

# Engenharia de Software I

- Crise de software e Sistemas legados
- Engenharia de Software
- Ciclo de vida
- Modelos de Processo de Software

# Crise de software e Sistemas Legados

# Crise de software

---

## **Motivo:**

- Rápido progresso do hardware
- Muitas demandas por sistemas cada vez mais complexos
- Engenharia de Software era uma disciplina incipiente (iniciante) e pouco aplicada pelas empresas
- Falta de comunicação entre equipe de desenvolvimento e clientes (antigos CPDs)
- Primeiros programadores eram matemáticos

## **A crise se manifestou de várias formas:**

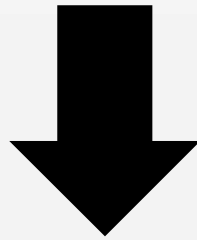
- Projetos estourando o orçamento
- Projetos estourando o prazo
- Software de baixa qualidade
- Software muitas vezes não atingiam os requisitos
- Projetos não gerenciados e difícil de manter e evoluir
- Clientes insatisfeitos

# Sistemas Legados

---

- Desenvolvidos usando tecnologia antiga ou obsoleta
- Sistemas críticos
- É muito arriscado descartar esses sistemas quando estão estáveis:
  - Exemplo: sistemas bancários, sistemas do governo, ERPs de grandes corporações
- As vezes é muito caro e arriscado descartar esses sistemas, por isso se mantem eles
- Falta de documentação, planejamento e qualidade dos projetos de software

**Crise de software (falhas)**



**Engenharia de software**

A Engenharia de Software surgiu com o objetivo de melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento.

# Engenharia de Software

# Engenharia de software

---



A Engenharia de Software trata de aspectos relacionados ao estabelecimento de **processos**, **métodos** e **ferramentas** para dar suporte ao desenvolvimento de softwares com **foco na qualidade**.

# Qualidade de Software

---

Então, qualidade é um conceito relativo. Está diretamente relacionada à:

- **Conformidade com requisitos:** requisitos são especificados e espera-se que sejam atendidos.
  - Requisitos são as características que definem os critérios de aceitação de um produto.
  - É o **confronto** entre a **promessa** e a **realização**.
- **Satisfação do cliente:** requisitos são especificados por pessoas (desenvolvedores) para satisfazer outras pessoas (cliente).
- **Produto sem defeitos:** desenvolvido corretamente (sem bugs).



## Como produzir software com qualidade?

---

**A qualidade é consequência boas adoções de :**

- dos processos
- dos métodos
- da tecnologia
- das pessoas

# Processo de Software

---

## O que é um processo?

Processo é um conjunto de atividades realizadas para atingir um objetivo específico.  
Um conjunto de regras que definem como um projeto deve ser executado.

## O que é um processo de software?

Um **processo de software** pode ser visto como o conjunto de atividades, métodos e práticas e que guiam pessoas na produção de software.

Exemplos:

- Processo de gerência de requisitos
- Processo de gerência de configuração
- Processo de gerência de projetos
- Processo de testes



**Elicitação/  
Especificação  
de requisitos**

**Manutenção  
Evolução**



**Projeto de software**



**Plano de testes**



**Codificação**



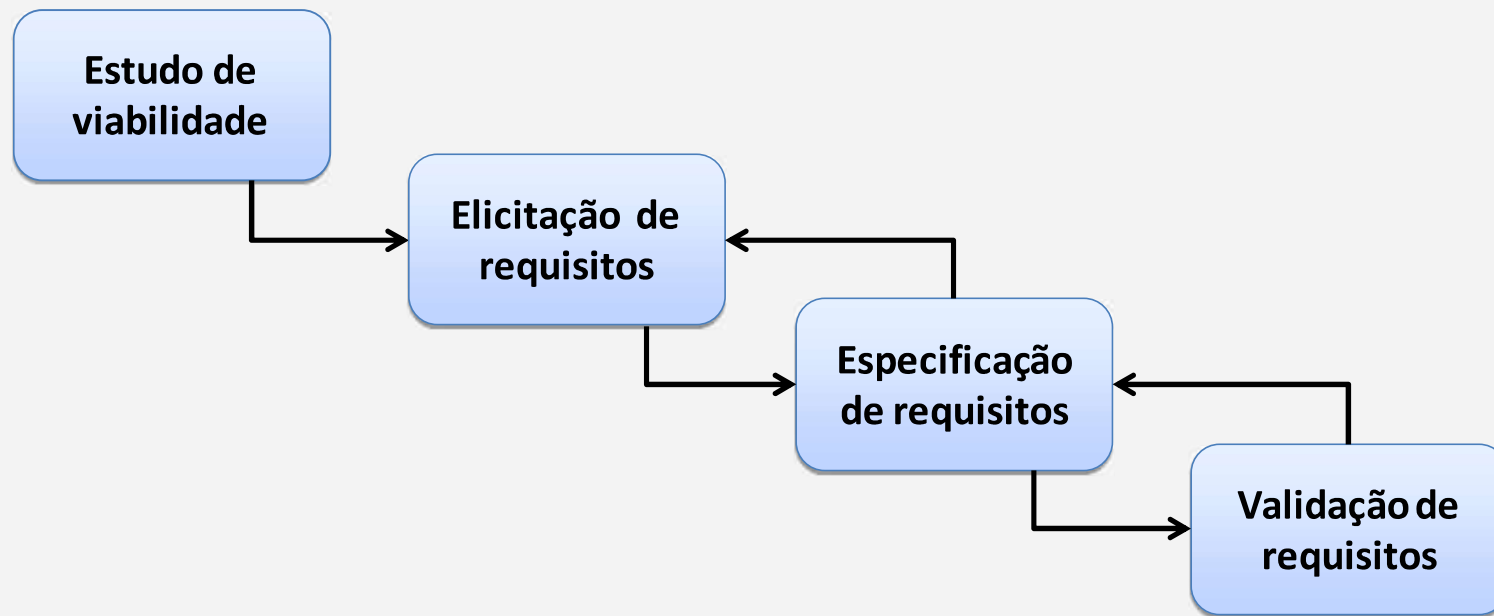
**Testes de software**



**Implantação**

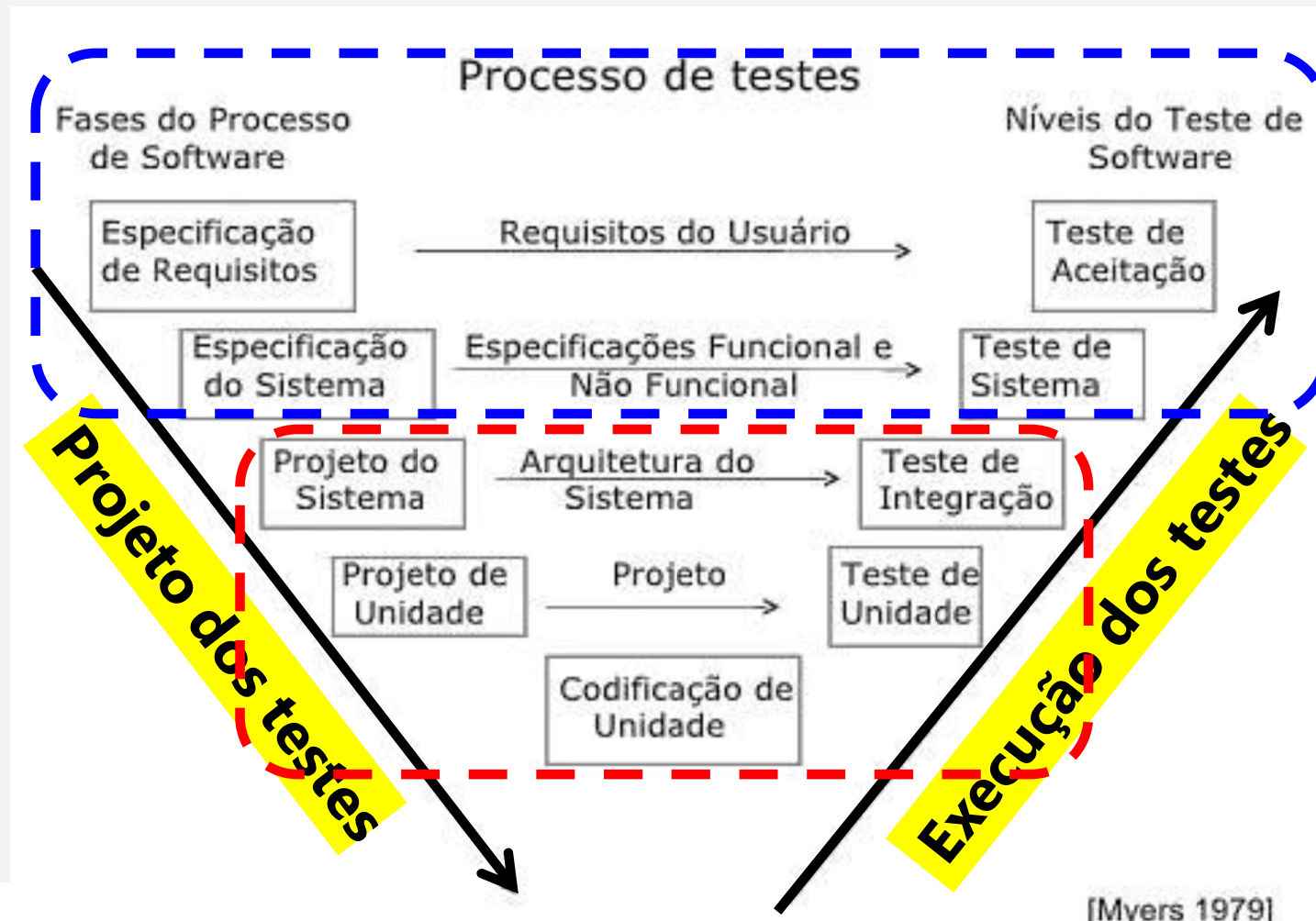
## Ex: Processo engenharia de requisitos

---



**Gerência de Requisitos**

## Ex.: Processo de testes de software



# Métodos

---

**Métodos:** englobam um conjunto de tarefas que definem “como fazer” para construir um software

Engloba um amplo conjunto de tarefas que incluem: planejamento e estimativa do projeto, análise de requisitos, projeto da estrutura de dados, implementação, teste e manutenção

Muitas vezes utilizam notação gráfica

- Ex1. método orientado a objetos → projeto e desenvolvimento
- Ex2. Método ágil Scrum → Para gestão de projetos
- Ex3. TDD → Método para execução dos testes de software

# Ferramentas

---

**Ferramentas**: é a escolha dos “instrumentos apropriados” para o desenvolvimento

Dão suporte automatizado ou semi-automatizado aos métodos.

Exemplo: IDEs, Banco de dados, Git para controle de versão, etc.

# Fórmula do sucesso



Engenharia  
de Software



Pessoas



Ouvir o cliente e  
Bom atendimento



Produto c/ qualidade  
Cliente satisfeito

> LUCRO



# Ciclo de vida de um software (etapas/fases obrigatórias)



**Elicitação/  
Especificação  
de requisitos**

**Manutenção  
Evolução**



**Projeto de software**



**Plano de testes**



**Codificação**



**Testes de software**



**Implantação**

**GERÊNCIA DE PROJETOS**

**Modelos de processo de software  
(como organizar a execução das fases)**

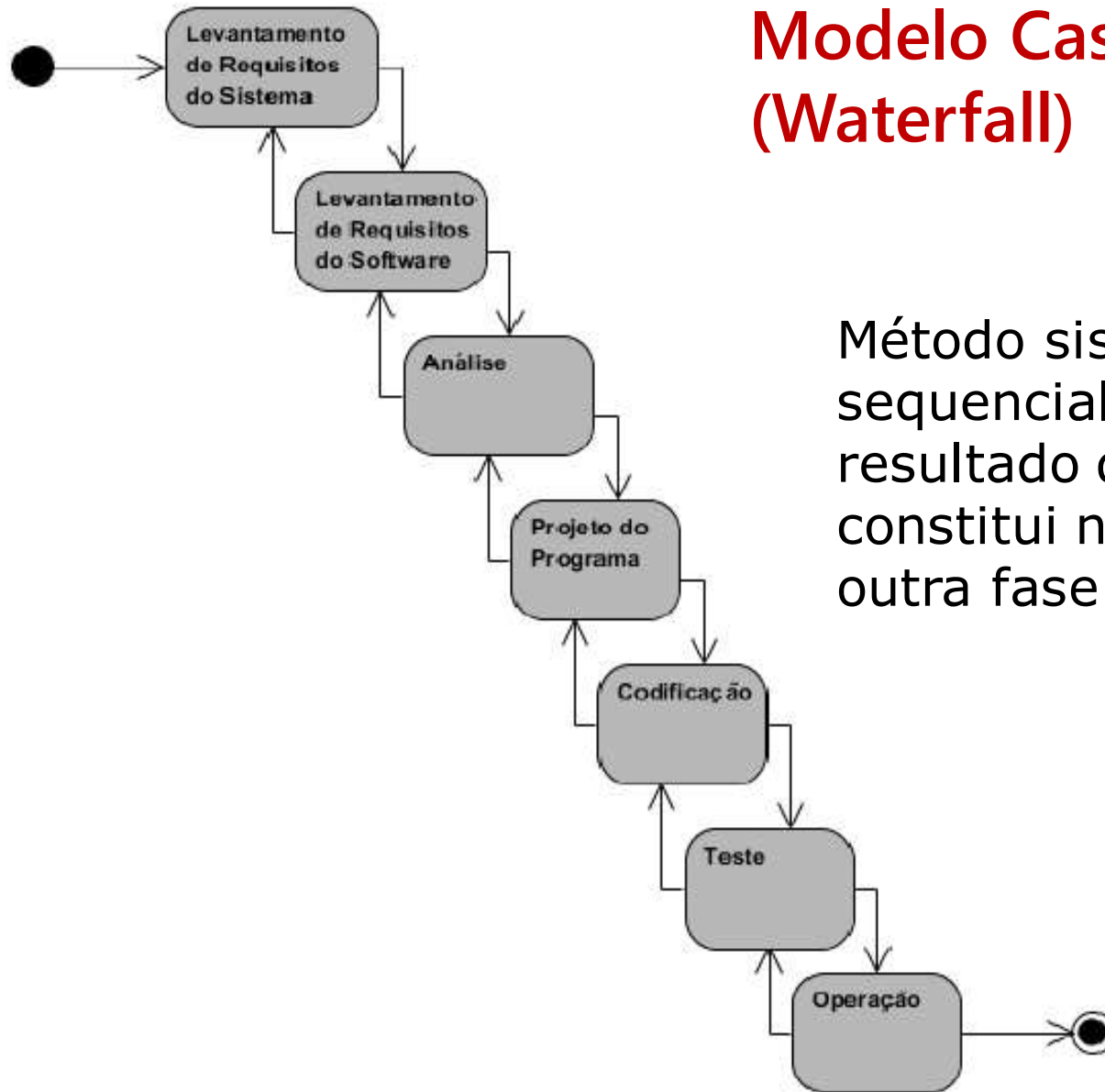
# Principais modelos

---

- Modelo Cascata
- Modelo Iterativo / Incremental
- existem outros, mas estes são os mais utilizados

## Modelo Cascata (Waterfall)

Método sistemático e sequencial, em que o resultado de uma fase se constitui na entrada da outra fase.



# Modelo cascata (Waterfall)

---

- Modelo mais antigo
- Mais amplamente usado na engenharia de software
- Modelo dirigido a planos
- Fases de especificação e desenvolvimento separadas e distintas

# Modelo Cascata

---

## **Problemas do modelo Cascata:**

- Requisitos devem ser estabelecidos de maneira completa correta e clara no início de um projeto
- Difícil avaliar o progresso verdadeiro do projeto durante as primeiras fases
- Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento
- Ao final do projeto, é necessário um grande esforço de integração e testes
- Gera grande quantidade de documentação

# Modelo Cascata

---

## Problemas do modelo Cascata:

Dificuldade de acomodação de mudanças depois que o processo já foi iniciado

- Por isso esse modelo só é **apropriado quando os requisitos são bem entendidos** e as mudanças durante o processo de projeto serão limitadas
- Poucos sistemas de negócio possuem requisitos estáveis.

**Este modelo tem sido muito criticado pelos defensores dos métodos ágeis**



# Modelo iterativo e incremental

Incremento 1



Incremento 2



Incremento 3



Incremento 4



# Modelo iterativo e incremental

---

- Ao invés de entregar o sistema em uma única entrega, o desenvolvimento e a entrega são distribuídos em incrementos, nos quais cada incremento entrega parte da funcionalidade necessária.
- O processo se repete até que um produto completo seja produzido.
- Os requisitos do usuário são priorizados e os requisitos de mais alta prioridade são incluídos nos primeiros incrementos
- Assim que o desenvolvimento de um incremento é iniciado os requisitos são congelados, mas os requisitos dos incrementos posteriores podem continuar a evoluir
- Necessidade de entrega de um produto funcional em pouco tempo
- A cada incremento é produzida uma versão operacional do software.
- Abordagem normalmente usada em métodos ágeis

# Benefícios do desenvolvimento iterativo e incremental

---

## **O custo para acomodar mudanças nos requisitos do cliente é reduzido:**

- A quantidade de análise e documentação que precisa ser feita é bem menor do que a necessária no modelo cascata.

## **É mais fácil obter feedback do cliente sobre o trabalho de desenvolvimento que tem sido feito:**

- Os clientes podem comentar demonstrações do software e ver o quanto foi implementado.

## **Possibilidade de mais rapidez na entrega e implantação de software útil para o cliente:**

- Os clientes podem usar e obter ganhos do software mais cedo do que é possível no processo cascata.

## **Menor risco de falha geral do projeto**

# Problemas do desenvolvimento iterativo e incremental

---

**A maioria dos sistemas requer um conjunto de funções básicas que são usadas por diferentes partes do sistema:**

- Como os requisitos não são definidos em detalhes até que um incremento seja implementado, pode ser difícil identificar funções comuns que são necessárias a todos os incrementos.

**A estrutura do sistema tende a degradar conforme novos incrementos são adicionados:**

- A menos que tempo e dinheiro sejam gastos na reconstrução para melhorar o software, as mudanças regulares tendem a corromper a estrutura do sistema. A incorporação posterior de mudanças no software se torna progressivamente mais difícil e cara.

**O gerenciamento de custo, cronograma e configuração é mais complexo**

**Se os requisitos são instáveis ou não descritos de forma completa quanto se esperava, alguns incrementos podem precisar ser retirados de uso ou causar retrabalhados**