

# PROJETO DETALHADO DE SOFTWARE

## AULA 02- CONCEITOS RELACIONADOS

Profa. Jacilane de Holanda Rabelo

# PROCESSO DE SOFTWARE



# Objetivo

- Fornecer motivação para os assuntos subsequentes da disciplina
- Conhecer os diferentes conceitos e processos de desenvolvimento de software
- Processo de desenvolvimento de software
  - Descreve uma abordagem para construção, implantação e manutenção de software

# Introdução

4

1. **Processos de software** podem ser **melhorados** através da **padronização dos processos** utilizados dentro de uma organização
2. A **modelagem de processos** torna-se essencial para a **definição de processos eficientes, capazes** de serem **replicados**
3. Modelagem de processos inclui responder as seguintes perguntas:
  - ▣ O quê?
  - ▣ Como?
  - ▣ Quem?
  - ▣ Quando?
  - ▣ Por quê?

# Modelos de Processo de Software

5

- São representações abstratas de um processo de software
  - ❖ Atividades, papéis e artefatos
- Cada **modelo representa um processo** a partir de uma perspectiva particular
- Definem as atividades para o desenvolvimento do software
- Especificam os produtos de cada atividade
- Indicam os papéis das pessoas envolvidas

# Modelos de Processo de Software

6

- Deve ser escolhido com base:
  - ❖ Na natureza do projeto e da aplicação
  - ❖ Nos métodos e ferramentas a serem utilizados
  - ❖ Nos controles e produtos que precisam ser entregues

# Modelos de Processo de Software - Vantagens

7

- Oferecem um roteiro útil para o trabalho de engenharia de software
- Mas, nenhum modelo de processo é perfeito
- Outras vantagens
  - ▣ Padronização dos artefatos
  - ▣ Melhor comunicação da equipe
  - ▣ Menos treinamento de pessoal

# Modelos de Processo de Software - Vantagens

8

- Um **método (ou modelo de processo)** é algo teórico, um conjunto de possíveis ações – conteúdo do método.
  - Define o *que, como e porque fazer*



# Modelos de Processo de Software - Vantagens

9

- O processo deve determinar ações práticas a serem realizadas pela equipe como prazos definidos e métricas para se avaliar como elas estão sendo realizadas.
  - Define *quem e quando fazer*.

Método + Planejamento = Processo

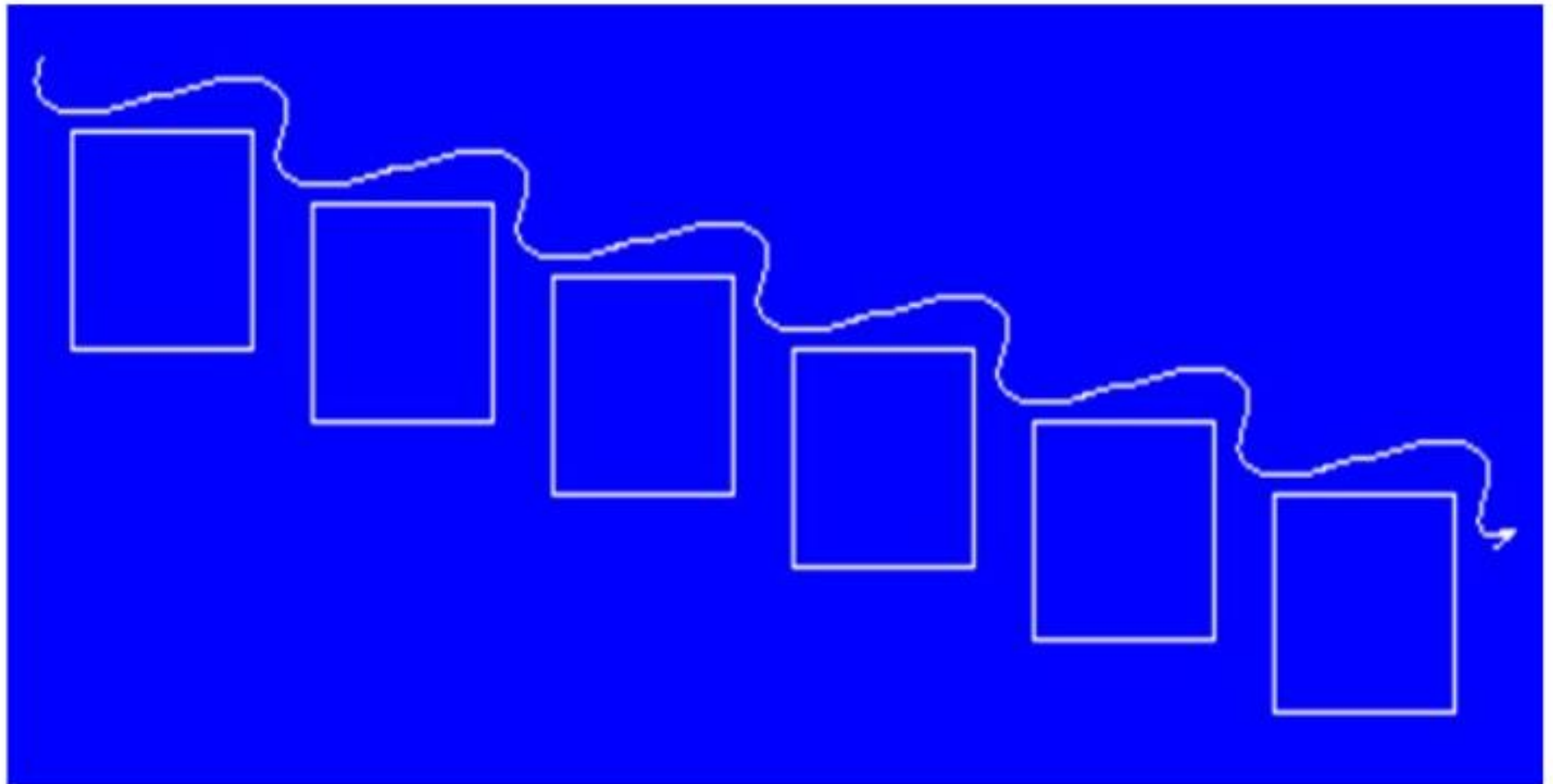
# Modelos de ciclo de vida de projeto

---

- Existem alguns processos pré-fabricados
  - Esses processos são conhecidos como modelos de ciclo de vida
  - Esses processos apresentam características predefinidas
  
- Devem ser adaptados para o contexto real de uso
  - Características do projeto
  - Características da equipe
  - Características do cliente
  
- Exemplos:
  - Cascata
  - Incremental
  - RAD (Rapid Application Development)
  - Prototipação
  - Espiral

# O Modelo em “Cascata”

1

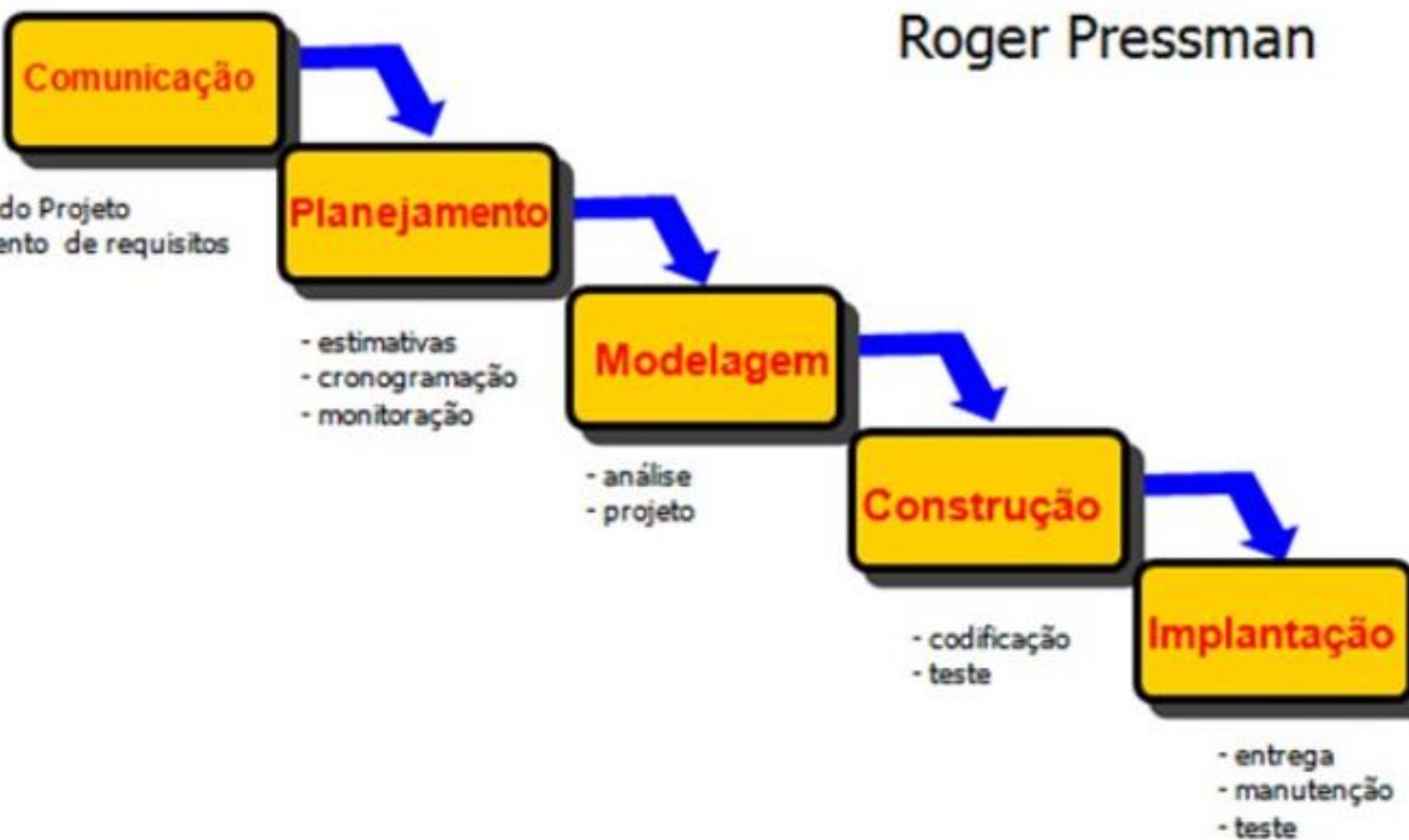


# O Modelo em “Cascata”

- Originou-se de outros processos de engenharia
- Derivado do mundo do hardware (linhas de montagens)
- Retrata um desenvolvimento gradual
- Sua estrutura é composta por várias etapas que são executadas de forma sistemática e sequencial

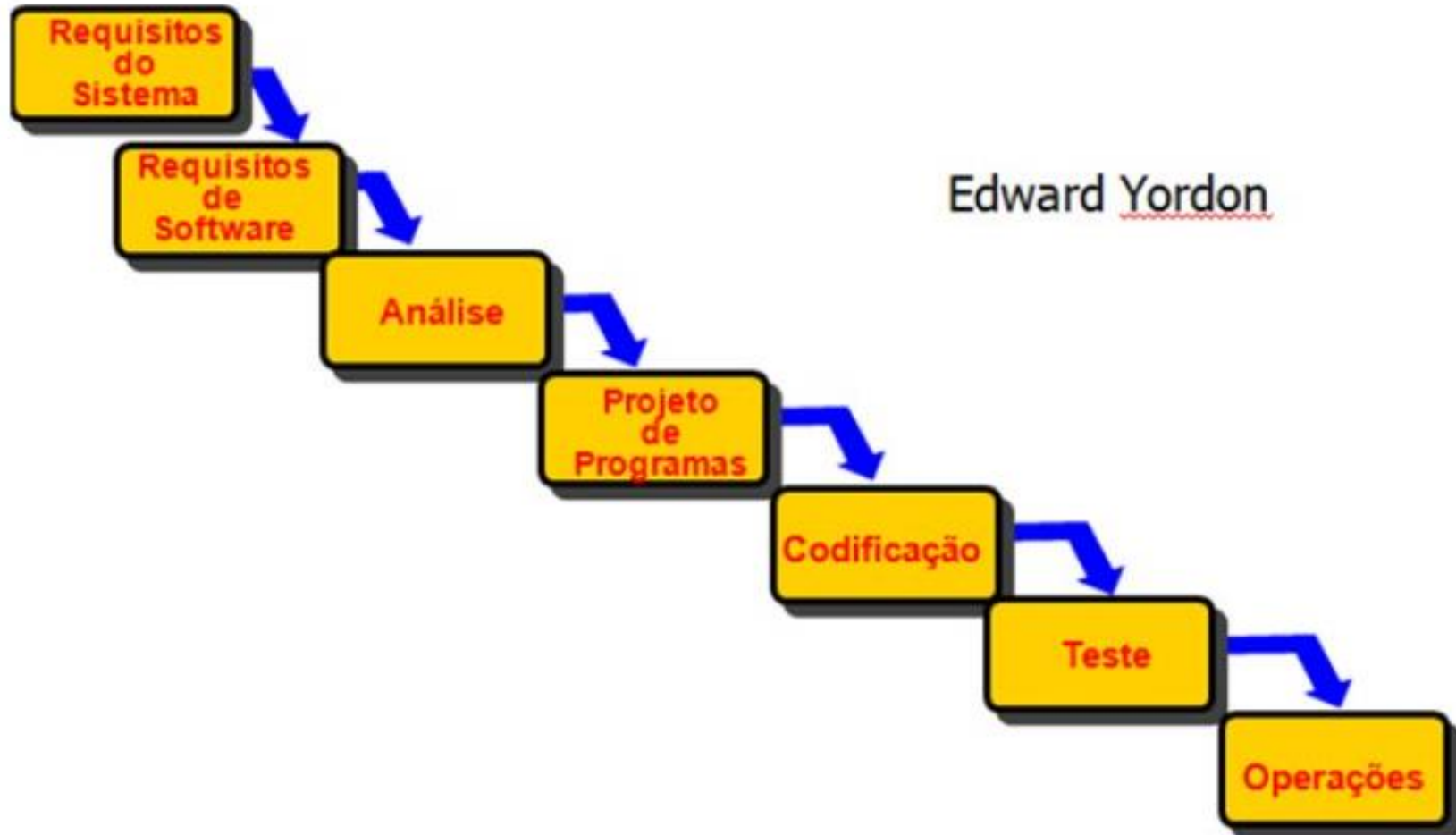
# O Modelo em “Cascata”

Roger Pressman

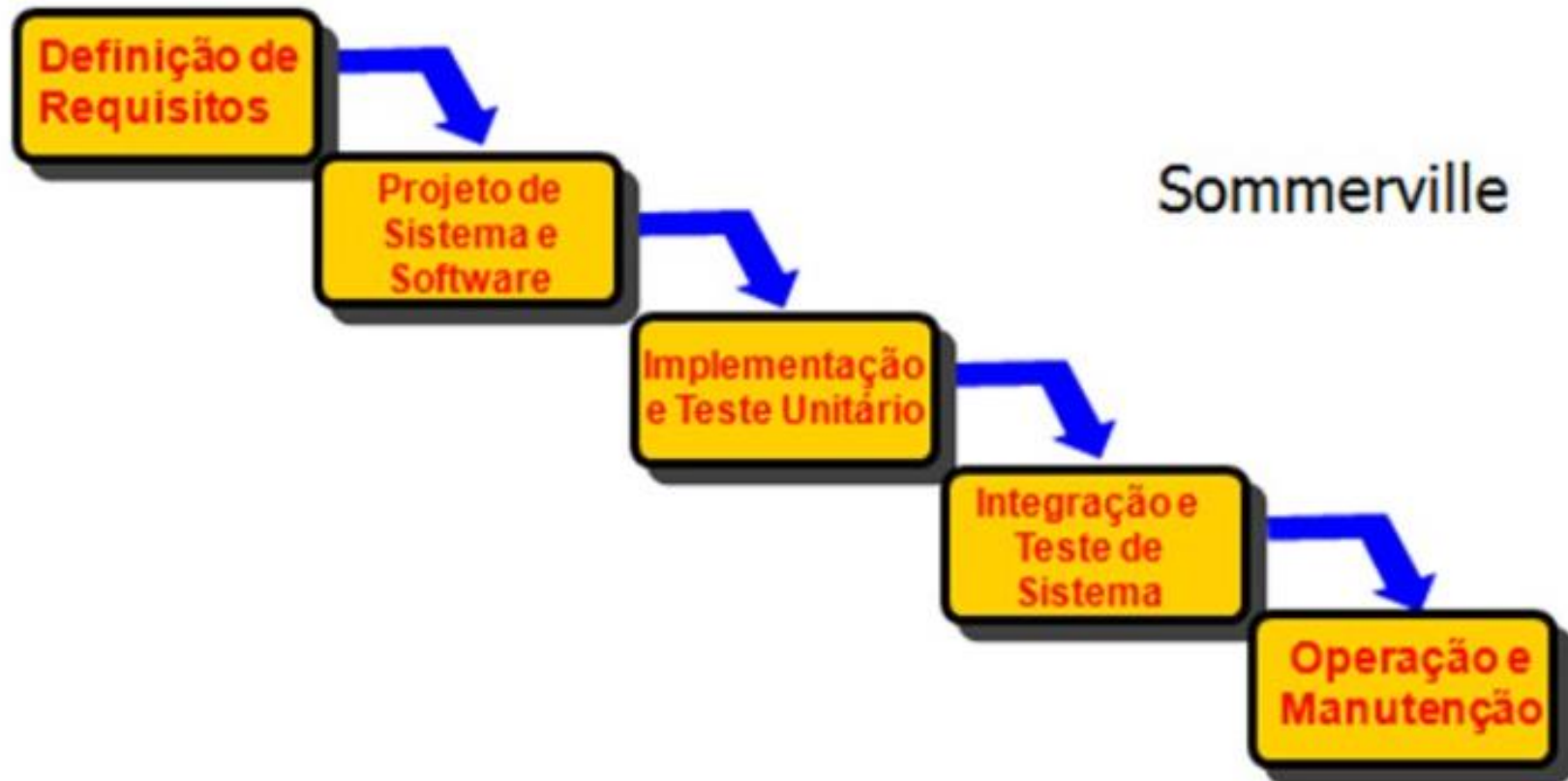


# O Modelo em “Cascata”

14

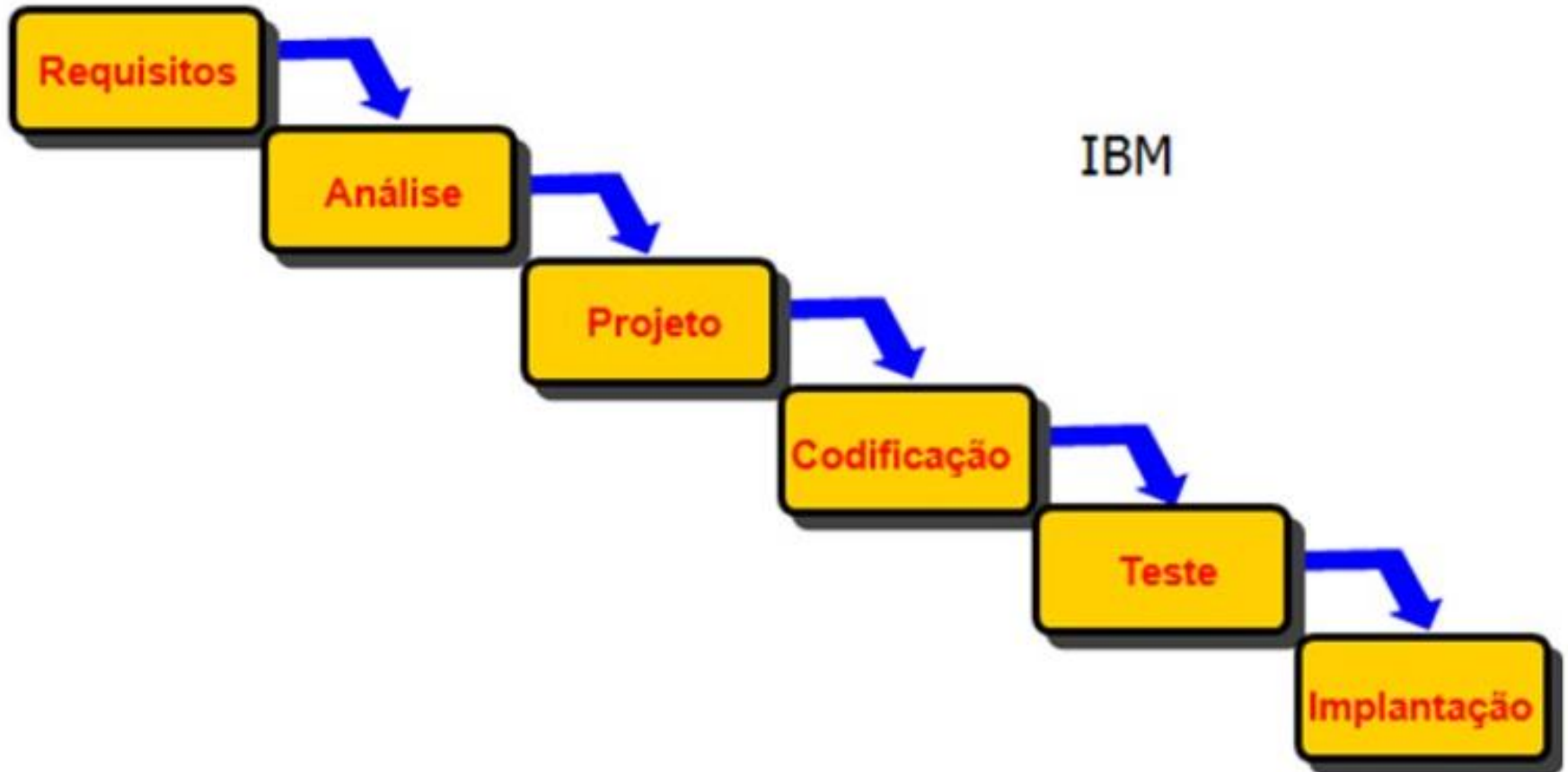


# O Modelo em “Cascata”



# O Modelo em “Cascata”

16





## Ciclo de vida Cascata

---

- Útil quando se tem requisitos estáveis e bem definidos
  - Ex.: Adicionar um novo dispositivo legal em um sistema de contabilidade
- Não lida bem com incertezas
- Fornece pouca visibilidade do estado do projeto
  - Muito tempo para a primeira entrega
  - Dificuldade na obtenção de feedback do cliente

# O Modelo em “Cascata” – Quando aplicar?

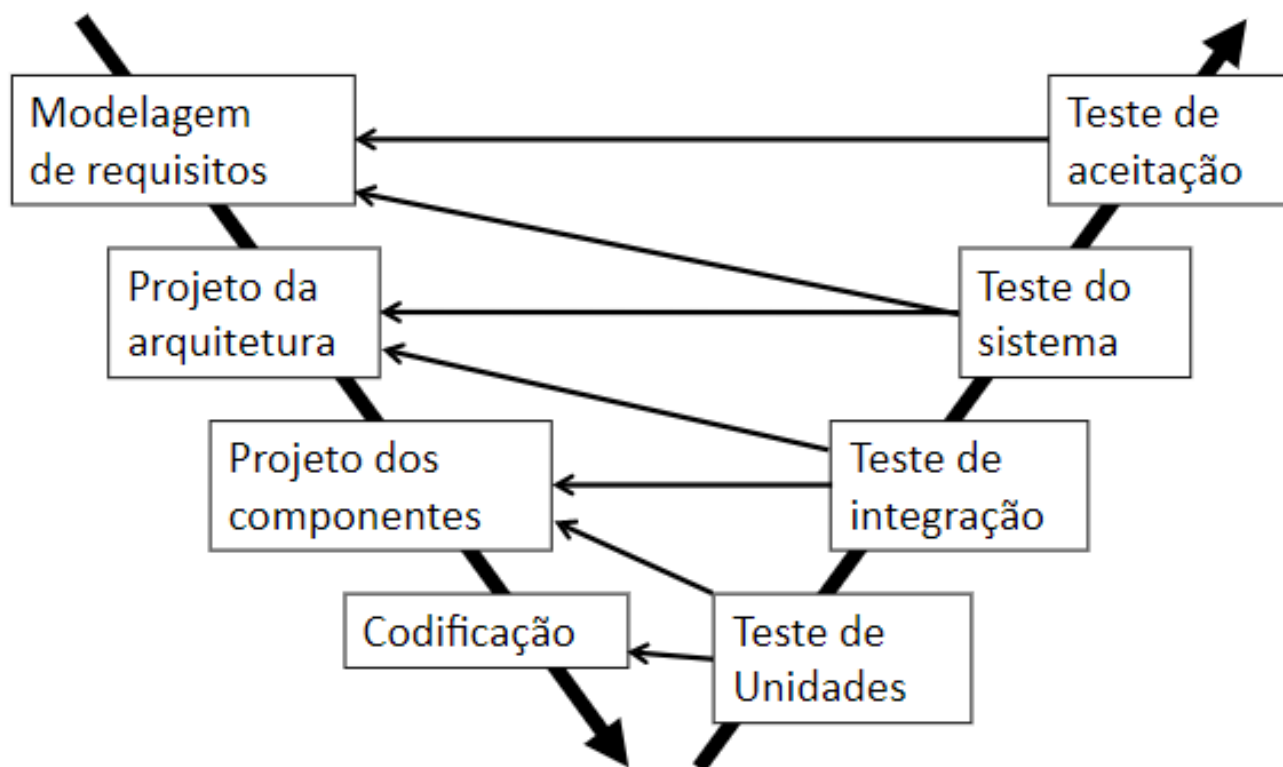
- Sistemas críticos
- Quando os requisitos são bem compreendidos
- Quando há pouca probabilidade dos requisitos mudarem

# O Modelo em “Cascata” - Importância

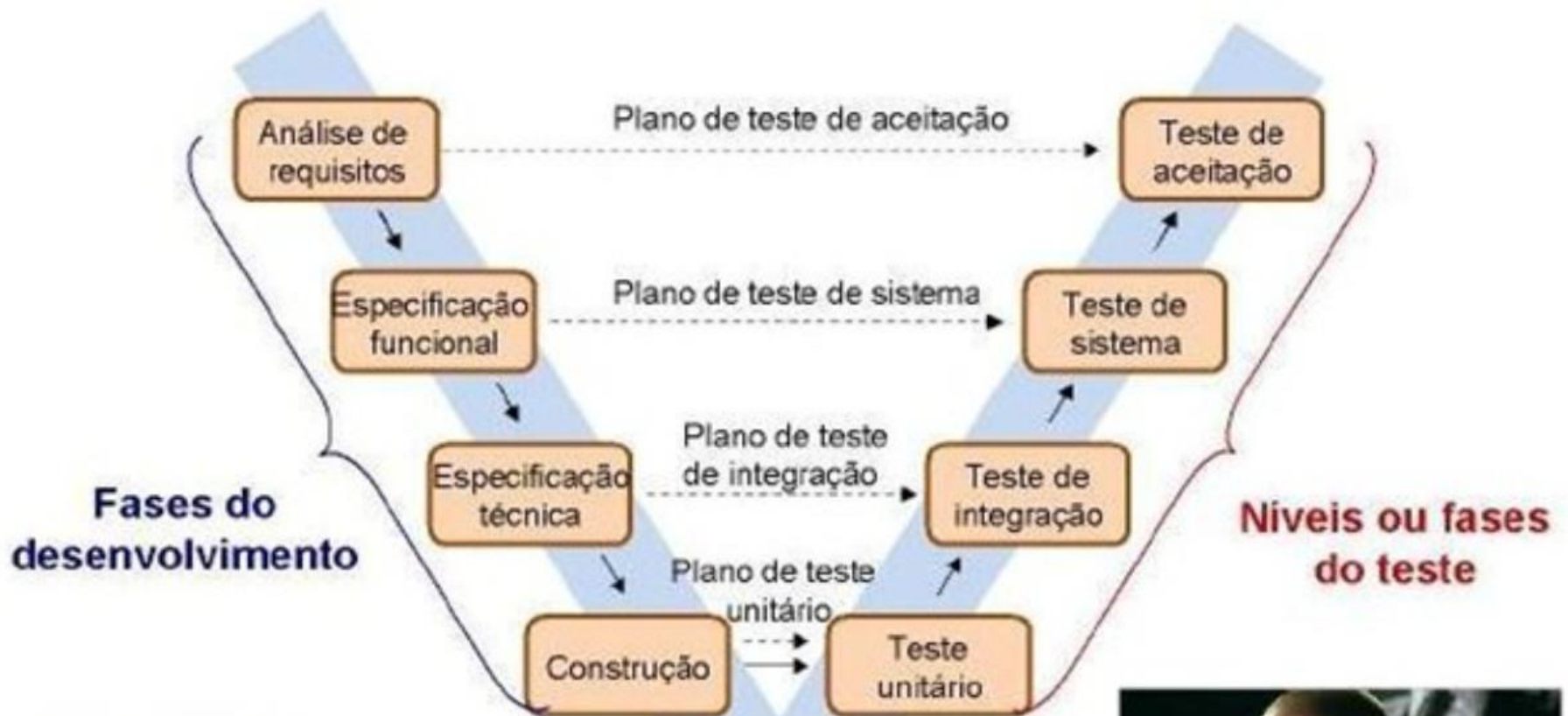
- Trouxe contribuições importantes para o processo de desenvolvimento de SW:
  - Imposição de **disciplina, planejamento e gerenciamento**
  - A implementação do produto deve ser **postergada até** que os objetivos tenham sido completamente entendidos
  - Permite gerência do *baseline, que identifica um conjunto fixo de documentos produzidos ao longo do processo de* desenvolvimento

# Modelo V

- Variação do Cascata
- Descreve o paralelismo entre as atividades de desenvolvimento e teste de software



# Modelo V



# Ciclo de vida Incremental



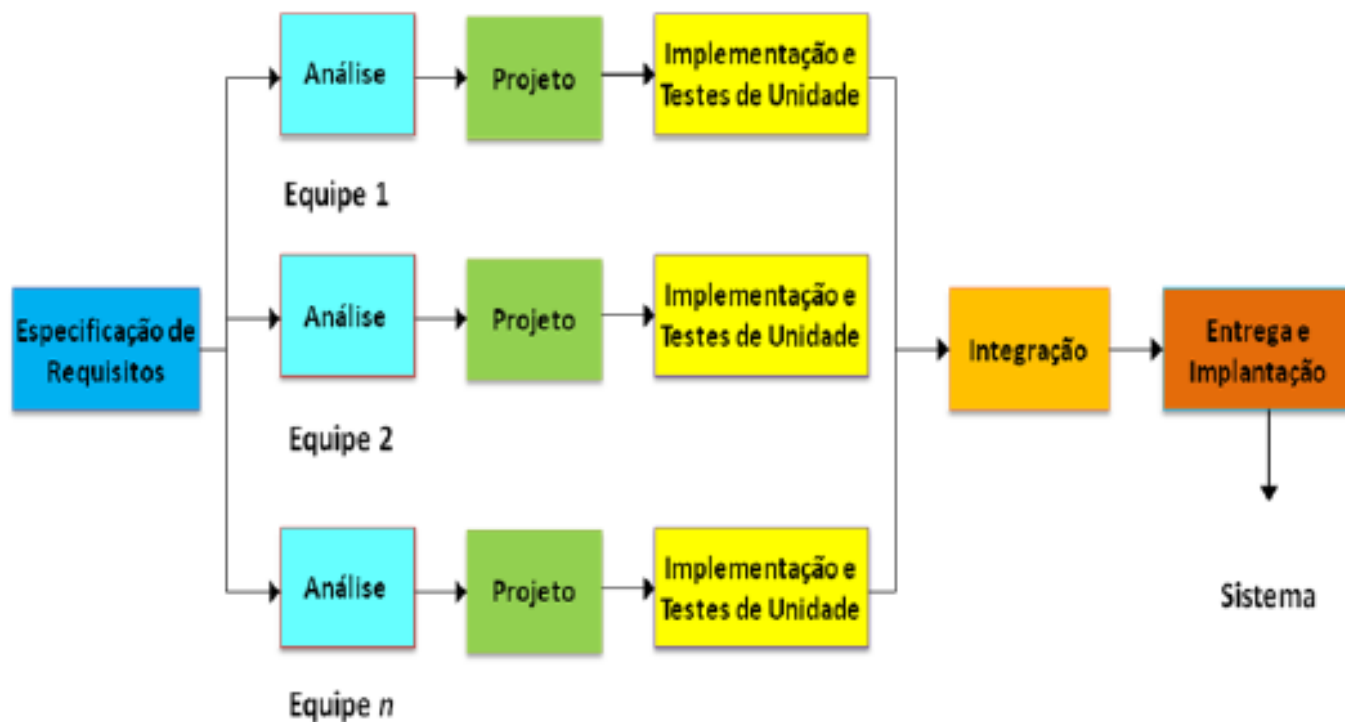
# Ciclo de vida Incremental

---

- Faz entregas incrementais do software
  - Cada incremento é construído via um mini-cascata
  - Cada incremento é um software operacional
- Versões anteriores ajudam a refinar o plano
  - Feedback constante do cliente
- Diminuição da ansiedade do cliente
  - O cliente rapidamente recebe uma versão funcional do software

# RAD (*Rapid Application Development*)

*Prima por um ciclo de desenvolvimento curto (tipicamente de até 90 dias). Os incrementos são desenvolvidos em paralelo por equipes distintas e apenas uma única entrega é feita*





# RAD (*Rapid Application Development*)

- Características:
  - Exige disponibilidade de equipes
  - Os requisitos têm de ser bem definidos, o escopo do projeto tem de ser restrito e o sistema modular.



## Ciclo de vida RAD

---

- Funcionamento equivalente ao cascata
- Principais diferenças
  - Visa entregar o sistema completo em 60 a 90 dias
  - Múltiplas equipes trabalham em paralelo na modelagem e construção
  - Assume a existência de componentes reutilizáveis e geração de código
- Difícil de ser utilizado em domínios novos ou instáveis

# Prototipação

- 2 ■ Auxilia o engenheiro de software e o cliente a entenderem melhor o que deve ser construído quando os requisitos estão confusos.
- O protótipo serve como um mecanismo para a identificação dos requisitos de sw
- Tipos de protótipos:
  - Exploratório: É descartado quando fica pronto, também chamado de protótipo para descarte.
  - Evolutivo: Gradualmente evolui para se tornar um sistema real.



# Prototipação

---

- Usualmente utilizado como auxílio a outro modelo de ciclo de vida
- Útil para
  - Validar um requisito obscuro com o cliente
  - Verificar o desempenho de um algoritmo específico
  - Gerenciar o risco do desenvolvimento
- Protótipo deveria ser jogado fora no final
  - Protótipos não são produtos
  - Usualmente os clientes desejam colocar protótipos em produção

# Prototipação

## Problemas

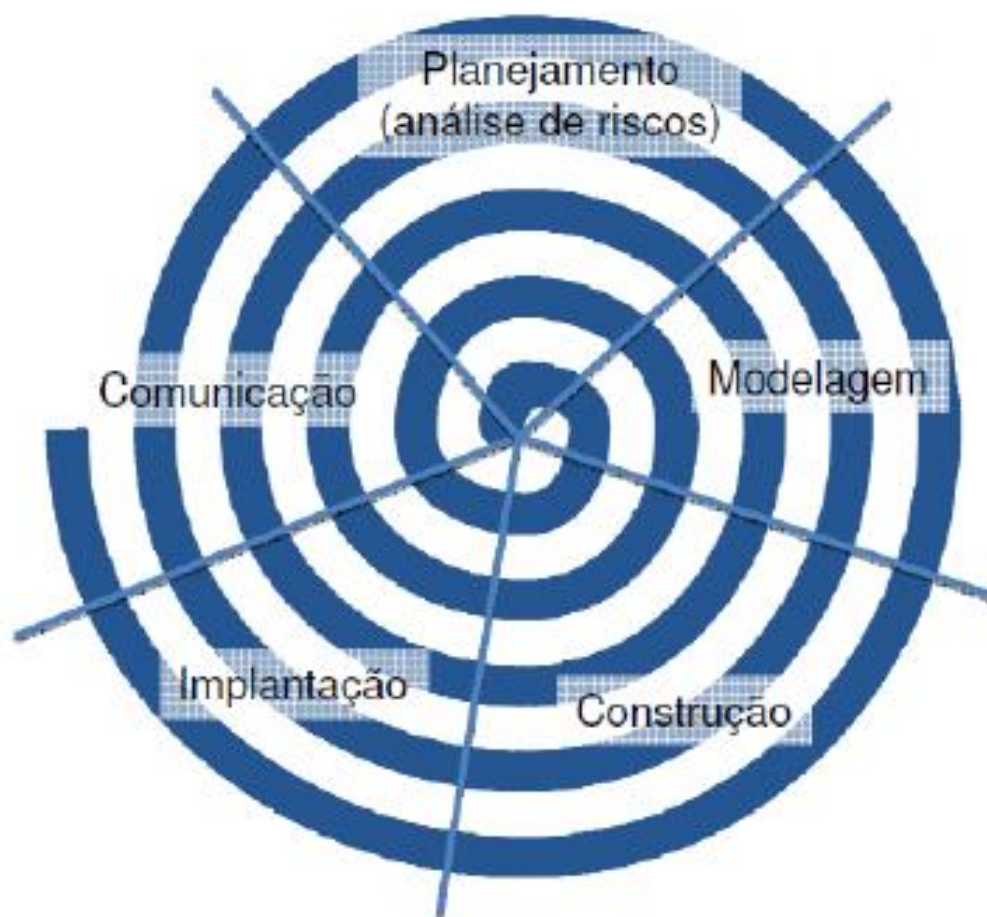
- O cliente muitas vezes não aceita mais uma iteração, aquela versão mesmo incompleta já serve.
- Não há necessidade de desenvolver uma versão final, modifica-se o protótipo.
- O desenvolvedor frequentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo.

## Solução

- Definir as regras do jogo logo no começo, o cliente deve concordar que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos

# Ciclo de vida Espiral

---



## Ciclo de vida Espiral

---

- Foco principal no gerenciamento de riscos
- A cada ciclo
  - O conhecimento aumenta
  - O planejamento é refinado
  - O produto gerado no ciclo anterior é evoluído (não é jogado fora)
- Cada ciclo evolui o sistema, mas não necessariamente entrega um software operacional
  - Modelo em papel
  - Protótipo
  - Versões do produto
  - Etc.
- O tempo no desenvolvimento pode aumentar mas ocorre a diminuição dos riscos



# Modelo em Espiral - Vantagens

- Abordagem realística para o desenvolvimento de software em grande escala
- Capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva
- Os riscos são explicitamente avaliados e resolvidos durante todo o processo
- Mantém o enfoque sistemático do ciclo clássico



# Modelo em Espiral - Desvantagens

3

- Pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável
- Requer boa capacidade para Análise de Riscos
  - Exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso

# PROCESSO UNIFICADO

34



Processo Unificado

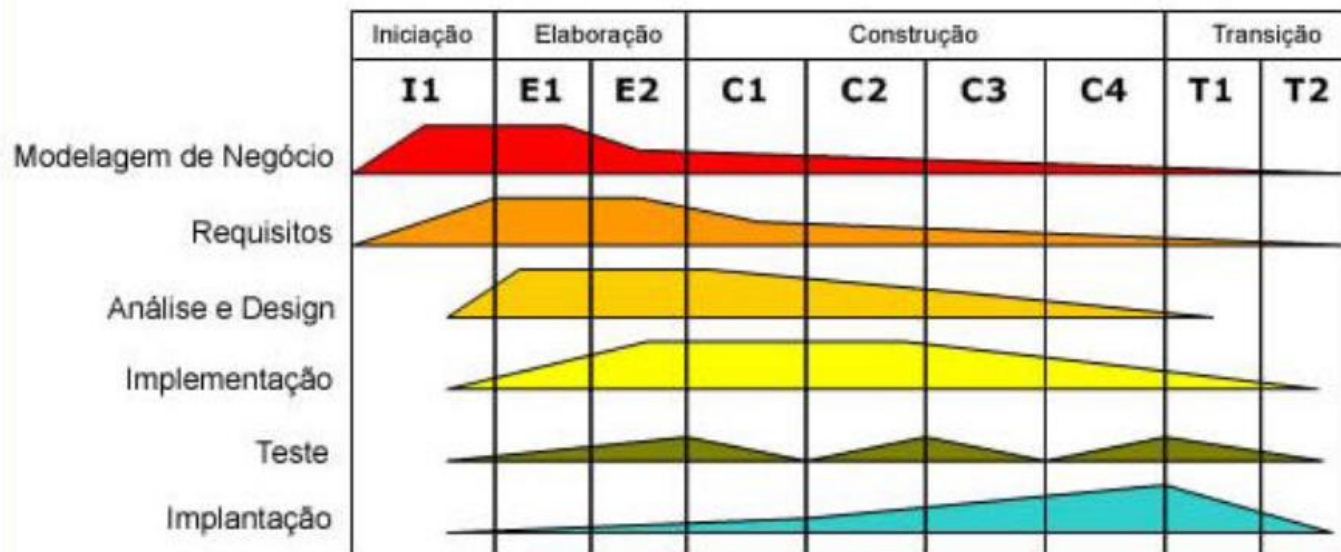
# Processo Unificado

## Características



UFC

### FASES DO RUP



# Processo Unificado

## Características



- A programação não é uma tradução mecânica do modelo para o código
- As iterações não duram meses, mas sim semanas
- O planejamento não é especulativo, mas sim refinado durante o projeto

# Processo Unificado em ação

- Em cada iteração:
  - Selecionar alguns casos de uso por ordem de prioridade para serem analisados em detalhes
  - Atribuir tarefas para a iteração a partir da análise detalhada desses casos de uso
  - Fazer projeto e programação de parte do software
  - Testar a parte do software recém projetada e programada e criar a baseline de iteração
  - Apresentar a baseline da iteração ao usuário

# Fases do Processo Unificado

- O desenvolvimento pode ser decomposto em fases, com o intuito de retratar a ênfase principal das iterações
  - Iniciação
  - Elaboração
  - Construção
  - Transição

# Fases do Processo Unificado

## Iniciação



- Identificação dos riscos
- Listagem inicial dos requisitos
- Esboço dos casos de uso
- Identificação das arquiteturas candidatas
- Estimativas iniciais de cronograma e custo

# Fases do Processo Unificado

## Iniciação



- Principais características
  - Menor fase do projeto
  - Escopo ainda vago
  - Estimativas ainda vagas
- Esforço e duração aproximadas
  - 5% do esforço do projeto
  - 10% da duração do projeto



# Fases do Processo Unificado

## Elaboração



- Mitigação dos riscos
- Detalhamento da maioria dos requisitos e casos de uso
- Estabelecimento e validação da arquitetura do software
- Detalhamento das estimativas de cronograma e custo

# Fases do Processo Unificado

## Elaboração



- Principais características
  - Grande parte das atividades de análise e projeto já concluída
  - Diminuição significativa das incertezas
  - Baseline da arquitetura é estabelecida
- Esforço e duração aproximadas
  - 20% do esforço do projeto
  - 30% da duração do projeto

# Fases do Processo Unificado

## Construção



- Implementação dos demais componentes da arquitetura
- Preparação para a implantação
- Principais características
  - Maior fase do projeto
  - Baseline de testes do produto é estabelecida
- Esforço e duração aproximadas
  - 65% do esforço do projeto
  - 50% da duração do projeto

# Fases do Processo Unificado

## Transição



- Execução de testes finais
- Implantação do produto
- Treinamento do usuário
- Principais características
  - Baseline da liberação do produto é estabelecida
- Esforço e duração aproximadas
  - 10% do esforço do projeto
  - 10% da duração do projeto

## Exercício

---

- Sua empresa foi contratada para fazer um software para um cliente. Qual é o melhor modelo de ciclo de vida para o caso abaixo?
- a) Este cliente possui os requisitos do produto que deseja muito bem especificados.
- b) Ele deseja receber o produto em etapas, i.e., deseja ver diferentes versões do produto, e não somente o produto já finalizado no prazo final de entrega.
- c) Sua empresa não está acostumada a desenvolver produtos para este domínio, por tanto obter feedback do cliente é importante, e cuidado com os riscos no desenvolvimento do produto.
- d) O prazo de entrega do produto é de 80 dias



## Cenário 1

**Objetivo:** desenvolver um sistema para acompanhamento de cirurgia cardíaca. A organização dispõe de uma quantidade adequada de desenvolvedores experientes no domínio da aplicação. O sistema pode ser modularizado. Além disso, a organização possui um conjunto de bibliotecas de componentes reutilizáveis.

- a) Qual é o melhor modelo de ciclo de vida para o caso acima? Justifique a sua resposta
- b) Pense em outro tipo de ciclo de vida para o mesmo caso. Quais seriam as mudanças necessárias para se utilizar esse novo ciclo de vida?



## Cenário 2

**Objetivo:** desenvolver um sistema para uma aplicação de comércio eletrônico. Apesar do cliente ter uma certa urgência em colocar o sistema em operação, os requisitos para o mesmo não se encontram bem definidos. O cliente se comprometeu em acompanhar o desenvolvimento. Porém, este possui dificuldades em expressar os requisitos do sistema.

- a) Qual é o melhor modelo de ciclo de vida para o caso acima? Justifique a sua resposta
- b) Pense em outro tipo de ciclo de vida para o mesmo caso. Quais seriam as mudanças necessárias para se utilizar esse novo ciclo de vida?



## Cenário 3

**Objetivo:** desenvolver um sistema de cadastro de usuários de uma biblioteca virtual. Os requisitos para o sistema foram fornecidos pelo usuário de antemão e estão relativamente bem definidos. A organização dispõe de uma quantidade adequada de desenvolvedores experientes no domínio da aplicação. Porém, há uma alta disputa interna entre a equipe de desenvolvimento.


**Possibilidades:** Cascata, Evolutivo, Espiral, Incremental, Prototipação, RAD.






No modelo espiral de Boehm, o processo de software é representado como uma espiral e não como uma sequência de atividades com retornos de uma para outra. O modelo espiral de Boehm é

- a) um *framework* de processo de software dirigido a riscos.
- b) dividido em três setores: definição de objetivos, desenvolvimento e planejamento.
- c) pouco tolerante a mudanças ao longo do processo de software.
- d) construído de forma que a volta mais externa define o início do processo de software.



No modelo espiral de Boehm, o processo de software é representado como uma espiral e não como uma sequência de atividades com retornos de uma para outra. O modelo espiral de Boehm é


- a) um *framework* de processo de software dirigido a riscos.
- b) dividido em três setores: definição de objetivos, desenvolvimento e planejamento.
- c) pouco tolerante a mudanças ao longo do processo de software.
- d) construído de forma que a volta mais externa define o início do processo de software.



O princípio fundamental é que, a cada ciclo, uma versão operacional do sistema será produzida e entregue para uso ou avaliação detalhada do cliente. Os requisitos têm de ser levantados e é preciso constatar que o sistema é modular.

Esse é o modelo


- a) Incremental.
- b) Espiral.
- c) Cascata.
- d) RAD
- e) XP



O princípio fundamental é que, a cada ciclo, uma versão operacional do sistema será produzida e entregue para uso ou avaliação detalhada do cliente. Os requisitos têm de ser levantados e é preciso constatar que o sistema é modular.


Esse é o modelo

- a) Incremental.
- b) Espiral.
- c) Cascata.
- d) RAD
- e) XP




Metodologias de desenvolvimento de software se baseiam em um modelo de ciclo de vida, tais como cascata, espiral e prototipagem; sendo assim, é correto afirmar que

- ☐ **a)** metodologias que seguem o modelo em espiral normalmente possuem um maior potencial de risco, uma vez que esse modelo não lida explicitamente com isso.
- ☐ **b)** metodologias que seguem o modelo de prototipagem devem, necessariamente, descartar os protótipos construídos; dessa forma, essas metodologias costumam ser mais custosas.
- ☐ **c)** metodologias que seguem o modelo em cascata possuem fases bem definidas, que podem ser desenvolvidas incrementalmente, em diferentes ciclos de desenvolvimento,. Isto é, a fase seguinte pode ser executada, ainda que a fase anterior não tenha sido finalizada completamente.
- ☐ **d)** metodologias que seguem o modelo em cascata possuem fases bem definidas e executadas sequencialmente. Além disso, não há sobreposição entre as fases, isto é, a fase seguinte somente pode ser executada após a finalização da fase anterior.
- ☐ **e)** em metodologias que seguem o modelo em espiral, o software é desenvolvido em apenas uma iteração.




Metodologias de desenvolvimento de software se baseiam em um modelo de ciclo de vida, tais como cascata, espiral e prototipagem; sendo assim, é correto afirmar que

- ☐ a) metodologias que seguem o modelo em espiral normalmente possuem um maior potencial de risco, uma vez que esse modelo não lida explicitamente com isso.
- ☐ b) metodologias que seguem o modelo de prototipagem devem, necessariamente, descartar os protótipos construídos; dessa forma, essas metodologias costumam ser mais custosas.
- ☐ c) metodologias que seguem o modelo em cascata possuem fases bem definidas, que podem ser desenvolvidas incrementalmente, em diferentes ciclos de desenvolvimento. Isto é, a fase seguinte pode ser executada, ainda que a fase anterior não tenha sido finalizada completamente.
- ☒ d) metodologias que seguem o modelo em cascata possuem fases bem definidas e executadas sequencialmente. Além disso, não há sobreposição entre as fases, isto é, a fase seguinte somente pode ser executada após a finalização da fase anterior.
- ☐ e) em metodologias que seguem o modelo em espiral, o software é desenvolvido em apenas uma iteração.



A principal metodologia tradicional utilizada no desenvolvimento de *software* é o modelo clássico também conhecido como cascata ou sequencial. Nesse modelo,

- ☐ **a)** cada etapa tem associada ao seu término uma documentação que deve ser aprovada para que a etapa posterior possa ter início.
- ☐ **b)** o projeto é dividido em fases de maneira flexível.
- ☐ **c)** o custo das alterações do *software* diminui à medida que o desenvolvimento progride.
- ☐ **d)** utiliza-se o desenvolvimento incremental e iterativo.
- ☐ **e)** os requisitos não podem ser estáveis.




A principal metodologia tradicional utilizada no desenvolvimento de *software* é o modelo clássico também conhecido como cascata ou sequencial. Nesse modelo,

- ☒ **a)** cada etapa tem associada ao seu término uma documentação que deve ser aprovada para que a etapa posterior possa ter início.
- ☐ **b)** o projeto é dividido em fases de maneira flexível.
- ☐ **c)** o custo das alterações do *software* diminui à medida que o desenvolvimento progride.
- ☐ **d)** utiliza-se o desenvolvimento incremental e iterativo.
- ☐ **e)** os requisitos não podem ser estáveis.



A etapa do projeto unificado e a sua correspondente característica são, respectivamente:

- A**      Concepção – levantamento de requisitos sistêmicos primários do ciclo
- B**      Construção – implementação dos elementos de maior risco e criticidade
-  **C**      Elaboração – mitigação dos problemas de alto risco do projeto
- D**      Incremento – diferenciação entre as entregas de duas etapas subsequentes
- E**      Transição – geração de um subconjunto executável do produto final

A etapa do projeto unificado e a sua correspondente característica são, respectivamente:

- ☐ A Concepção – levantamento de requisitos sistêmicos primários do ciclo
- ☐ B Construção – implementação dos elementos de maior risco e criticidade
- ☒ C Elaboração – mitigação dos problemas de alto risco do projeto
- ☐ D Incremento – diferenciação entre as entregas de duas etapas subsequentes
- ☐ E Transição – geração de um subconjunto executável do produto final

# O que é agilidade?

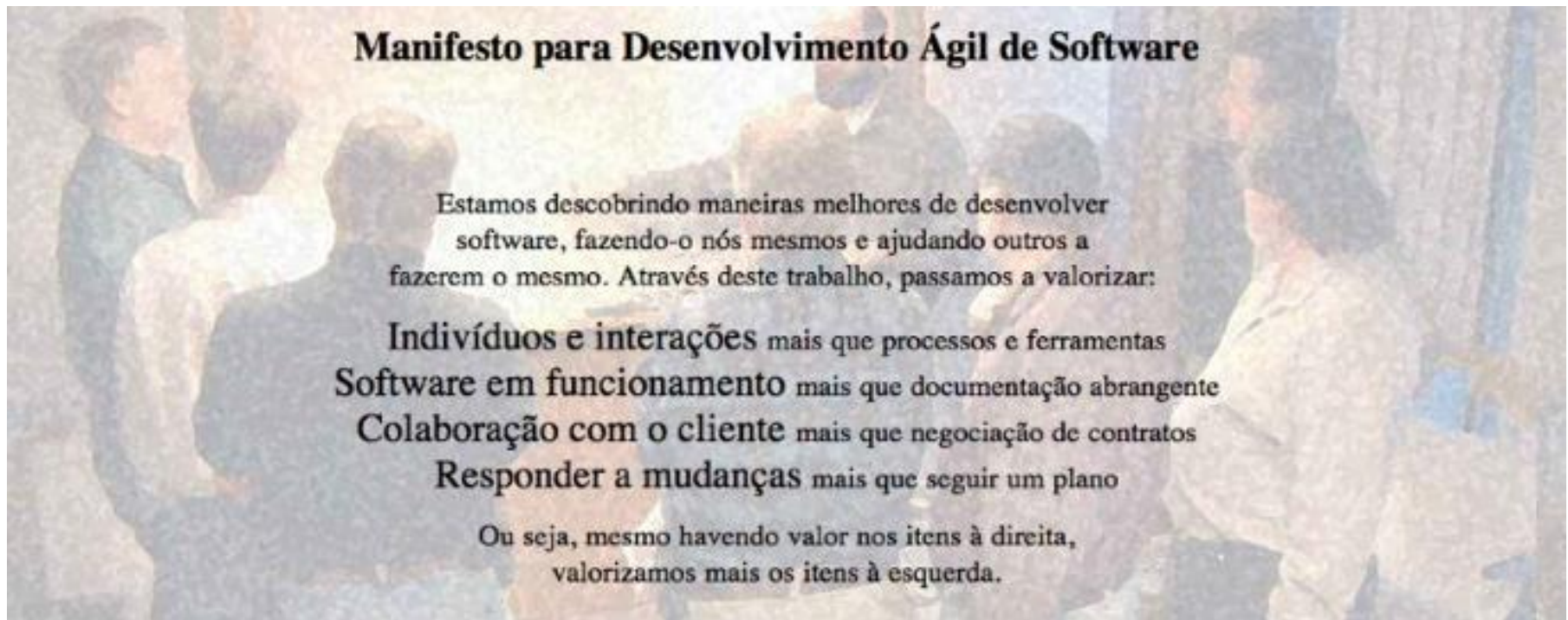
59

Movimento que surgiu em meados dos anos 90 em resposta aos pesados métodos de gerenciamento de desenvolvimento de *software* que predominavam na época.



# Como surgiram as metodologias ágeis?

- Em 2001 Kent Beck, e mais 16 reconhecidos desenvolvedores, se reuniram para definirem o ***Manifesto Ágil***.





# Como surgiram as metodologias ágeis?

□ En  
de  
M



los  
o

# Idéias defendidas pelo manifesto ágil

Indivíduos e  
interações

Software que  
funciona

Colaboração do  
cliente

Resposta à  
mudanças

ao  
invés  
de

Processos e  
ferramentas

Documentação  
abrangente

Negociação de  
contrato

Seguir um plano

# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva
3. Software funcionando
4. Negócio e desenvolvedores juntos
5. Motivação
6. Cara-a-cara
7. Medindo o progresso
8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva

<http://blog.myscrumhalf.com/tag/principios-manifesto-agil/page/2/>

# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva

**“Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado”**

8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva





# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva
3. Software funcionando

**“Receber bem as mudanças dos requisitos,  
mesmo em estágios tardios do  
desenvolvimento”**

8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva
3. Software funcionando
4. Negócio e desenvolvedores juntos

**“Trabalhando para entregar software, em intervalos de 2 semanas até 2 meses”**

9. Excelência técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva
3. Software funcionando
4. Negócio e desenvolvedores juntos
5. Motivação

**“Empresários e desenvolvedores  
devem trabalhar juntos diariamente  
durante todo o projeto”**

11. Equipes auto organizáveis
12. Retrospectiva



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva
3. Software funcionando
4. Negócio e desenvolvedores juntos
5. Motivação
6. Cara-a-cara
7. Medindo o progresso

**“Construa projetos com indivíduos motivados, dê-lhes o ambiente e o suporte que precisam, e confie neles para ter o trabalho realizado”**



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva
3. Software funcionando
4. Negócio e desenvolvedores juntos
5. Motivação
6. Cara-a-cara
7. Medindo o progresso

**“O método mais eficiente e efetivo de transmitir informação para a equipe de desenvolvimento está na conversa cara-a-cara”**



# Princípios básicos do Manifesto Ágil

## 1. Entrega contínua

**“Software funcionando é a principal medida de progresso”**

- 6. Cara-a-cara
- 7. Medindo o progresso
- 8. Ritmo sustentável
- 9. Excelência Técnica
- 10. Simplicidade
- 11. Equipes auto organizáveis
- 12. Retrospectiva



# Princípios básicos do Manifesto Ágil

## 1. Entrega contínua

Vantagem competitiva

**“Processos ágeis promovem o desenvolvimento sustentável.”**

## 3. Menor risco

## 6. Cara-a-cara

## 7. Medindo o progresso

## 8. Ritmo sustentável

## 9. Excelência Técnica

## 10. Simplicidade

## 11. Equipes auto organizáveis

## 12. Retrospectiva



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva

**“Atenção contínua à excelência técnica e bom design aumenta a agilidade”**

3. Colaboração com o cliente
4. Resposta rápida a mudanças
5. Trabalho com indivíduos
6. Comunicação face a face
7. Medindo o progresso
8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva





# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva

**“Simplicidade – a arte de maximizar a quantidade de trabalho não feito – é essencial.”**

3. Colaboração com o cliente
4. Resposta rápida
5. Comunicação face a face
6. Funcionando juntos
7. Medindo o progresso
8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva

**"As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis"**

3. Colaboração com o cliente
4. Resposta rápida a mudanças
5. Trabalho com indivíduos e equipes
6. Comunicação direta
7. Medindo o progresso
8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva



# Princípios básicos do Manifesto Ágil

1. Entrega contínua
2. Vantagem competitiva

**“Em intervalos regulares, as equipes devem refletir sobre como tornaram-se mais efetivas, em seguida aprimorar e ajustar de acordo com seu comportamento.”**

3. Colaboração com o cliente
4. Colaboração com o time
5. Resposta rápida
6. Trabalho com pessoas
7. Medindo o progresso
8. Ritmo sustentável
9. Excelência Técnica
10. Simplicidade
11. Equipes auto organizáveis
12. Retrospectiva



# Atividade

Os **métodos ágeis** são um **CONJUNTO DE PRÁTICAS** eficazes que se destinam a permitir a entrega rápida e de alta qualidade do produto, tendo uma abordagem de negócios que alinha o desenvolvimento do projeto com as necessidades do cliente e os objetivos da empresa.

- **Fundamentação Teórica**
- **Papéis (se houver)**
- **Práticas**
  - prática é a ação que se desenrola com a aplicação de certos conhecimentos
- **Ciclo de Vida**
- **Relação com outras metodologias ágeis**
- **Ferramentas**
- **Atualidade – que empresas de TI ou não utilizam**
- **Outros**

# Atividade

**Façam um mapa mental no papel**

- **Individual**
- **Vale 1,0 bônus que pode ser utilizado na prova 1**
- **Entrega: Terça-feira (29/08) – 10h.**

**Na aula do dia 29/08 vamos começar com uma discursão sobre os temas**

- **Quem não participar, irá diminuir a nota do bônus**

# Atividade

## Metodologia Ágil

1 - TDD – Test-Driven Development ou Desenvolvimento Orientado a Testes

2 - BDD - Behaviour Driven Development ou Desenvolvimento orientado por comportamento

3 - XP - Extreme Programming

4 - ASD - Adaptive Software Development OU Desenvolvimento de Software Adaptativo

5 - DSDM - Dynamic Systems Development Method ou Desenvolvimento dirigido à funcionalidade

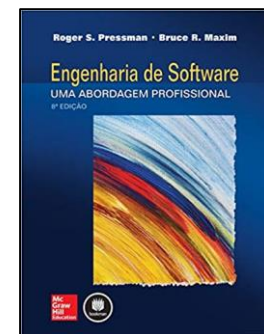
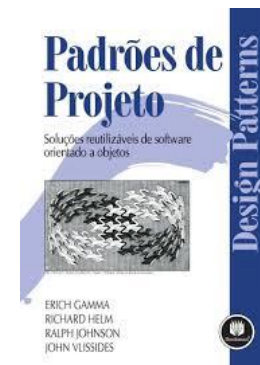
6 - Família Crystal de Cockburn

7- LSD - Desenvolvimento de Software Enxuto ou Lean Software Development

8 - FDD - Feature-Driven Development ou Desenvolvimento Guiado por Funcionalidades

# Referências Bibliográficas

- LARMAN, Craig. **Utilizando UML e padrões**: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007. 695 p. ISBN 9788560031528 (broch.).
- GAMMA, Erich. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2005. ISBN: 9788573076103.
- MARTIN, Robert C.; MARTIN, Micah. **Princípios, padrões e práticas ágeis em C#**. Porto Alegre: Bookman, 2011. 735 p. ISBN 9788577808410 (broch.).
- .



# Obrigada!



☐ Perguntas?

E-mail: [jacilane.rabelo@ufc.br](mailto:jacilane.rabelo@ufc.br)