## Projeto de arquitetura

- Estabelecer a estrutura geral de um sistema de software



## Objetivos

- Apresentar o projeto arquitetural e discutir a sua importância
- Explicar por quê múltiplos modelos são exigidos são exigidos para documentar uma arquitetura de software
- Descrever tipos de modelos de arquitetura que podem ser usados
- Discutir como modelos de referência específicos do domínio podem ser usados como uma base para linhas de produto e comparar arquiteturas de software



## Tópicos abordados

- Estrutura do sistema
- Modelos de controle
- Decomposição modular
- Arquiteturas específicas de domínio



## Arquitetura de software

- O processo de projeto para identificar os sub-sistemas que comporão um sistema e o framework para controle e comunicação dos sub-sistemas é chamado *projeto arquitetural*
- A saída do processo de projeto é uma descrição da *arquitetura do software*

# Projeto da arquitetura

- Um processo inicial do processo de projeto de sistemas
- Representa a ligação entre os processos de especificação e projeto
- É freqüentemente conduzido em paralelo com algumas atividades de especificação
- Envolve a identificação dos principais componentes do sistema e a comunicação entre eles



# Vantagens da arquitetura explícita

- Comunicação com os stakeholders
  - A arquitetura pode ser usada como um foco de discussão com os stakeholders do sistema
- Análise do sistema
  - Significa que é possível analisar se um sistema pode satisfazer os seus requisitos não funcionais
- Reuso em larga escala
  - A arquitetura pode ser reusável entre uma gama de sistemas



# Processo de projeto arquitetural

- Estruturação do sistema
  - O sistema é decomposto em vários sub-sistemas principais e são identificadas as comunicações entre esses sub-sistemas
- Modelagem de controle
  - Um modelo dos relacionamentos de controle entre as diferentes partes do sistema é estabelecido
- Decomposição modular
  - Os sub-sistemas identificados são decompostos em módulos



# Sub-sistemas e módulos

- Um *sub-sistema* é um sistema em si, cuja operação é independente dos serviços fornecidos por outros sub-sistemas.
- Um *módulo* é um componente do sistema que fornece serviços para outros componentes, mas que normalmente não deveria ser considerado como um sistema separado

# Modelos de arquitetura

- Diferentes modelos de arquitetura podem ser produzidos durante o processo de projeto
- Cada modelo apresenta perspectivas diferentes da arquitetura



# Modelos de arquitetura

- Modelo estático da estrutura que mostra os principais componentes do sistema
- Modelo dinâmico de processo que mostra a estrutura dos processos do sistema
- Modelo de interface que define as interfaces entre os sub-sistemas
- Modelo de relacionamentos, como um modelo de fluxo de dados



# Estilos de arquitetura

- O modelo da arquitetura de um sistema pode estar de acordo com um modelo ou estilo de arquitetura genérico
- Um conhecimento desses estilos pode simplificar o problema de definir arquiteturas do sistema
- Entretanto, a maioria dos sistemas são heterogêneos e não seguem um único estilo de arquitetura



# Atributos da arquitetura

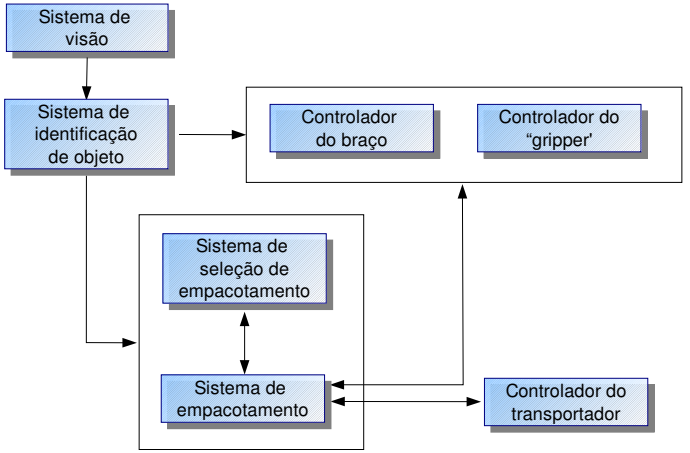
- Performance
  - Localizar operações para minimizar a comunicação entre os sub-sistemas
- Segurança
  - Usar uma arquitetura em camadas com verificações críticas nas camadas internas
- Proteção
  - Isolar componentes de proteção crítica
- Disponibilidade
  - Incluir componentes redundantes na arquitetura
- Manutenibilidade
  - Usar componentes de granularidade fina, auto-contidos

# Estruturação do sistema

- Preocupa-se com a decomposição do sistema em sub-sistemas que interagem
- O projeto da arquitetura é normalmente expresso como um diagrama de blocos, apresentando uma visão global da arquitetura do sistema
- Modelos mais específicos mostrando como os sub-sistemas compartilham dados, são distribuídos e interagem com cada um dos outros também podem ser desenvolvidos



# Sistema de controle de empacotamento robotizado



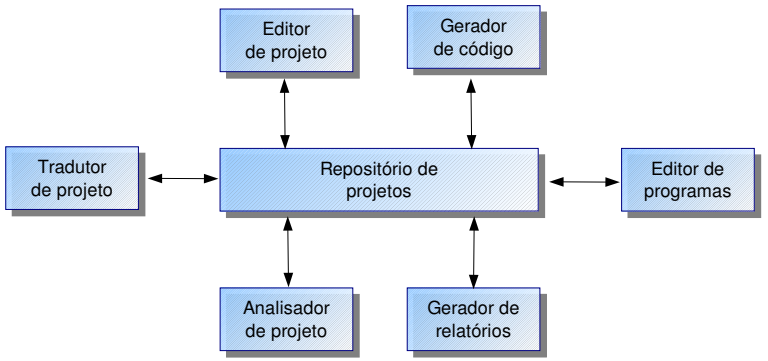



# O modelo de repositório

- Os sub-systems devem trocar dados. Isso pode ser feito de duas maneiras:
  - Dados compartilhados são mantidos em um banco de dados ou repositório central que pode ser acessado por todos os sub-sistemas
  - Cada sub-sistema mantém o seu banco de dados próprio e envia dados explicitamente para outros sub-sistemas
- Quando grandes quantidades de dados devem ser compartilhados, o modelo de repositório de compartilhamento é mais comumente usado



# Arquitetura de um conjunto de ferramentas CASE



# Características do modelo de repositório

- Vantagens
  - Maneira eficiente de compartilhar grandes volumes de dados
  - Os sub-sistemas não precisam se preocupar com a forma como os dados são produzidos
  - Gerenciamento centralizado. P.ex.: backup, segurança, etc.
  - O modelo compartilhado é publicado como o esquema do repositório
- Desvantagens
  - Os sub-sistemas devem concordar com o modelo de dados
  - A evolução dos dados é difícil e cara
  - Não há espaço para políticas de gerenciamento específicas
  - Difícil de distribuir de forma eficiente

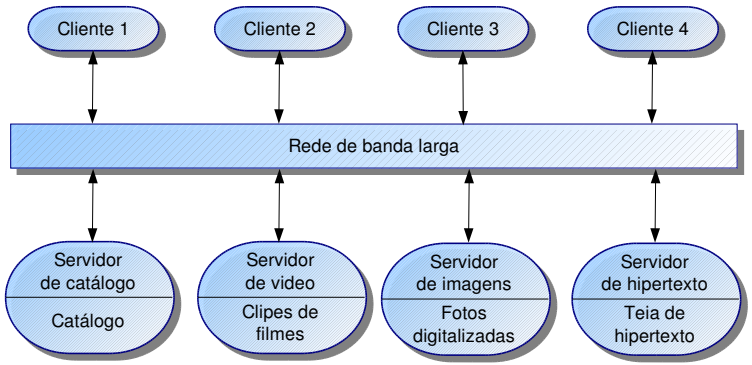


# Arquitetura cliente-servidor

- Modelo de sistema distribuído que mostra como os dados e o processamento é distribuído entre uma série de componentes
- Conjunto de servidores stand-alone que fornecem serviços específicos, tais como impressão, gerenciamento de dados, etc.
- Conjunto de clientes que solicitam estes serviços
- Rede que permite aos clientes o acesso aos servidores



# Biblioteca de filmes e imagens






# Características do modelo cliente-servidor

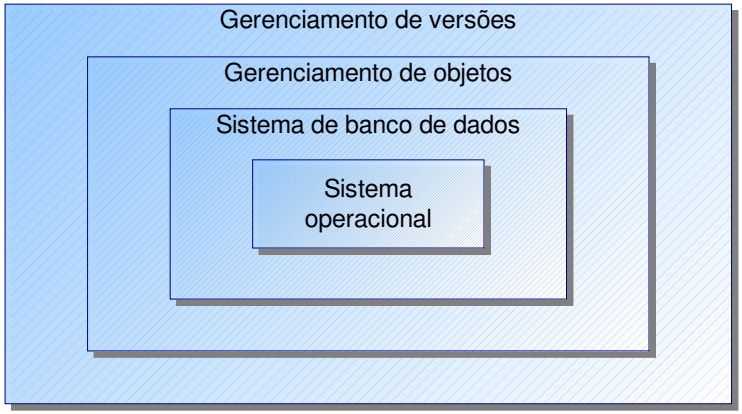
- Vantagens
  - A distribuição dos dados é fácil
  - Faz uso efetivo de sistemas em rede. Pode exigir hardware mais barato
  - É fácil adicionar novos servidores ou evoluir servidores existentes
- Desvantagens
  - Não existe um modelo de dados compartilhado, portanto os sub-sistemas usam organizações diferentes de dados. O intercâmbio de dados pode ser ineficiente
  - Gerenciamento redundante em cada servidor
  - Não há um registro central de nomes e serviços. Pode ser difícil de descobrir quais são os servidores e os serviços

# Modelo de máquina abstrata

- Usado para modelar a interface dos sub-sistemas
- Organiza o sistema em um conjunto de camadas (ou máquinas abstratas), cada uma das quais oferece um conjunto de serviços
- Suporta o desenvolvimento incremental de sub-sistemas em diferentes camadas. Quando a interface de uma camada muda, somente a camada adjacente é afetada
- Entretanto, é freqüentemente difícil de estruturar os sistemas desta maneira



# Sistema de gerenciamento de versões






# Modelos de controle

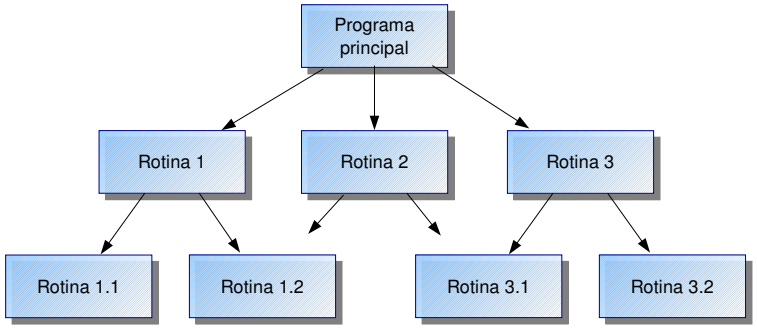
- Preocupam-se com o fluxo de controle entre sub-sistemas. Diferente do modelo de decomposição de sistemas
- Controle centralizado
  - Um sub-sistema tem responsabilidade geral para controlar e inicia e pára outros sub-sistemas
- Controle baseado em eventos
  - Cada sub-sistema pode responder a eventos gerados externamente por outros sub-sistemas ou pelo ambiente do sistema



# Controle centralizado

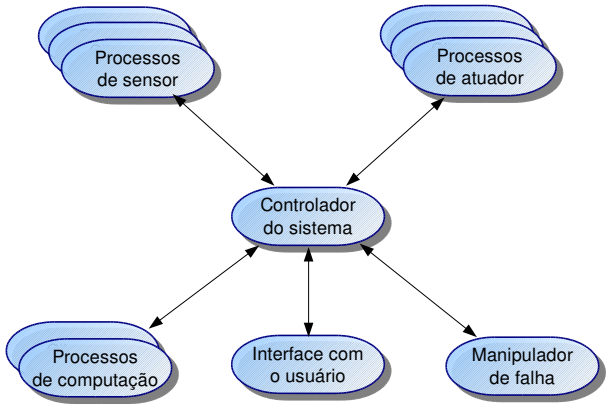
- Um sub-sistema de controle assume a responsabilidade de gerenciar a execução dos outros sub-sistemas
- Modelo “call-return”
  - Modelo de sub-rotina top-down, onde o controle inicia no topo de uma hierarquia de sub-rotinas e se move para baixo. Aplicável a sistemas seqüenciais
- Modelo de gerente
  - Aplicável a sistemas concorrentes. Um componente do sistema controla a parada, inicialização e coordenação de outros processos do sistema. Pode ser implementado em sistemas seqüenciais como um comando *case*

# Modelo call-return






# Controle de sistema de tempo real






# Sistemas orientados por eventos

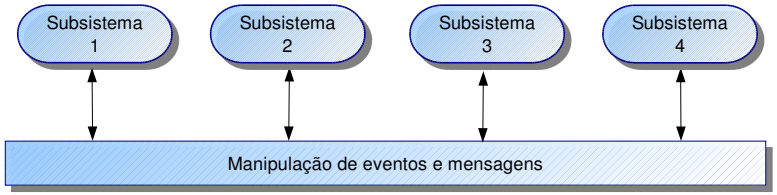
- Orientados por eventos gerados externamente, onde o evento está fora do controle dos sub-sistemas que processam o evento
- Dois modelos principais orientados por eventos
  - Modelos de difusão. Um evento é difundido para todos os sub-sistemas. Qualquer sub-sistema que consiga manipular o evento pode fazê-lo
  - Modelos orientados a interrupção. Usado em sistemas de tempo real onde interrupções são detectadas por um controlador de interrupções e passados para algum outro componente para processamento
- Outros modelos orientados a eventos incluem planilhas de cálculo e sistemas de produção



# Modelo de difusão

- Eficaz na integração de sub-sistemas em diferentes computadores em uma rede
- Os sub-sistemas registram interesse em eventos específicos. Quando eles ocorrem, o controle é transferido para os sub-sistemas que podem manipular o evento
- A política de controle não está embutida no evento e no gerenciador de mensagens. Os sub-sistemas escolhem os eventos que lhes interessam
- Entretanto, os sub-sistemas não sabem se ou quanto um evento será manipulado

# Difusão seletiva



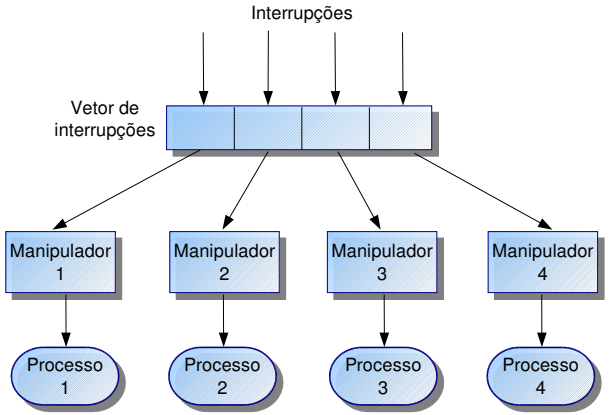



# Sistemas orientados por interrupções

- Usados em sistemas de tempo real, onde resposta rápida a um evento é essencial
- Há tipos conhecidos de interrupções, com um manipulador definido para cada tipo
- Cada tipo associado com uma localização na memória e um seletor de hardware causa a transferência para o seu manipulador
- Permite resposta rápida, mas é complexo de programar e difícil de validar



# Controle orientado por interrupções






# Decomposição modular

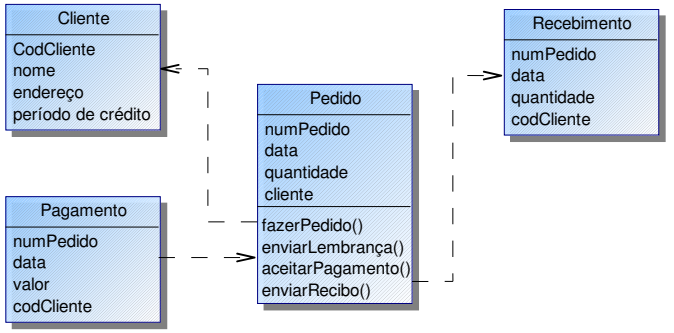
- Outro nível estrutural onde os sub-sistemas são decompostos em módulos
- Dois modelos de decomposição modular
  - Um modelo de objeto, onde o sistema é decomposto em objetos que interagem
  - Um modelo de fluxo de dados, onde o sistema é decomposto em módulos funcionais que transformam entradas em saídas. Também conhecido como modelo *pipeline*
- Se possível, decisões sobre concorrência devem ser retardadas até que os módulos estejam implementados

# Modelos de objetos

- Estruturam o sistema em um conjunto de objetos fracamente acoplados com interfaces bem definidas
- A decomposição orientada a objetos preocupa-se em identificar classes de objetos, seus atributos e operações
- Após a implementação, objetos são criados a partir destas classes, e algum modelo de controle é usado para coordenar as operações dos objetos



# Sistema de processamento de pedidos



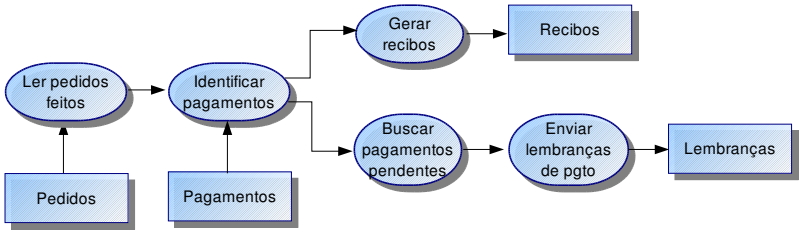



# Modelos de fluxo de dados

- Transformações funcionais processam as suas entradas para produzir saídas
- Pode ser chamado de modelo *pipeline* e modelo filtro (como em um *shell* UNIX)
- Variações desta abordagem são bastante comuns. Quando as transformações são seqüenciais, tem-se um modelo seqüencial em batch, que é largamente usado em sistemas de processamento de dados
- Não é adequado para sistemas interativos



# Sistema de processamento de pedidos



# Arquiteturas de domínio específico

- Modelos de arquitetura que são específicos a um domínio de aplicação
- Dois tipos de modelos de domínio específico
  - Modelos genéricos, que são abstrações de uma série de sistemas reais e que encapsulam as características principais destes sistemas
  - Modelso de referência, que são modelos mais abstrados e idealizados. Fornecem meios de informação sobre aquele tipo de sistema e da comparação entre diferentes arquiteturas
- Modelos genéricos são normalmente *bottom-up*; modelos de referência normalmente são *top-down*

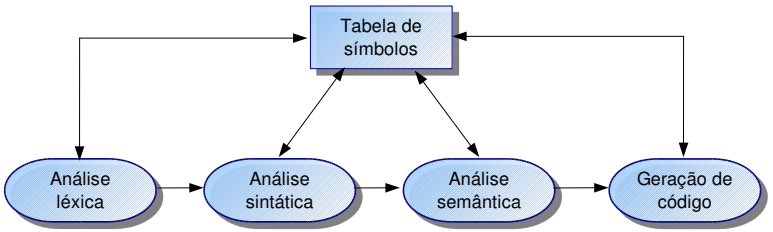


# Modelos genéricos

- O modelo de compilador é um exemplo bem conhecido, embora existam outros modelos em domínios de aplicação mais especializados
  - Analisador léxico
  - Tabela de símbolos
  - Analisador sintático
  - Árvore sintática
  - Analisador semântico
  - Gerador de código
- O modelo de compilador genérico pode ser organizado de acordo com diferentes modelos de arquitetura

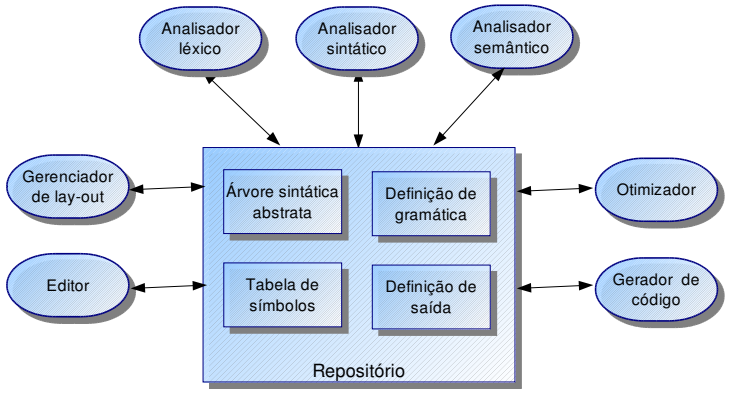


# Modelo de compilador






# Sistema de processamento de linguagem

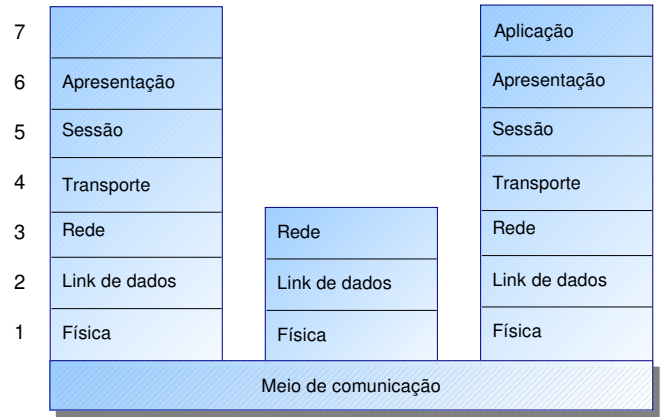


# Arquiteturas de referência

- Modelos de referência são derivados de um estudo do domínio da aplicação, em vez de sistemas existentes
- Pode ser usado como uma base para a implementação do sistema ou para comparar diferentes sistemas. Funciona como um padrão contra o qual os sistemas podem ser avaliados
- Ex.: O modelo OSI é um modelo em camadas para sistemas de comunicação



# Modelo de referência OSI






# Pontos principais

- O arquiteto de software é responsável por derivar um modelo de estrutura do sistema, um modelo de controle e um modelo de decomposição de sub-sistemas
- Sistemas grandes raramente seguem um único modelo de arquitetura
- Modelos de decomposição de sistemas incluem modelos de repositório, modelos cliente-servidor e modelos de máquina abstrata
- Modelos de controle incluem controle centralizado e modelos orientados a eventos



# Pontos principais

- Modelos de decomposição modular incluem modelos de fluxo de dados e modelos de objeto
- Modelos de arquitetura de domínio específico são abstrações sobre um domínio de aplicação. Eles podem ser construídos pela abstração de sistemas existentes ou podem ser modelos de referência idealizados