# PROCESSOS DE SOFTWARE

Atividades para especificar, projetar, implementar e testar sistemas de software

Profa. Maria Auxiliadora

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Processos de software

## Uma Visão Genérica: 3 Fases

- **Definição - "o que"**
  - **Engenharia do Sistema**
  - **Planejamento do Projeto**
  - **Análise de Requisitos**
- **Desenvolvimento - "como"**
  - **Projeto**
  - **Geração do Código**
  - **Teste**
- **Manutenção**



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Processos de software

- Um **processo de software** é um método para desenvolver ou produzir software.

- Define **quem faz o *que*, *quando* e como**.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Processos de software

**O processo é o instrumento capaz de responder a qualquer momento:**

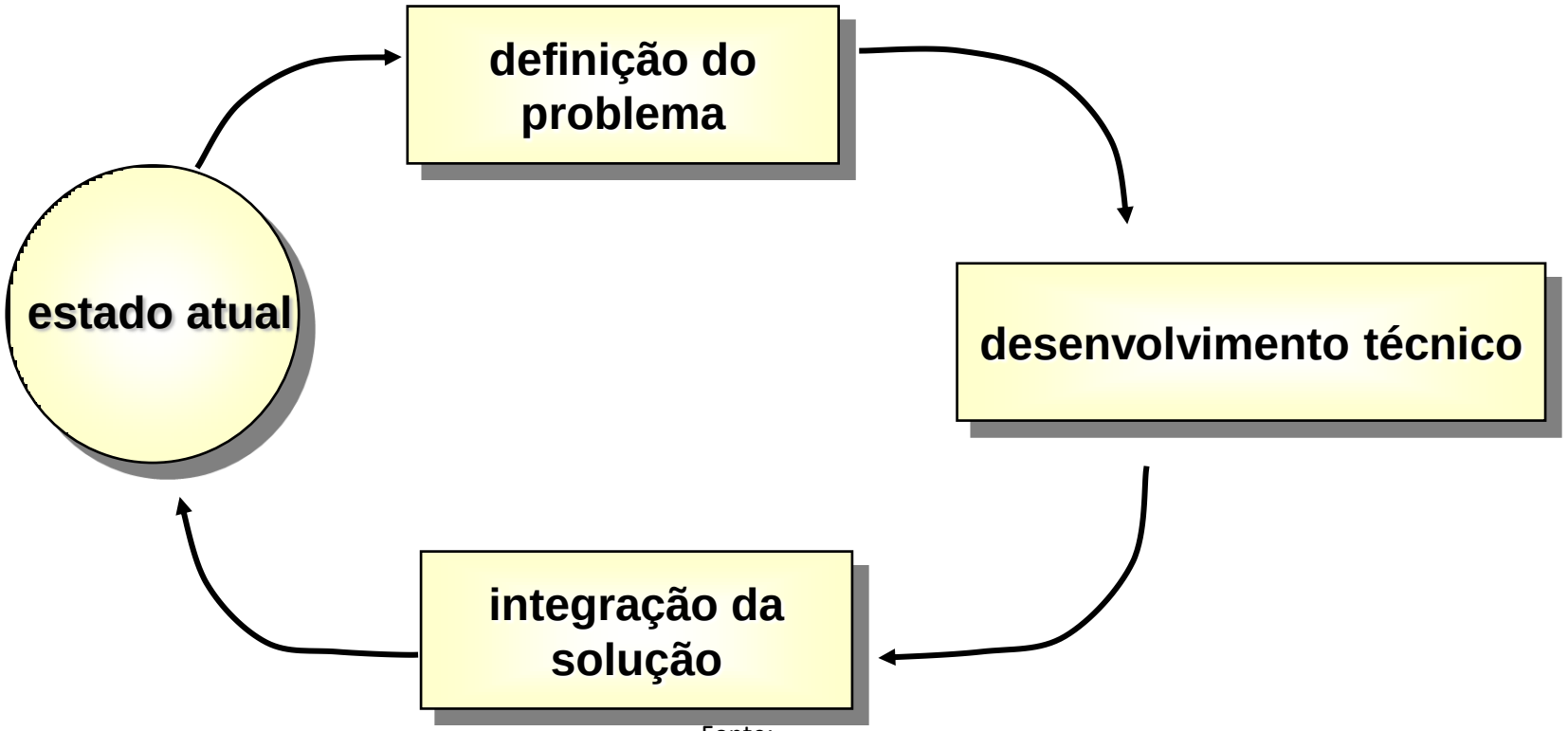
- **O que é feito?** → Produto
- **Como é feito?** → Passos
- **Por quem é feito?** → Agente
- **O que usa?** → Insumos
- **O que Produz?** → Resultados



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo de Processo de Software

**Processo** → deve incorporar uma *estratégia* de desenvolvimento




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

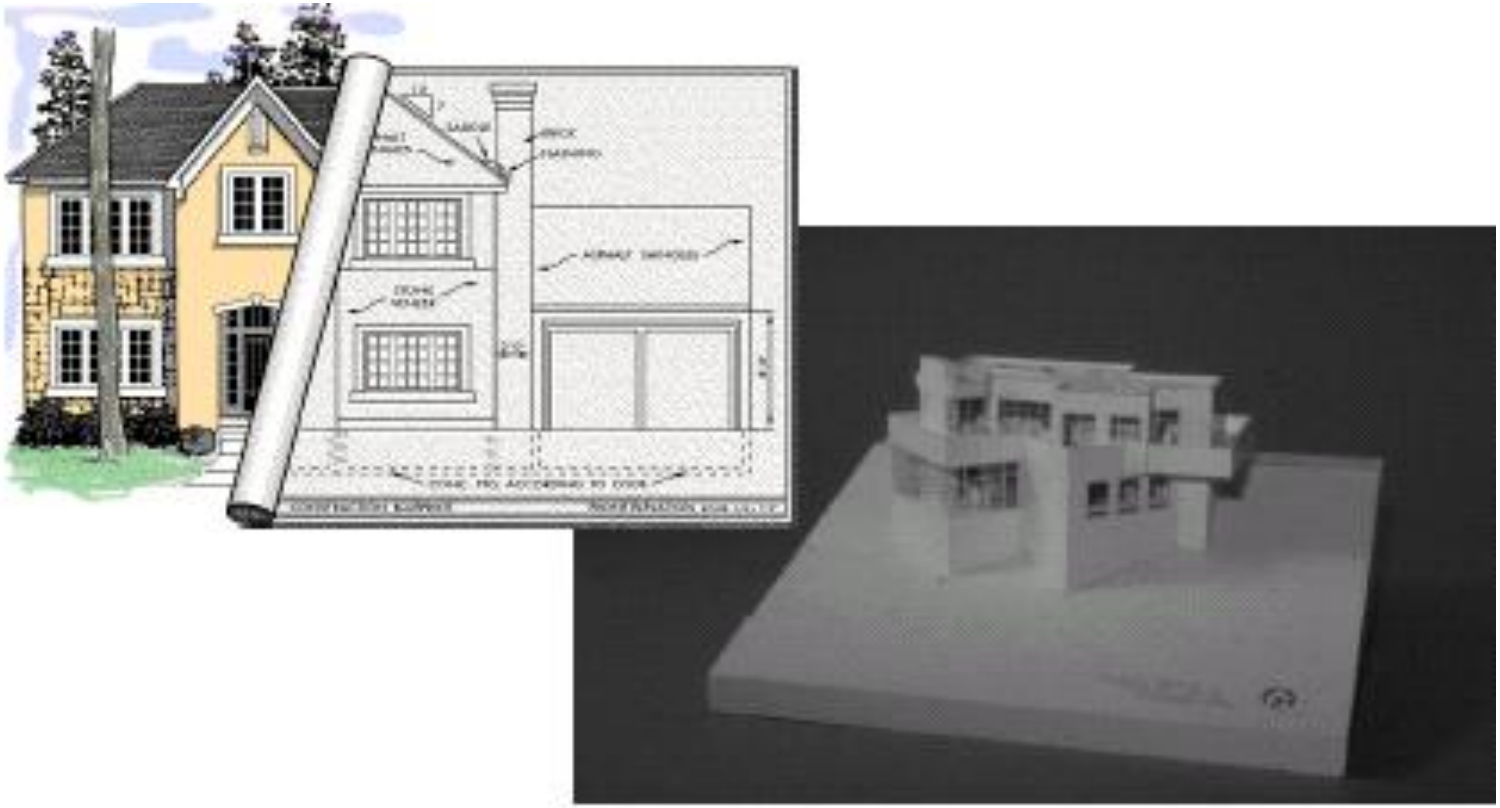
# Modelagem

- Modelagem é uma técnica de engenharia aprovada e bem aceita
  - modelos de arquitetura de casas e de grandes prédios
  - modelos matemáticos a fim de analisar os efeitos de ventos e tremores de terra --> causas



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelos



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo X Processo

- Um **modelo** é algo teórico, um conjunto de possíveis ações.
- O **processo** deve determinar ações práticas a serem realizadas pela equipe como prazos definidos e métricas para se avaliar como elas estão sendo realizadas



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo X Processo

**Modelo + Planejamento =**

**Processo**



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelos de processo de software

- Um conjunto  de atividades fundamentais exigida para desenvolver um sistema de software
  - Especificação.
  - Projeto e implementação.
  - Validação.
  - Evolução.



Fonte:
PRESSMAN, ROGER - Engenharia de Software  - 6° Edição
SOMMERVILLE - Engenharia de Software  - 8° Edição

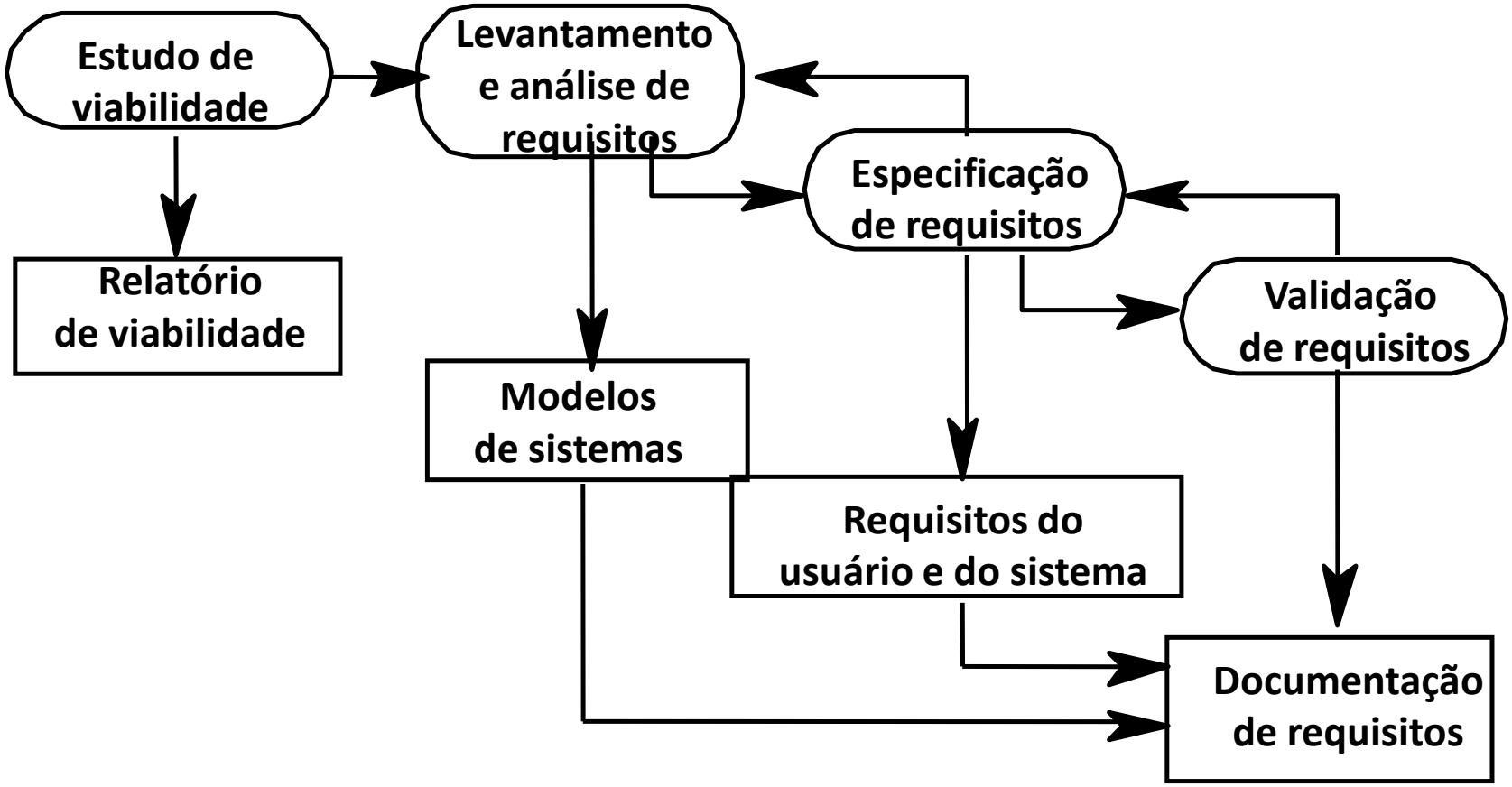
# Modelos de processo de software

- **Especificação** - definição do quê o sistema deve fazer;
- **Projeto e implementação** - definição da organização do sistema e implementação do sistema;
- **Validação** - checagem de que o sistema faz o que o cliente deseja;
- **Evolução** - evolução em resposta as mudanças nas necessidades do cliente.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Processo de software




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Processo de software

- Processo de Engenharia de Requisitos
  - Estudo de viabilidade
    - Econômica – relação custo/benefício;
    - Técnica – tecnologia e capacitação;
    - Jurídica – aspectos legais.
  - Levantamento e análise de requisitos
    - Entrevista, observação, reuniões



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Processo de software

- Especificação de requisitos
  - Documento contendo os requisitos do usuário e do sistema – funcionais e não-funcionais

- Validação de requisitos
  - Avaliação do documento de requisitos –consistência e integralidade.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelos de processo de software

- Exemplos de modelos de processo:
  - Workflow - sucessão de atividades
  - Fluxo de Dados - fluxo de informação
  - Papel/ação – representa os papéis das pessoas e as atividades pelas quais elas são responsáveis.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelos de processo de software – (paradigmas)

Uma estratégia de desenvolvimento que englobe processos, métodos e ferramentas, e as fases de desenvolvimento...



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

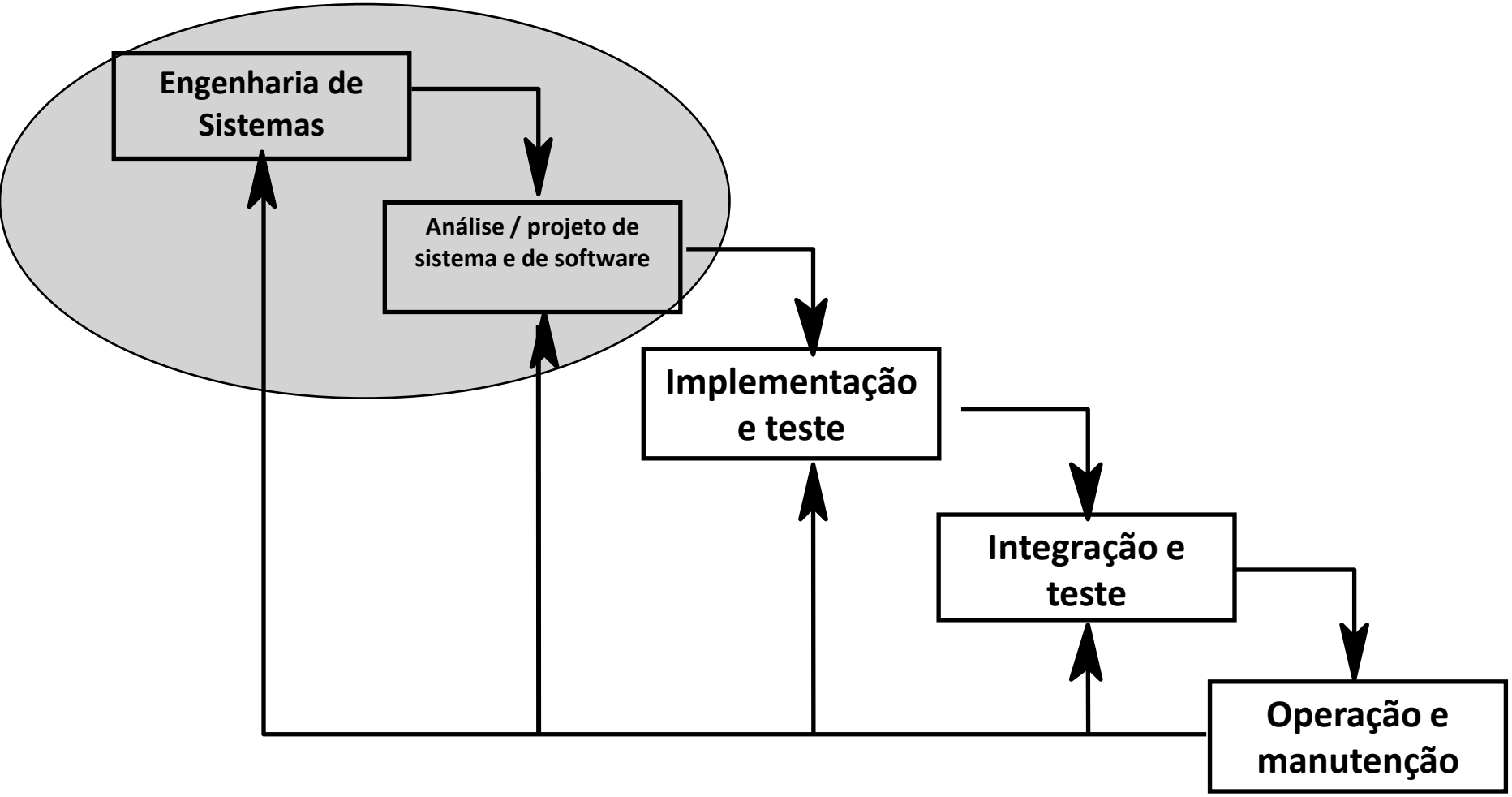
# Modelos de processo de software – (paradigmas)

- **Modelo Sequencial Linear (**ciclo de vida clássico**)**
- **Modelos Evolucionários**
  - **Prototipação**
  - **Incremental**
  - **Espiral**
  - **Métodos Ágeis**
- **Modelo de Métodos Formais**
- **Técnicas de 4a Geração**
- **Orientado a reúso**



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **ENGENHARIA DE SISTEMAS**

  - envolve a coleta de requisitos em nível do sistema, pequena quantidade de projeto e análise de alto nível

  - definir quais os requisitos do produto de software, sem especificar como esses requisitos serão obtidos.

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **ENGENHARIA DE SISTEMAS**
  - deve-se **analisar os requisitos**, recursos e restrições para:
    - **apresentar soluções;**
    - **estudar a viabilidade;**
    - **planejar** e **gerenciar** o desenvolvimento a partir de **estimativas** e **análise de riscos** que se utilizam de **métricas.**



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **Estudo de viabilidade**

  - Analisar o problema em nível global
  - Identificar soluções alternativas (custos / benefícios)
  - Simulação do futuro processo de desenvolvimento.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **Estudo de viabilidade (cont)**

**Resultado → documentação contendo**:

  - Definição do problema
  - Soluções alternativas, com os benefícios esperados
  - Fontes necessárias, custos e datas de entrega para cada solução proposta.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **ANÁLISE DE REQUISITOS DE SOFTWARE**

  - processo de coleta dos requisitos é intensificado e concentrado especificamente no software.

  - deve-se compreender o domínio da informação, a função, desempenho e interfaces exigidos.

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **ANÁLISE DE REQUISITOS DE SOFTWARE**

  - os requisitos (para o sistema e para o software) são documentados e revistos com o cliente.

**Resultado** → o contrato de desenvolvimento



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **PROJETO**
  - Definição de **"como"** o produto deve ser implementado.
  - Dividido em:
    - Projeto de alto nível – decomposição lógica
    - Projeto detalhado – decomposição física.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **PROJETO**
  - **Concentra-se em:**
    - Estrutura de Dados;
    - Arquitetura de Software;
    - Detalhes Procedimentais e
    - Caracterização de Interfaces.

**Resultado** → documentação de especificação de projeto



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **IMPLEMENTAÇÃO**

  - Tradução das representações do projeto para uma linguagem "artificial" resultando em instruções executáveis pelo computador.

**Resultado** → coleção de programas implementados e testados.

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **INTEGRAÇÃO**

  - Programas ou unidades de programas são integrados e testados como sistema. Programas ou unidades são integradas à medida em que forem sendo desenvolvidos.

**Resultado** → produto pronto para ser entregue ao cliente.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **OPERAÇÃO**
  - Instalação e configuração
  - Utilização – inicialmente operado por um grupo de usuário



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

- **MANUTENÇÃO**

  - **Corretiva:** correção de erros remanescentes
  - **Adaptativa:** adaptação dos produtos às mudanças novas versões e novas situações de operação – hardware, sistemas operacionais.
  - **Evolutiva:** alteração dos requisitos e manutenção da qualidade.

**Resultado** → produto em funcionamento.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

| ETAPA | PERGUNTAS-CHAVES | CRITÉRIOS DE SAÍDA |
|---|---|---|
| | | |
| Definição do problema | Qual é o problema | Declaração da delimitação e objetivos. |
| Estudo de viabilidade | Há uma solução viável | Análise geral de custo/benefício<br>Alcance e objetivos do sistema. |
| Análise | O que terá de ser feito para resolver o problema? | Modelo lógico do sistema:<br>Diagrama de Fluxo de Dados;<br>Diagrama de Entidade e Relacionamento<br>Diagrama de Transição de Estado;<br>Dicionário de Dados;<br>Especificação de Processos.<br>UML. |



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Sequencial Linear (ciclo de vida clássico)

| ETAPA | PERGUNTAS-CHAVES | CRITÉRIOS DE SAÍDA |
|---|---|---|
| Projeto | Como o problema deve ser resolvido?<br>Como o sistema deve ser implementado? | Soluções Alternativas<br>Especificação de hard/soft;<br>Plano de implementação;<br>Plano de teste preliminares;<br>Procedimento de segurança;<br>Procedimento de auditoria. |
| Implementação | Faça | Programas;<br>Plano de testes;<br>Procedimento de segurança;<br>Procedimento de auditoria. |
| Teste | Verificar o sistema | Testes do geral do sistema. |
| Manutenção | Modificar o sistema conforme necessidade. | Apoio continuado. |



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# CONTRIBUIÇÕES DO CICLO DE VIDA CLÁSSICO

- Processo de desenvolvimento de software deve ser sujeito à disciplina, planejamento e gerenciamento.
- A implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos.
- Deve ser utilizado quando os requisitos estão bem claros no inicio do desenvolvimento.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROBLEMAS DO CICLO DE VIDA CLÁSSICO

- Rigidez
- Qualquer desvio é desencorajado
- Todo o planejamento é orientado para a entrega do produto de software em uma data única.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Evolucionário

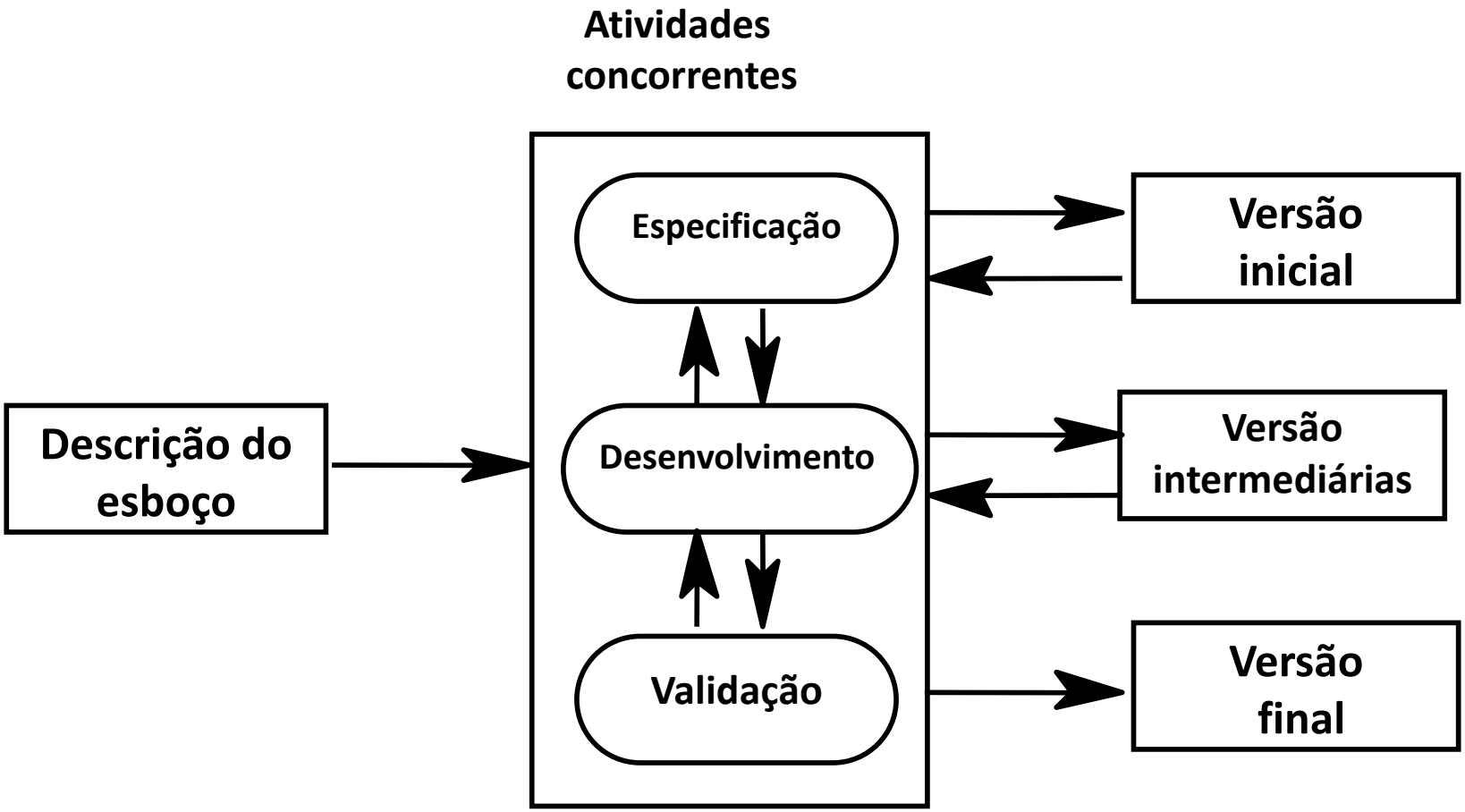
- Abordagem baseada na idéia de desenvolver uma implementação inicial, expor o resultado ao comentário do usuário e fazer seu aprimoramento por meio de muitas versões.

- Especificação e desenvolvimento são intercalados



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo evolucionário
## (Desenvolvimento exploratório )




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo evolucionário
## (Desenvolvimento exploratório )

- Trabalhar junto com o cliente, a fim de explorar seus requisitos e entregar um sistema final.

- O desenvolvimento se inicia com as partes do sistema que são mais bem compreendidas.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo evolucionário
## (Desenvolvimento exploratório )

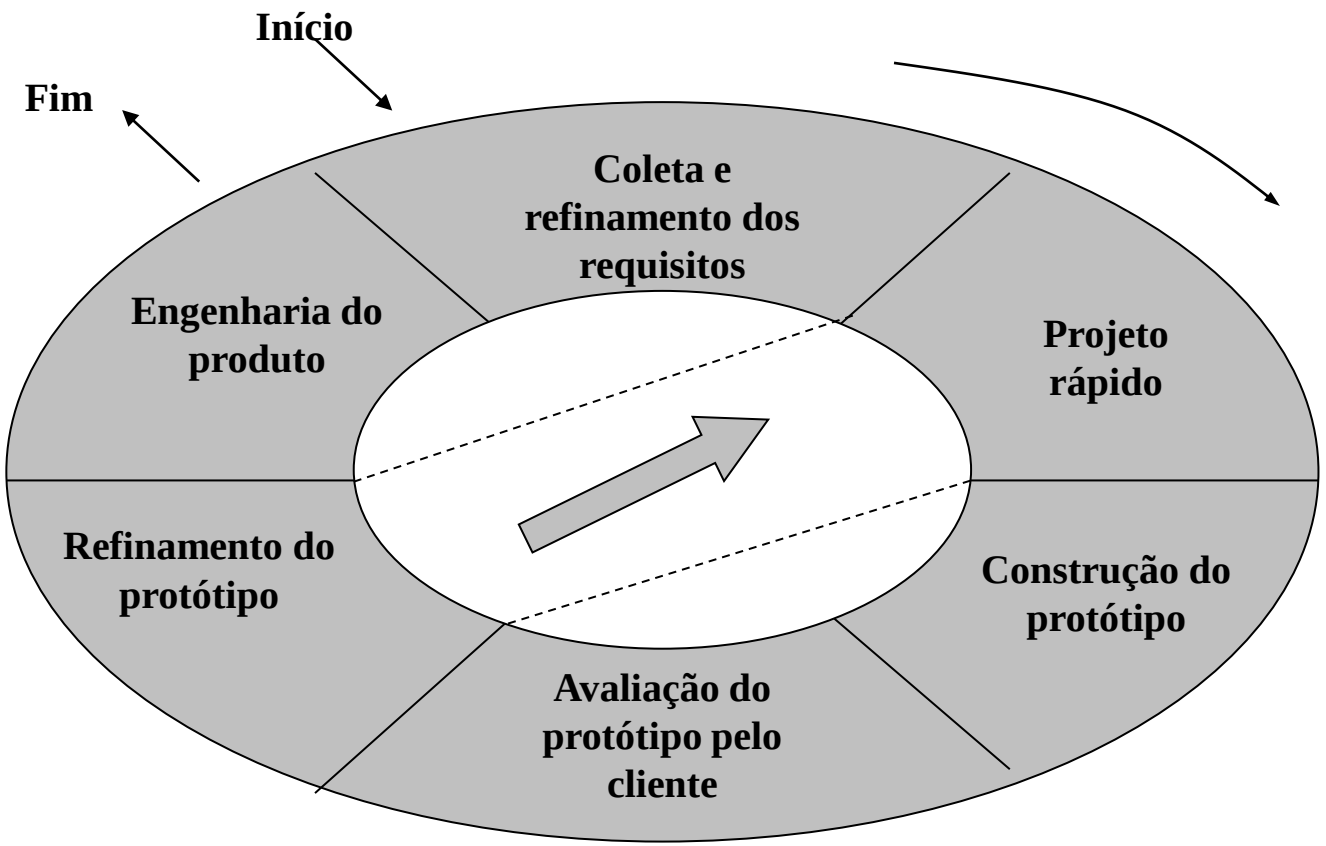
- O sistema evolui com o acréscimo de novas características à medida que elas são propostas pelo cliente.

- Importante quando é difícil, ou mesmo impossível, estabelecer uma especificação detalhada dos requisitos do sistema a priori.



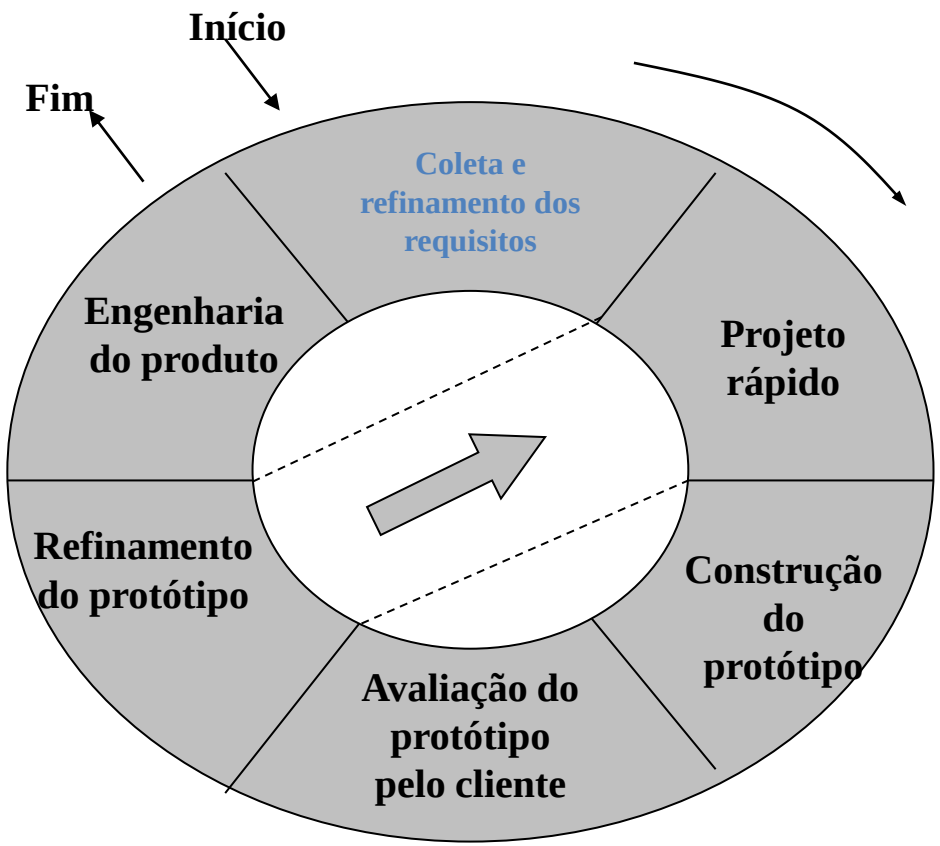
Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO



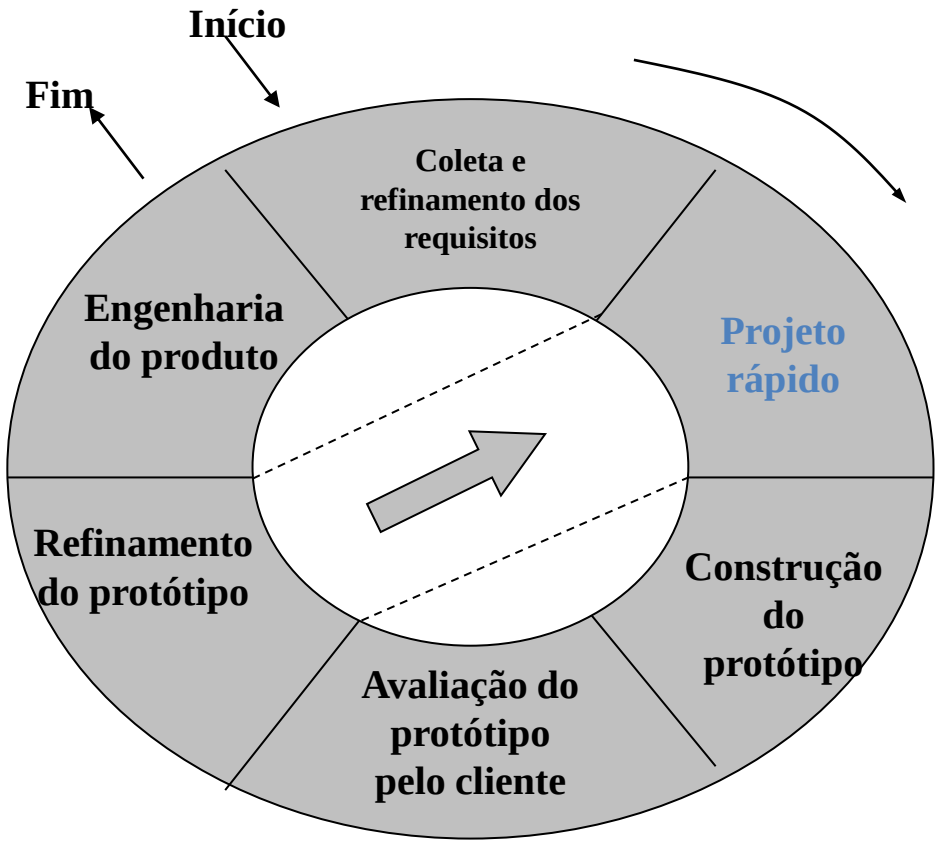

**COLETA DOS REQUISITOS:**

desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO



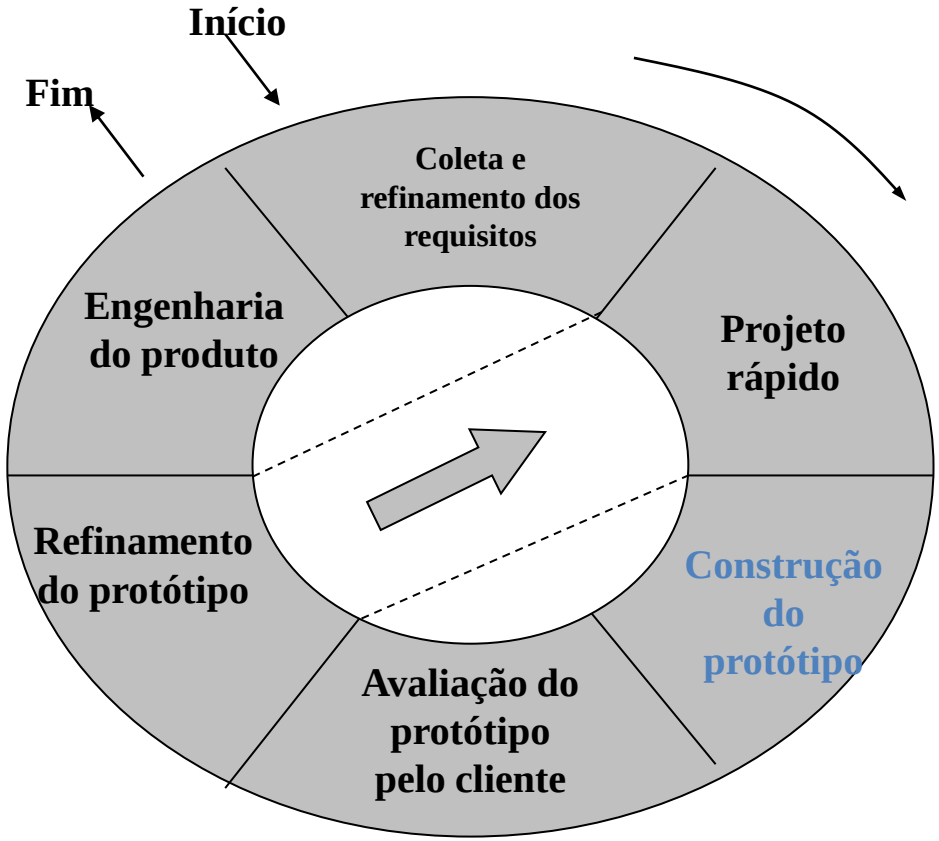

**PROJETO RÁPIDO:**

representação dos aspectos do software que são visíveis ao usuário (abordagens de entrada e formatos de saída)



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO

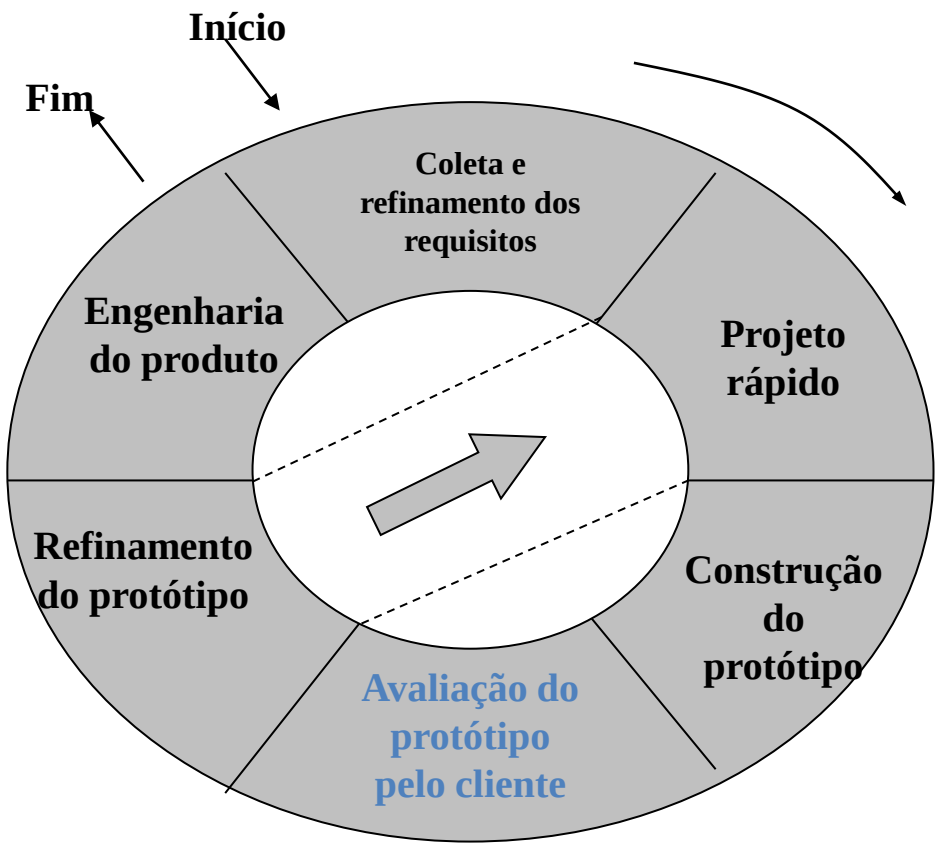



**CONSTRUÇÃO PROTÓTIPO:**

Implementação do projeto rápido serve como o "primeiro sistema" - recomendado que se jogue fora futuramente

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO

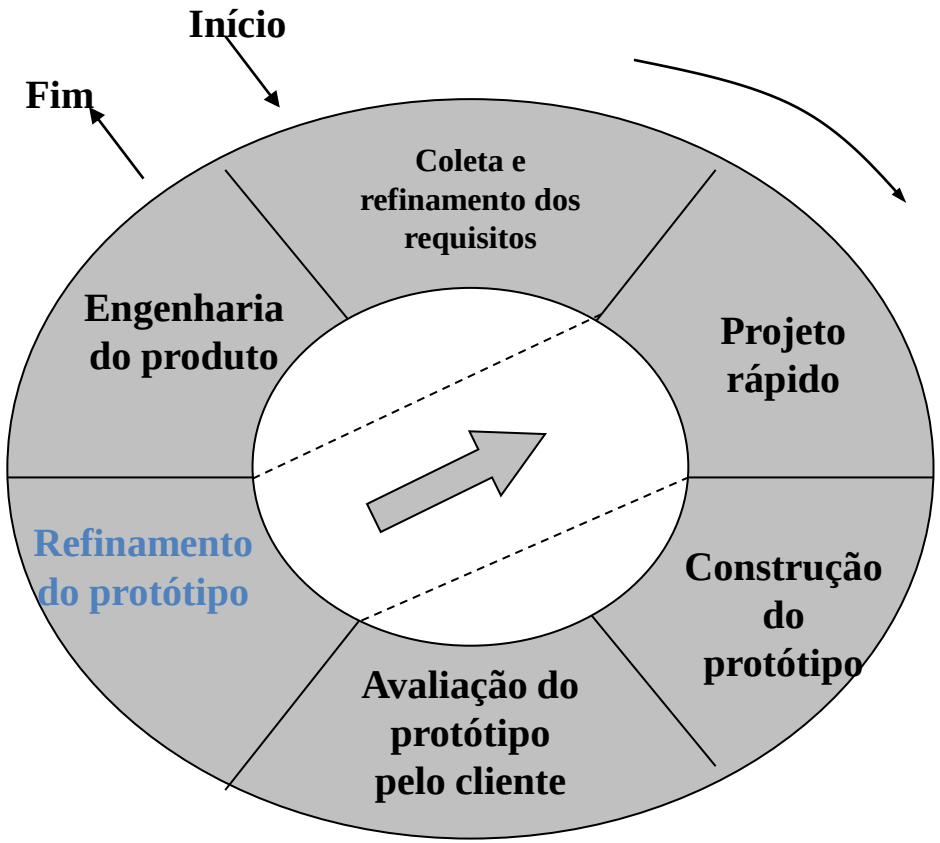



## AVALIAÇÃO DO PROTÓTIPO:

Cliente e desenvolvedor avaliam o protótipo

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO



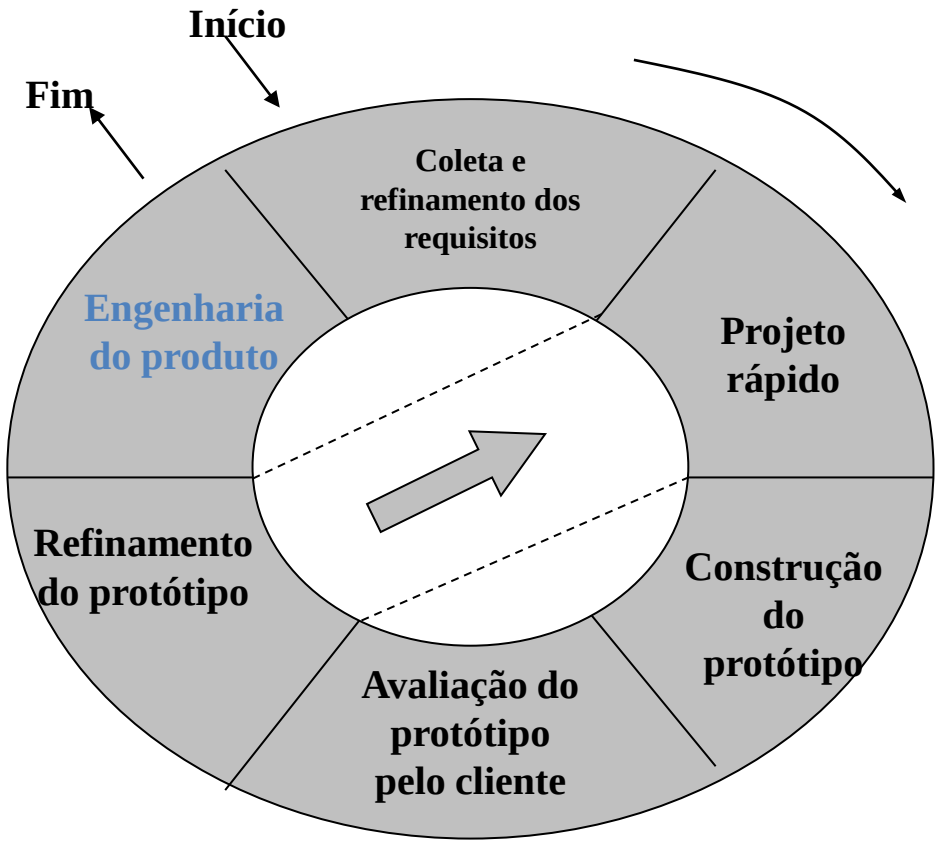

## ***REFINAMENTO DOS REQUISITOS:***

Cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROTOTIPAÇÃO




**CONSTRUÇÃO PRODUTO:**

identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# CONTRIBUIÇÕES DA PROTOTIPAÇÃO

- Sistemas pequenos
- Útil quando os requisitos estão obscuros
- Especificação é construída gradativamente
- Possibilitam um rápido desenvolvimento da aplicação



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

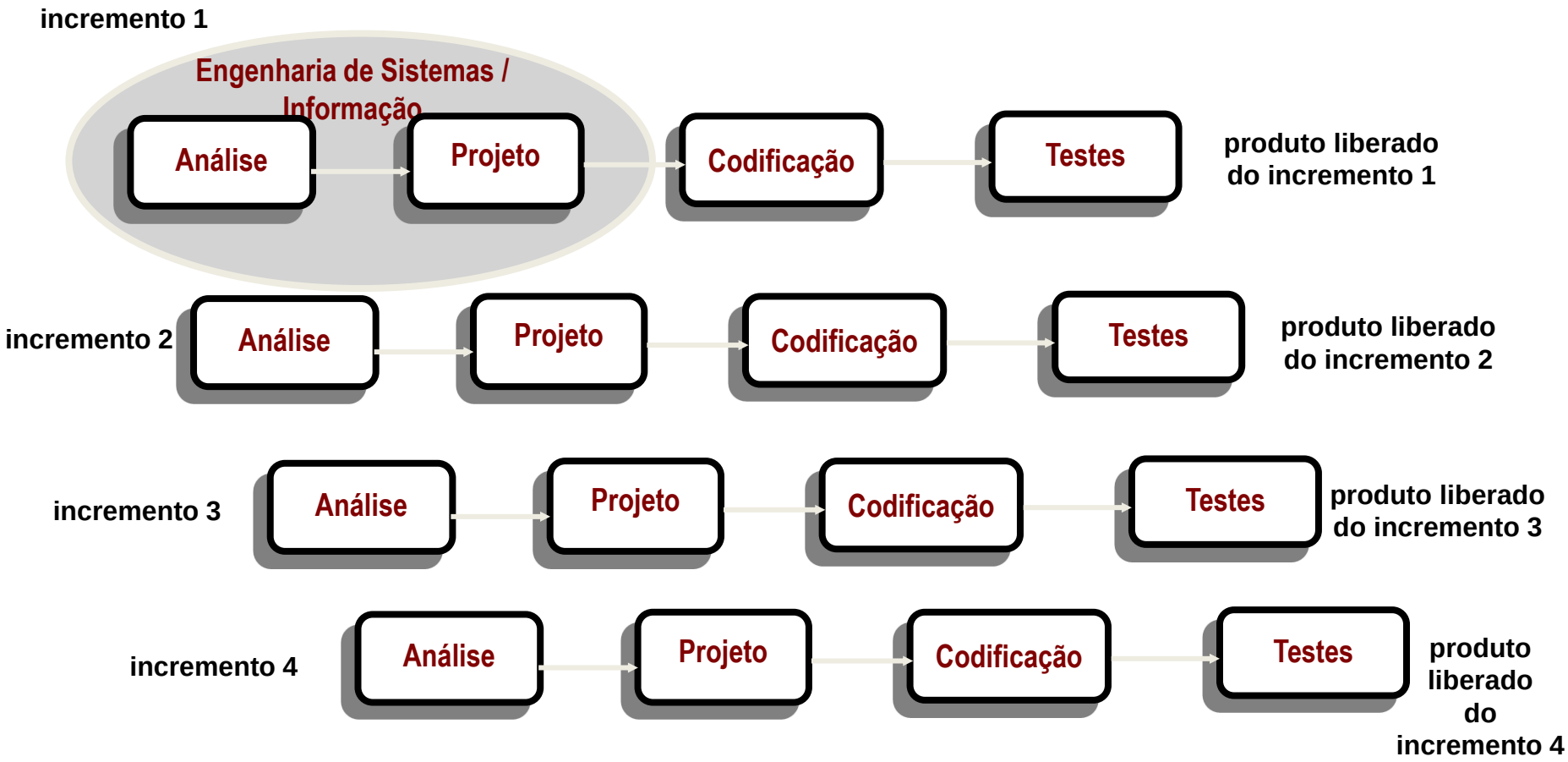
# PROBLEMAS DA PROTOTIPAÇÃO

- O processo não é visível
- Os sistemas são frequentemente mal-estruturados e mal-documentados
- Processo não é claro, dificuldade de planejamento e gerenciamento



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# MODELO INCREMENTAL






Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# MODELO INCREMENTAL

- Combina elementos do Modelo Linear com a filosofia da Prototipação.
- Aplica sequências lineares numa abordagem de “saltos” à medida que o tempo progride.
- Cada sequência linear produz um incremento do software (proc. de texto)



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# MODELO INCREMENTAL

- O processo se repete até que um produto completo seja produzido.

- Difere da Prototipação, pois a cada incremento produz uma versão operacional do software.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# CONTRIBUIÇÕES DO MODELO INCREMENTAL

- **Entrega acelerada dos serviços de cliente.**
  Cada incremento fornece a funcionalidade de mais alta prioridade para o cliente.

- **Engajamento do usuário com o sistema.**
  Os usuários têm de estar envolvidos no processo de desenvolvimento, o que significa que o sistema muito provavelmente atenderá aos seus requisitos, e que os usuários estarão mais comprometidos com ele.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROBLEMAS DO MODELO INCREMENTAL

- **Problemas de gerenciamento**

  O progresso pode ser difícil de julgar e os problemas, difíceis de serem encontrados, porque não há documentação que mostre o que foi feito.

- **Problemas contratuais**

  O contrato normal pode incluir uma especificação; sem uma especificação, formulários diferentes de contrato têm de ser usados.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROBLEMAS DO MODELO INCREMENTAL

- **Problemas de validação**

  Sem uma especificação, contra o que o sistema está sendo testado.

- **Problemas de manutenção**

  Mudanças contínuas tendem a corromper a estrutura do software, o que torna mais dispendioso mudar e evoluir para atender aos novos requisitos.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

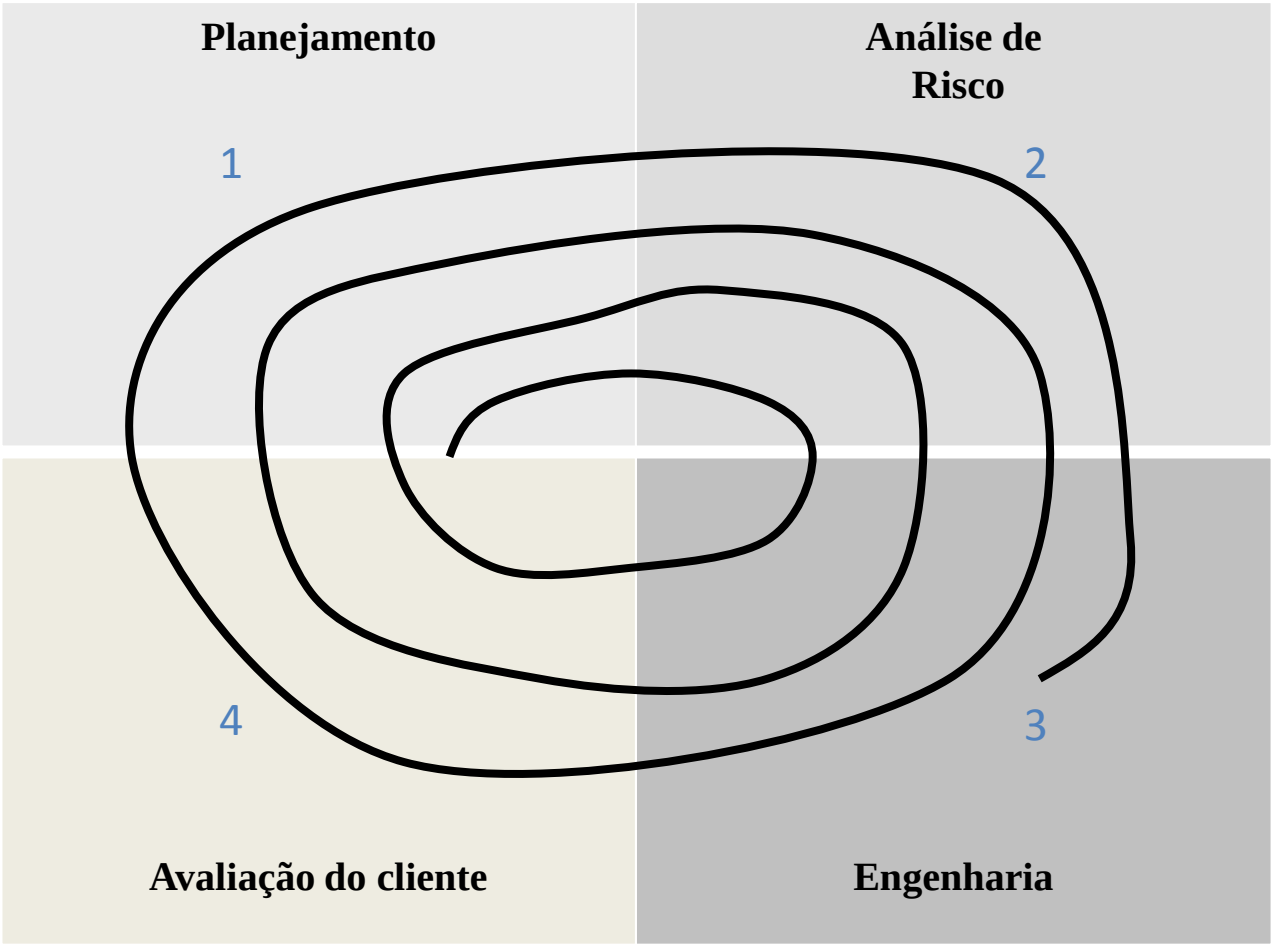
# MODELO ESPIRAL (Boehm)

- Desenvolvido para englobar as melhores características do ciclo de vida clássico e do paradigma evolutivo.
- São avaliados riscos explicitamente e são solucionados ao longo do processo.
- Processo é representado como uma espiral em lugar de ser representado como uma sequência de atividades



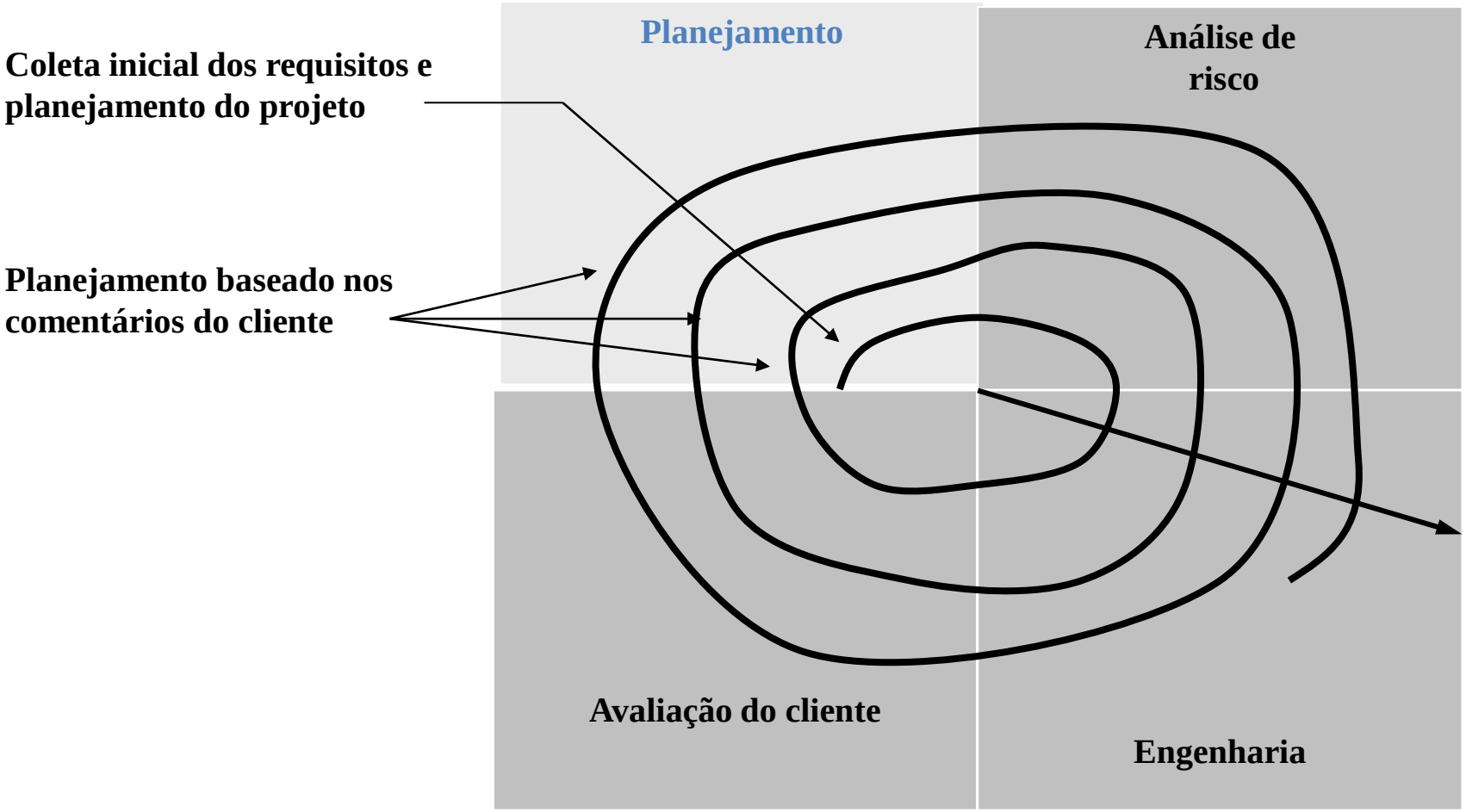
Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# MODELO ESPIRAL (Boehm)




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Espiral




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Espiral

• São identificados objetivos específicos, tais como desempenho e funcionalidade.

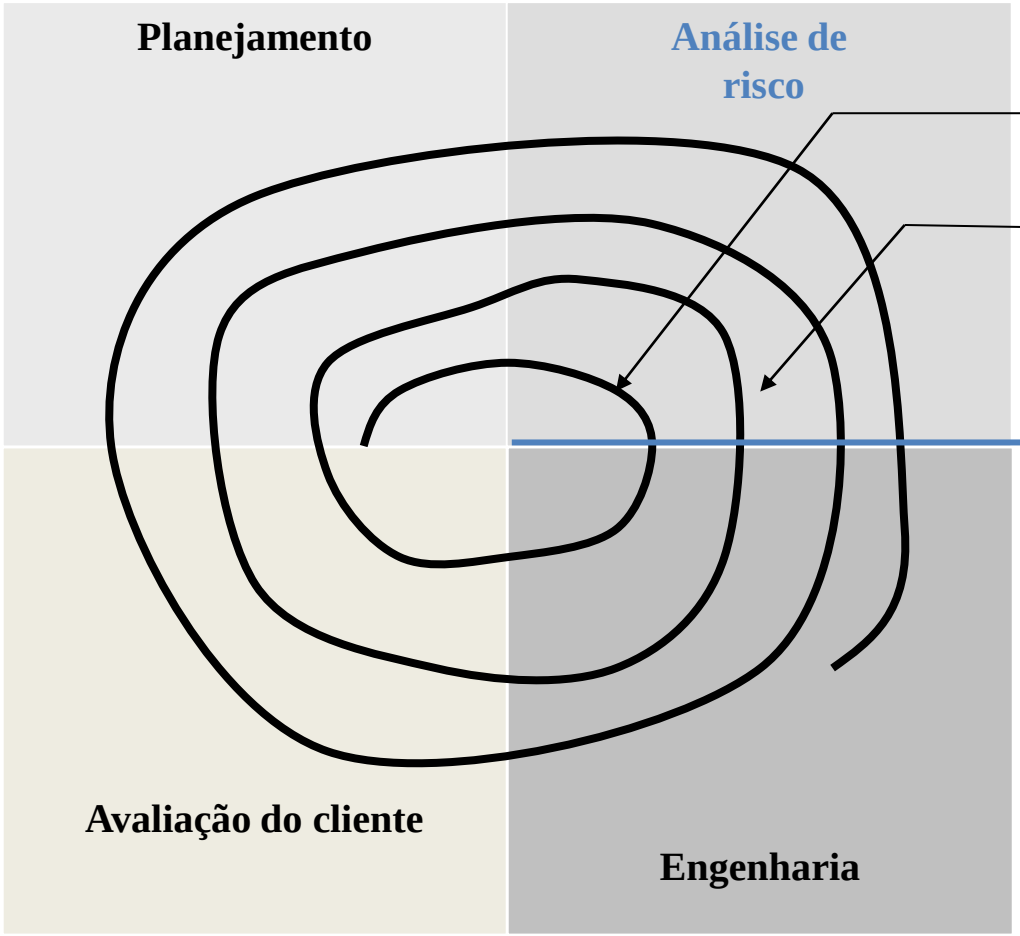
• São determinadas alternativas para atingir estes objetivos.

• São identificadas restrições do processo e do produto e é elaborado um relatório de gestão detalhado.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Espiral

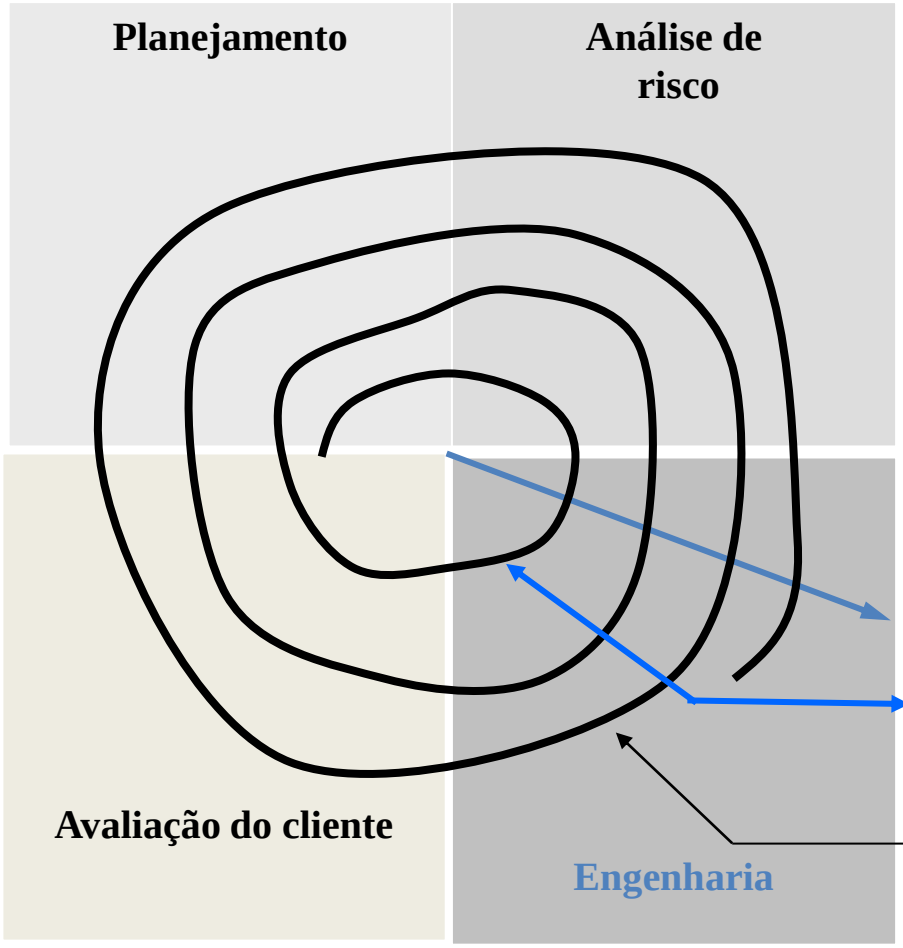



Para cada risco do projeto identificado em planejamento é levada a cabo uma análise detalhada.

Decisão de prosseguir/não prosseguir

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Espiral




Na direção de um sistema concluído

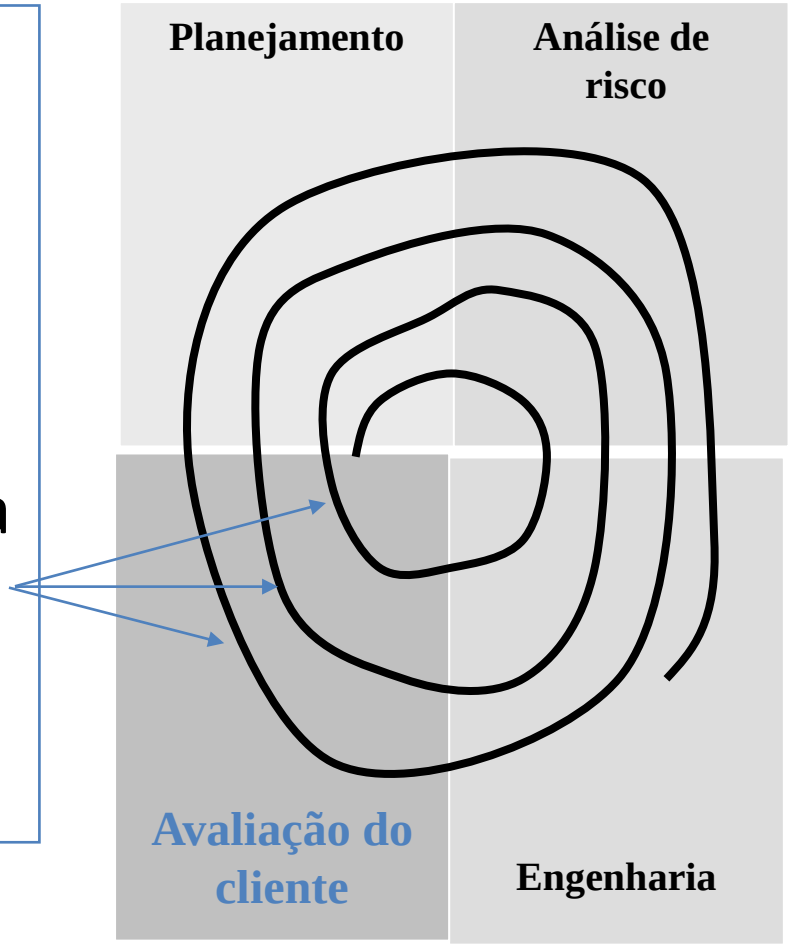
**Protótipo de software inicial**

**Sistema construído pela engenharia**

Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Modelo Espiral

- tarefas requeridas para obter um feedback do cliente baseado na avaliação da representação do software criado durante a fase de engenharia e implementado durante a fase de instalação




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# DESENVOLVIMENTO RÁPIDO DE SOFTWARE

- Métodos ágeis
- Extreme programming
- Desenvolvimento rápido de aplicações
- Prototipação de software



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# DESENVOLVIMENTO RÁPIDO DE SOFTWARE

• Devido à rápida mudança dos ambientes de negócio, os negócios devem responder às novas oportunidades e à competição.

• Isso requer software e desenvolvimento rápido, e a entrega é, frequentemente, o requisito mais crítico para sistemas de software.

• Os negócios podem estar dispostos a aceitar um software de baixa qualidade se a entrega rápida e a funcionalidade essencial for possível.

# MÉTODOS ÁGEIS

- Movimento iniciado por programadores experientes e consultores em desenvolvimento de software.
- **Objetivo:** satisfazer o cliente entregando, rapidamente e com frequência, sistemas com algum valor.
- Os **métodos ágeis** são, provavelmente, os mais adequados para sistemas de negócio de porte pequeno/médio ou produtos para PC.

# MÉTODOS ÁGEIS

- A insatisfação com os overheads envolvidos nos métodos de projeto levou à criação dos métodos ágeis. Esses métodos:
  - Enfocam o código ao invés do projeto;
  - São baseados na abordagem iterativa para desenvolvimento de software;
  - São destinados a entregar software de trabalho e evoluí-lo rapidamente para atender aos requisitos que se alteram.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PRINCÍPIOS DOS MÉTODOS ÁGEIS

## Envolvimento do cliente →

Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema.Clientes devem ser profundamente envolvidos no processo de desenvolvimento.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PRINCÍPIOS DOS MÉTODOS ÁGEIS

## Entrega incremental →

O software é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada incremento.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PRINCÍPIOS DOS MÉTODOS ÁGEIS

## Pessoas, não processo →

As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar sem processos prescritivos.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PRINCÍPIOS DOS MÉTODOS ÁGEIS

**Aceite as mudanças →**

Projete o sistema para acomodar mudanças.

**Mantenha a simplicidade →**

Elimine a complexidade do sistema.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROBLEMAS COM MÉTODOS ÁGEIS

- Difícil manter o interesse dos clientes que estão envolvidos no processo.

- Os membros da equipe podem ser inadequados para o intenso envolvimento que caracteriza os métodos ágeis.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# PROBLEMAS COM MÉTODOS ÁGEIS

- A priorização de mudanças pode ser difícil onde existem múltiplos stakeholders.

- A manutenção da simplicidade requer trabalho extra.

- Problemas nos contratos.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# EXTREME PROGRAMMING

- É talvez o mais conhecido e mais amplamente usado dos métodos ágeis.

- A eXtreme Programming (XP) leva uma abordagem “extrema” para desenvolvimento iterativo.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# EXTREME PROGRAMMING

- Novas versões podem ser compiladas várias vezes por dia. Os incrementos são entregues para os clientes a cada 2 semanas.

- Todos os testes devem ser realizados para cada nova versão.

# OS 4 VALORES DE XP

- Comunicação
- Simplicidade
- Retorno (feedback)
- Coragem



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# EXTREME PROGRAMMING
# a quem se destina

- Grupos de 2 a 10 programadores
- Projetos de 1 a 36 meses (calendário)
- De 1000 a 250 000 linhas de código

- Papéis:
  - Programadores →foco central (sem hierarquia)
  - "Treinador" ou "Técnico" →(*coach*)
  - "Acompanhador" (*tracker*)
  - Cliente



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# EXTREME PROGRAMMING
# "Treinador" ou "Técnico" (*coach*)

- Em geral, o mais experiente do grupo. Identifica quem é bom no que. Comunica-se com outros gerentes e diretoria.

- Concentra-se na execução e evolução técnica do projeto.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# EXTREME PROGRAMMING
# "Treinador" ou "Técnico" (*coach*)

- Eventualmente faz programação pareada.
- Não desenha arquitetura, apenas chama a atenção para oportunidades de melhorias.
- Seu papel diminui à medida em que o time fica mais maduro.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# EXTREME PROGRAMMING
# *Tracker* (Acompanhador)

- Coleta estatísticas sobre o andamento do projeto.
  Alguns exemplos:
  - Número de histórias definidas e implementadas.
  - Número de *unit tests.*
  - Número de testes funcionais definidos e funcionando.
  - Número de classes, métodos, linhas de código

- Mantém histórico do progresso. Faz estimativas para o futuro.

# Um Dia na Vida de um Programador XP

- Escolhe uma história do cliente.
- Procura um par livre.
- Escolhe um computador para programação pareada (*pair programming*).
- Seleciona uma tarefa claramente relacionada a uma característica (*feature*) desejada pelo cliente.

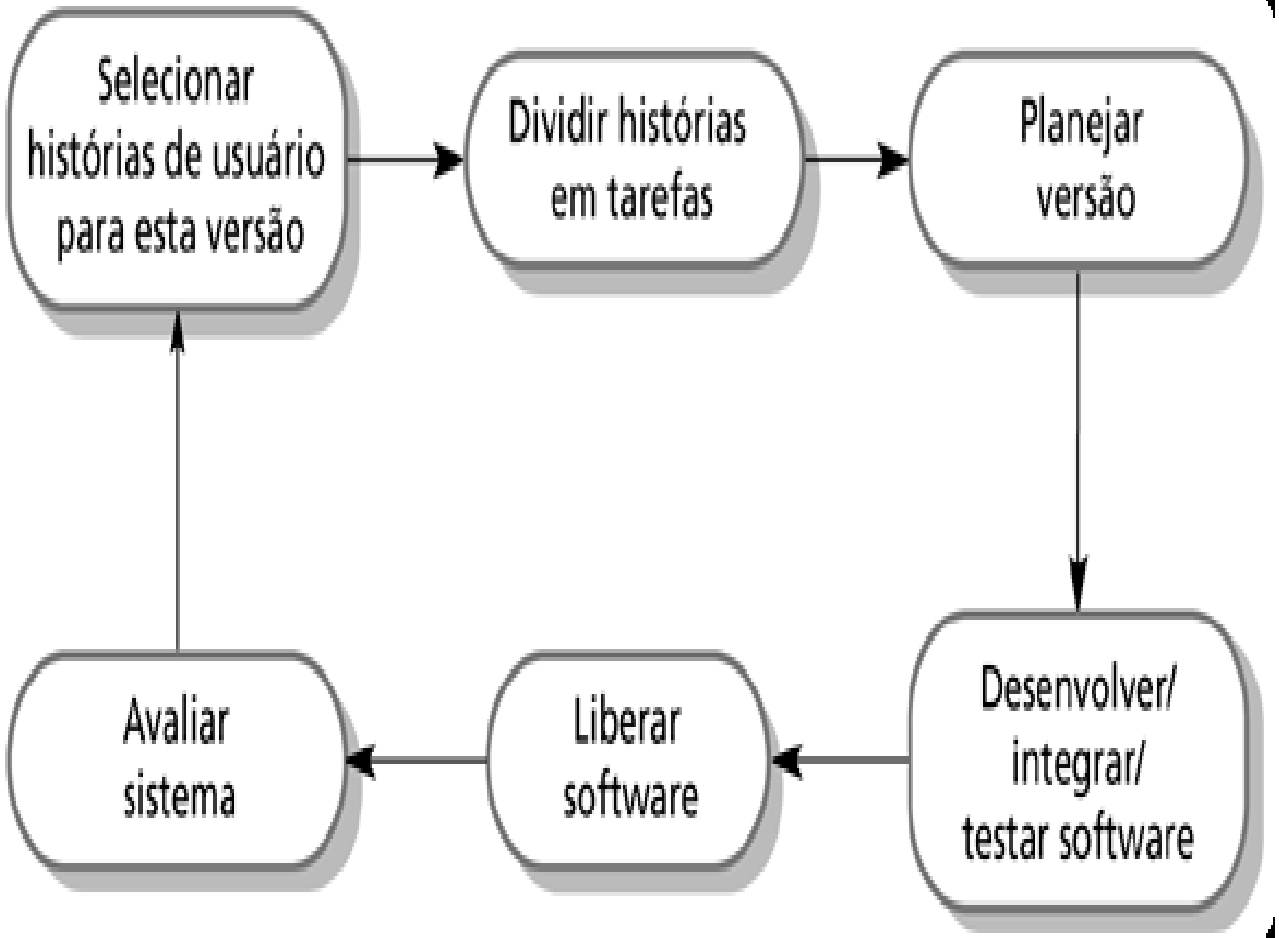


Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# O ciclo de release em XP

**Figura 17.3**

Ciclo de um release em *extreme programming*




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Práticas do Extreme Programming

**Planejamento Incremental →**
Registrados em cartões de histórias

**Pequenos releases →**Conjunto mínimo útil de funcionalidade é desenvolvido.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Práticas do Extreme Programming

**Projeto simples →**

Projeto suficiente para atender aos requisitos atuais.

**Desenvolvimento test-first →**

Uso um framework automatizado.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Práticas do Extreme Programming

**Refactoring →**

Espera-se que todos os desenvolvedores recriem o código continuamente.

**Programação em pares →**

Os desenvolvedores trabalham em pares.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Práticas do Extreme Programming

**Propriedade coletiva →**

Os pares trabalham em todas as áreas do sistema.

**Integração contínua →**

Tarefa concluída é automaticamente integrada ao sistema.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Práticas do Extreme Programming

**Ritmo sustentável →**

Não aceitar grande quantidade de horas extras.

**Cliente on-site →**

Um usuário do sistema deve estar disponível em tempo integral..



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Quando XP Não Deve Ser Experimentada?

- Quando o cliente não aceita as regras do jogo.
- Quando o cliente quer uma especificação detalhada do sistema antes de começar.
- Quando os programadores não estão dispostos a seguir (todas) as regras.
- Se (quase) todos os programadores do time não são experientes.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Quando XP Não Deve Ser Experimentada?

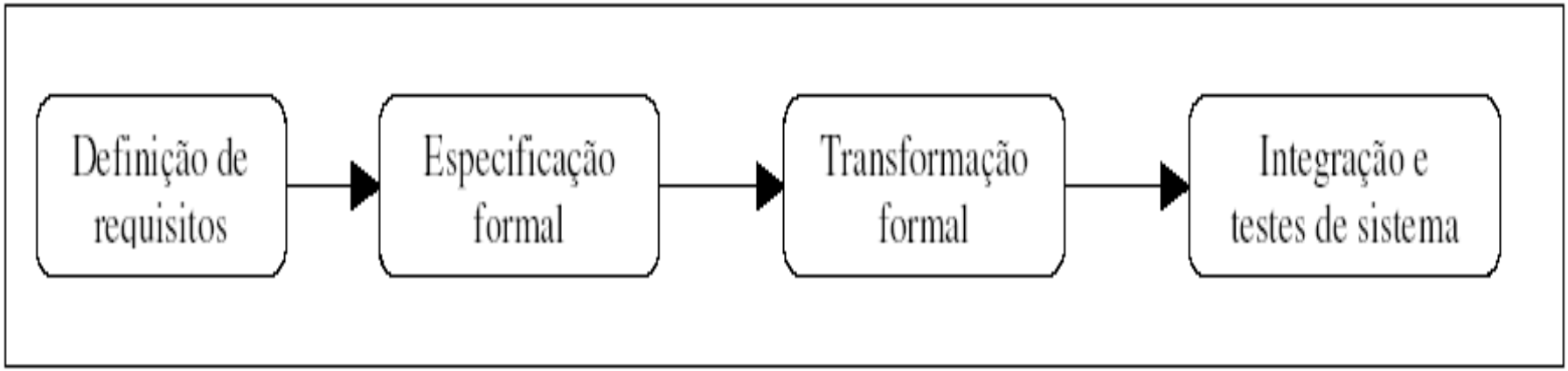
- Grupos grandes (>10 programadores).
- Quando *feedback* rápido não é possível:
  - sistema demora 6h para compilar.
  - testes demoram 12h para rodar.
  - exigência de certificação que demora meses.
- Quando o custo de mudanças é essencialmente exponencial.
- Quando não é possível realizar testes (muito raro).



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# MODELO DE MÉTODOS FORMAIS

Um modelo de sistema matemático é transformado formalmente em uma implementação




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

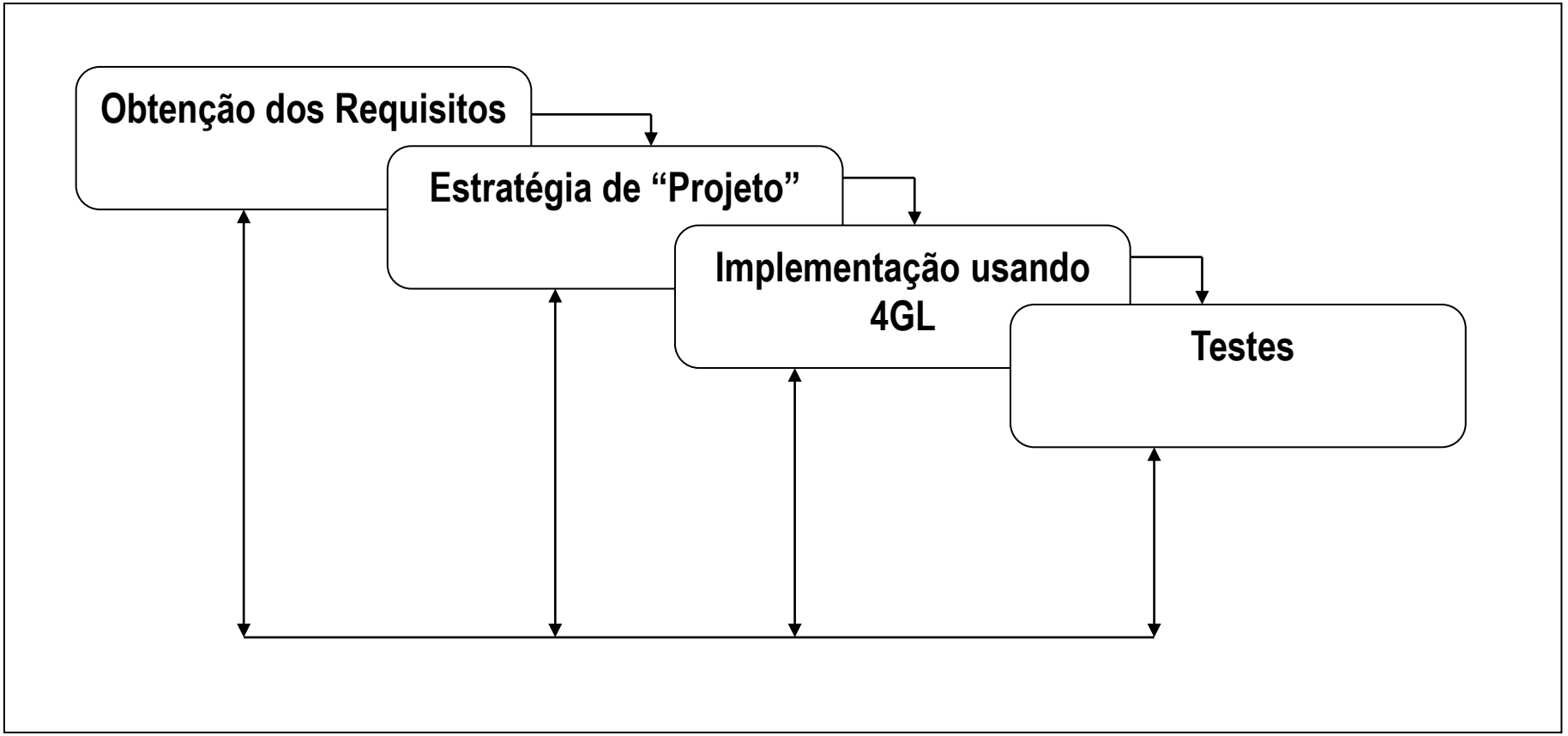
# MODELO DE MÉTODOS FORMAIS

- Compreende um conjunto de atividades que determinam uma especificação matemática para o software.
- A especificação de requisitos de software é redefinida em uma especificação formal detalhada, que é expressa em uma notação matemática.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# TÉCNICAS DE 4ª GERAÇÃO




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# TÉCNICAS DE 4ª GERAÇÃO

- Concentra-se na capacidade de se especificar o software a uma máquina em um nível que esteja próximo à linguagem natural.

- Engloba um conjunto de ferramentas de software que possibilitam que:

  ⇨ o sistema seja especificado em uma linguagem de alto nível e

  ⇨o código fonte seja gerado automaticamente a partir dessas especificações



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

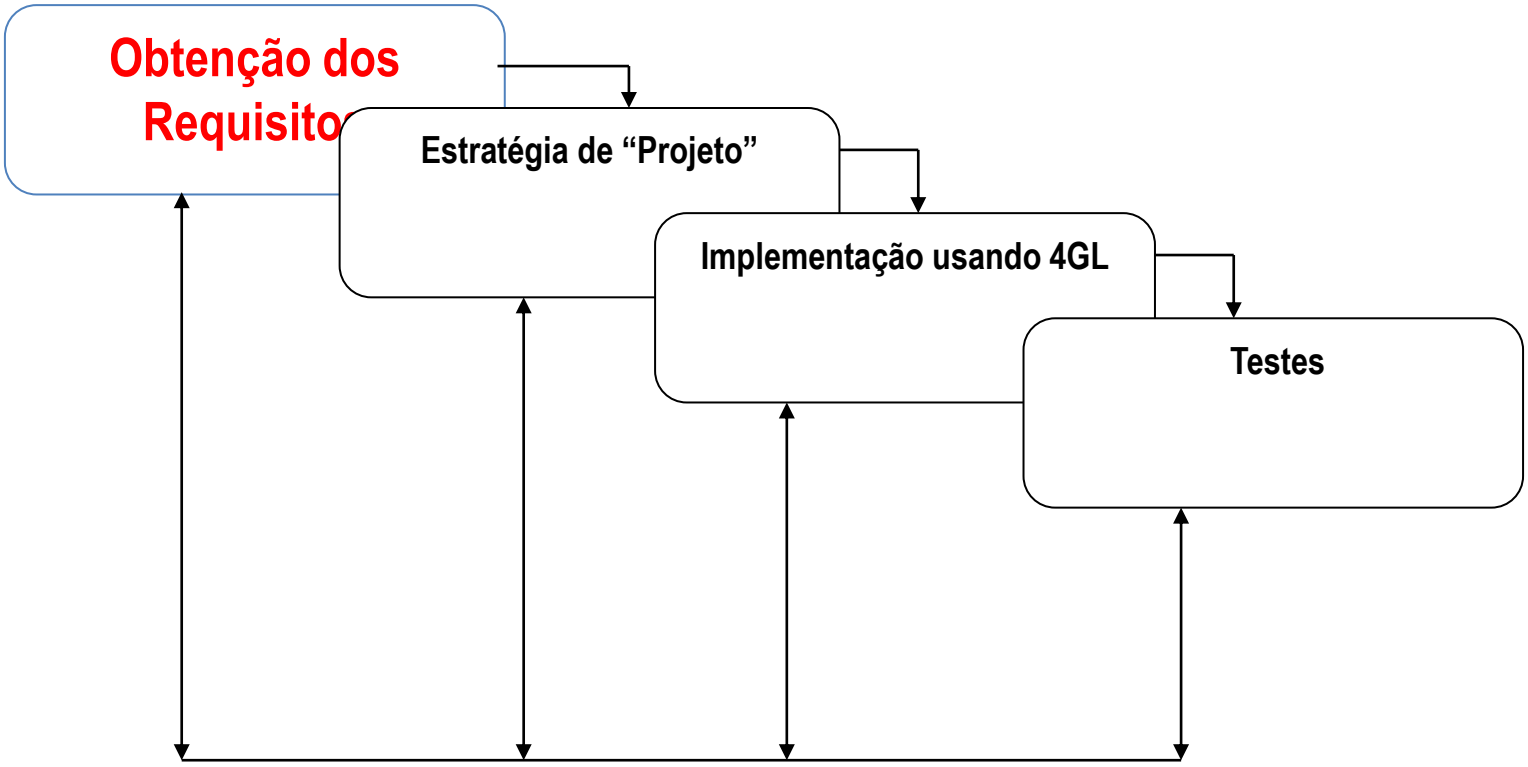
# TÉCNICAS DE 4ª GERAÇÃO

- O ambiente de desenvolvimento inclui as ferramentas:
  - linguagens não procedimentais para consulta de banco de dados
  - geração de relatórios
  - manipulação de dados
  - interação e definição de telas
  - geração de códigos
  - capacidade gráfica de alto nível
  - capacidade de planilhas eletrônicas



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

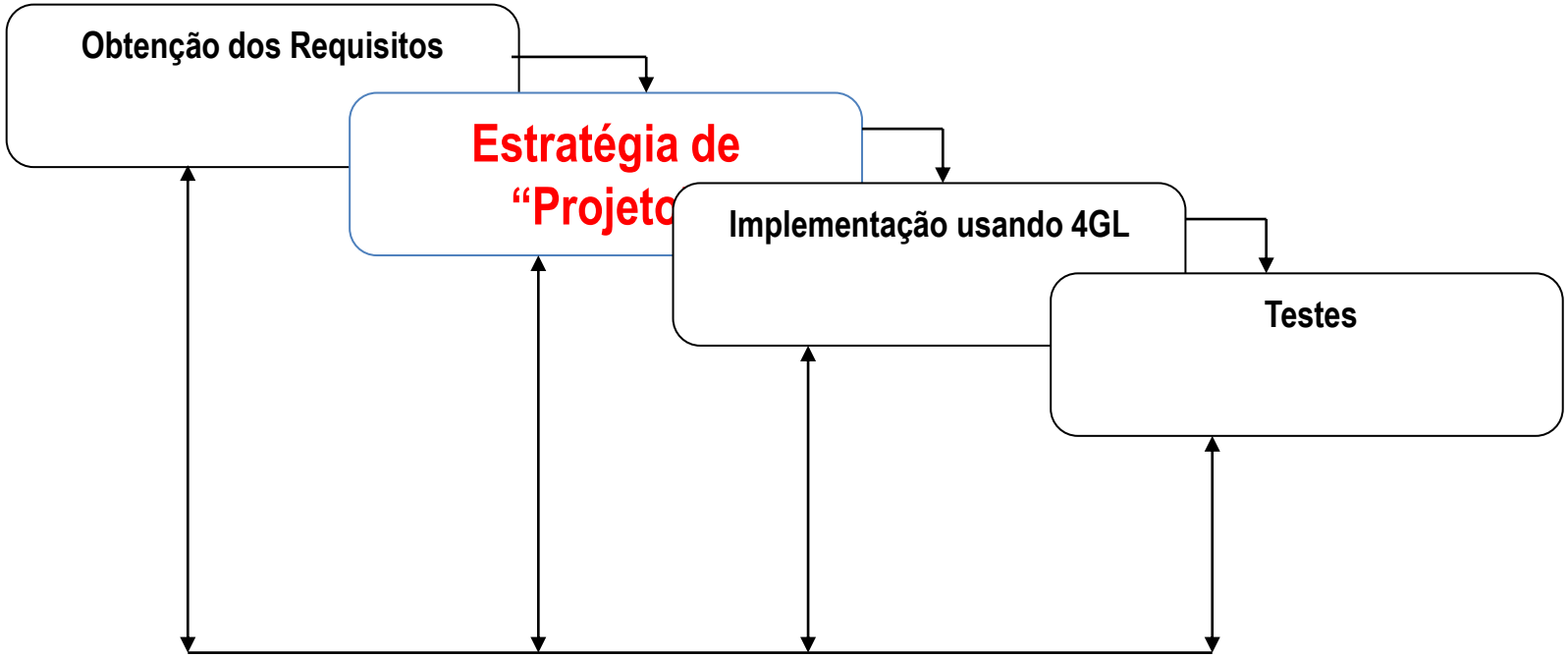
# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO

- **OBTENÇÃO DOS REQUISITOS:** o cliente descreve os requisitos os quais são traduzidos para um protótipo operacional
  - O cliente pode estar inseguro quanto aos requisitos
  - O cliente pode ser incapaz de especificar as informações de um modo que uma ferramenta 4GL possa consumir



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

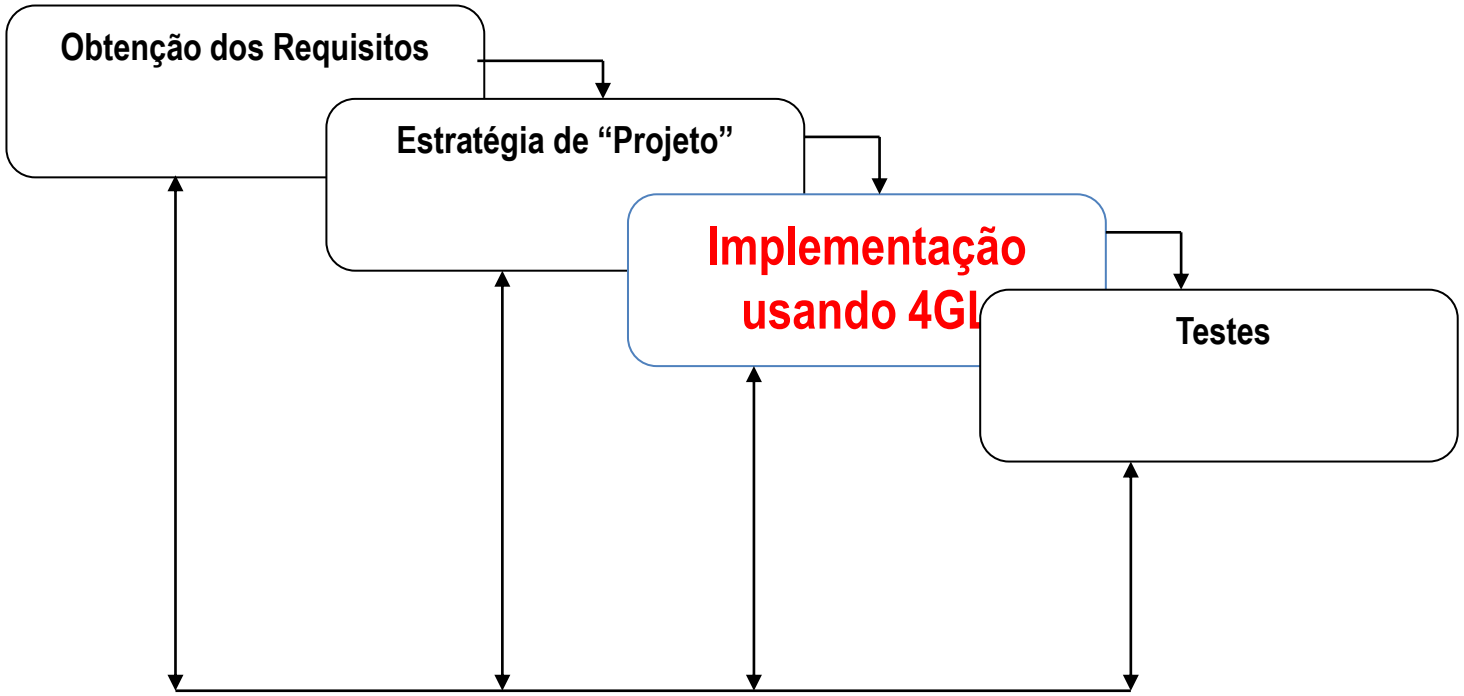
# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO

- **ESTRATÉGIA DE "PROJETO":**
  - para pequenas aplicações é possível mover-se do passo de  Obtenção dos Requisitos para o passo de Implementação
  - Para grandes projetos é necessário desenvolver uma estratégia de projeto. De outro modo ocorrerão os mesmos problemas encontrados quando se usa abordagem convencional (baixa qualidade, manutenibilidade ruim, má aceitação do cliente)



Fonte:
PRESSMAN, ROGER - Engenharia de Software  - 6° Edição
SOMMERVILLE - Engenharia de Software  - 8° Edição

# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

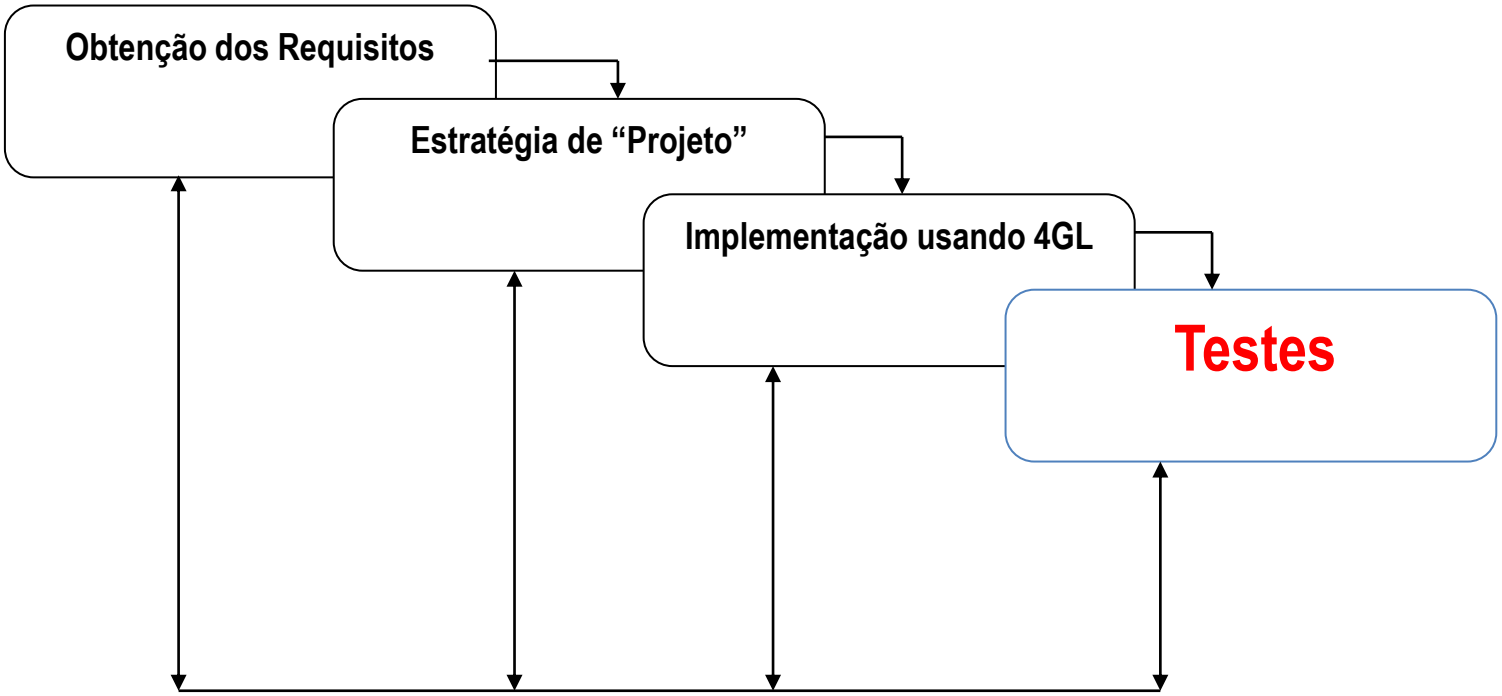
# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO

- **IMPLEMENTAÇÃO USANDO 4GL:**
  - os resultados desejados são representados de modo que haja geração automática de código.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# ATIVIDADES DAS TÉCNICAS DE 4ª GERAÇÃO

- **TESTE:**
  - o desenvolvedor deve efetuar testes e desenvolver uma documentação significativa.
  - O software desenvolvido deve ser construído de maneira que a manutenção possa ser efetuada prontamente.



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

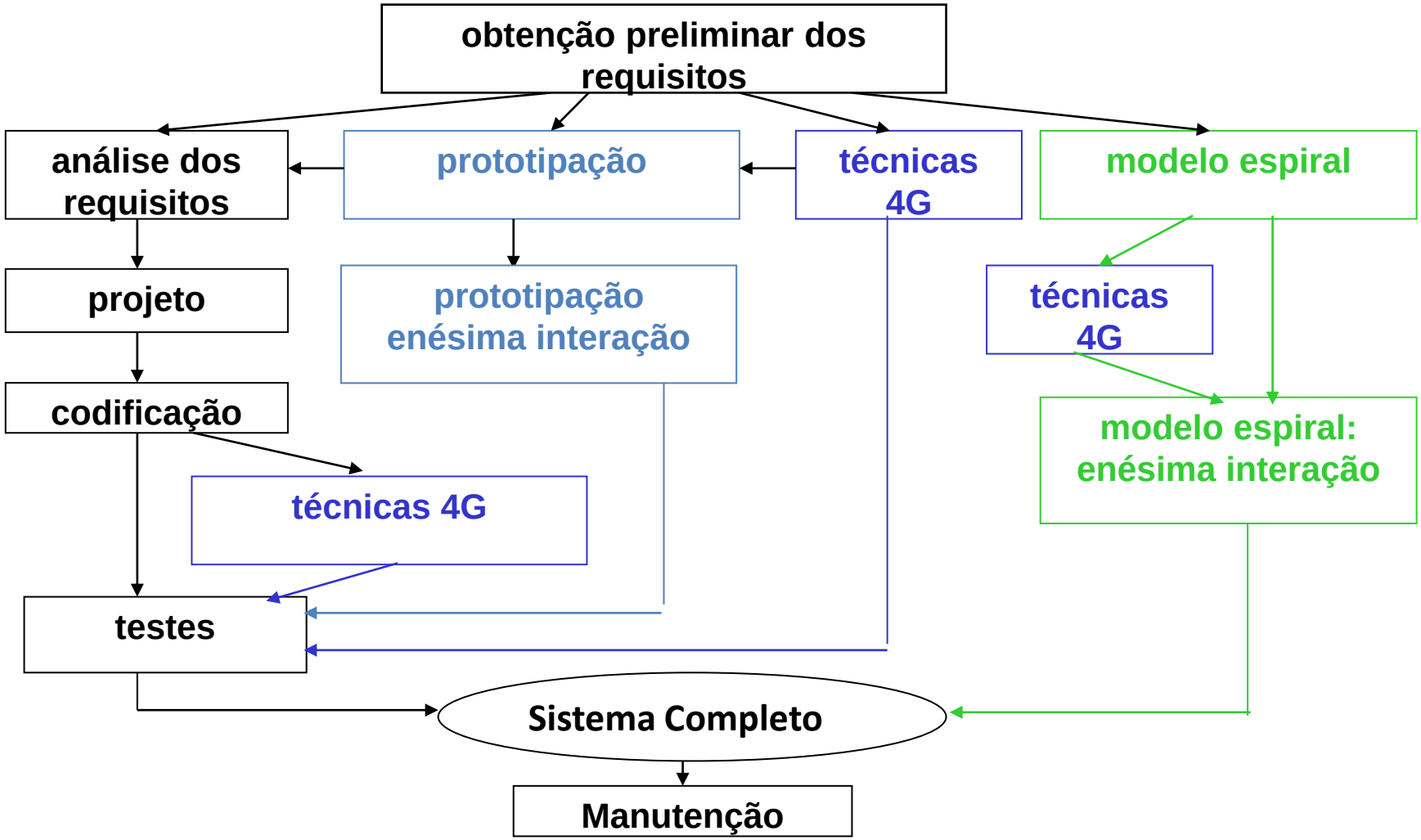
# Escolher um modelo de desenvolvimento para o sistema

- Riscos na integração dos sub-sistemas → **Modelo Cascata**
- Riscos significativos na interface com o utilizador → **Desenvolvimento evolutivo.**
- Riscos de segurança → **Desenvolvimento Formal**



Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição

# Combinando Paradigmas




Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6° Edição
SOMMERVILLE - Engenharia de Software - 8° Edição