

# Engenharia de Software II

## Processos de Manutenção

Prof. André Hora  
DCC/UFMG  
2019.1

# Agenda

- 1. Manutenção em modelos tradicionais e ágeis**
2. Modelos de manutenção
3. Atividades de manutenção

# Modelos Tradicionais

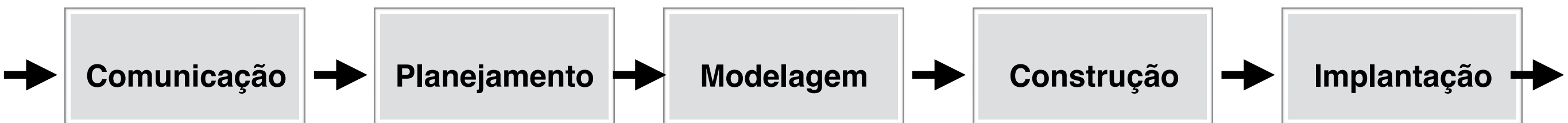
- Também conhecidos como **modelos prescritivos**
- Modelos:
  - Cascata
  - Incremental
  - Evolucionário

# Modelos Tradicionais

1. **Cascata**
2. Incremental
3. Evolucionário

# Modelo Cascata

- Executa atividades de forma **sequencial**
- Começa na comunicação e termina na implantação
- **Difícil** de ocorrer em **projetos reais** atualmente



# Modelo Cascata

Quando usar?

- Requisitos do problema são **bem entendidos, definidos e estáveis**
- Quando há **pouca chance** dos requisitos **mudarem**
- Sistemas críticos
- Ex: melhoria de sistemas através da adição de novas funcionalidades



# Modelo Cascata

## Pontos fortes

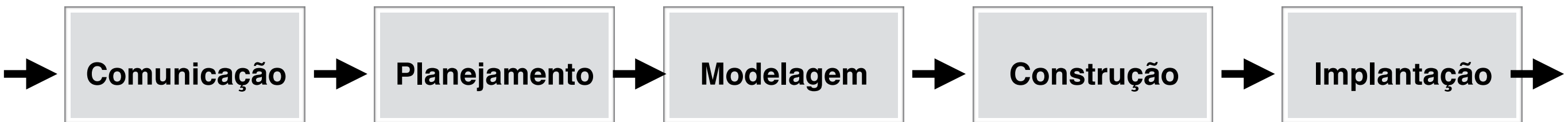


- Paradigma +antigo, estável e testado
- Documentação robusta em cada atividade
- Pode ser combinado com outros modelos
- Simples; atividades são claras e bem definidas



# Modelo Cascata

**Pontos fracos ?**



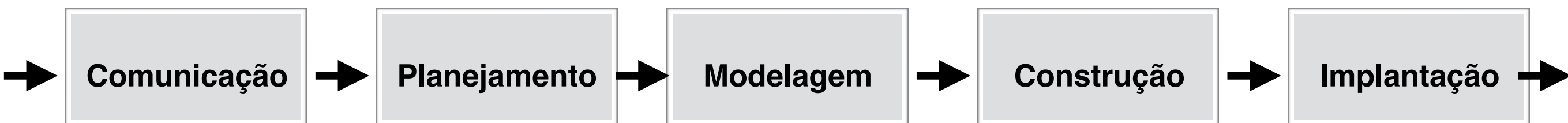


# Modelo Cascata

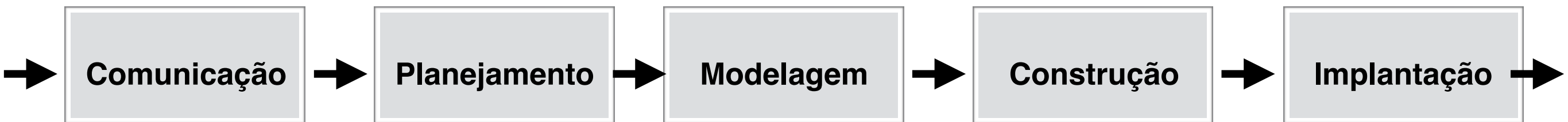
## Pontos fracos



- Projetos reais raramente seguem um fluxo sequencial
- Difícil para o cliente definir todos os requisitos antes
- Clientes devem esperar (versão do produto somente ficará pronta no final do projeto)
- Difícil para lidar com mudanças inevitáveis de requisitos
- Atrasos em fase reflete nas demais

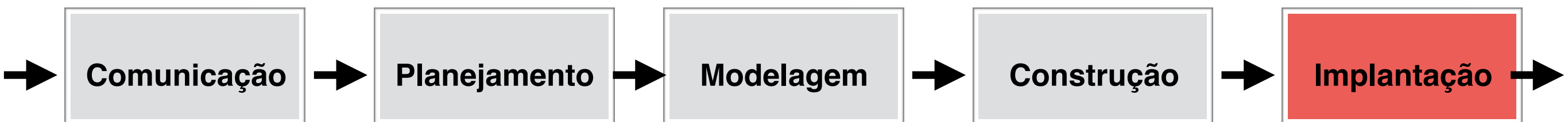


# Manutenção



# Manutenção

- **Manutenção** é realizada apenas no final do ciclo de vida



# Modelos Tradicionais

1. Cascata
2. **Incremental**
3. Evolucionário

# Modelo Incremental

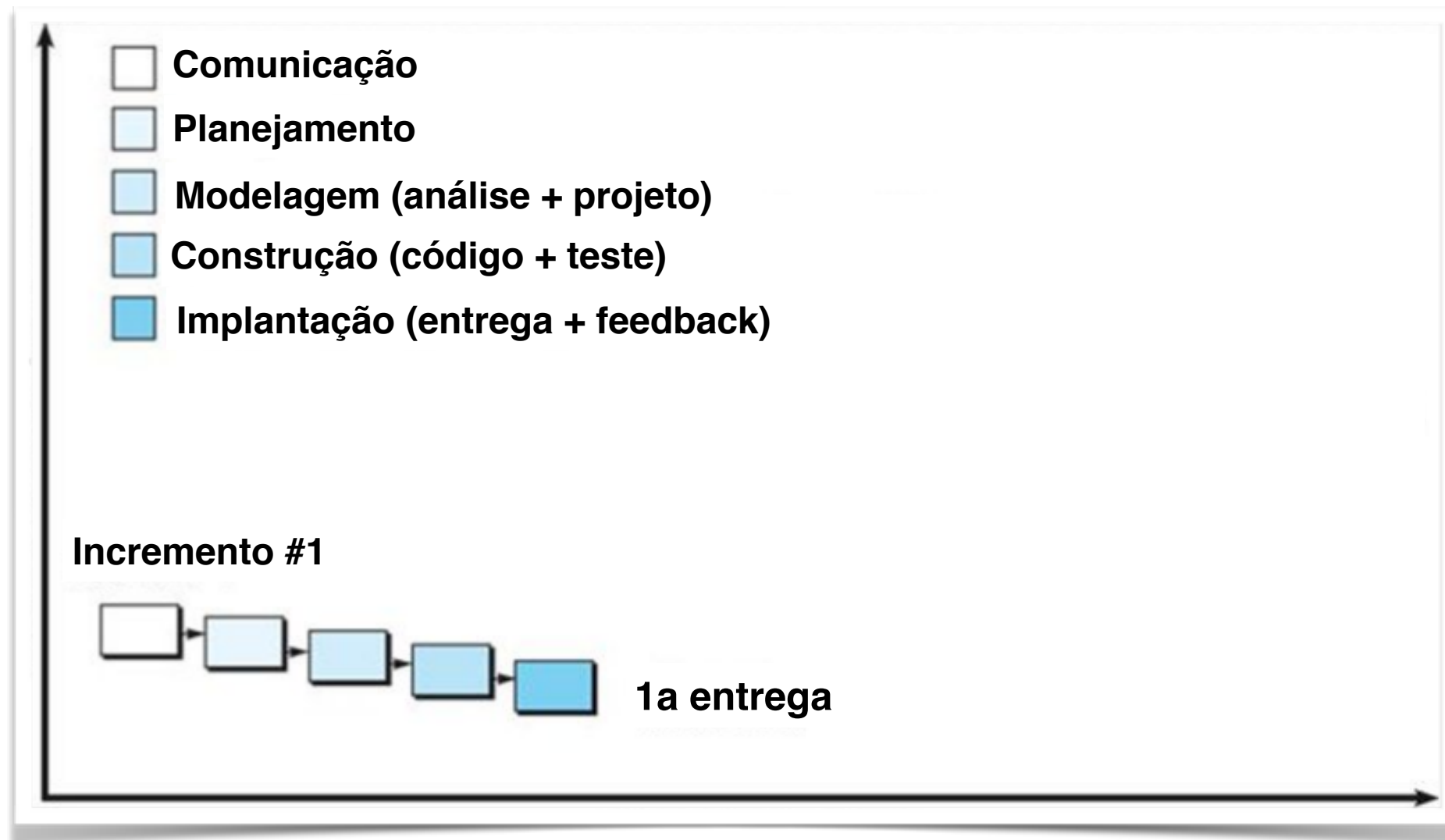
- Combina elementos do modelo **Cascata (fluxo linear) & fluxo paralelo**
- Cada sequência produz um **incremento** (isto é, um produto operacional)
- Exemplo: Processador de Texto
  - 1a entrega: abrir, editar e salvar documentos (núcleo do produto)
  - 2a entrega: undo, redo, copiar, colar
  - 3a entrega: corretor ortográfico
  - 4a entrega: configuração de layout

# Modelo Incremental

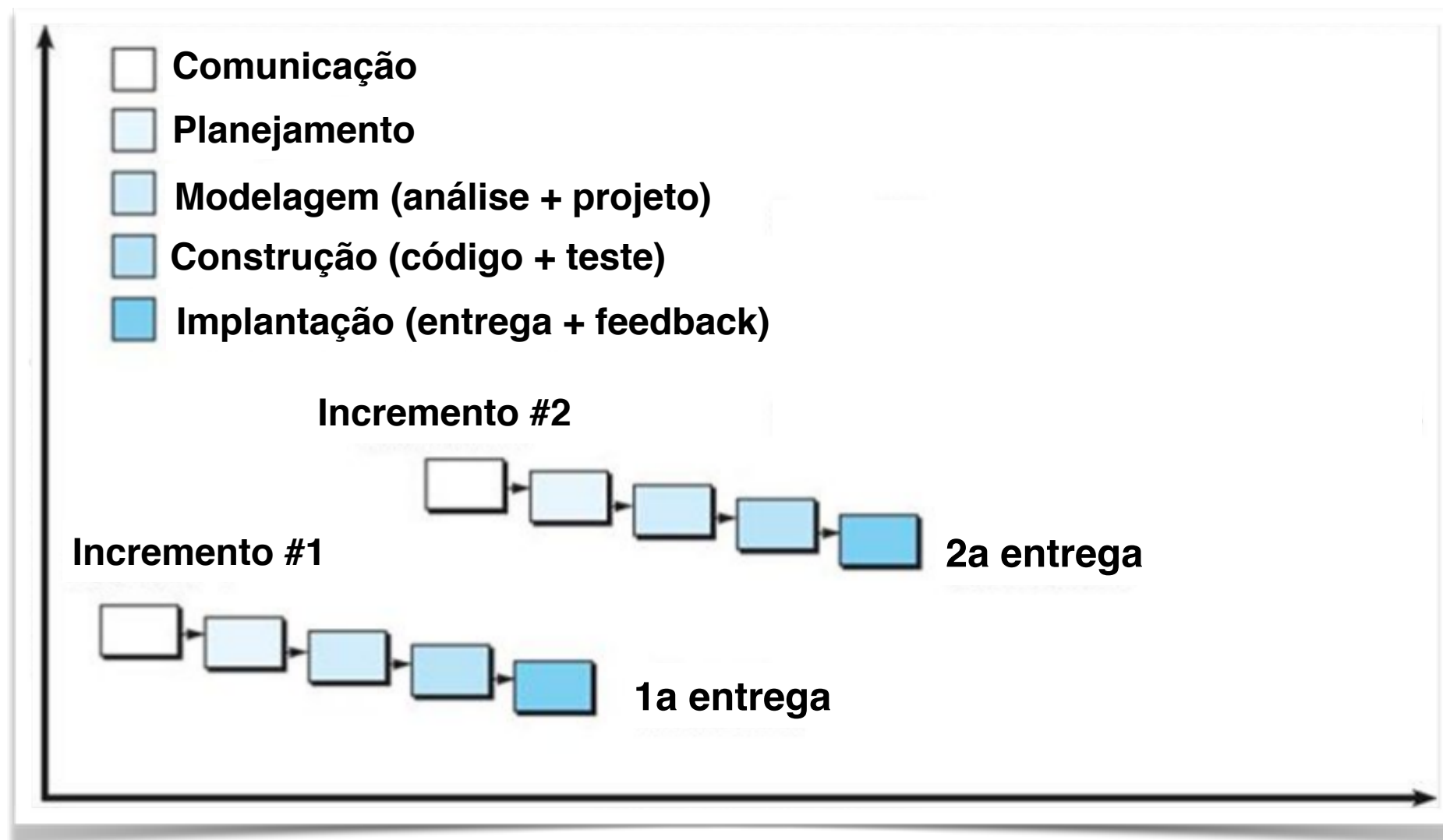
Quando usar?

- Requisitos de software estão **razoavelmente** bem definidos
- Escopo de desenvolvimento **não pode ser puramente linear**
- Necessidade de fornecer número limitado de **funcionalidades** para usuários **rapidamente**

# Modelo Incremental

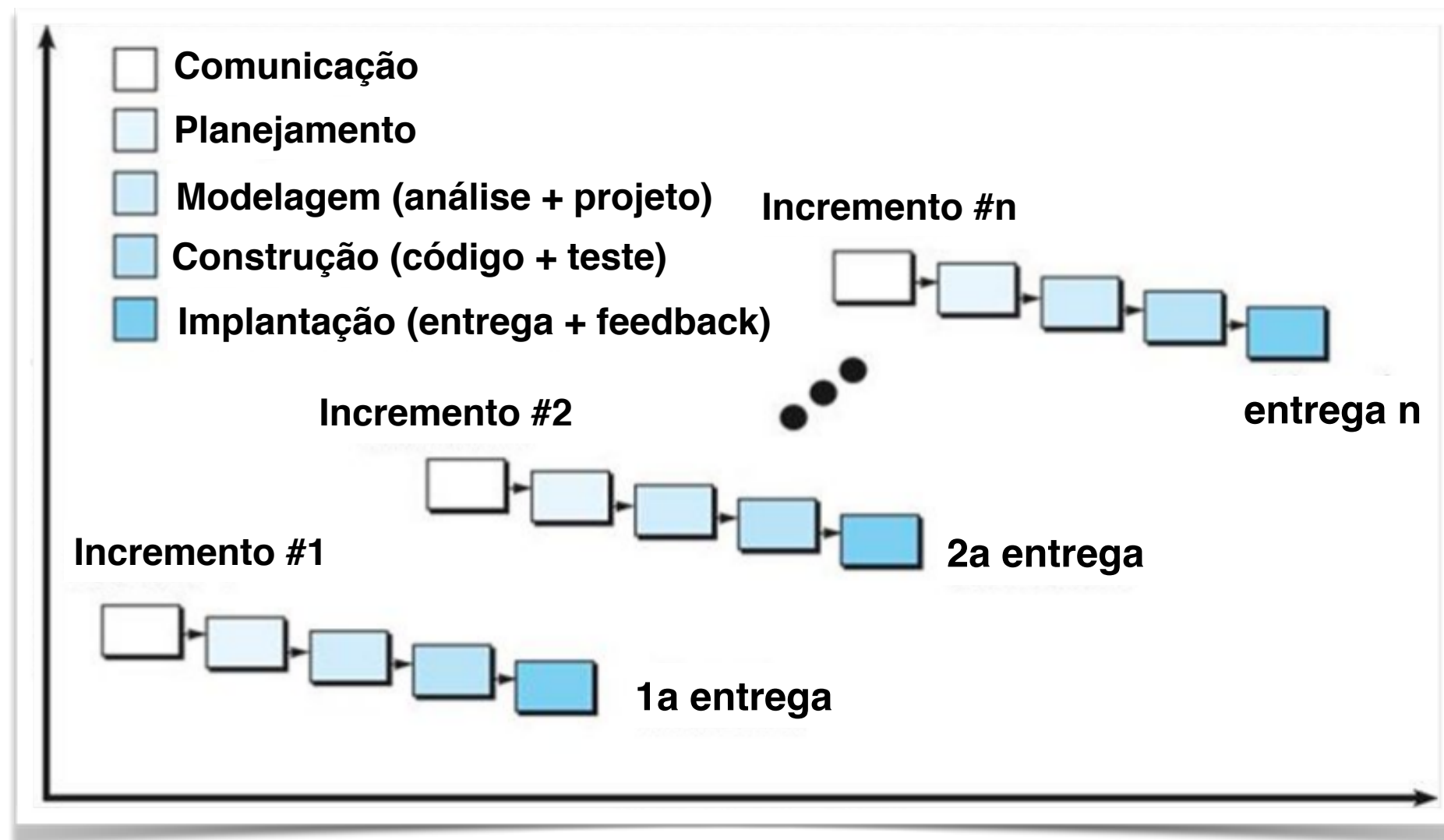


# Modelo Incremental





# Modelo Incremental



# Modelo Incremental

## Pontos fortes



- Custo reduzido para acomodar mudanças
- Feedback do cliente
- Cliente acompanha evolução do sistema (desde o início do projeto)
- Início do sistema pelas partes melhor entendidas
- Riscos críticos são resolvidos antes de grandes investimentos

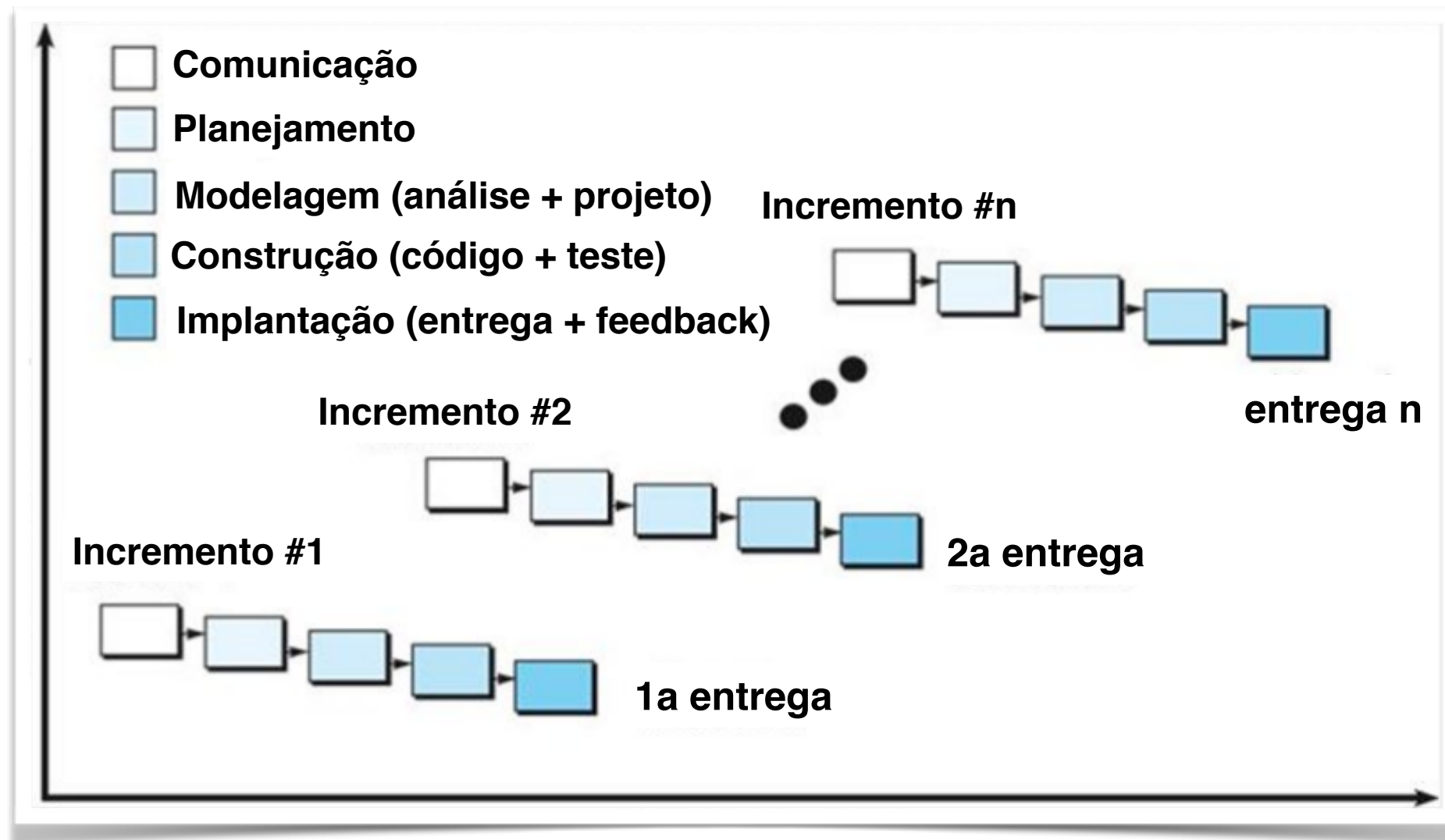
# Modelo Incremental

## Pontos fracos



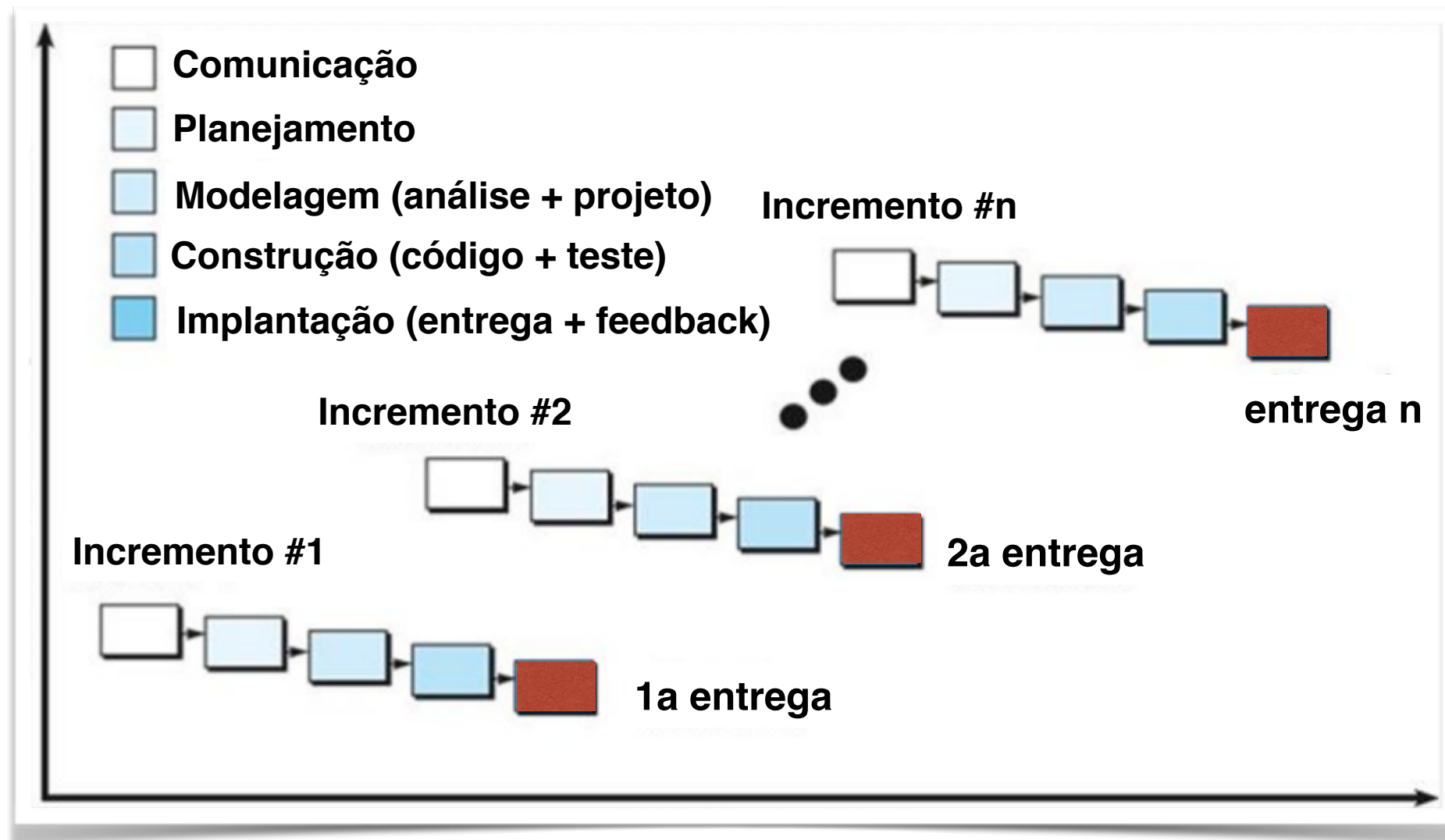
- Cuidado ao definir o incremento para que não se aproxime do Modelo Cascata
- Difícil gerência de software pois o sistema não é completamente especificado anteriormente
- Produto pode se corromper com novos incrementos e se tornar mal estruturado

# Manutenção



# Manutenção

- **Manutenção** é realizada apenas no final de cada incremento



# Modelos Tradicionais

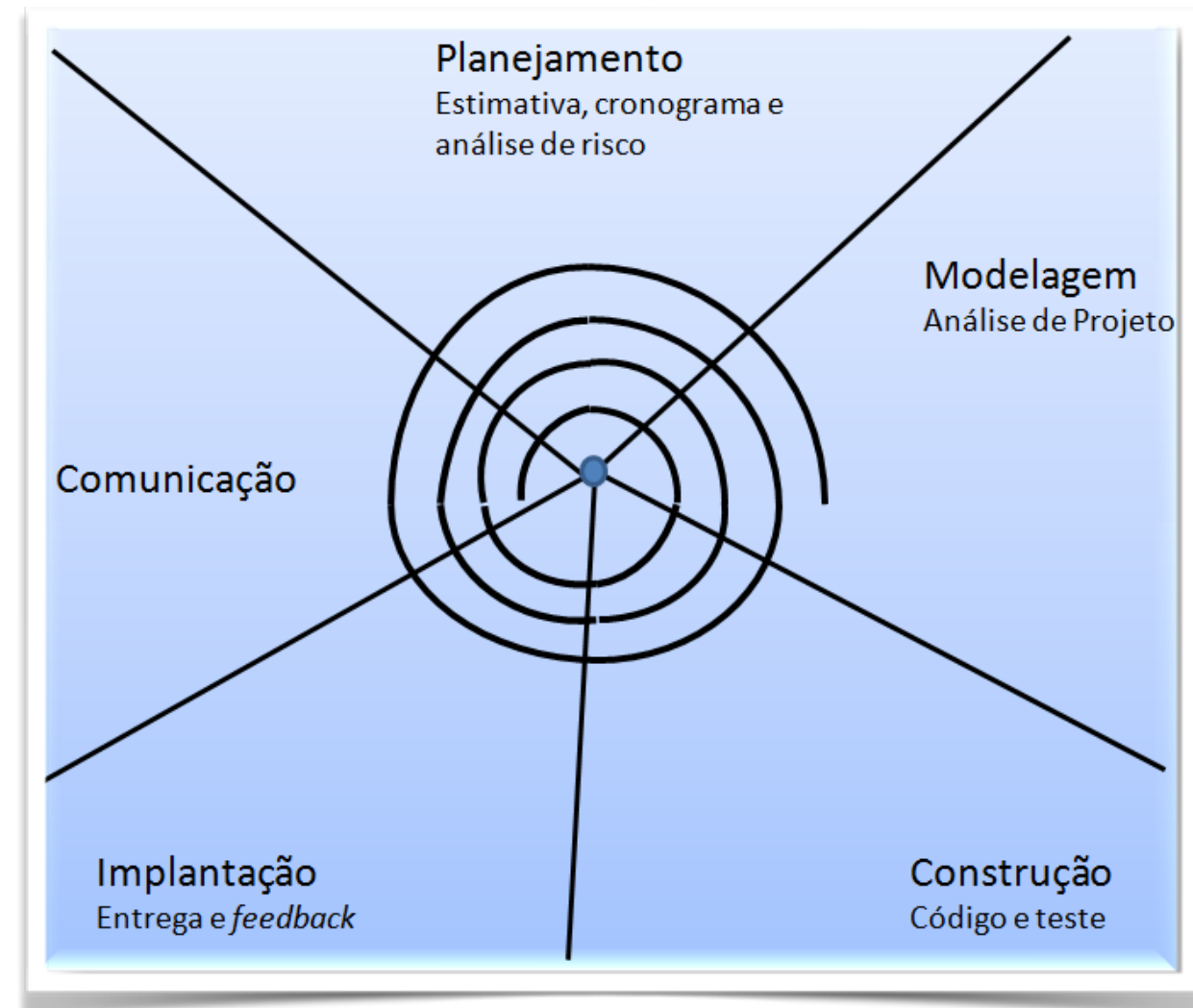
1. Cascata
2. Incremental
3. **Evolucionário**

# Modelo Evolucionário

- Modelos evolucionários são **iterativos** (repete atividades)
- Entregam uma **versão mais completa** do sistema a cada iteração (curto prazo)
- Produtos **crescem** e **mudam** com o tempo
- Algumas funcionalidades são bem entendidas, mas **detalhes precisam ser definidos**

# Modelo Evolucionário: Espiral

- Combina a natureza iterativa da Prototipação + aspectos sistemáticos do Cascata
- Possibilita o desenvolvimento rápido de versões mais completas
- Prototipação no início e incremental depois






# Manutenção

- Assim como ocorre no modelo Incremental, a **manutenção** deve ocorrer ao final de cada incremento

# Comparação: Modelos Tradicionais

Propriedade	Cascata	Incremental	Espiral
Especificação dos req.	sim, todos	não todos e mudam	não todos e mudam
Planejamento anterior	sim	sim	sim
Retorna para fase anterior	não	sim	sim
Software de grande porte	não apropriado	não apropriado	apropriado
Flexibilidade p/ mudança	difícil	fácil	fácil
Envolvimento do usuário	somente no início	intermediário	alto
Fluxo	linear	linear + paralelo	iterativo + evol.
<b>Manutenção</b>	<b>no final</b>	<b>no final de cada incremento</b>	<b>no final de cada incremento</b>

# Desenvolvimento Ágil

- Lida com as fraquezas na **ES convencional**
  - Diversos benéficos, mas não aplicáveis a todos os casos
- Na economia moderna: **difícil** prever como um sistema vai evoluir
  - Condições de mercado mudam rapidamente, necessidades dos usuários evoluem, novos competidores aparecem
- Mudança custa caro 
- Característica principal: **reduzir os custos** de mudanças no processo de software

# Processos Tradicionais x Ágeis

Diferenças ?

# Processos

## Tradicionais x Ágeis (1)

- **Processos tradicionais:** custo de mudança aumentam de forma não linear com o tempo
  - Fácil mudar na etapa inicial (requisitos)
  - Custo de mudança aumenta nas demais etapas
- **Métodos ágeis:** reduzem custo de mudança
  - Software entregue em incrementos junto com aplicação de práticas ágeis
  - Mudanças são melhor controladas

# Processos

## Tradicionais x Ágeis (2)

- **Processos tradicionais:** especificações detalhadas de requisitos, projeto e teste
- **Métodos ágeis:** objetivo de criar software útil rapidamente
  - Não se preocupam com documentação completa em todas as fases

# Boas Práticas para Manutenção

- Testes de regressão
- Integração contínua
- Refatoração
- Programação em pares
- Cobertura de teste

# Desenvolvimento Ágil

- Como a manutenção é realizada em um ambiente de desenvolvimento ágil?
- Mais informações:
  - Para o Ciclo de Manutenção, Agilidade Já! (IBM)
  - <https://www.ibm.com/developerworks/br/local/rational/smarter/smarter-pdf.pdf>



# Agenda

1. Manutenção em modelos tradicionais e ágeis
- 2. Modelos de manutenção**
3. Atividades de manutenção

# Modelos de Manutenção

- Modelo Quick-Fix
- Modelo de melhoria iterativa
- Modelo de Taute

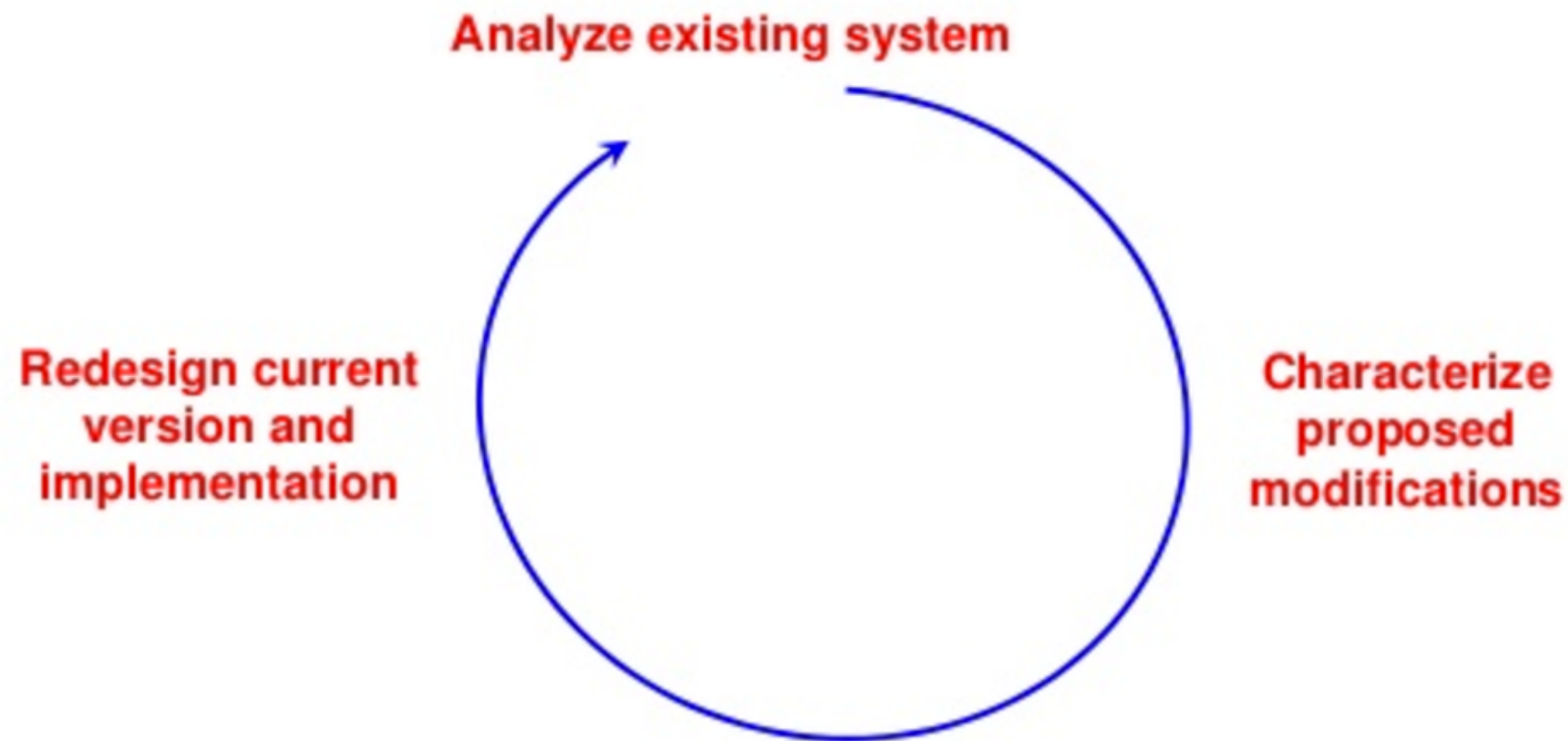
# Modelo Quick-Fix



# Modelo Quick-Fix

- Abordagem adhoc (informal)
- Espera o problema ocorrer para resolver o mais rápido possível
- Mudanças são realizadas sem análises detalhadas
- Pode funcionar se sistema é mantido por apenas uma pessoa ou em ambientes que com pressão (mas com alto risco para o longo prazo)

# Modelo de Melhoria Iterativa

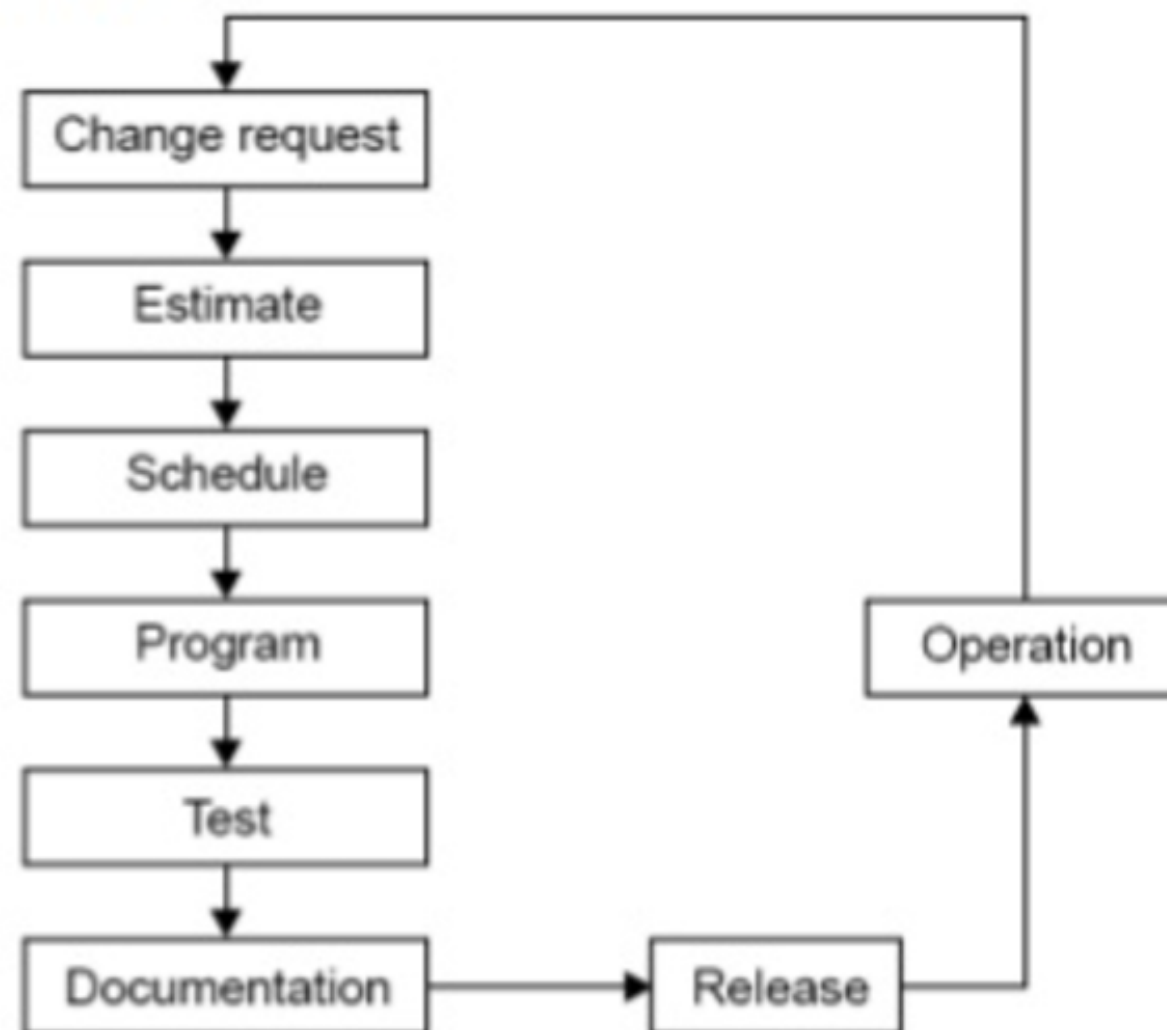


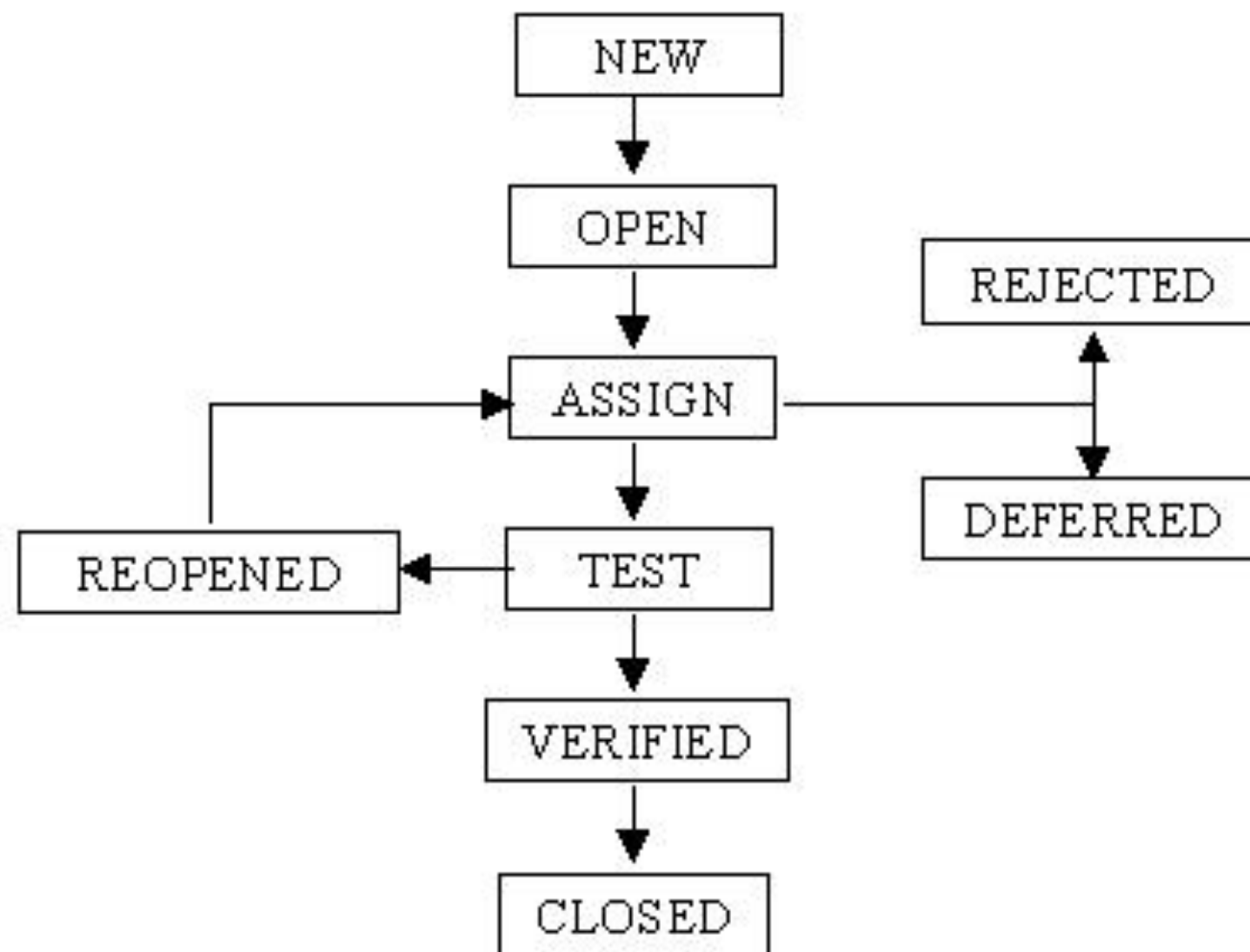
# Modelo de Melhoria Iterativa

- Foca em melhorar o sistema de forma iterativa
- Originalmente proposto como um modelo de desenvolvimento
- Adaptado para manutenção
- Estágios:
  - Análise
  - Caracterização da proposta de modificação
  - Novo projeto e implementação

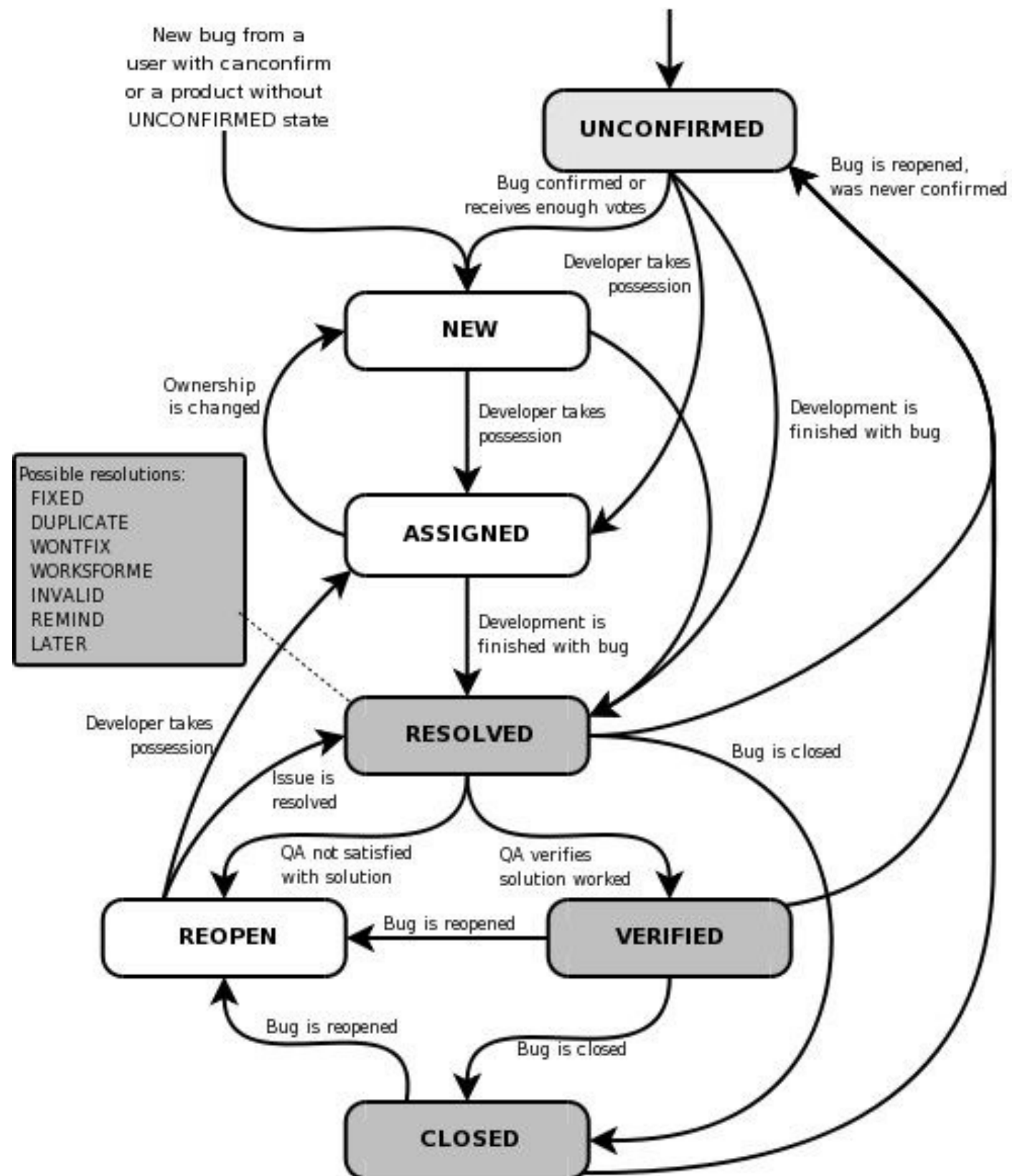
# Modelo de Taute

- Modelo de manutenção com 8 fases



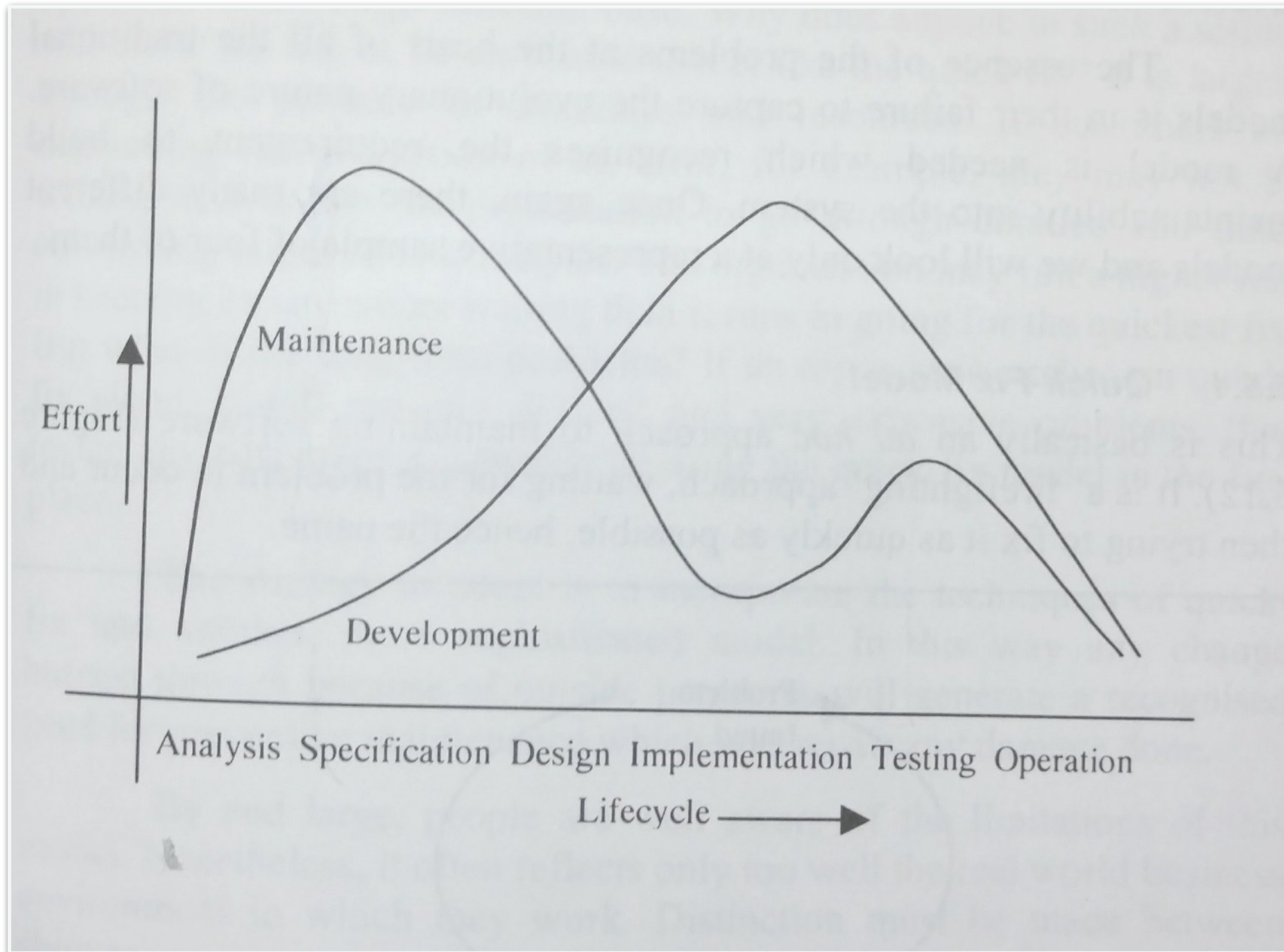






# Exercício

Interpretar o gráfico abaixo sobre esforço de Desenvolvimento x Manutenção



# Agenda

1. Manutenção em modelos tradicionais e ágeis
2. Modelos de manutenção
- 3. Atividades de manutenção**

# Atividades de Desenvolvimento x Manutenção

- Processos de manutenção possuem as atividades e técnicas necessárias para modificar um sistema existente
- Muitas atividades da manutenção são similares as atividades do desenvolvimento (ex: projeto, implementação, etc)
- Algumas práticas são específicas da manutenção

# Exercício

- Apresente práticas de desenvolvimento específicas da manutenção

# Práticas de Manutenção

- Compreensão de programas: atividades para entender o software existente
- Refatoração de código
- Solicitação de mudanças: equipe para estimar priorização, custo (ex: tamanho, esforço, complexidade)
- Análise de impacto: áreas impactadas pela mudança em potencial

# Atividades de Suporte à Manutenção

- Gerência de configuração
- Verificação, validação e teste
- Garantia de qualidade (métricas)

# Técnicas de Manutenção

- Reengenharia
- Engenharia de Reversa
- Migração