

Desenho de Software

**O processo de desenho
Estratégias de desenho
Qualidade de desenho**

João Araújo

Engenharia de SW

Desenho de Software

- Processo criativo
- Dedicção e talento
- Prática e aprendizagem → experiência e estudo de sistemas existentes
- Qualquer problema de desenho deve ser tratado em 3 estágios:
 - Estudo e compreensão do problema → problema examinado em ângulos diferentes
 - Escolha de uma solução → alternativas; experiência e disponibilidade; soluções familiares
 - Descrever cada abstração usada na solução

João Araújo

Engenharia de SW

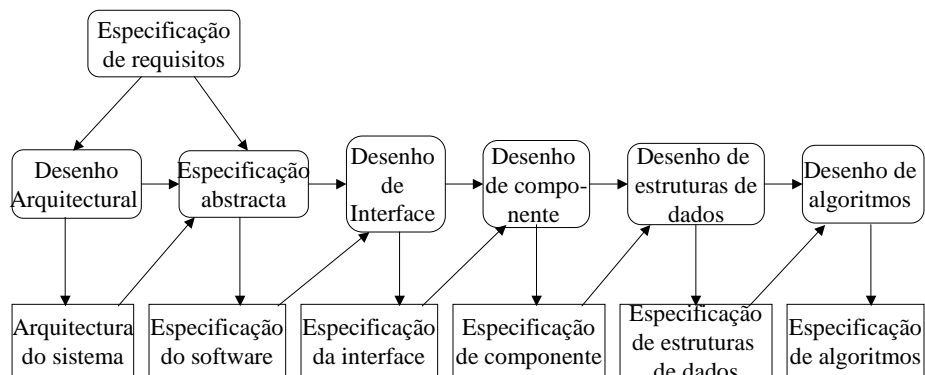
O Processo de Desenho

- Ponto de partida → especificação de requisitos
- Refinamento, mais detalhes
- Decomposição → níveis mais baixos de abstracção
- Erros e omissões revelados
- Actividades
 - Desenho de arquitectura → decomposição em subsistemas
 - Especificação abstracta → para cada subsistema uma especificação dos serviços e restrições são produzidos
 - Desenho de interface → para cada subsistema sua interface com outros subsistemas é projectada e documentada
 - Desenho de componentes → Serviços são alocados a componentes diferentes e as interfaces são desenhadas
 - Desenho de estruturas de dados
 - Desenho de algoritmos

João Araújo

Engenharia de SW

O Processo de Desenho

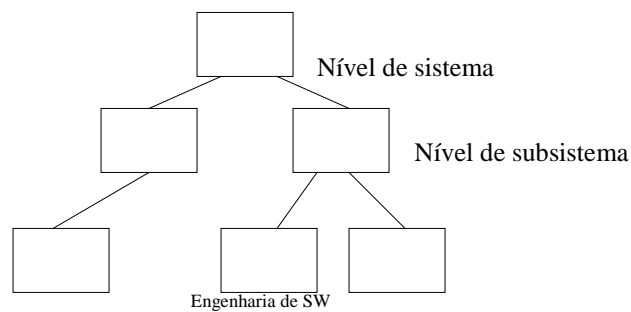


João Araújo

Engenharia de SW

O Processo de Desenho

- Estratégia top-down
 - Problema particionado em sub-problemas
 - Decomposição sistemática
 - Uso de conhecimento prévio



João Araújo

Engenharia de SW

Métodos e Descrição

- Diagramas + CASE tools
- Budgen descreve os métodos mais comuns
- Métodos estruturados (funcionais)
 - Ex: Yourdon, Gane & Sarson
- Métodos orientados a estruturas de dados
 - Ex: JSD, Warnier-Orr
- Métodos orientados a objectos
 - Ex: Booch, OMT, Coad & Yourdon
- Descrição
 - Notação gráfica
 - PDL
 - Texto

João Araújo

Engenharia de SW

Estratégias de Desenho

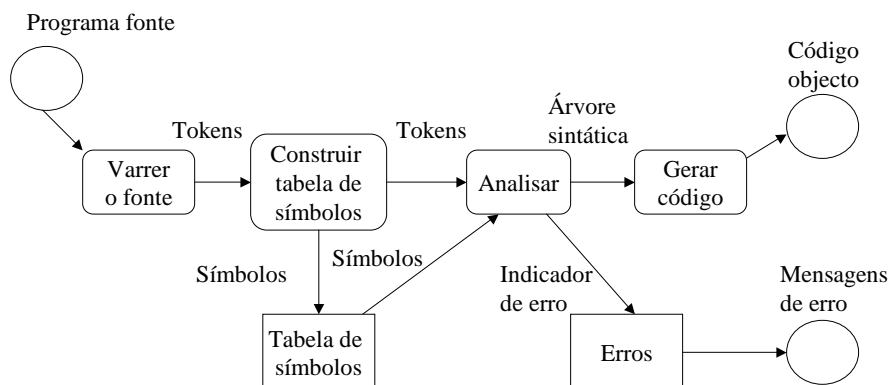
- Funcional → o estado do sistema é centralizado e compartilhado entre as funções
 - Ex: Yourdon, Gane & Sarson, JSD
- Orientada a Objectos → Visto como colecção de objectos
 - O estado do sistema é descentralizado
 - Cada objecto gere o seu próprio estado
 - Os objectos comunicam-se através de troca de mensagens

João Araújo

Engenharia de SW

Estratégias de Desenho

Visão funcional de um compilador

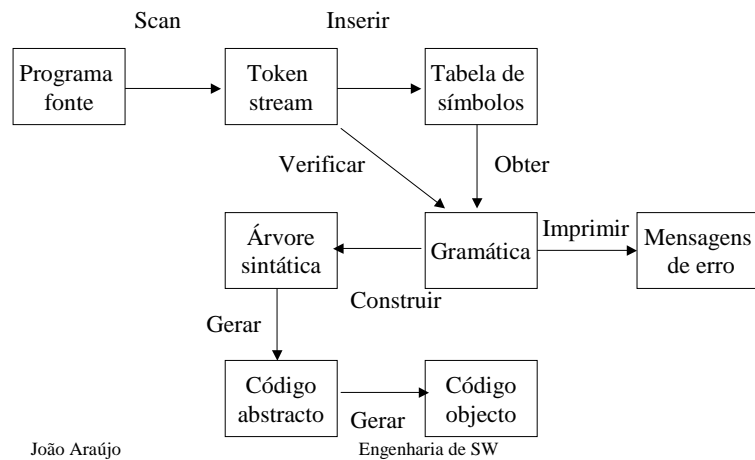


João Araújo

Engenharia de SW

Estratégias de Desenho

Visão orientada a objectos de um compilador



Qualidade de Desenho

- Um bom desenho:
 - O que produz código eficiente
 - O que produz implementação compacta
 - O que produz um sistema fácil de manter
- Coesão
 - A coesão de um componente é uma medida da proximidade das relações entre seus componentes
 - Um componente → função lógica ou entidades simples
 - Facilita a mudança

Qualidade de Desenho

- Acoplamento
 - Grau de interconexões entre os componentes
 - É desejável que os os sistemas sejam fracamente acoplados
 - OO → favorece o acoplamento fraco pois não tem estado compartilhado

João Araújo

Engenharia de SW

Qualidade de Desenho

- Coesão (cont.)
 - Níveis:
 - Coincidental
 - Lógica
 - Temporal
 - Procedimento → uma sequencia de controle
 - Comunicação → mesmo input e output
 - Sequencial → output serve como input
 - Funcional → cada parte necessária p/ execução de função;
 - OO → coesão natural

João Araújo

Engenharia de SW

Qualidade de Desenho

- Facilidade de Compreensão
 - Facilita manutenção de desenho
 - Coesão, acoplamento
 - Nomes significativos
 - Documentação clara
 - Complexidade dos algoritmos
- Adaptabilidade
 - Estimativa geral de quão fácil é mudar o desenho
 - Componentes fracamente acoplados
 - Bem documentado
 - Documentação fácil de compreender
 - Consistente com implementação
 - Alto nível de traço → clara relação entre diferentes níveis no desenho

João Araújo

Engenharia de SW

Desenho Arquitectural

Estruturação do sistema
Modelos de controlo
Decomposição modular

João Araújo

Engenharia de SW

Desenho Architectural

- Processo de identificar subsistemas e estabelecer um framework p/ controlo e comunicação de subsistemas
- Actividades principais:
 - Estruturação do Sistema → o sistema é estruturado em subsistemas onde cada subsistema é uma unidade de software independente
 - Modelagem de Controlo → Um modelo geral dos relacionamentos de controle entre as partes do sistema é estabelecido
 - Decomposição modular → cada subsistema é decomposto em módulos
- Output → documento de desenho architectural
 - Representação gráfica dos modelos+ texto
 - Sistema → subsistemas → módulos

João Araújo

Engenharia de SW

Desenho Architectural

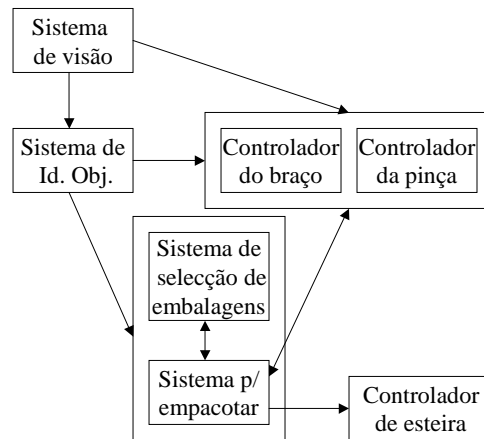
- Subsistema → é um sistema cuja operação não depende dos serviços de outros subsistemas.
 - Composto por módulos
 - Interfaces são utilizadas p/ comunicação entre subsistemas
- Módulo → componente que fornece e utiliza um ou mais serviços p/ outros módulos
- Modelo afeta:
 - Desempenho
 - Robustez
 - Distribuição
 - Manutenção

João Araújo

Engenharia de SW

Estruturação de Sistema

- Nível mais geral:
Diagrama de blocos.
 - Cada bloco é um subsistema
 - Setas são passagem de dados/controlro
- Tipos mais específicos:
 - Repositório
 - Cliente-servidor
 - Máquina-abstrata

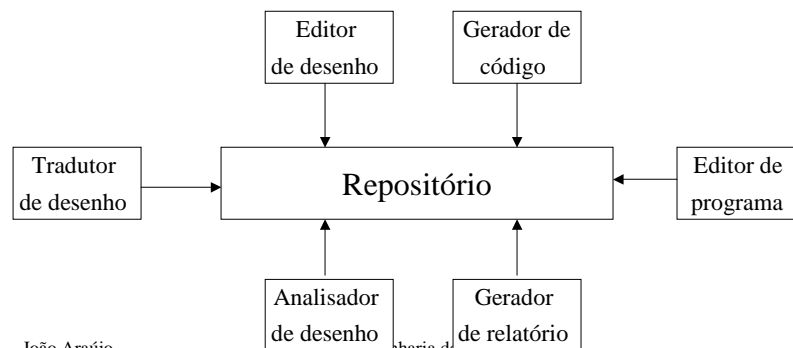


João Araújo

Engenharia de SW

Modelo Repositório

- Todos os dados compartilhados estão guardados em uma base de dados central acedida por todos os subsistemas
 - Ex: Sistemas de gestão de informação, sistemas CAD
- Arquitectura de um CASE integrado:



João Araújo

Engenharia de SW

Modelo Repositório

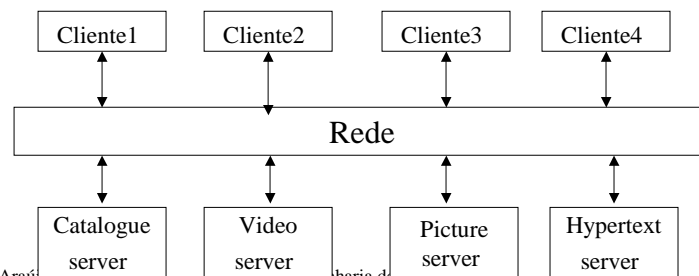
- Vantagens (+) e desvantagens (-)
 - + Eficiente para compartilhar grandes quantidades de dados
 - - Todos os subsistemas devem concordar com um modelo de dados. Difícil de integrar novos subsistemas que não concordam com o modelo de dados
 - + Subsistemas que produzem dados não precisam se preocupar como estes são usados por outros subsistemas
 - - Evolução mais difícil → deve-se estar de acordo com o modelo
 - + Actividades de back-up, segurança, controle de acesso, são centralizadas no gestor do repositório
 - - Subsistemas diferentes podem ter diferentes políticas para estas atividades

João Araújo

Engenharia de SW

Modelo Cliente-Servidor

- É um modelo de sistemas distribuído e mostra como dados e processamento estão distribuídos através dos processadores
- Componentes do modelo
 - Conjunto de servidores que oferecem serviços a outros subsistemas
 - Conjunto de clientes que usam os serviços dos servidores
 - Uma rede que permite clientes acessarem serviços
- Arquitectura de um sistema de uma biblioteca de fotos e filmes:



João Araújo

Engenharia de SW

Modelo Cliente-Servidor

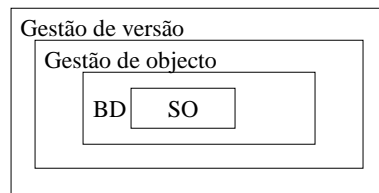
- Clientes devem saber o nome e os serviços dos servidores. Os servidores não
- Clientes acedem aos serviços de um servidor através de chamadas remotas
- Vantagens
 - A distribuição é rápida e directa
 - Fácil de incluir novo servidor
 - Uso efetivo pode ser feito c/ vários processadores distribuídos
- Desvantagens
 - Não há modelo de dados compartilhado → difícil de antecipar problemas e integrar dados de um novo servidor
 - Cada servidor é responsável pelas rotinas de gestão (backup)

João Araújo

Engenharia de SW

Máquina abstrata

- Organiza o sistema em uma série de camadas onde cada camada provê um conjunto de serviços
- Cada camada define uma máquina abstrata onde sua linguagem de máquina (seus serviços) é utilizada para implementar o próximo nível (camada)
- Exemplos
 - Definição de protocolos de rede, APSE
- Sistema de gestão de versões:



João Araújo

Máquina abstrata

- Vantagens:
 - Dá suporte desenvolvimento incremental
 - Sua arquitetura é modificável
 - Sua arquitetura é portátil
 - Se a interface é preservada uma camada pode ser trocada por outra
- Desvantagens:
 - Estruturação pode ser difícil
 - Facilidades básicas (gestão de ficheiros) devem ser providas por camadas mais interiores subvertendo o modelo
 - Desempenho pode ser comprometido se houver muitas camadas

João Araújo

Engenharia de SW

Modelos de Controlo

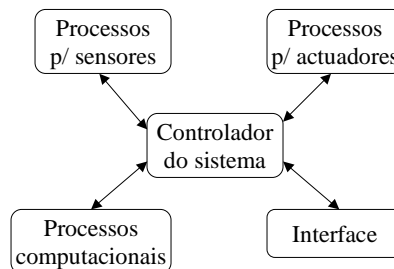
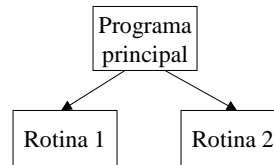
- Subsistemas devem ser controlados tal que seus serviços sejam prestados no lugar e tempo certo
- Modelam o fluxo de controle entre subsistemas
- Tipos:
 - *Centralizado* → um subsistema é responsável por todo o control, início e parada dos outros subsistemas
 - *Baseado em eventos* → Cada subsistema pode responder a eventos gerados externamente que podem vir de outros subsistemas ou do ambiente

João Araújo

Engenharia de SW

Controle Centralizado

- Modelo call-return
 - Top-down
 - Hierarquia de subrotinas
 - Aplica-se a sistemas sequenciais
- Modelo Gestor
 - Aplicável a sistemas concorrentes
 - Um componente do sistema é responsável e controla o início, parada e coordenação de outros processos do sistema
 - Um processo é um subsistema ou módulo que pode executar em paralelo com outros processos

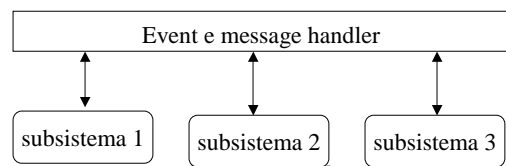


João Araújo

Engenharia de SW

Sistemas Orientados a Eventos

- Orientados por eventos gerados externamente
- Modelos broadcast → um evento é transmitido p/ todos os subsistemas
 - Na prática o gestor de mensagens mantém um registo de subsistemas e dos eventos que interessam-lhes
 - Vantagem → evolução simples, a inserção de novo subsistema é transparente para outros subsistemas
 - Desvantagem → os subsistemas não sabem quando ou se os eventos serão realizados; conflitos podem ser gerados

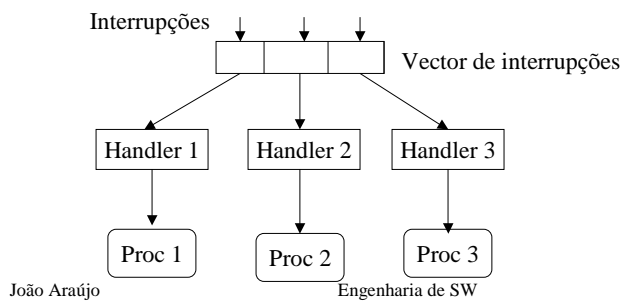


João Araújo

Engenharia de SW

Sistemas Orientados a Eventos

- Modelo orientado à interrupções → estas são detectadas por um *handler* que são passadas para outro componente para processamento
- Ex.: sistemas de tempo real
 - Vantagem → proporciona respostas rápidas aos eventos
 - Desvantagem → é complicado para programar e validar



Decomposição Modular

- Subsistema → Módulos
- Modelo de fluxo de dados
 - Transformações (processos)
 - + Intuitivo
 - - Difícil de manter
- Modelo orientado a objectos
 - Cada classe encapsula atributos e operações
 - + Módulos fracamente acoplados e fortemente coesos
 - + Representação mais fácil
 - + Reutilização
 - Desvantagens ?

João Araújo

Engenharia de SW