Estilos Arquiteturais

Estilos Arquiteturais

- A arquitetura de um sistema pode aderir a um ou mais estilos arquiteturais
  - Um estilo define os tipos de elementos que podem aparecer em uma arquitetura e as regras que regem a sua interconexão
- Esses estilos pode simplificar o problema de definição de arquiteturas de sistema.
- A maioria dos sistemas de grande porte adere a vários estilos
- Estilos arquiteturais = "modelos arquiteturais"

Exemplos de Estilos Arquiteturais

- Cliente-Servidor
- "Em camadas"
- Filtros e dutos (*pipes and filters*)
- *Baseado em repositório*
- *Orientado a eventos (publisher/subscriber)*
- *Transferência Representacional de Estado (REST)*
- *Objetos distribuídos*
- *C2*

Estilos Arquiteturais e Escolhas de Projeto

- Um estilo arquitetural representa um conjunto de escolhas de projeto
  - Conjunto de características comuns a diversos sistemas nos quais as mesmas escolhas foram feitas
    - Padrões arquiteturais
  - Um sistema aderente a determinado estilo "ganha" as características inerentes a ele
- Estilos podem ser usados para descrever uma determinada arquitetura
  - Foco nas soluções de projeto e não em sua documentação

Organização de sistema

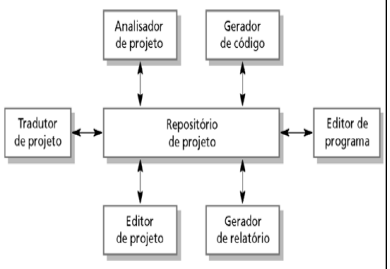
- Reflete a estratégia básica que é usada para estruturar um sistema
- Exemplos:
  - O estilo de repositório de dados compartilhados;
  - Estilo de serviços e servidores compartilhados;
  - Estilo de máquina abstrata ou em camadas
  - Orientado a objetos (ou Objetos Distribuídos)
  - *Pipes and Filters ou Pipelining*
- *Classificação para fins de estudo*

Estilo de repositório

- Sistemas cujas partes precisam trocar dados com frequência:
  - Dados compartilhados podem ser mantidos em um banco de dados central e acessados por todos os subsistemas
  - Cada subsistema mantém seu próprio banco de dados e passa dados para outros subsistemas
    - Podem usar uma abstração de repositório centralizado
    - Implementação distribuída

Arquitetura de conjunto de ferramentas CASE

Figura 11.2
Arquitetura de um conjunto de ferramentas CASE integradas.



Características do Estilo Arquitetural de Repositório

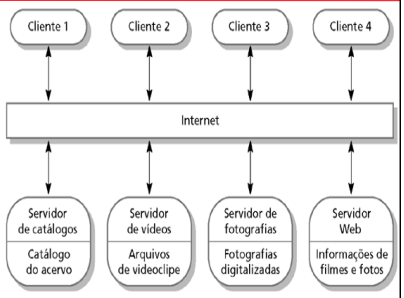
- Vantagens
  - É uma maneira eficiente de compartilhar grandes quantidades de dados
  - Dados aderem a uma representação comum
  - Simplifica a projeto de aplicações fortemente baseadas em dados
    - Tanto para troca de info. quanto para armazenamento
- Desvantagens
  - Os subsistemas devem estar de acordo com um modelo de dados padronizado
  - A evolução de dados é difícil e dispendiosa;
  - Dificuldade para distribuir de forma eficiente.

Estilo Cliente-Servidor

- Mostra como dados e processamento são distribuídos por uma variedade de componentes.
- Servidores independentes que fornecem serviços tais como impressão, transferência de arquivos, gerenciamento de dados, etc.
- Clientes utilizam esses serviços.
- Clientes e servidores normalmente se comunicam através de uma rede
  - Diversas tecnologias de comunicação são possíveis

Biblioteca de filmes e fotografias

Figura 11.3
Arquitetura de um sistema de acervo de filmes e fotografias.



Características do Estilo Cliente-Servidor

- Vantagens
  - Separação de interesses
  - Inerentemente distribuído
    - Balanceamento de carga, tolerância a falhas
  - É fácil adicionar novos servidores ou atualizar servidores existentes.
- Desvantagens
  - Gerenciamento redundante em cada servidor;
  - Nenhum registro central de nomes e serviços – pode ser difícil descobrir quais servidores e serviços estão disponíveis
  - Requisições e respostas casadas

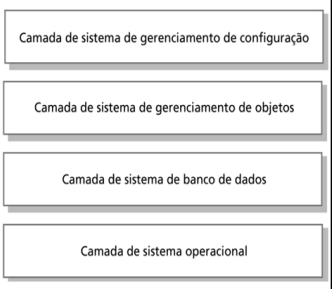
Modelo de Máquina Abstrata (Em Camadas)

- Organiza o sistema em um conjunto de camadas (ou máquinas abstratas)
  - Cada uma fornece um conjunto de serviços
  - Cada camada é cliente da camada subjacente
- Generalização do estilo Cliente-Servidor
  - Não precisa ser distribuído
- Apóia o desenvolvimento incremental dos subsistemas em camadas diferentes.
  - Ex. Se mudarmos a camada de negócios, só as camadas acima precisam ser modificadas

Sistema de gerenciamento de versões

Figura 11.4

Modelo em camadas de um sistema de gerenciamento de versões.



Características do Estilo em Camadas

- Vantagens
  - Facilidade de compreensão
  - Facilidade de manutenção
  - Desenvolvimento independente
- Desvantagens
  - Duplicação de funcionalidade
  - Às vezes é difícil estruturar um sistema através de camadas
    - É comum que a estruturação seja violada
  - *Overhead* de implementação e desempenho

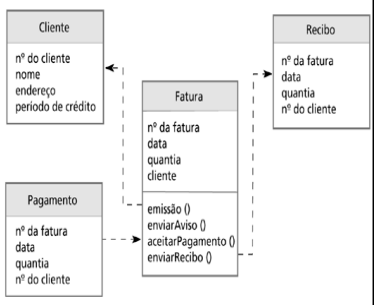
Estilo Arquitetural de Objetos

- Sistema como um conjunto de objetos fracamente acoplados e com interfaces bem definidas
  - Cada objeto oferece um conjunto de serviços
- No nível arquitetural, é frequentemente empregado na construção de sistemas distribuídos
  - Objetos distribuídos
- Uma implementação OO não implica em uma arquitetura OO

Sistema de processamento de faturas

Figura 11.5

Modelo de objeto de um sistema de processamento de faturas.



Características do Estilo Arquitetural de Objetos

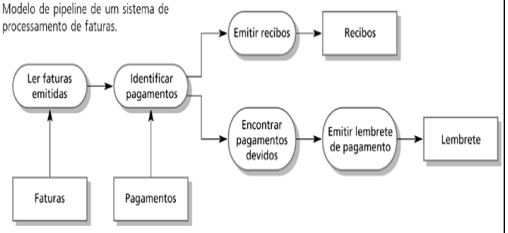
- Vantagens
  - Objetos são fracamente acoplados devido ao uso de interfaces
  - Linguagens de implementação orientada a objeto são amplamente usadas.
- Desvantagens
  - Mudanças de interface têm alto impacto
  - Não envolve restrições topológicas, o que pode dificultar a manutenção
    - Dependências entre objetos não são limitadas

Estilo Dutos e Filtros (*Pipelining*)

- Originário de sistemas operacionais UNIX e do projeto de compiladores
- Transformações funcionais processam entradas para produzir saídas.
  - Componentes são chamados de filtros
  - Conectores são dutos (*pipes)*
- Útil para aplicações de processamento de informação que interagem pouco com usuários
- Variação distribuída: processamento de *streams*

Sistema de processamento de faturas

Figura 11.6 Modelo de pipeline de um sistema de processamento de faturas.



Características do Estilo Dutos e Filtros

- Vantagens
  - Apóia reuso de transformações.
  - É fácil adicionar novas transformações.
  - É relativamente simples implementar como sistema concorrente ou seqüencial.
- Desvantagens
  - Requer um formato comum para a transferência de dados ao longo do *pipeline*
  - Não é apropriado para aplicações interativas
    - Mais especificamente: só é apropriado para realizar processamento sequencial

Fluxo de Controle

- Estilos arquiteturais relacionados com o fluxo de controle entre os componentes arquiteturais
- Controle centralizado
  - Um subsistema tem responsabilidade global pelo controle e inicia e pára outros sistemas
- Controle baseado em eventos
  - Cada componente responde a eventos gerados por outros subsistemas

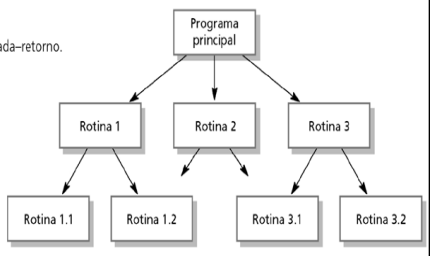
Controle centralizado

- Um componente é responsável pelo gerenciamento da execução de outros componente.
- O estilo Chamada-Retorno
  - Controle se inicia no topo de uma hierarquia de subrotinas e move-se para baixo na hierarquia.
  - Pode ser sequencial ou concorrente
- O estilo de Gerenciador
  - Aplicável a sistemas concorrentes e de tempo real
  - Um componente controla a parada, o início e a coordenação de outros processos de sistema

Chamada-Retorno

Figura 11.7

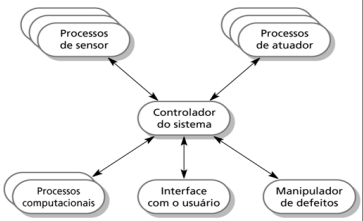
Modelo de controle chamada–retorno.



Gerenciador para um Sistema Tempo Real

Figura 11.8

Modelo de controle centralizado para um sistema de tempo real.



Comunicação entre o Controlador e os outros componentes pode ser baseada em eventos, chamadas de procedimentos, etc.

Sistemas orientados a eventos

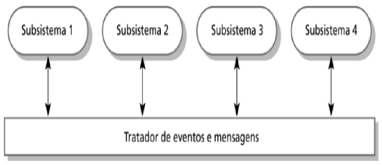
- Dirigidos por eventos gerados externamente
  - O *timing* dos eventos está fora do controle dos componentes que os processam
- Estilo *Publisher/Subscriber*
  - Eventos são transmitidos a todos os componentes.
  - Qualquer componente interessado pode respondê-los
- Estilo Orientado a Interrupções
  - Usado em sistemas de tempo real
  - Interrupções são detectadas por tratadores e passadas por outro componente para processamento.

Modelo Publisher/Subscriber

- É efetivo na integração de componentes em computadores diferentes em uma rede
  - Desacoplamento espacial e temporal
  - Componentes não sabem se um evento será tratado e nem quando será.
- Alguns componentes (*publishers*) publicam eventos
- Componentes (*subscribers*) registram interesse em eventos específicos e podem tratá-los
- A política de controle não é embutida no tratador de eventos e mensagens

Publisher/Subscriber

Figura 11.9
Modelo de controle baseado em broadcast seletivo.

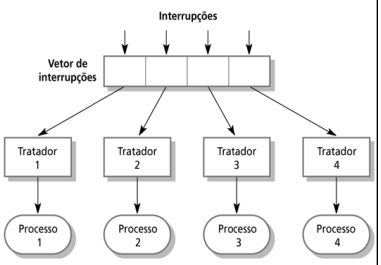


Estilo Orientado a Interrupções

- Usado em sistemas de tempo real onde a resposta rápida para um evento é essencial
- Existem tipos de interrupções conhecidos
  - Um tratador definido para cada tipo
- Cada tipo é associado a uma localização da memória
  - Uma chave de hardware causa a transferência de controle para um tratador.
- Permite respostas rápidas, mas é complexo para programar e difícil de validar.

Controle dirigido a interrupções

Figura 11.10
Modelo de controle orientado a interrupções.

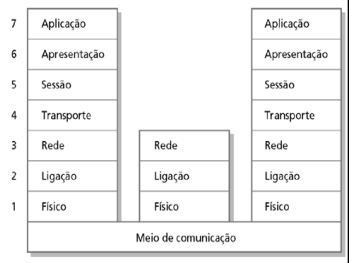


Arquiteturas de Referência

- Derivadas de um estudo de domínio de aplicação, ao invés de sistemas existentes.
- Podem ser usadas como base para a implementação de sistemas ou comparação de sistemas diferentes.
  - Atua como um padrão com relação ao qual os sistemas podem ser avaliados.
- Exs.
  - Modelo OSI para sistemas de comunicação
  - Organização tradicional de compiladores em vanguarda e retaguarda (e seus elementos internos)

Modelo de referência OSI

Figura 11.11
Arquitetura de modelo de referência OSI.



Arquitetura de um Compilador