Analisar, identificar, compreender e solucionar o seu problema com a estrutura da técnica, é a justificativa principal para realização do presente trabalho, dado que o aumento de performance e redução de custo de tempo para a solução de um problema e aumento da facilidade na resolução, senão é a parte mais importante para que o ser humano possa obter mais tempo livre em sua vida e obter menos períodos de estresse.

2.2 OBJETIVO

O objetivo desta monografia é fazer com que qualquer indivíduo consiga utilizar-se do pensamento computacional para conseguir realizar a resolução de problemas complexos, tornar fácil o ato de solucionar problemas complexos utilizando a técnica de algoritmia e seus pilares, conhecida como pensamento computacional, sendo o principal problema a ser solucionado pela técnica, que é custo de tempo para a resolução de problemas sendo muito alto e a não facilidade em solucionar problemas complexos, o problema podendo ser desde a criação de um software ou até mesmo algo rotineiro.

Qual o intuito do pensamento computacional ?

O pensamento computacional não deve ser considerado um recurso a ser utilizado somente por cientistas da computação. Dessa forma, podemos considerar que o pensamento computacional envolve diversas tarefas, que não estão somente ligadas à resolução de problemas e à projeção de sistemas, mas também à capacidade de compreender o comportamento humano pela extração de conceitos fundamentais da ciência da computação. Algumas considerações ainda podem ser apontadas sobre o conceito de pensamento computacional e suas características:

- Compreender a forma como as informações são utilizadas em determinado cenário para resolver um problema complexo;
- Reformular um problema difícil em algo que sabemos como resolver;
- Ser capaz de julgar um programa pela sua estética, e a interface de um sistema pela sua simplicidade e elegância.

  (Beecher, 2019)

3 FUNDAMENTAÇÃO TEÓRICA

3.1 PENSAMENTO LÓGICO E ALGORÍTMICO

O termo pensamento computacional (do inglês, computational thinking) foi apresentado na década de 1980 por Seymour Papert e, em 2006, a pesquisaora Jeannette Wing publicou um artigo sobre o tema, que passou a receber atenção da comunidade científica e popularizou a ideia de utilizar a tecnologia como recurso interdisciplinar. A primeira impressão que se tem do tema é que está ligado ao uso da tecnologia, mas vai muito além disso, sendo fortemente relacionado ao estímulo ao raciocínio lógico e à criatividade. (SILVA et al , 2021)

O conceito visa ao desenvolvimento lógico e algorítmico para resolução de problemas não apenas da área da computação, mas de todas as áreas de conhecimento. Para isso, é de grande importância o estudo de dois outros conceitos: o pensamento lógico e o pensamento algorítmico. Ambas as áreas são habilidades que podem ser desenvolvidas por qualquer pessoa, e não somente por cientistas da computação e profissionais de tecnologia, e aplicadas nas atividades diárias, estimulando a criatividade e a produtividade por meio da lógica e da construção de algoritmos. (SILVA et al , 2021)

3.2 LÓGICA E PENSAMENTO COMPUTACIONAL

O pensamento computacional pode não ser um conceito novo por estar ligado às raízes da própria ciência computacional. Embora já houvesse trabalhos que difundiam a sua relevância para a construção do conhecimento e desenvolvimento do pensamento criativo (PAPERT,1980), o conceito ganhou reforço nos últimos tempos. Alguns autores como Jeanette Wing, vice-presidente da Microsoft Research e pesquisadora reconhecida na comunidade acadêmica, auxiliaram a analisar de que forma como a ciência computacional analisa o mundo e soluciona problemas seria aplicável a diversos outros contextos (WING, 2006).

Para Wing (2006), o pensamento computacional não está ligado apenas à programação de computadores, mas também à habilidade de pensar logicamente, algoritmicamente e recursivamente. Assim, a ideia não é que as pessoas pensem como um computador, que é limitado ao conhecimento de seu programador, mas que apliquem os conceitos computacionais como estratégia para resolver de forma eficaz problemas nas diversas áreas de conhecimento e no seu cotidiano. Em sua definição, o conceito é baseado em quatro pilares fundamentais, os quais são descritos a seguir. (SILVA et al , 2021)

1. **Decomposição**: segmentar (dividir) problemas complexos em partes mais simples de resolver.
2. **Reconhecimento de padrões**: após a decomposição de um problema, é possível identificar similaridades que algumas dessas partes compartilham. Assim, é possível fazer uso de soluções utilizadas previamente com base em experiências anteriores.
3. **Abstração**: elementos não necessários para resolução do problema são ignorados para que se possa fixar o foco nos detalhes relevantes. Aqui o desafio é selecionar os detalhes que facilitem a compreensão do problema sem perder informações que possam ser importantes para a resolução.
4. **Algoritmo**: o ponto de junção de todos os demais, representado pelo conjunto de instruções claras e finitas necessárias para a solução do problema.

Um dos pilares com grande abrangência em diversas atividades do pensamento computacional, o desenvolvimento do pensamento algorítmico está intrinsecamente ligado ao desenvolvimento da lógica do indivíduo. A seguir, você irá conhecer um pouco mais sobre o seu papel no pensamento computacional. (SILVA et al , 2021)

3.3 EVOLUÇÃO DA LÓGICA COMPUTACIONAL

Embora não seja especificado como um Pilar, o pensamento lógico permeia todas as atividades que envolvem o pensamento computacional. A lógica clássica, difundida por Aristóteles e outros pensadores antigos, teve sua origem com o uso de silogismos, uma forma de raciocínio simbolizado por declarações, também chamados de premissas, que descrevem fatos ou conhecimentos de senso comum, e serviram de base para a matemática e para a própria ciência computacional. Tal representação ordenada e sequencial das premissas descrevem um caminho para resolução do problema. Como exemplo, observe as premissas a seguir.

1. Duas pessoas podem ser consideradas irmãs quando têm ao menos o mesmo pai ou a mesma mãe.

2. Maria é filha de Augusto.

3. João é filho de Augusto.

Considerando que Augusto é mesma pessoa das premissas **2** e **3**, podemos inferir que João e Maria são irmãos. Essas representações também são amplamente utilizadas na matemática, na formulação de teoremas, como: "Se duas grandezas são idênticas a uma terceira, então são idênticas entre si". Este teorema pode ser formalizado e demonstrado como: "Se **A**=**C** e **B**=**C**, Logo **A**=**B**". (SILVA et al , 2021)

Observe que essa representação matemática é totalmente simbólica: **A** pode ser qualquer objeto ou valor; mesmo assim, a representação matemática ainda continuará válida.(SILVA et al , 2021)

A lógica é empregada em todas as áreas das ciências, e o conhecimento científico está formalmente representado segundo a lógica, bastante impulsionada por diversos nomes, como George Boole (1815—1864), Gottfried Wilhelm Leibniz (1646—1716), Alfred North Whitehead (1861—1947), Alan Turing (1912—1954), Bertrand Russell (1872—1970) e muitos outros (FIGUEIREDO; LAMB, 2016). Essa ciência deu um salto exponencial a partir do século XX, desde a automatização de verificações matemáticas para a corrida espacial até a atualidade, com a construção de modelos computacionais e simulações para as mais diversas finalidades (modelos climáticos, biomedicina, indústria, esportes de rendimento, mercado financeiro, dentre outras) (FIGUEIREDO; LAMB, 2016).

3.4 A LÓGICA E A RESOLUÇÃO DE PROBLEMAS

Quando se fala em resolução de problemas, frequentemente são encontrados métodos de raciocínio lógico: o **indutivo** e o **dedutivo**. Em alguns momentos, eles podem até parecer sinônimos, mas existem diferenças entre eles. (SILVA et al , 2021)

***** É **importante** destacar que o termo **problema** deve ser entendido como qualquer informação, descoberta ou mesmo solução que se deseja encontrar, e não é necessariamente uma situação difícil que precisa de solução. Nesse contexto, esse problema como é tecnicamente denominado desafio lógico. (SILVA et al , 2021)

3.4.1 MÉTODO INDUTIVO

As premissas ou fatos conduzem a uma "lei de formação" pela repetição de algum padrão de comportamento (LAW, 2008). Veja um exemplo.

- Uma caixa contém 99 bolas vermelhas e uma bola preta.
- 100 pessoas retiraram uma bola da caixa.
- João é uma dessas 100 pessoas.
- Logo, João provavelmente tirou uma bola vermelha.

No método indutivo, deve-se **analisar** as premissas sobre o problema, buscando padrões que sigam uma lei de formação que possa levar a uma solução. Para esse exemplo, observe que somente uma pessoa irá retirar a bola preta, portanto, conclui-se que a maior probabilidade é de que se retire uma bola vermelha. O método indutivo é utilizado com **cautela** no meio científico, pois as conclusões obtidas por meio dele podem conduzir a falsas conclusões. (SILVA et al , 2021)

A forma do argumento não garante que a conclusão seja verdadeira, mas **provavelmente** resulta em uma conclusão confiável e próxima à realidade. (SILVA et al , 2021)

3.4.2 MÉTODO DEDUTIVO

Neste método, as premissas são avaliadas e ordenadas em uma sequência de causa e efeito, de modo que a conclusão obtida seja resultante da sequência completa, emergindo como fato novo extraído ou deduzido a partir das premissas constituintes (LAW, 2008). Veja um exemplo com as seguintes premissas.

- Sempre que chove, o gramado irá molhar.
- Hoje está chovendo.
- Logo, o gramado encontra-se molhado.

O método dedutivo pode precisar de maior quantidade de premissas (**todas válidas**) para, então, levar a uma conclusão também válida. Esse método é bastante utilizado no método científico, pois busca encontrar a verdade a partir da avaliação de fatos e conhecimentos que possam ser verdadeiramente comprovados, **sem qualquer generalização**. (SILVA et al , 2021)

Os computadores usam a lógica da maneira como fazem cálculos, mas isso não é exatamente o mesmo que pensar logicamente no sentido do pensamento computacional. Os próprios computadores devem ser programados (ensinados) para fazer raciocínio lógico por meio de algoritmos (Conjunto de instruções). A lógica de computacional é fundamental na criação dos algoritmos, pois tem por objeto de estudo as leis gerais do pensamento e as formas de aplicar essas leis corretamente na investigação da verdade. (SILVA et al , 2021)

3.5 PENSAMENTO ALGORÍTMICO

O pensamento lógico é utilizado cotidianamente sem que as pessoas percebam, por exemplo, ao acordarmos pela manhã e darmos início a uma sequência de etapas até que estejamos prontos para sair de casa, que é quando já estamos utilizando o pensamento lógico e algorítmico sem perceber. Caso essas etapas fossem descritas em ordem de execução como itens de tarefas, estaríamos construindo um algoritmo (Conjunto de instruções). Na ciência computacional não é diferente, pois também podemos solucionar problemas utilizando o mesmo pensamento **lógico-algorítmico**, descrevendo instruções em etapas bem definidas com a finalidade de atingir determinado objetivo. (SILVA et al , 2021)

No momento em que é feita a transição do pensamento lógico para o pensamento algorítmico, é necessário analisar que esses conceitos não são os mesmos. Com a lógica, você tem a compreensão do problema e como ele pode ser subdividido e encontra possíveis soluções para ele. Já no momento em que descrevemos todas as tarefas necessárias e a ordem em que elas devem ser executadas para que o problema seja resolvido, estamos criando um algoritmo. (SILVA et al , 2021)

De acordo com Beecher (2017), essa sequência de passos pode ser utilizada para explicar a um terceiro (seja humano ou uma máquina) como resolver o problema, seguindo tais regras com precisão. Assim como já ocorre com o pensamento lógico, o ser humano possui uma compreensão intuitiva do que é um algoritmo, tão utilizado pela ciência computacional, e

desenvolver seu pensamento algorítmico torna-se essencial para o profissional da computação, pois um algoritmo correto é a base final de qualquer solução executada por computador. (SILVA et al , 2021)

Estudiosos da programação como Ada Lovelace (1815—1852), considerada a primeira programadora da história, buscaram formas de comunicar ideias para as máquinas para que elas pudessem compreender e computar por nós, humanos (BEECHER, 2017). Então, concluiu-se que o nosso modelo intuitivo era insuficiente, gerando a necessidade de formular regras de forma clara e inequívoca, como os computadores exigem, evoluindo para estruturas para fornecer instruções aos computadores. Por mais precisos que tenham se tornado, os algoritmos também se tornaram mais complexos, tanto que uma descrição formal deles pode se estender por vários parágrafos. Em nosso caso, a melhor maneira de compreender sua construção é estudar suas estruturas básicas. (SILVA et al, 2021)

3.6 ALGORITMO

É a habilidade de montar **uma sequência lógica para resolver o problema**. Normalmente, em nosso dia a dia quando vamos fazer uma determinada tarefa, já temos uma sequência lógica de como realizar determinada função, mas muitas vezes isso acaba sendo feito de uma forma automática que acabamos por não percebê-la.

Suponhamos que vamos fazer um bolo. Muitas vezes, costuma-se seguir uma determinada receita para obtermos o resultado correto. Sendo assim, se não seguirmos, pode ocorrer o fracasso da receita. (NOLETO, BETRYBE, 2022)

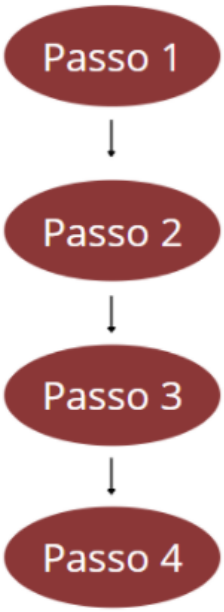

***Figura 1**: Algoritmo - **Fonte**: BLOG.BETRYBE*

*** Aviso do Autor da Monografia**: Não se preocupe caso não entenda os passos abaixo, o meu foco é que você leitor entenda **TODO** o processo para a **aplicação** da técnica, o tópico a seguir é apenas uma inserção detalhada e mais profunda do que realmente é um algoritmo, o entendimento completo de como um algoritmo é composto é mais voltado para desenvolvedores de software, se o seu caso é apenas um problema que não será transcrito para a criação um software (programa de computador), fique limitado a explicação de algoritmos acima e pule para o **capítulo 3.8**.

3.7 ESTRUTURA BÁSICA DE UM ALGORITMO

A estrutura básica de um algoritmo é formada por vários componentes que determinam como se dará o fluxo de execução da tarefa, desde a inicialização até sua conclusão (MOKARZEL; SOMA, 2008). Para que se possa estruturar o pensamento lógico, é preciso atender a alguns conceitos, os quais são apresentados a seguir.

• **Precisão**: cada etapa em um algoritmo pode ter apenas um significado; caso contrário, a ambiguidade poderá "travar" a sua execução.

• **Ações sequenciais**: as etapas que compõem o processo devem ser realizadas na ordem especificada.

• **Controle dos estados**: deve-se ter controle do status de todas as informações gerenciadas pelo sistema em cada momento de execução.

• **Repetição de tarefas**: algumas etapas devem ser executadas múltiplas vezes para evitar que seja preciso reescrevê-las a cada momento.

(SILVA et al , 2021)

3.8 PENSAMENTO COMPUTACIONAL: RESOLUÇÃO DE PROBLEMAS

A resolução de problemas é uma área importante para diversas áreas de conhecimento, tendo grande relevância nas ciências computacionais. Segundo os conceitos de pensamento computacional, um problema não precisa ser necessariamente resolvido de forma computacional, mas, sim, de forma sistemática, podendo, também, ser resolvido por meios tecnológicos. Porém, encontrar a solução para um problema, mesmo para problemas simples, pode não ser uma tarefa muito trivial. Um exame mais detalhado do problema pode revelar detalhes que não foram percebidos inicialmente, nuances complexas e requisitos a serem considerados. Para tanto, definir sistematicamente soluções para problemas pode ser uma boa estratégia para que seus resultados possam ser replicados e servir de base para aplicação em contextos semelhantes. (SILVA et al , 2021)

3.9 CONCEITO DE PROBLEMA E RESOLUÇÕES SISTEMÁTICAS

Um problema pode ser compreendido como uma situação ou um acontecimento que necessita de uma solução. Em uma perspectiva conceitual, são situações em que um indivíduo necessita resolver, mas não dispõe de uma forma direta que a leve à solução (POZO, 1998). Assim, conforme Pozo (1998), para definirmos uma situação-problema, ela deve ser complexa o suficiente para provocar a reflexão ou uma tomada de decisão de qual passos seguir para encontrar a solução. (SILVA et al , 2021)

Pensar sobre as estratégias utilizadas na solução de uma situação possibilita, ao estudante, planejar e organizar suas próprias atividades de resolução de problemas, seguindo um conjunto de regras que configuram uma lógica (SNEIDER et al., 2014).

A lógica tem papel importante na elucidação de problemas, tendo raízes na época de Aristóteles e no estudo dos silogismos. Nos dias atuais, é utilizada em diversas áreas de conhecimento técnicas de modelagem computacional e em simulações de cenários para resolução de problemas de alta complexidade, em um mundo onde a informação é armazenada, acessada e manipulada por software (RILEY; HUNT, 2014).

Computacionalmente, a resolução de um problema engloba o desenvolvimento de métodos para a solução de problemas independentemente da área do conhecimento, soluções que podem ser implementadas ou não em um dispositivo computacional. Essa sequência de passos ordenados que resolve determinado problema é conhecida como algoritmo (PIVA JR. et al., 2012).

Para o pensamento computacional, o mais importante não é programar a solução, mas desafiar o raciocínio, propondo formas de solucionar problemas. Caso você consiga organizar seu pensamento para compreender "como" resolver o problema, a implementação será apenas a transcrição do pensamento para uma forma escrita em alguma linguagem de programação. Segundo Ribeiro, Foss e Cavalheiro (2017), programar é a parte fácil: o complicado é saber construir a solução do problema. (SILVA et al , 2021)

Além da área da computação, diferentes áreas de conhecimento definem estratégias sistemáticas para sua resolução. Por exemplo, no campo da matemática, área que impulsionou o desenvolvimento da computação, Polya (1977) definiu uma possível sequência de passos para a resolução de qualquer problema:

1. compreender o problema;

2. estabelecer um plano para sua resolução;

3. executar o plano;

4. examinar a solução obtida.

De um modo geral, as etapas de Polya (1977) têm relação com os operadores do pensamento computacional (análise e abstração), sendo que a diferença é que a matemática traz a resposta para um problema, enquanto a computação descreve o como fazer para que seja possível que máquinas resolvam de forma automática esses problemas. Apesar de as áreas em questão

compartilharem as habilidades de análise e abstração, a automação, habilidade de implementar a solução pensada para ser executada por uma máquina, é própria da computação. (SILVA et al , 2021)

Na etapa de resolução, a implementação e os algoritmos ainda não são necessários; o momento deve ser dedicado somente à compreensão do objetivo. Você conhece a frase "Como saber quando chegou ao destino se não sabes aonde estás indo?". Pois ela se aplica aqui: antes de pensar em como implementar, deve-se identificar quando o problema estará resolvido, quando se pode dizer que o objetivo foi atingido. O objetivo pode ser automatizar parte do processo de uma fábrica ou aumentar em 30% a eficiência da equipe de vendas. Independentemente de como é definido o objetivo, ele deve ser descrito de forma clara e específica. Por exemplo, caso o objetivo seja incrementar a velocidade do atual sistema, a descrição não deve ser simplesmente "deve ser mais rápido", mas ele deve ser descrito de forma que possua precisão mensurável. Por último, é interessante criar a especificação dos objetivos como se outra pessoa pudesse eventualmente pegar o que você produziu e verificar por si mesma se você resolveu o problema, avaliando se consegue replicar o procedimento e obter os mesmos resultados. Quanto à solução de problemas, é relevante levar em consideração alguns pontos. Vejamos.

- **Qualidade**. Para qualquer problema, geralmente existem várias soluções, algumas mais adequadas que outras. Você deve se concentrar em encontrar a melhor solução possível. Talvez não exista uma solução perfeita, mas, por outro lado, partes individuais de um problema podem ser "aperfeiçoadas" à medida que você vai compreendendo melhor o problema e otimizando a solução.
- **Colaboração**. Um esforço em equipe pode auxiliar na resolução de um problema. Simplesmente explicar seu trabalho em voz alta pode auxiliar a identificar erros ou melhorias potenciais. Duas pessoas podem ter pontos de vista diferentes sobre o mesmo problema, muitas vezes complementares.
- **Iteração**. São as várias tentativas que você executa até encontrar a solução, e raramente sua primeira tentativa será a melhor. Em vez de tentar resolver tudo na primeira vez, opte uma abordagem iterativa, repetindo etapas anteriores na tentativa de aprimorar sua solução atual. Cada iteração trará uma nova visão da solução e melhorias ao resultado.

Tendo em mente esses pontos, passamos, então, a buscar estratégias para resolver o problema especificado. Como você verá em breve, o pensamento computacional destaca a estratégia de **decomposição** como uma das formas mais aceitas para a resolução de problemas, inclusive sendo a decomposição um dos quatro pilares do pensamento computacional. Nas próximas seções, o papel do pensamento computacional na resolução de problemas e a estratégia de decomposição serão mais bem detalhados, e também veremos brevemente algumas outras formas que podem ser aplicadas. (SILVA et al , 2021)

3.10 DECOMPOSIÇÃO

Como uma das principais heurísticas relacionadas ao pensamento computacional, a decomposição se trata da divisão de um problema complexo em partes mais simples e fáceis de gerenciar. Computacionalmente, um desenvolvedor, com frequência, lida com problemas complexos compostos por grande quantidade de partes inter-relacionadas. Embora outras heurísticas possam ser úteis em alguns casos, a decomposição ainda é o método mais aceito para a resolução de problemas em que o objetivo é uma solução computacional. Em especial, a decomposição ajuda na colaboração. Se você decompuser bem um problema (de forma que os subproblemas possam ser resolvidos de forma independente), pessoas diferentes podem trabalhar em tarefas diferentes, possivelmente em paralelo. (SILVA et al , 2021)

A estratégia utilizada pela decomposição é conhecida como divisão e conquista, bastante utilizada na computação, mas não exclusiva desta. Podemos encontrar a estratégia em outras áreas de atuação, como na guerra, por exemplo, em que se força o inimigo a se separar, formando grupos menores e com menos resistência ao ataque, depois derrotando um grupo de cada vez. (SILVA et al , 2021)

Para deixar mais claro, vamos a um exemplo. Na vida acadêmica, em algum momento, você já produziu, ou vai produzir, alguma publicação (monografia, artigo, dissertação, livro, etc.) e, por si só, esse problema pode gerar certa ansiedade. Parte dessa sensação se deve a tratar o problema como uma tarefa única ("escrever uma publicação"), um conjunto de dezenas de milhares de palavras, de forma que geralmente não sabemos sequer por onde começar. Porém, aplicando a decomposição ao problema principal, teremos as seções menores ilustradas no esquema da Figura 2. (SILVA et al , 2021)

***Figura 2**: Decomposição de um trabalho acadêmico. - **Fonte**: Adaptada de Beecher (2017).*

Cada uma dessas seções possui menos detalhes, de modo que podemos concentrar a atenção em cada parte e solucioná-las de forma muito mais eficaz. Ao resolver cada subproblema (escrever as seções), você acaba resolvendo o problema geral. Ainda assim, essa primeira iteração apenas decompôs um pouco a tarefa principal; algumas das subtarefas ainda podem ser muito grandes. Podemos continuar aplicando a decomposição de forma **recursiva**, segmentando cada sessão em novas subtarefas. Vamos pegar como exemplo a seção Desenvolvimento, que pode ser subdividida como:

- **Trabalhos relacionados**
- **Metodologia**
- **Análise dos resultados**

Ainda, a subseção Metodologia pode ser subdividia em:

- **Desenvolvimento do modelo**
- **Experimentação**

(SILVA et al , 2021)

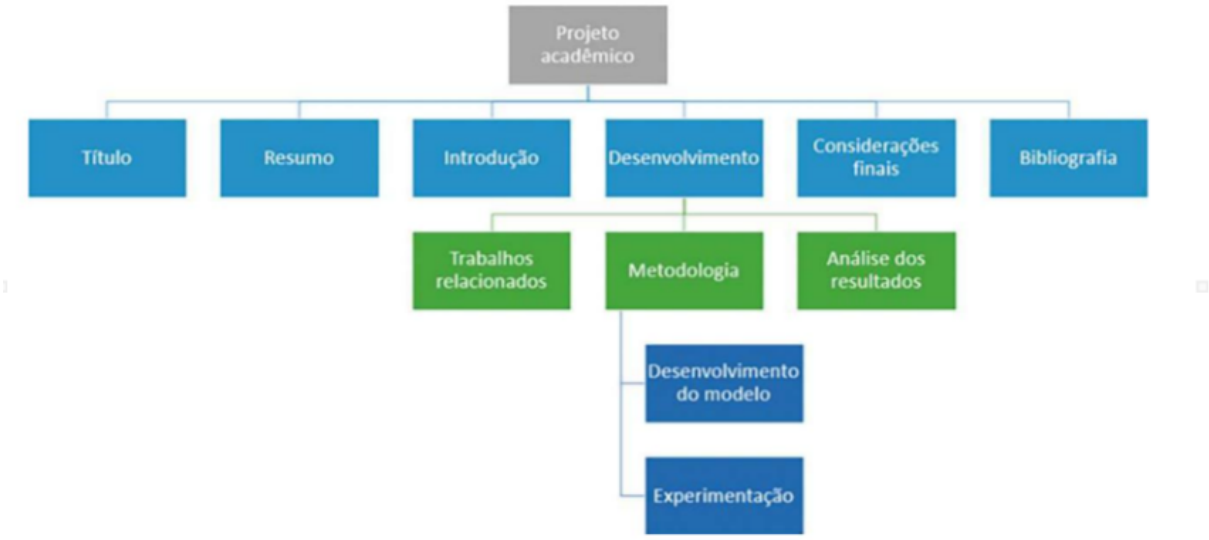

***Figura 3**: Decomposição dos subproblemas identificados: segundo nível, após a decomposição do subproblema Desenvolvimento, e terceiro nível, após a decomposição do subproblema Metodologia. -* ***Fonte**: Adaptada de Beecher (2017)*

O processo de decomposição pode seguir conforme a necessidade em um processo recursivo, em que cada nova subtarefa é mais simples que seu conjunto original, contendo não mais do que algumas centenas de palavras, e alguns até podem ser gerados dinamicamente pelo próprio processador de texto. Após resolvido cada problema, começamos a juntar as partes e, ao final, teremos a solução completa. O uso da recursividade é indicado para problemas em que é possível subdividir o problema principal em partes semelhantes ou iguais. Um exemplo é a ordenação de árvores: cada "galho" da estrutura é uma versão semelhante ao "tronco" principal. O conceito de árvore é chave para o estudo de estruturas de dados, em que a árvore representa um conjunto de entidades organizadas em relacionamentos hierárquicos. Cada

entidade "pai" pode ou não ter uma quantidade de entidades "filhas". O nome se deve a seu formato, que se assemelha a uma árvore invertida e pode representar uma grande variedade de estruturas diferentes. No exemplo do texto, a raiz é o problema principal "escrever uma publicação", e os primeiros galhos seriam Título, Resumo, etc. Cada galho, ainda, pode ter os próprios galhos. No exemplo, do galho Desenvolvimento, saem Trabalhos relacionados, Metodologia e Análise dos resultados. (SILVA et al , 2021)

Observação: Lembre-se de que a decomposição é o processo de subdividir o problema para que ele fique mais simples e gerenciável: ainda não é necessário pensar em detalhes de como resolver os subproblemas. Nessa etapa, como comentamos, a preocupação é compreender "o que" fazer, não "como" fazer. Também cabe frisar que a decomposição não necessariamente dita a ordem em que as tarefas devem ser executadas, mas pode auxiliar a definir um plano de ação. Uma definição de problema totalmente decomposta pode mostrar todas as tarefas individuais, mas não necessariamente a ordem em que você deve lidar com elas. No entanto, as relações entre subproblemas individuais devem se tornar aparentes. (SILVA et al , 2021)

3.11 PAPEL DO PENSAMENTO COMPUTACIONAL NA RESOLUÇÃO DE PROBLEMAS

Desde sua origem até meados dos anos 1960, a ciência computacional era voltada para o estudo do hardware, quando, até então, as limitadas aplicações de software eram voltadas para a resolução de problemas matemáticos, utilizados nas faculdades e em grandes empresas (DENNING; TEDRE, 2019). A partir de então, com a popularização dos computadores, aumentou o leque de problemas que poderiam ser solucionados com o uso do computador, impulsionando o surgimento de novas áreas de estudo, como a engenharia de software, que aplica técnicas de engenharia para analisar problemas e projetar e implementar aplicações capazes de resolvê-los (RILEY; HUNT, 2014).

Atualmente, a computação está amplamente disseminada em nosso cotidiano. Seja fornecendo ferramentas de suporte para negócios, planejamento estratégico e tomada de decisões, ou de forma embarcada em seu celular ou televisor, para acesso a algum canal de streaming de vídeo: onde houver novas necessidades ou problemas, lá estará a computação para auxiliar a resolução. Diversas descobertas não teriam sido possíveis sem a utilização de computadores, como o sequenciamento do genoma humano, por exemplo. Novas áreas de concentração, como inteligência artificial, segurança da informação, robótica, redes e computação gráfica, surgiram conforme novos problemas foram sendo apresentados (RILEY; HUNT, 2014).

No entanto, você deve estar se perguntando: como o pensamento computacional se encaixa nessa discussão sobre hardware, software, abrangência da ciência da computação e de suas áreas correlacionadas? Por trás de todos esses recursos, existem pesquisadores, analistas, desenvolvedores, entre outros profissionais empenhados em projetar e atender essas necessidades, podendo ser considerados verdadeiros **solucionadores de problemas**, dada a importância que as máquinas ganharam em nossas vidas (RILEY; HUNT, 2014).
Segundo Wing (2006), o pensamento computacional faz uso de conceitos da ciência da computação para criar estratégias para a resolução de problemas que podem ser aplicadas a toda e qualquer área do conhecimento (WING, 2006). Ou seja, o pensamento computacional pode ser entendido como a forma como os cientistas da computação pensam, racionalizam e resolvem problemas. (SILVA et al , 2021)

Observação: O pensamento computacional não é voltado somente para problemas computacionais, mas serve para qualquer indivíduo. Segundo Nunes(2011), o pensamento computacional é um processo cognitivo, uma forma de sistematização e de organização para auxiliar a solucionar problemas. A ideia não é que as pessoas pensem como um computador, mas que apliquem os conceitos computacionais como estratégia para resolver, de forma eficaz, problemas nas diversas áreas de conhecimento e em seu cotidiano. (SILVA et al , 2021)

Devido ao vasto volume de conhecimento acumulado durante as últimas décadas, é fato que não é viável utilizar todos os conceitos da computação. Portanto, o pensamento computacional faz uso de alguns conceitos-chave emprestados das diversas áreas de conhecimento da ciência da computação que dão suporte à definição de seus fundamentos, também chamados de **pilares do pensamento computacional**, que são:
1. A decomposição;
2. O reconhecimento de padrões;
3. A abstração;
4. Os algoritmos.
(SILVA et al , 2021)

A decomposição já foi apresentada e discutida neste capítulo, então vejamos um resumo dos demais pitares. (SILVA et al , 2021)

• **Reconhecimento de padrões**: após decompor um problema, é comum notarmos tarefas semelhantes ou idênticas. Tais similaridades auxiliam na reutilização de soluções com base em experiências anteriores.
• **Abstração**: eliminação de detalhes não relevantes para a solução do problema, auxiliando a dar foco nas informações realmente importantes.
• **Algoritmos**: conjunto de instruções claras e finitas necessárias para a solução do problema.

Esses pilares trabalham em **conjunto**, como engrenagens do pensamento computacional, sendo igualmente importantes e codependentes entre si. Todos participam, portanto, de forma sistemática na resolução de problemas. Embora não exista uma ordem explícita para sua aplicação, como exemplo, podemos seguir a **ordem abaixo**.

1. Para auxiliar na definição do problema e melhor elucidar possíveis soluções, podemos eliminar informações que não fazem parte do problema geral e que possam atrapalhar nossa atenção. Assim, após essa "limpeza", ficaremos apenas com os detalhes significativos, descartando as informações não relevantes para solução (**abstração**).

2. Com o problema definido, podemos fragmentá-lo em partes menores e gerenciáveis (**decomposição**). Cada subproblema, então, pode ser verificado com mais atenção, podendo ser novamente subdividido até a compreensão total do problema.

3. Uma vez conhecidos todos seus componentes, reduzidos a partes mais simples, também é mais fácil notar as similaridades entre algumas partes do problema e com outros problemas já resolvidos. Essa comparação ajuda a aproveitar a experiência obtida em problemas anteriores, agilizando a definição de possíveis soluções para os subproblemas em tela (**reconhecimento de padrões**).

4. Ao final, com o problema compreendido e a solução definida, é possível criar um plano de ação, uma sequência de ações práticas e sequenciais para executar a resolução de cada um dos subproblemas identificados(**algoritmos**).

Ainda que não seja elencado como um dos pilares, podemos considerar o raciocínio lógico como peça fundamental em todos os processos de resolução de problemas de desenvolvimento de software. Com raízes na Grécia antiga, a lógica e a computação têm laços

estreitos, tanto no "baixo nível", na representação de verdadeiro ou falso como bits de memória, quanto na construção dos algoritmos, na forma de instruções lógicas. O raciocínio lógico, assim, é o conhecimento que permeia e conecta todos os pilares. (SILVA et al , 2021)

Desenvolver o pensamento computacional e compreender como aplicar esses pilares nas práticas diárias não diz respeito, necessariamente, a saber programar, embora esse não seja um consenso entre os pesquisadores da área. Por exemplo, para Blikstein (2008), atualmente é necessário, para o ser humano, desenvolver competências para identificar e resolver problemas com a utilização do computador. Já para Liukas (2015), o pensamento computacional se baseia no raciocínio lógico e no reconhecimento de padrões, algoritmos, decomposição e abstração de problemas, de forma que as pessoas devem refletir sobre os problemas, mas permitir que computadores os resolvam. (SILVA et al , 2021)

Apesar dessa discussão sobre o uso ou não do computador, note que alguns pontos são comuns entre seus argumentos. É de comum acordo que pensamento computacional é uma habilidade natural que todos possuem, voltada para a definição e a resolução de problemas em diversas áreas de negócios e conhecimento. Porém, é fato que nem todos concordam sobre o uso de dispositivos computacionais para o desenvolvimento da solução, embora boa parte dos autores indique a necessidade da execução utilizando tecnologia. (SILVA et al , 2021)

Resumindo, as capacidades de raciocinar e de resolver problemas são inatas ao ser humano, mas pode ser aperfeiçoada pelo pensamento computacional, incrementando a capacidade intelectual pela estruturação de idealizações que podemos elaborar, compreender e elucidar complexos problemas desenvolvidas com algoritmos, de forma que tanto uma pessoa quanto um computador possam executá-los, independentemente da área a que se aplica. (SILVA et al , 2021)

3.12 RECONHECIMENTO DE PADRÕES

Os padrões são todas as características que tornam os problemas semelhantes entre si, ou seja, características idênticas que eles compartilham. Reconhecer os padrões facilita a forma como os problemas podem ser solucionados, já que, com isso, é possível construir uma base de soluções possíveis para cada um dos cenários identificados e conduzir um processo de forma mais simples. (ROSA et al , 2021)

É a habilidade de **olhar para o problema e tentar enxergar padrões**, ou seja, aspectos comuns, repetitivos e que possam facilitar a obtenção de uma solução mais rápida e assertiva.

Imaginando o seguinte cenário — no qual temos um quebra cabeça —, para facilitar a montagem podemos tentar definir dois padrões como: as cores das peças e as peças que formam o contorno do jogo, para que no momento em que começar o processo de montagem podemos iniciar pelas peças que já têm algum padrão. (NOLETO, BETRYBE, 2022)

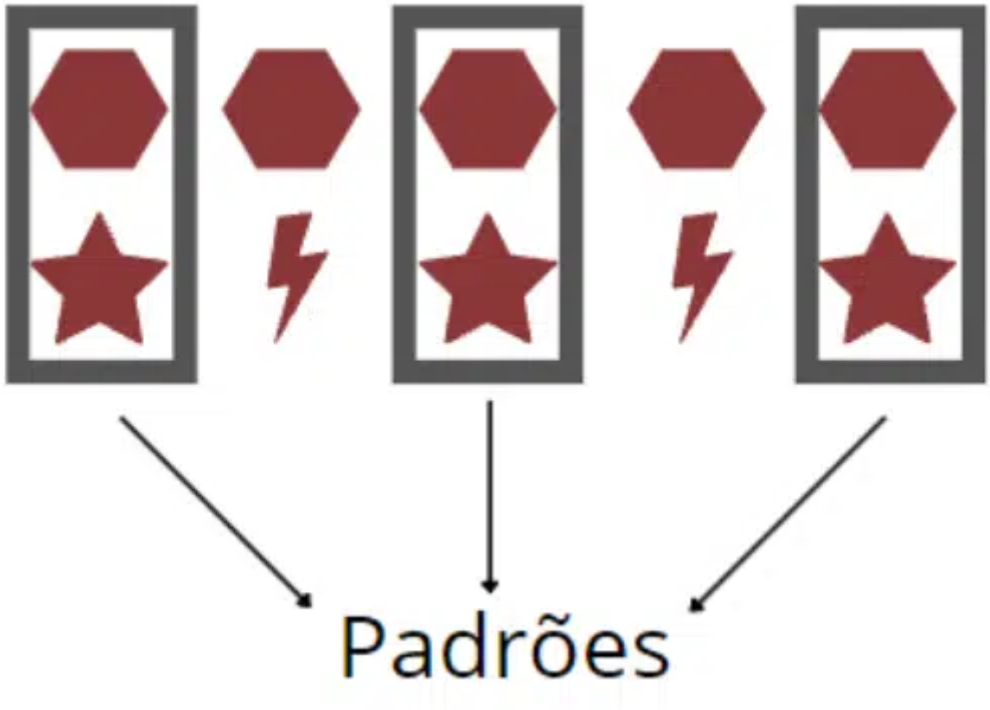

***Figura 4**: Reconhecimento de Padrões. - **Fonte**: BLOG.BETRYBE*

3.1

3.14 PENSAMENTO COMPUTACIONAL: ABSTRAÇÕES

A abstração é conceito essencial do pensamento computacional, capaz de criar representações simplificadas de qualquer ser vivo ou objeto. Quando em excesso, os detalhes sobre a entidade podem desviar o foco da solução ou trazer complexidade extra para o processamento. Modelos abstratos são muito utilizados em várias áreas do conhecimento. Por exemplo, um cientista cria modelos simplificados ignorando condições climáticas, atrito e resistência do ar, e mesmo que não reflitam completamente a realidade, são capazes de simular situações específicas e resolver grandes questões. (SILVA et al , 2021)

Abstrair é uma habilidade do ser humano, que consiste na capacidade de expressar algo de maneira concisa, eliminando detalhes desnecessários. Saber trabalhar com diferentes camadas

de abstração é um requisito para o profissional da computação e, ao mesmo tempo, um constante desafio. Diversos riscos e limitações a seu uso estão presentes, algumas vezes não sendo possível solucioná-los, apenas adaptar a abstração para absorvê-los. (SILVA et al , 2021)

3.15 ABSTRAÇÃO E PENSAMENTO COMPUTACIONAL

O pensamento computacional pode ser compreendido com uma estratégia para resolução de problemas utilizando conceitos computacionais (WING, 2006). A proposta é utilizar a forma de raciocinar de um cientista da computação, mas não atrelar a solução ao uso do computador. Até porque a maioria desses conceitos foram originados antes mesmo da criação do computador eletrônico, e sua aplicação transcende a ciência da computação (RILEY; HUNT, 2014). Além de estimular o pensamento lógico e sistemático, também promove o pensamento crítico, estratégico e criativo. (SILVA et al , 2021)

Como comentado anteriormente, o pensamento computacional faz uso de conceitos emprestados da computação, que dão suporte para a definição de seus fundamentos, também chamados de pilares do pensamento computacional: a decomposição, o reconhecimento de padrões, a abstração e os algoritmos (WING, 2006). Cada um deles trabalha em conjunto com os demais para atingir o objetivo. A **decomposição** reduz um problema em partes menores e mais simples de resolver, e a **abstração** simplifica essas partes, delas retirando os detalhes sem relevância. Já o **reconhecimento de padrões** nos auxilia a identificar similaridades entre algumas partes do problema ou comparar com problemas já resolvidos, auxiliando no reaproveitamento da experiência obtida em problemas anteriores. Ao final, temos compreensão do que deve ser feito e um planejamento das tarefas. Com base nesse planejamento, criamos listas ordenadas de ações sequenciais, os **algoritmos**, que executam o planejamento e encontram a solução buscada. A seguir, você vai conhecer melhor o conceito de abstração. (SILVA et al , 2021)

3.15.1 CONCEITO DE ABSTRAÇÃO

Segundo Wing (2006), a abstração é um conceito-chave tanto da computação quanto do pensamento computacional, chegando a descrever a ciência da computação como "**a automação da abstração**". Nesse contexto, o computador é considerado uma máquina que não pensa realmente, mas interpreta o mundo a sua volta, somente executando tarefas com base na **descrição** de mundo realizada pelo cientista da computação. (SILVA et al , 2021)

A tarefa de descrever algo nem sempre é fácil e depende muito do ponto de vista e das experiências de quem observa. Por exemplo, imagine um automóvel qualquer. Agora o descreva utilizando, no mínimo, 300 palavras. Talvez você tenha que incluir informações sobre o motor, o desempenho, o peso, etc., para completar o texto. Agora repita a tarefa, mas utilizando somente 50 palavras. Perceba que o nível de detalhamento será diferente nos dois textos, embora o veículo seja o mesmo. Agora reflita: qual das duas descrições é melhor? (SILVA et al , 2021)

Na verdade, a resposta vai **depender** da **finalidade**, de **quanta informação necessitamos** para **resolver** a **tarefa**. (SILVA et al , 2021)

Quando precisamos que o computador execute uma tarefa, devemos explicar somente as informações necessárias para que ele consiga executar, até porque, se não filtrarmos (e dependendo do nível de detalhes), podemos confundi-lo com o excesso de informações e desviar do objetivo principal. (SILVA et al , 2021)

De maneira semelhante, o cientista cria modelos do mundo real, modelos que contêm informação suficiente para não desviar a atenção do que está analisando, e então utiliza esse modelo para definir soluções para os problemas por intermédio deles. Com esses modelos reduzidos, sem informações que desviem a atenção, quando se consegue compreender suficientemente o problema, é possível instruir o computador com algoritmos, **ensinando** a solução encontrada. (SILVA et al , 2021)

Esse processo de selecionar quais informações são mais ou menos relevantes é o que chamamos de **abstração**. Segundo Guttag (2013), a essência da abstração é manter as informações relevantes de determinado contexto e omitir todo o resto. Lembrando que esse "**relevante**" sempre vai **depender** da **finalidade da tarefa**. (SILVA et al , 2021)

O pensamento abstrato é uma habilidade cognitiva importante do ser humano e que aplicamos a todo momento, sem perceber. Criamos **abstrações**, **ou simplificações**, para tudo que está a nosso redor, desde conceitos simples até artefatos complexos. Por exemplo, pense na palavra "número": só de ouvir esse termo você já sabe do que se trata, mesmo sem ouvir outros detalhes. Os números são abstrações para representar a quantificação de algo. (SILVA et al , 2021)

Observe um exemplo mais robusto, a criação de um mapa (BEECHER,2017). Ele é uma representação simplificada da realidade. Nem todas as informações necessitam estar lá e, dependendo de sua finalidade, também não é necessário que seja totalmente fiel à realidade. Mas qual é a utilidade de algo se não reproduz a realidade? Vejamos um exemplo: observe o mapa que representa a linha férrea de uma região do Sul do Brasil (Figura 5) . (SILVA et al , 2021)

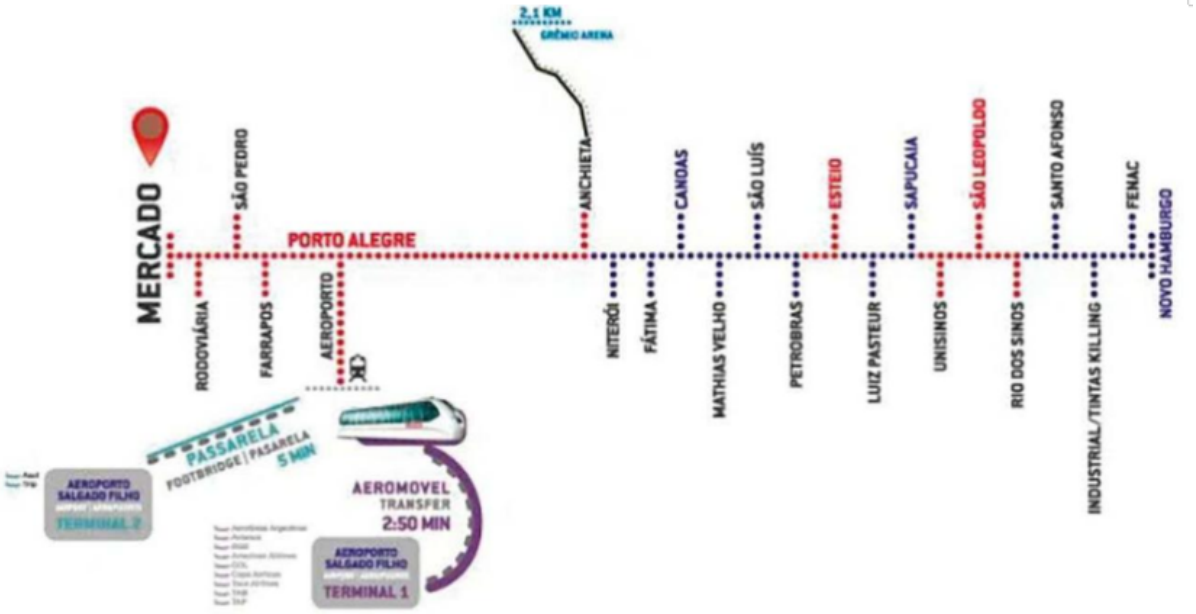

***Figura 5**: Representação do mapa metroviário. - **Fonte**: Adaptada de Beecher (2017).*

Note que ela não representa a região de forma realista, começando pela disposição das paradas ao longo da linha, que não são equidistantes entre si. Considerando que o **objetivo** desse **mapa** é **apresentar** a **ordem** das estações, a omissão de detalhes acaba por simplificar sua utilização pelos passageiros. Portanto, um mapa é um exemplo bem comum de abstração, e **dependendo** de sua **finalidade**, podemos ter **diferentes versões** dele. Veja a Figura 6. (SILVA et al , 2021)

***Figura 6**: Outras representações do mesmo mapa metroviário com diferentes níveis de abstração. - **Fonte**: Adaptada de beecher (2017).*

Essa diferença no nível de detalhamento das abstrações está ligada, como já comentado, a sua finalidade. Aumentar ou reduzir o nível de detalhamento de um elemento é o que chamamos de camadas de abstração, assunto da próxima seção. (SILVA et al , 2021)

3.15.2 ABSTRAÇÕES EM CAMADAS

Como visto anteriormente, a **abstração** é a capacidade de **identificar** os **detalhes importantes** do que se está observando, mas a **quantidade** desses "detalhes" que atribuímos ao elemento **depende do objetivo** em que será empregado. O volume de informações vai impactar **significativamente** a representação. Também vimos que é possível criar diferentes versões de representação para o mesmo elemento, com níveis diferentes de detalhamento. Note, na Figura 7, uma sequência de estudos do pintor Pablo Picasso, com diversas representações do mesmo elemento (o touro), somente diferindo o nível de detalhes. (SILVA et al , 2021)

Figura 7*:Compilado de diversos estudos do artista Pablo Picasso, apresentando diferentes níveis de abstração sobre o mesmo elemento. Os quadros estão expostos no Norton Simonuseum.* ***Fonte****: Adaptada de Norton Simon Museum (c2020).*

Essa diferença entre os níveis de abstração é conhecida como **camadas de abstração**. O conceito é simples e podemos utilizar outro exemplo do cotidiano. Dessa vez, vamos observar a utilização de areia como material de construção. Em diversos momentos, o material é utilizado e, em cada um, necessita de uma granularidade (mais "fina" ou mais "grossa") diferente. Mas o material à disposição é muito granular. Podemos, então, utilizar peneiras para obter o material na granularidade correta. Para estruturas e vigas, não precisamos nos preocupar tanto: utilizamos uma peneira com furos maiores somente para retirar pedras e outros tipos de sujeira. Para paredes, piso e outros processos, precisamos de areia com média granularidade, pois afeta o alinhamento da estrutura. Já para o acabamento, precisamos de material bastante refinado, então nosso refinamento deve ser o maior possível, deixando passar somente os menores grãos. (SILVA et al , 2021)

Por mais estranho que possa parecer, nosso exemplo demonstra o uso de camadas (peneira), em que cada uma delas só disponibiliza a granularidade **adequada** para **cada**

contexto. Cada camada representa uma quantidade diferente de informações e, cada vez que a trocamos, podemos **aumentar** os **detalhes** (**reduzindo a abstração**) **ou adicionar** mais **camadas** de abstração para **suprimir** detalhes (**aumentando a abstração**). (SILVA et al , 2021)

* **Observação**: Segundo Abowd e Dey (1999), contexto é qualquer informação que pode ser utilizada para identificar a situação de uma entidade (seja uma pessoa, lugar ou objteto considerada relevante para a interaçao entre um usuário e uma aplicação. Ou seja, para uma pessoa, o contexto pode ser o lugar onde ela se encontra (em casa, no shopping, na faculdade), o grupo social (amigos, colegas de faculdade, parentes), etc. Informações contextuais são muito utilizadas em várias áreas da computação, como na computação ubíqua, cujo objetivo é oferecer serviços sem mesmo percebermos, baseado em perfis de comportamento criados com base nas informações contextuais do usuário. (SILVA et al , 2021)

O número de camadas ou níveis de abstração para a representação de uma entidade pode ser infinita, e a definição de qual camada é mais apropriada **depende** do **contexto** em que será empregada; ou seja, no caso da aplicação de abstrações, a escolha da camada ideal deve levar em conta o **contexto**. Como exemplo, podemos instruir um aplicativo a emitir alertas sobre promoções para produtos que o usuário tenha pesquisado recentemente (contexto do usuário) ou aumentar o nível de abstração e realizar as recomendações com base nas pesquisas realizadas por usuários da mesma faixa etária ou da mesma região (contexto de idade ou contexto regional). (SILVA et al , 2021)

3.15.3 RISCOS E LIMITAÇÕES DO USO DE ABSTRAÇÕES

Como visto nas seções anteriores, a abstração, além de função cognitiva inata do ser humano, é um dos pilares do pensamento computacional, permitindo que possamos **representar** elementos do **mundo real** de forma **simplificada** e, assim, utilizá-los na **solução de problemas**. Também foi apresentado como uma abstração da realidade pode ser representada de múltiplas formas, dependendo da quantidade de camadas utilizadas em sua construção. (SILVA et al , 2021)

O fato é que abstrações são conceitos muito ligados à **experiência** e à **perspectiva** do **observador**. Como comentado anteriormente, **cada** analista descreve uma abstração levando em conta as particularidades importantes para ele. Outro exercício de abstração: como você descreveria um gato? (SILVA et al , 2021)

O exercício serve para recordar que não existe uma definição mais correta, mas a mais **adequada** para determinado cenário. Note, na **Figura 8**, esse exemplo da representação do gato, em que são apresentados dois pontos de vista: para a senhora idosa, o gato é uma "bola de pelos" graciosa, enquanto, do ponto de vista da cirurgiã veterinária, o gato é representado por sua anatomia, esqueleto e órgãos internos. Cada uma das representações é adequada na perspectiva das observadoras. (SILVA et al , 2021)

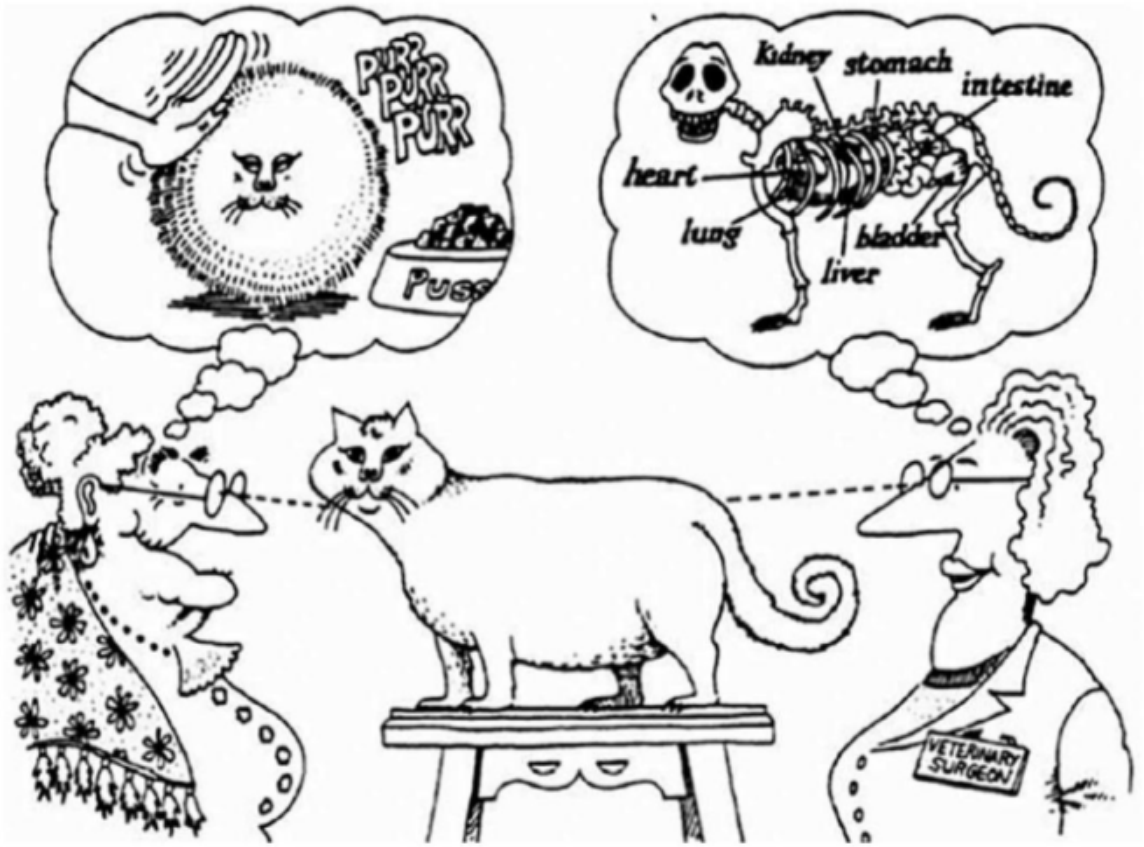

***Figura 8**: Experiência do analista e diferentes perspectivas podem influenciar o resultado da abstração. - **Fonte**: Adaptada de Booch (2004).*

Abstrações são **comuns** em **algoritmos**. Linguagens que utilizam **orientação a objetos** a utilizam para a construção de uma classe, por exemplo, que é a abstração de um elemento do mundo real, e uma invocação de método, por sua vez, é a abstração de algum comportamento concreto. (SILVA et al , 2021)

As abstrações são essenciais para o pensamento computacional, mas é necessário atenção para que elas possam ser realmente úteis. A definição de uma abstração é uma tarefa complexa, de forma que o risco de erros está sempre presente. Um dos problemas mais comuns é o **vazamento de abstração**, que ocorre quando uma entidade vaza detalhes de sua construção, como propriedades ou comportamento específico de implementação (VAN DEURSEN; SEEMANN 2019).

Para deixar mais claro o que é o vazamento, vamos voltar ao exemplo dos carros. Como comentamos na seção anterior, um carro possui várias abstrações, que fornecem uma interface amigável para o motorista. Pensando em sua composição (motor, eixos, sistema elétrico, sistema de frenagem, entre outros), seria complexo conhecer o funcionamento de cada uma das partes para conduzi-lo, mas, ao contrário, o carro oculta todos esses detalhes atrás interfaces mais simples, como o volante, os pedais e a alavanca de câmbio. Você não precisa saber o que ocorre internamente ao interagir com as interfaces para conduzir o

veículo, e é justamente nesse ponto que vamos encontrar possíveis vazamentos. Considerando que todo motorista aprende que o pedal de embreagem deve ser pressionado somente para a troca da marcha, se não assim o fizer, haverá desgaste no sistema de freio. Essa informação sobre o funcionamento do sistema de freios nos fez descobrir um detalhe da implementação do pedal de embreagem, como é sua **função** e como ele afeta **outros componentes** da abstração. (SILVA et al , 2021)

Quando esses detalhes de implementação ficam evidentes, temos o chamado **vazamento**. Podemos continuar encontrando vazamentos no carro, como no câmbio manual, por exemplo: o motorista é obrigado a conhecer o mecanismo para conduzir, sendo que o carro deveria trocar as marchas sozinho. Nesse exemplo, o câmbio automático seria uma abstração sem o vazamento citado, oferecendo maior conforto e comodidade ao motorista. Embora o uso de uma abstração mais "fechada" traga alguns benefícios, o uso deve ser sempre avaliado. Por exemplo, alguns especialistas (SILVA, 2019) afirmam que o câmbio automático afeta o desempenho e a eficiência (consumo de combustível, por exemplo) devido a sua estrutura mais complexa e porque suas trocas de marchas ocorrem com rotações mais altas. A abstração mais adequada vai **depender da aplicação**, ou seja, de qual característica vai **atender melhor** o **usuário**: **desempenho** ou **comodidade**. (SILVA et al , 2021)

Na **computação**, podemos citar vários exemplos, como os frameworks, uma abstração com várias interfaces para a construção de sistemas. Esses pacotes auxiliam a construção, mas não resolvem todos os problemas, deixando brechas para o desenvolvedor sobrescrever suas funcionalidades: outro vazamento. No caso da função "Garbage collector" da linguagem Java, cujo papel é eliminar instâncias descartadas da memória do computador, é permitido realizar o gerenciamento manual de memória: mais um vazamento. (SILVA et al , 2021)

Quando você está trabalhando com um número em representação textual e precisa convertê-lo para formato número, vazamento novamente (VAN DEURSEN; SEEMANN, 2019). Lembrando que não **necessariamente** tais vazamentos sejam negativos, pois permitir que um desenvolvedor experiente customize funcionalidades ou trate manualmente o gerenciamento de memória pode trazer incrementos importantes no desempenho ou no consumo do produto. (SILVA et al , 2021)

Quando vazamentos ocorrem, detalhes que deveriam ser **ocultos** tornam-se indispensáveis para seu funcionamento. Dificilmente uma abstração de alta complexidade será à prova de vazamentos, e raramente será possível prever o fato. Devemos, assim, **aprender a tratá-los**. **Na medida do possível**, o modelo deve ser corrigido para considerar o problema e os detalhes que se revelaram importantes, que devem ser acrescentados ao modelo de forma explícita. (SILVA et al , 2021)

Por exemplo, no caso antes citado sobre o sistema de freios, o motorista acaba aprendendo detalhes da implementação para saber o momento e a forma certa de executar a frenagem. O freio ABS (anti-lock braking system) e seu sistema de controle de frenagem retira um pouco dessa responsabilidade do motorista, sendo uma forma de reduzir o vazamento de abstração sobre o sistema de frenagem. (SILVA et al , 2021)

Por outro lado, tentar evitar todos os vazamentos pode levar a outros problemas de implementação. Para que a abstração seja utilizada, é necessário transformar em algo concreto, ou seja, criar uma instância com detalhes concretos. Anteriormente, comentamos sobre orientação a objetos e como uma classe representa a abstração de um elemento; porém, para utilizá-la, necessitamos instanciar a classe, transformá-la em um objeto com informações concretas (valores para os atributos, por exemplo) para que ela possa executar sua função. Tentar evitar todos os vazamentos pode causar falhas no detalhamento e aumentar a complexidade da codificação da abstração ou até mesmo seu formato de utilização. (SILVA et al , 2021)

Esses são riscos intrínsecos (internos) ao processo de abstração. Assim, deve-se **identificar** a quantidade correta de detalhamento para que o modelo fique **equilibrado**, sem detalhes em excesso, de forma que o modelo se torne mais complexo que o necessário, nem detalhes de menos, de modo que não tenha os requisitos mínimos para executar sua tarefa. Também devemos evitar os vazamentos de abstração, mas sem aumentar sua complexidade sem razão. (SILVA et al , 2021)

Tudo vai depender do contexto para o qual a abstração é necessária, talvez altamente abstrato para alguns casos e flexível em outros. (SILVA et al , 2021)

Abstrações podem simplificar as tarefas do desenvolvedor, facilitando seu trabalho e aumentando sua produtividade. Mas abstrações tendem a ter um impacto negativo no desempenho da aplicação. **É muito difícil generalizar e ser eficiente**. A criação de abstrações de software está longe de ser perfeita, mas, sem ela, com certeza seria mais complexo construirmos soluções de software computacionalmente. (SILVA et al , 2021)

4 CONCLUSÃO E TRABALHOS FUTUROS

4.1.1 CONCLUSÃO

A partir do processo investigativo sobre o **Pensamento Computacional** e seus pilares fundamentais para a aplicação dá técnica, pudemos ver que todo problema recebe um nível e ou grau de complexidade diferente, tanto pela percepção baseada na experiência do "elaborar da solução" ou em seu **nível** de abstração **necessário**, tanto quanto pela **especificidade** do problema em questão, Concluímos que não é necessária uma ordem de execução dos pilares para que possamos obter resultados, porém sim se quisermos obter **melhores** resultados, Também observamos os processos de **reconhecer padrões** e que se **reutilizarmos** soluções já implementadas e que são de mesmo valor para problemas diferentes, pode-se ser muito útil para poupar tempo e não necessitar pensar em todo o processo novamente para um problema idêntico ou similar, assim poderemos **economizar** tempo, Também foi visto que se dividirmos o

problema em seções e se necessário em subseções de seções, conseguiremos solucioná-lo de forma mais simples e até mais eficiente, pois teremos menos atribuições de uma única vez, assim ficando mais fácil a resolução e também podendo designar outras seções do problema para outros membros, e ao final bastando juntar as partes divididas e solucionadas como um todo, então podemos ter na conclusão total na íntegra sobre o pensamento computacional, que é uma excelente técnica para solucionar problemas, porém ficando atrelada a experiência do usuário e em sua percepção que em teoria deve-se se basear em **pensamentos lógicos**, **padrões**, **fatos**, **necessidades** e um pouco de **senso comum**.

4.1.2 PROPOSTAS DE TRABALHOS FUTUROS

Para futuras aplicações desta pesquisa sobre a técnica de algoritmia conhecida Pensamento Computacional, me proponho a analisar os padrões de aplicação da técnica e dentre a sua “arquitetura” que foi criada para facilitar a resolução de problemas complexos, observar e tentar encontrar dificuldades e déficits na aplicação da mesma, e caso necessário elaborar uma aprimoração na técnica, tornando-a mais fácil, performática e eficiente do que já se propõe a ser.

5 REFERÊNCIAS BIBLIOGRÁFICAS

BERNARDES, Luana. Divisão da História. Todo Estudo. Disponível em: https://www.todoestudo.com.br/historia/divisao-da-historia. Acesso em: 12 de October de 2023.

NOLETO, Caio. Pensamento Computacional: Tudo Sobre. Trybe Blog, 2022. Disponível em: https://blog.betrybe.com/tecnologia/pensamento-computacional-tudo-sobre/#1. Acesso em: 18 de outubro de 2023.

SILVA, Marcelo. *Pensamento Computacional*. Editora: SAGAH EDUCAÇÃO S.A., 2021.

Mundo Educação. Idade Média. Mundo Educação. Disponível em: https://mundoeducacao.uol.com.br/historiageral/idade-media.htm. Acesso em: 20 de outubro de 2023.

Mundo Educação. Idade Moderna. Mundo Educação. Disponível em: https://mundoeducacao.uol.com.br/historiageral/idade-moderna.htm. Acesso em: 21 de outubro de 2023.

Brasil Escola. O que é Idade Contemporânea. Brasil Escola. Disponível em: https://brasilescola.uol.com.br/o-que-e/historia/o-que-e-idade-contemporanea.htm. Acesso em: 22 de outubro de 2023.

BEECHER, K. Computational thinking a beginner's guide to problem-solving and programming. Swindon: BCS, 2017.

FIGUEIREDO, C. M. H.; LAMB, L. C. Teoria da Computação: uma Introdução à Complexidade e à Lógica Computacional. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 34., 2015, Porto Alegre. Anais Porto Alegre: Sociedade Brasileira de Computação, 2015, p. 10-67. Disponível em:

https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/6/6/19-1?inline=1. Acesso em: 18 dez. 2020.

MOKARZEL, F.; SOMA, N. Introdução à ciência da computação. Rio de Janeiro: Elsevier,2008.429 p.

NORONHA, C. Lógica e Algoritmos de Programação. Medium, San Francisco, 13 fev. 2019. Disponível em: https://medium.com/@caio.cnoronha/l%C3%B3gica-e-algoritmos-de-Acesso em: 18 dez. 2020.

PAPERT, S. A. Mindstorms: children, computers, and powerful ideas. New York: Basic Books, 1980.

WING, J. M. Computational thinking. Communications of the ACM, v. 49, n° 3, p. 33-35, 2006. Disponível em: http://www.cs.cmu.edu/-15110-s13/Wing06-ct.pdf. Acesso em: 22 dez. 2020.

BRACKMANN, C. H. Desenvolvimento do Pensamento Computacional através de atividades desplugadas na computação básica. 2017.266 f. Tese (Mestrado em Informática na Educação) — Programa de Pós-Graduação em Informática na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017. Disponível em: https://docplayer.com.br/73392937-Universidade-federal-do-rio-grande-do-sul-centro-interdisciplinar-de-novas-tecnologias-na-educacao.html. Acesso em: 29 nov. 2020.

DENNING, P. J.; TEDRE, M. Computational thinking. Cambridge, MA: MIT Press, 2019.

EDUCAÇÃO Básica infantil e fundamental. CIEB, [2020]. Disponível em: https://curriculo. cieb.net.br/. Acesso em: 9 dez. 2020.

SCHOLOGL, L. E. et al. Ensino do Pensamento Computacional na educação básica. Revista de Sistemas e Computação, v. 7, n. 2, p. 304-322, 2017. Disponível em: https://revistas.unifacs.br/index.php/rsc/articIe/view/5106. Acesso em: 29 nov. 2020.

BLIKSTEIN, P. O pensamento computacional e a reinvenção do computador na educação. 2008. Disponível em: http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html. Acesso em: 22 dez. 2020.

CHERRY, K. Heuristics and cognitive biases. Verywell Mind, 2020. Disponível em: https:// www.verywelImind.com/what-is-a-heuristic-2795235.Acesso em: 22 dez. 2020.

FERRAGINA, P.; LUCCIO, F. Computational thinking: first algorithms, then code. Berlin: Springer, 2018.

LIUKAS, L. Hello Ruby: adventures in coding. New York: Feiwel & Friends, 2015.

NUNES, D. J. Ciência da computação na educação básica. Jornal da Ciência, v. 9, n? 9, 2011.

PIVA JR., D. et al. Algoritmos e programação de computadores. Rio de Janeiro: Elsevier, 2012.

POLYA, G. A arte de resolver problemas: um novo aspecto do método matemático. Rio de Janeiro: Interciência, 1977.

POZO, Juan Ignacio. A solução de problemas: aprender a resolver, resolver para aprender. Porto Alegre: Artmed, 1998.

RIBEIRO, L.; FOSS, L.; CAVALHEIRO, S. A. da C. Entendendo o pensamento computacional. 2017. Disponível em: https://arxiv.org/pdf/1707.00338.pdf. Acesso em: 22 dez. 2020.

RILEY, D. D.; HUNT, K. A. Computational thinking for the modern problem solver. Boca Raton: CRC press, 2014.

SNEIDER, C. et al. Exploring the science framework and NGSS: computational thinking in the science classroom. Science Teacher, v. 81, n° 5, p. 10-15, 2014.