

Engenharia de Software

Unidade 01

Prof. Daniel Caixeta





Conteúdo Programático

01. O início

- 1.1. Panoramas e novos cenários.
- 1.2. Comparação entre as curvas de *hardware* e *softwares*.

02. Campos de aplicação [...]

- 2.1. *Softwares* de sistemas.
- 2.2. *Softwares* de aplicação.
- 2.3. *Softwares* científicos e para engenharia.
- 2.4. *Softwares* embutidos/embarcados.
- 2.5. *Softwares* para linhas de produtos.
- 2.6. Aplicações para WEB.
- 2.7. *Software* de Inteligência Artificial (I.A).

03. O legado de gerações anteriores

- 3.1. Resumo [...].
- 3.2. Computação mundial aberta.
- 3.3. *Netsourcing* (Recursos via *internet*).



04. Um breve histórico [...]

4.1. A evolução do *software*.

4.2. Linha histórica.

05. Os conceitos [...]

5.1. O que é Engenharia de *Software*.

06. Os fundamentos da E.S.

6.1. A interdisciplinaridade.

6.1.1. Administração de Projetos.

6.1.2. Comunicação.

6.1.3. Ciência da computação.

6.1.4. Técnicas baseadas em solução de problemas.

07. Mitos *versus* Realidades

7.1. Mitos *versus* Realidades.

Referências

Panoramas e novos cenários [...]

Panoramas e novos cenários [...]

1.1. PANORAMA SEGUNDO PRESSMAN

- Um *software* é um produto desenvolvido por profissionais.
- Possui suporte de longo prazo.
- Abrange programas/aplicativos que são executados em computadores de qualquer porte ou arquitetura, conteúdos, informações descritivas tanto na forma impressa (*hard copy*) como na virtual.



O(a) profissional

Por que são tão importantes?

- *Softwares* são importantes porque estão presentes em quase todos os aspectos de nosso cotidiano e tornou-se pervasivo (incorporado) no comércio, na cultura e em nossas atividades cotidianas.
- A E.S é importante porque nos capacita para o desenvolvimento de sistemas complexos dentro do prazo e com alta qualidade.

Aplica-se [...]

- Em processos adaptáveis e ágeis que conduzam a resultados de qualidade satisfatória, atendendo às necessidades dos clientes.

O que são artefatos?



- Do ponto de vista da engenharia de software, é um conjunto de programas, conteúdo (dados) e outros elementos que agregados darão origem a um programa/aplicativo.
- Porém, do ponto de vista do usuário, artefatos consistem em informações resultantes que, de alguma forma, o orientará no uso do *software*.





- Serviços nacionais e internacionais são controlados por sistemas computacionais, sendo que a maioria dos produtos eletrônicos possui um *hardware* dedicado e um *software* que os controla. *E.g.*, nas indústrias (setor de logística e automação), no campo (agropecuária), nos sistemas financeiros, hospitalar, etc.
- Portanto, não faz sentido procurar métodos ou técnicas universais, porque diferentes tipos de *softwares* exigem diferentes abordagens.
- Para Sommerville (2018), o mundo moderno não poderia existir sem o *software*.



Tecnologia no campo



Sala de controle de
tráfego aéreo

- Para Pressman (2011), a indústria de *software* tem-se tornado fator dominante e determinante nas economias mundiais.
- Equipes de especialistas, cada qual concentrando-se numa parte da tecnologia, têm substituído o papel do programador solitário de antigamente. Ainda assim, as questões levantadas por esse programador solitário continuam as mesmas feitas hoje. Algumas delas:
 - Por que concluir um *software* leva tanto tempo? 
 - Por que os custos de desenvolvimento são tão altos? 
 - Por que não conseguimos encontrar todos os erros antes de entregarmos o *software* aos clientes? 
 - Por que gastamos tanto tempo e esforço mantendo programas existentes? 

1.2. COMPARAÇÃO ENTRE AS CURVAS DE *HARDWARES* E *SOFTWARES*

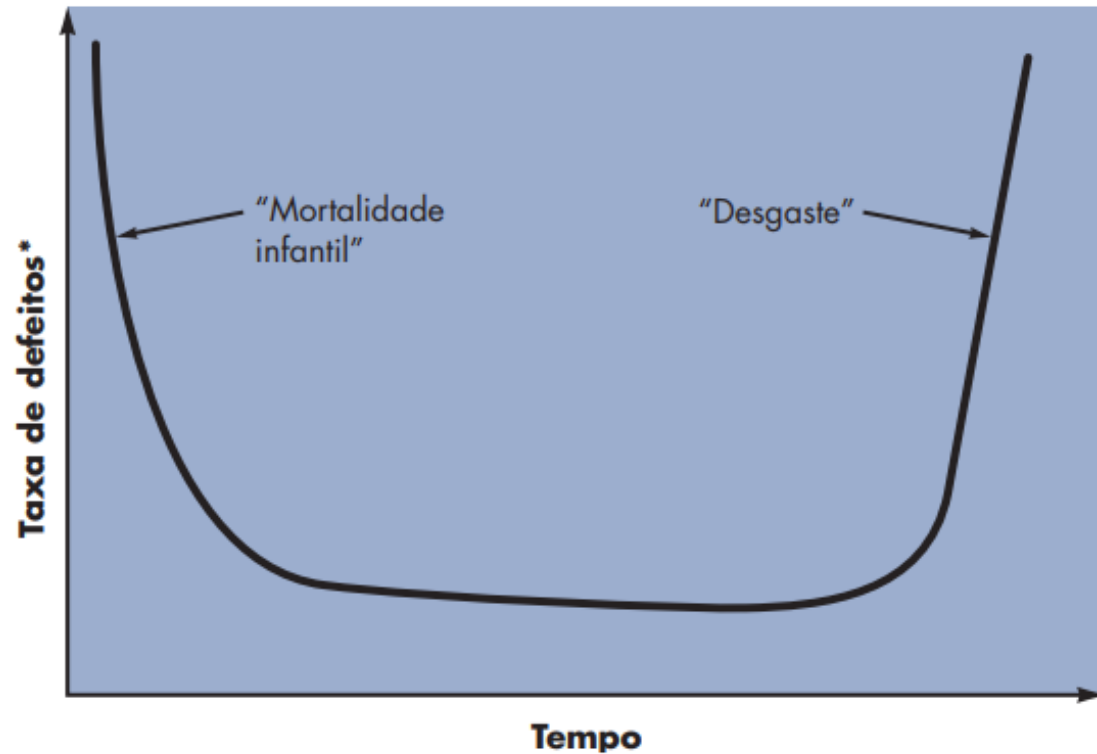


Figura 1. Curva de defeitos para *hardwares*. (Pressman, 2011).

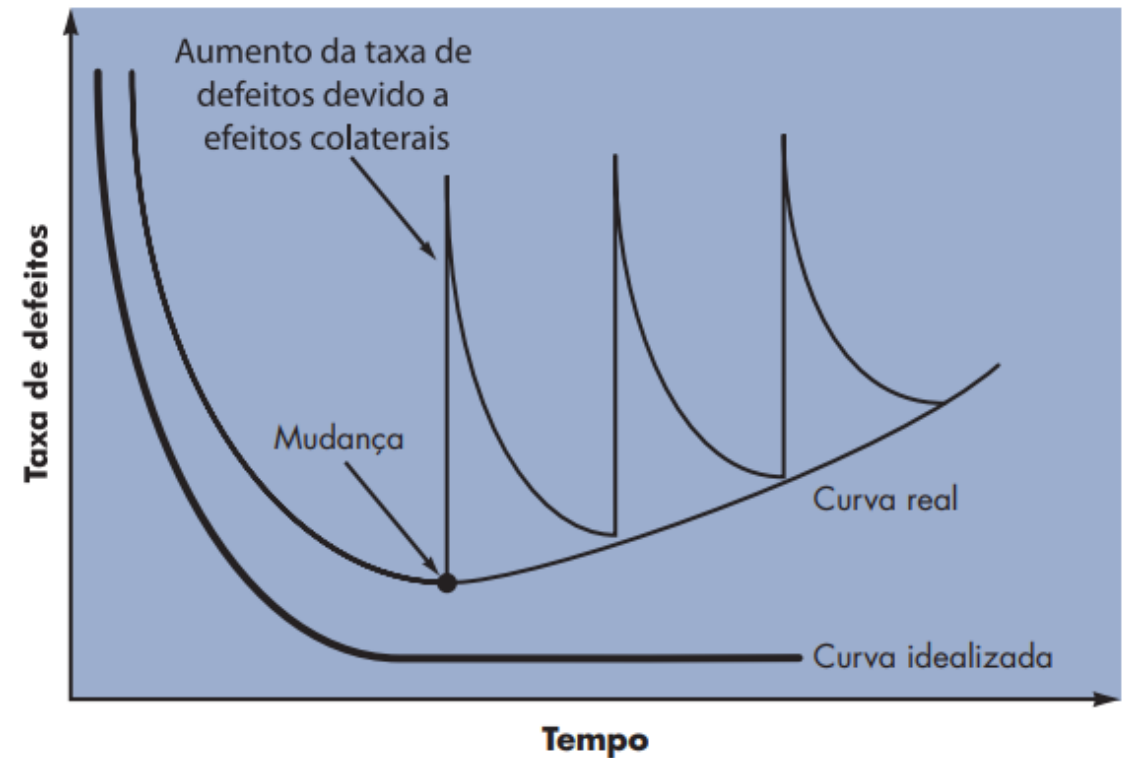



Figura 2. Curva de defeitos para *softwares*. (Pressman, 2011).

The background of the slide is a dark, textured surface. In the center, there is a large, stylized eye. The iris is a glowing blue circle with concentric rings, resembling a digital or optical sensor. The pupil is a solid black circle. The eye is surrounded by various digital elements: binary code (0s and 1s) is scattered throughout, some appearing as if they are floating or falling. There are also snippets of code and text, such as "IMG", "otcer te", "ta potcente", "TR", "TD", "439", "image:", and "1001010101010101". The overall aesthetic is high-tech and futuristic, suggesting a theme of digital vision or artificial intelligence.

Campos de aplicação

Onde atuam [...]

2. CAMPOS DE APLICAÇÃO [...]

- São sete as categorias de *software* segundo Pressman (2011):

2.1. Softwares de sistemas

- Conjunto de programas feito para atender a outros programas. *E.g.*: Compiladores, S.O, *software* de redes, *drivers*, editores, utilitários para gerenciamento de arquivos, etc.



2.2. Softwares de aplicação

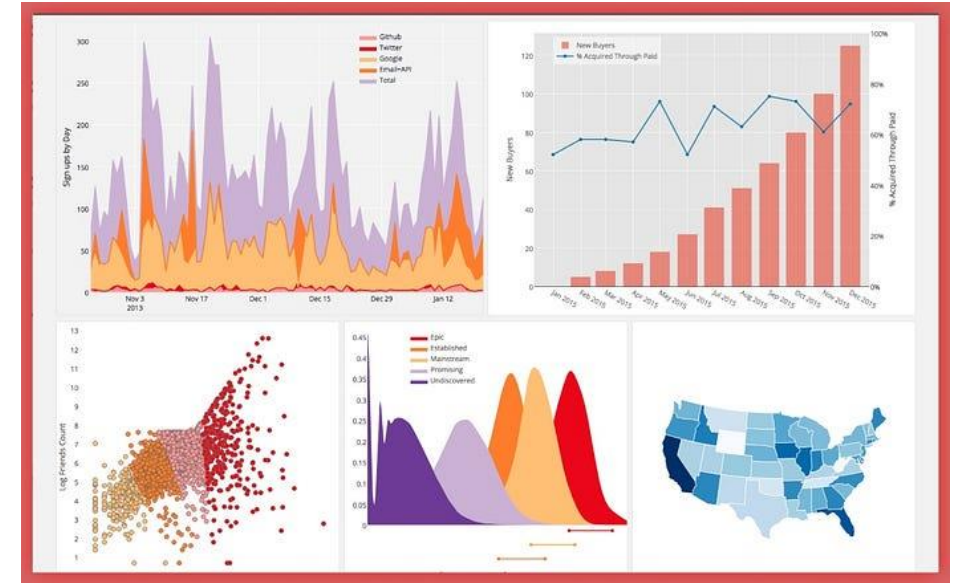
- Aplicações que processam dados responsáveis por tomadas de decisões tanto administrativas como técnicas.
- São usados para controlar funções de negócio em tempo real (e.g., processamento de transações contábeis, controle de processos de fabricação em tempo real, etc.).



Sistema ERP (*Enterprise Resource Planning*)

2.3. Softwares científico e de engenharia

- É caracterizado por algoritmos “*number crunching*” (processamento numérico pesado).
- Atualmente estas aplicações estão se afastando dos algoritmos numéricos convencionais para algoritmos de alta complexidade.



Matplotlib no Python

2.4. Softwares embutidos/embarcados

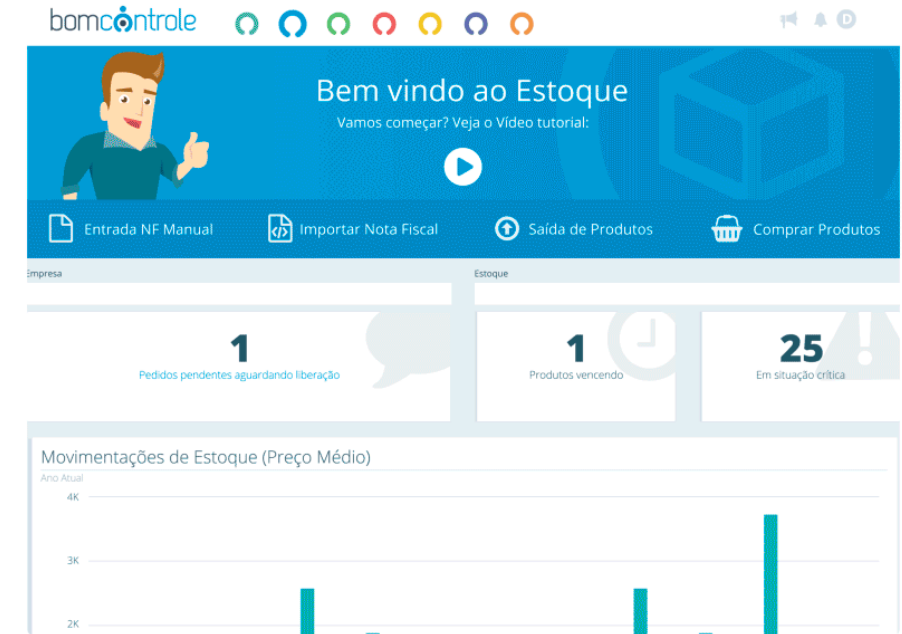
- Residente em um produto ou sistema, é utilizado para implementar e controlar características e funções para o usuário final e para o próprio sistema. *E.g.*, controle do painel de um forno microondas ou função de controle em automóveis (computador de bordo).



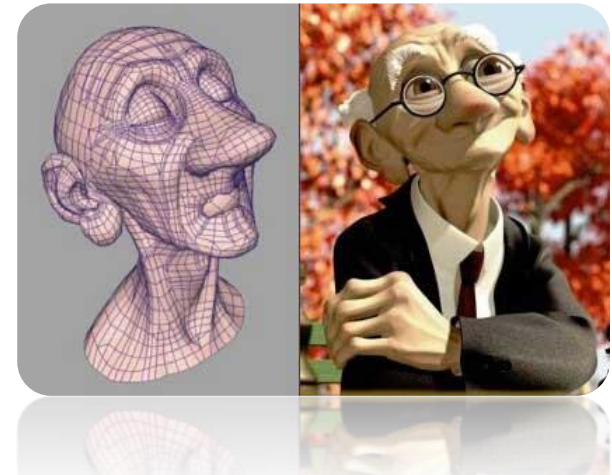
Computador de bordo

2.5. Softwares para linhas de produtos

- Projetado para prover capacidade específica de utilização por muitos clientes diferentes.
- Pode focalizar um mercado limitado e particularizado (e.g., produtos para controle de estoques) ou direcionar-se para mercados de alto consumo (e.g., processamento de texto, planilhas eletrônicas, computação gráfica, multimídia, entretenimento, gerenciamento de bancos de dados e aplicações financeiras pessoais e comerciais).



Sistema de controle financeiro



Computação gráfica

NETFLIX



Spotify

YouTube

2.6. Aplicações para WEB

- Essa categoria de *software* é centralizada em redes com uma vasta gama de aplicações.
- Não fornecem apenas recursos especializados, mas funções computacionais e conteúdos para diversos usuários finais.

2.7. Software de Inteligência Artificial (I.A)

- Faz uso de algoritmos não numéricos para solucionar problemas complexos que não são passíveis de computação ou de análise direta.



Robótica



O futuro dos sistemas de *softwares* [...]

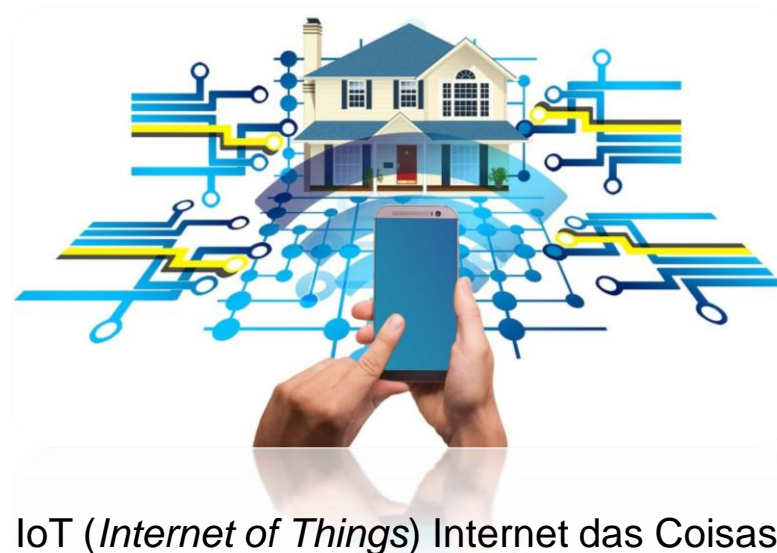
Evolução constante [...]

3.1. UM RESUMO

- Atualmente, milhões de especialistas estão trabalhando em milhares de projetos de *softwares*.
- Em alguns casos, novos sistemas estão sendo construídos, mas em muitos outros, aplicações já existentes estão sendo corrigidas, adaptadas e aperfeiçoadas.
- Não é incomum para um desenvolvedor trabalhar em sistemas de *softwares* que foram desenvolvidos por gerações anteriores.
- Citemos duas tendências [...]

3.2. COMPUTAÇÃO MUNDIAL ABERTA

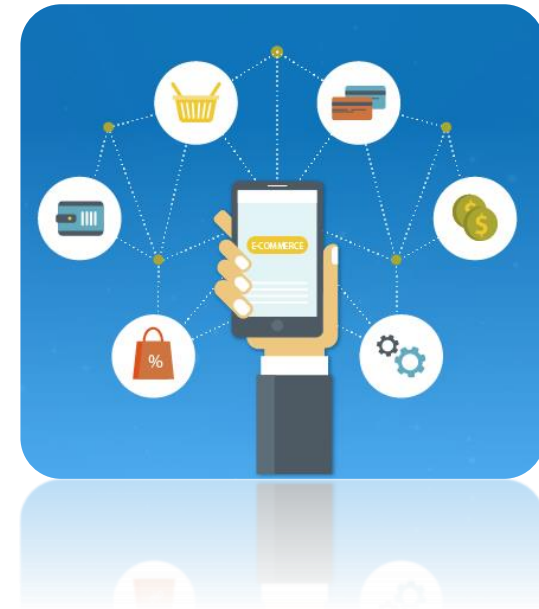
- O rápido crescimento das redes sem fio, tem conduzido os meios de comunicação a uma verdadeira computação distribuída e pervasiva (ampliada, compartilhada e incorporada em ambientes domésticos e comerciais).
- O desafio atual está no desenvolvimento de sistemas de *softwares* que permitam que dispositivos móveis, computadores pessoais e sistemas corporativos se comuniquem através de extensas redes.




IoT (*Internet of Things*) Internet das Coisas.

3.3. NETSOURSING (RECURSOS VIA INTERNET)

- A *internet* tem se tornando, rapidamente, tanto um mecanismo computacional, como um provedor de conteúdo.
- O desafio então consiste em desenvolver aplicações simples e sofisticadas que forneçam benefícios aos usuários finais.

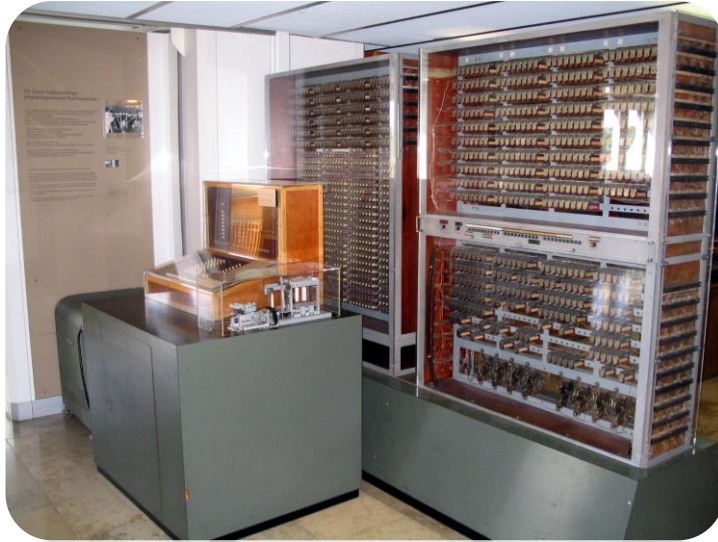


The background of the slide is a dark, textured surface. In the center, there is a large, stylized eye. The iris of the eye is a complex, circular digital pattern with concentric rings of blue and green light, resembling a target or a radar screen. The pupil is a solid black circle. The eye is surrounded by various digital elements: binary code (0s and 1s) is scattered throughout, some appearing as if they are floating or falling. There are also snippets of code and text, such as "IMG", "otcer te", "ta potcente", "TR", "TD", "439", "image:", and "1001010101010101". The overall aesthetic is high-tech and futuristic, suggesting a theme of digital technology and software development.

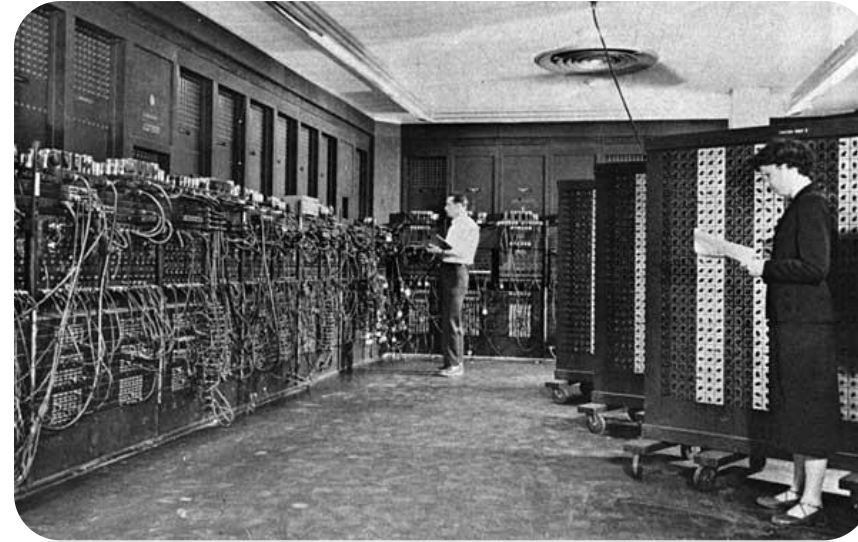
Um breve histórico [...]

A evolução do software.

4.1. A EVOLUÇÃO [...]



Z3 (1941)



ENIAC (Electronic Numerical
Integrator and Computer) (1946)

- Desde os primeiros computadores eletrônicos, como o alemão Z3 e o americano ENIAC, que se questiona a evolução e integração entre *hardware* e *software* em sistemas computacionais.

- Antes, a programação destes *mainframes* eram feitas por equipes técnicas especializadas que se ocupavam diariamente em ligar e desligar centenas de cabos e válvulas, por onde passavam as tomadas de decisões, e após obtenção de resultados eram impressas em cartões perfurados.



Harvard Mark I. Início do desenvolvimento, 1939. Entrou em operação, 1944.

- Hoje, o *software* é composto de instruções escritas em linguagens de programação específicas, armazenadas em memória eletrônica e executadas por um microprocessador, ou chip. Por exemplo:

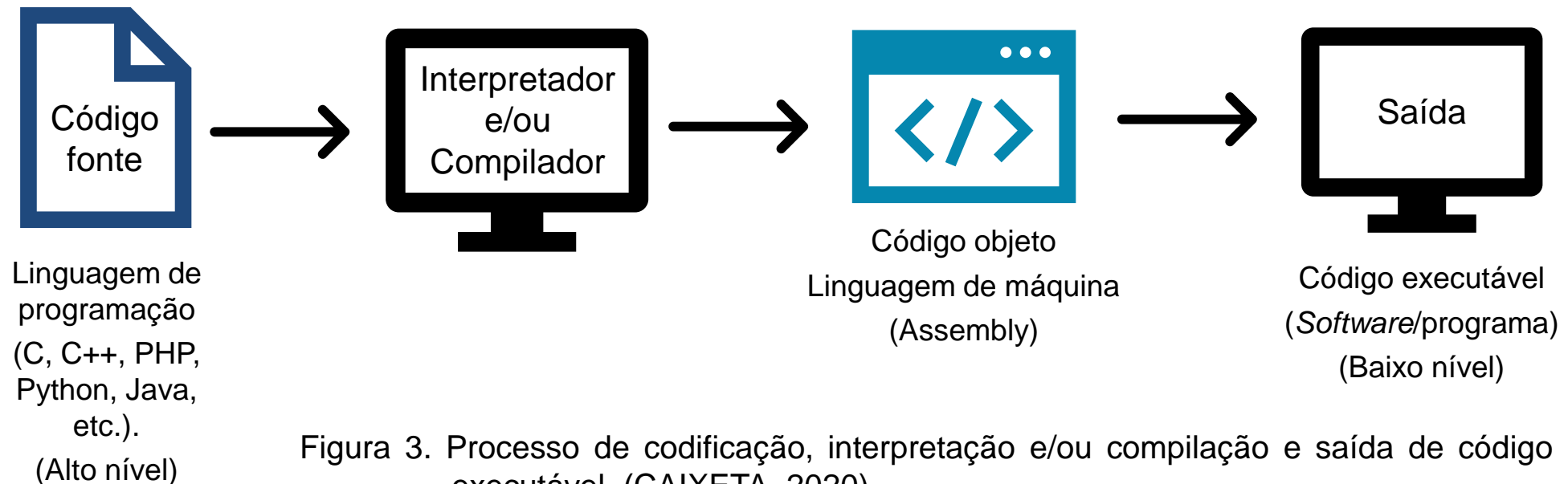


Figura 3. Processo de codificação, interpretação e/ou compilação e saída de código executável. (CAIXETA, 2020).

- Historicamente, o primeiro *software* surgiu na Inglaterra em 1946, baseado na arquitetura de sistema de John von Neumann (matemático, 1903-1957).
- Entretanto, Neumann teve uma grande contribuição. Basta olharmos para ± 100 anos antes, quando uma jovem chamada Ada Lovelace (1815-1852), desenvolveu os primeiros algoritmos de rotinas e subrotinas² para o computador mecânico ou “Máquina Analítica” de Charles Babbage (1791-1871).
- Vejamos então, a evolução do *software* apartir do final da década 1960.

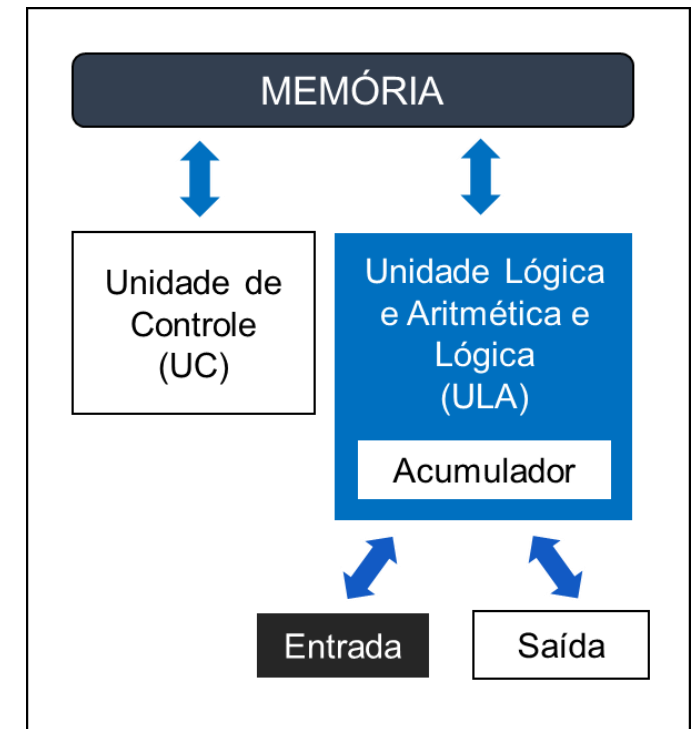
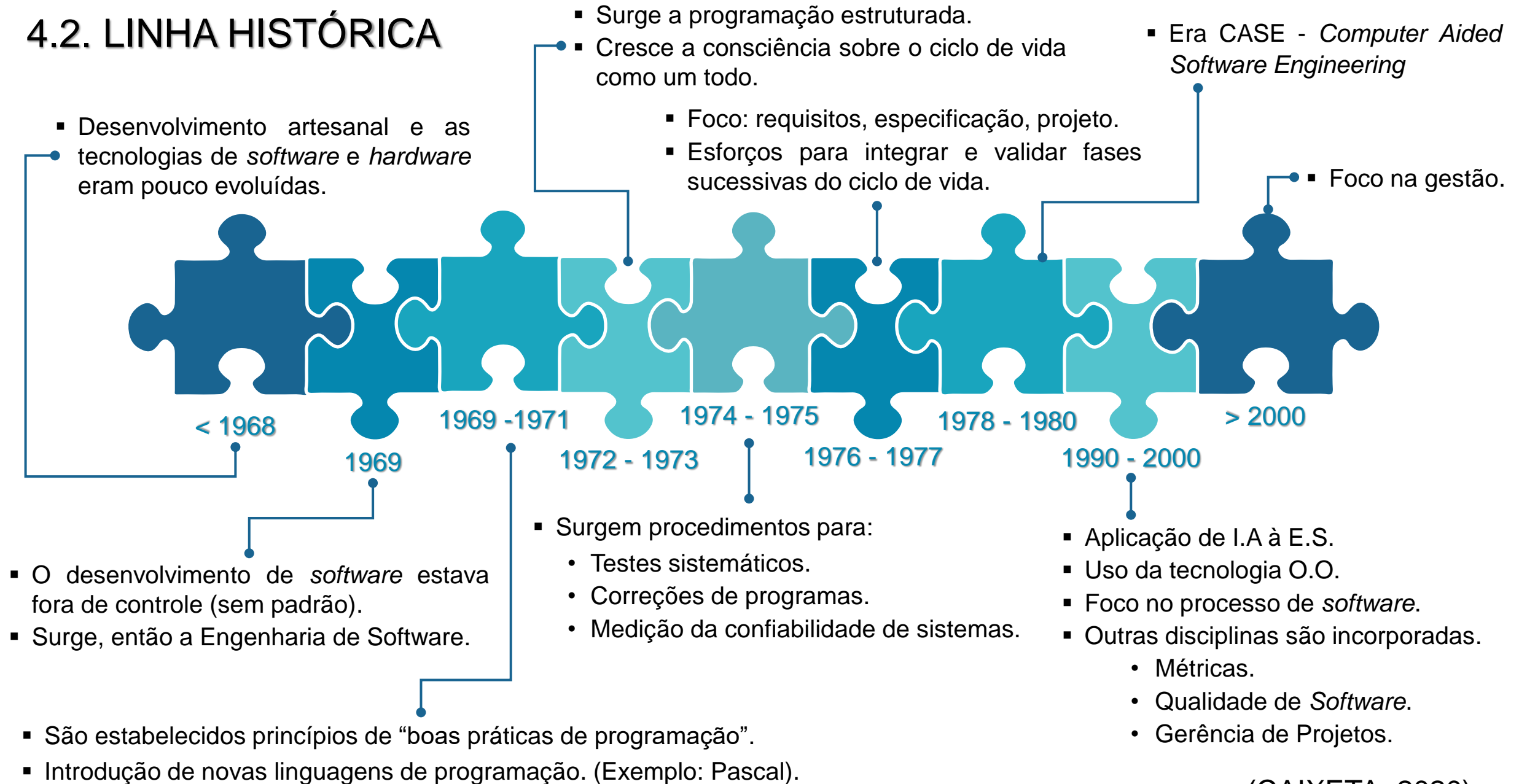


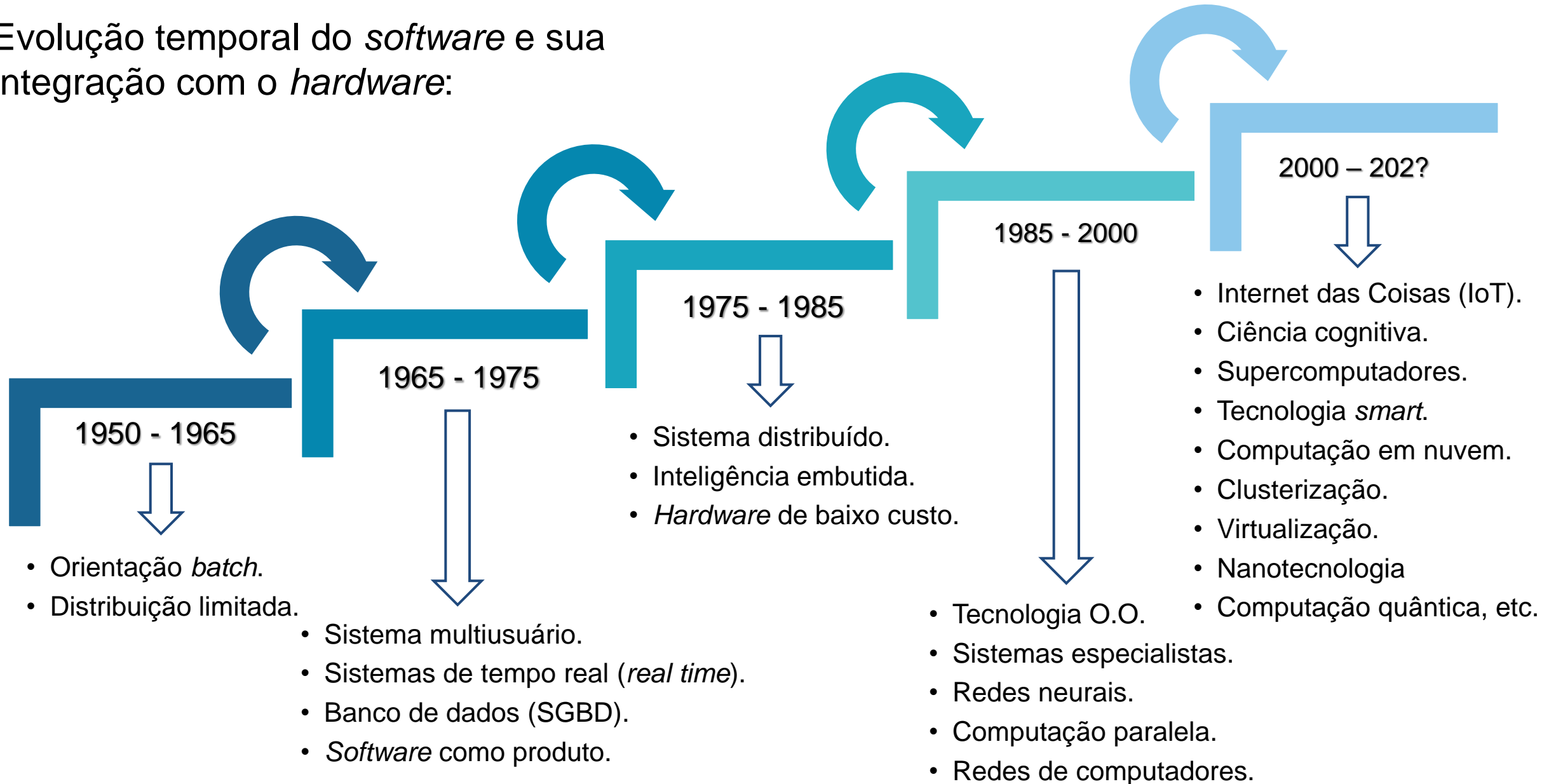
Figura 4. Arquitetura de von Neumann. (CAIXETA, 2018).

4.2. LINHA HISTÓRICA



(CAIXETA, 2020)

Evolução temporal do *software* e sua integração com o *hardware*:



The background image is a dark, textured composition. In the center is a large, stylized eye. The iris is a glowing blue circle with concentric rings and a digital, pixelated appearance. The pupil is a solid black circle. The eye is surrounded by various elements: binary code (0s and 1s) in different colors (blue, green, white) and sizes, some of which are blurred as if they are floating or moving. There are also snippets of code or text, such as 'IMG', 'image:', '18p>', '439"', and '294', scattered around the eye. The overall aesthetic is high-tech and digital, suggesting themes of artificial intelligence, data, or the intersection of the human and digital worlds.

Os conceitos.

Algumas definições [...]

5.1. O QUE É A ENGENHARIA DE SOFTWARE?

Conceito:

- É a disciplina que utiliza um conjunto de métodos, técnicas e ferramentas para analisar, projetar e gerenciar o desenvolvimento e a manutenção de *software*.


Os objetivos:

- Melhorar a qualidade do produto (*software*) e aumentar a produtividade e a satisfação dos profissionais envolvidos no projeto.
- Sintetizar a produção, a manutenção, a evolução e a recuperação do *software*.
- Administrar o processo de desenvolvimento de um produto (*software*) no ciclo: Construção, Implantação e Manutenção.



- Já a definição segundo I3E³, na seção *Glossary of Software Terminology*:

“Enfoque sistemático para o desenvolvimento, operação, manutenção e descontinuação do *software*”.
 - Já para Pressman (2011), destacamos a definição de Fritz Bauer:

É o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter *software* de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais. (NAUR & RANDALL, 1969).
-
-  Portanto, possibilita um processo de desenvolvimento claro, eficiente, visando garantir a produção com qualidade.

- Já para Sommerville (2018, *grifo meu*), Engenharia de Software é:

[...] uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.

- Destaquemos as duas expressões grifadas em Sommerville (*ibidem*):

1. Disciplina de engenharia: Engenheiros aplicam teorias, métodos e ferramentas onde for apropriado. Buscam soluções para problemas, mesmo quando não há teorias e métodos aplicáveis. Reconhecem que devem trabalhar de acordo com as restrições organizacionais e financeiras, portanto, buscam soluções dentro dessas restrições.
2. Todos os aspectos da produção de software. A engenharia de software não se preocupa apenas com os processos técnicos do desenvolvimento de *software*. Ela também inclui atividades como gerenciamento de projeto de *software* e desenvolvimento de ferramentas, métodos e teorias para apoiar a produção de *software*.

- Mas então como criar *software* “economicamente viável” e de modo “confiável”? O que é necessário para desenvolver programas de computador que funcionem “de forma eficiente” não apenas em uma, mas sim em várias e diferentes “máquinas reais”? Essas são as questões que continuam a desafiar os engenheiros de software. (PRESSMAN, 2011).
- Existem inúmeras metodologias utilizadas [...], que vão desde das mais tradicionais às metodologias ágeis. (*ibidem*).
- O que não podemos esquecer, é que por mais que envolvam tecnologias, o fator humano é um dos principais elos entre um produto razoável e um produto excelente. (*ibidem*).
- Habilidades, competências e atitudes são construídas ao longo dos anos, *i.e.*, após inúmeros erros e acertos. Portanto, não há como desconsiderar este fator, que é bastante relevante nesse processo. (*ibidem*).

- Em relação às tecnologias, Pressman (2011) apresenta o modelo de desenvolvimento em camadas. Vejamos.

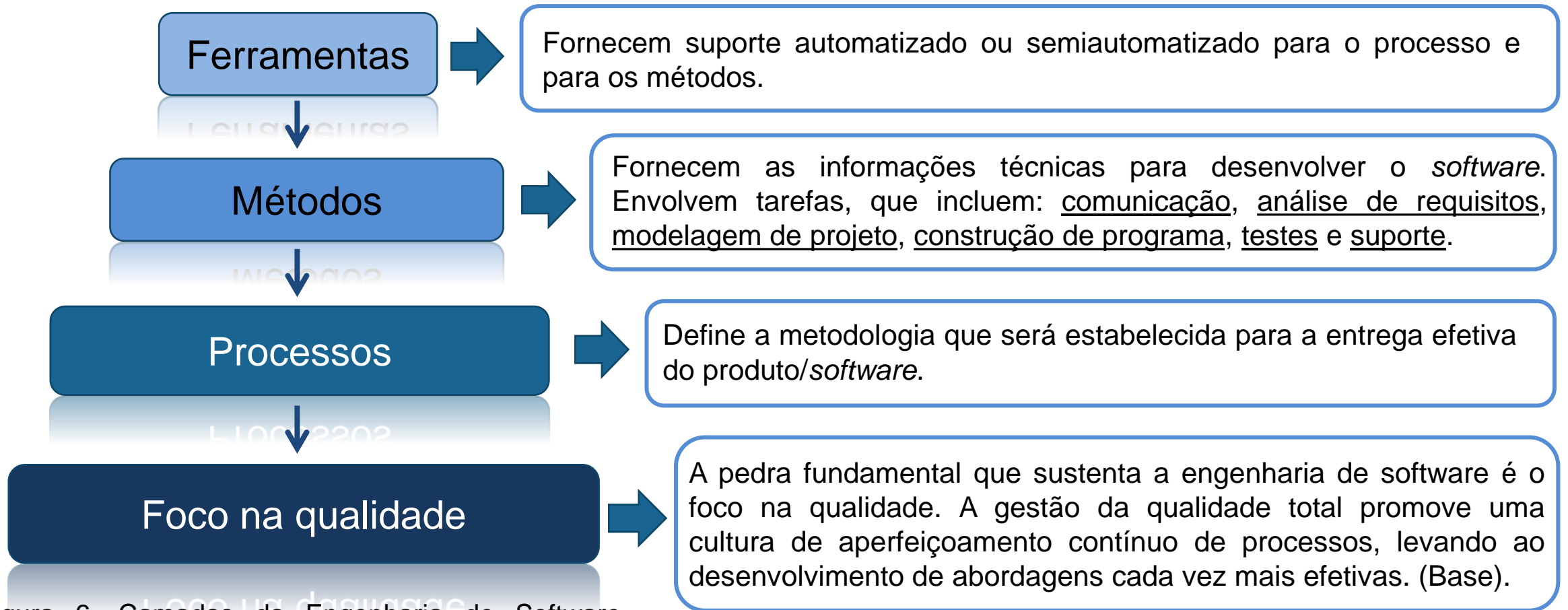


Figura 6. Camadas da Engenharia de Software segundo Pressman (2011 *adaptado*, CAIXETA, 2020)



Os fundamentos [...]

A interdisciplinaridade [...]

6.1. A INTERDISCIPLINARIDADE

- A Engenharia de Software é considerada uma área interdisciplinar, baseada nos fundamentos da:



6.1.1. ADMINISTRAÇÃO DE PROJETOS

- É a disciplina que apresenta os requisitos fundamentais para o gerenciamento de projetos.
- Entre as atividades de gestão podemos citar:

- Planejamento.

- Gestão de recursos.

- Elaboração e acompanhamento de cronogramas de atividades.

- Definição da estrutura organizacional do projeto.

- “Torna o Engenheiro de Software um Gestor de Projetos.”

6.1.2. COMUNICAÇÃO

- A Engenharia de Software supõem um alto nível de interação pessoal.
- É importante que o Engenheiro de Software possua domínio e facilidade de comunicação interpessoal, tanto escrita como oral.
- Relacionada à Gestão de Projetos é importante que possua habilidades de:
 - Resolver problemas interpessoais.
 - Motivar os envolvidos no processo.



6.1.3. CIÊNCIA DA COMPUTAÇÃO

- É a área do conhecimento que atua no desenvolvimento de programas para diferentes dispositivos. Programação, banco de dados e sistemas operacionais.
- Deve possuir domínio sobre as técnicas de desenvolvimento de *softwares*.
- É importante que possua habilidades de:
 - Analisar as necessidades dos usuários/clientes.
 - Desenvolver *softwares* usando linguagens de programação, estruturação e gerenciamento de banco de dados, testes, segurança, etc.
 - Manter e oferecer suporte aos usuários.



6.1.4. TÉCNICA BASEADA EM SOLUÇÃO DE PROBLEMAS

- Considerando o conceito de engenharia, observa-se que a busca por soluções de problemas relacionados ao desenvolvimento de *softwares* é algo natural à esta disciplina.
- O engenheiro de sistema é naturalmente um solucionador de problemas, que se utiliza de técnicas e métodos apropriados para alcançar determinados objetivos.
- Possíveis problemas?

- Falhas no levantamento de requisitos.
- Erros de codificação (*bugs*), erros de tipos (lógicos).
- Falhas em Banco de Dados.
- Falta de seguranças em sistemas e *softwares*. Etc.

The background image is a dark, digital-themed composition. It features a large, stylized eye in the center-left. The iris of the eye is replaced by a glowing blue and green clock face with concentric circles and radial lines. The eye is surrounded by various digital artifacts, including binary code (0s and 1s) in different colors (blue, green, white), glitch lines, and semi-transparent text fragments like "IMG", "image:", "18p", "439", and "294". The overall aesthetic is futuristic and tech-oriented.

Mitos *versus* Realidades

Nem tudo são flores [...]

7.1. MITO *versus* REALIDADE

1. “Minha equipe tem as ferramentas mais atuais de Engenharia de Software e os melhores computadores.”
2. “O problema de atraso no cronograma pode ser resolvido aumentando a equipe.”
3. “Todos os desenvolvedores/programadores são iguais.”
4. “O programa/sistema está 95% pronto.”
5. “Para iniciar a programação basta uma identificação geral dos objetivos. Os detalhes podem ser identificados depois.”
6. “Enquanto não se tem um programa 'rodando' não é possível avaliar a sua qualidade.”
7. “O único produto de um projeto de desenvolvimento de *software* é um programa funcionando.”

REFERÊNCIAS

PRESSMAN, R. S. Engenharia de Software: Uma abordagem profissional. 7ª edição. Dados eletrônicos. Porto Alegre: AMGH-McGrawHill, 2011.

SOMMERVILLE, I. Engenharia de Software. 10ª edição. São Paulo: Pearson Prentice Hall, 2018.



Obrigado!

Engenharia de Software

