Processos de Software

Metodologias e Desenvolvimento de Software

Pedro Salgueiro
pds@uevora.pt
CLV-256

Processos de software

- Conjunto estruturado de atividades usado para desenvolver software

- Grande variedade de processos de software, mas todos partilham:
  - Especificação: define-se o que o sistema deve fazer;
  - Desenho e implementação: definir a organização do sistema e implementar o sistema;
  - Validação: verificar se o sistema faz o que deve
  - Evolução: alterar o sistema de acordo com novos requisitos

- Modelo de processo de software
  - Representação abstrata de um processo
  - Descrição de um processo visto de uma perspetiva especifica

Descrição dos processos

- Conjunto de atividades:
  - Especificar o modelo de dados;
  - Desenhar o interface de utilizador;
  - …
- **Ordem entre estas atividades**
- Podem incluir:
  - Produtos resultantes de uma atividade;
  - “Papéis” que refletem as várias responsabilidades das pessoas envolvidas no processo;
  - Pré e pós-condições que devem ser verificadas antes e depois do processo ou produto;

Tipos de processos de software

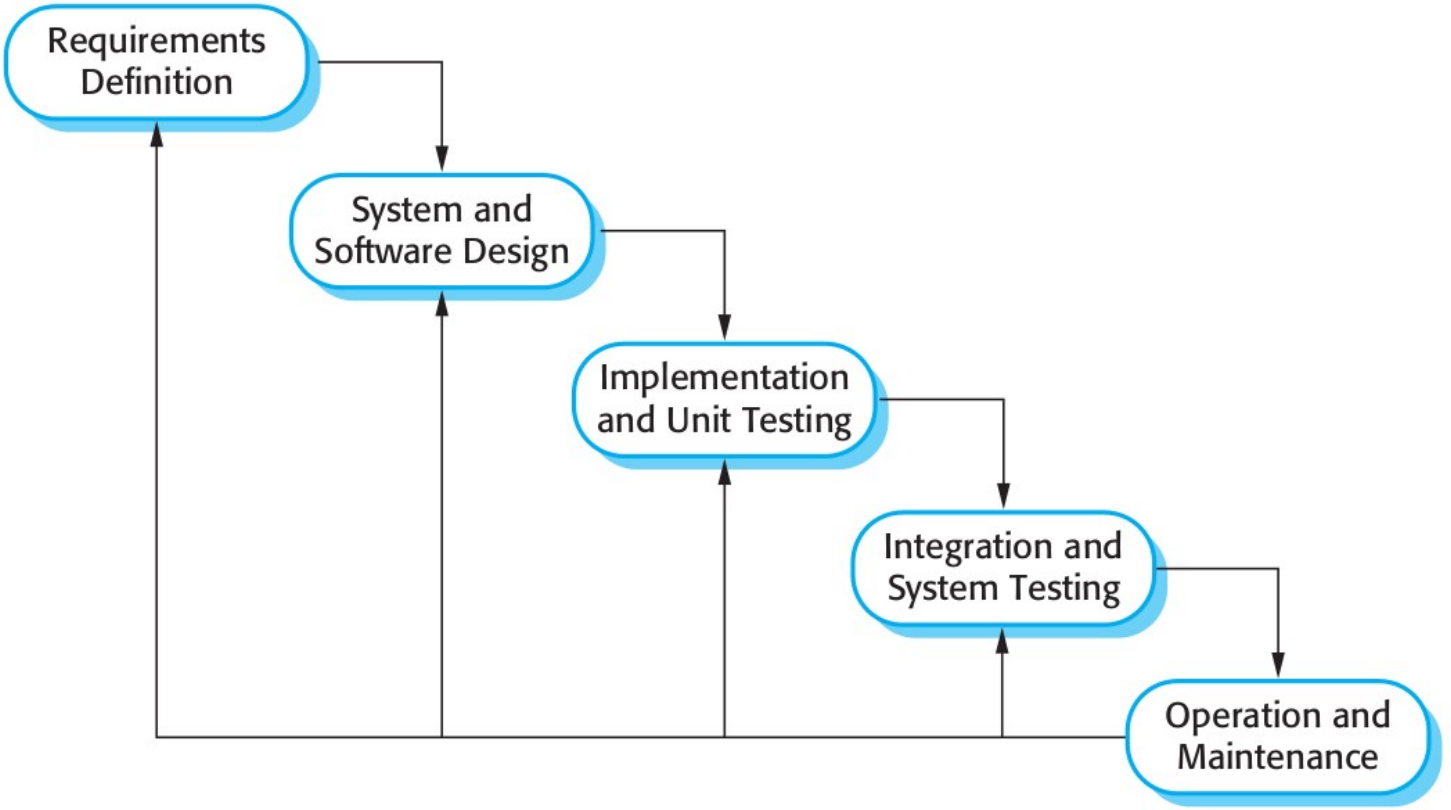
- Baseados em planos
  - Todas as atividades são planeadas com antecedência
  - Progresso é verificado de acordo com o o plano
- Processos ágeis
  - Planeamento incremental
  - Mais fácil alterar o processo
    - Refletir as alterações dos requisitos
- Na prática
  - Elementos dois tipos
- Não existe um processo correto ou errado

Processos de software

Modelos de processos de software

- Waterfall (cascata)
  - Baseado em planos
  - Especificação e desenvolvimento
    - Etapas separadas e distintas
- Desenvolvimento incremental
  - Especificação, desenvolvimento e validação
    - Entrelaçadas
  - Podem ser baseados em planos ou processos ágeis
- Integração e configuração
  - O sistema é construído através da configuração de vários componentes.
  - Pode ser baseado em planos ou métodos ágeis
- Na prática
  - Processos reais incluem elementos de métodos baseados em planos e de métodos ágeis
- Não existem processos de software melhores ou piores
  - Depende do tipo de software

Modelo Watterfall - overview



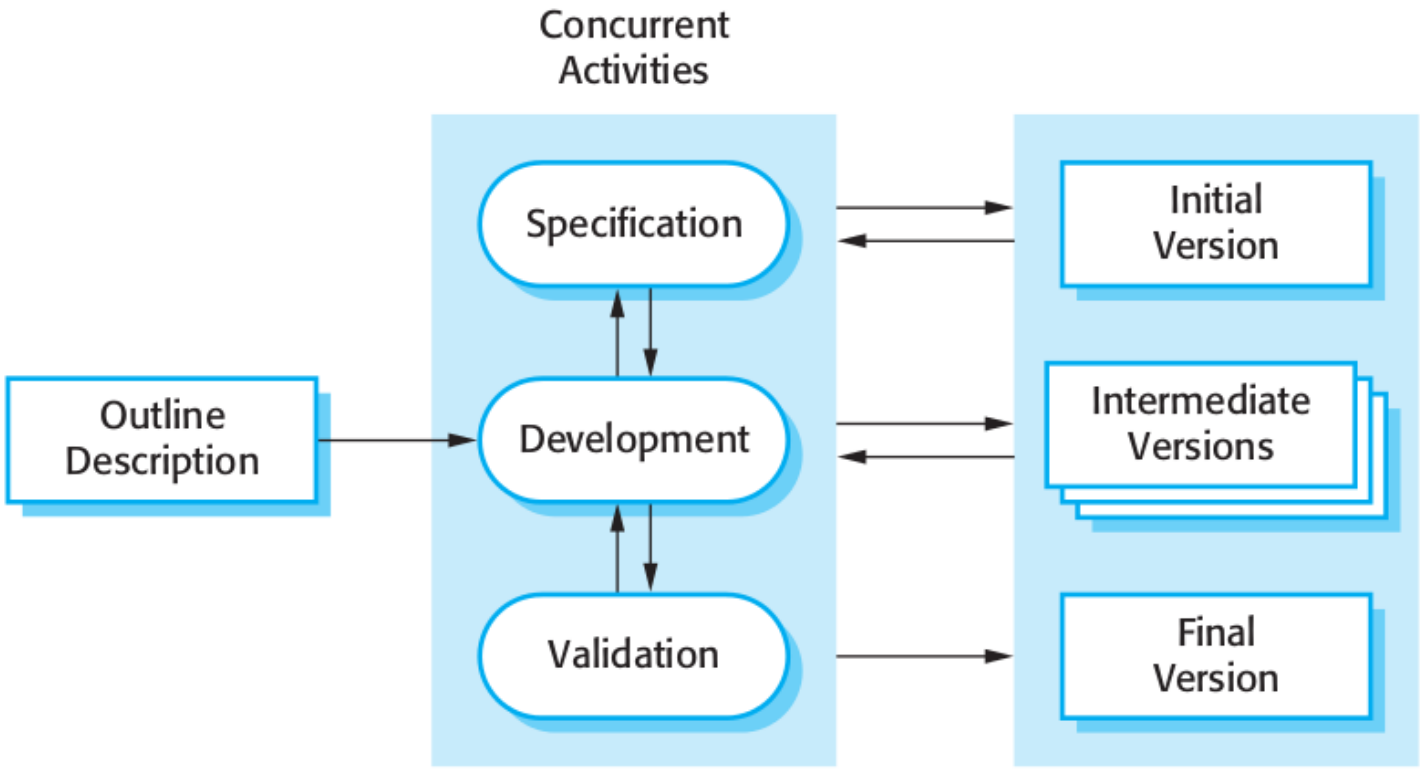
Modelo Waterfall

- Etapas
  - Análise e especificação dos requisitos
  - Desenho do software
  - Implementação e testes unitários
  - Integração e testes de sistema
  - Operação e manutenção
- Desvantagens
  - Dificuldade em incluir alterações depois de dar início ao processo
  - Cada etapa apenas começa depois da ultima terminar
    - Normalmente!

Modelo Waterfall

- Problemas
  - Divisão inflexível do projeto em diferentes etapas
    - Dificulta a resposta a alterações de requisitos
      - Apenas apropriado quando os requisitos são bem conhecidos desde o inicio e as alterações serão limitadas durante todo o processo;
      - Poucos sistemas têm requisitos estáveis.
  - Utilização (normalmente):
    - para sistemas grandes;
    - desenvolvidos por equipas grandes;
    - Em diferentes locais (geográficos) diferentes
      - Processos baseados em planos ajudam a coordenar o trabalho;

Modelo incremental - overview



Modelo incremental

- Vantagens
  - Custos para incluir alterações de requisitos é reduzido
    - Menos análise e documentação
  - Mais fácil obter *feedback* do cliente relativo ao desenvolvimento que está a ser feito
    - Clientes podem analisar demonstrações do software e perceber o que já está implementado
  - Entrega e *deployment (instalação, colocação em funcionamento)* mais rápido de software utilizável
    - Clientes podem usar o software mais cedo

Modelo incremental

- Problemas
  - O processo não é “visível”
    - Difícil de medir o progresso
      - Software desenvolvido de forma rápida
      - Não é eficaz produzir documentação que acompanhe todas as versões do sistema
  - Estrutura do software degrada-se a cada incremento
    - Incorporar novas funcionalidades torna-se cada vez mais difícil
    - “Solução”
      - *Refactoring*

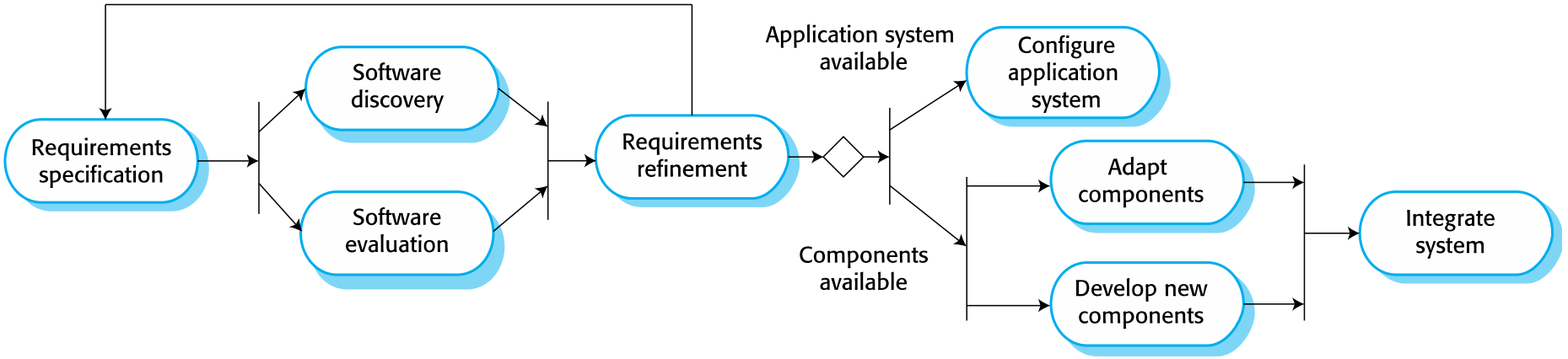
Integração e configuração

- Baseado na reutilização de software
  - Sistemas são construídos usando componentes ou aplicações existentes (COTS – Commercial-off-the shelf systemas)
- Elementos reutilizáveis podem ser configurados
  - Adaptar o seu funcionamento aos requisitos do novo sistema
- Reutilização é uma abordagem comum na construção de muitos sistemas

Tipos de componentes disponíveis

- *Web services*
  - Disponíveis para serem usados remotamente (ou localmente)
- Coleções de objetos
  - Funcionalidades específicas
  - Exemplos: bibliotecas diversas
- Sistemas *standalone*
  - Configuráveis para diferentes ambientes

Modelos baseados em reutilização - overview



Integração e configuração

- Etapas do processo
  - Especificação dos requisitos
  - Pesquisa e análise do software
  - Alteração/Adaptação de requisitos
  - Configuração das aplicações do sistema
  - Adaptação e integração de componentes

Atividades

- Processos de software reais
  - Sequências entrelaçadas/*imbricadas*
    - Atividades técnicas, colaborativas e de gestão
    - Objetivo
      - Especificar, desenhar, implementar e testar software
- Atividades básicas
  - Especificação, desenvolvimento, validação e evolução
  - Organizadas de forma diferente em diferentes processos
    - Waterfall: sequência
    - Incremental: entrelaçadas

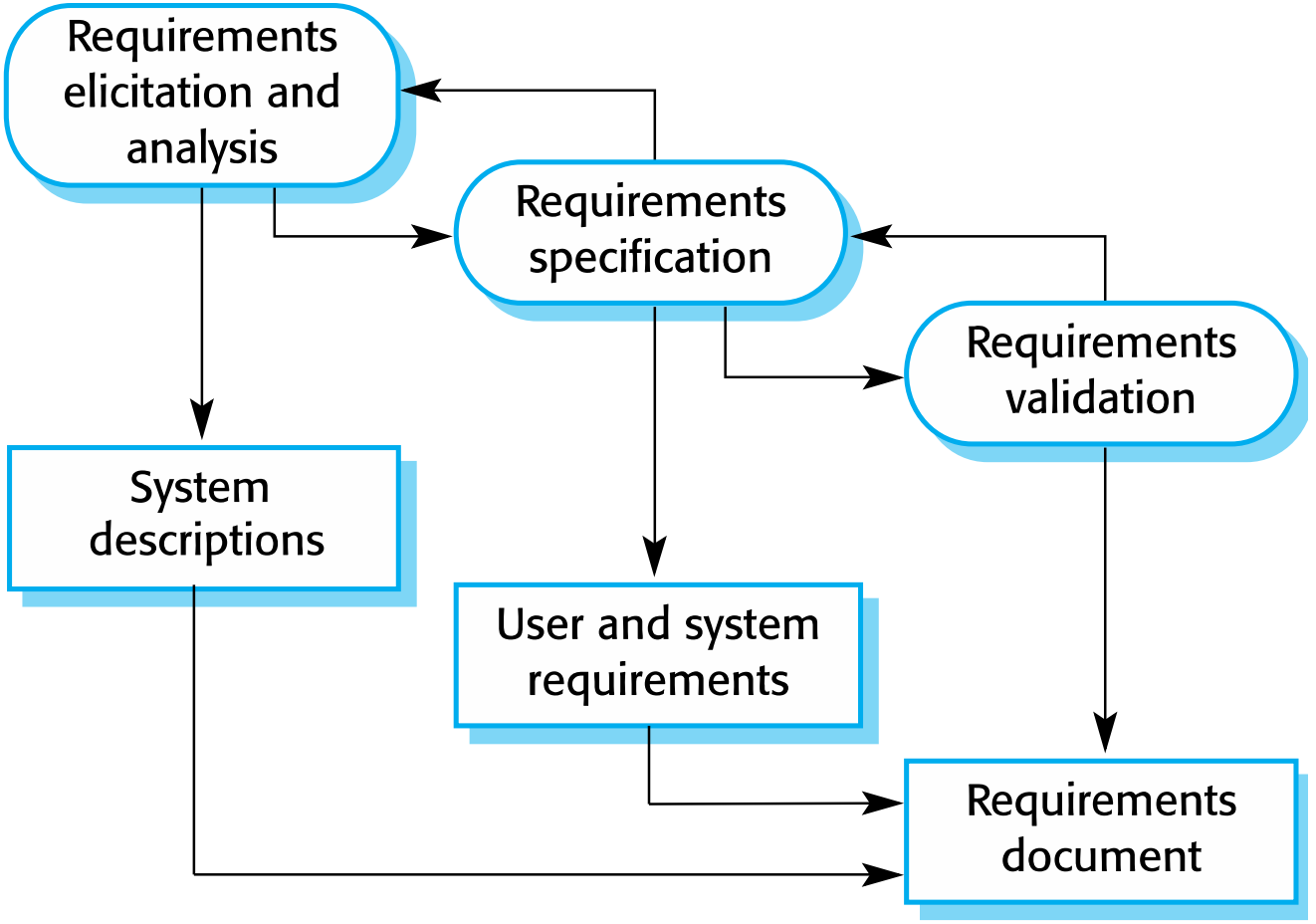
Integração e configuração

- Vantagens e desvantagens
  - Custos e riscos reduzidos
    - Menos software é criado de raiz
  - Entregas e *deployment* do sistema mais rápidas
  - Compromissos com os requisitos
    - Sistema pode não estar de acordo com as reais necessidades dos utilizadores
  - Não existe controle sobre a evolução dos componentes reutilizados

Actividades do processo

- Processos reais de software consistem em sequências intercaladas de atividades técnicas, colaborativas e de gestão
  - Com o objetivo de especificar, desenhar, implementar e testar um sistema de software
- As atividades básicas são:
  - Especificação, desenvolvimento e evolução
  - Organizadas de forma diferente em diferentes processos
- Waterfall
  - Organizadas em sequência
- Incremental
  - Intercaladas

Processo de engenharia de requisitos



Especificação de software

- Processo para especificar
  - Funcionalidades do sistema;
  - Restrições:
    - Do sistema;
    - Do processo de desenvolvimento do sistema;

- Processo de engenharia de requisitos
  - Análise dos requisitos
    - O que cada interessado necessita ou espera do sistema
  - Especificação dos requisitos
    - Definir os requisitos em detalhe
  - Validação dos requisitos
    - Verificar a validade dos requisitos

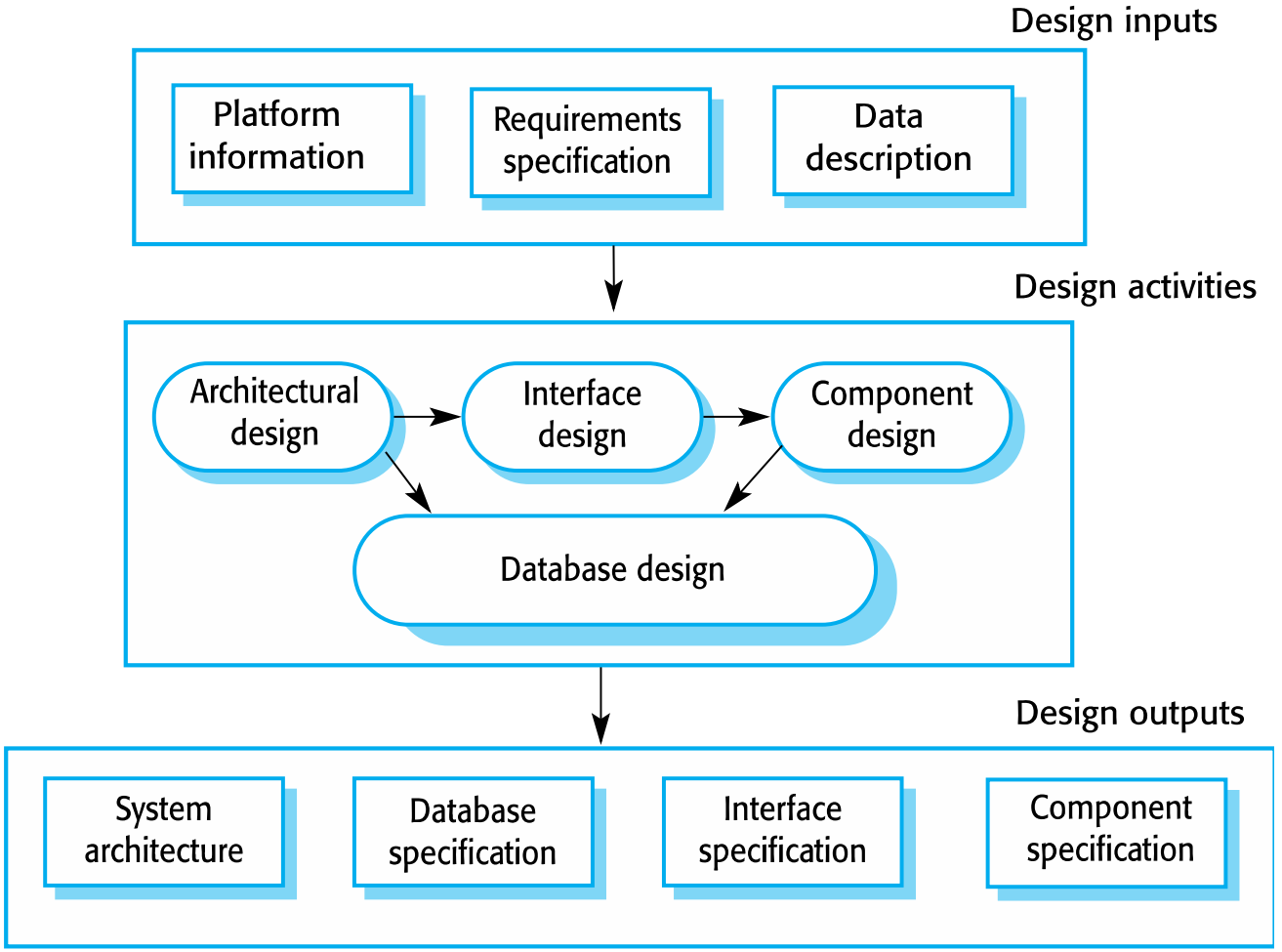
Desenho e implementação

- Converter a especificação num sistema executável/usável

- Tarefas
  - Desenho do software
    - Desenhar a estrutura que concretiza a especificação;
  - Implementação
    - “Traduzir” a estrutura num programa executável

- Desenho e implementação
  - Tarefas intrinsecamente relacionadas
  - Podem ser intercaladas

Processo de desenho - overview



Atividades de desenho

- Atividades
  - Desenho da arquitetura
    - Arquitetura global do sistema
    - Componentes principais (componentes ou módulos)
      - Relações entre si
  - Desenho da base de dados
    - Estrutura dos dados
    - Representação dos dados na base de dados
  - Desenho do interface
    - entre os diversos componentes
  - Escolha e desenho dos componentes
    - Pesquisa de componentes reutilizáveis
    - E/ou desenho de componentes

Implementação do sistema

- Implementação
  - Desenvolvimento de software
  - Configuração de sistemas existentes
- Design e implementação
  - São tipicamente atividades intercaladas
- Programação/Desenvolvimento
  - Atividade individual, sem processo standard
- *Debugging*
  - Atividade de encontrar e corrigir problemas

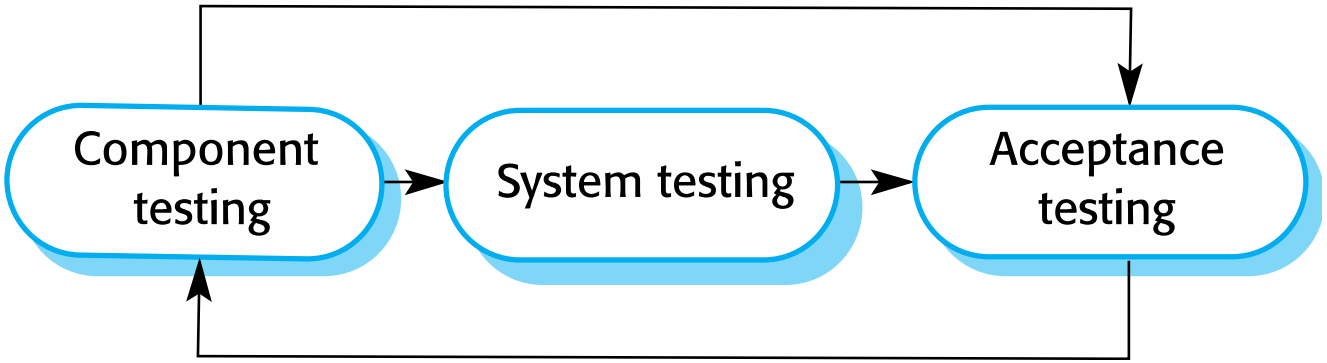
Validação de software

- Objetivo
  - Verificação e validação;
  - Está de acordo com as especificações;
  - Cumpre os requisitos do cliente;

- Métodos
  - Processos de revisão e verificação;
  - Testes do sistema (o mais usado);

- Testes
  - Executar o sistema;
  - Casos derivados das especificações e dos dados reais;
  - Encontrar erros;

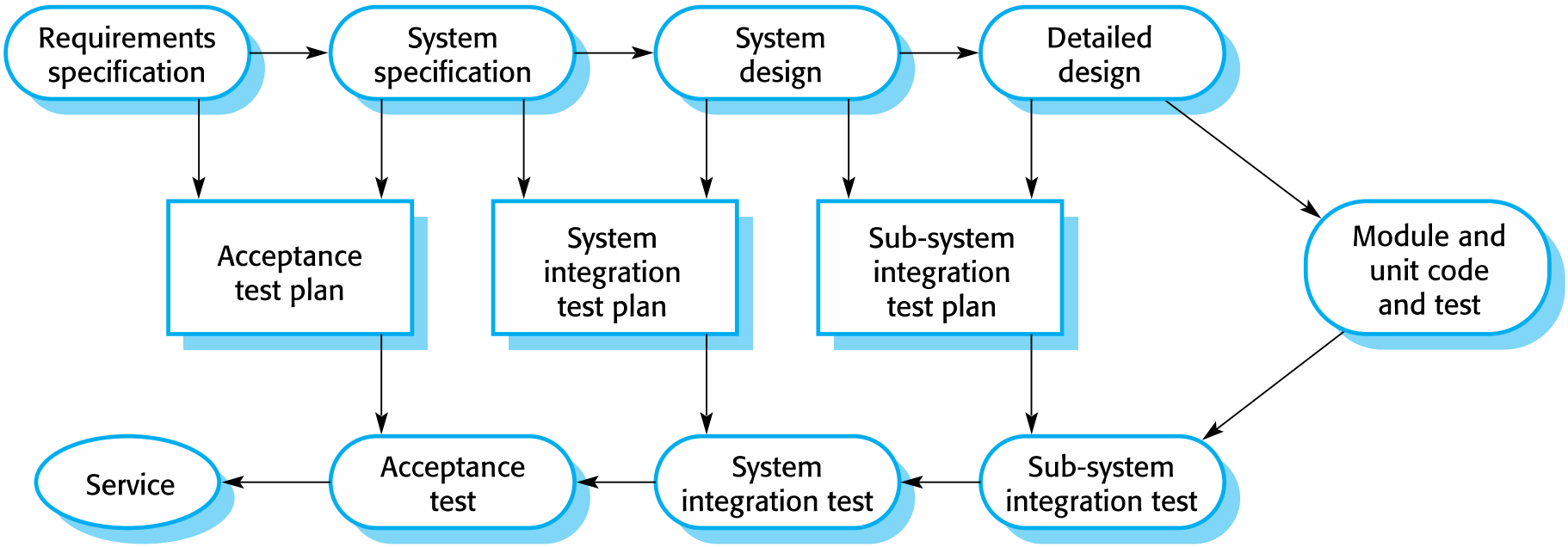
Etapas dos testes



Etapas dos testes

- Testes de desenvolvimento ou de componentes
  - Componentes individuais são testados de forma independente
  - Componentes: funções, objetos ou grupos (de funções e objetos)
- Testes de sistema
  - Testar o sistema como um só
  - Testar propriedades/funcionalidades é importante
- Testes de aceitação
  - Testes ao sistema usando dados do cliente
  - Verificar se está de acordo com as necessidades do cliente

Etapas de testes - processo baseado em planos

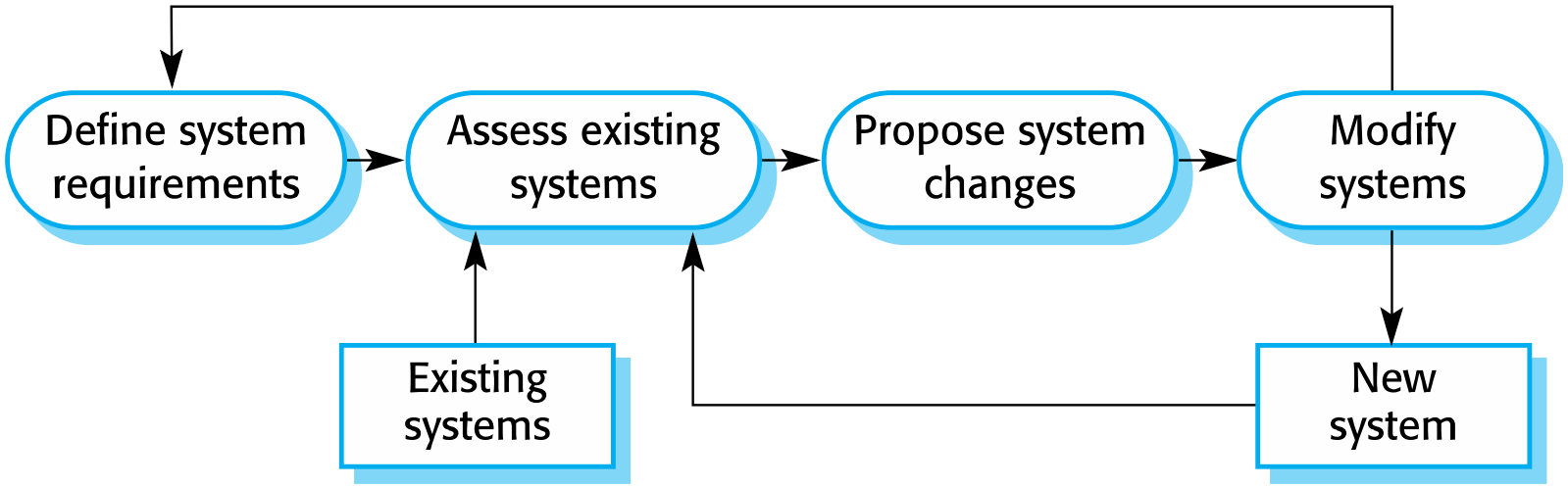


Evolução de software

- Software é flexível e altera-se (evolui)
  - Regras do negócio mudam
  - Requisitos mudam
  - Software tem de adaptar-se

- Desenvolvimento e evolução (manutenção)
  - Atividades consideradas distintas
  - Evolução
    - Continuação do desenvolvimento
  - Grande parte dos sistemas não são feitos completamente de raiz
    - Evolução tem um papel importante

Overview


Define system requirements
Assess existing systems
Propose system changes
Modify systems
Existing systems
New system

Entrega incremental

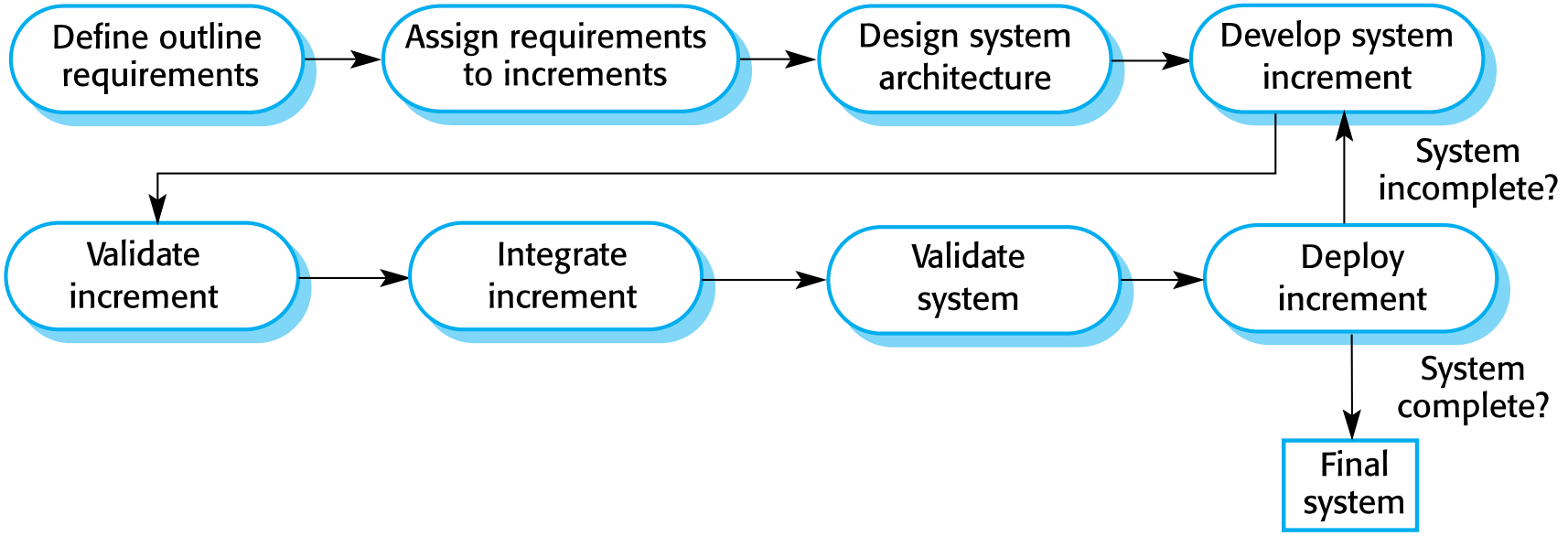
- Sistema entregue por incrementos
  - Desenvolvimento e entrega são divididos em “incrementos”
  - Cada incremento
    - Introduz uma funcionalidade (ou parte)
- Prioridades
  - Requisitos do utilizador têm maior prioridade.
    - Incluídos nos primeiros incrementos
- Em cada incremento
  - Requisitos desse incremento são “congelados”
    - Requisitos para os incrementos seguintes continuam a evoluir

Desenvolvimento e entrega incremental

- Desenvolvimento incremental
  - Desenvolver o sistema por partes/incrementos
    - Avaliar cada incremento antes de passar para o próximo
  - Prática comum em processos ágeis
  - Avaliação pode ser feita pelo cliente
- Entrega incremental
  - Permite fazer o *deploy* de incrementos
    - Usáveis pelo utilizador final
    - Avaliação mais real sobre a utilização prática do sistema

Entrega incremental

Overview



Entrega incremental

Vantagens

- A cada incremento podem ser entregues novas funcionalidades ao cliente
  - Sistema está disponível mais cedo
- Incrementos iniciais podem servir de protótipos
  - Podem ajudar a identificar requisitos para os próximos incrementos
- Menor risco do projeto falhar
- Os serviços/funcionalidades mais importantes tendem a ser mais testados
  - São implementados nas fases iniciais

Entrega incremental

Desvantagens/problemas

- Muitos sistemas necessitam de um conjunto base de funcionalidades comum a todo o sistema
  - Como os requisitos apenas são especificados em detalhe quando o incremento vai ser implementado, pode tornar-se difícil identificar as funcionalidades comuns
- Especificação desenvolvida juntamente com o software
  - Pode ser um conflito para algumas empresas
    - Onde a especificação completa do sistema pode fazer parte do “contrato” para desenvolver o sistema

Bibliografia

- *Software Engineering.* Ian Sommerville. 10th Edition. Addison-Wesley. 2016. Capítulo 2.