Introdução
Engenharia de Software

EMENTA

- Parte 1

  Conceitos de Engenharia de Software. Processo de desenvolvimento de software. Comparação entre os Paradigmas de Desenvolvimento Software. Levantamento, especificação e análise de requisitos.

- Parte 2

  Verificação, validação e manutenção de software. Planejamento e gestão de projetos. Qualidade de software. Métricas de software

Desafio do Desenvolvimento de Software

No mercado atual, não há dúvida de que os profissionais de TI envolvidos com projetos de desenvolvimento de software e soluções corporativas têm um claro desafio:

PRODUZIR soluções mais rápidas, melhores e mais baratas que antes (melhor ainda ser mais rápidas, melhores e mais baratas que a concorrência)

Evolução do Software

- **1970s:**
  - Lower-CASE (*Computer-Aided Software Engineering*) ferramenta
    → programação, depuração (ex. **Jbuilder, JUnit)**
  - Ciclo de vida cascata
  - Desenvolvimento estruturado
- **1980s:**
  - Ciclo de vida espiral
  - Desenvolvimento orientado a objetos

Evolução do Software

- **1990s:**

– Upper-CASE ferramenta

→ fases de planejamento, análise e projeto do programa

(Ex. **JUDE, Poseidon**, ArgoUML)

– Processos

– Modelagem

- **Atualmente:**

– Métodos ágeis

– Desenvolvimento dirigido por modelos

– Linhas de produto

– Experimentação

Níveis dos sistemas de informação dentro de uma organização

Engenharia de Software

- **Métodos e Técnicas** para o disciplinar o processo de desenvolvimento do software

.

O que é Engenharia de Software?

- Fritz Bauer – 1969 (primeira definição)

"*O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais*"

O que é Engenharia de Software?

- IEEE, 1993

"A aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção do software. O estudo de abordagens e princípios a fim de obter economicamente softwares confiáveis e que executem de forma eficiente nas máquinas reais"

O que é Engenharia de Software?

O **engenheiro de software** se utiliza de recursos mais adequadas para determinar quais são os elementos necessários para resolução de um problema complexo

Programador (técnicas)	Engenheiro (técnicas)
1. Paradigma de tentativa e erro	1. Paradigma adaptado ao escopo do sistema
2. Estrutura de Dados	2. Análise e Projeto
3. Linguagens de Programação	3. Ferramentas CASE e SGBD´s

Áreas da Engenharia de Software segundo SWEBOK –

SoftWare Engineering Body of Knowledge

- Requisitos de Software
- Projeto de Software
- Construção de Software
- Teste de Software
- Manutenção de Software
- Gerenciamento de Configuração de Software
- Gerenciamento de Engenharia de Software
- Processo de Engenharia de Software
- Ferramentas e Métodos de Engenharia de Software
- Qualidade de Software

10 Áreas da Engenharia de Software

1 - Requisitos de Software

A Área do Conhecimento de Requisitos de Software está preocupada com a elicitação, análise, especificação e validação da requisitos de software.

10 Áreas da Engenharia de Software

2 – Design(projeto) de Software

É definido como o processo de definição da arquitetura, componentes, interfaces e outras características de um sistema ou componente e também o resultado desse processo.

10 Áreas da Engenharia de Software?

3 - Construção de Software

Criação detalhada de programas funcionais a partir de uma combinação de codificação, verificação, teste unitário, teste integrado e debugging.

10 Áreas da Engenharia de Software?

4 - Teste de Software

Verificação dinâmica do comportamento do programa através do uso de um conjunto finito de casos de teste – adequadamente selecionados de um domínio de execuções usualmente infinito - contra o comportamento esperado deste

10 Áreas da Engenharia de Software?

5 - Manutenção de Software

Atividades de suporte custo-efetivo a um sistema de software, que pode ocorrer antes e após a entrega do software. Após a entrega do software são feitas modificações com o objetivo de corrigir falhas, melhorar seu desempenho ou adapta-lo a um ambiente modificado. Antes da entrega do software são feitas atividades de planejamento.

10 Áreas da Engenharia de Software?

6 - Gerência de Configuração de Software

Identifica a configuração do sistema (características documentadas do hardware e software que o compõem) em pontos discretos no tempo, de modo a controlar sistematicamente suas mudanças e manter sua integridade e rastreabilidade durante o ciclo de vida do sistema.

10 Áreas da Engenharia de Software?

7 - Gerência de Engenharia de Software

a aplicação de atividades de gestão – planejamento, coordenação, medição, monitoramento, controle e divulgação – para garantir que o desenvolvimento e manutenção de software seja sistemática, disciplinada e quantificada.

10 Áreas da Engenharia de Software?

8 - Processo de Engenharia de Software

Define, implementa, mede, gerencia, modifica e aperfeiçoa o processo de desenvolvimento de software. O Processo de engenharia de software inclui atividades técnicas e de gestão dentro dos processos o ciclo de vida de software.

10 Áreas da Engenharia de Software?

9 - Ferramentas e Métodos

Ferramentas de software automatizam o processo de engenharia de software.

Métodos impõem estrutura sobre a atividade de desenvolvimento e manutenção de software com o objetivo de torná-la sistemática e mais propensa ao sucesso.

10 Áreas da Engenharia de Software?

10 - Qualidade de Software

Conjunto de atividades relacionadas com garantia de qualidade de software, que transcende os processos do ciclo de vida de Software.

Elementos da Engenharia de Software?

- É uma disciplina que integra **métodos, ferramentas** e **procedimentos** para o desenvolvimento de software de computador.
- Possibilitar ao gerente o controle do processo de desenvolvimento
- Oferecer ao profissional uma base para a construção de software de alta qualidade

Engenharia de Software - Método

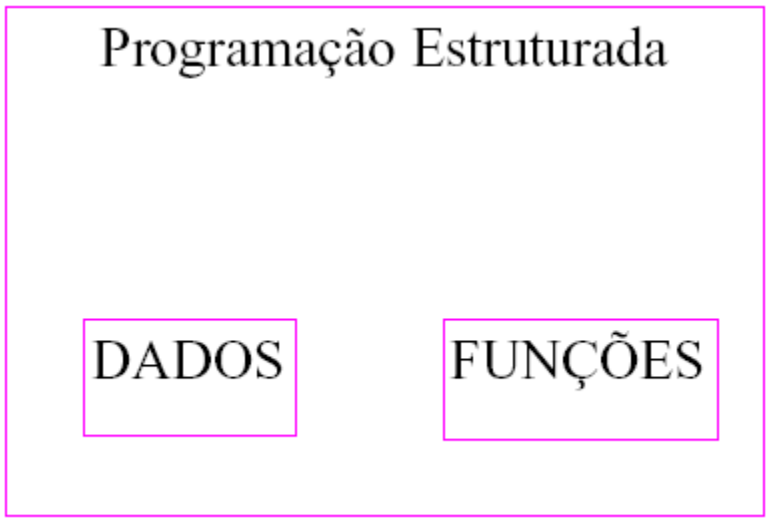
- **Métodos:** proporcionam os detalhes de **"como fazer"** para construir o software.
- Envolvem um amplo conjunto de tarefas.
- Um método de ES é uma aproximação estruturada para o desenvolvimento de software.
- **Objetivo:** Produção de software de alta qualidade de um modo cost-effective (custo-beneficio).

Engenharia de Software - Método

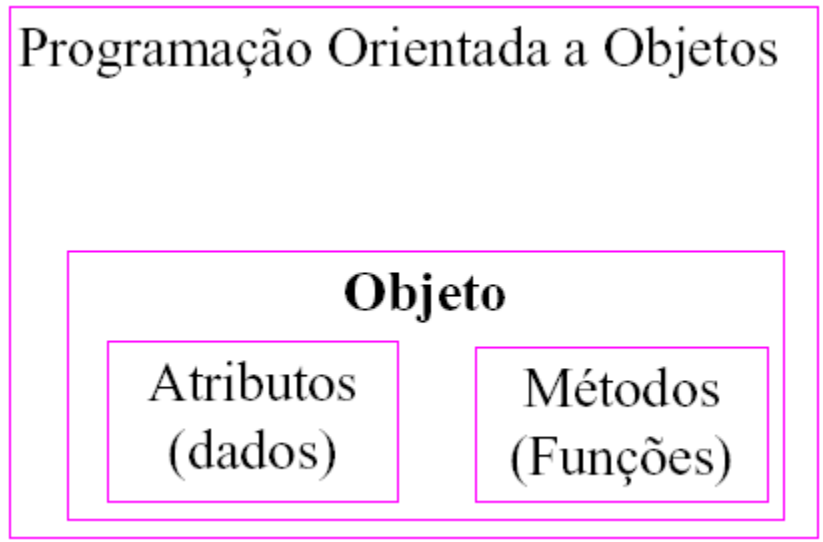
- Início na década de 70 (DeMarco e Jackson):
  - Identificação dos componentes funcionais básicos de um sistema;
  - Orientado à função;
  - Bibliotecas de algoritmos.
- 80s-90s Métodos orientados a objeto (Booch e Rumbaugh).
  - classes,objetos.
  - Atualmente os diferentes métodos estão integrados numa aproximação unificada baseada em Unified Modeling Language (UML).

Engenharia de Software - Método

década de 70



década de 80s-90s



Engenharia de Software - Método

- Métodos devem incluir os seguintes componentes:
  - Descrição gráficas
  - Regras
  - Recomendações
  - Diretrizes de processo

Engenharia de Software - Ferramentas

- **Ferramentas** : fornecem suporte automatizado ou semi aos métodos.
  - Existem atualmente ferramentas para sustentar cada um dos métodos
  - Quando as ferramentas são integradas é estabelecido um sistema de suporte ao desenvolvimento de software chamado *CASE - Computer Aided Software Engineering*

O que é CASE

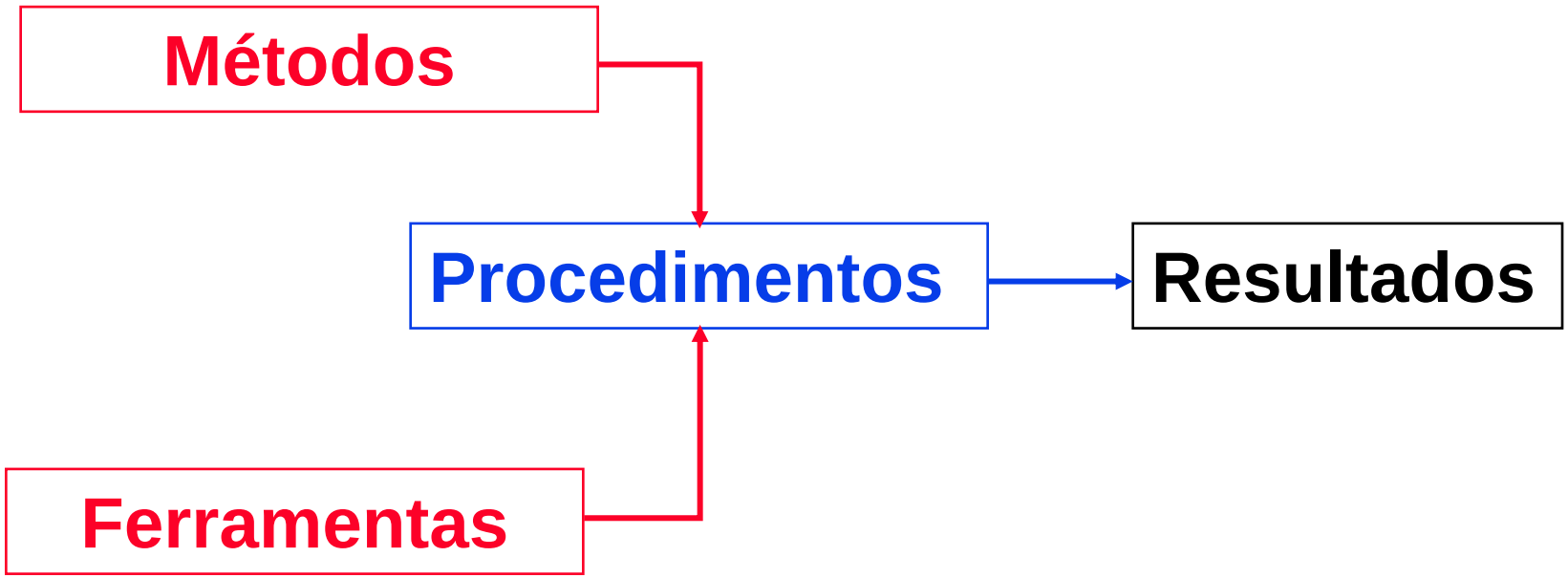
(Computer-Aided Software Engineering)

- Sistemas de software automatizado utilizados para apoiar as atividades de processo de software.
- Upper-CASE - Ferramenta para dar apoio às fases iniciais do processo de software.
- Lower-CASE - Ferramenta para dar apoio à implementação a aos testes.
- Ex.(Poseidon para UML , ArgoUML....)

Engenharia de Software - Procedimentos

- **Procedimentos** : constituem o elo de ligação que mantém juntos os métodos e as ferramentas para desenvolvimento do software.
  - Sequência em que os métodos serão aplicados.
  - Produtos (*deliverables*) que se exige que sejam entregues.
  - Controles que ajudam assegurar a qualidade e coordenar as alterações.
  - Marcos de referência que possibilitam administrar o progresso do software.

Engenharia de Software - Procedimentos



Princípios da Engenharia de Software

- Todo engenheiro de software deve desenvolver com:
  - Rigor e Formalidade
  - Separação de interesses
  - Modularidade
  - Abstração
  - Antecipação de mudanças
  - Generalidade
  - Possibilidades de evolução

Engenharia de Software
Visão Genérica: 3 Fases

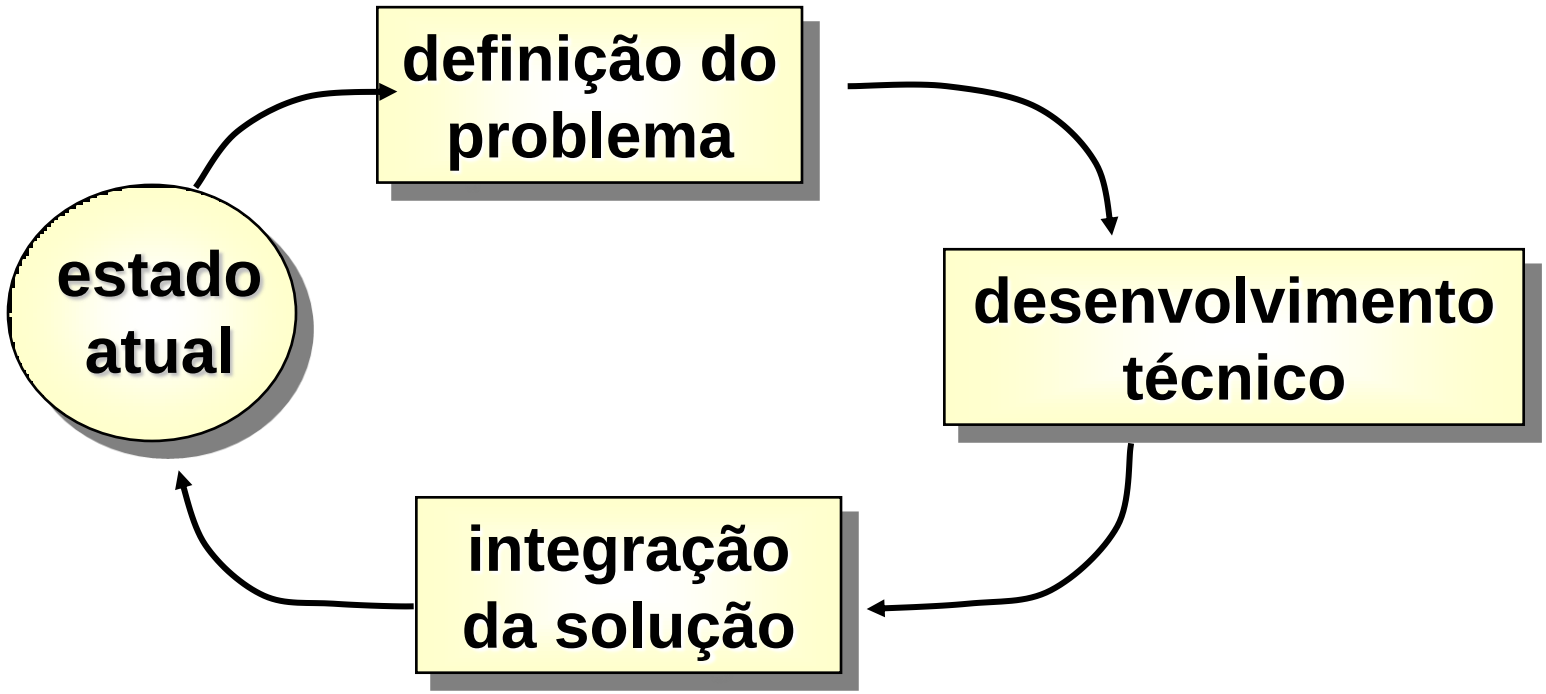
1 - Definição - "**o que?**"
- Engenharia do Sistema
- Planejamento do Projeto
- Análise de Requisitos

2 - Desenvolvimento -"**como?**"
- Projeto
- Geração do Código
- Teste

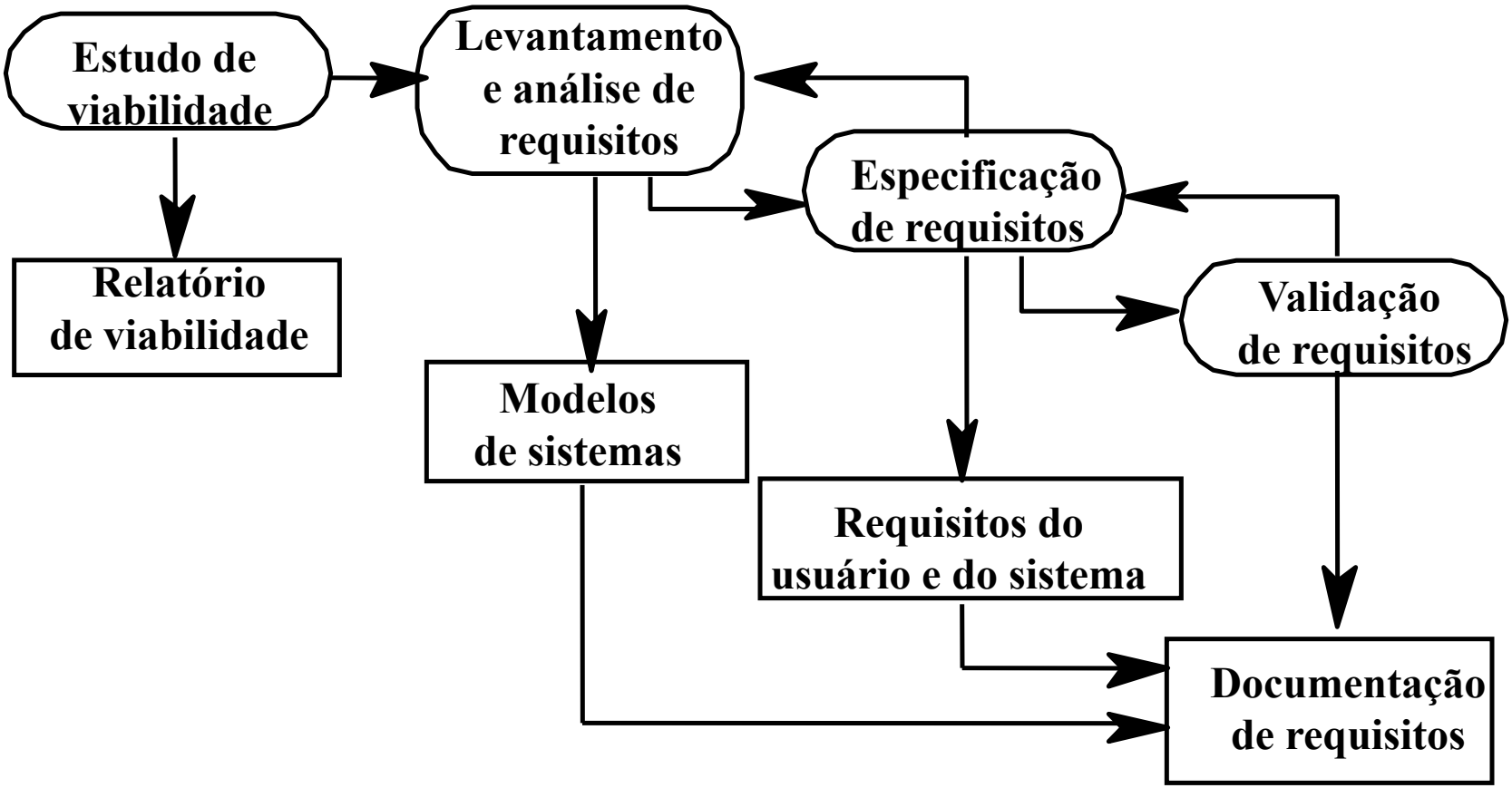
3 - Manutenção

Modelo de Processo de Software

Processo → conjunto de atividades ações e tarefas realizadas na criação de algum produto



Modelo de Processo de Software



Modelo 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**

Modelo de Processo de Software (paradigmas)

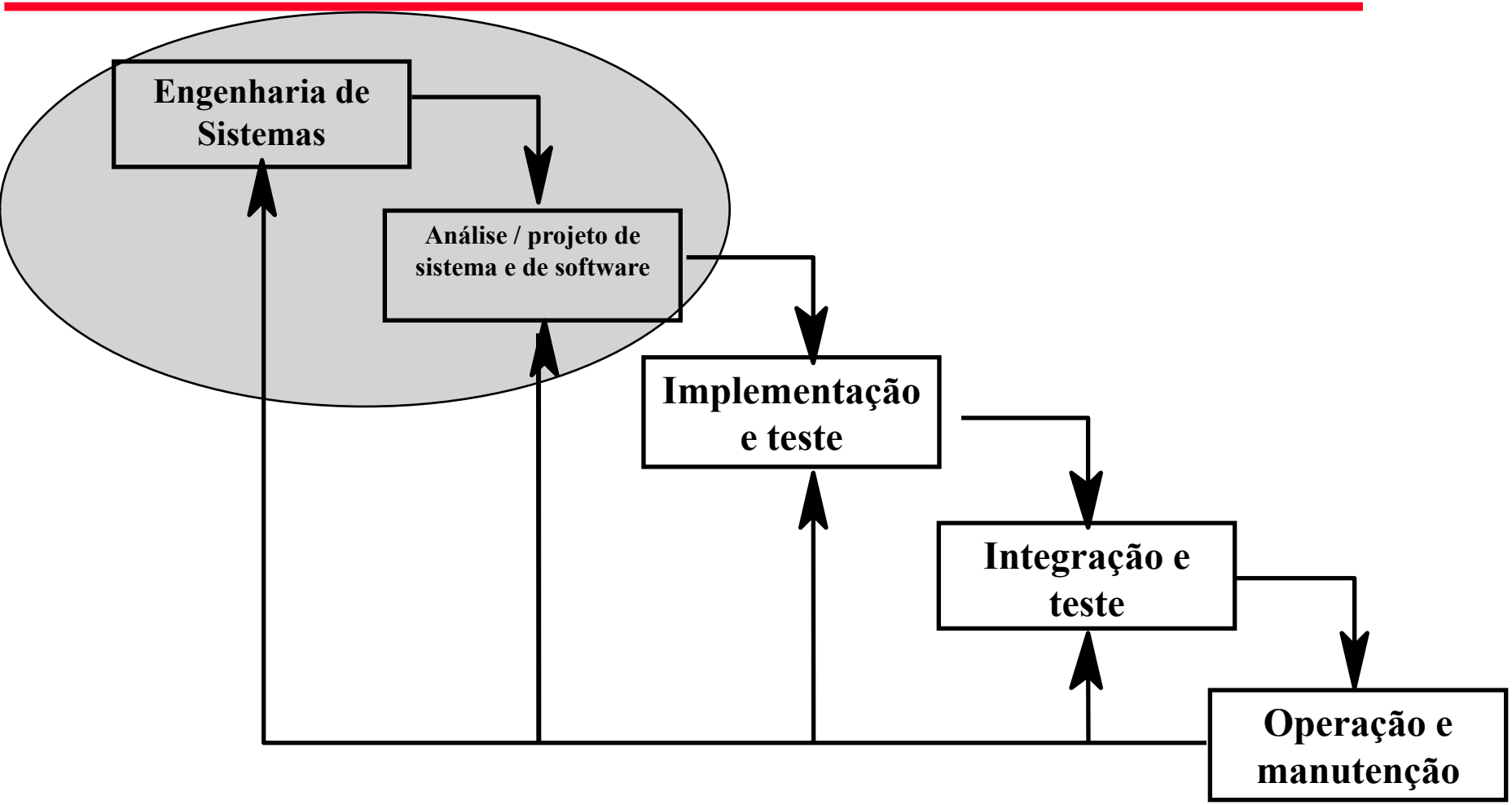
É escolhido com base:

- Na natureza do projeto e da aplicação.
- Nos métodos e ferramentas a serem utilizados.
- Nos controles e produtos que precisam ser entregues.

Modelo Sequencial Linear
(ciclo de vida clássico)

- Método sistemático e sequencial
- O resultado de uma fase se constitui na entrada da outra.
- Cada fase é estruturada como um conjunto de atividades que podem ser executadas por pessoas diferentes, simultaneamente.

Modelo Sequencial Linear
(ciclo de vida clássico)



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;

  - 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.**

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

Resultado → o contrato de desenvolvimento

Modelo Sequencial Linear (ciclo de vida clássico)

- **PROJETO**
  - É definida a solução do problema
  - **Concentra-se em:**
    - Estrutura de Dados;
    - Arquitetura de Software;
    - Detalhes Procedimentais e
    - Caracterização de Interfaces.

Resultado → documentação de especificação de projeto

Modelo Sequencial Linear (ciclo de vida clássico)

- **IMPLEMENTAÇÃO**

  • tradução das representações do projeto para uma linguagem "artificial" resultando em instruções executáveis pelo computador.
  • O projeto é transformado em um programa, ou unidades de programa. (Teste de Unicidade)

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

Modelo Sequencial Linear (ciclo de vida clássico)

- **INTEGRAÇÃO**

  - Programas ou unidades de programas são integrados e testados como sistema.

  - Integração incremental → programas ou unidades são integradas à medida em que forem sendo desenvolvidos.

Resultado → produto pronto para ser entregue ao cliente.

Modelo Sequencial Linear (ciclo de vida clássico)

- **OPERAÇÃO / MANUTENÇÃO**
  - **Operação**
    - Instalação e configuração
    - Utilização – inicialmente operado por um grupo de usuário
  - **Manutenção**
    - Corretiva
    - Adaptativa
    - Evolutiva.

Resultado → produto em funcionamento.

Problemas com o Modelo Sequencial Linear (ciclo de vida clássico)

• Todo o planejamento é orientado para a entrega do produto de software em uma data única.

• Fornece pouca visibilidade do estado do projeto

• Logo no início é difícil estabelecer explicitamente todos os requisitos. No começo dos projetos sempre existe uma incerteza natural.

Modelo Evolucionário

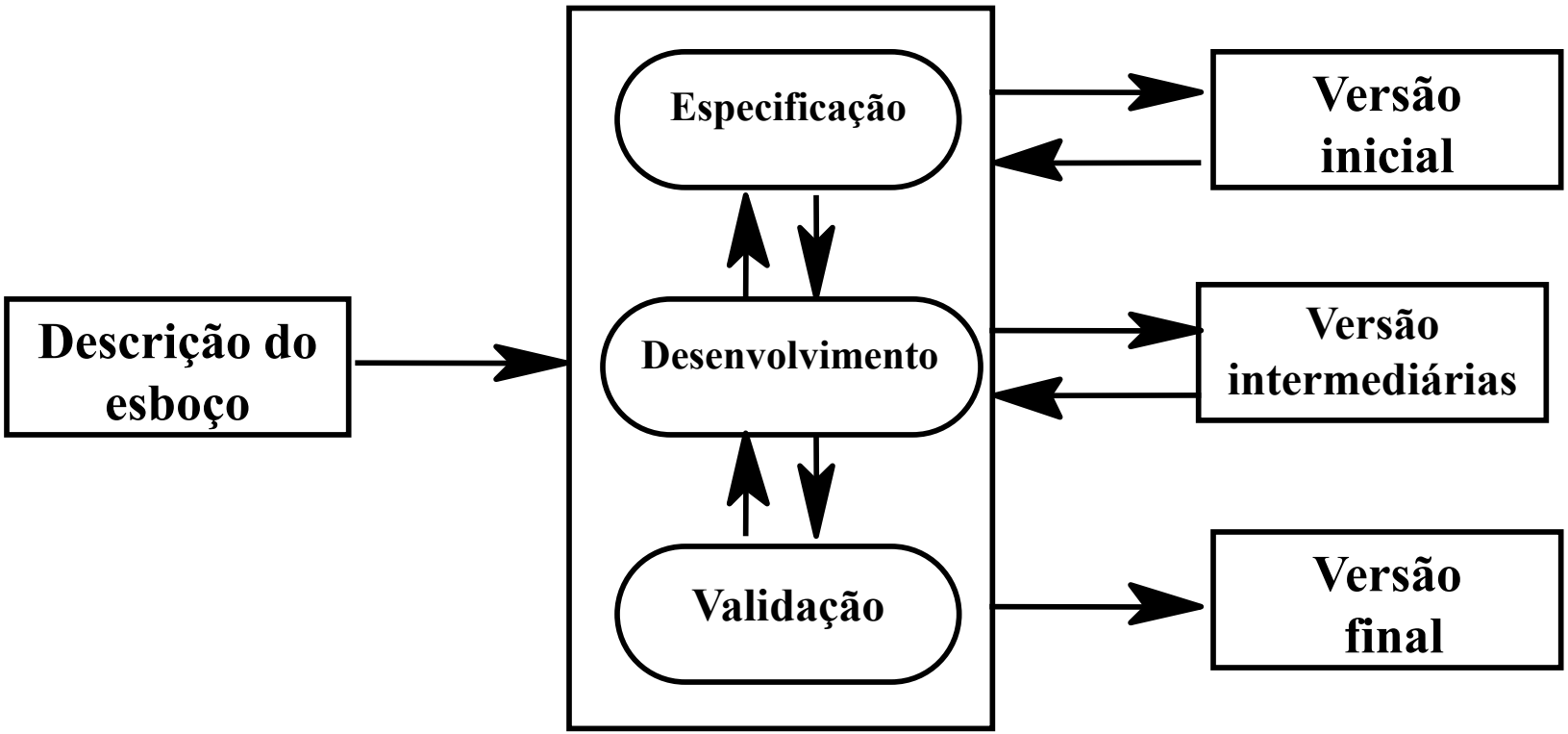
- Abordagem baseada na ideia de desenvolver uma implementação inicial, expor o resultado ao comentário do usuário e fazer seu aprimoramento por meio de muitas versões.

- As atividades de desenvolvimento e validação são desempenhadas paralelamente, com um rápido feedback entre elas.

- Os detalhes e extensões ainda devem ser definidos

Ex: editor de texto

Modelo Evolucionário

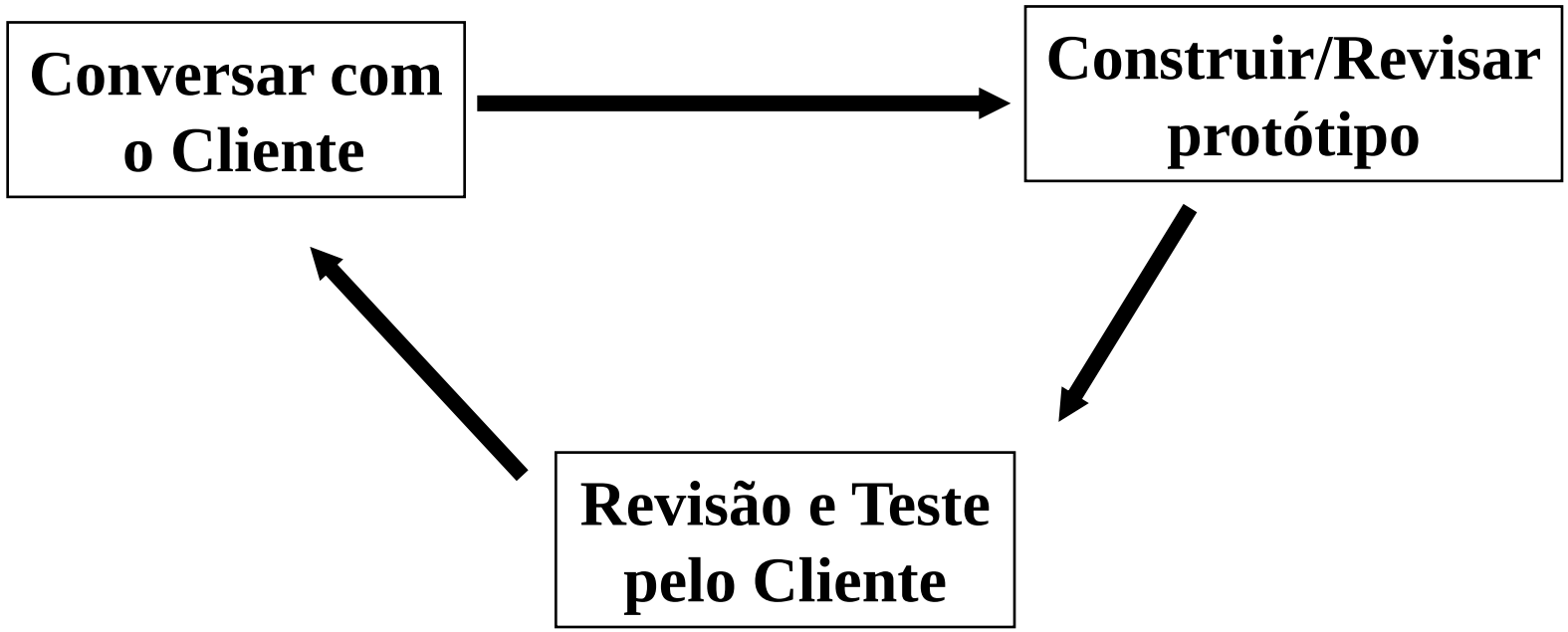


Modelo Evolucionário

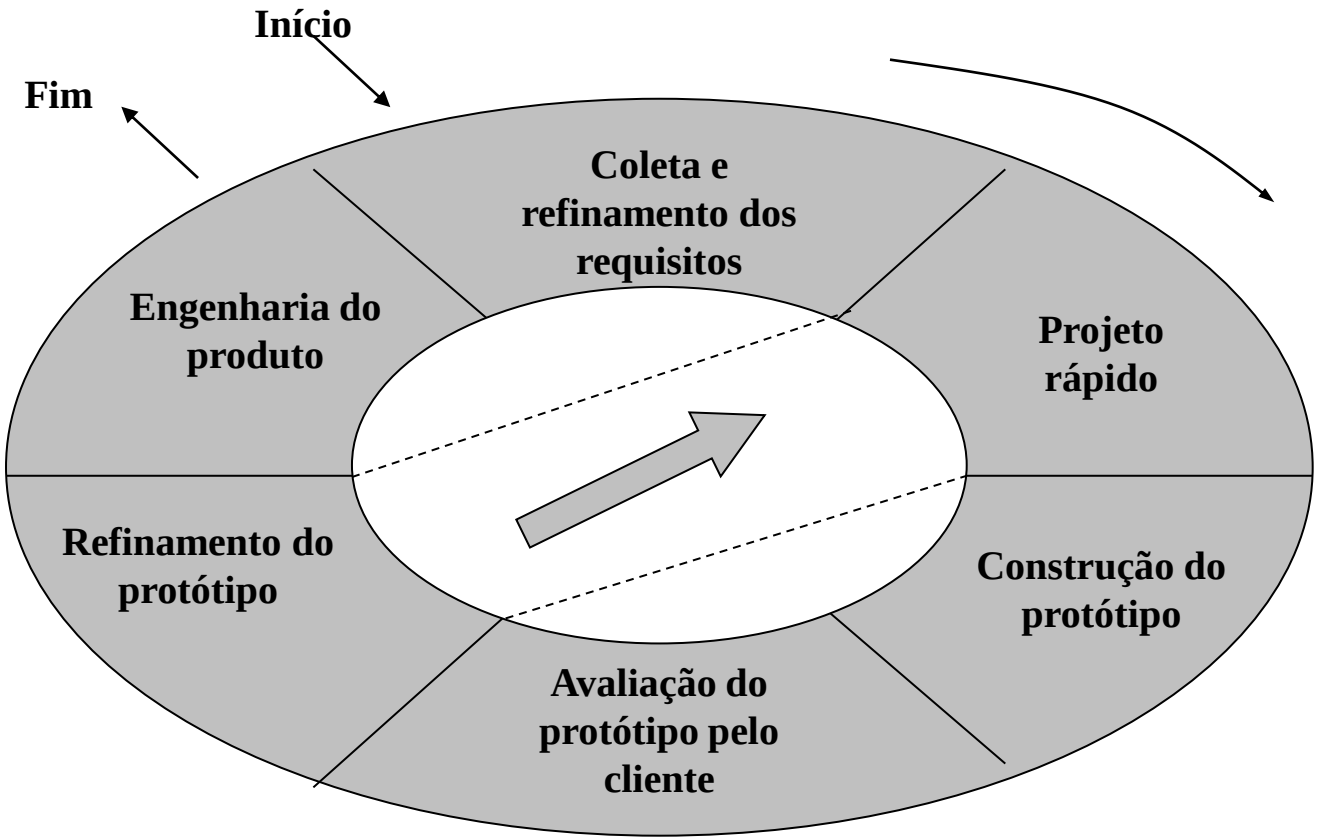
Tipos:

- Prototipação
- Incremental
- Espiral
- Métodos Ágeis

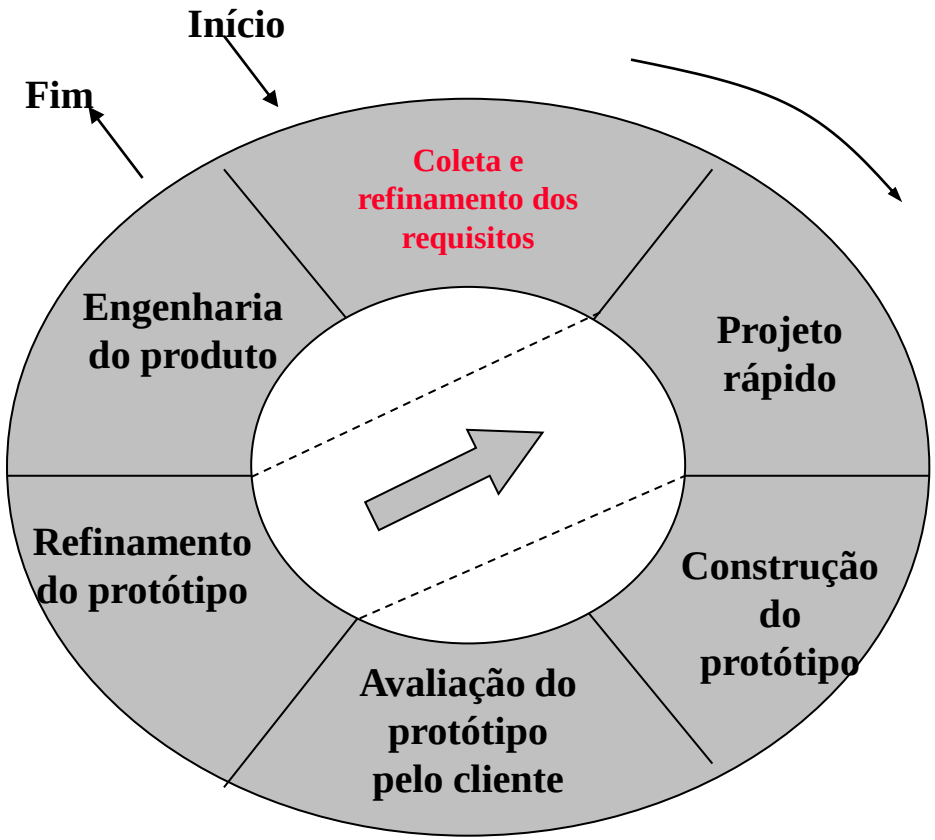
PROTOTIPAÇÃO



PROTOTIPAÇÃO



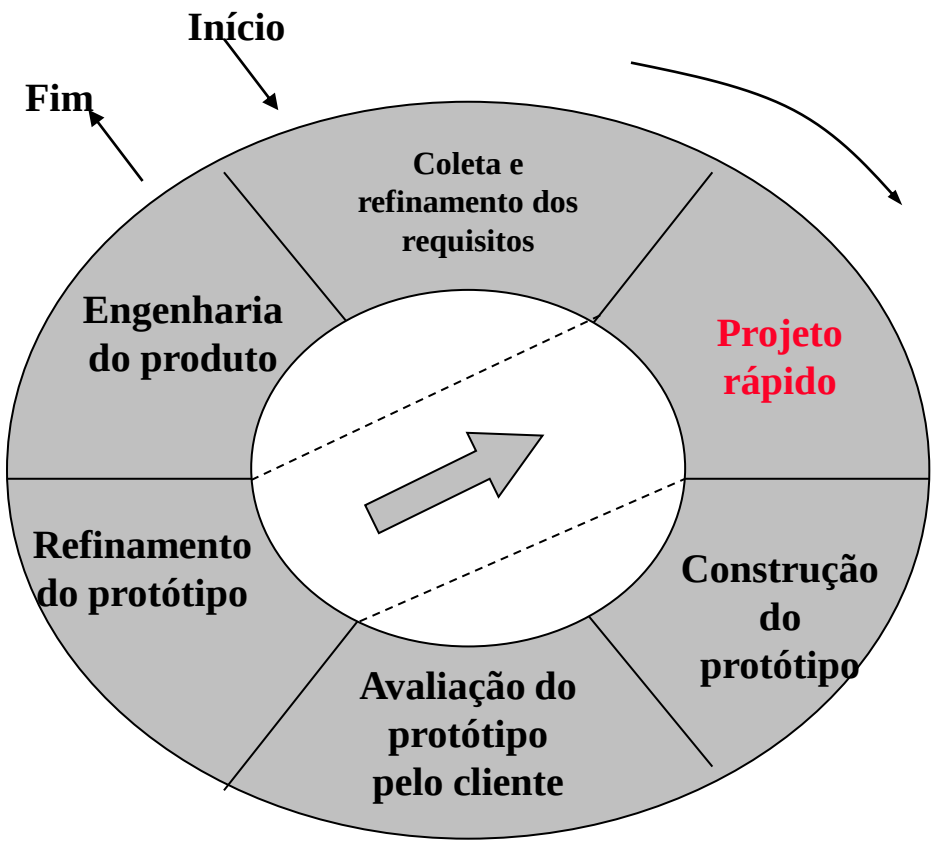
PROTOTIPAÇÃO



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

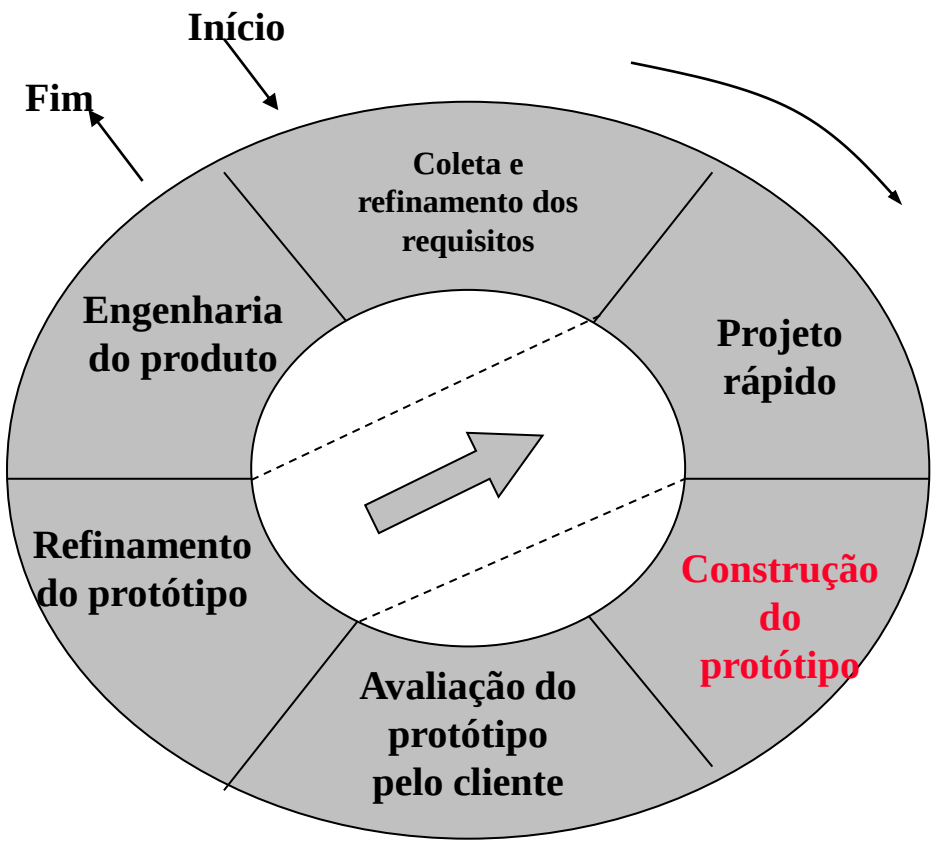
PROTOTIPAÇÃO



PROJETO RÁPIDO:

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

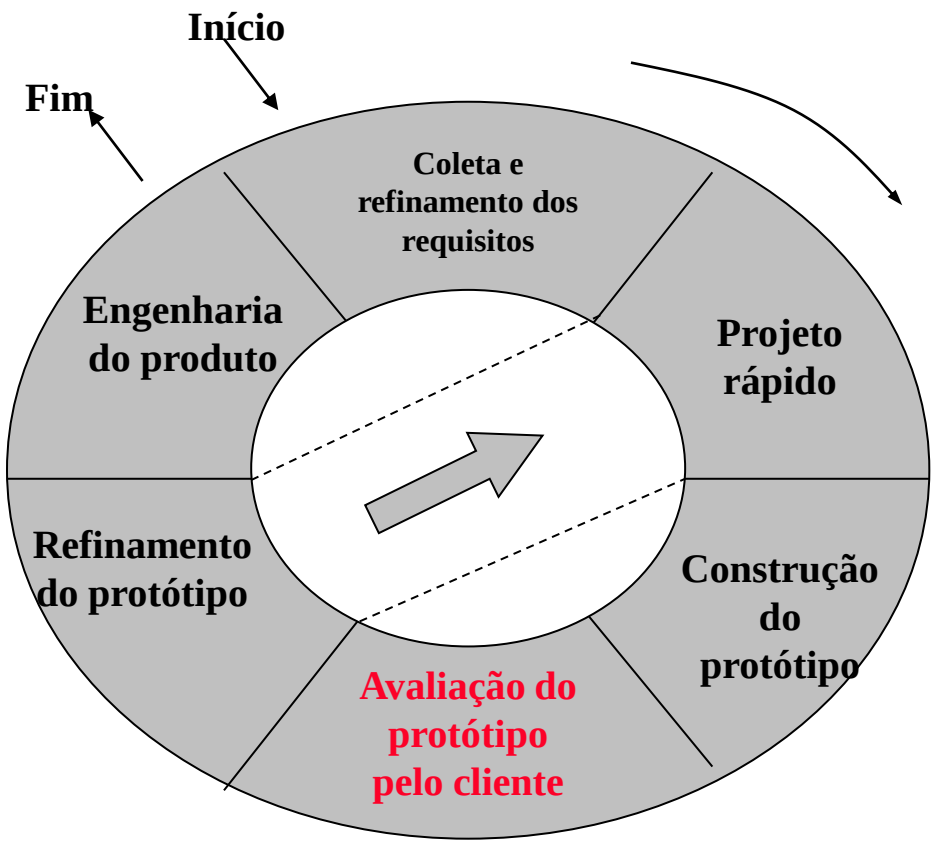
PROTOTIPAÇÃO



CONSTRUÇÃO PROTÓTIPO:

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

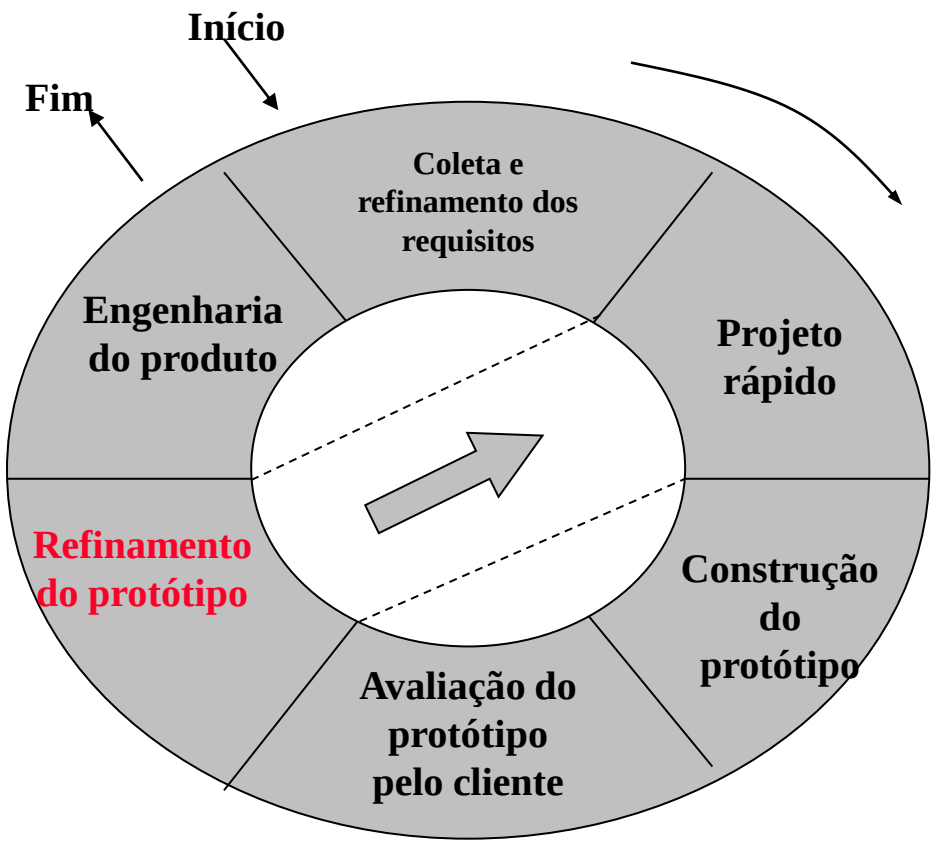
PROTOTIPAÇÃO



AVALIAÇÃO DO PROTÓTIPO:

Cliente e desenvolvedor avaliam o protótipo

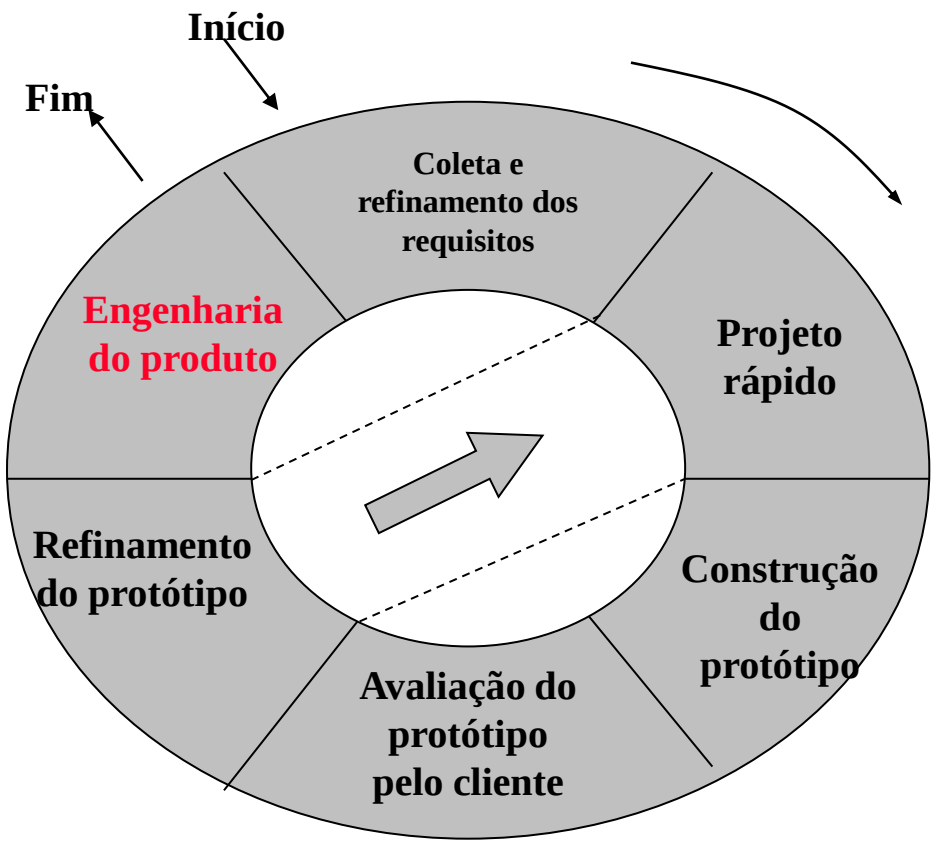
PROTOTIPAÇÃO



REFINAMENTO DOS REQUISITOS:

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

PROTOTIPAÇÃO



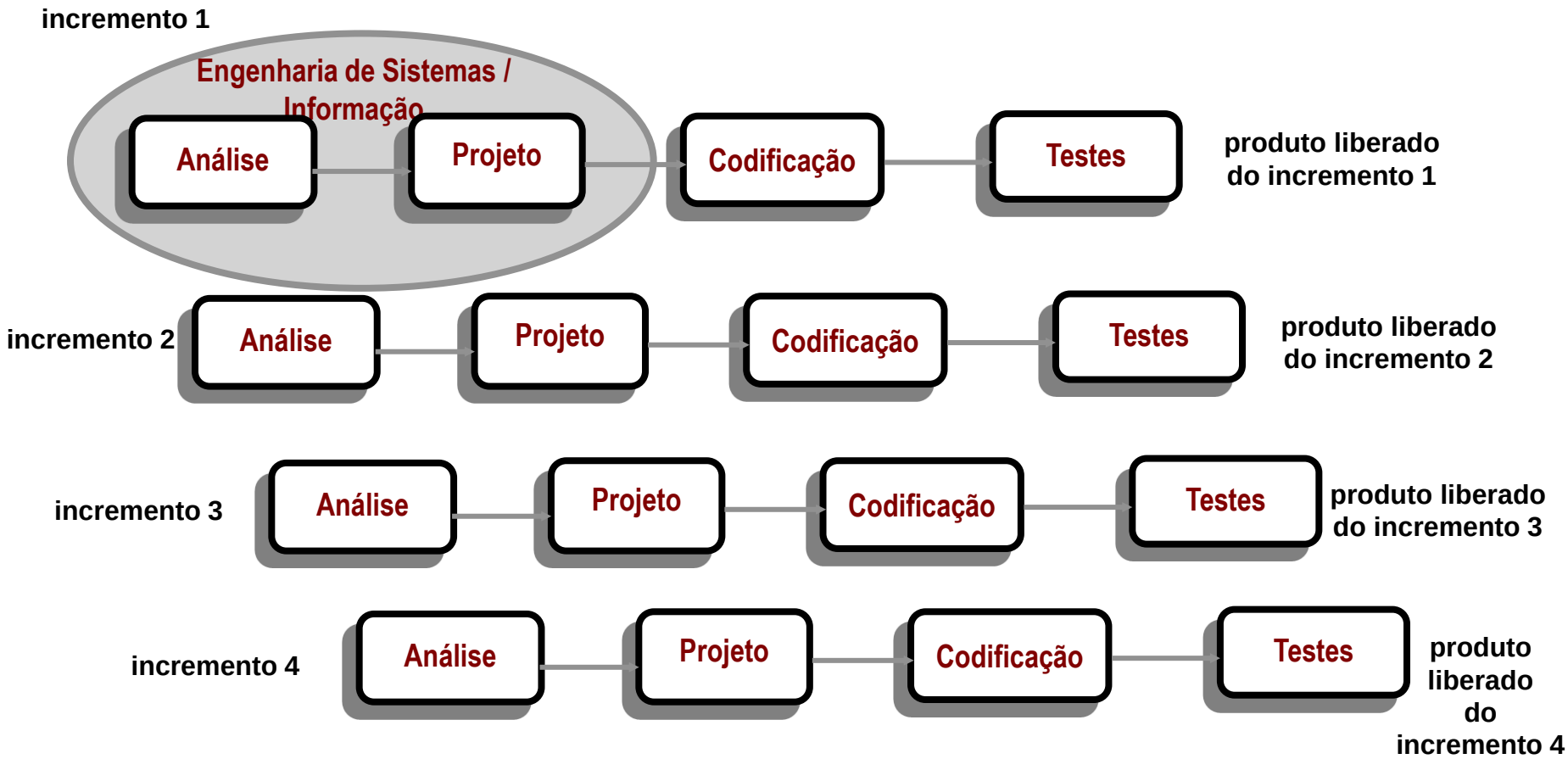
CONSTRUÇÃO PRODUTO:

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

Problemas com a Prototipação

- O processo não é visível.
- Os sistemas são freqüentemente mal-estruturados e mal-documentados.
- Pode exigir ferramentas e técnicas especiais.
- Processo não é claro, dificuldade de planejamento e gerenciamento.

Modelo Incremental

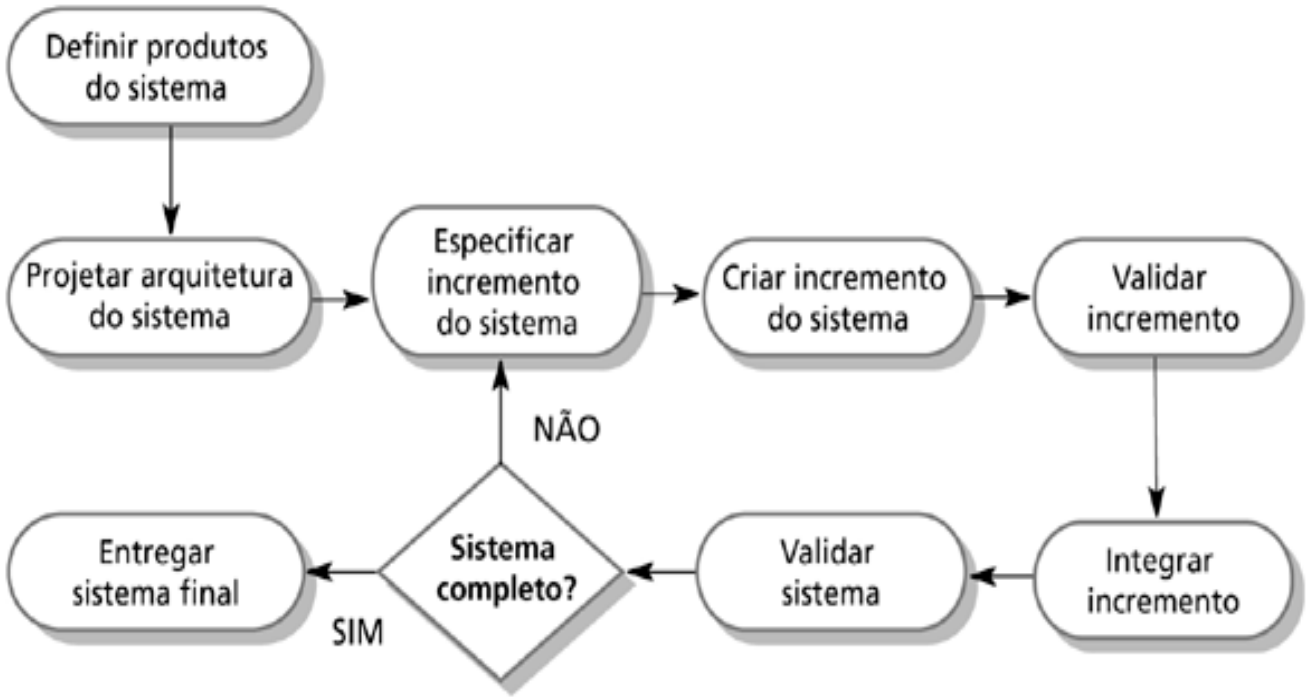


Fonte: Engenharia de Software 8º Edição / ©Ian Sommerville 2007
Engenharia de Softawre 6º Edição / Roger Pressman
Engenharia de Software 1º Edição / Ariadne Carvalho

Um processo de desenvolvimento iterativo

Figura 17.1

Processo de desenvolvimento iterativo.



Vantagens do desenvolvimento 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.

Problemas com desenvolvimento 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.

- **Problemas de validação**

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

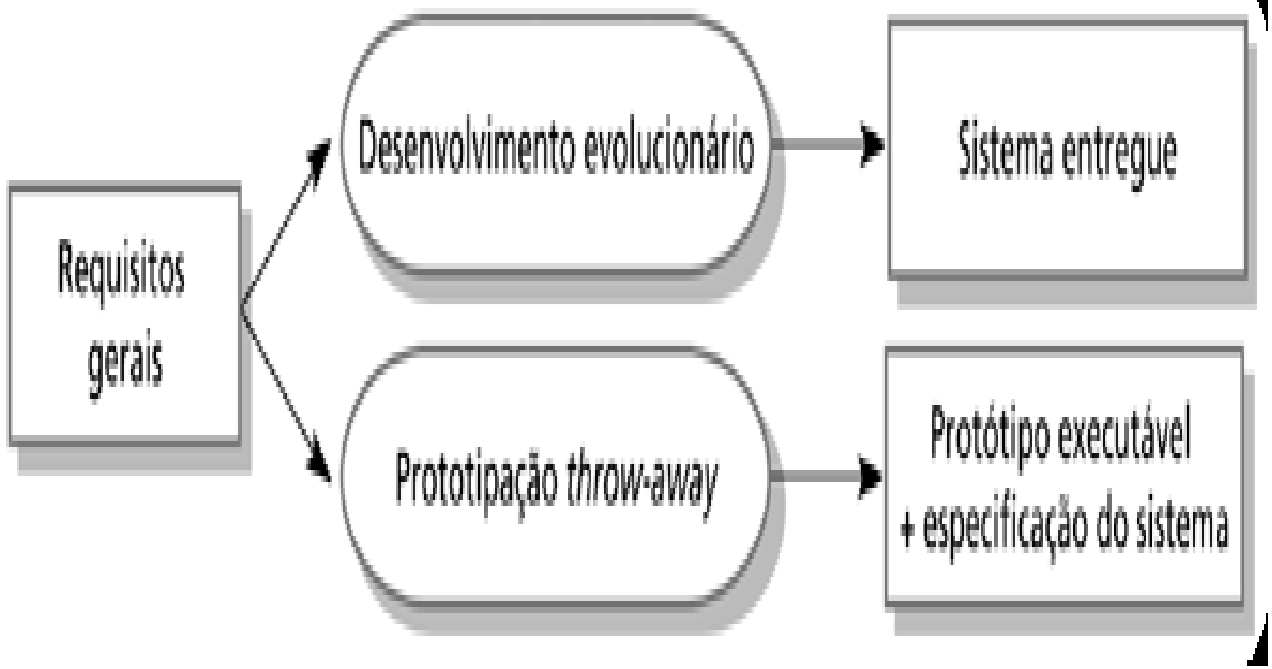
- **Problemas de manutenção**

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

Desenvolvimento incremental e prototipação

Figura 17.2

Desenvolvimento incremental e prototipação.

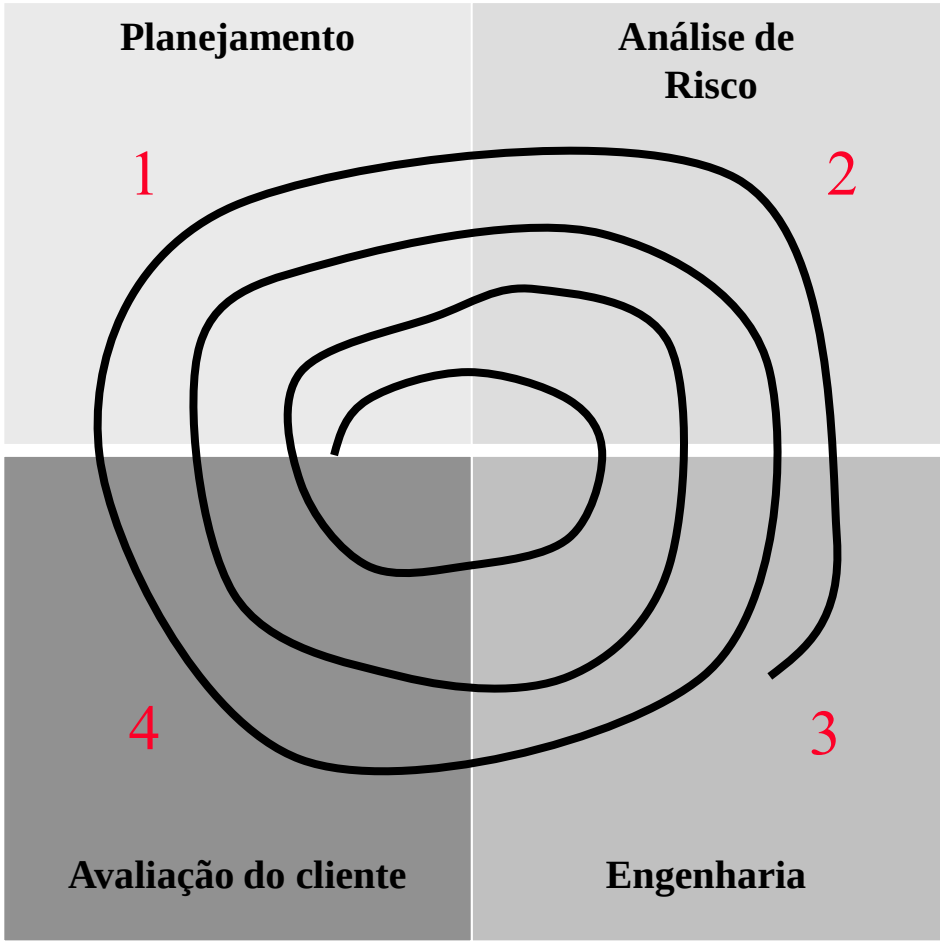


Desenvolvimento Incremental x Prototipação

- Para alguns sistemas grandes, o desenvolvimento e a entrega **incremental e iterativa** pode não ser prático; isso é especialmente verdadeiro quando múltiplas equipes estão trabalhando em localidades diferentes.

- **Prototipação,** no sentido de um sistema experimental, é desenvolvido como base para a formulação de requisitos que podem ser usados. O processo de prototipação inicia com aqueles requisitos que não são bem compreendidos. Esse sistema é descartado quando a especificação do sistema for acordado.

Modelo Espiral (Boehm)



Planejamento: determinação dos objetivos, alternativas e restrições

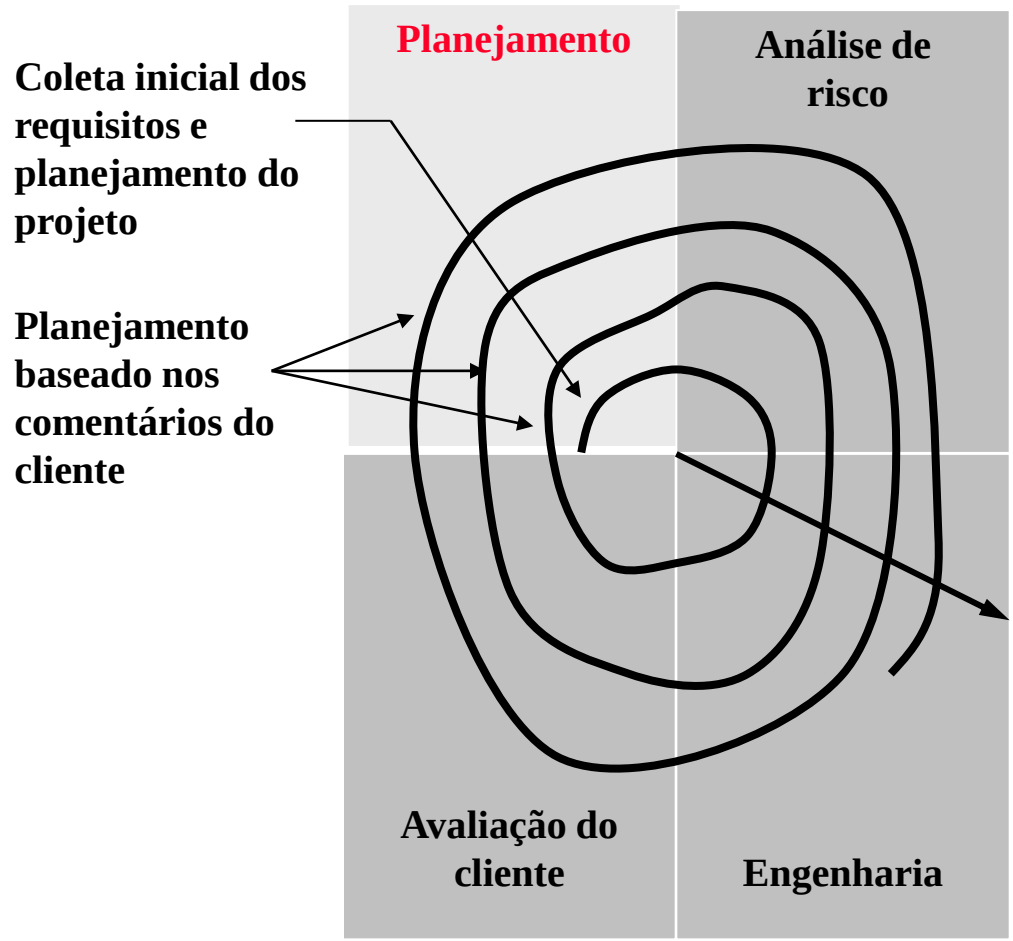
Análise dos riscos: análise de alternativas e identificação /resolução dos riscos

Engenharia: desenvolvimento do produto no "nível seguinte"

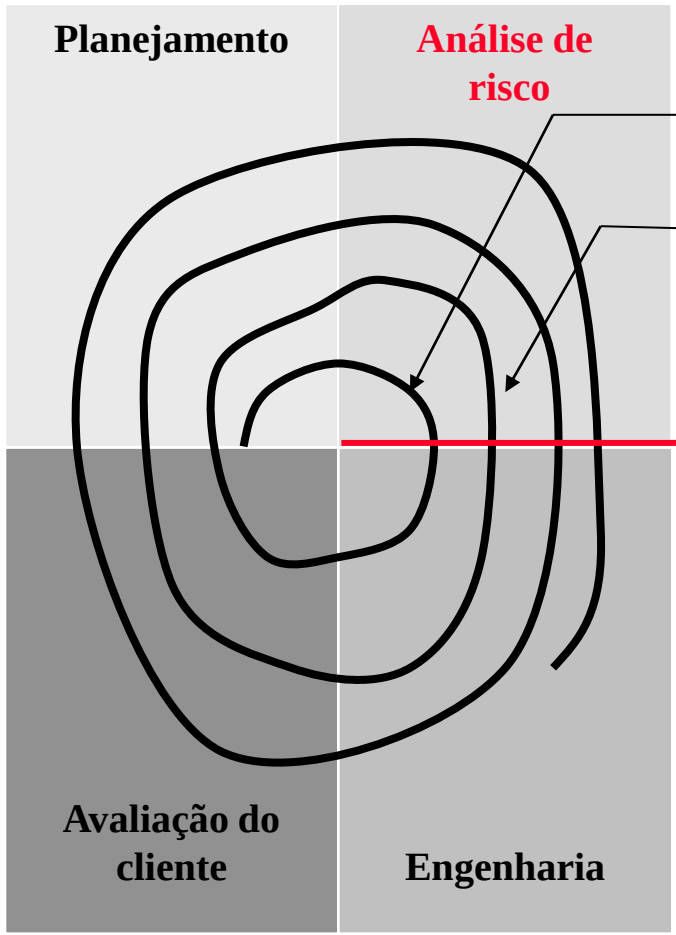
Avaliação feita pelo cliente: avaliação dos resultados da engenharia

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.
- Estratégias alternativas, dependendo dos riscos detectados, podem ser planejados.



Modelo Espiral

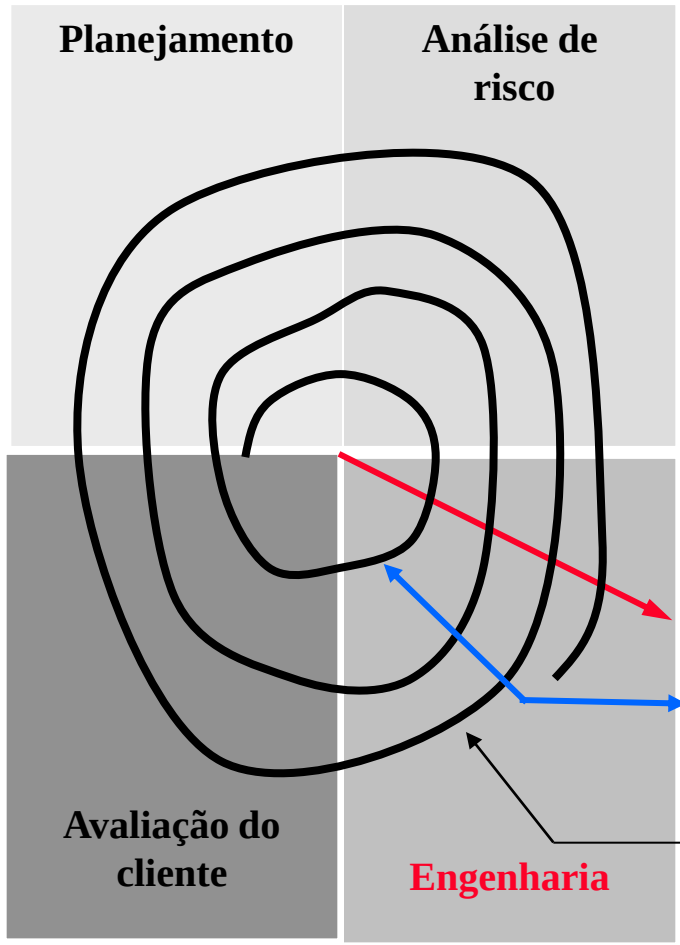


Para cada risco do projeto identificado em Planejamento é levada a cabo uma análise detalhada.

Decisão de prosseguir/não prosseguir

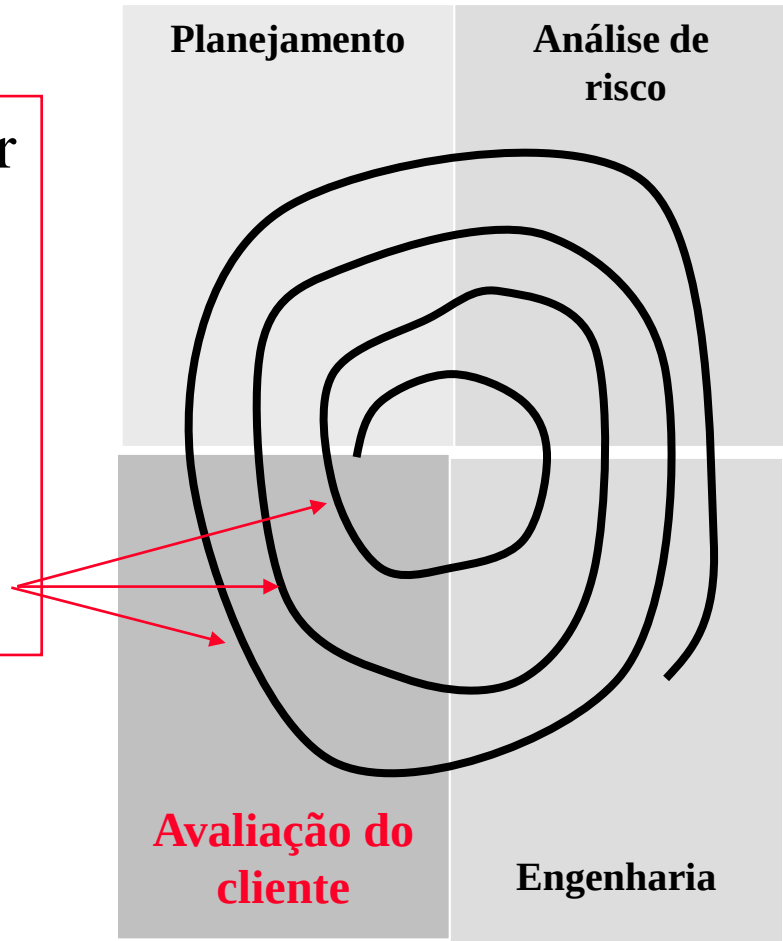
- Riscos:
  - Riscos de projeto
  - Riscos técnicos
  - Riscos de produto

Modelo Espiral

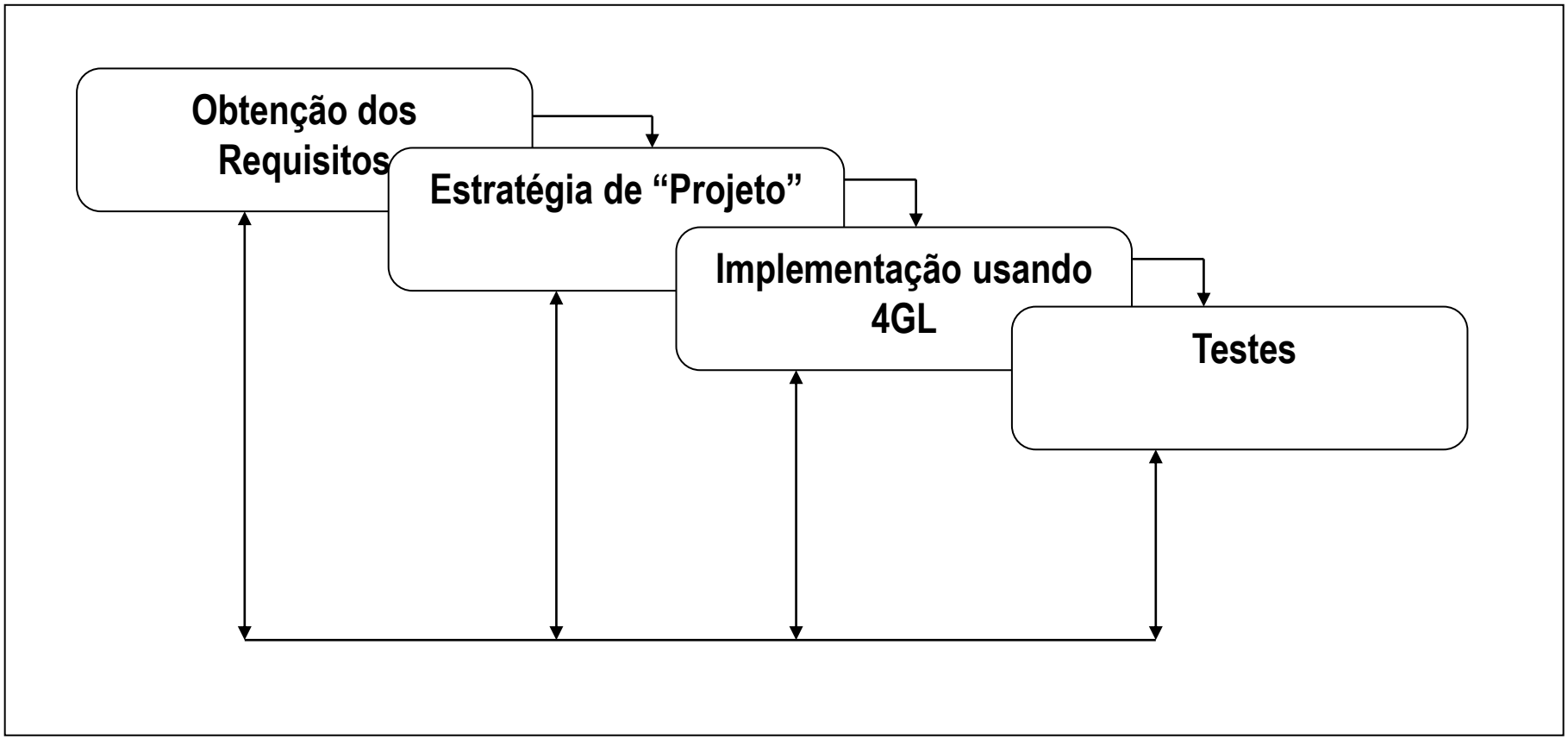


Modelo Espiral

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



Técnicas de 4ª Geração



Técnicas de 4ª Geração

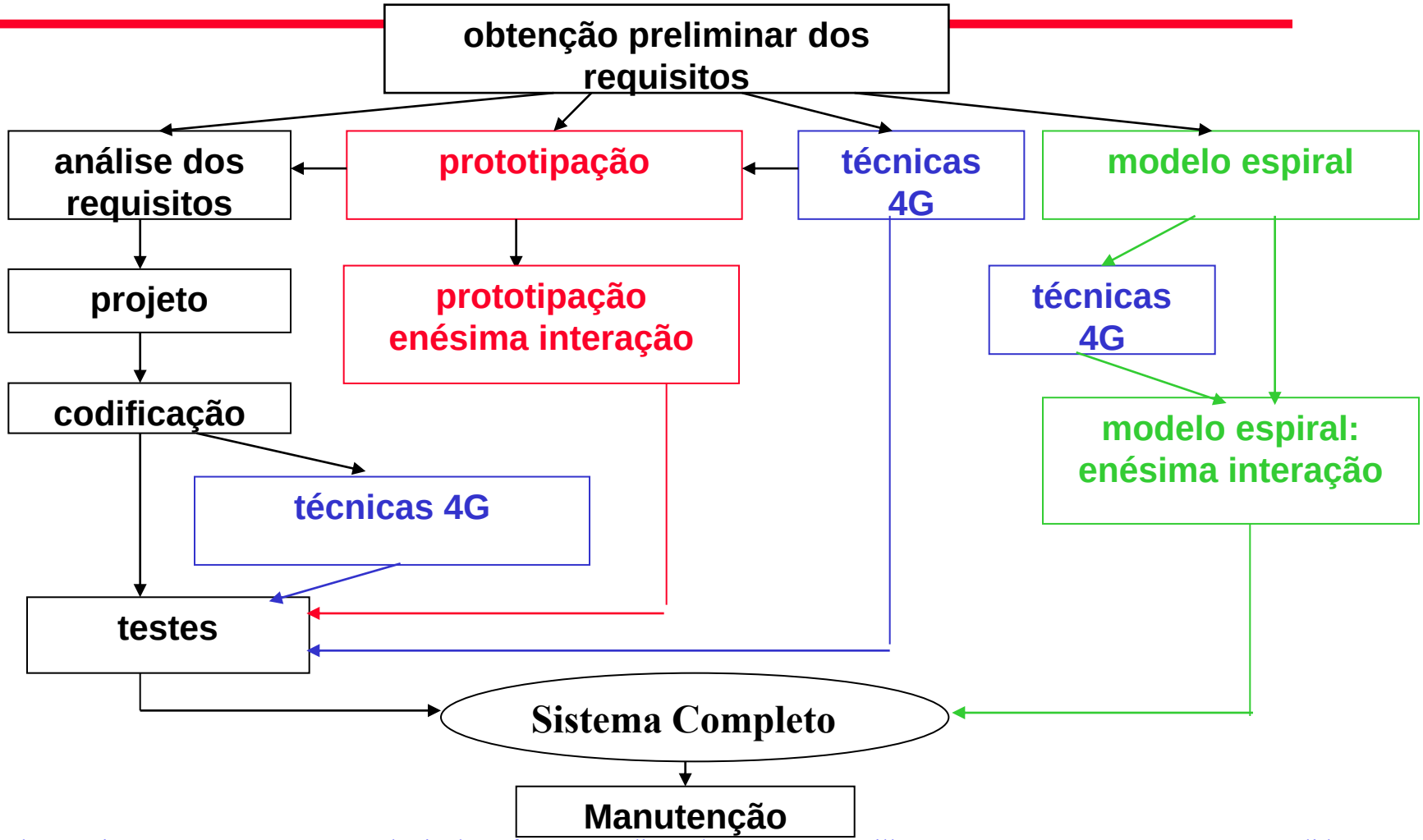
- Concentra-se na capacidade de se especificar o software a uma máquina em um nível que esteja próximo à linguagem natural.

- 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.

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

Combinando Paradigmas



Fonte: Engenharia de Software 8º Edição / ©Ian Sommerville 2007
Engenharia de Softawre 6º Edição / Roger Pressman
Engenharia de Software 1º Edição / Ariadne Carvalho

Qual Modelo Escolher

- Escolher um modelo de desenvolvimento para o sistema
  - Riscos significativos na interface com o utilizador → desenvolvimento evolutivo.
  - Riscos de segurança → Desenvolvimento Formal
  - Riscos na integração dos sub-sistemas → Modelo Cascata

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.

- Os métodos ágeis são, provavelmente, os mais adequados para sistemas de negócio de pequeno/médioporte.

Métodos Ágeis

• 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.

Princípios dos Métodos Ágeis

Envolvimento do cliente	Clientes devem ser profundamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema.
Entrega incremental	O software é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada incremento.
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.
Aceite as mudanças	Projete o sistema para acomodar mudanças.
Mantenha a simplicidade	Elimine a complexidade do sistema.

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.
- A priorização de mudanças pode ser difícil onde existem múltiplos stakeholders.
- A manutenção da simplicidade requer trabalho extra.
- Problemas nos contratos.

XP - 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.
- 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.

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

Os 4 valores de XP

- Comunicação
- Simplicidade.
- Retorno (feedback)
- Coragem

Práticas do eXtreme Programming

Planejamento Incremental	consiste em decidir o que é necessário ser feito e o que pode ser adiado no projeto.
Pequenos releases	Conjunto mínimo útil de funcionalidade é desenvolvido
Projeto simples	Projeto suficiente para atender aos requisitos atuais.
Desenvolvimento test-first	Uso um framework automatizado.
Refactoring	Espera-se que todos os desenvolvedores recriem o código continuamente.
Programação em pares	Os desenvolvedores trabalham em pares.
Propriedade coletiva	Os pares trabalham em todas as áreas do sistema.
Integração contínua	Tarefa concluída é automaticamente integrada ao sistema.
Ritmo sustentável	Não aceitar grande quantidade de horas extras.
Cliente on-site	Um usuário do sistema deve estar disponível em tempo integral.

Cenários de requisitos

- Em um processo XP, os requisitos de usuários são expressos como cenários ou histórias de usuários.
- Essas histórias são escritas em cartões, e a equipe de desenvolvimento quebra-os em tarefas de implementação. Essas tarefas são as bases das estimativas de cronograma e de custo.
- O cliente escolhe as histórias para inclusão no próximo release, baseado nas suas prioridades e nas estimativas de cronograma.

Cartão de estória para baixar documentos

Figura 17.4

Cartão de histórias para baixar documentos.

Baixando e imprimindo um artigo

Primeiro, você seleciona o artigo que deseja em uma lista. Depois você precisa informar ao sistema como quer pagar pelo artigo — isso pode ser feito por meio de uma assinatura, de uma conta empresarial ou por cartão de crédito.

Após, você obtém do sistema um formulário de direitos autorais para preenchimento. Após enviar o formulário, o artigo desejado é baixado para seu computador.

Em seguida, você escolhe uma impressora para imprimir uma cópia do artigo. Você informa ao sistema que a impressão foi bem-sucedida.

Se o artigo for somente para impressão, você não poderá manter uma versão em PDF, de modo que o artigo será excluído automaticamente de seu computador.

Programação Pareada

- Erro de um detectado imediatamente pelo outro (grande economia de tempo).
- Maior diversidade de idéias, técnicas, algoritmos.
- Enquanto um escreve, o outro pensa em contra exemplos, problemas de eficiência, etc.
- Vergonha de escrever código feio (*gambiarras*) na frente do seu par.

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.

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).