PROGRAMA EDUCACIONAL EM SAÚDE DIGITAL DA UNIVERSIDADE FEDERAL DE GOIÁS

EDUCAÇÃO E CAPACITAÇÃO DE RECURSOS HUMANOS EM SAÚDE DIGITAL

Pensamento computacional

Organizadores

Ana Laura de Sene Amâncio Zara
Fábio Nogueira de Lucena
Rejane Faria Ribeiro-Rotta
Renata Dutra Braga
Rita Goreti Amaral
Sheila Mara Pedrosa
Silvana de Lima Vieira dos Santos
Taciana Novo Kudo

Cegraf UFG

Comissão de Governança da Informação em Saúde (CGIS)

Silvana de Lima Vieira dos Santos

Centro de Inovação em Gestão da Educação e do Trabalho em Saúde (CIGETS) e Laboratório de Pesquisa em Empreendedorismo e Inovação (LAPEI)

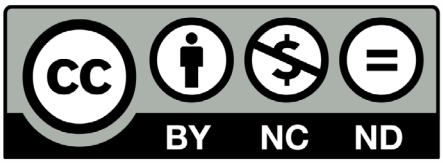
Cândido Vieira Borges Júnior

Laboratório de Inovação e Estratégia em Governo (LineGov)

Antônio Isidro da Silva Filho

Ministério da Saúde / Secretaria Executiva / Departamento de Informática do Sistema Único de Saúde (DATASUS)

Jacson Venancio de Barros



Pensamento computacional

Ana Laura de Sene Amâncio Zara
Fábio Nogueira de Lucena
Rejane Faria Ribeiro-Rotta
Renata Dutra Braga
Rita Goreti Amaral
Sheila Mara Pedrosa
Silvana de Lima Vieira dos Santos
Taciana Novo Kudo
(Organizadores)

Cegraf UFG

2021

Revisão editorial
Ana Laura Sene Amâncio Zara

Revisão técnica
Ana Claudia Sayeg Freire Murahovschi (Ministério da Saúde)
Ana Paula de Andrade Pannuti (Ministério da Saúde)
Andréia Cristina de Souza Santos (Ministério da Saúde)
Gabriella Nunes Neves (Ministério da Saúde)
Josélio Emar de Araújo Queiroz (Ministério da Saúde)
Juliana Pereira de Souza Zinader (Ministério da Saúde)
Mara Lucia dos Santos Costa (Ministério da Saúde)
Marcia Elizabeth Marinho da Silva (Ministério da Saúde)
Maria Cristina Ferreira de Abreu (Ministério da Saúde)
Patricia dos Santos Irigaray Rodrigues (Ministério da Saúde)
Robson Willian de Melo Matos (Ministério da Saúde)
Thais Lucena de Oliveira (Ministério da Saúde)

Capa
Iuri Vaz Miranda - graduando (UFG)

Editoração Eletrônica
Luma Wanderley de Oliveira - doutoranda (UFG)
Virgínia de Fernandes Souza - graduanda (UFG)

https://doi.org/10.5216/PEN.ebook.978-85-495-0363-3/2021

Dados Internacionais de Catalogação na Publicação (CIP)
GPT/BC/UFG

P418 Pensamento computacional [E-book] / organizadores, Ana Laura de Sene Amâncio Zara ... [et al.]. - Goiânia : Cegraf UFG, 2021.
52 p. : il.

Inclui referências.
ISBN (E-book): 978-85-495-0363-3

1. Sistemas de computação. 2. Inteligência artificial - Saúde digital. 3. Programação de sistemas. 4. Engenharia de software. 5. Saúde - Estudo e ensino. I. Zara, Ana Laura de Sene Amâncio.

CDU: 614.39:004

Bibliotecária responsável: Adriana Pereira de Aguiar / CRB1: 3172

Pensamento computacional

Instituição responsável

Universidade Federal de Goiás (UFG)

Comissão de Governança da Informação em Saúde da UFG (CGIS-UFG)

Centro de Inovação em Gestão da Educação e do Trabalho em Saúde (CIGETS)

Laboratório de Pesquisa em Empreendedorismo e Inovação da Universidade Federal de Goiás (LAPEI-UFG)

Instituição financiadora

Ministério da Saúde (MS)

Secretaria Executiva (SE)

Departamento de Informática do Sistema Único de Saúde (DATASUS)

Secretaria de Gestão do Trabalho e da Educação na Saúde (SGTES)

Apoio

Ministério da Saúde (MS):

Secretaria de Atenção Primária à Saúde (SAPS)

Demais parceiros

Laboratório de Inovação e Estratégia em Governo (LineGov)

Abreviaturas e Siglas

3G	Associação Brasileira de Normas Técnicas
4G	*Australian Digital Health Agency* - Agência Australiana de Saúde Digital
5G	Agência para o Desenvolvimento do Governo Eletrônico e da Sociedade da Informação e Conhecimento
CBIS	Assessoria Internacional de Saúde
CGIS	Base Nacional da Assistência Farmacêutica
CHM	Base Nacional de Notificações
CIGETS	Centro de Complexidade em Oncologia
DATASUS	Coordenação-Geral de Informação da Atenção Primária
ENIAC	Comissão de Governança da Informação em Saúde
ESD28	Estratégia de Saúde Digital para o Brasil 2020-2028
g	Gramas
GB	*Gigabyte*
GHz	*Gigahertz*
IaaS	*Infrastructure as a Service* - Infraestrutura como Serviço
IBM	*International Business Machines Corporation*
IMC	*Índice de Massa Corporal*
IoT	*Internet of Things* - Internet das Coisas
KB	*Kilobyte*
kg	Quilogramas
LAPEI	Laboratório de Pesquisa em Empreendedorismo e Inovação
LineGov	Laboratório de Inovação e Estratégia em Governo
MHz	*Megahertz*
PC	*Personal Computer* - Computador Pessoal
RAM	*Random Access Memory - Memória de Acesso Aleatório*
RNDS	*Rede Nacional de Dados em Saúde*
SaaS	*Software as a Service - Software como Serviço*
SAPS	*Secretaria de Atenção Primária à Saúde*
SBIS	Sociedade Brasileira de Informática em Saúde
SGTES	Secretaria de Gestão do Trabalho e da Educação na Saúde
SUS	Sistema Único de Saúde

SWEBOK	*Software Engineering Body of Knowledge* - Conjunto de Conhecimentos de Engenharia de *Software*
TFLOP	*Teraflop*
TIC	Tecnologia da Informação e Comunicação
UCP	Unidade Central de Processamento
UFG	Universidade Federal de Goiás
UNA-SUS	Universidade Aberta do Sistema Único de Saúde
UnB	Universidade de Brasília
Wi-Fi	*Wireless Fidelity* - Fidelidade sem Fio

Lista de Figuras e Vídeos

Sumário

Apresentação

Prezado(a) Participante,

Seja bem-vinda(o) ao Microcurso **Pensamento Computacional**!

A implementação da Saúde Digital no Brasil inclui um conjunto de ações de Tecnologia da Informação e Comunicação (TIC). A efetividade dessas ações depende da capacitação de profissionais da saúde e de gestores de saúde, o que motiva a criação do presente Microcurso.

Este documento contém um conjunto simplificado, mas, ao mesmo tempo, coerente e coeso de informações, além de referências relevantes para a compreensão do pensamento computacional.

O que é pensamento computacional? Todo o conteúdo deste documento vai contribuir com essa definição, mas é possível oferecer uma orientação preliminar.

Pensamento Computacional é apresentado como ferramenta para explicar/entender como os computadores funcionam e a tecnologia disponível tanto para o projeto (design) de software quanto para a solução de problemas. Isso inclui a compreensão dos limites da computação.

Dito de outra forma:

- como os computadores funcionam (mecanismo e funcionamento; como são controlados por algoritmos, como podemos expressar algoritmos em uma linguagem de programação, como podemos combinar vários módulos de software em um sistema funcionando);
- o que pode ser projetado/desenvolvido (sensibilidade para entender o contexto no qual usuários atuam; noção da tecnologia disponível e de projeto/desenvolvimento de software para realizar um trabalho ou resolver um problema).

Bom proveito!

EDUCAÇÃO E CAPACITAÇÃO
DE RECURSOS HUMANOS
EM SAÚDE DIGITAL

Pensamento computacional

Unidade 1
Como funcionam os sistemas computacionais

Fábio Nogueira de Lucena
Sérgio Teixeira de Carvalho

Unidade 1: Como Funcionam os Sistemas Computacionais

A computação está presente em quase tudo no nosso cotidiano, seja em casa, no trabalho ou mesmo quando nos divertimos. Essa presença maciça da computação no nosso dia a dia tem consequências, o que faz surgir várias questões:

O que precisamos entender sobre computadores?
Qual o papel dos algoritmos e da programação?
O que devemos fazer para que um computador trabalhe em nosso benefício?
Em que os computadores não são bons?

O pensamento computacional reúne um corpo de conhecimento que contribui com respostas para essas perguntas.

Nesta Unidade, vamos estudar os elementos da computação, os seus mecanismos e o seu funcionamento. Com esse conhecimento, teremos os insumos suficientes para compreender como o pensamento computacional, eventualmente, podendo ser empregado em outros domínios, ajudando-nos a projetar soluções para os problemas do nosso cotidiano.

Adicionalmente, mesmo que a computação não seja a nossa área de conhecimento, ela permite que estejamos mais preparados para colaborar com a concepção e evolução de sistemas computacionais, o que parece inevitável, no momento, em todo e qualquer setor da sociedade.

1.1 Elementos Básicos da Computação

1.1.1 Computação Presente no Dia a Dia

No nosso dia a dia, contamos com a presença maciça de dispositivos computacionais, seja para a execução de tarefas em casa, no trabalho, ou entretenimento, eles estão ao nosso redor e por toda parte. Até pouco tempo, os mais comuns eram os computadores de mesa (chamados de *desktops*), atualmente, o emprego de portáteis (chamados de *notebooks* ou *laptops*), além dos *tablets* e *smartphones* que têm se expandido de forma significativa. O mercado dos portáteis, em todo o mundo, está em torno de 57%[1], enquanto os computadores de mesa ficam com 43% [1].

Esses dispositivos são considerados dispositivos computacionais, pois têm a capacidade de receber e processar dados e entregar algum resultado. Por exemplo, podemos usar nosso smartphone para reservar um quarto em um hotel, realizar algum serviço bancário, ou mesmo para selecionar e assistir a um filme. Em outras palavras, esses dispositivos são computadores. Além desses, podemos citar outros exemplos, como *smartwatches* e *smart TVs*.

1 Dados atualizados em abril de 2021.

Há ainda dispositivos computacionais com propósito mais específico. Na área de saúde, por exemplo, há os de pressão arterial, de glicemia, de frequência cardíaca e medidores de temperatura corporal -- muitas vezes denominados como medidores, máquinas de raio-x e máquinas de ultrassom, dentre outros.

Na área de segurança, há sensores de presença para alarmes de segurança patrimonial, câmeras de segurança, equipamentos de controle de acesso a locais, como prédios residenciais ou empresariais (digitais, senhas, reconhecimento facial). Há também eletrodomésticos, como geladeiras, fornos micro-ondas e máquinas de lavar que têm recebido, em seus projetos, capacidades computacionais. Há também os automóveis, repletos de *hardware* e *software*, seja para controlar o *air-bag*, a temperatura interna e, em alguns casos, até a direção, dispensando a direção humana.

Quando conectados à *internet*, sensores e outros dispositivos portáteis formam o que se denomina Internet das Coisas (IdC) – em inglês, IoT (*Internet of Things*) .

1.1.2 *Hardware / Software*

Neste amplo universo de dispositivos/aparelhos/equipamentos, tem-se o *hardware*, o componente físico, e o *software* que oferece os serviços que realizamos com eles -- comércio eletrônico, serviços bancários, serviços de transporte, reservas em hotel, segurança, saúde, cursos, pesquisas, livraria, aluguel de carro -- e que quase tudo isso é apoiado por computadores "invisíveis", em grande número, espalhados por todo o planeta, em uma rede chamada "nuvem", também conhecida pelo termo em inglês *cloud*.

Quer ouvir a pronúncia de **cloud**?
https://forvo.com/word/cloud/

A computação percebida no nosso cotidiano, portanto, acontece pela combinação do *hardware* (dispositivos/aparelhos/equipamentos), do *software* (aplicativos, programas instalados) e da nuvem (*cloud*).

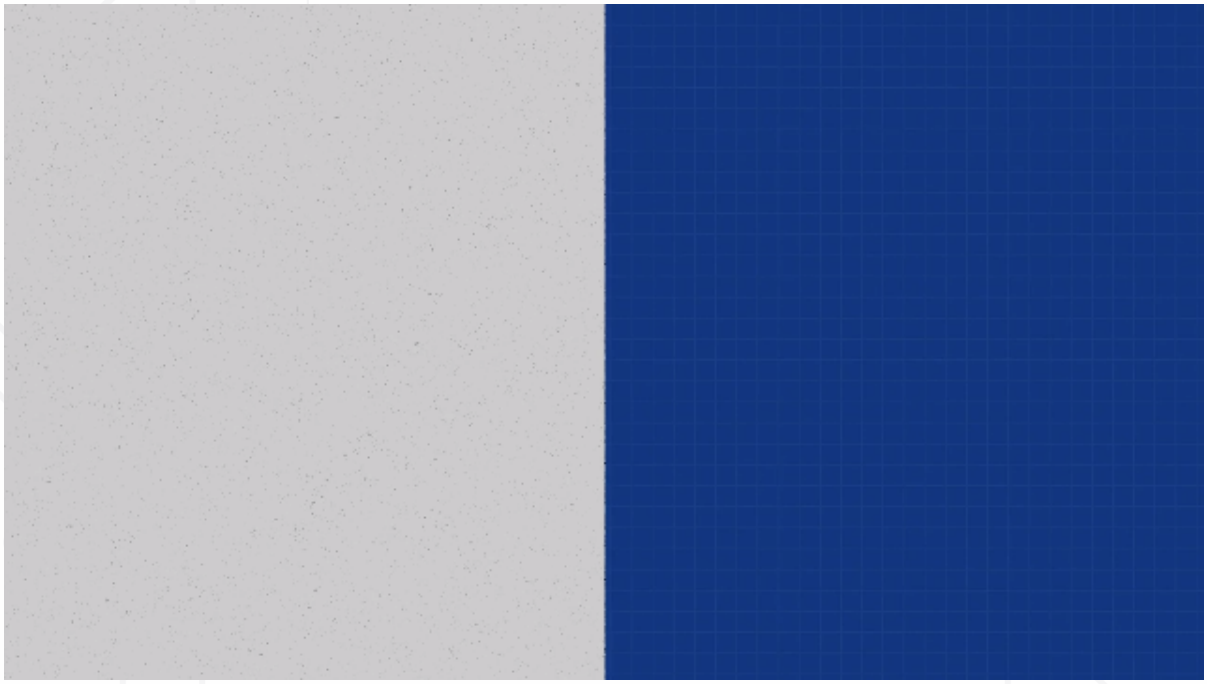
O *hardware* que carregamos nos nossos bolsos ou usamos em nosso ambiente de trabalho, ou em casa, não é útil se não houver o *software*, ou seja, um programa de computador previamente projetado para ser instalado/embarcado no *hardware*. Um *desktop*, por exemplo, usado em um escritório, não seria muito útil sem o *software* para editar textos, planilhas ou apresentações (Microsoft® Office Word®, Excel®, PowerPoint®, LibreOffice®, etc.), ou sem um navegador *web* (Google® Chrome®, Firefox®, etc.) Os *smartphones* não são diferentes. *Smartwatches* também devem ter um *software* neles embarcado para que apresentem os dados na sua tela/interface.

Dispositivos de saúde vestíveis, já citados, como, por exemplo, um de pressão arterial adquirido em uma farmácia, precisa de um *software* nele embarcado para receber, processar e apresentar a medida coletada, bem como as bombas de infusão de insulina, os medidores de glicemia ou os para a promoção de atividade física.. Muitos outros exemplos podem ser citados:

configurar o micro-ondas para descongelar o alimento em 5 minutos; configurar uma geladeira para emitir um sinal sonoro se a sua porta ficar aberta por mais de 2 minutos. Em todos esses exemplos, há um *software* embarcado no equipamento.

Assista, abaixo, um vídeo sobre o funcionamento conjunto entre o *hardware* e o *software*, produzido pela GCFGlobal® e disponível no YouTube®.

Vídeo 1 - Entenda a diferença entre *hardware* e *software*.



Boa parte do *software* é formada por um conjunto de instruções definido a partir de algoritmos que deve ser executado pelo computador. Ao contrário do *software*, o *hardware* é tátil e possui massa, dentre outras propriedades comuns a objetos, como uma cadeira, por exemplo. O *hardware* é imaginado, em geral, como um conjunto de placas, circuitos impressos, cabos e conectores. *Software* é um elemento mais conceitual do que material. A materialização de *software* resulta em documentos, textos escritos em linguagens de programação, dados. Observe que a legislação brasileira não usa o termo *software*, mas programa de computador, conforme a Lei Nº 9.609, de 19 de fevereiro de 1998[2], na qual programa de computador é definido como

> a expressão de um conjunto organizado de instruções em linguagem natural ou codificada, contida em suporte físico de qualquer natureza, de emprego necessário em máquinas automáticas de tratamento da informação, dispositivos, instrumentos ou equipamentos periféricos, baseados em técnica digital ou análoga, para fazê-los funcionar de modo e para fins determinados.[2]

No art. 2º, da Lei supracitada, há outra informação que contribui com a percepção de *software*. Programa de computador, no Brasil, recebe a mesma proteção de propriedade intelectual conferida às obras literárias [2].

O presente Microcurso inclui conteúdo visando um entendimento básico sobre os mecanismos empregados por um *software*, o funcionamento e acerca de como se projeta um *software*. Isso contribui com a compreensão sobre computadores, o papel dos algoritmos e da programação. Em outras palavras, o conhecimento essencial para desenvolvermos o nosso pensamento computacional, inclusive no sentido de participarmos da evolução da Saúde Digital.

1.1.3 Ciência da Computação

Embora o primeiro computador eletrônico e digital automático, denominado ENIAC (*Electronic Numerical Integrator and Computer*), tenha sido criado em 1946 e, de lá para cá, o avanço permanece em crescente aceleração, a noção de algoritmos teve início no século VII, por meio do matemático indiano Brahmagupta e do matemático persa al-Khwarizmi. Esse último escreveu o livro "Calculando com numerais hindus" que recebeu uma tradução para o latim ("*Algoritmi de numero Indorum*"). Estes livros trouxeram novos conceitos no sentido de definir sequências de passos para completar tarefas, ou, em outras palavras, a noção de programação [3].

Em 1837, a ideia de computador como conhecemos hoje começou a tomar forma por meio do trabalho de Charles Babbage e seu computador analítico. Foi nessa época que os primeiros programas de computador começaram a ser desenvolvidos pela matemática Ada Lovelace, considerada a primeira programadora[3].

A Ciência da Computação surgiu, no entanto, apenas por volta do ano 1940 por meio do pesquisador britânico Alan Turing, considerado o pai da computação. É que, antes disso, o termo computador estava associado a pessoas que realizavam cálculos, em geral liderados por físicos. Foi somente após a década de 1920 que o termo máquina computacional começou a ser usado para referir-se a uma máquina que pudesse realizar o trabalho de uma pessoa. Posteriormente, no final da década de 1940, o termo foi atualizado para computador. Convém destacar que um computador faz o que um ser humano é capaz de fazer, contudo, com uma extraordinária eficiência[3].

Entre 1946 e 1974, os computadores tiveram o seu uso ampliado nas grandes companhias e no serviço público. No entanto, as máquinas eram muito grandes, ocupando enormes salas das instalações das instituições, e, além disso, seu uso era restrito aos poucos profissionais de computação da época[3].

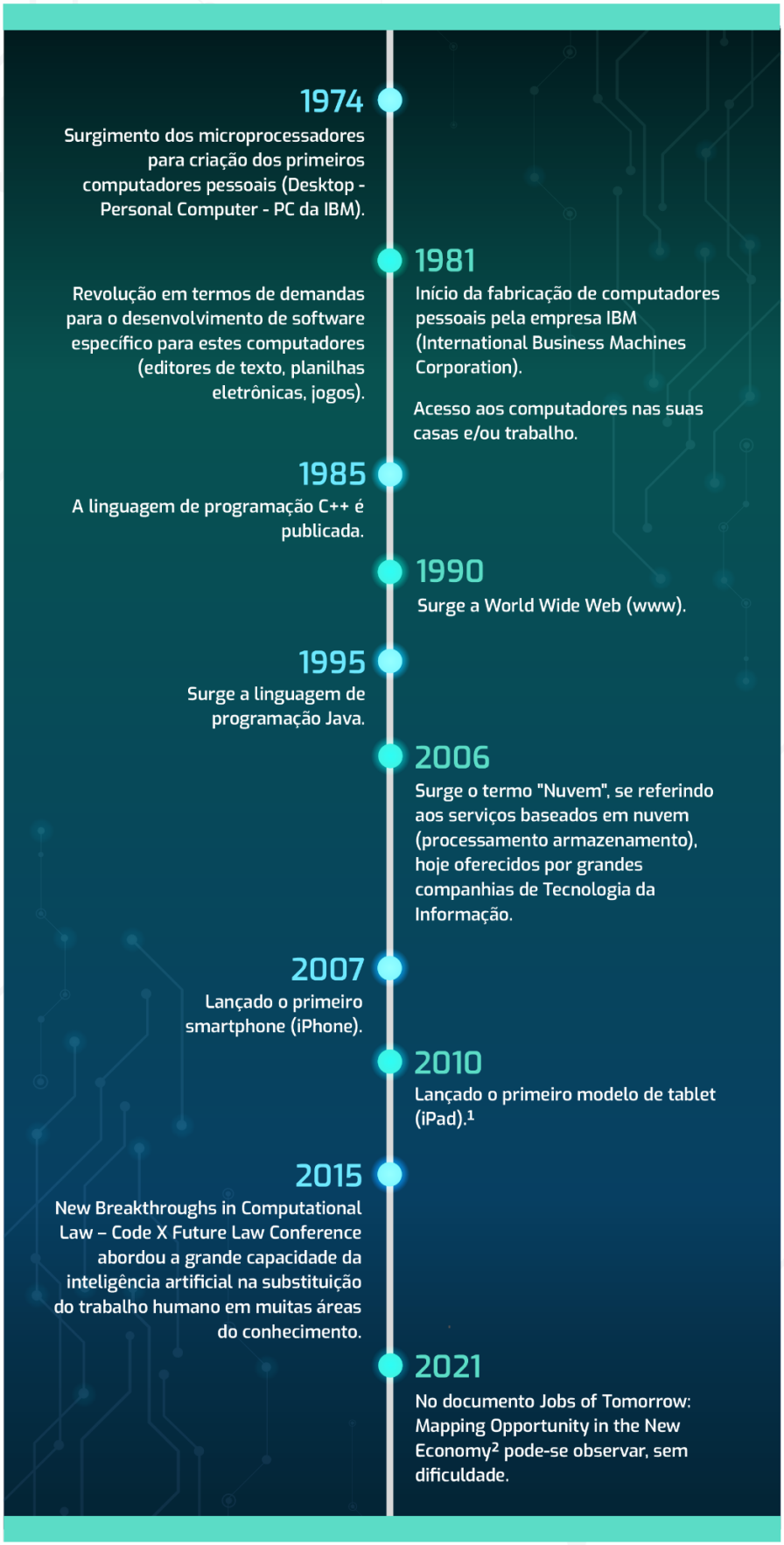
O surgimento dos microprocessadores no ano de 1974 viabilizaram os primeiros computadores pessoais (*Desktop - Personal Computer* [PC] da IBM®), cuja fabricação foi iniciada pela empresa IBM (*International Business Machines Corporation*) em 1981. Isso estendeu o acesso aos computadores além dos profissionais da computação, em suas casas e/ou trabalho, resultando em uma revolução em termos de demandas para o desenvolvimento de *software* específico para esses computadores (editores de texto, planilhas eletrônicas, jogos)[3].

Nos últimos anos do século XX e primeiros anos do século XXI, uma nova mudança ocorreu com o surgimento dos *palmtops*, *smartphones* e *tablets*, modificando sensivelmente a forma das pessoas interagirem com os computadores, aproximando-as ainda mais dos recursos tecnológicos. Essa situação, por conseguinte, reforça a importância de compreendermos melhor como funcionam e como são projetados os *softwares* que usamos. Claro que, associada a essa

evolução, está ainda o uso intenso da *internet* no cotidiano das pessoas: redes 3G, 4G, 5G (G significa a geração) e Wi-Fi (*Wireless Fidelity*) conectando dispositivos móveis e computadores[3].

A história do computador é assunto do Museu do Computador[3]. O leitor também pode consultar aqui a linha do tempo completa dessa história desenvolvida pelo *Computer History Museum* (CHM). Uma alternativa interessante oferece a classificação dos componentes do computador e a história de cada um deles[4].

Figura 1 - A história do computador



Fonte: autoria própria.

A Ciência da Computação é considerada uma ciência jovem e que reúne os estudos das técnicas, metodologias, instrumentos computacionais e aplicações tecnológicas. Embora seja uma ciência recente, a noção de algoritmos e programas computacionais tiveram início ainda no século VII, seguindo com marcos no ano de 1837 e no ano de 1940, como anteriormente descrito. O pensar de forma computacional implica em compreender que tais princípios permanecem presentes no mecanismo e concepção dos algoritmos empregados no *software* moderno. É nesse sentido que, na próxima seção do nosso estudo sobre Pensamento Computacional, vamos explorar esses mecanismos e o funcionamento dos sistemas computacionais.

1.1.4 O Que Torna um Computador Fascinante?

Podemos ilustrar com uma quantidade significativa de exemplos que podem te impressionar ou não, tendo em vista que já fazemos parte da sociedade digital, ou seja, você já está inserido em um contexto onde estas "coisas interessantes" acontecem, o que veremos mais adiante. Por ora, observemos a essência da capacidade de cômputos realizada por um computador atual, disponível em "prateleiras", ou seja, nem precisamos falar em supercomputadores [5].

Vamos falar de um XBox Series X®, o último modelo disponibilizado pela Microsoft®. Se preferir, também pode empregar o PlayStation 5® da Sony®. Convém mencionar que nosso objetivo aqui não é fazer propaganda de nenhum produto. Não há aqui nenhuma recomendação. E esses exemplos não são os únicos. Você pode adquirir um computador com placa de vídeo mais potente do que esses exemplos, a NVidia RTX 3070, já superada por outras duas versões da própria NVidia®, é cerca de três vezes mais eficiente do que esses dois primeiros exemplos.

Já sabemos que não estamos falando de supercomputadores, mas de ferramentas ao nosso alcance cuja capacidade de processamento é medida em *teraflop* (TFLOP). Um TFLOP equivale a 1 trilhão de operações em ponto-flutuante realizadas em 1 segundo. Uma operação em ponto-flutuante é uma operação envolvendo um número real, por exemplo, 5,6 multiplicado por 234. Um trilhão dessas operações realizada em um único segundo equivale a 1 TFLOP. E os exemplos que empregamos acima ultrapassam a capacidade de 10 TFLOPs. Ou seja, podem executar impressionantes 10 trilhões de operações em ponto-flutuante em um único segundo.

Isso significa que eles são "muito rápidos". Ao contrário, um ser humano, faça as suas contas, em toda a sua existência, será capaz de fazer apenas uma fração pequena dessa quantidade de cálculos, realizada em um único segundo por um dispositivo "convencional" hoje em dia.

E a evolução dessa capacidade tem crescido de forma significativa. Em 1969, um computador foi fundamental para a Apollo 11 pousar na Lua. Ele tinha 4 KB (*kilobyte*) de memória RAM (*Random Access Memory*) (inferior ao tamanho dos ícones que empregamos hoje), e 75 KB "de disco", pesava 32 kg, tela de 4 dígitos e uma frequência de 2 MHz (*megahertz)*. Um *smartphone* hoje, não precisa ser o melhor, tem 32 GB "de disco" (isto é cerca de 1 milhão de vezes maior), pesa menos de 200 g (isso é cerca de cem vezes menor), possui um *display* que se aproxima de uma resolução 4 K[2] (como comparar com 4 dígitos?) e possui oito unidades de processamento, cada uma operando com uma frequência de cerca de 2 GHz (mil vezes mais eficiente, cada uma das unidades). Ou seja, parece que repetir a façanha da Apollo 11 não é um obstáculo computacional. A mensagem aqui é que essas máquinas possuem capacidade extraordinária de cálculo e que a capacidade continua se expandindo de forma significativa (veja a Lei de Moore[6]).

2 4K: padrão de resolução da tela que corresponde a 3.840 *pixels* horizontais por 2.160 *pixels* verticais.

Reflita sobre isso...

Se pudermos transformar a solução para um dado problema em números, em cômputos, então um computador pode ajudar.

1.2 Mecanismos e Funcionamento dos Sistemas Computacionais

Entendemos, até agora, como a computação faz parte do nosso cotidiano, incluindo os fundamentos conceituais de *software* e *hardware* e a grande área de conhecimento que a envolve, a Ciência da Computação.

Mas, enfim, como funciona um programa de computador (software)? E o sistema de computação (hardware) que o executa?

Para responder a primeira pergunta, vamos entender mais sobre programas e algoritmos. Um programa de computador é muitas vezes conhecido também por algoritmo, embora existam algumas diferenças. O programa de computador é desenvolvido utilizando-se uma linguagem de programação, em outras palavras, um conjunto de instruções/comandos (também chamado de código) que o sistema de computação consegue interpretar. Há uma diversidade de linguagens de programação disponíveis e utilizadas pelos programadores. Apenas para exemplificar, podemos citar algumas, como a C, C++, Java®, JavaScript®, PHP®, Python®. O algoritmo, por sua vez, constitui-se em uma sequência de passos lógicos que também é elaborada pelo programador, contudo, isso ocorre em uma etapa sucedida pela correspondente implementação usando uma linguagem de programação. Ou seja, o programador elabora o algoritmo e, na sequência, o converte para código em uma linguagem de programação.

A seguir, um exemplo de um algoritmo para calcular o índice de massa corporal (IMC) de um indivíduo (Figura 2).

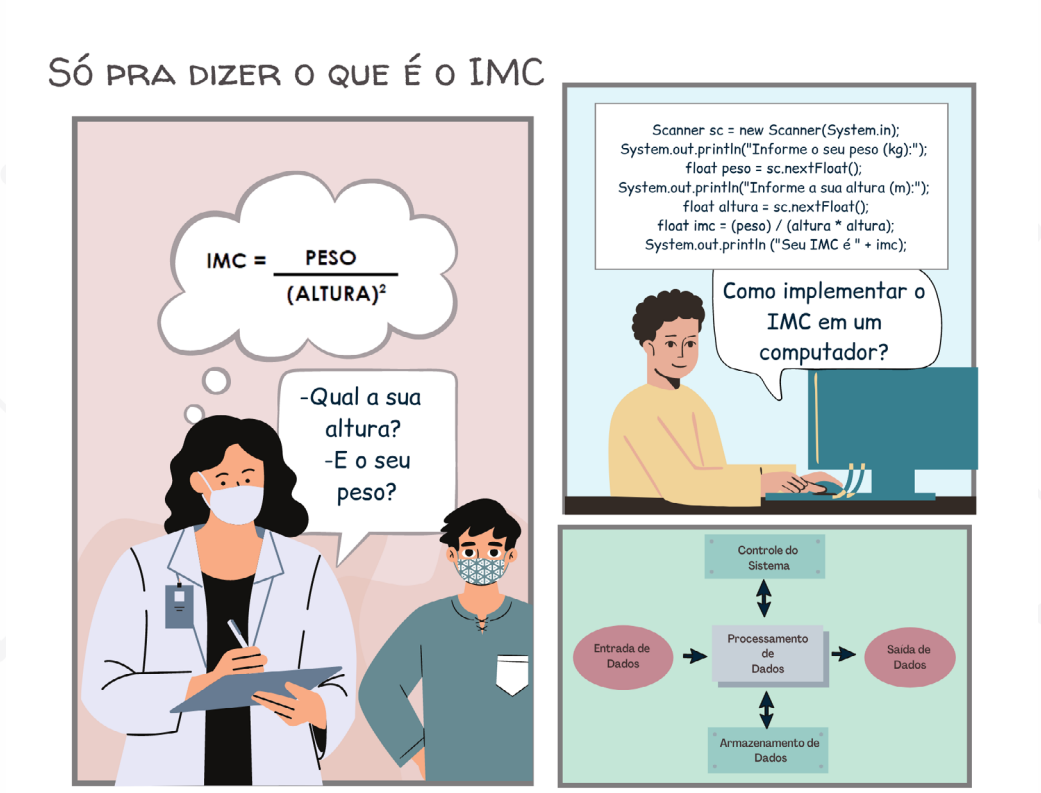
1 - Perguntar ao usuário o seu peso em quilogramas (kg);

2 - Perguntar ao usuário a sua altura em metros (m);

3 - Calcular o IMC utilizando a seguinte fórmula: dividir o peso (em kg) pelo quadrado da altura (em metros);

4 - Apresentar ao usuário o resultado do cálculo do IMC.

Figura 2 - Um exemplo de algoritmo



Fonte: autoria própria.

Quem precisa consultar a internet antes de se aventurar a fazer um pudim, por exemplo, não deve estranhar o algoritmo apresentado. A diferença é que este algoritmo tem como foco a produção de um programa a ser executado por um computador, enquanto a receita de culinária ninguém sabe qual será o resultado (ironia).

O algoritmo apresentado anteriormente estabelece a estratégia a ser adotada pelo programa. Contudo, precisa ser convertido para versão equivalente em uma linguagem de programação. Abaixo, para saciar a curiosidade do leitor, segue trecho parcial de uma possível conversão, simplificada, usando a linguagem de programação Java®:

```
Scanner sc = new Scanner(System.in);
System.out.println("Informe o seu peso (kg):");
float peso = sc.nextFloat();
System.out.println("Informe a sua altura (m):");
float altura = sc.nextFloat();
float imc = (peso) / (altura * altura);
System.out.println ("Seu IMC é " + imc);
```

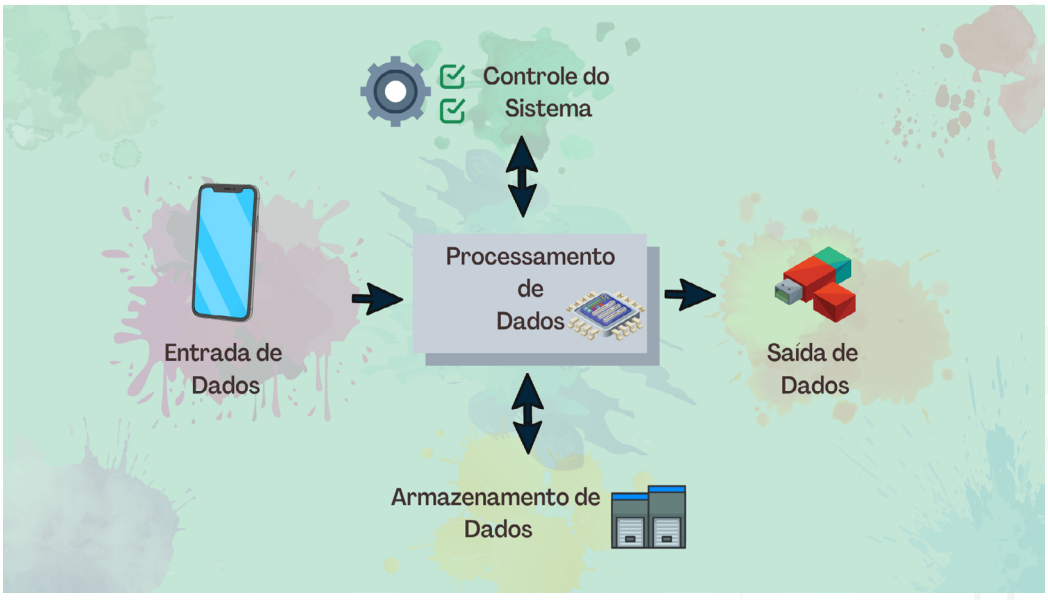
A programação de computadores não faz parte deste Microcurso, ou seja, não é imprescindível compreender o trecho de código apresentado anteriormente. O que deve ser compreendido é que o computador só consegue realizar cálculos, regras e ações codificadas em linguagens especificamente desenvolvidas para tal.

Dando prosseguimento com a missão de esclarecer os mecanismos de funcionamento de um computador, a outra pergunta repetida aqui é:

Como funciona o sistema de computação que executa o programa?

Dado que um programa é uma sequência de instruções, conforme mencionado anteriormente, ele deve ser executado passo a passo pelo sistema (veja a Figura 3).

Figura 3 - Esquema de sequência de instruções para execução de um programa



Fonte: autoria própria.

Caso o nosso programa exemplificado na Figura 2 seja implementado e instalado em um *smartphone* (aplicativo), a entrada de dados (linhas 1 e 2) seria feita digitando-se o peso e a altura diretamente na tela do *smartphone*. Feita a entrada do peso e da altura, o processador (Unidade Central de Processamento [UCP]) do *smartphone* realiza a operação de cálculo do IMC (linha 3), e o resultado é enviado para a tela do *smartphone*. Cabe ressaltar que os dados que transitam pelo sistema (entrada-processamento-saída) são armazenados na memória do sistema computacional. Perceba que se o mesmo programa fosse executado em um *desktop* ou em um *laptop*, a entrada de dados poderia ser feita usando-se um teclado e mouse, e o resultado seria enviado para um monitor de vídeo.

A execução do programa (*software*) é realizada pelo *hardware*, composto por dispositivos de entrada, dispositivos de saída, processador e memória. No nosso dia a dia, é comum termos que tomar decisões, como, por exemplo, capacidade de memória e capacidade de processamento de um computador, seja do tipo *desktop*, *notebook*, *tablet* ou *smartphone*, ou mesmo, *smartwatch* e *smart TV*. A capacidade necessária de memória (para armazenar os dados) e de processador (para processar os dados) está diretamente relacionada à demanda que se espera que o computador em questão possa atender, ou seja, a quantidade de programas que são executados no sistema computacional e do volume de dados. Quanto mais programas estiverem em execução simultaneamente em um sistema computacional, mais demanda haverá por processamento e por memória.

Imagine que você esteja usando o seu *smartphone* para participar de uma reunião remota com colegas do trabalho. Para isso, você utiliza um *software* específico que acessa a câmera de vídeo, áudio e microfone. Durante a reunião, muitos dados estão transitando (texto, áudio e vídeo) e ocupando recursos do processador e da memória, além da conexão com a *internet*. No mesmo *smartphone*, durante a mesma reunião, você decide utilizar outro aplicativo, por exemplo, um editor de textos. A demanda resultante pode fazer com que você experimente certa lentidão, não detectada anteriormente, mas presente após a maior exigência de processamento e tráfego de dados.

É importante evidenciar que os mecanismos e funcionamento dos sistemas computacionais são intrinsecamente complexos. A próxima unidade observa a forma na qual sistemas são construídos, em especial, com foco na Engenharia de *Software* como um dos elementos de conhecimento essenciais para a compreensão do pensamento computacional.

Vídeo 2 - Videoaula: computação presente em quase tudo

Estamos vivenciando uma época de intensas mudanças, como jamais vistas, patrocinadas pelo que conhecemos por transformação digital. “Tudo” está sendo digitalizado, resultando na sociedade digital, na qual tudo está conectado. Até o nosso corpo está se apropriando de uma infinidade de sensores, a entrega pode ser feita via drones, carros autônomos, cidades inteligentes. Imagine a conexão em uma escala ainda maior do que a conhecemos hoje, reunindo praticamente tudo que vemos e inclusive o que não vemos, nem ouvimos ao nosso redor. A internet das coisas (IoT) é um habilitador dessa interconexão sem precedentes.

Hoje, você tem uma assistente para muitas tarefas (Google® Assistant®), mais de um milhão de títulos em suas mãos, “impressos” em branco de papel, e a prova d’água (Kindle Paperwhite®). Aspirador de pó, condicionador de ar, máquina de lavar, geladeiras conversando entre si. No domínio da saúde, apenas para estabelecer um foco de atenção, reúna tudo isso com uma atenção ao nosso sono (Sleep as Android®), monitoramento de batimentos cardíacos com precisão (Polar®), monitoramento do estresse (Pip®), eletrocardiograma onde quer que você esteja (AliveCor®), sinais vitais (CheckMe Pro®), eletroencefalograma dirigido para o sono e a meditação (Muse®), acompanhar com precisão os movimentos de todo o corpo (Teslasuit®), exoesqueleto para reduzir riscos (EksoBionics®) e luvas para auxiliar o trabalho com as mãos (ProGlove®), dentre muitos outros recursos disponíveis. Imagine a quantidade significativa de informações que podem transitar pela Rede Nacional de Dados em Saúde (RNDS) e serem empregadas em benefício da nossa saúde. E nem falamos das interfaces entre cérebro e computador (Emotiv®) ou de um eletromiógrafo de superfície que pode viabilizar um “diálogo silencioso”, quem sabe, estamos apenas no início.

1.3 Para relembrar...

Como um programa (software) funciona?

RESPOSTA

O que é um algoritmo?

RESPOSTA

Qual a importância da Ciência da Computação?

RESPOSTA

De que forma a computação está presente no nosso cotidiano?

RESPOSTA

Os smartphones são considerados dispositivos computacionais?

RESPOSTA

É hora do quiz!

1- Software é tudo que está ligado à parte física de um dispositivo.

Verdadeiro

Falso

RESPOSTA

2- Nos últimos anos do século XX e primeiros anos do século XXI, uma nova mudança ocorreu com o surgimento dos palmtops, smartphones e tablets, modificando sensivelmente a forma das pessoas interagirem com os computadores, isso proporcionou a aproximação ainda maior dos recursos tecnológicos, modificando a forma das pessoas interagirem com os computadores.

Verdadeiro

Falso

RESPOSTA

EDUCAÇÃO E CAPACITAÇÃO
DE RECURSOS HUMANOS
EM SAÚDE DIGITAL

Pensamento computacional

Unidade 2

Como são construídos os sistemas computacionais

Fábio Nogueira de Lucena
Sérgio Teixeira de Carvalho

2.1 Introdução da Unidade

Um *software* não é gerado de forma espontânea pela natureza, ele não surge, ele não aparece simplesmente. É preciso um esforço intencional e dirigido para produzi-lo. Em geral, esse esforço é significativo, o que resulta em custos e também alimenta um mercado enorme em todo o mundo. Este mercado é produto de desejos ou necessidades que existem externamente ao *software*.

A Engenharia de *Software* é a área do conhecimento que trata das estratégias que orientam o esforço de produção de *software*, desde a "captura" da necessidade até a correspondente implantação e a posterior evolução (manutenção). Ao longo do tempo, esse esforço é, geralmente, distribuído no que é conhecido por ciclo de vida de *software*.

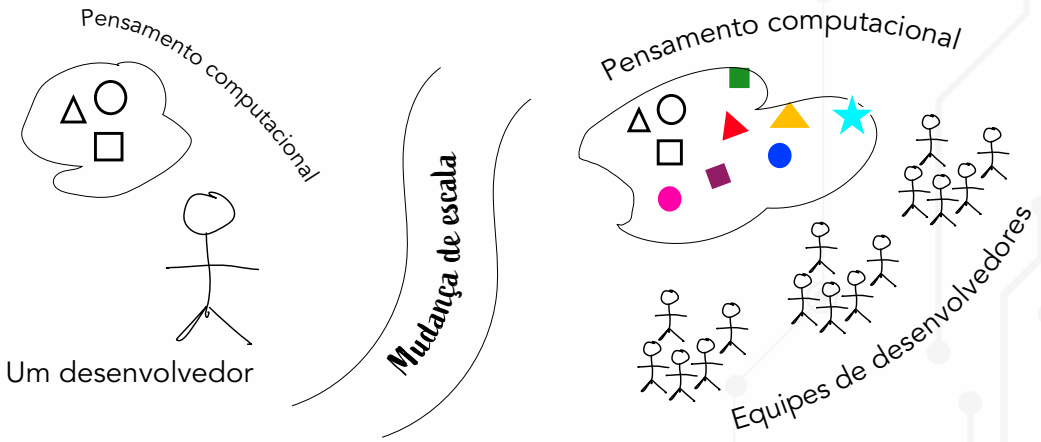
Nesta Unidade, em linguagem informal, menos acadêmica, são apresentadas algumas das questões de "significativo" impacto da Engenharia de *Software*. Primeiro, o porquê da Engenharia de *Software*.

2.1.1 Engenharia de *Software*: Uma Necessidade

Convém destacar que a Engenharia de *Software* surge apenas perante a limitação do pensamento computacional quando restrito ao que é possível de ser feito por um único desenvolvedor. Se as habilidades empregadas por um único desenvolvedor ou o pensamento computacional necessário são suficientes para desenvolver e manter um determinado *software*, então, a Engenharia de *Software* terá menos impacto nesse contexto.

Dito de outra forma, quando a produção e evolução de *software* demanda equipes de programadores, são necessárias práticas e ferramentas mentais para equipes de programadores e não apenas aquelas adequadas para um único programador. Essa dicotomia é conhecida, em inglês, por *thinking in the small* e *thinking in the large*. Na Figura 4, são ilustradas essas duas perspectivas.

Figura 4 - Pensamento computacional na Engenharia de *Software*



Fonte: autoria própria.

Tal mudança, convém ressaltar, é necessária para lidar com *software* "mais complexo", ou problema "grande", e as dificuldades inerentes. É justamente essa mudança de escala que cria o contexto preenchido pela Engenharia de *Software*.

Na obra de Denning & Tedre (2019)[7], essa alteração de escala é apresentada em detalhes. De forma resumida, o pensamento computacional foi estendido, ao longo do tempo, com novas ferramentas propostas para atender os cenários cada vez mais complexos, para os quais recursos básicos de programação não são suficientes.

Dentre os elementos que se tornaram relevantes para atender essa mudança de escala, apenas para satisfazer a curiosidade do leitor, dado que as explanações pertinentes ocupam mais espaço e tempo do que aqueles disponíveis, encontram-se:

a) novas práticas de decomposição;
b) abstração de dados;
c) encapsulamento;
d) ocultamento de informação;
e) recursão;
f) gerenciamento de projeto; e
g) ciclo de vida de *software*.

Esta lista não é exaustiva.

2.1.2 Noção de Tamanho, Custo e Esforço

Em *Information is beautiful*[8], estão disponíveis dados sobre bases de código. Um deles é sobre tamanho de *software*. Por exemplo, um carro "moderno" em 2013, continha cerca de 100 milhões de linhas de código (diz-se, "código embarcado"). Conforme dados do portal, código "robusto" desenvolvido com qualidade custa 40 dólares cada linha. Se nem tanta qualidade é necessária, então pode cair para 15 dólares cada linha. Também cita que, para produzir 1 milhão de linhas de código você vai precisar de 67 pessoas ao longo de 40 meses.

Apesar dos valores do parágrafo anterior não estabelecerem regras universais, são suficientes como referência e, em particular, para fornecer uma noção da complexidade e esforço necessários para se produzir *software*.

Na Figura 4 vemos um programador e várias linhas de código. E como tais linhas se comparam em volume com outras tipos de linha? Observe, por exemplo, o tamanho de livros. A Bíblia possui cerca de 800.000 palavras (observe que não são linhas), enquanto a obra literária mais extensa, segundo o Wikipédia, possui menos de 4.000.000 de palavras 9.

Antes de prosseguir, convém fornecer alguma luz sobre a distinção entre ciência e engenharia na seção seguinte. Nas posteriores, respectivamente, é fornecida uma noção de como, ao longo do tempo, *software* é desenvolvido, uma visão mais detalhada de projeto (*design*) de *software* e, por fim, considerações finais.

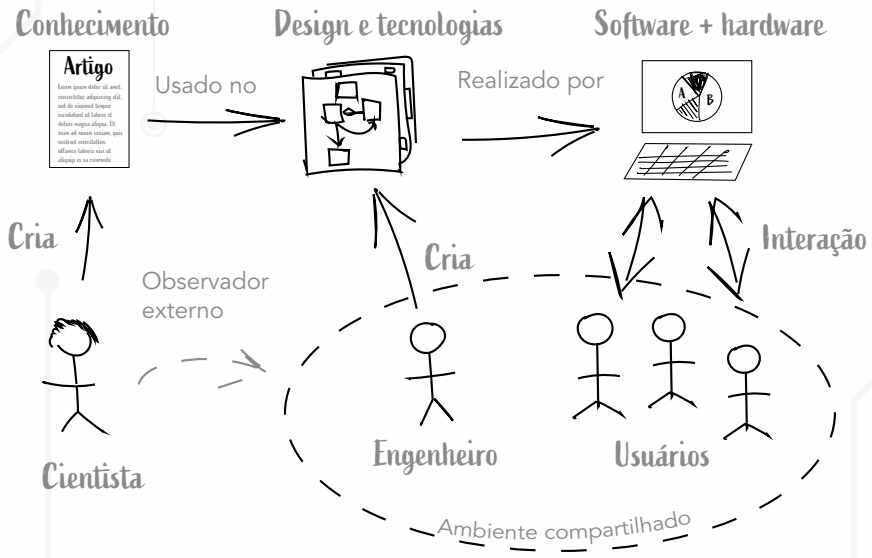
2.2 Ciência e Engenharia (Inclusive de *Software*)

A distinção entre ciência e engenharia é útil para caracterizar a Engenharia de *Software* e, dessa forma, contribui com a percepção do pensamento computacional necessário para lidar com *software*, em particular, *software* a ser usufruído por muitos usuários, com uma expectativa longa de vida e formado por muitos programas.

Antes que usuários possam interagir com um sistema computacional, possivelmente milhares deles, distribuídos por vários fusos horários, engenheiros tiveram que dialogar com alguns deles, assim como outros interessados (*stakeholders*) até o topo da gestão do negócio em questão. Também foi necessário usar o conhecimento disponível no *Software Engineering Body of Knowledge (*SWEBOK*)*, propor alternativas de solução, selecionar um *design*, produzir e usar tecnologias na construção desse sistema computacional.

Conforme mencionado anteriormente, os engenheiros estiveram próximos dos usuários, compartilharam o ambiente de uso do sistema, para produzi-lo, enquanto os cientistas, se observaram esse ambiente, foi externamente. Os cientistas possuem a atenção no corpo de conhecimento, nos dados e nas informações que fazem parte do corpo de conhecimento (Figura 5).

Figura 5 - Ciência *versus* Engenharia



Fonte: autoria própria.

Os engenheiros realizaram suas atividades visando atender a demanda de usuários e de outros interessados (*stakeholders*). Por exemplo, por um lado pode ser desejável um determinado recurso para os usuários, por outro, o tempo necessário e custos correspondentes podem ser proibitivos para a direção da empresa em questão. Engenheiros devem conciliar, muitas vezes, objetivos conflitantes. Ou seja, apresentar uma solução prática para o problema, sem ignorar outras questões como o custo, dentre outras. Nesse esforço realizado pelos engenheiros, o conhecimento produzido por cientistas é insumo. Cientistas estão atentos a leis, engenheiros necessitam considerar os detalhes para que os sistemas funcionem.

2.3 Modelo de Desenvolvimento de *software*

A criação de um *software* depende da execução de várias atividades que podem ser genericamente agrupadas em[10]:

a) comunicação
b) planejamento
c) modelagem;
d) construção; e
e) implantação.

Essas atividades não são as únicas, há outras, mais afastadas do "núcleo técnico", tais como, gerenciamento de risco e garantia da qualidade, por exemplo. Simplesmente citar tais atividades não é esclarecedor.

Por outro lado, é possível visualizar a execução dessas atividades sem fazer um curso superior em Engenharia de Software? Podemos tentar.

Vamos nos concentrar no código propriamente dito e, em particular, nas mudanças que o código recebe ao longo do tempo. Ou seja, colocar o foco em parte destas atividades, além de ignorar interações entre os membros da equipe de desenvolvimento, a análise de alternativas, o esforço de projeto e o desafio intelectual (comentado adiante), dentre outras questões. O que resta, com essas exclusões, é a parte mais "concreta" do desenvolvimento de *software*, conforme já mencionado, a produção de código.

A produção de código de um *software* amplamente empregado em todo o mundo, a saber, o código pertinente ao núcleo do sistema operacional Linux®, após as exclusões citadas no parágrafo anterior, é ilustrada no vídeo https://youtu.be/5iFnzr73XXk.

Essa visualização mostra alterações ocorridas ao longo do tempo sobre um mesmo código, por vários desenvolvedores. Ela permite ao leitor refletir sobre as interconexões entre mudanças, a complexidade inerente quando mais pessoas participam do trabalho, e ilustra apenas parte do desenvolvimento de um *software* real. Convém reiterar que não inclui comunicação, planejamento, modelagem nem implantação, mas apenas parte da atividade de construção, citada no início.

Não é necessário acompanhar todo o vídeo de quase duas horas, você pode pular partes, a intenção é disparar a reflexão das inúmeras interações entre pessoas e entre pessoas e código produzido por elas ao longo do tempo. Talvez seja um exercício mental interessante comparar com a sua prática profissional, qualquer que seja ela.

Em vez de detalhar as atividades apresentadas no primeiro parágrafo, o presente texto segue o sentido inverso, em uma perspectiva ainda mais abstrata. Esse distanciamento dos detalhes revela que desenvolver *software* pode ser visto como um processo composto pela criação de modelos e, na sequência, conversão deles em um formato que computadores são capazes de executar.

Nessa perspectiva, ao longo do desenvolvimento de um *software* está-se sempre produzindo modelos e transformando-os em formato correspondente, mas que é possível de ser executado por um computador. Tal esforço depende de significativo conjunto de processos, métodos e ferramentas, todo ele orientado pelo foco na qualidade. Os detalhes aqui evitados podem ser esclarecidos em Pressman & Maxim (2020)[10], uma das principais referências sobre o assunto.

Métodos ágeis de desenvolvimento de *software*, Scrum, arquitetura de *software*, engenharia de requisitos e microsserviços são termos que provavelmente já chamaram a sua atenção, todos são pertinentes a *software*. Outro do momento é a Inteligência Artificial, principalmente por meio de aprendizado de máquina (*machine learning*), robótica, sistemas de apoio à decisão, visão computacional, reconhecimento de voz, interpretação de texto, jogos e outros. Novamente, tudo isso se realiza por meio de *software* e ilustra a extensão do assunto.

Em vez de tratar cada um desses elementos ou escolher aqueles "mais relevantes", o que depende do observador, optou-se pelas habilidades e práticas mentais dirigidas ao esforço de projeto (*designing*), conforme já mencionado. Não apenas porque está no centro das atenções do desenvolvimento de *software*, mas também por acomodar muitas estratégias que refletem o pensamento computacional.

2.4 Projeto de *Software*

Projeto de *software* é o que todo engenheiro de *software* deseja fazer, é o espaço da criatividade onde requisitos e considerações técnicas são empregados na definição do *software*[10]. O resultado é uma representação ou modelo do *software* que inclui elementos como arquitetura, estruturas de dados, interfaces e componentes. Tais elementos não se confundem com o código propriamente dito, que é produzido a partir de tais abstrações. O projeto de *software*, portanto, resulta em representações ou modelos que são realizados por meio de código.

O projeto e a construção são referenciados em Brooks-Jr. (1987)[11] como tarefas essenciais e acidentais, respectivamente. Segundo ele, todo desenvolvimento de qualquer *software* envolve a execução de tarefas essenciais, a elaboração de estruturas conceituais complexas que compõem a entidade de *software* abstrata, e tarefas acidentais, a representação destas entidades abstratas em linguagens de programação e outros elementos computacionais.

Ao longo de décadas foram acumulados muitos conhecimentos sobre projeto de *software* na forma de princípios, conceitos e práticas[10]. A ênfase dessa seção é depositada no que é pertinente a *thinking in the large* ou problemas "grandes", o que justifica a necessidade da Engenharia de *Software*. *Thinking in the small* também é importante, embora não discutido nesta Unidade.

2.4.1 Gerenciamento da Complexidade: o Principal Foco do *Design*

O gerenciamento da complexidade é o mais importante tópico técnico do desenvolvimento de *software*[12]. Convém ressaltar que complexidade não é uma característica nova de desenvolvimento de *software*. McConnell cita que a computação é a única profissão na qual uma única mente é obrigada a percorrer a distância de várias ordens de grandeza, o que representa um desafio intelectual sem precedentes, ao desenvolver um *software*. De fato, entre

1 bit até algumas centenas de terabytes, o que é onze ordens de grandeza superior, é um cenário comum atualmente[12].

McConnell ainda acrescenta que há três estratégias de projeto não eficientes para lidar com a complexidade[12]:

1) Solução complexa para um problema simples.
2) Uma solução simples, mas incorreta, para um problema complexo.
3) Uma solução complexa, inapropriada, para um problema complexo.

O que se deseja destacar é que, embora a complexidade do problema faça parte da natureza do mesmo e que provavelmente não possa ser alterada (entrada), a solução (saída) é projetada, é criada pelo ser humano. O projeto realizado pelo ser humano está suscetível a uma das abordagens não eficientes citadas acima (evitá-las é um desafio).

A complexidade do sistema computacional criado para lidar com um problema é definida pela complexidade do domínio do problema em questão, que não pode ser alterada, adicionada da complexidade da solução proposta, esta última pode introduzir uma complexidade não essencial, além daquela necessária, por exemplo, por não conseguir evitar uma das estratégias inadequadas citadas acima. Em consequência, uma das principais características desejáveis em um projeto de *software* é "reduzir a complexidade".

2.4.2 Arquitetura de *Software*

Um sistema computacional envolve *hardware* e *software*. Alguns dependem de *hardware* específico ou exclusivamente projetado para a finalidade em questão, outros não. Por exemplo, em geral um oxímetro possui um *hardware* e *software* específicos. Por outro lado, um aplicativo de celular que se propõe a realizar o papel de um oxímetro, faz uso de um *hardware* de propósito geral.

Observe que sistemas computacionais podem ser avaliados em "bons" e "ruins", nem todos são "bons", e que está além do escopo deste texto fornecer qualquer detalhe sobre a qualidade de um software executado em aparelho de celular para desempenhar o papel de um oxímetro. Sistemas computacionais podem causar danos.

Independentemente de um sistema computacional ser intensivo em *software* ou não, todo sistema computacional possui uma arquitetura de sistema correspondente e, a parte que cabe a *software*, nesse sistema, possui a sua própria arquitetura, a arquitetura de *software*. A arquitetura é importante, pois possui impacto significativo no custo e na qualidade do *software* em questão.

A arquitetura de *software* define como um *software* está organizado em componentes e como esses componentes interagem entre si. Uma ferramenta indispensável para a definição da arquitetura de *software* é a decomposição. A identificação de "partes", os componentes, não é tão simples quanto parece. Muitos critérios podem ser empregados e isso depende de um julgamento do cenário em questão.

Observe, por exemplo, a noção corrente de que as funções a serem executadas por um *software*, em geral, não estão claramente definidas no início do desenvolvimento. Ou seja, é

amplamente aceito que as funções atribuídas a um *software* vão mudar ao longo do tempo, seja porque não se tinha conhecimento delas no início do desenvolvimento ou porque mudanças nas leis, nas regras, no mundo, exigem adaptações correspondentes no *software* em questão. Em consequência, é preciso evoluir, ao longo do tempo, uma arquitetura de *software* inicialmente projetada para as funções conhecidas naquele momento inicial, e constantemente ajustá-la para acomodar as inevitáveis mudanças nas funções que o *software* em questão deve desempenhar.

O parágrafo anterior não tem a intenção de investigar um tópico técnico de projeto de *software*, mas fornecer alguma orientação de que a definição da arquitetura de um *software* é assunto extenso e complexo por si só. Não existe "bala de prata", não existe a ideia de que microsserviço, para citar termo muito frequente e em moda quando o assunto é arquitetura de *software*, é um remédio para todos os males. O leitor inclinado a uma maior investigação de arquitetura de *software* pode consultar Ford *et al*. (2017)[13].

Por fim, em um mundo cujas mudanças são a regra, e cada vez mais rápidas, de onde vem a ideia de que a organização de um software pode ser pré-fabricada?

2.4.3 Estimativa de *Software*

Um *software* desenvolvido por equipes de desenvolvedores, leia-se custo significativo, precisa de um acompanhamento que possa refletir o andamento do projeto. A dificuldade de acompanhar o desenvolvimento de *software* realizado por vários desenvolvedores conta com a ajuda do livro The *Mythical Man-Month*[14], provavelmente, o mais clássico dos livros da literatura em Engenharia de *Software*.

Segundo Brooks (1995)[14], técnicas para a estimativa de *software* ainda não estão adequadamente desenvolvidas, e existe uma falácia na qual pode-se "trocar" homens por meses. Dito de outra forma, em *software*, o que é feito por um desenvolvedor em três meses não necessariamente pode ser feito por três desenvolvedores em um mês. Sobre este assunto, dentre outras, o livro ainda acrescenta que quando um projeto está atrasado, mais pessoas são adicionadas à equipe, em uma resposta natural para satisfazer as metas previstas, mas, isso agrava a situação.

Naturalmente, os parágrafos anteriores não são suficientes para tratar as questões pertinentes com a devida atenção, mas trazem à consciência do leitor dificuldades que, de outra forma, poderiam não ser percebidas. Outra questão inerente a problemas "grandes" inclui a Lei de Conway.

Conway, em uma história interessante, que pode ser consultada no portal do próprio autor[15], conta como surge esta lei, em 1967. Vamos omitir a história e ir direto à lei, em uma tradução livre: "Toda organização que projeta um sistema produzirá um projeto cuja estrutura é uma cópia da estrutura de comunicação da organização." (Lei de Conway)

Qual a consequência da Lei de Conway?

A expectativa natural em qualquer iniciativa de desenvolvimento de *software*, convém deixar clara, é que o *software* satisfaça os requisitos estabelecidos e, não necessariamente, o que já está definido pelos canais de comunicação da organização que o desenvolve. Em consequência, há quem diga que a organização tenha que ser alterada para que não reflita de forma indevida na organização do *software* a ser desenvolvido. O fato é que o desenvolvedor deve evitar essa ingerência da estrutura da organização na estrutura do *software*.

2.4.4 Computação Positiva

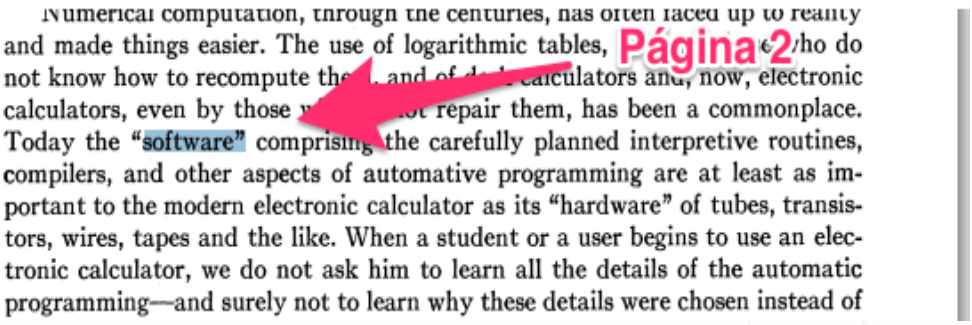
A atenção a aspectos de usabilidade, visando melhor aceitação e produtividade dos usuários de sistemas computacionais, o que tem dominado o projeto de *software* que interage com usuários, por muito tempo, já não é mais "suficiente". A psicologia positiva é a base da computação positiva[16] que visa o projeto e o desenvolvimento de tecnologias para apoiar o bem-estar psicológico e o potencial humano.

A computação positiva é uma proposta significativa de avanço do papel de sistemas computacionais bem além de apoiar a execução de processos, ou de cômputos. Para ilustrar, são muito comuns as referências a computadores e usuários raivosos, possivelmente, a própria experiência do leitor contribui com uma história. Isso é conhecido como *computer rage* [17]. Não há dúvidas sobre possíveis impactos psicológicos negativos do uso de sistemas computacionais e a computação positiva busca o oposto, ou seja, o uso desses sistemas como instrumento para a promoção do bem-estar dos seus usuários.

2.5 Curiosidades e Considerações Finais

A primeira citação conhecida do termo "*software*" ocorreu no artigo *The Teaching of Concrete Mathematics*, de John W. Turkey, página 2, publicado no *The American Mathematical Monthly*, em 1958 [18], conforme destacada a seguir (Figura 6).

Figura 6 - Recorte do artigo *The Teaching of Concrete Mathematics*


Numerical computation, through the centuries, has often faced up to reality
and made things easier. The use of logarithmic tables, [illegible]who do
not know how to recompute the[illegible], and of [illegible]calculators and, now, electronic
calculators, even by those [illegible]repair them, has been a commonplace.
Today the "software" comprising the carefully planned interpretive routines,
compilers, and other aspects of automative programming are at least as im-
portant to the modern electronic calculator as its "hardware" of tubes, transis-
tors, wires, tapes and the like. When a student or a user begins to use an elec-
tronic calculator, we do not ask him to learn all the details of the automatic
programming—and surely not to learn why these details were chosen instead of

Fonte: Turkey (1958)[18].

O termo software é recente e, somente uma década após, surge Engenharia de *Software*, atribuído a Margaret Hamilton[19]. Muitos outros atores contribuíram para avanços nessa área e uma perspectiva histórica pode ser consultada para detalhes[20]. Alan Turing é um deles, inclusive, empresta o nome para o maior prêmio concedido a realizações no âmbito da computação. De fato, as premiações desde 1966 podem ser consultadas[21] como uma outra perspectiva histórica.

Feitas as considerações históricas, essa Unidade citou um livro clássico, *The Mythical Man-Month*[14], reeditado em 1995 no seu vigésimo aniversário e ainda útil, quase meio século após a primeira edição. O autor, Frederick Brooks, é o mesmo de um artigo também clássico em Engenharia de *Software*: *No silver bullet: essence and accident in software engineering*[22].

A ausência de uma "bala de prata" significa que não há expectativa de desenvolvimento de "solução mágica" para lidar com algumas dificuldades da Engenharia de *Software*. Brooks ainda afirma que a programação é um processo criativo e que alguns projetistas são melhores que outros. Projeto de *software* (*design*) é um tópico amplamente discutido, contudo, o contexto apresentado por Brooks persiste e não é por acaso que tal autor foi laureado com o Prêmio Turing em 1999. Em vídeo recente, uma rica e inspiradora associação entre arte e programação é uma sugestão para se aproximar do fascinante mundo da produção de *software* e suas especificidades[23].

Concluímos essa Unidade com uma tradução de trecho disponível em no livro texto clássico[10] em cursos de Engenharia de *Software* pelo mundo, segundo o qual a distância entre *software* complexo e métodos para lidar com eles persiste, reiterando a inexistência de uma "bala de prata":

> Mesmo que as experiências do passado digam o contrário, há um tácito desejo para encontrar a 'bala de prata' (*silver bullet*) - o processo mágico ou tecnologia que nos permitirá construir sistemas de software grandes e complexos, com facilidade, sem confusão, sem erros, sem atraso - sem os vários problemas que continuam a atormentar o desenvolvimento de software. (p. 583)[10]

Para relembrar...

O termo software é antigo?

RESPOSTA

Há poucas mulheres no domínio da Engenharia de Software?

RESPOSTA

O mercado de software é enorme.

RESPOSTA

Existem vários desafios na produção de software.

RESPOSTA

Gerenciar a complexidade é o principal desafio do design de software?

RESPOSTA

De onde vem a complexidade?

RESPOSTA

Pode-se trocar homens por meses quando o assunto é software?

RESPOSTA

Engenharia de software trata de thinking in the large?

RESPOSTA

É hora do quiz!

1- Projeto de Software é, segundo McConnell (2016) o mais importante termo técnico do desenvolvimento de software, uma vez que tem seu foco voltado para a codificação.

Verdadeiro

Falso

RESPOSTA

2- A psicologia positiva é a base da computação positiva, uma vez que tem por objetivo o desenvolvimento de tecnologias que apoiam o bem-estar psicológico e o potencial humano.

Verdadeiro

Falso

RESPOSTA

EDUCAÇÃO E CAPACITAÇÃO
DE RECURSOS HUMANOS
EM SAÚDE DIGITAL

Pensamento computacional

Unidade 3
Perspectivas

Fábio Nogueira de Lucena
Sérgio Teixeira de Carvalho

Unidade 3: Perspectivas

3.1 Sociedade Digital e Saúde Digital

Há uma infinidade de dispositivos computacionais que nos cerca (veja Unidade 1). A adoção crescente e integração desses dispositivos, seja no trabalho, em casa, no ensino ou na recreação, tem até um nome: sociedade digital. Isso é tão intenso que a alfabetização e a inclusão digitais estão entre iniciativas comuns divulgadas em noticiários. Estamos em clara e rápida expansão em direção à sociedade digital, na qual praticamente tudo faz uso de algoritmos, processamento de dados ou, para simplificar, está baseado no emprego de *hardware* e *software*.

A saúde também está presente na sociedade digital, inclusive desfruta de uma denominação específica que retrata a penetração de sistemas computacionais neste domínio: saúde digital. A Organização Mundial da Saúde publicou recentemente um conjunto de diretrizes para orientar o emprego de intervenções digitais para o fortalecimento de sistemas de saúde[24]. Nesse caso, são intervenções já baseadas em evidências. Conforme essas diretrizes, as aplicações de sistemas computacionais podem estar classificadas naquelas dirigidas para os:

a) usuários;
b) profissionais da saúde;
c) gestores dos sistemas de saúde e
d) serviços de dados.

Ou seja, as diretrizes fornecem uma visão da extensão das possibilidades de uso de sistemas computacionais na saúde. Essa extensão conta até com a definição de uma área do conhecimento: a informática em saúde, a qual pode ser vista como um campo multidisciplinar do conhecimento que lida com questões pertinentes ao uso de recursos de TIC para fornecer serviços de saúde. Há vários periódicos especializados nessa área, como o *Journal of Health Informatics*[25].

No Brasil, outra evidência está na organização de uma sociedade específica. A Sociedade Brasileira de Informática em Saúde (SBIS)[26] tem apoiado ações para promover a Saúde Digital. A SBIS é responsável pela realização, a cada dois anos, do Congresso Brasileiro de Informática em Saúde (CBIS), que reúne os principais nomes nacionais sobre o assunto, além de ser uma fonte significativa de informações atualizadas sobre esse domínio.

Ao considerar o escopo nacional, o Ministério da Saúde, por meio do Departamento de Informática do Sistema Único de Saúde (DATASUS), lidera iniciativas como a RNDS e a Estratégia de Saúde Digital para o Brasil 2020-2028 (ESD28)[27].

3.2 Engenharia de *Software* e Inteligência Artificial

Neste Microcurso, tanto Engenharia de *Software* quanto Inteligência Artificial são tratadas como áreas do conhecimento, cada uma com o seu próprio corpo de conhecimento, métodos, técnicas e ferramentas. Ambas contribuem com o desenvolvimento de soluções computacionais. Em geral, onde uma "brilha", a outra tem menos impacto, o que sugere que a distinção possa ser melhor estabelecida pelo domínio dos problemas contemplados por cada uma delas. É comum encontrar a associação entre Inteligência Artificial e problemas "complexos" ou problemas que demandam o uso de heurísticas para a busca de soluções.

Observe, por exemplo, a proposta de um automóvel autônomo. Há *software* de milhões de linhas de código, desenvolvidas usando os métodos da Engenharia de *Software*, embarcado nesse automóvel (a unidade anterior oferece dados sobre isso). Nem tudo, porém, pode ser adequadamente tratado pela Engenharia de *Software*. Por exemplo, parece inviável definir um conjunto de regras precisas a serem seguidas por um motorista visando a segurança de todos, portanto, a autonomia do automóvel não virá dos métodos da Engenharia de *Software*, mas daqueles da Inteligência Artificial.

Na saúde, a avaliação de risco de um paciente, por exemplo, faz uso de informações de um domínio extenso. À semelhança do cenário anterior, não parece exequível estabelecer um conjunto de regras precisas que, quando executadas, resultem no risco desejado. Talvez, um valor preciso nem seja viável, mas a simples ordenação entre dois pacientes que não compartilham as mesmas doenças. Novamente, cenários assim demandam o emprego de métodos da Inteligência Artificial.

Os métodos da Inteligência Artificial visam lidar com questões na tradução e interpretação de texto, no reconhecimento facial, na identificação do valor calórico de um dado prato pela imagem correspondente, na sugestão de um filme, e assim por diante. Nesses cenários, os métodos da Engenharia de *Software* não vão produzir os resultados esperados.

O inverso também é verdade. Por exemplo, para problemas para os quais existem regras bem definidas de solução, os métodos da Engenharia de *Software* tendem a prevalecer, até porque um resultado aproximado pode não ser admitido.

Convém observar que uma solução que faz uso dos métodos da Inteligência Artificial se materializa por meio de *software* e, neste sentido, métodos da Engenharia de *Software* podem ser empregados ao longo do uso dos métodos da Inteligência Artificial. Curiosamente, o inverso também é verdade, ou seja, a aplicação de métodos da Engenharia de *Software* pode ser auxiliada pelo emprego de métodos de Inteligência Artificial.

Por fim, um medicamento é prescrito conforme a condição do paciente e, em sistemas computacionais, o mesmo também é válido. Dependendo da necessidade, a prescrição se refere aos métodos da Engenharia de *Software*, ou se refere aos da Inteligência Artificial. Em muitos casos, contudo, ambos são necessários, são complementares, devem coexistir no mesmo sistema computacional e não competem entre si.

3.2.1 Inteligência Artificial na Saúde

As aplicações da Inteligência Artificial na saúde são numerosas e os resultados positivos desses usos, uma perspectiva histórica, e o promissor futuro próximo são apresentados em Topol (2019)[28]. Essa aplicação também é conhecida por *deep medicine*, uma referência ao termo *deep learning*. O termo *deep learning* é parte de uma família de métodos mais ampla, conhecida por *machine learning* que, por sua vez, faz parte do escopo da Inteligência Artificial.

É necessária toda uma referência, como aquela citada anteriormente, para obter uma noção das possibilidades de uso da Inteligência Artificial no contexto da saúde. As pesquisas, em particular, têm crescido de forma significativa em praticamente todas as especialidades, seja na

dermatologia e na radiologia, onde imagens são abundantes, até no diagnóstico de doenças e na interpretação de registros eletrônicos, dentre muitas outras.

3.3 Computação em Nuvem e *Software* como Serviço

Computação em nuvem é a oferta sob demanda de recursos computacionais. Isso significa que não é necessário adquirir *hardware*, mas "usar" o que for necessário, que flutua ao longo do tempo, e pagar apenas pelo que é usado. Por recurso computacional entenda principalmente tanto capacidade de processamento quanto de armazenamento.

O cenário oposto é denominado de *on premise*. Nesse, o *hardware* é estimado e adquirido. Se a capacidade precisa ser estendida, então, todo um processo de aquisição é realizado e o *hardware* adquirido é acrescido ao que já estava disponível. Naturalmente, esse processo é lento e não contempla adequadamente flutuações, ao contrário da computação em nuvem.

Embora a elasticidade, que é a capacidade de atender flutuações de demanda, seja uma das principais características, há outras. Disponibilidade é outro atributo desejado para manter o funcionamento de um sistema, mesmo na presença de inevitáveis falhas.

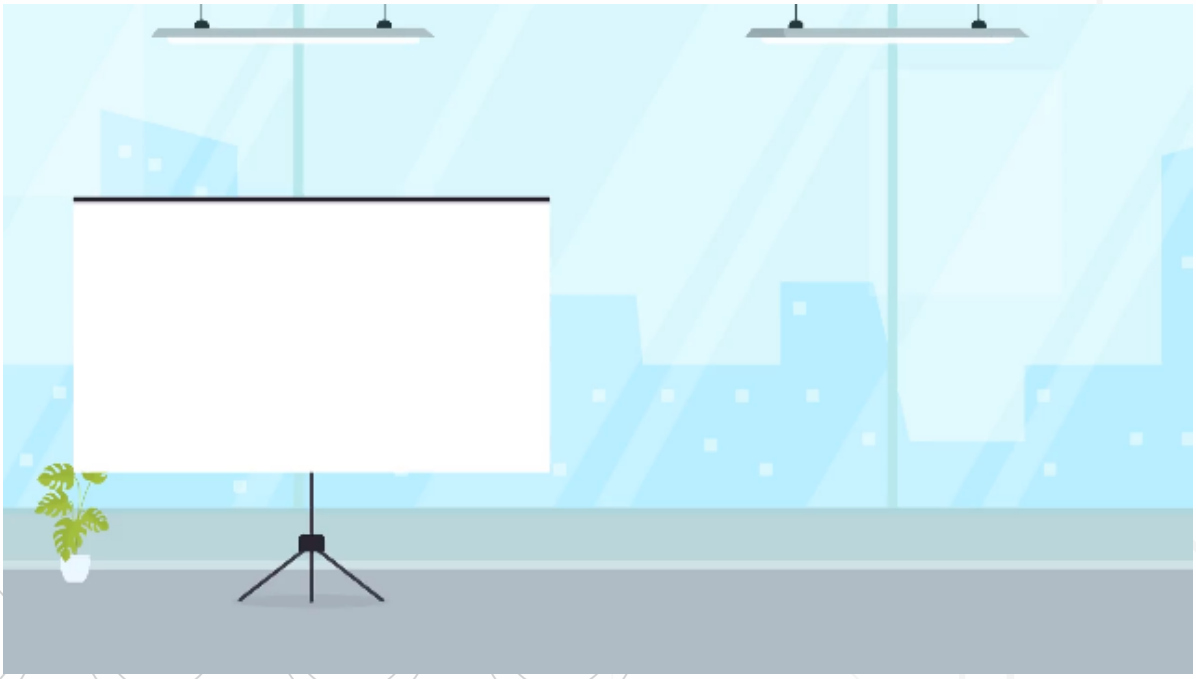
Dentre os fornecedores de computação em nuvem encontram-se gigantes como Amazon®, Microsoft® e Google®. Eles não se limitam a fornecer *hardware* (infraestrutura), mas também serviços como o envio de mensagens, hospedagem de páginas na *web* e muitos outros. Além de um vasto conjunto de serviços "comuns", também há aqueles específicos, como serviços baseados no padrão FHIR (*Fast Healthcare Interoperability Resources*). FHIR é um padrão proposto para troca de informação em saúde, o mesmo padrão adotado pelo Brasil para integração com a RNDS. Esses três fornecedores, respectivamente, oferecem os seguintes serviços baseados no FHIR: *FHIR Works on AWS*, *Azure API for FHIR* e *Cloud Healthcare API*.

Quando o que é oferecido é basicamente o acesso a computadores (capacidade de processamento), tem-se o que é denominado de Infraestrutura como Serviço (*Infrastructure as a Service* [IaaS]). Quando não apenas o *hardware* é transparentemente oferecido para o consumidor, mas tudo o que é necessário para oferecer serviços como os citados no parágrafo anterior, tem-se *Software* como Serviço (*Software as a Service* [SaaS]). Um serviço que permite reunir participantes em uma videoconferência como Google Meet® ou Microsoft Teams® é um exemplo de SaaS. Há outros modelos de serviços, contudo, o relevante aqui é reconhecer que fornecedores de computação em nuvem oferecem, em geral, bem mais do que apenas *hardware* sob demanda.

Cenário 1 - Telessaúde? Na saúde, a avaliação de risco de um paciente, por exemplo, faz uso de informações de um domínio extenso. À semelhança do cenário anterior, não parece exequível estabelecer um conjunto de regras precisas que, quando executadas, resultem no risco desejado.

Assista abaixo uma animação proposta para os cenários apresentados.

Vídeo 3 - Videoaula: animação dos cenários que ilustram a telessaúde, inteligência artificial, dentre outros



Para relembrar...

O que é heurística?

RESPOSTA

Engenharia de Software é a área que lida com a produção e evolução de software?

RESPOSTA

Inteligência artificial é a área que reúne métodos para resolução de problemas complexos?

RESPOSTA

Há conflito entre Inteligência Artificial e Engenharia de Software?

RESPOSTA

EDUCAÇÃO E CAPACITAÇÃO
DE RECURSOS HUMANOS
EM SAÚDE DIGITAL

Pensamento computacional

Unidade 4
Encerramento do microcurso

Fábio Nogueira de Lucena
Sérgio Teixeira de Carvalho

Unidade 4: Encerramento do Microcurso

O emprego de sistemas computacionais tem se expandido pela sociedade, resultando na sociedade digital. Para melhor compreender esse cenário, do qual fazem parte tecnologias como Inteligência Artificial, robótica, *internet* das coisas, geração de dados (*big data*), entre outras, é útil se apropriar do pensar computacionalmente. Essa compreensão empodera o indivíduo a avaliar o impacto desses sistemas computacionais em sua própria vida e, naturalmente, até contribuir com a construção deles.

Elementos básicos da computação (conceitos de *hardware*, *software* e bases da área de Ciência da Computação) somados aos mecanismos e funcionamento desses elementos e à forma de construí-los por meio das técnicas, métodos e ferramentas da Engenharia de *Software* constituíram os fundamentos desse Microcurso.

O objetivo foi apresentar as bases do pensamento computacional, visando inserir o leitor em um contexto, não apenas de consumidores, mas, sobretudo, de potenciais atores ativos na produção de conteúdos e tecnologias, em especial, no que se refere à Saúde Digital no Brasil.

Esperamos que a conclusão desse Microcurso coincida com a aquisição de um alicerce baseado no conteúdo apresentado, um alicerce que consideramos útil para a compreensão das iniciativas de Saúde Digital no Brasil. O próximo passo é aprofundar nas ações de implementação da ESD28 a saber, o "**Sistema de Saúde Brasileiro**".

Até lá!

Referências

1. STATCOUNTER. **Desktop vs Mobile vs Tablet Market Share Worldwide - May 2020 - May 2021** [Internet]. Acesso em 15 jun. 2021. Disponível em: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet.

2. BRASIL. **Lei Nº 9.609, de 19 de fevereiro de 1998. Dispõe sobre a proteção da propriedade intelectual de programa de computador, sua comercialização no País, e dá outras providências.** Acesso em 20 maio 2021. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/l9609.htm.

3. MUSEU DO COMPUTADOR. **A história do computador** [Internet]. Acesso em 20 maio 2021. Disponível em: https://museudocomputador.org.br/.

4. ETHW. **Main page** [Internet]. Acesso em 15 abr. 2021. Disponível em: https://ethw.org/

5. WIKIPEDIA. **Supercomputer** [Internet]. Acesso em 15 abr. 2021. Disponível em: https://en.wikipedia.org/wiki/Supercomputer.

6. WIKIPEDIA. **Moore's law** [Internet]. Acesso em 15 abr. 2021. Disponível em: https://en.wikipedia.org/wiki/Moore%27s_law.

7. DENNING, P. J.; TEDRE, M. **Computational thinking**. Cambridge: The MIT Press, 2019. 264 pp.

8. INFORMATION IS BEAUTIFUL. **Tech and Digital** [Internet]. Acesso em 10 maio 2021. Disponível em: https://informationisbeautiful.net.

9. WIKIPEDIA. **List of longest novels** [Internet]. Acesso em 10 abr. 2021. Disponível em: https://en.wikipedia.org/wiki/List_of_longest_novels.

10. PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de *Software*: uma abordagem profissional**. Trad. ARAKAKI, R.; ARAKAKIi, J.; ANDRADE; R. M. Porto Alegre: AMGH, 9. ed., 2016. 968 pp.

11. BROOKS-JR, F. P. No silver bullet: essence and accidents of Software Engineering. **IEEE Computer**, v. 20, n. 4, p. 10-19, 1987. Acesso em 16 jun. de 2021. Disponível em: http://www.cs.unc.edu/techreports/86-020.pdf.

12. MCCONNELL, S. **Code complete**. 2. ed., Microsoft, 2004. Acesso em 16 jun. de 2021. 952 pp. Disponível em http://aroma.vn/web/wp-content/uploads/2016/11/code-complete-2nd-edition-v413hav.pdf.

13. FORD, N.; PARSONS, R.; KUA, P.; **Building evolutionary architectures: support constant change**. Sebastopol: O'Reilly Media, Inc., 2017. 217 pp. Acesso em 16 jun. de 2021.

Disponível em: https://dokumen.pub/building-evolutionary-architectures-support-constant-change-1nbsped-9781491986363.html.

14. BROOKS-JR, F. P. **The mythical man-month: essays on Software Engineering.** Anniversary Edition, Addison-Wesley, 1995. Acesso em: 16 jun. de 2021. Disponível em: http://www.cesarkallas.net/arquivos/livros/informatica/Addison.Wesley.The.Mythical.Man-Month.Essays.on.Software.Engineering.20th.Anniversary.Edition.pdf.

15. MEL CONWAY'S HOME PAGE. **Conway's Law**. Acesso em 20 jun. 2021. Disponível em: http://www.melconway.com/Home/Conways_Law.html

16. CALVO, R. A.; PETERS, D. P**ositive computing: technology for wellbeing and human potential**. Cambridge: MIT Press, 2017.

17. WIKIPEDIA. **Computer rage** [Internet]. Acesso em 20 maio 2021. Disponível em: https://en.wikipedia.org/wiki/Computer_rage.

18. TUKEY, J. W. The teaching of concrete mathematics. **The American Mathematical Monthly.** 1958, v. 65, n. 1, pp. 1–9. Acesso em 23 jun. 2021. DOI: 10.2307/2310294.

19. WIKIPEDIA. **Margaret Hamilton (software engineer)** [Internet]. Acesso em 15 jun. 2021. Disponível em: https://en.wikipedia.org/wiki/Margaret_Hamilton_(software_engineer)

20. WIKIPEDIA. **History of software engineering** [Internet]. Acesso em 15 jun. 2021. Disponível em: https://en.wikipedia.org/wiki/History_of_software_engineering

21. WIKIPEDIA. **Turing Award** [Internet]. Acesso em 15 jun. 2021. Disponível em: https://en.wikipedia.org/wiki/Turing_Award

22. WIKIPEDIA. **No silver bullet** [Internet]. Acesso em 15 jun. 2021. Disponível em: https://en.wikipedia.org/wiki/No_Silver_Bullet

23. Beattie, D. **The art of code** [Video]. Acesso em 15 jun. 2021. Disponível em: https://youtu.be/6avJHaC3C2U.

24. WHO. **WHO guidelines: recommendations on digital interventions for health system strengthening.** Geneva: WHO, 2019. 124 pp. Acesso em 4 maio 2021. Disponível em: https://www.ncbi.nlm.nih.gov/books/NBK541902/pdf/Bookshelf_NBK541902.pdf.

25. **Journal of Health Informatics**. ISSN 2175-4411. Acesso em: 4 jun. 2021. Disponível em: http://www.jhi-sbis.saude.ws/ojs-jhi/index.php/jhi-sbis

26. SBIS. Sociedade Brasileira de Informática em Saúde. **Conheça a SBIS**. Acesso em: 5 jun. 2021. Disponível em: http://sbis.org.br/.

27. BRASIL. MINISTÉRIO DA SAÚDE. SECRETARIA-EXECUTIVA. DEPARTAMENTO DE INFORMÁTICA DO SUS. **Estratégia de Saúde Digital para o Brasil 2020-2028**. Brasília: Ministério da Saúde, 2020. 128 pp. Acesso em 5 maio 2021. Disponível em: http://bvsms.saude.gov.br/bvs/publicacoes/estrategia_saude_digital_Brasil.pdf.

28. TOPOL, E. Deep medicine: how artificial intelligence can make healthcare human again. New York: Basic Books, 2019. 400 pp.

Saiba mais...

BRASIL. MINISTÉRIO DA SAÚDE. SECRETARIA-EXECUTIVA. DEPARTAMENTO DE INFORMÁTICA DO SUS. **Relatório final do projeto piloto Conecte SUS: análise dos avanços obtidos entre outubro/2019 e junho/2020**. Brasília: Ministério da Saúde, 2020. 59 pp. Acesso em 5 jun. 2021. Disponível em: https://bvsms.saude.gov.br/bvs/publicacoes/relatorio_projeto_piloto_conectesus_outubro.pdf.

COUGHLIN, S. S.; STEWART, J. Use of consumer wearable devices to promote physical activity: a review of health intervention studies. **Journal of Environment and Health Sciences.** 2016, v. 2, n. 6. DOI: 10.15436/2378-6841.16.1123.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Pesquisa Nacional por Amostra de Domicílios Contínua 2018/2019: acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal 2019** [Internet]. 2021. Acesso em 5 jun. 2021. Disponível em: https://biblioteca.ibge.gov.br/visualizacao/livros/liv101794_informativo.pdf.

WAZLAWICK, R. S. **História da computação**, São Paulo: GEN LTC, 2016. 584 pp.

Minibiografias

Organizadores

Ana Laura de Sene Amâncio Zara é graduada em Farmácia e em Análises Clínicas (UFMT), especialista em Avaliação de Tecnologias em Saúde (UFRGS) e em Docência do Ensino Superior (UCDB). Possui mestrado e doutorado em Epidemiologia pelo Programa de Pós-Graduação em Medicina Tropical e Saúde Pública (UFG) e pós-doutorado pelo Programa de Pós-graduação de Odontologia da Faculdade de Odontologia (UFG). Atualmente, é professora do Departamento de Saúde Coletiva da UFG. Ensina, pesquisa e orienta nas áreas de Epidemiologia, Saúde Coletiva, Metodologia e Editoração Científicas, Economia da Saúde, Bioestatística, Informática em Saúde e Revisões Sistemáticas.

E-mail: analauraufg@gmail.com

Fábio Nogueira de Lucena é graduado em Ciência da Computação (UFG), mestre e doutor em Ciência da Computação (UNICAMP), especialista em Informática em Saúde (UNIFESP), Project Management Professional (PMI) e Certified Software Development Professional (IEEE), além de possuir outras certificações da indústria de software. É professor titular do curso de Engenharia de Software do Instituto de Informática da UFG.

Github: https://github.com/kyriosdata

E-mail: kyriosdata@ufg.br

Rejane Faria Ribeiro-Rotta é graduada em Odontologia (UFG), especialista em Radiologia Bucomaxilofacial e Estomatologia, mestre e doutora em Odontologia (Diagnóstico Bucal) (USP-Bauru), com experiência em colaborações internacionais em pesquisa e intercâmbios, e na gestão institucional do ensino superior. Professora titular da Faculdade de Odontologia da UFG. Fundadora do Centro Goiano de Doenças da Boca da Faculdade de Odontologia da UFG (CGDB-FO-UFG) e da Comissão de Governança da Informação em Saúde da UFG. Principais temáticas de pesquisa: Diagnóstico de lesões da região bucomaxilofacial / Câncer de boca; Dores crônicas orofaciais; Diagnóstico por imagem da região bucomaxilofacial; Prática baseada em evidência, Informação e Informática em saúde.

E-mail: rejanefrr@ufg.br

Renata Dutra Braga é professora adjunta do Instituto de Informática da Universidade Federal de Goiás (UFG). É mestre e doutora em Ciências da Saúde pela Faculdade de Medicina da UFG, pós-graduada em Informática em Saúde (UNIFESP) e em Qualidade e Gestão de Software (PUC-GO) e é graduada em Sistemas de Informação (UniEvangélica). É atualmente vice-coordenadora da Comissão de Governança da Informação em Saúde (CGIS-UFG). Ensina, pesquisa, orienta e desenvolve projetos de extensão na área de saúde digital, com interesse, principalmente em modelagem de processos de negócios, engenharia de requisitos, modelos de informação, terminologias clínicas e padrões para a troca da informação em saúde.

E-mail: renatadbraga@ufg.br

Rita Goreti Amaral é professora titular da Faculdade de Farmácia da Universidade Federal de Goiás (UFG), com atuação na graduação e pós-graduação. Graduada em Farmácia e Bioquímica e especialista em Citologia Clínica (UFG). Mestre em Biologia Celular e Molecular (USP) e Doutora em tocoginecologia pela Faculdade de Ciências Médicas (UNICAMP). Coordenadora do Laboratório de Monitoramento Externa da Qualidade da Faculdade de Farmácia (UFG). Desenvolve projetos de pesquisa e extensão na área de Citologia Clínica e Saúde Pública, atuando nos seguintes temas: controle da qualidade em citopatologia do colo do útero, prevenção, detecção precoce de doenças, aperfeiçoamento de métodos diagnósticos, desenvolvimento e validação de práticas de cuidado do paciente nas doenças crônicas transmissíveis e não transmissíveis, informática em saúde e assistência farmacêutica.

E-mail: rita@ufg.br

Sheila Mara Pedrosa é graduada e mestre em Enfermagem pela Faculdade de Enfermagem (UFG), especialista em Saúde Coletiva e Regulação em Saúde no SUS (IEP/HSL) e doutora em Ciências da Saúde pela Faculdade de Medicina (UFG). Atualmente é professora adjunta do Centro Universitário de Anápolis e desenvolve pesquisa e extensão no âmbito das violências e vulnerabilidade social. É membro da Comissão de Governança da Informação em Saúde (CGIS-UFG) e participa de projetos voltados à saúde digital.

E-mail: sheilaenf@gmail.com

Silvana de Lima Vieira dos Santos - é enfermeira, mestre e doutora em Ciências da Saúde (UFG), Especialista em Enfermagem em Infectologia (USP) e em Informática em Saúde (UNIFESP). É professora associada da Faculdade de Enfermagem (UFG). Vice líder do Núcleo de Estudos e Pesquisa de Enfermagem em Prevenção e Controle de Infecção Relacionada à Assistência à Saúde (NEPIH), vinculado ao CNPq. Experiência na área de prevenção e controle de infecções relacionadas à assistência à saúde, epidemiologia e informática em saúde. Coordenadora da Comissão de Governança da Informação em Saúde (CGIS-UFG).

E-mail: silvanalvsantos@ufg.br

Taciana Novo Kudo é professora adjunta do Instituto de Informática da Universidade Federal de Goiás (UFG). É mestre e doutora em Ciência da Computação pelo Departamento de Computação (UFSCar) e graduada em Ciência da Computação (UNIMAR). Possui experiência profissional na área de Engenharia de Software, especificamente em Engenharia de Requisitos e Gerência de Projetos, em institutos de pesquisa e empresas de São Paulo e Goiás. Como pesquisadora, atua em projetos voltados para Engenharia de Software, Engenharia de Requisitos e Informática aplicada à Educação e à Saúde.

E-mail: taciana@ufg.br

PROGRAMA EDUCACIONAL EM SAÚDE DIGITAL DA UNIVERSIDADE FEDERAL DE GOIÁS

CGIS UFG Comissão de Governança da Informação em Saúde

CIGETS CENTRO DE INOVAÇÃO EM GESTÃO DA EDUCAÇÃO E DO TRABALHO EM SAÚDE

DATASUS DEPARTAMENTO DE INFORMÁTICA DO SUS

Ministério da Saúde

SOBRE O E-BOOK

Tipografia: Montserrat

Publicação: Cegraf UFG
Câmpus Samambaia, Goiânia - Goiás. Brasil. CEP 74690-900
Fone: (62) 3521-1358
https://cegraf.ufg.br