Engenharia de Software

Introdução

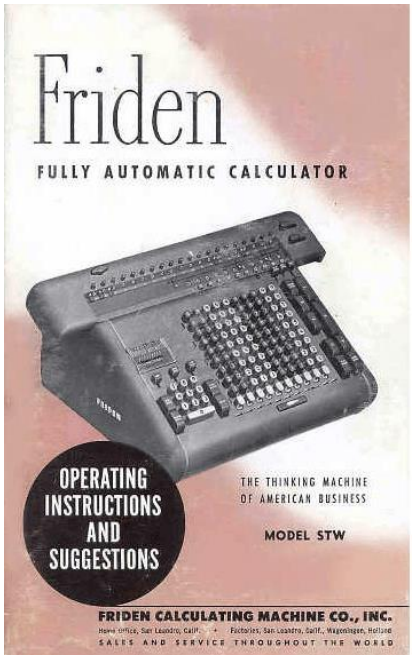
Luís Morgado

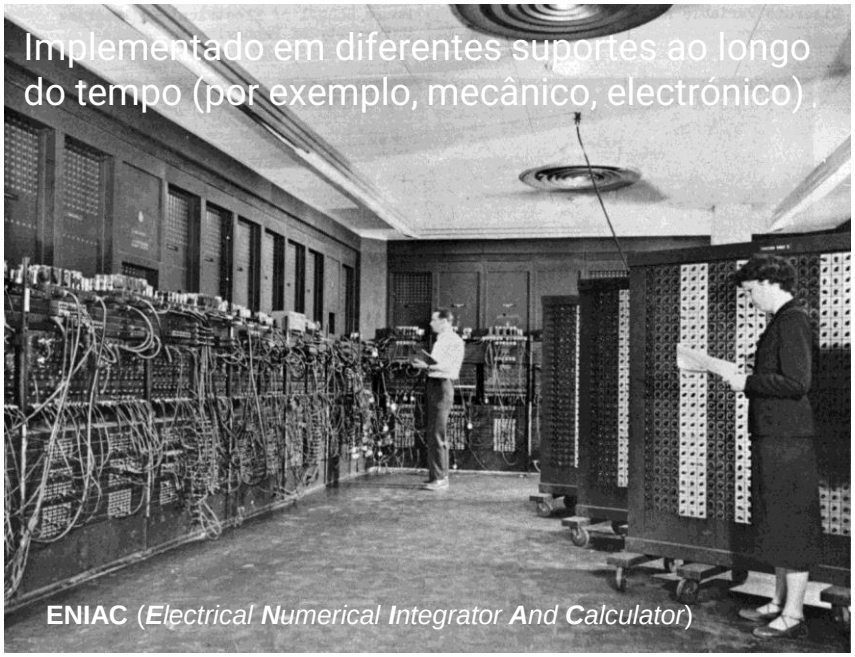
Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

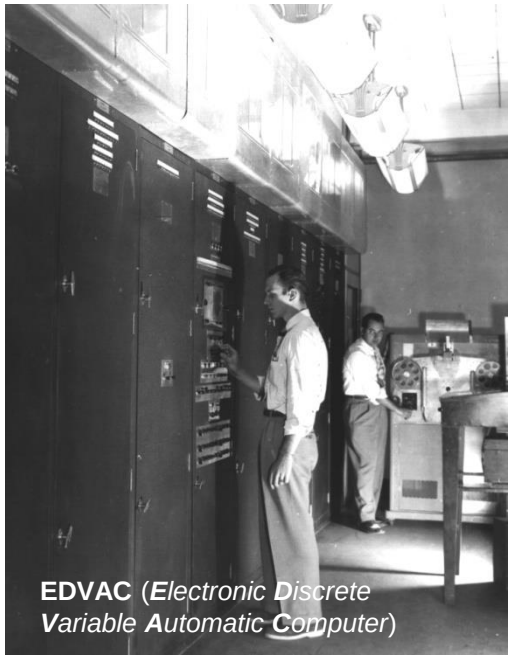
Engenharia de Software

- A *engenharia de software* é uma área de engenharia orientada para a especificação, desenvolvimento e manutenção de software, que tem por objectivo o desenvolvimento, operação e manutenção de software de modo *sistemático* e *quantificável*
- **Sistemático**
  - Significa a capacidade de realizar o desenvolvimento de software de forma organizada e previsível, de modo a garantir a satisfação dos requisitos definidos, incluindo tempo e recursos necessários
- **Quantificável**
  - Significa a capacidade de avaliar os meios envolvidos e os resultados produzidos no desenvolvimento de software, utilizando métodos e processos adequados para garantir a satisfação dos objectivos e a qualidade do software produzido, bem como a monitorização do processo de desenvolvimento
  - O desenvolvimento de software de forma quantificável é importante para garantir que os requisitos do software sejam cumpridos, bem como para prever e gerir os recursos necessários para o desenvolvimento e operação do software produzido

Software







Computador

- Máquina que realiza cálculo automático
- Componente física: *hardware*
- Componente lógica: *software*

Software

- Especificação que determina o cálculo realizado por um computador
- Pode ter diferentes formas, no caso de máquinas capazes de interpretar instruções de uma *linguagem de programação*, consiste num conjunto de instruções e de dados que determinam a operação de um computador

Software

Tendências de evolução:

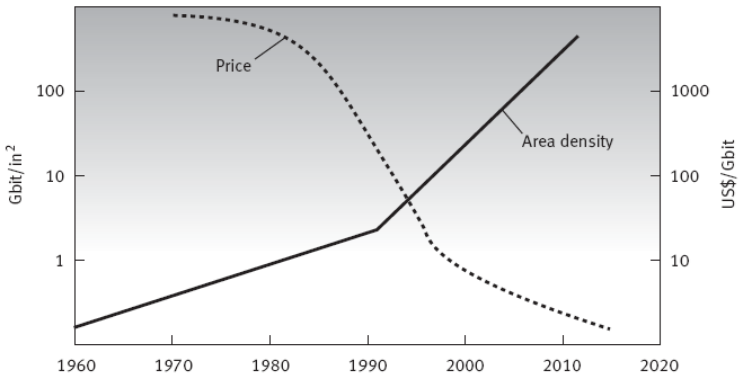
- O software tornou-se essencial em muitos aspectos da sociedade
- As exigências tecnológicas tornaram-se cada vez mais complexas
- A complexidade do software tem crescido de forma exponencial em muitas áreas

Evolução tecnológica

- Miniaturização
- Interligação
- Massificação
- Ubiquidade
- Volatilidade
- Complexidade
- Dependência crítica

Exemplo: Memória secundária

- Redução de custo
- Aumento da capacidade (exponencial)



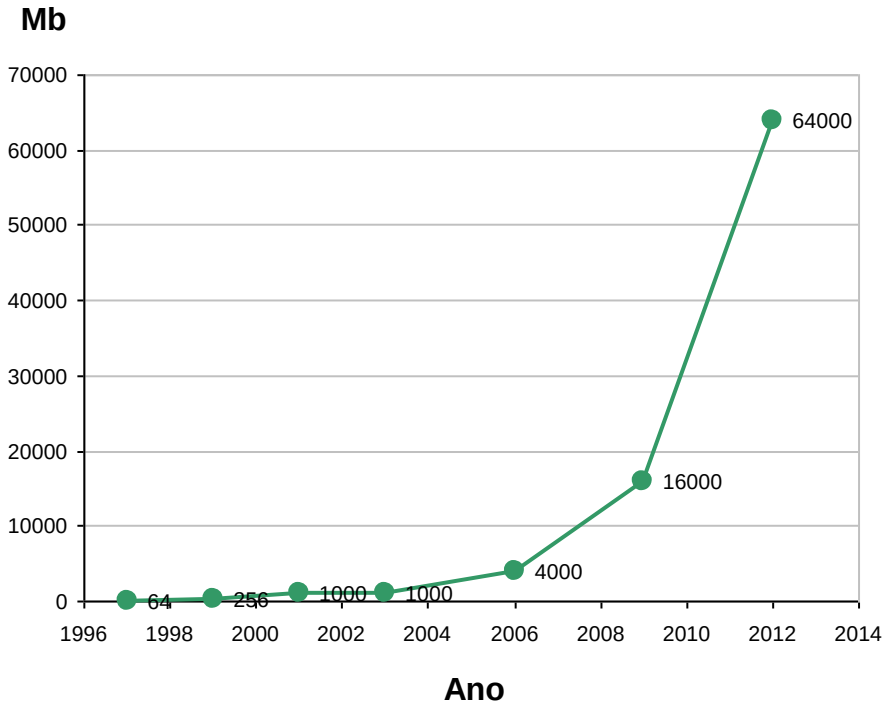
Memória secundária [Endres & Romback, 2003]

Engenharia de Software

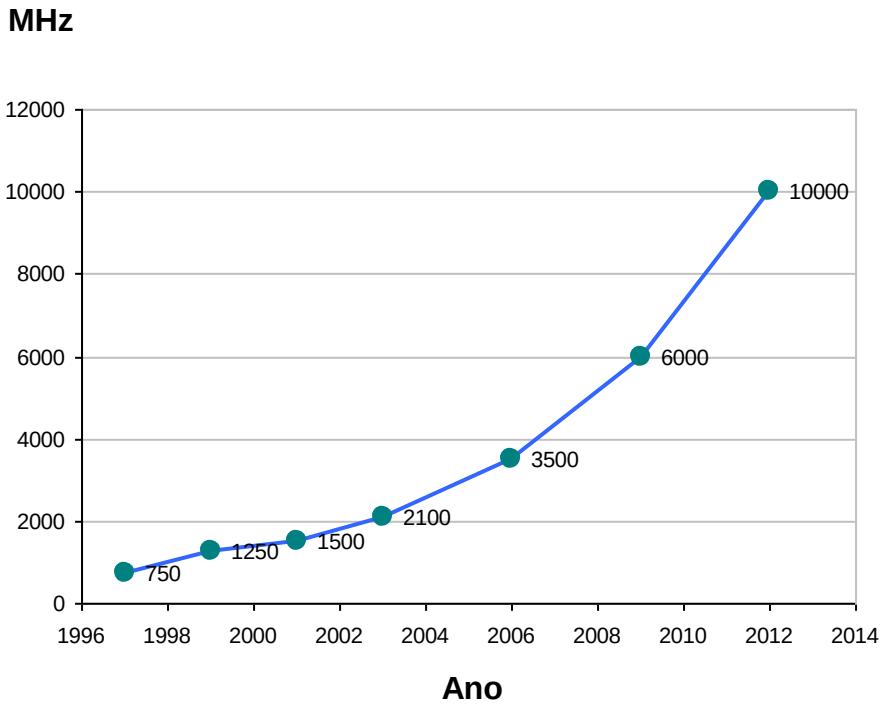
- Com o crescimento da utilização de sistemas baseados em software, que vão desde aplicações informáticas e sistemas de informação, a sistemas de controlo industrial ou de veículos autónomos, dois problemas principais têm relevância crescente no desenvolvimento, operação e manutenção de software, *complexidade* e *mudança*
- **Complexidade**
  - A complexidade é expressa em concreto na crescente dificuldade de desenvolvimento, operação e manutenção de software, na crescente sofisticação do software produzido, bem como na quantidade crescente de recursos computacionais envolvidos, nomeadamente, em termos de capacidade de processamento e de memória utilizada
- **Mudança**
  - A mudança é expressa no ritmo crescente a que o software necessita de ser produzido, ou modificado, para satisfazer as necessidades dos respetivos contextos de utilização, quer a nível de funcionalidades disponibilizadas, quer a nível das tecnologias utilizadas

Evolução Tecnológica

Evolução Exponencial



Evolução da quantidade de memória por circuito integrado [ITRS, 2001]



Evolução da frequência de operação em microprocessadores [ITRS, 2001]

Evolução Exponencial

IBM Real-Time Computer Complex - NASA Manned Spacecraft Center
Década de 1960

Como exemplo de comparação do crescimento da capacidade computacional disponível nos dispositivos actuais, muitos dos telefones móveis de hoje têm mais capacidade computacional que os computadores envolvidos na missão lunar da NASA na década de 1960

Hoje

Exemplo: Indústria Automóvel

SPECTRUM

This Car Runs on Code

By **Robert N. Charette**

IMAGE: DAIMLER

The avionics system in the F-22 Raptor, the current U.S. Air Force frontline jet fighter, consists of about 1.7 million lines of software code. The F-35 Joint Strike Fighter, scheduled to become operational in 2010, will require about 5.7 million lines of code to operate its onboard systems. And Boeing's new 787 Dreamliner, scheduled to be delivered to customers in 2010, requires about 6.5 million lines of software code to operate its avionics and onboard support systems.

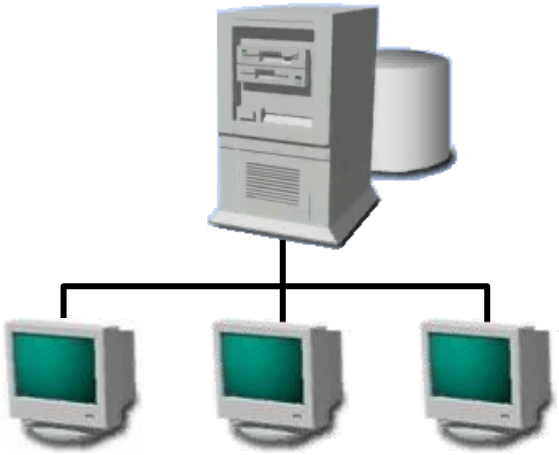
These are impressive amounts of software, yet if you bought a premium-class automobile recently, "it probably contains close to 100 million lines of software code," says Manfred Broy, a professor of informatics at Technical University, Munich, and a leading expert on software in cars. All that software executes on 70 to 100 microprocessor-based electronic control units (ECUs) networked throughout the body of your car.

Evolução de Paradigmas de Computação

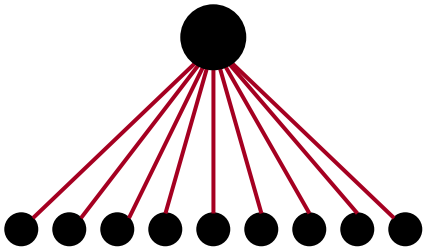
A par com a evolução tecnológica, também os paradigmas de computação evoluíram ao longo do tempo

- ≈ 1970: Computação centralizada com arquitecturas monolíticas, aplicações executadas num servidor (*mainframe*), sendo a interacção com o utilizador realizada através de terminais alfanuméricos
- ≈ 1980: Computação descentralizada, computadores pessoais pouco dispendiosos puderam ser dedicados a utilizadores específicos, computadores menos dispendiosos ao nível departamental
- ≈ 1990: Computação em rede, com aumentos significativos das capacidades de processamento e de armazenamento de dados, a par da utilização em larga escala da Internet, meios de comunicação e processamento heterogéneos, sendo a interoperabilidade um dos principais requisitos
- ≈ 2000: Evolução e massificação da computação em rede

Computação Centralizada



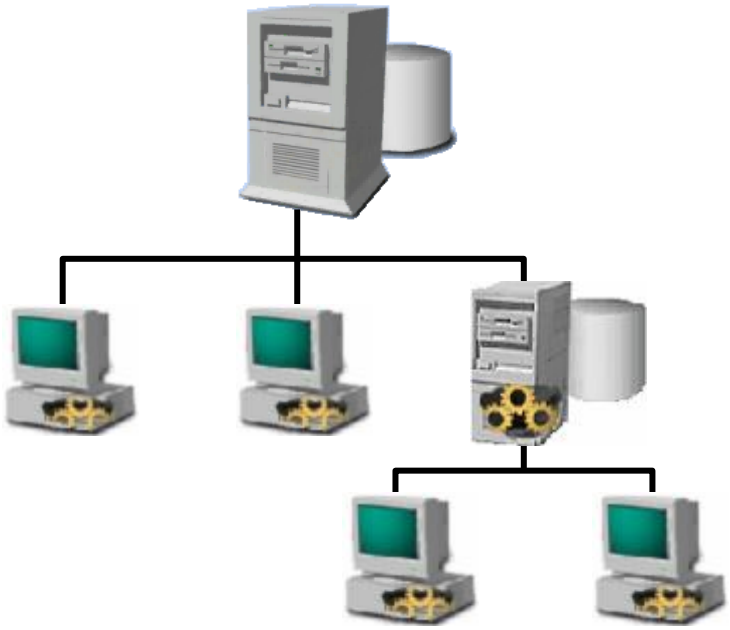
Organização hierárquica simples



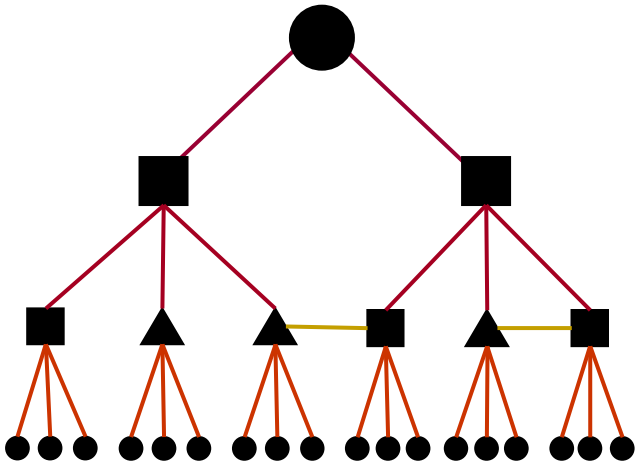
Principais Características:

- Um computador muitos utilizadores
- Computadores acessíveis apenas a grandes organizações
- Interacção com utilizador através de terminais alfanuméricos
- Arquitectura aplicacional monolítica

Computação Descentralizada



Organização hierárquica multi-nível



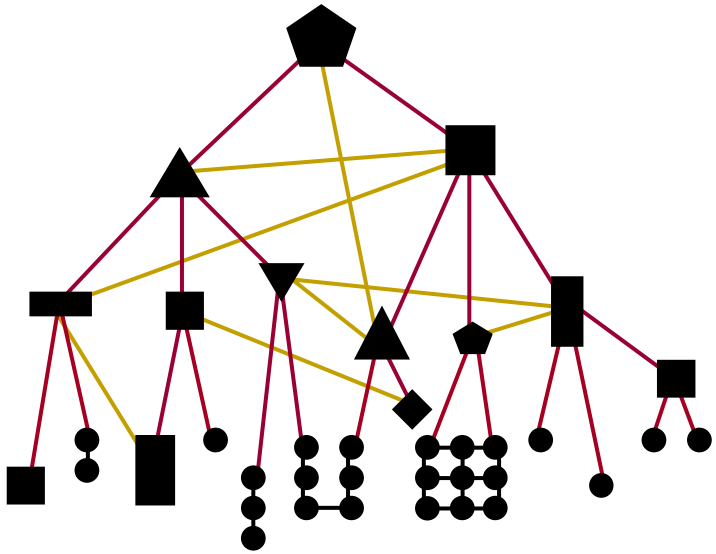
Principais Características:

- Um utilizador um computador
- Computadores menos dispendiosos ao nível departamental
- Interacção com utilizador através de interfaces gráficas
- Arquitecturas cliente/servidor

Evolução de Paradigmas de Computação

Computação em Rede

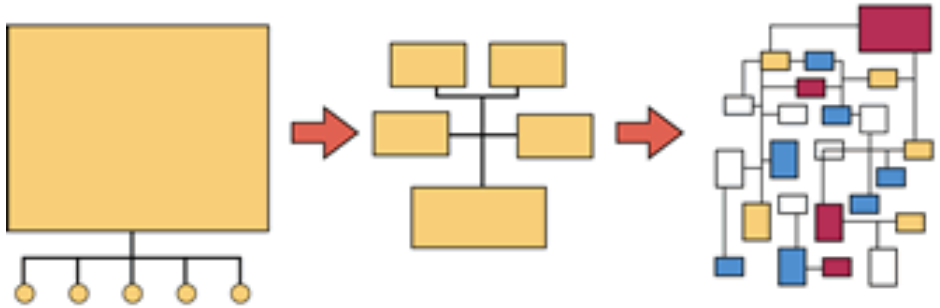
Organização híbrida



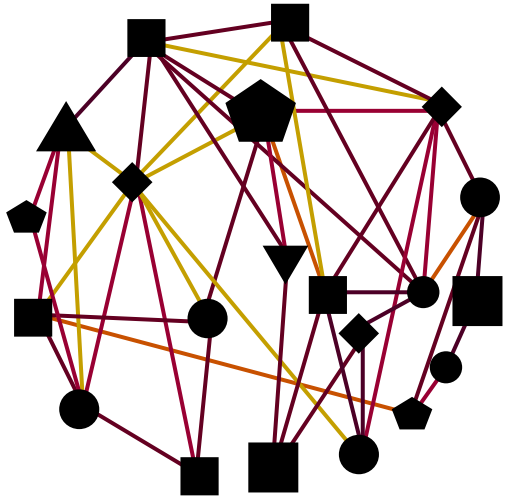
Principais Características:

- Um utilizador muitos computadores
- Processamento e dados dispersos por múltiplos dispositivos heterogéneos
- Interacção com o utilizador através interfaces multi-modais
- Arquitecturas multi-camada

Computação em Rede - Evolução



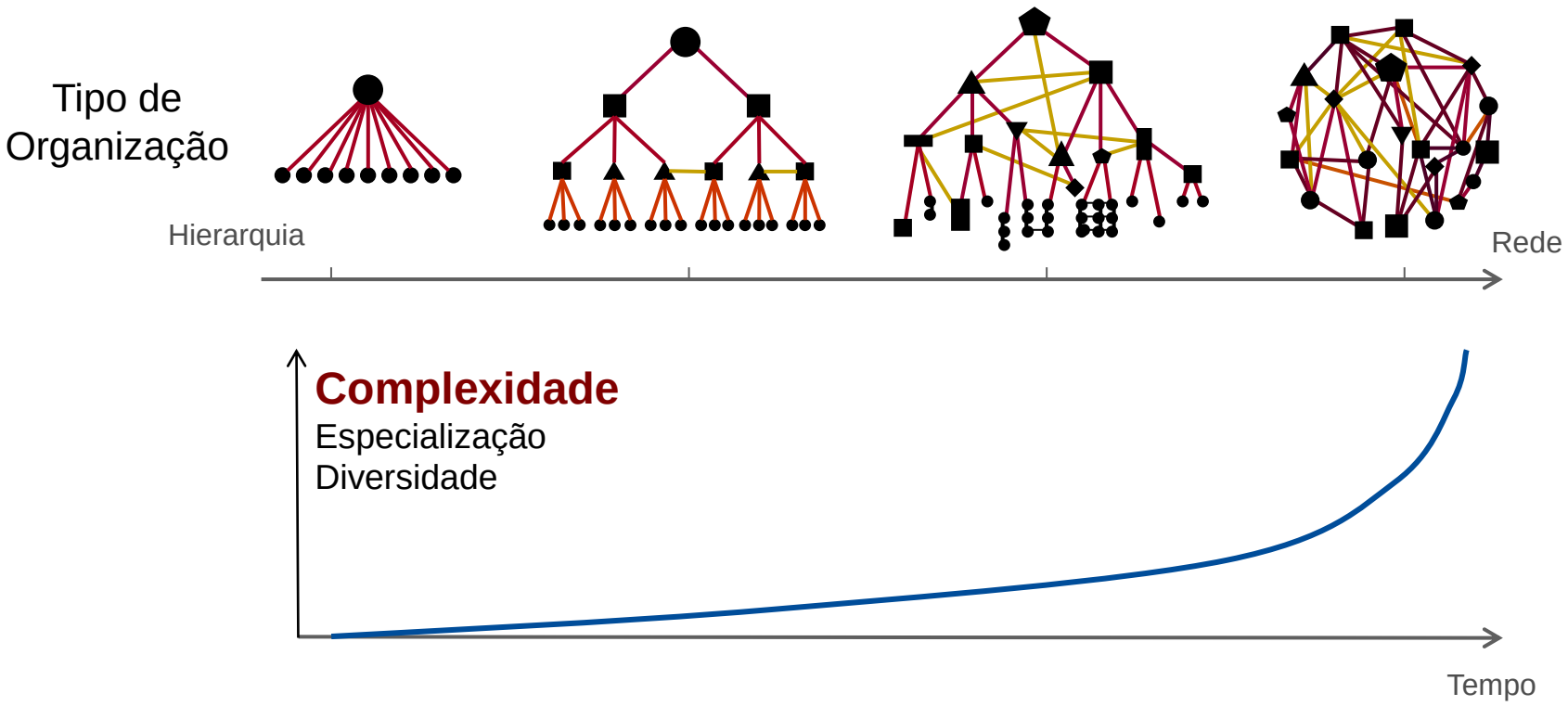
Organização em rede



Principais Características:

- Múltiplos elementos heterogéneos organizados em redes de múltiplas dependências
- Estrutura dinâmica que se adapta às condições e necessidades, de modo a tirar partido das múltiplas capacidades heterogéneas disponíveis na rede

Complexidade Crescente



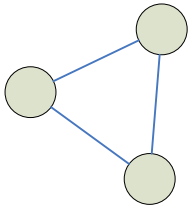
"*Complexity is the business we are in and complexity is what limits us.*"

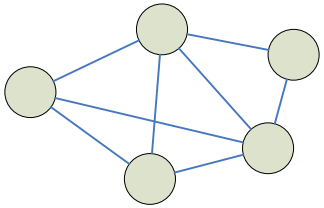
Frederick Brooks, *The Mythical Man-Month*, 1995

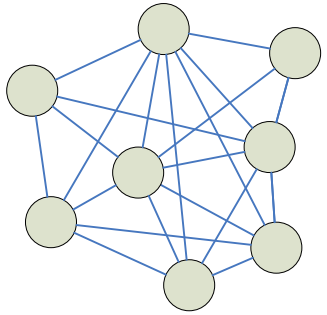
O Problema da Complexidade

Complexidade de um sistema: *grau de dificuldade em descrever esse sistema*

Menos complexo







mais complexo

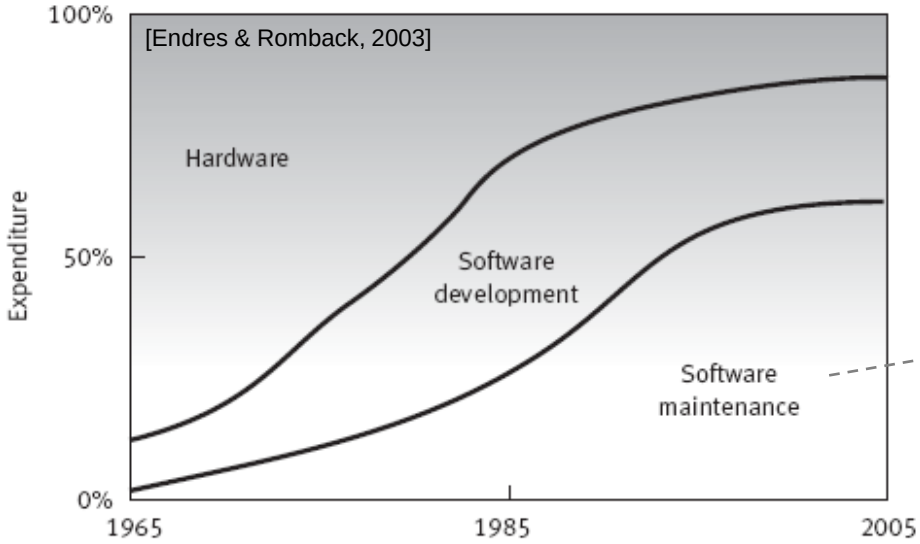
- **Um problema de interacção**
  - De partes do sistema
  - De elementos de informação
  - De elementos das equipas de desenvolvimento
- **Explosão combinatória**
  - O número de possibilidades de inter-relação das partes de um sistema cresce de forma exponencial com o número de partes
  - Um sistema com duas vezes mais partes é muito mais do que duas vezes mais complexo

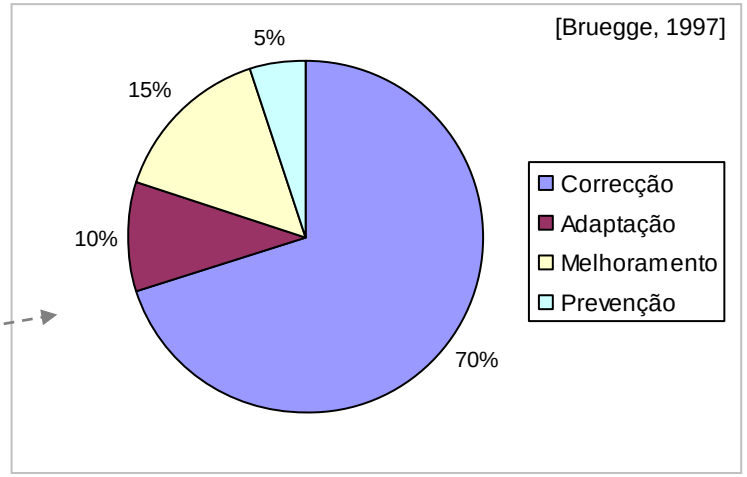
Crescimento exponencial da complexidade

Complexidade Crescente

- Evolução dos sistemas computacionais caracterizada por complexidade crescente!
  - Cada nova geração de computadores permite criar sistemas maiores e mais complexos
  - Aplicações industriais facilmente atingem vários milhões de linhas de código
- Vulnerabilidades críticas:
  - Robustez:
    - Crescente número e impacto de falhas
  - Segurança:
    - Crescente vulnerabilidade a ataques e intrusões
  - Manutenção:
    - Crescentes custos de manutenção
  - Monitorização e Gestão:
    - Crescente dificuldade de configuração, monitorização e gestão

Complexidade de Software





- **Problemas latentes**
  - Esforço de desenvolvimento (tempo, recursos)
  - Gestão e manutenção
  - Garantia de qualidade:
    - Crescente vulnerabilidade a ataques e intrusões
    - Crescente número e impacto de falhas

Exemplo: Falhas de Software

Y16.7 trillion ($182 billion) Mistake by Deutsche Bank Nearly Sinks Osaka Exchange

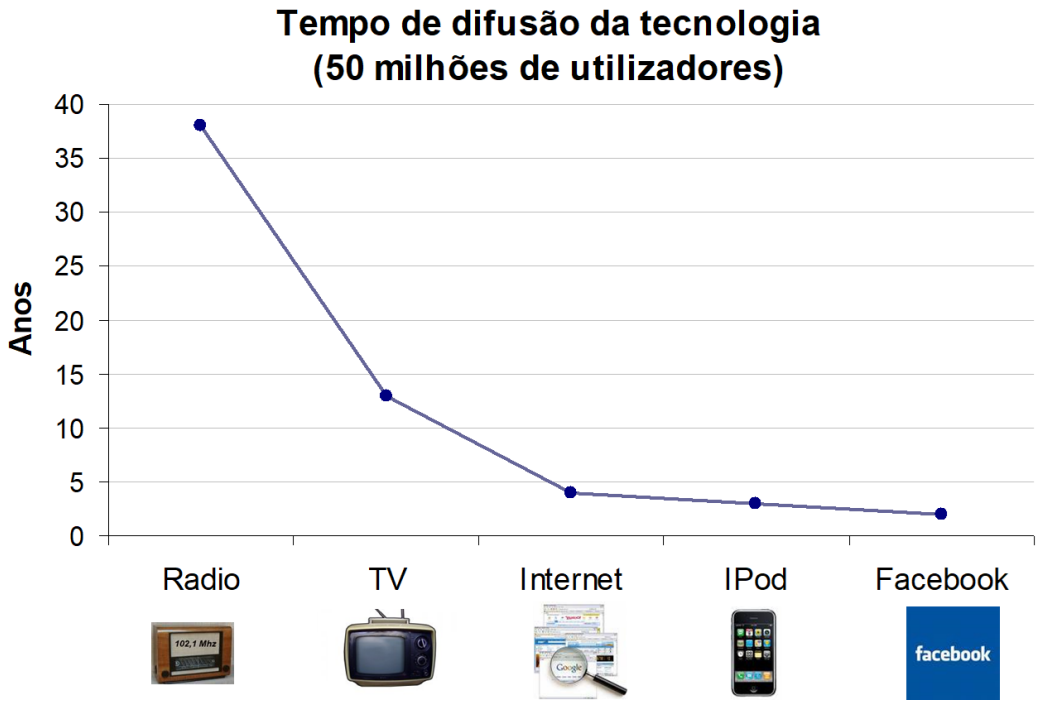
[IEEE SPECTRUM, 04-06-2010]

The international stock markets dodged a bullet this week. According to news reports like this one inReuters, Deutsche Bank's proprietary trading unit in Japan mistakenly placed sell orders of Y16.7 trillion ($182 billion) at the Osaka Securities Exchange on Tuesday when it first opened.
The reason was a software problem.

The error caused the Tokyo Stock Exchange's Nikkei-225 futures index to lose 110 points or one per cent. If mistake hadn't been caught quickly, it could have caused a repeat of the Dow flash meltdown of a few weeks ago some analysts said.

Mudança Rápida e Contínua

- Objectivos, sistemas e pessoas evoluem
- Evolução permanente
- Necessidade de estabilidade para a concretização de sistemas viáveis
- **Necessidade de conciliar mudança com estabilidade**



Mudança Rápida e Contínua

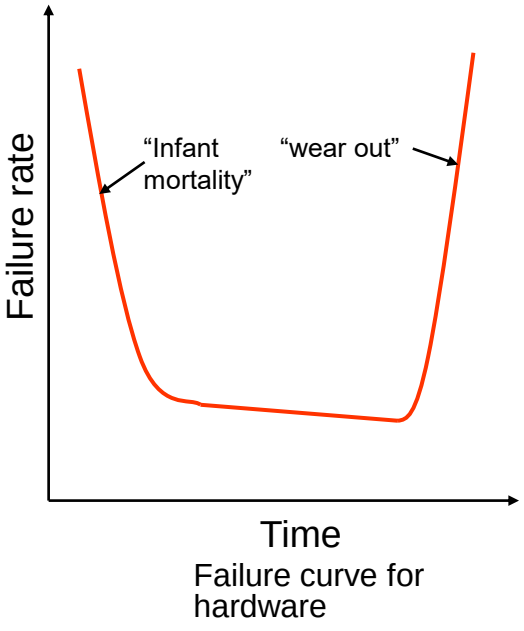
A mudança rápida no âmbito do desenvolvimento de software tem consequências relevantes:

- **Mudanças nas necessidades dos domínios de utilização** do software, com alteração de necessidades e requisitos, o que requer capacidade de adaptação rápida a essa mudança
- **Custo de mudança exponencial** se não ocorrer no tempo adequado, quanto mais tarde a mudança for feita, maior será o esforço e o custo associados, aumentando a importância de entregar o software o mais rápido possível, com versões incrementais e interativas
- **Optimização e automação nos processos de desenvolvimento** no sentido de obter uma maior eficiência e produtividade no desenvolvimento de software

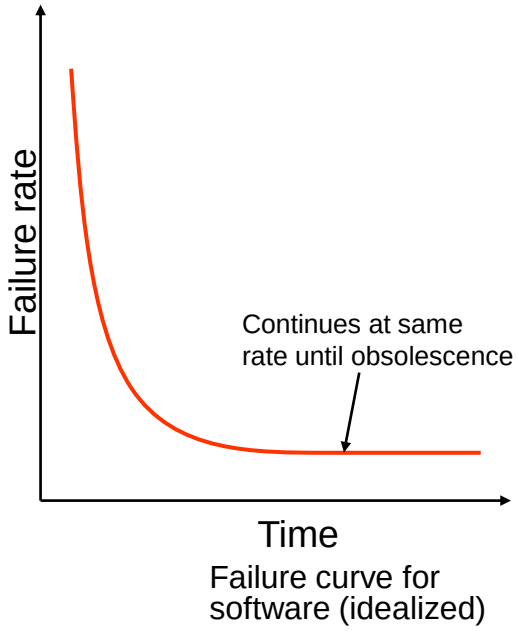
Características do Software

- O software é desenvolvido e não produzido no sentido clássico
- O processo de desenvolvimento é difícil de gerir
- Os domínios dos problemas a resolver são complexos
- O software oferece uma elevada flexibilidade
- Os sistemas de software possuem dinâmicas internas complexas

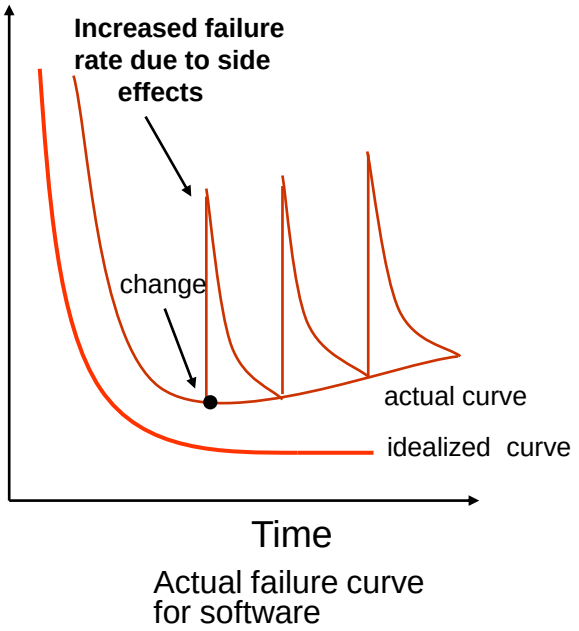
Desenvolvimento de hardware vs. software



Failure curve for hardware



Failure curve for software (idealized)



Actual failure curve for software

[Pressman, 2003]

Software: algumas ideias pré-concebidas

Ideia:

- Se um projecto estiver atrasado, o atraso pode ser recuperado pela utilização de mais programadores

Realidade:

- O desenvolvimento de software não é um processo mecânico como a manufactura - "***adding people to a late software project makes it later***" [Brooks, 1975]

Ideia:

- Uma descrição geral dos objectivos é suficiente para começar a escrever programas - os detalhes podem ser definidos posteriormente

Realidade:

- Uma especificação inicial deficiente é uma das principais causas de insucesso na realização de software. ***"Quanto mais cedo se começar a programar, mais tarde o programa estará pronto***" [Pressman, 2003].

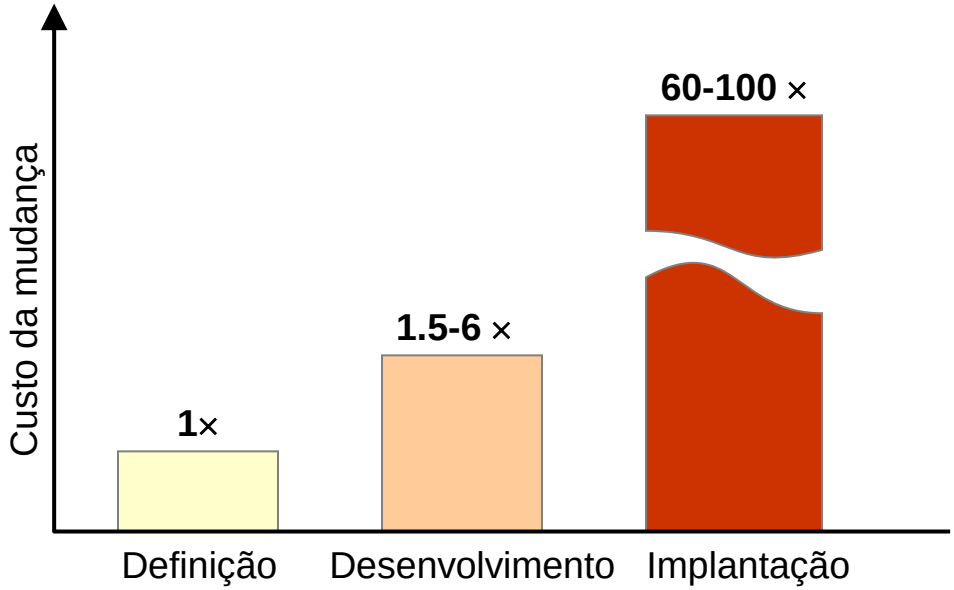
Software: algumas ideias pré-concebidas

Ideia:

- Os requisitos de um programa mudam continuamente, mas essas mudanças podem ser facilmente acomodadas porque o software é flexível

Realidade:

- É verdade que o software é flexível, mas o impacto da mudança varia com a altura em que é introduzida



[Pressman, 2003]

A realidade do desenvolvimento de software actual

- **O software tornou-se essencial** em praticamente todos os aspectos da sociedade. O número de pessoas que têm interesse nas funcionalidades e recursos fornecidos por uma aplicação específica cresceu dramaticamente. *É necessário entender o problema antes de desenvolver uma solução de software*;
- **As exigências tecnológicas tornam-se cada vez mais complexas a cada ano que passa**. Software sofisticado que anteriormente era implementado em ambientes de computação previsíveis e independentes, agora está embutido em todo o tipo de dispositivos e veículos. *A arquitectura de software tornou-se uma atividade fundamental*;
- **Indivíduos, empresas e governos dependem cada vez mais de software**. Se o software falhar, pessoas e organizações podem ter desde pequenas perturbações até consequências catastróficas. *O software deve apresentar alta qualidade*;
- **O valor das aplicações tende a aumentar com o tempo à medida que o número de utilizadores aumenta**, pelo que as necessidades de manutenção, melhoria e evolução também aumentam. *O software deve ser adequadamente mantido*.

Estas realidades requerem que, em todas as suas formas e em todos os seus domínios de aplicação, **o software deve ser desenvolvido numa perspectiva de engenharia**.

[Pressman & Maxim, 2019]

Engenharia de Software

- Surgiu de forma consistente no final da década de 1960, impulsionada pelas falhas crónicas em projectos de software, nomeadamente em termos de prazos e de orçamentos
  - Necessidade de aplicações de grande dimensão e elevada qualidade
  - Aplicações militares
  - O que foi designado como "***A crise do software***"
- O termo surgiu no seguimento da Conferência da NATO em Garmisch Partenkirchen (Alemanha), 1968, onde participaram, entre outros, Tony Hoare, Edsger Dijkstra, Alan Perlis e Niklaus Wirth
- **Resultou na compreensão de que os princípios da engenharia deveriam ser aplicados ao desenvolvimento de software**

Engenharia de Software

- Desenvolvimento de Software como uma actividade de **engenharia**:
  - Sistemático
  - Quantificável
- **Engenharia de Software**:
  - Aplicação de abordagens sistemáticas, disciplinadas e quantificáveis ao desenvolvimento, operação e manutenção de software [IEEE, 1990]

Engenharia de Software

- Conjunto de metodologias, técnicas e ferramentas para a realização de
  - sistemas informáticos de qualidade
  - de acordo com o orçamento definido
  - no prazo estabelecido
- Mesmo perante a complexidade e a mudança

Bibliografia

[Brooks, 1975]

Frederick Brooks, The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, 1975.

[Bruegge, 1997]

Bernd Bruegge, Lecture Notes on Software Engineering, Carnegie Mellon University - School of Computer Science, 1997.

[Endres & Rombach, 203]

Albert Endres, Dieter Rombach, *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories,* Addison Wesley, 2003.

[IEEE, 1990]

IEEE 610.12-1990, Standard Glossary of Software Engineering Terminology, 1990.

[Leveson & Turner, 1993]

Nancy Leveson, Clark S. Turner, An Investigation of the Therac-25 Accidents, IEEE Computer, Vol. 26, No. 7, July 1993, pp. 18-41.

[Pressman, 2003]

Roger Pressman, *Software Engineering: a Practitioner's Approach,* McGraw-Hill, 2003.

[Pressman & Maxim, 2019]

Roger Pressman, Bruce Maxim, *Software Engineering: a Practitioner's Approach,* McGraw-Hill, 2019.

[Schach, 2011]

Stephen Schach, *Object-Oriented and Classical Software Engineering,* McGraw-Hill, 2011.

[Roubine 2000]

O. Roubine - E-Development. Rational, 2000.