Aula 3 – Processos de Software

- Um conjunto coerente de atividades para a especificação, *design*, implementação e testes de sistemas de software.

Baseado nos slides de:

Objetivos

- Introduzir os modelos de processos de software.
- Descrever um número diferente de modelos de processos e quando devem ser usados.
- Descrever e compreender, em linhas gerais, os modelos para engenharia de requisitos, desenvolvimento de software, realização de testes e evolução.
- Conhecer a tecnologia CASE para apoio ao processo de software.

Assuntos abordados

- Modelos de processo de software
- Interação de processo
- Especificação de software
- *Design* e implementação de software
- Validação de software
- Evolução de software
- Apoio ao processo automatizado

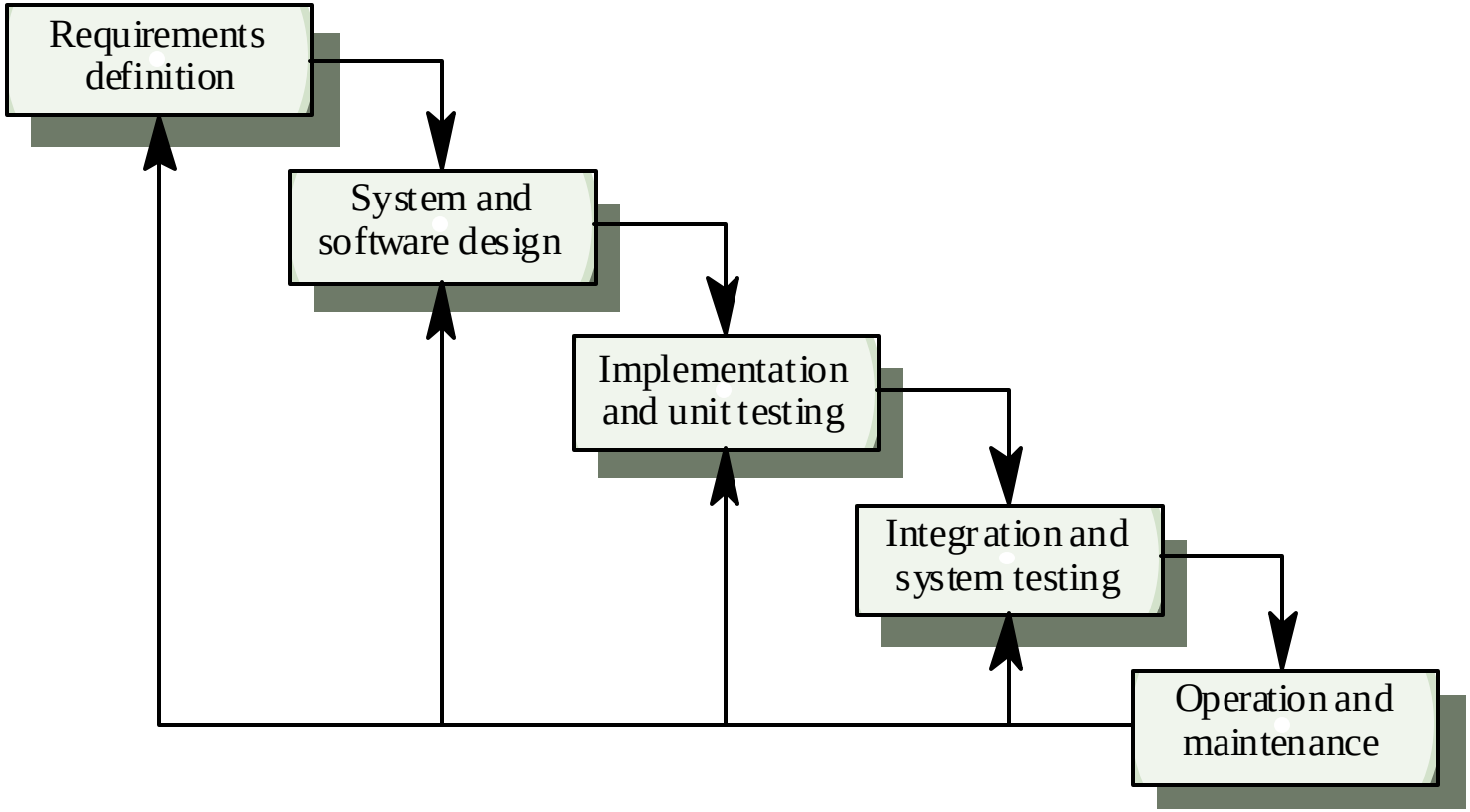
O processo de software

- Um conjunto estruturado de atividades necessárias para desenvolver um sistema de software. As principais atividades:
  - Especificação
  - *Design*
  - Validação
  - Evolução
- Um modelo de processo de software é uma representação abstrata de um processo. Ele apresenta uma descrição de um processo de um determinado ponto de vista.

Modelos genéricos de processos de software

- O modelo *Cascata* (“*Waterfall*”)
  - Consistem em fases separadas e distintas de especificação e desenvolvimento.
- Desenvolvimento Evolucionário
  - Especificação e desenvolvimento sobrepostos.
- Desenvolvimento Formal
  - Um modelo matemático é formalmente transformado em implementação.
- Desenvolvimento baseado em Reuso de Componentes
  - O sistema é concebido a partir de componentes existentes.

Modelo Cascata



Fases do Modelo Cascata

- Análise e definição de Requisitos
- *Design* de Sistemas e de Software
- Implementação e Testes de Unidades
- Integração e Testes de Sistemas
- Operação e Manutenção

Um dos problemas encontrados no modelo cascata é a dificuldade de acomodar alterações pedidas após a finalização da etapa de especificação.

Problemas do Modelo Cascata

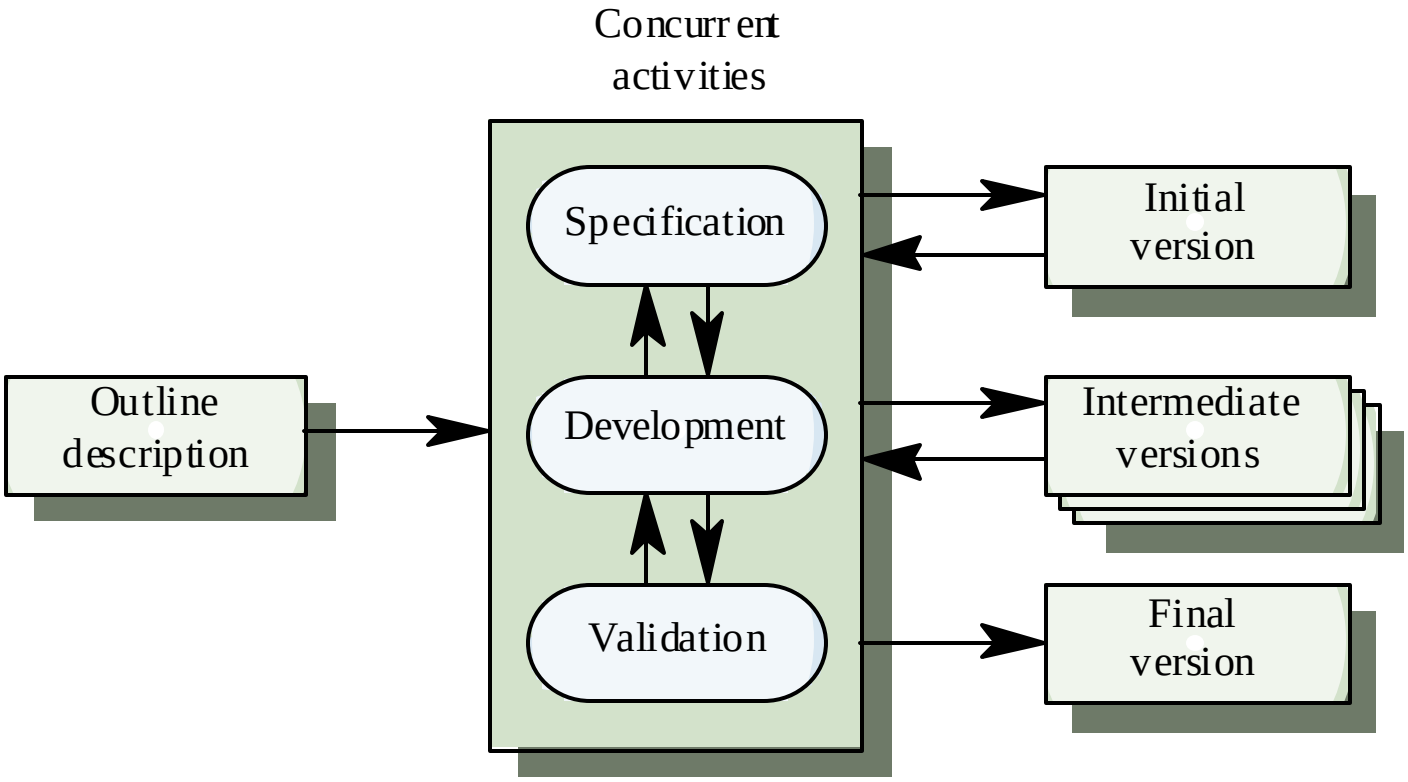
- Particionamento inflexível do projeto em fases distintas.
- Dificuldade em responder a alterações pedidas pelo cliente.
- Contudo, este modelo é o único apropriado quando os requisitos forem bem compreendidos.

Desenvolvimento Evolucionário

Dois tipos de desenvolvimento:

- Desenvolvimento exploratório
  - O objetivo é trabalhar com clientes e evolui-lo até um sistema final a partir de especificações genéricas. Deve iniciar a partir dos requisitos bem compreendidos.
- Protótipos descartáveis
  - O Objective é entender os requisitos do sistema. Deve iniciar com requisitos pouco compreendidos.

Desenvolvimento Evolucionário



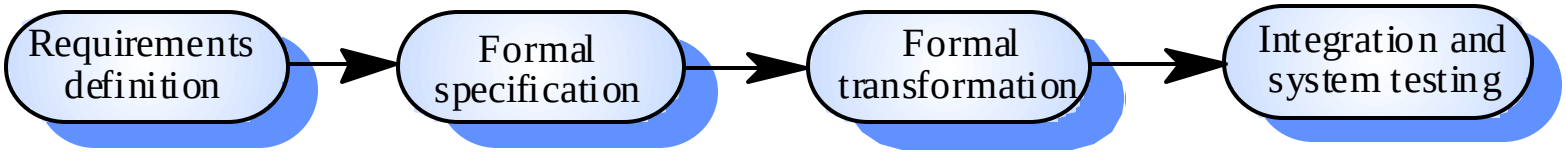
Desenvolvimento Evolucionário

- Problemas
  - Falta de visibilidade processo – Dificuldade de gerência,
  - Os sistemas são geralmente mal-estruturados – Dificuldade de manter o software produzido.
  - Ferramentas e técnicas especiais – Pode requerer pessoas com conhecimentos específicos ou linguagens diferenciadas.
- Aplicabilidade
  - Para sistemas interativos pequenos ou médios.
  - Para partes de grandes sistemas (exemplo: interface de usuários)
  - Para sistemas com ciclo de vida curto.

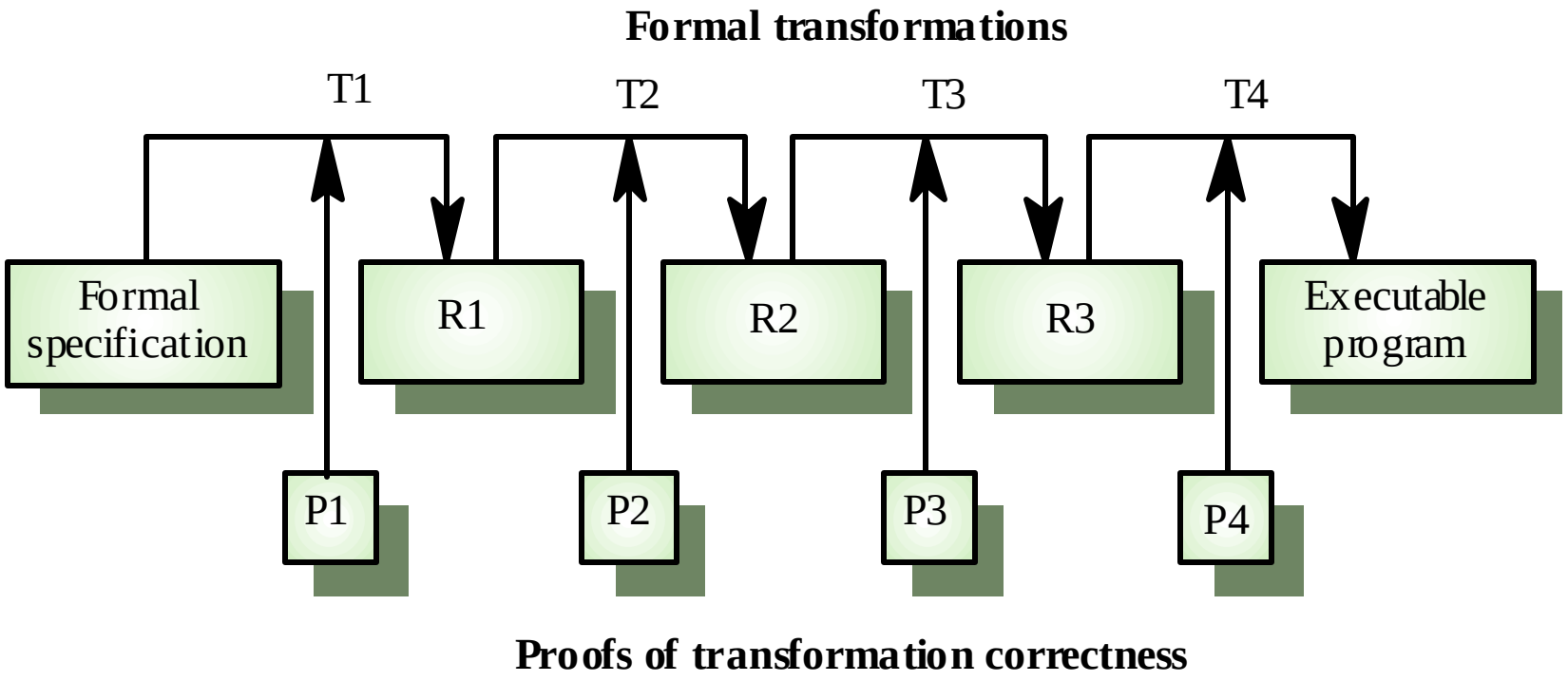
Desenvolvimento Formal de Sistemas

- Baseado em transformações de especificações matemáticas através de diferentes representações para um programa executável.
- Transformações são preservacionais, de forma que o programa será exatamente como sua especificação.
- O processo Cleanroom da IBM é o mais conhecido baseado em desenvolvimento formal.

Desenvolvimento Formal de Sistemas



Transformações Formais



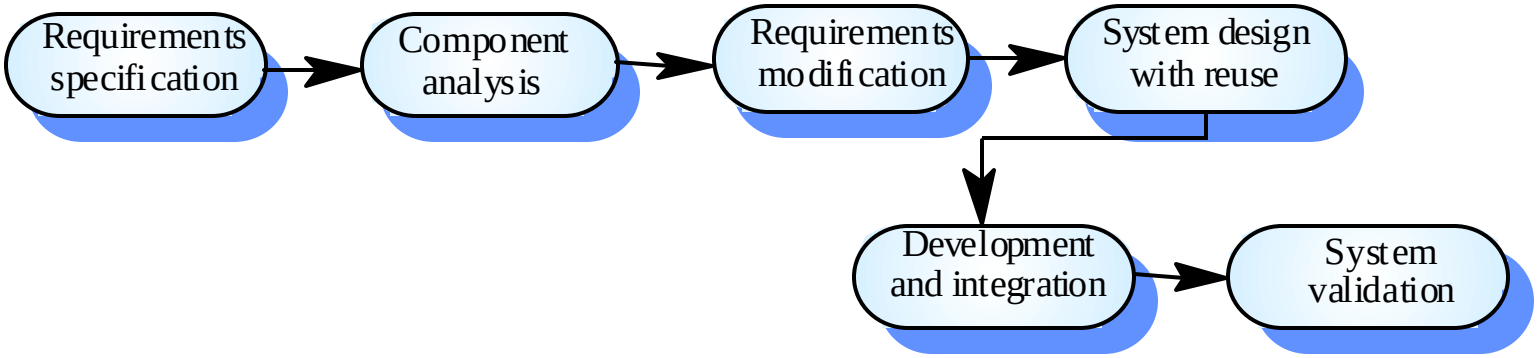
Desenvolvimento Formal de Sistemas

- Problemas
  - Necessita pessoas especialmente qualificadas e treinamento para aplicar a técnica.
  - Dificuldade para formalizar certos aspectos do sistema, como as interfaces de usuário.
- Aplicabilidade
  - Sistemas de missão-crítica que requerem segurança e confiabilidade extrema ao serem colocados em operação. Exemplo: sistemas de UTI.

Desenvolvimento orientado a Reuso

- Baseado no reuso sistemático onde os sistemas são integrados de componentes existentes ou de sistemas COTS (Commercial-off-the-shelf – Sistemas comerciais de prateleira)
- Estágios do processo
  - Análise de componentes
  - Modificação de requisitos
  - *Design* com reuso
  - Desenvolvimento e integração
- Esta abordagem tem crescido e se tornado importante, mas ainda precisa de mais dados e casos de sucesso.

Desenvolvimento orientado a Reuso



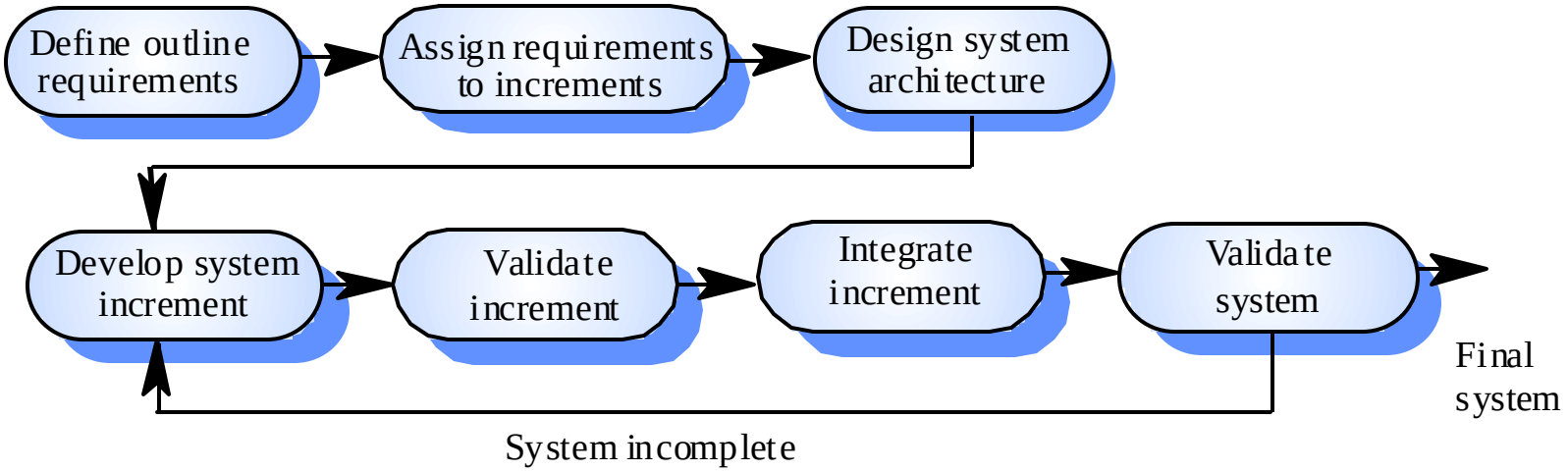
Iteração de processo

- Os requisitos de sistemas SEMPRE evoluem durante o projeto, de forma que a iteração de processo de estágios prévios sejam retrabalhados é uma atividade comum em processos de grandes sistemas.
- As iterações podem ser aplicadas em qualquer um dos modelos de processos genéricos.
- Duas aboradagens (modelos hibrídos):
  - Desenvolvimento Incremental
  - Desenvolvimento Espiral

Desenvolvimento Incremental

- Em vez de entregar um sistema em uma única etapa, o desevolvimento e a entrega e quebrado em incrementos, sendo que cada incremento contém parte dos requisitos e funcionalidades.
- Os requisitos do usuário são priorizados e todos os requisitos de alta prioridade são incluídos nos incrementos iniciais.
- Uma vez que o desenvolvimento de um incremento e inicializado, os requisitos desde são congelados enquanto os requisitos de incrementos futuros podem continuar a evoluir e mudar.

Desenvolvimento Incremental



Desenvolvimento Incremental

- Vantagens:
  - O cliente não precisa esperar até que o sistema esteja concluído para usar. As primeiras funcionalidades já podem ser usadas desde o primeiro incremento entregue.
  - Os primeiros incrementos atuam com protótipo e ajudam a levantar requisitos para incrementos futuros.
  - Risco menor de fracasso do projeto.
  - Os serviços prioritários do sistema tendem a receber a maior carga de testes.
- Problemas:
  - Os incrementos devem ser pequenos (20kloc). Dificuldade de alocar requisitos para os incrementos em quantidades adequadas.

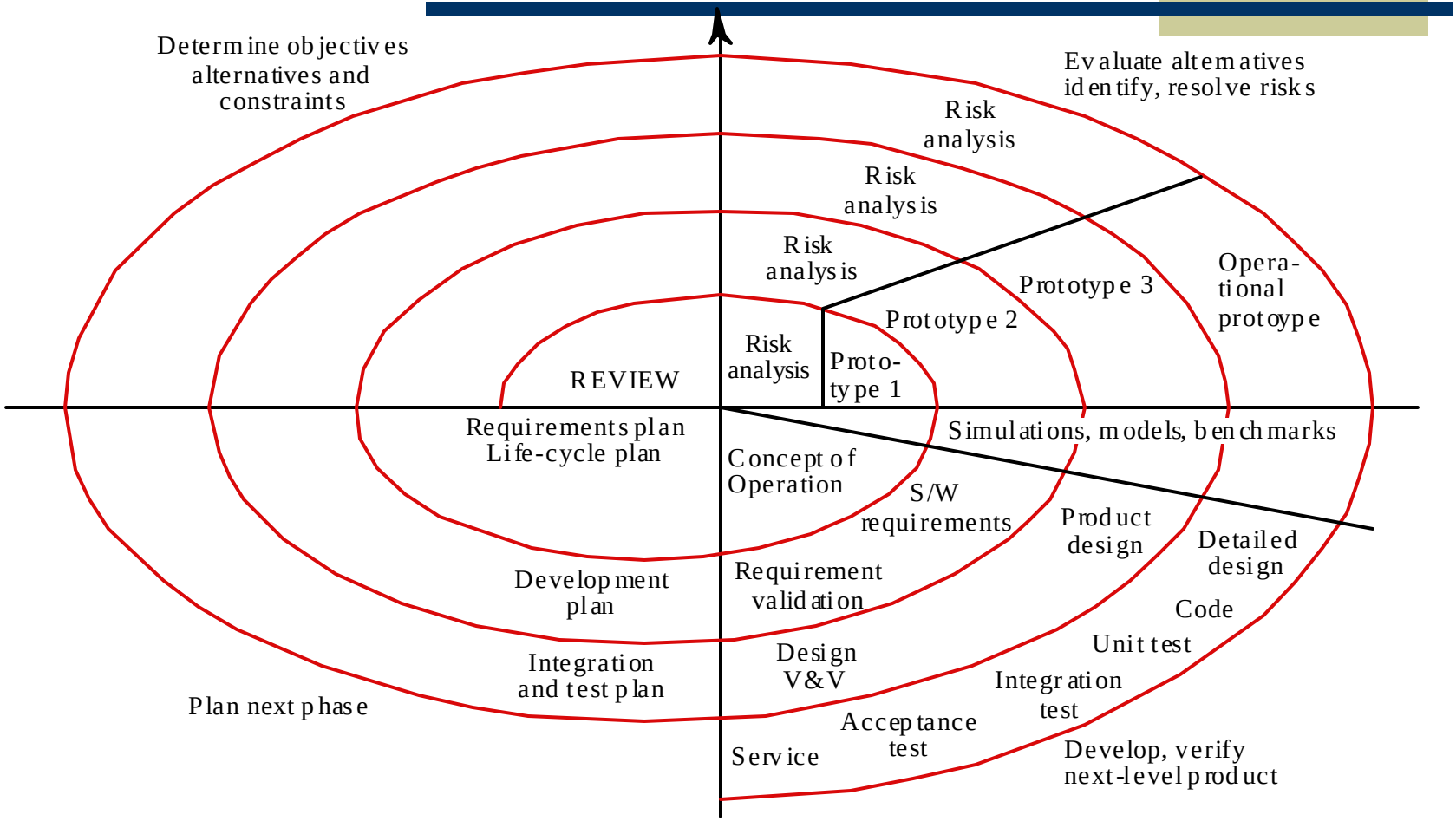
Extreme programming

- Nova abordagem para desenvolver baseada no desenvolvimento e entrega de pequenos incrementos de funcionalidades.
- Baseado na constante melhoria de código, envolvimento de usuários e time de desenvolvimento, programação impessoal.
- Muitas vezes desenvolvido “à quatro mãos”.
- A metodologia proposta por Beck (1999), apresenta relatórios de casos de sucesso dessa técnica, no entanto ainda é muito cedo para se afirmar se a mesma se consolidará ou não como um processo de desenvolvimento de software.

Desenvolvimento em Espiral

- O processo é representado como uma espiral em vez de uma sequência de atividades com retorno de uma atividade para outra.
- Cada loop na espiral representa uma fase no processo.
- Não há número fixo de fases, nem na especificação, nem no design. O número de loops na espiral dependerá das necessidades do projeto.
- Os riscos são levantados, avaliados e resolvidos através do processo.

Modelo espiral de desenvolvimento de software



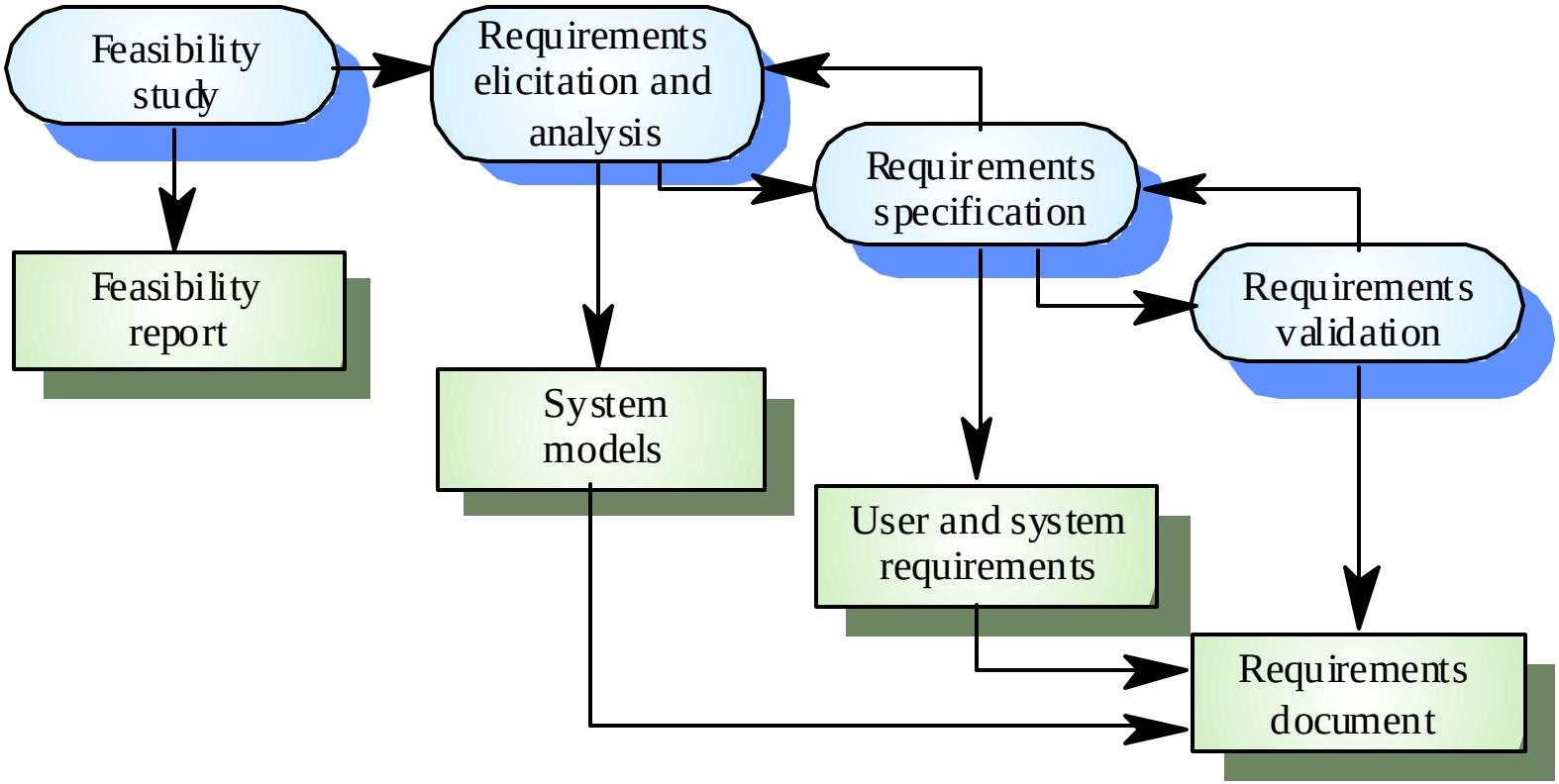
Setores do Modelo espiral

- Determinação dos objetivos
  - Especificar os objetivos para a fase são identificados.
- Avaliação de riscos e redução
  - Os riscos são avaliados e as atividades alocadas de forma a reduzir os riscos chave.
- Desenvolvimento e Validação
  - Um modelo de desenvolvimento para um sistema é escolhido. Entre os quais pode ser qualquer um dos modelos genéricos.
- Planejamento
  - O projeto é revisado e a próxima fase da espiral é planejado.

Especificação de Software

- O processo de estabelecer quais serviços são requeridos e quais as restrições sobre a operação e o desenvolvimento do sistema.
- Processo de Engenharia de Requisitos (prox. página):
  - Estudos de viabilidade
  - Levantamento e análise de requisitos
  - Especificação de requisitos
  - Validação de requisitos
- As atividades não precisam ser necessariamente seguidas em uma ordem rigorosa. Em geral, são intercaladas.

Processo de Engenharia de Requisitos



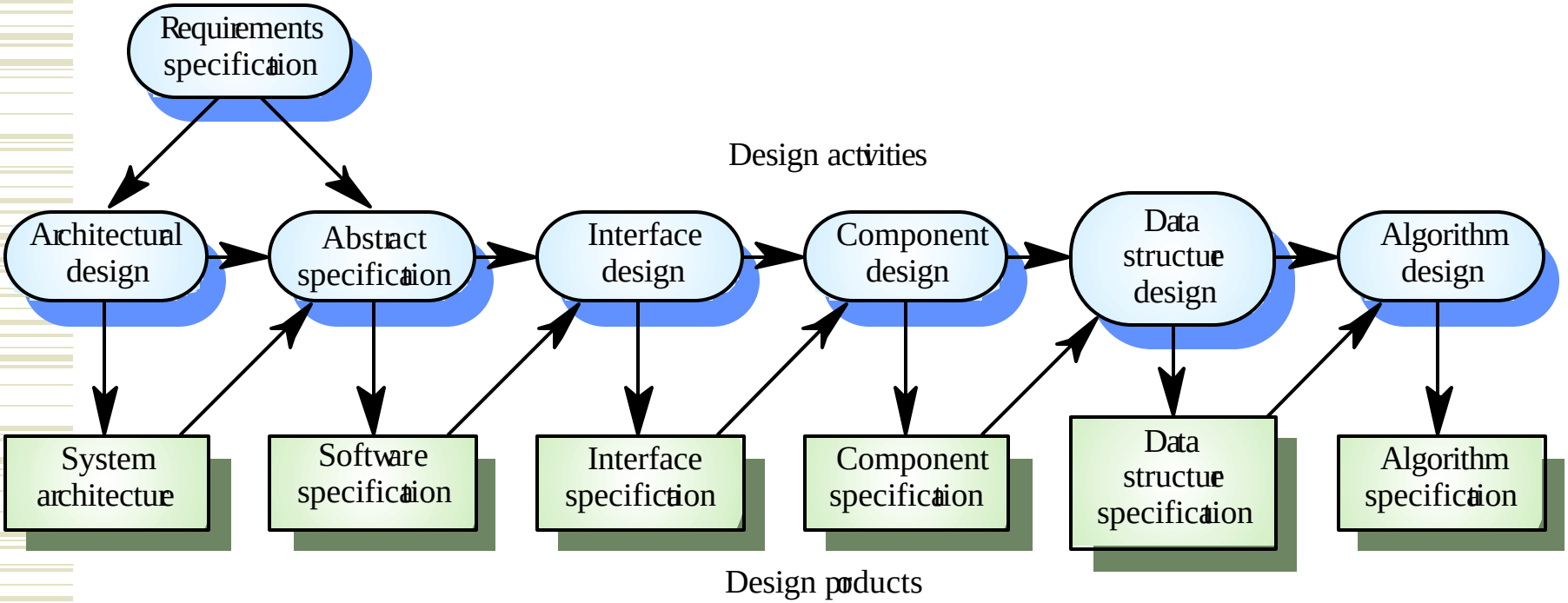
Design e implementação de Software

- É o processo de conversão da especificação do sistema em um sistema executável.
- *Design* do Software
  - *Projetar* uma estrutura de software que esteja de acordo com a especificação.
- Implementação
  - Traduzir a estrutura em um programa executável.
- As atividades de *design* e implementação estão muito próximas e podem inclusive estarem interligadas.

Atividades no Processo de *Design*

- *Design* arquitetônico
- Especificação abstrata
- *Design* de interfaces
- *Design* de componentes
- *Design* da estrutura de dados
- *Design* de Algoritmos

Processo de *Design* de Software



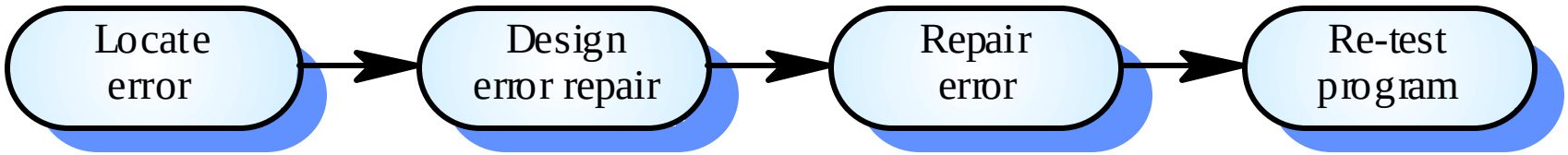
Métodos de *design*

- Várias abordagens sistemáticas são propostas para desenvolver um *design* de software.
- O *design* é usualmente documentado com um conjunto de modelos gráficos.
- Modelos possíveis:
  - Modelagem de fluxo de dados (DFD)
  - Modelagem de Entidade e Relacionamento (MER).
  - Modelagem estrutural (DeMarco)
  - Modelagem de Objetos (UML)

Codificação e depuração (debug)

- Consiste em traduzir o *design* em um programa e remover os erros do programa.
- Codificar é uma atividade pessoal - Não há um processo geral para realizar programação. Existem guias com “boas práticas de programação”.
- Os programadores realizam alguns testes no código para descobir falhas no programa e remove-las através de um processo de depuração (debug).
- Testes de software e depuração (debug) são atividades completamente distintas!!! O teste estabelece a existência de defeitos, enquanto a depuração se ocupa em localizar e corrigir esses defeitos.

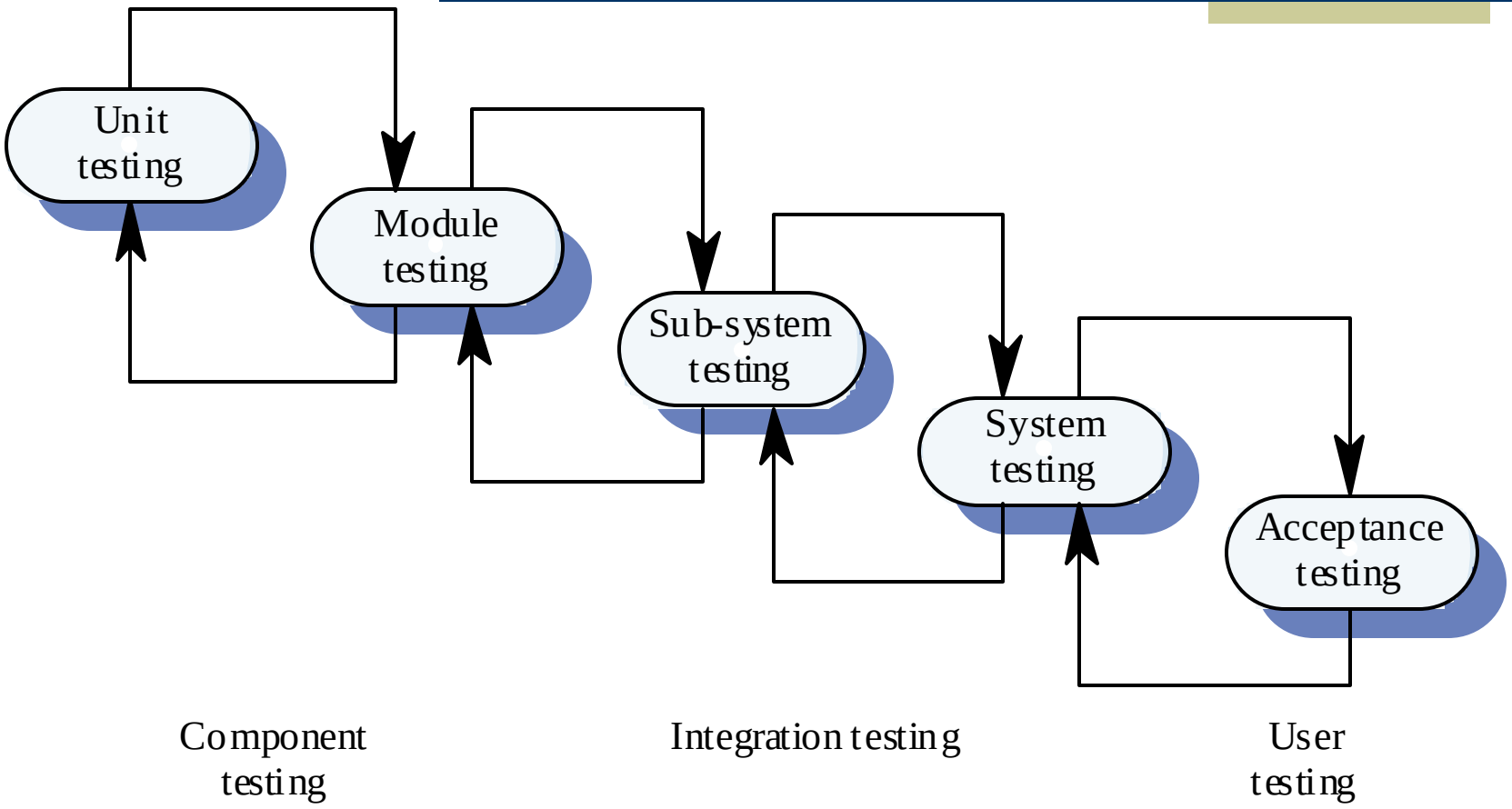
O processo de depuração (debug)



Validação de Software

- A verificação e validação tem como pretensão mostrar que um sistema está em conformidade com sua especificação e atende os requisitos do cliente do sistema.
- Envolve os processos checagem e revisão e testar o sistema.
- Testar os sistemas envolve executar o sistema como os casos de testes que são derivados da especificação de dados reais a serem processados pelo sistema.

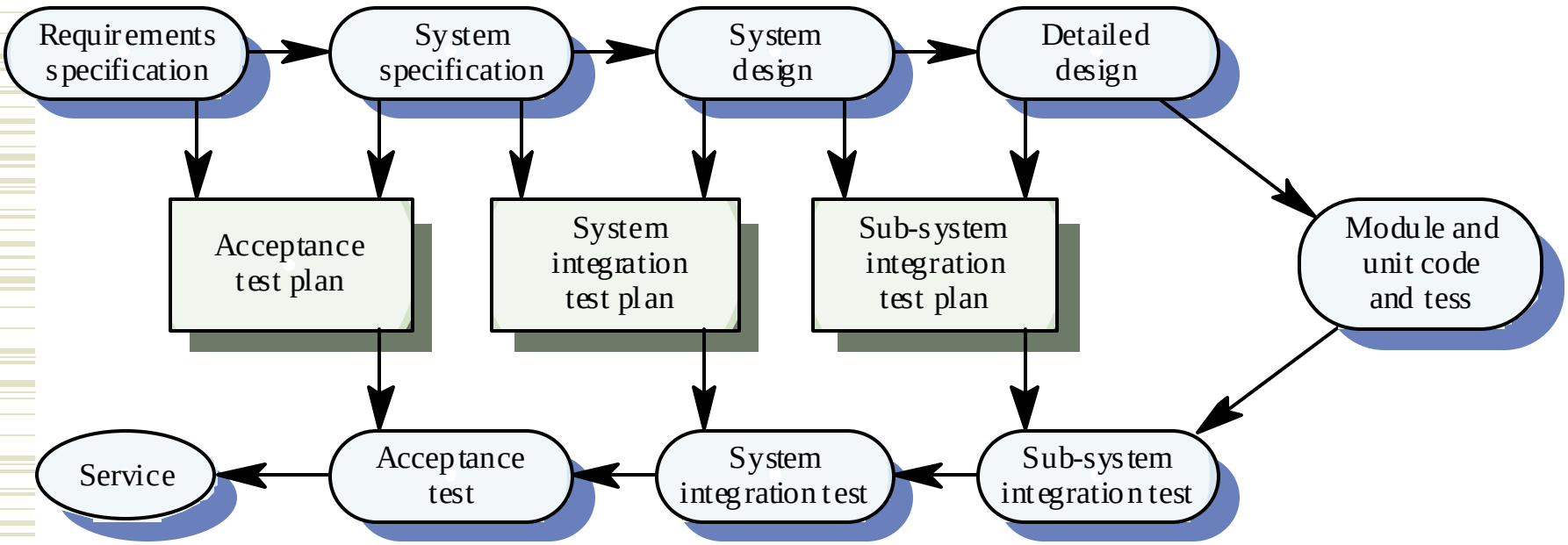
O Processo de Testes



Estágios da fase de testes

- Teste Unitário (ou Teste de Unidade)
  - Componentes individuais são testados
- Testes de Módulos
  - Coleções de componentes dependentes e relacionados são testados.
- Testes de Sub-Sistemas (ou Teste de Integração)
  - Módulos são integrados em sub-sistemas e testados. O foco aqui deve ser nos testes de interface.
- Testes de Sistema
  - Testar o sistema como um todo. Testar as propriedades emergentes.
- Testes de Aceitação
  - Testar com dados do cliente para checar se o sistema é aceitável. Em geral, é executado pelo cliente e consiste num subconjunto dos casos de testes de sistema.

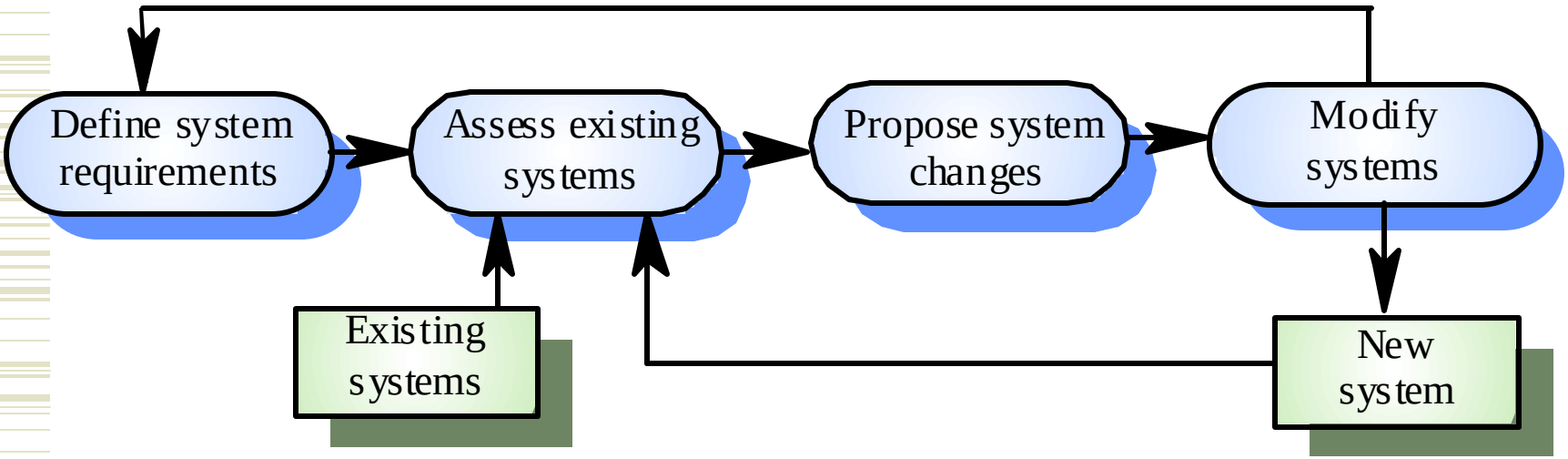
Etapas de Testes



Evolução de Software

- O software é inerentemente flexível e pode mudar.
- Os requisitos mudam de acordo com mudanças nas circunstâncias do negócio, sendo assim, o software que suporta esse negócio deve evoluir e mudar.
- Historicamente sempre houve uma demarcação entre desenvolvimento e evolução de software (manutenção), sendo que esta última sempre foi considerada uma parte menor e desinteressante, apesar de ser com freqüência responsável pela maior parte do custo de um software.

Evolução de software



Apoio ao processo automatizado (CASE)

- Computer-aided software engineering (CASE) é o software que apoia os processos de desenvolvimento e evolução de software.
- Atividades de automação:
  - Editores gráficso para desenvolvimento de modelos de sistemas.
  - Dicionários de dados para gerenciar entidades de *design*.
  - Geração de interfaces gráficas com usuários.
  - Depuradores (debugers) para auxiliar a localização de defeitos de software.
  - Tradutores automatizados para gerar novas versões de um programa em outras linguagens.

Tecnologia CASE

- As Tecnologias CASE tem gerado melhorias significantes no processo de software, mas são limitadas a alguns fatores:
  - Engenharia de software requer criatividade – Esta atividade não é suportada pela tecnologia CASE.
  - Engenharia de software é uma atividade em equipe, e para grandes projetos, muito tempo é gasto nas interações entre os times. As tecnologias CASE não suportam ainda esse tipo de atividade.

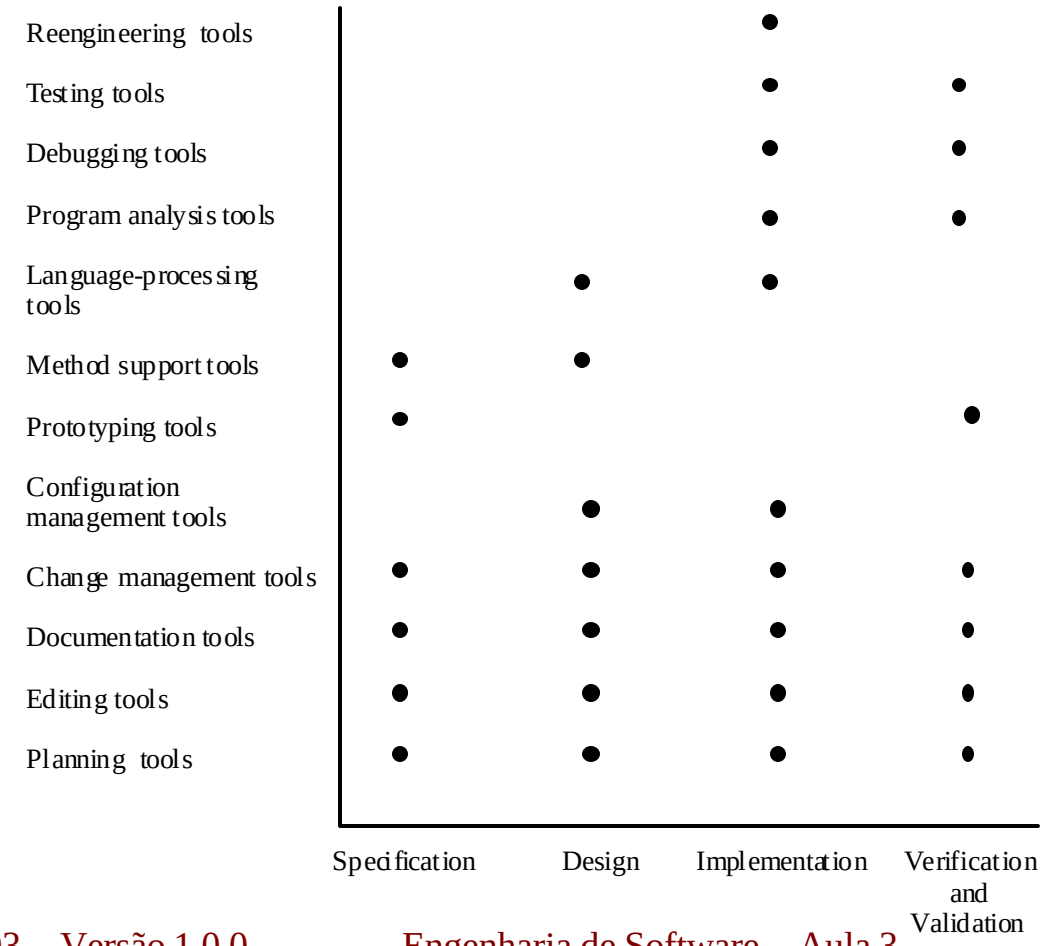
Classificação CASE

- Uma classificação ajuda a entender os diferentes tipos de ferramentas CASE e como podem apoiar os atividades do processo de software:
  - Perspectiva funcional
    - As ferramentas são classificadas de acordo com suas funções específicas.
  - Perspectiva de processo
    - As ferramentas são classificadas de acordo com que atividades de processo elas suportam.
  - Perspectiva de integração
    - As ferramentas são classificadas de acordo como são organizadas em unidades integradas.

Classificação funcional das ferramentas

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

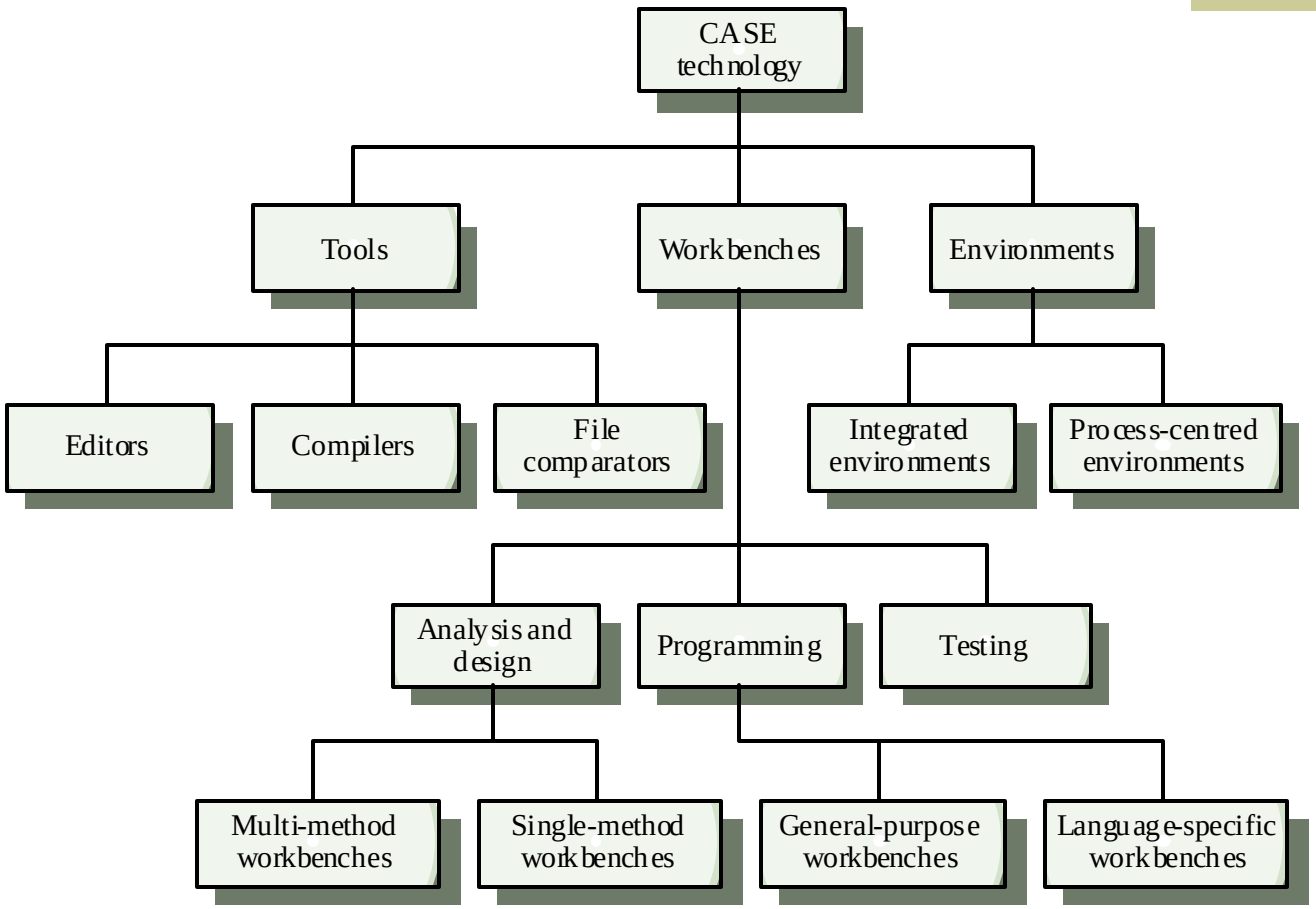
Classificação CASE por atividades



Integração CASE

- Ferramentas
  - Apoiar tarefas de processos individuais como checagem de consistência de design, edição de textos, etc…
- Workbenches
  - Apoiar uma fase do processo como especificação ou design. Normalmente incluir um número integrado de ferramentas.
- Ambiente
  - Apoia todas ou parte substancial de um processo de software inteiro. Normalmente inclui vários workbenches integrados.

Ferramentas, workbenches, e ambientes



Pontos-Chave

- Processos de software são as atividades envolvidas na produção de um sistema de software. Eles são representados nos modelos de processos de software.
- Atividades genéricas como especificação, *design*, implementação, validação e evolução fazem parte de todos os modelos de processo.
- Modelos de processos genéricos descrevem a organização dos processos de software.
- Modelos de processos iterativos descrevem o processo de softeware como ciclos de atividades.

Pontos-Chave

- A Engenharia de Requisitos é o processo de desenvolvimento de uma especificação de software.
- Os processos de *design* e implementação transformam a especificação em um programa executável.
- A validação é o processo de verificar ser o sistema está em conformidade com sua especificação e se atende às reais necessidades dos usuários.
- A evolução se ocupa com as modificações do sistema após ele entrar em uso.
- As tecnologias CASE apoiam uma ou mais atividades do processo de software, de acordo com sua classificação.

Conclusões

- Foi apresentada uma visão geral das atividades que compõem os processos de software.
- Embora não exista um processo de software “ideal”, exitem muitas oportunidades de trabalho para melhorá-lo, em muitas organizações.
- Cada projeto tem suas características peculiares e merece um processo que seja adequado.

Leitura Recomendada

- Ian Sommerville; Engenharia de Software, 6.a Edição; Addison-Wesley, **2003** Capítulo 3 (Texto base)
- Roger S. Pressman; Engenharia de Software, 5.a Edição; McGraw-Hill, **2002**
- A. Tucker; Software process models; CRC Press, **1997**
- M. Ould; Managing software quality and business risk; JohnWiley and Sons, **1999** (Capítulo 4)

Questões

1. Justifique suas respostas, com vase no tipo de sistema que está sendo desenvolvido, sugerindo o modelo mais apropriado de processo de software genérico que pode ser utilizado como base para o gerenciamento do desenvolviemtno dos seguintes sistemas:
   - Um sistema para controlar o mecanismo contra arrombamento de fechaduras, em um veículo;
   - Um sistema de realidade virtual para poiar a manutenção de software;
   - Um sistema de contabilidade para universidades, que substitua um sistema existente;
   - Um sistema interativo para passageiros de ferrovias, que encontre os horários dos trens a partir de terminais instalados nas estações.
2. Explique por que programas que são desenvolvidos utilizando o desenvolvimento evolucionário apresentam difícil manutenção.
3. Explique como o modelo em cascata do processo de software e o modelo de prototipação podem ser acomodados no modelo de processo em espiral.

Questões

4. Sugira por que é importante fazer uma distinção entre desenvolver os requisitos do usuário e desenvolver os requisitos do sistema, no processo de engenharia de requisitos.
5. Descreva as principais atividades no processo de *design* de software e as saídas dessas atividades. Utilizando um diagrama de entidade-relacionamento (MER), mostre os possíveis relacionamentos entre as saídas dessas atividades.
6. Quais os cinco componentes de um método de *design*? Considere qualquer método que você conheça e descreva seus componentes. Avalie o grau de exatidão do método escolhido.
7. Projete um modelo de processo para efetuar testes de sistema e registrar seus resultados.
8. Explique por que um sistema de software utilizado em um ambiente real deve ser modificado ou, progressivamente, se tornará menos útil.

Questões

9. Sugira como um esquema de classificação da tecnologia CASE pode ser útil para os gerentes responsáveis pela aquisição de sistemas CASE.
10. Pesquise a disponibilidade de ferramentas em seu ambiente local de desenvolvimento e classifique as ferramentas de acordo com os parâmetros (função, atividade, amplitude de apoio) sugeridos no texto.
11. Historicamente, a introdução de tecnologia tem provocado profundas mudanças no mercado de trabalho e, pelo menos temporariamente, fez com que algumas pessoas perdessem o emprego. Discuta se a introdução da tecnologia CASE pode ter as mesmas conseqüências para os engenheiros de software. Se você achar que não, explique por quê. Se achar que essa tecnologia reduzirá as oportunidades de emprego, será ético que os engenheiros que podem ser afetados resistam à introdução dessa tecnologia, passiva e ativamente?