# SOFTWARE PROCESSES

Ian Sommerville, 8° edição – Capítulo 4

Aula de Luiz Eduardo Guarino de Vasconcelos

# Objetivos

- Introduzir modelos de processo de software
- Descrever uma variedade de modelos de processo e quando eles podem ser usados
- Descrever esboços de modelos de processo para engenharia de requisitos, desenvolvimento de software, teste e evolução
- Apresentar a tecnologia CASE para dar suporte às atividades de processo de software

# Topics covered

- Software process models
- Process iteration
- Process activities
- The Rational Unified Process
- Computer-aided software engineering

# Windows Vista

- 5000 desenvolvedores (sem incluir pessoal não técnico);
- 50 milhões de linhas de código;
- 16 milhões de linhas de código somente nos últimos 3 anos;
- Versões compiláveis todos os dias;
- Testes de regressão;
- Intervalo de 3 dias para uma mudança submetida aparecer no executável;

# Windows Vista

- ~1.7 pessoas testando para cada programador;
- Precisa ter compatibilidade com versões anteriores;
- Precisa ser instalado em “milhares” de configurações diferentes;

Imaginem os riscos!!!

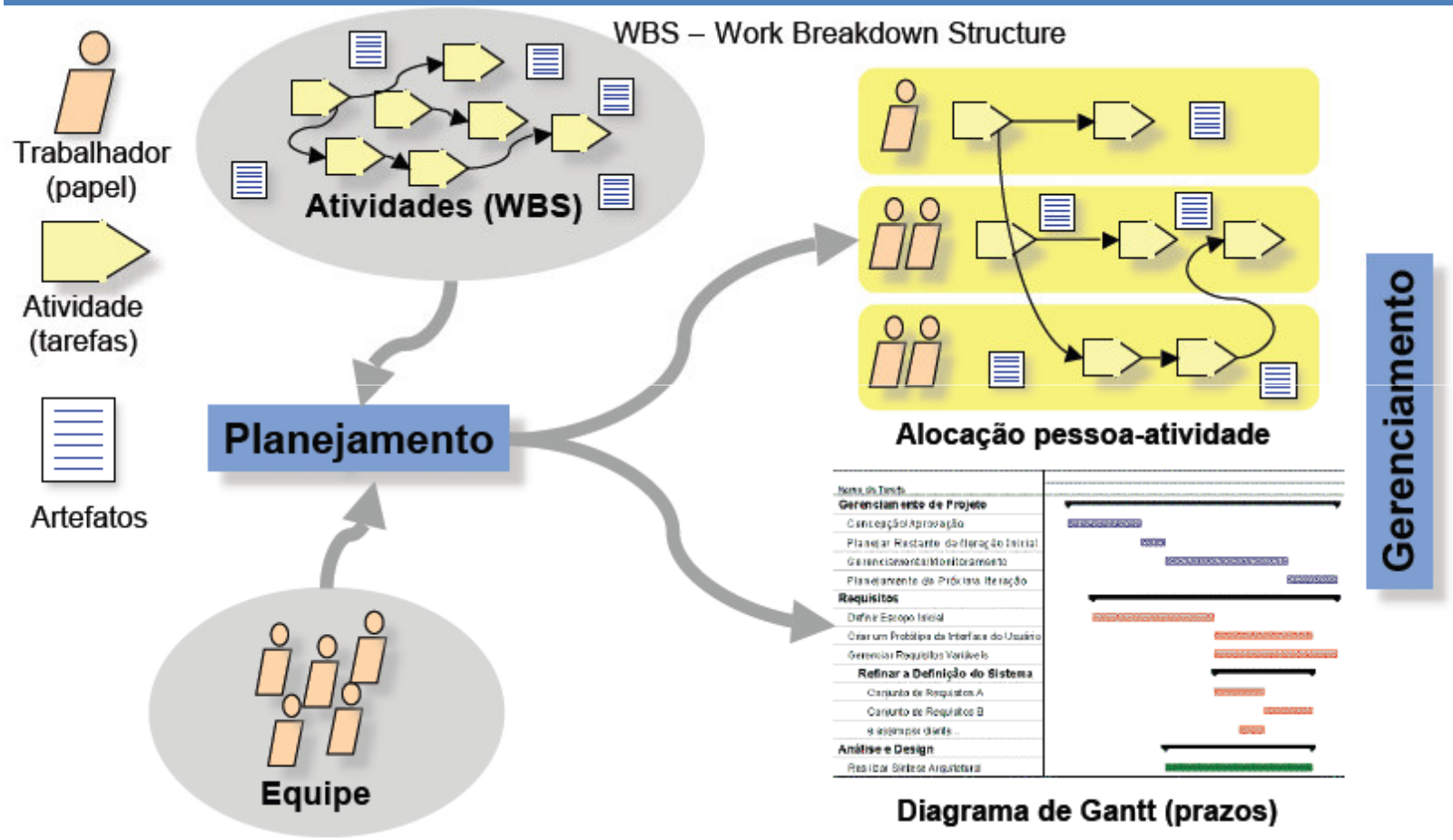
# Windows Vista

- Como organizar as atividades tal que 5000 pessoas possam trabalhar juntas ao mesmo tempo?
  - Processo de software;
  - Ferramentas de Gerência de Configuração;
- Como testar tanto código? E ainda para tantas plataformas diferentes?
- Como projetar um sistema com 50 milhões de linhas de código? Como garantir a integridade deste projeto?
  - Arquitetura / Modular / Projeto de software;

# O processo de software

- Um conjunto estruturado de atividades requeridas para desenvolver um sistema de software
  - Especificação
  - Projeto
  - Validação
  - Evolução
- Um modelo de processo de software é uma representação abstrata de um processo. Apresenta uma descrição de um processo de alguma perspectiva particular

# Visão geral de um Processo



# Modelos genéricos de processo de software

- O modelo cascata
  - Separa e distingue fases de especificação e desenvolvimento
- Desenvolvimento evolucionário
  - Especificação e desenvolvimento são entrelaçados
- Desenvolvimento baseado na reutilização
  - O sistema é montado a partir de componentes existentes
- Existem n variantes destes modelos!

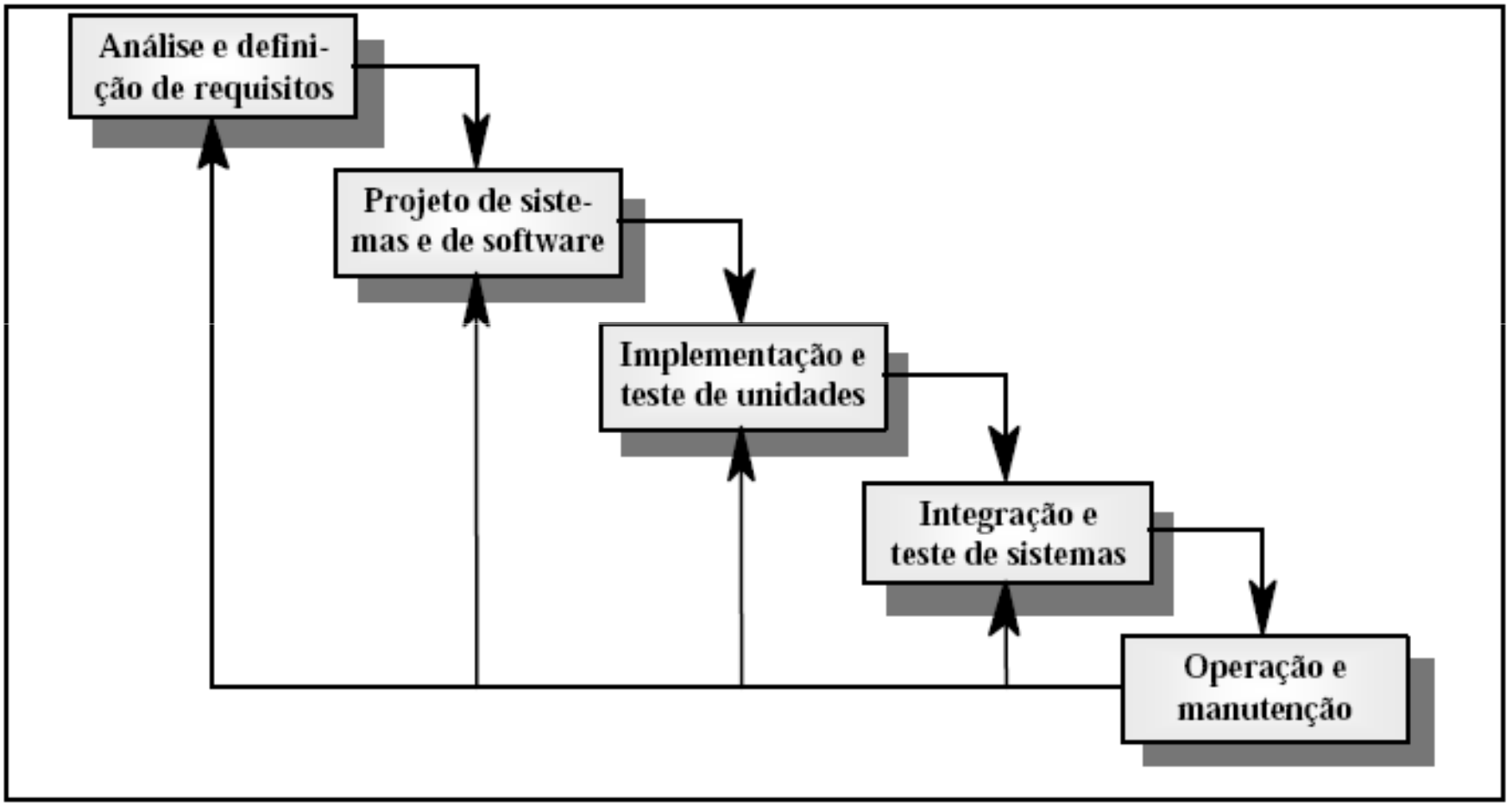
# Classificação das metodologias

- Pesadas ou grande, burocrática, densa ou de cerimônia
  - Usada em organizações que exigem comunicação formal
  - Onde projetos são longos, complexos e requisitos devem ser bem definidos
  - Projetos críticos que envolvam risco de vida
  - Projetos que suportam, normalmente, acima de 12 pessoas
  - Usadas para padronizar ações das pessoas
  - São: Cascata, Fast-tracking, incremental, RUP, OPEN, Catalysis.

# Classificação das metodologias

- Ágeis ou pequenas, leves ou adaptáveis
  - Usada por organizações que dão ênfase à colaboração
  - Abordagem flexível
  - Usada para projetos que tem requisitos que mudam muito (mercado, necessidades organização, leis, propósito do projeto, troca de chefia)
  - Comunicação muito próxima com o cliente
  - Exige menos pessoas, mas muito mais capacitadas, afinal, participarão de quase todas as fases do projeto.
  - São: Iterativo, Espiral, XP, Scrum, Crystal, etc

# Modelo Cascata



# Fases do modelo cascata

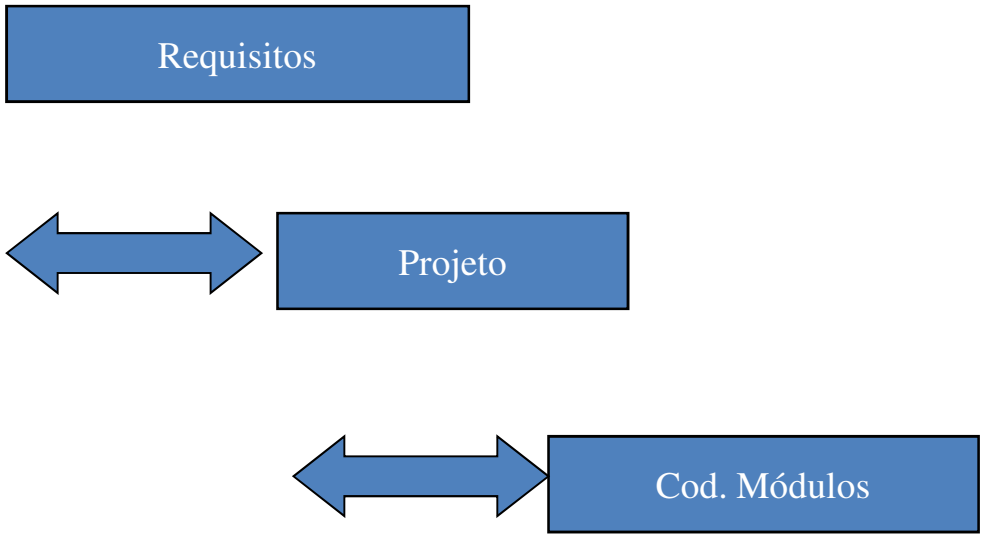
- Processo Linear
  - Análise e definição de requisitos (**Estudo de Viabilidade**)
  - Projeto do sistema e do software
  - Implementação e teste da unidade
  - Integração e teste do sistema
  - Operação e manutenção
- A desvantagem do modelo cascata é a dificuldade de acomodar mudanças depois que o processo está em andamento
  - Uma fase tem de estar completa antes de passar para a próxima

# Problemas do modelo cascata

- Processo não força antecipação das mudanças
- Processo baseado na produção de documentos (burocrático)
- Dificuldade de responder a mudanças de requisitos de clientes. Poucos sistemas de negócio tem requisitos estáveis.
- Assim, o modelo somente é apropriado quando os requisitos estiverem bem consolidados e entendidos.
- Dificuldade de estimar com pouca informação
- Usado para grandes projetos quando os requisitos são bem compreendidos e quando mudanças são limitadas durante o desenvolvimento

# Modelo Cascata – FAST TRACKING

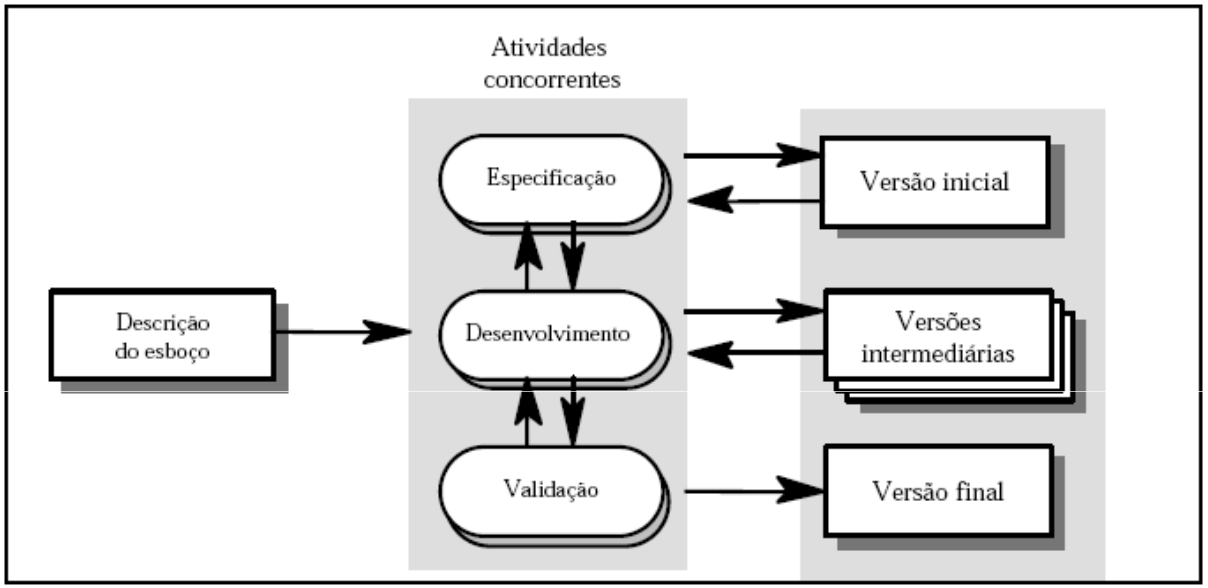
- Variante do modelo sequencial linear
- Fase inicia antes que a(s) precursora(s) tenha(m) terminado
- Aumenta risco de retrabalho

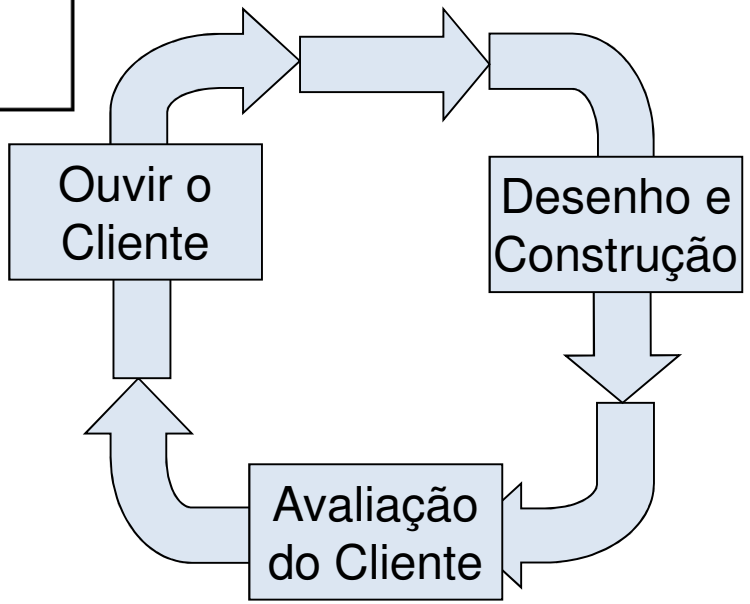


# Desenvolvimento evolucionário

- Desenvolvimento exploratório
  - O objetivo é trabalhar com clientes e evoluir o sistema final de um esboço de especificação inicial. Deve começar com os requisitos que estão bem entendidos
- Preparação de protótipos descartáveis
  - Objetivo é entender os requisitos do sistema. Deve começar com requisitos pobremente entendidos

# Desenvolvimento evolucionário



# Desenvolvimento evolucionário

- Aplicabilidade
  - Para sistemas interativos pequenos ou médios
  - Para partes de sistemas grandes (ex. a interface de usuário)
  - Para sistemas de curto-prazo
- Vantagens
  - Grande interação com o usuário
  - Qualidade da definição da interface

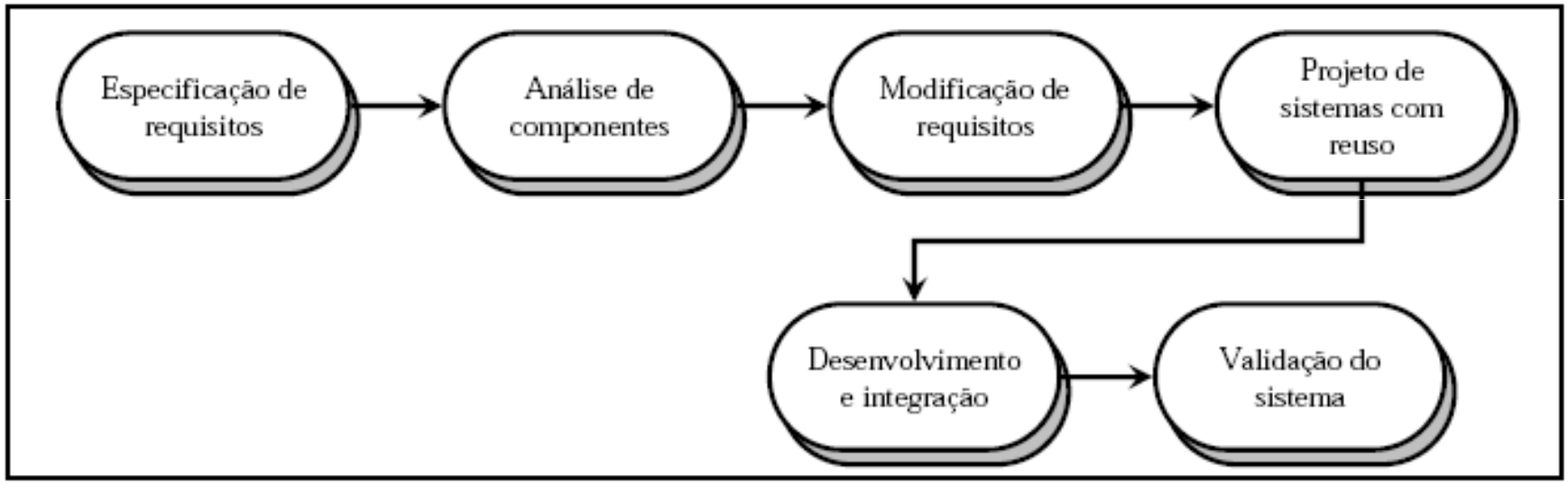
# Desenvolvimento evolucionário

- Problemas
  - Falta de visibilidade do processo
  - Sistemas são, em geral, pobremente estruturados
  - Habilidades especiais (ex. em línguas para rápida preparação de protótipos ) podem ser requeridas
  - Expectativa do usuário pois o protótipo não é o real.
  - O cliente vê o que parece ser uma versão executável do software, ignorando que o protótipo funciona de maneira precária
  - O desenvolvedor freqüentemente faz concessões na implementação a fim de conseguir rapidamente um protótipo executável

# Desenvolvimento orientado ao reuso

- Baseado no reuso sistemático, onde os sistemas são integrados de componentes existentes ou sistemas padronizados (ou COTS)
- Estágios do Processo
  - Análise do componente
  - Modificação dos requisitos
  - Projeto do sistema com reuso
  - Desenvolvimento e integração
- Esta abordagem está se tornando mais importante, mas a experiência ainda é limitada com ela

# Desenvolvimento orientado ao reuso



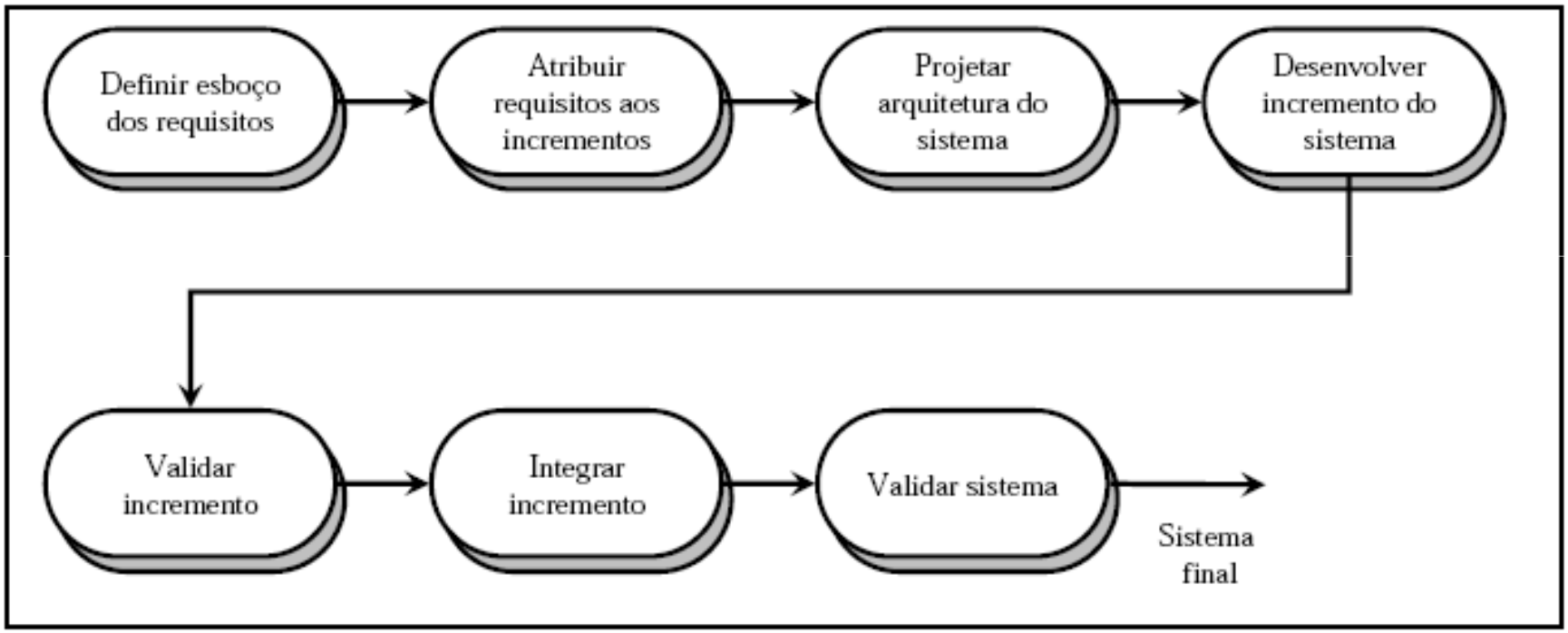
# Iteração do Processo

- Requisitos do sistema SEMPRE evoluem no decorrer de um projeto, então a iteração do processo, onde estágios anteriores são re-trabalhados, é sempre parte de um processo para sistemas maiores
- Iteração pode ser aplicada para qualquer modelo de processo genérico
- Duas abordagens (relacionadas)
  - Desenvolvimento incremental
  - Desenvolvimento espiral

# Desenvolvimento incremental

- Ao invés de entregar o sistema de uma única vez, o desenvolvimento e a entrega é dividida em incrementos com cada incremento entregando parte da funcionalidade requerida
- Os requisitos dos usuários são priorizados e os requisitos de maior prioridade são incluídos em incrementos iniciais
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados embora requisitos para incrementos posteriores possam continuar a evoluir

# Desenvolvimento incremental



# Vantagens do desenvolvimento incremental

- O valor agregado ao Cliente está na entrega em cada incremento de modo que a funcionalidade do sistema estará disponível mais cedo
- Incrementos iniciais funcionam como protótipos para ajudar a evocar requisitos para incrementos posteriores
- Menores riscos de falha no projeto em geral
- Os serviços do sistema de alta prioridade tendem a receber a maioria dos testes

# Desvantagens do desenvolvimento incremental

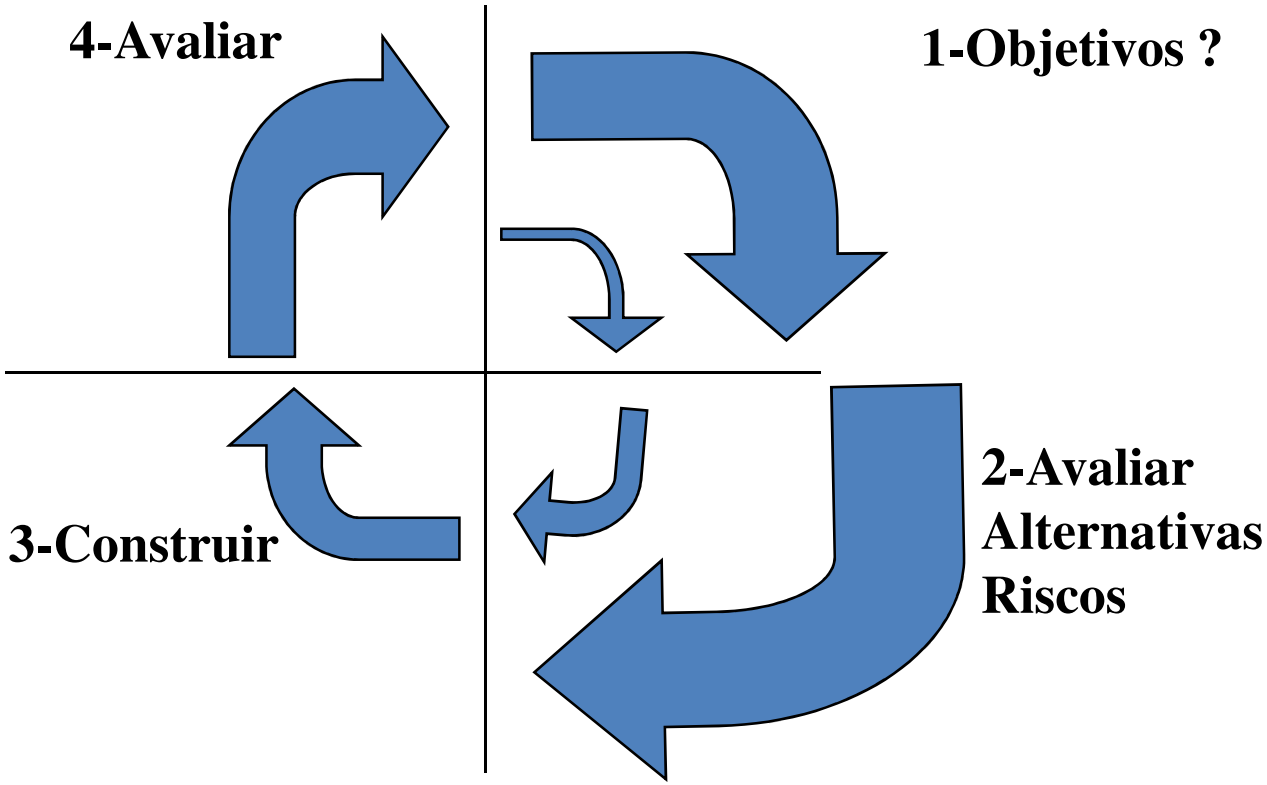
- Nem todos os tipos de aplicação são apropriados para o RAD. Se o sistema não puder ser adequadamente modularizado, a construção e seleção de componentes será problemática

- Não é adequado quando são enfrentados riscos técnicos elevados. Por exemplo, adoção profunda de uma nova tecnologia

# Desenvolvimento espiral

- Processo é representado como uma espiral ao invés de uma seqüência de atividades com retorno
- Cada volta na espiral representa uma fase no processo.
- Não existem fases fixas como especificação ou projeto – as voltas na espiral são escolhidas de acordo com o que é requerido
- Os riscos são explicitamente cotados e resolvidos durante todo o processo

# Modelo espiral do processo de software



# Setores do modelo espiral

- Estabelecimento de objetivos
  - Objetivos específicos para a fase são identificados
- Avaliação e redução de riscos
  - Os riscos são avaliados e atividades postas em prática para reduzir os riscos principias
- Desenvolvimento e validação
  - Um modelo de desenvolvimento para o sistema é escolhido, podendo ser qualquer um dos modelos genéricos
- Planejamento e avaliação do cliente
  - O projeto é revisado e a fase seguinte da espiral é planejada

# Comentários sobre o Ciclo de Vida em Espiral

- É uma abordagem realística para o desenvolvimento de software em grande escala.
- Usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva.
- Pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável.
- Exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso
- A cada iteração ao redor da espiral, versões progressivamente mais completas do software são construídas
- O modelo é relativamente novo e não tem sido amplamente usado

# Características do Modelo em Espiral

- Desvantagens
  - Bem aplicado somente a sistemas de larga escala
  - Sistemas devem ser produtos internos da empresa
- Vantagens
  - Fácil de decidir o quanto testar
  - Não faz distinção entre desenvolvimento e manutenção

# Extreme programming

- Nova abordagem para o desenvolvimento de software baseado no desenvolvimento e entrega de incrementos de funcionalidade bem pequenos
- Conta com melhoramento constante do código, envolvimento do usuário no time de desenvolvimento e programação em pares
- Abordado mais adiante

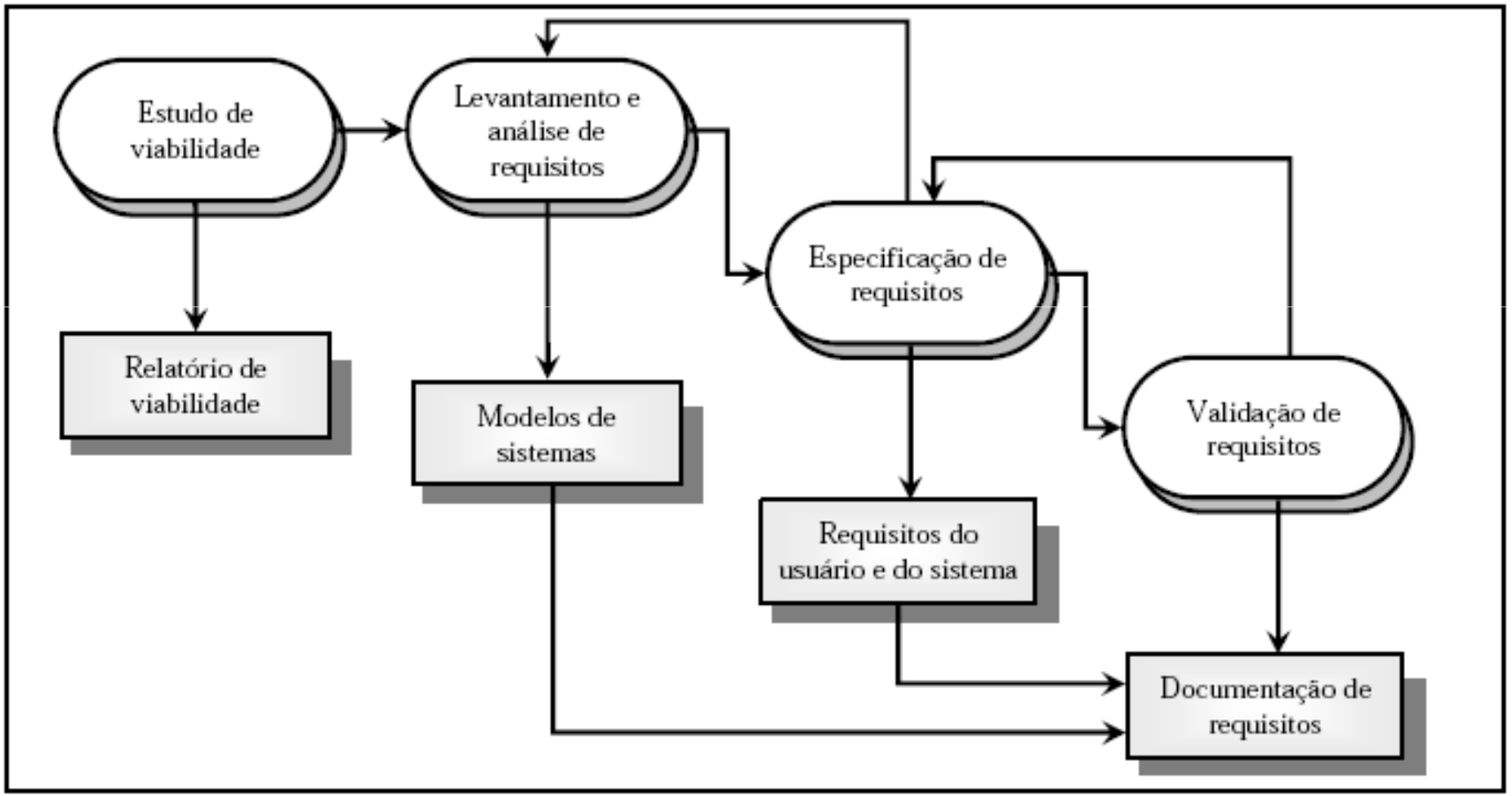
# Atividades de Processo

- Especificação de Software
- Projeto e implementação de software
- Validação de software
- Evolução de software

# Especificação do Software

- O processo de estabelecer que serviços são requisitados e quais as restrições na operação e desenvolvimento do sistema
- Processo de engenharia de requisitos
  - Estudo de viabilidade
  - Elicitação e análise dos requisitos
  - Especificação dos requisitos
  - Validação dos requisitos

# O processo de engenharia de requisitos



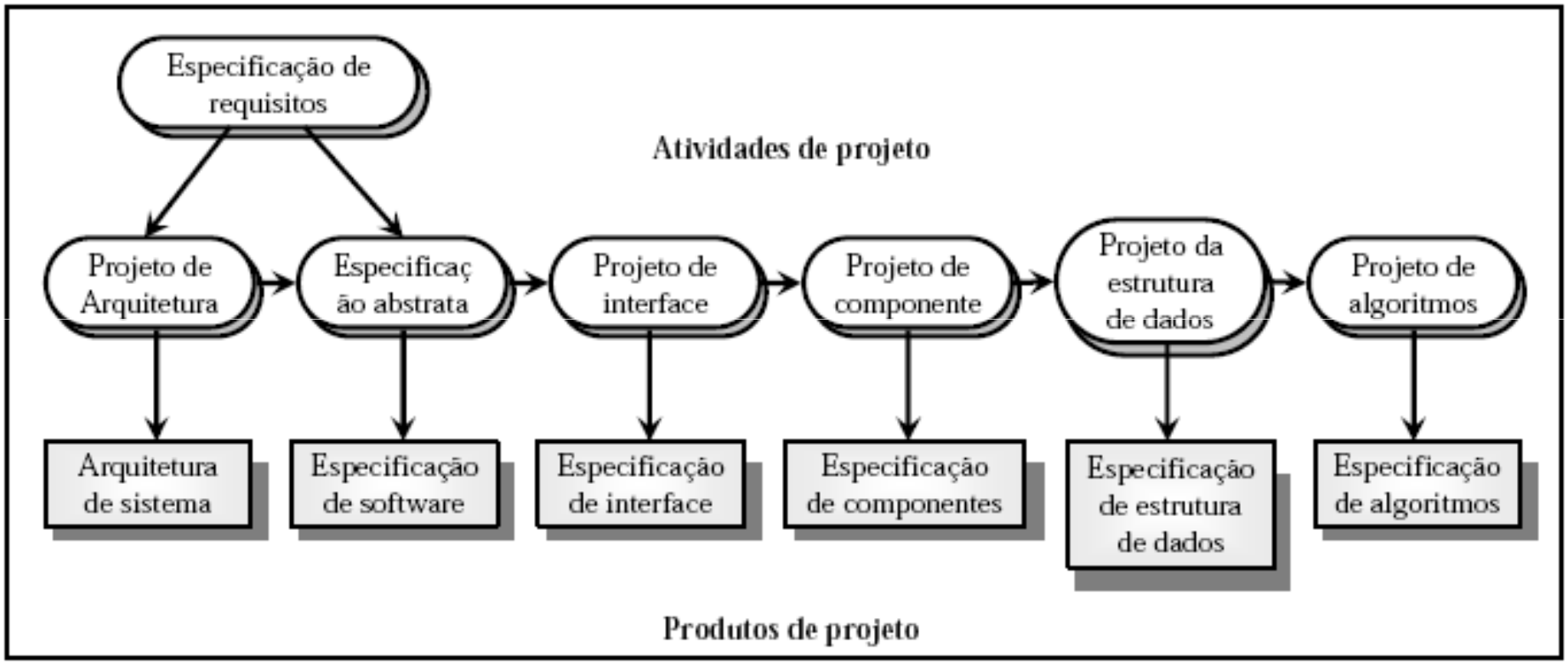
# Projeto e implementação de Software

- O processo de converter a especificação do sistema em um sistema executável
- Projeto de Software
  - Projeto de uma estrutura de software que perceba a especificação
- Implementação
  - Transformar esta estrutura em um programa executável
- As atividades de projeto e implementação são intimamente relacionadas e podem ser entrelaçadas

# Atividades de processo de projeto

- Projeto arquitetural
- Especificação abstrata
- Projeto de interface
- Projeto de componente
- Projeto de estrutura de dados
- Projeto de algoritmo

# O processo do projeto de software



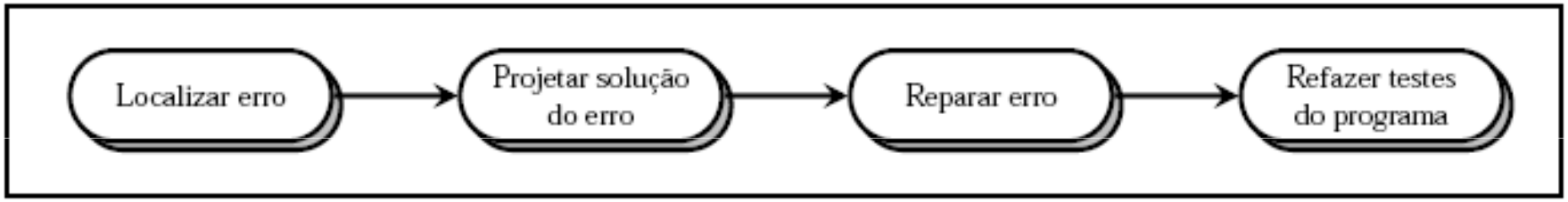
# Métodos do Projeto

- Abordagens sistemáticas para desenvolver um projeto de software
- O projeto é geralmente documentado como uma série de modelos gráficos
- Modelos possíveis
  - Object model;
  - Sequence model;
  - State transition model;
  - Structural model;
  - Data-flow model.

# Programando e Depurando

- Transformar um projeto em um programa e remover erros do programa
- Programação é uma atividade pessoal – não existe processo de programação genérico
- Programadores realizam alguns testes de programa para detectar falhas no programa e remover tais falhas no processo de depuração

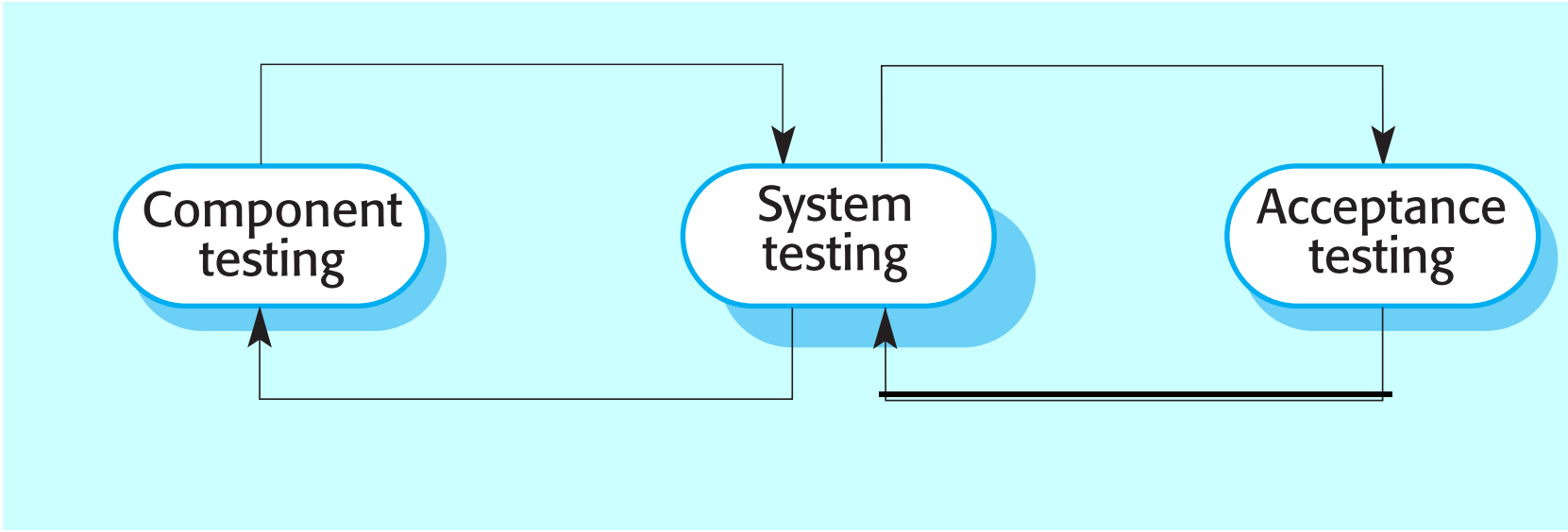
# O processo de depuração / debugging



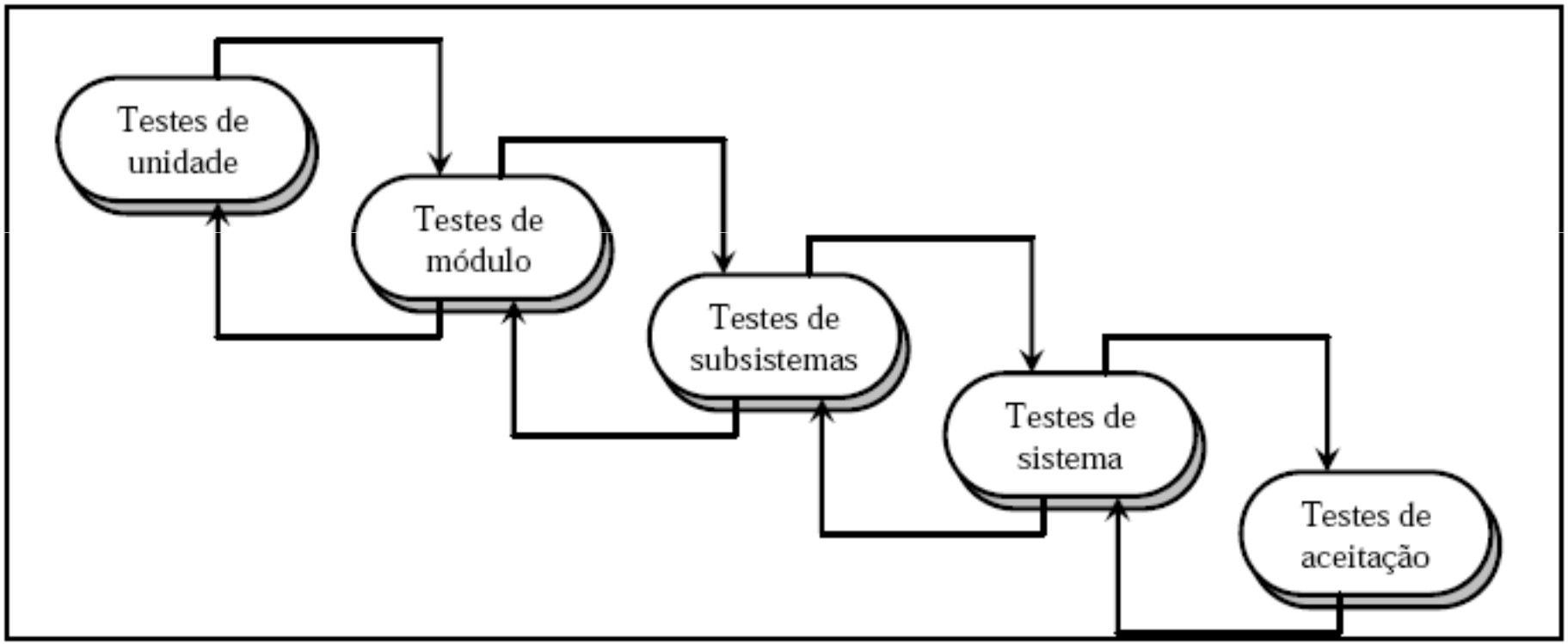
# Validação do Software

- Verificação e validação (V & V) pretendem mostrar que um sistema está de acordo com sua especificação e cumpre os requisitos do cliente do sistema
- Envolve a verificação e a revisão de processos e teste do sistema
- Teste de sistema envolve a execução do sistema com cases de teste que são derivados da especificação dos dados reais a serem processados pelo sistema

# The testing process



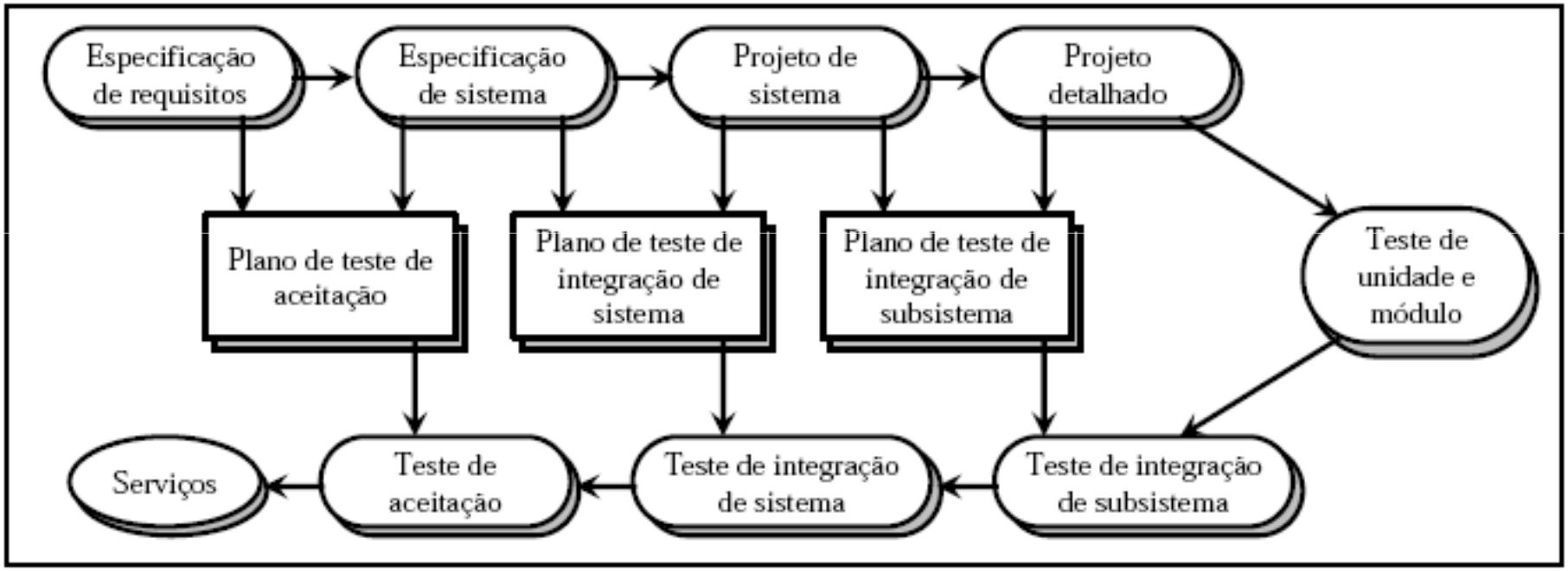
# The testing process



# Testing stages

- Teste da Unidade
  - Os componentes individuais são testados
- Teste do Módulo
  - Conjuntos de componentes dependentes relacionados são testados
- Teste do Sub-sistema
  - Os módulos são integrados em sub-sistemas e testados. O foco aqui deve ser no teste da interface
- Teste do Sistema
  - Teste do sistema como um todo. Teste das propriedades emergentes
- Teste de Aceitação
  - Teste com dados do consumidor para verificar que é aceitável

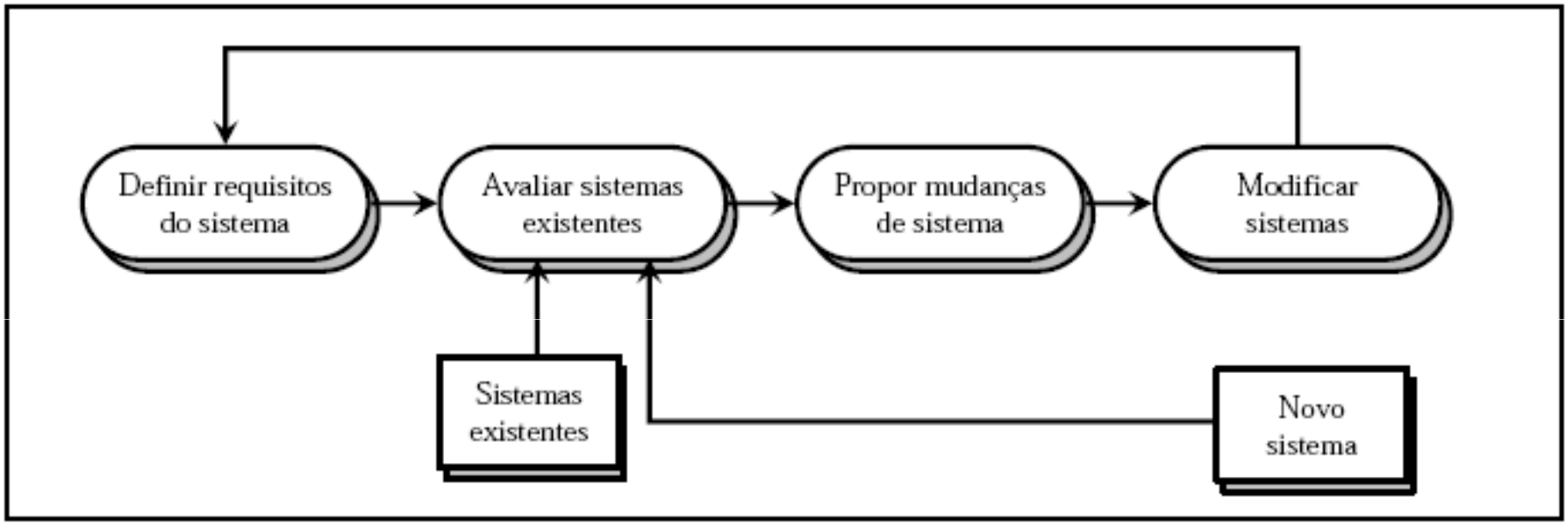
# Testing phases



# Evolução do Software

- Software é hereditariamente flexível e pode ser mudado.
- Como os requisitos mudam ao se alterar as circunstâncias de negócios, o software que suporta o negócio também deve evoluir e mudar
- Embora tenha havido uma demarcação entre desenvolvimento e evolução (manutenção), este é cada vez mais irrelevante na medida que menos e menos sistemas são totalmente novos

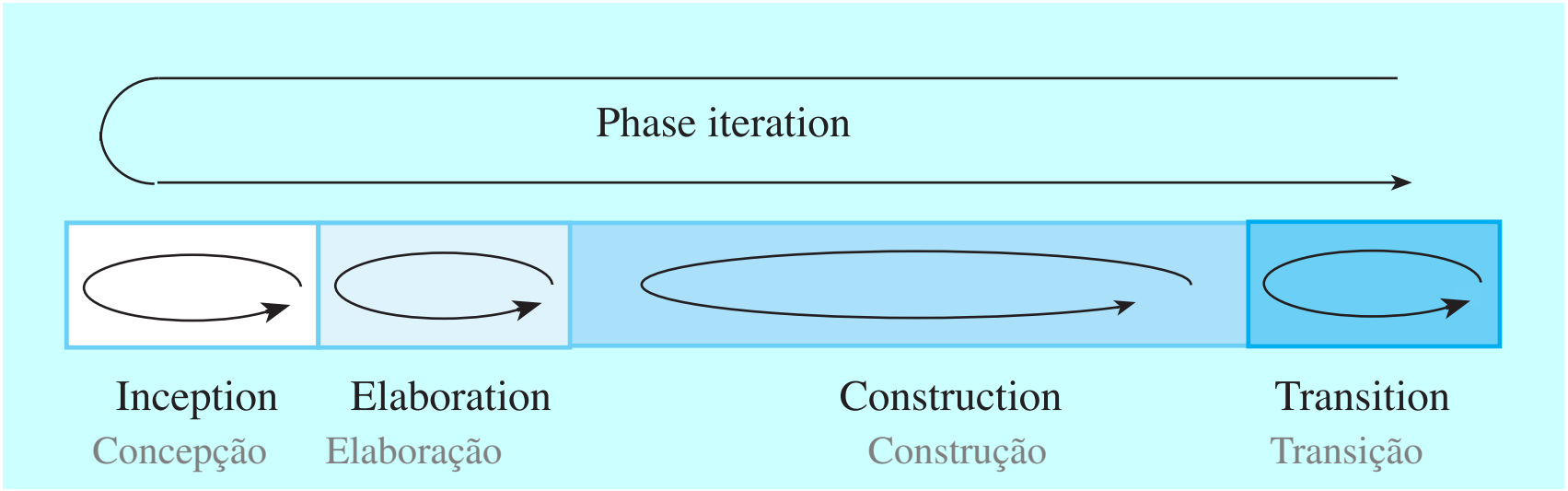
# Evolução do Software



# The Rational Unified Process

- Um modelo de processo moderno derivado do trabalho de UML e do Processo Unificado (PU) de Desenvolvimento de Software.
- Normalmente descreve 3 perspectivas
  - Uma perspectiva dinâmica que mostra as fases do modelo ao longo do tempo
  - Uma perspectiva estática que mostra as atividades realizadas no processo
  - Uma perspectiva prática, que sugere boas práticas a serem usadas durante o processo

# RUP phase model



# RUP phases

- Concepção
  - Estabelece o business case para o sistema. Identificar entidades externas que irão interagir e depois definir as interações.
- Elaboração
  - Desenvolver um entendimento sobre o domínio do problema e a arquitetura do sistema.
- Construção
  - System design, programming and testing.
- Transição
  - Deploy the system in its operating environment.

# RUP good practice

- Desenvolver software iterativamente
- Gerenciar requisitos
- Usar arquiteturas baseadas em componentes
- Modelar o software visualmente
- Verificar a qualidade do software
- Controlar as mudanças do software

# Static workflows

| Workflow | Description |
|---|---|
| Business modelling | Os processos de negócios são modelados usando casos de uso de negócios |
| Requisitos | Os agentes que interagem com o sistema são identificados e os casos de uso são desenvolvidos para modelar os requisitos do sistema |
| Análise e projeto | Um modelo de projeto é criado e documentado usando modelos de arquitetura, modelo de componente, modelo de objeto e de sequência |
| Implementation | Os componentes de sistema são implementados e estruturados em subsistemas de implementação. A geração automatic de código com base nos modelos de projeto ajuda a acelerar esse processo |
| Test | O teste é um processo iterative realizado em conjunto com a implementação. O teste de sistema segue o término da implementação |
| Deployment | Uma versão do produto é criada, distribuída aos usuários e instalada no local de trabalho |
| Configuration and change management | Este workflow de apoio gerencia as mudanças do sistema |
| Project management | Este workflow de apoio gerencia o desenvolvimento do sistema |
| Environment / Ambiente | Relacionado à disponibilização de ferramentas apropriadas de software para a equipe de desenvolvimento |

# Computer-aided software engineering (CASE)

- Engenharia de software auxiliada por computador (CASE) é um software para dar suporte aos processos de desenvolvimento e evolução do software
- Automação da atividade
  - Editores gráficos para o desenvolvimento de modelos de sistema
  - Dicionário de dados para gerenciar entidades de projeto
  - Construtor Gráfico UI para a construção de interface para usuário
  - Depuradores para suportar detecção de falhas no sistema
  - Tradutores automáticos para gerar novas versões de um programa

# Case technology

- Tecnologia Case tem levado a melhorias significantes no processo de software embora não na ordem de magnitude de melhorias que foram antes previstos
  - A engenharia de software requer pensamento criativo – isto não é prontamente automatizável
  - A engenharia de software é uma atividade de grupo e, para grandes projetos, muito tempo é utilizado em interações do grupo. A tecnologia CASE não os suporta de fato

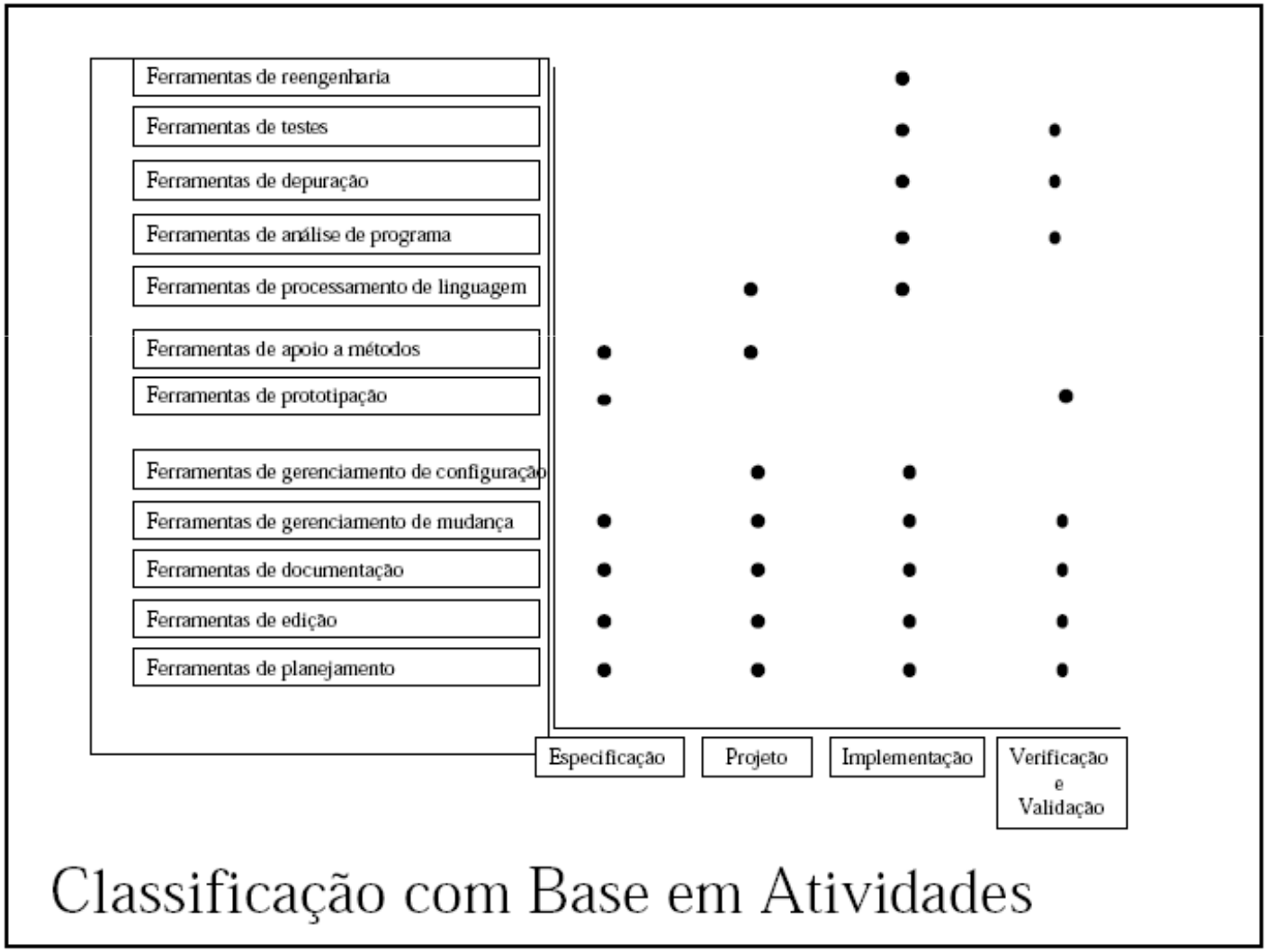
# CASE classification

- A classificação nos ajuda a entender os diferentes tipos de ferramentas de CASE e seu suporte para atividades do processo
- Perspectiva Funcional
  - As ferramentas são classificadas de acordo com suas funções específicas
- Perspectiva do Processo
  - As ferramentas são classificadas de acordo com as atividades do processo que suportam
- Perspectiva da Integração
  - As ferramentas são classificadas de acordo com sua organização em unidades integradas

# Classificação funcional de ferramentas CASE

| Ferramentas de | Exemplos |
|---|---|
| Planejamento | Ferramentas PERT, ferramentas de estimativa, planilha de cálculo |
| Edição | Editores de testos, editores de programas, processadores de texto |
| Gerenciamento de mudança | Ferramentas de controle de requisitos, sistemas de controle de mudanças |
| Gerenciamento de configuração | Sistemas de gerenciamento de versão, ferramentas de construção de sistemas |
| Prototipação | Linguagens de nível muito alto, geradores de interfaces com o usuário |
| Apoio a métodos | Editores de projeto, dicionário de dados, geradores de códigos |
| Processamento de linguagem | Compiladores, interpretadores |
| Análise de programa | Geradores de referência cruzada, analisadores estáticos, analisadores dinâmicos |
| Testes | Geradores de dados de testes, comparadores de arquivos |
| Depuração | Sistemas interativos de depuração |
| Documentação | Programas de layout de página, editores de imagem |
| Reengenharia | Sistemas de referência cruzada, sistemas de reestruturação de programas |

# Classificação baseada em atividades

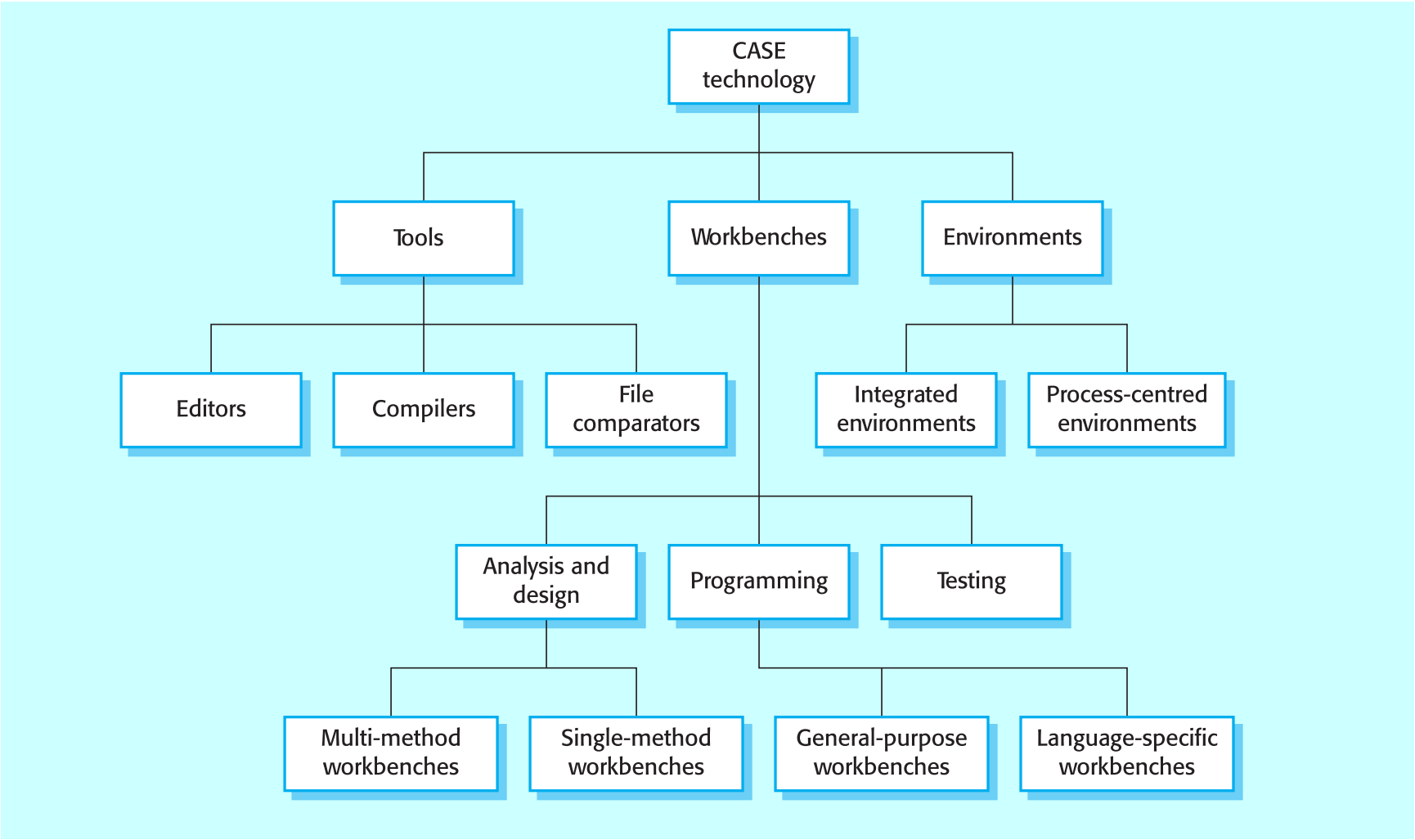



Classificação com Base em Atividades

# Perspectiva de Integração CASE

- Ferramentas
  - Suporta tarefas individuais do processo como verificação da consistência de um projeto, edição de texto, etc.
  - GASPRO, QSSrequireit, DOORS, SLATE, GENEXXUS
- Áreas de trabalho (workbenches)
  - Suporte a fases do processo como especificação ou projeto. Normalmente inclui uma variedade de ferramentas integradas
- Ambientes
  - Suporta tudo ou uma parte substancial de todo um processo de software. Normalmente inclui várias áreas de trabalho integradas

# Tools, workbenches, environments



# Pontos-chave

- Processos de software são as atividades envolvidas na produção e desenvolvimento de um sistema de software. Eles são representados num modelo de processo de software.
- Atividades comuns aos processos de software são especificação, projeto e implementação, validação e evolução.
- Modelos genéricos de processos descrevem a organização do processo de software.
- Modelos de processos iterativos descrevem o processo de software com um ciclo de atividades

# Pontos-chave

- Engenharia de requisitos é o processo de desenvolver uma especificação de software
- Os processos de projeto e implementação transformam a especificação em um programa executável
- A Validação envolve verificar que o sistema cumpre com as especificações e as necessidades do usuário
- Evolução se preocupa em modificar o sistema depois que ele está em uso
- Tecnologia CASE suporta atividades de processo de software

# Pesquisa – Capítulo 4

- Identifique todos os modelos de processos apresentados e construa um quadro comparativo
- Colocar todas as referências
- Individual na próxima aula
  - Método principal
  - Descrição resumida
  - Vantagens e desvantagens
  - Quais os softwares mais apropriados
  - Faça uma análise crítica de cada modelo
- Entregar no e-mail
- Assunto: FES_CAP4
- Arquivo: Nome_FES_CAP4

# Pesquisa - Apresentação

- 8 grupos – 15 minutos de apresentação
- 2 apresentadores (máximo) – 4 a 6 páginas formato da SBC
- Temas
  - Frameworks de desenvolvimento (.NET Framework, 2 frameworks em Java e mais um em outra linguagem)
  - Design Patterns
  - CMMI e MPSBR
  - Estimativas (COCOMO, COCOMO II) e Estimativas (APF, Análise por Caso de Uso)
  - RUP, Ferramentas CASE
  - Testes de Software
  - Estudo de Software House
  - Segurança em Software
- Entregar antes no e-mail
- Assunto: FES_AP1
- Arquivo: Nome_FES_AP1