

Projeto de Software Arquitetura

Eduardo Figueiredo

<http://www.dcc.ufmg.br/~figueiredo>

dcc603@gmail.com

02 Abril 2012

Agenda da Aula

- Arquitetura de Software
- Padrões arquiteturais
 - Arquitetura em Camadas
 - Dutos e Filtros
 - Repositório Compartilhado
 - Cliente-Servidor
 - Modelo-Visão-Controlle
 - Microkernel

Atividades Comuns em Processos

1. Especificação de requisitos
2. Projeto de Software
3. Implementação
4. Validação do software
5. Evolução de software

Onde estamos

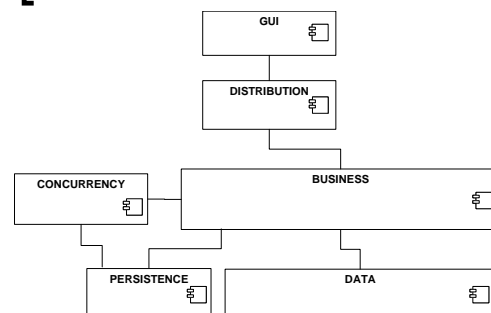
1. Especificação de requisitos
2. **Projeto de Software**
3. Implementação
4. Validação do software
5. Evolução de software

Modelagem de Software

Dividido em duas etapas

- **Projeto de Arquitetura** define a estrutura modular do software, as interfaces e as estruturas de dados utilizadas
- **Projeto Detalhado** descreve detalhadamente cada módulo definido no projeto preliminar

Projeto Arquitetural de Software



[Uma arquitetura pode afetar...]

- O desempenho do sistema
 - Exemplo: se incluir gargalos de comunicação
- A facilidade de distribuição
 - Sistemas complexos executam em várias máquinas
- A facilidade de manutenção
 - Componentes devem ser substituíveis

[Requisitos Não Funcionais]

- A arquitetura deve considerar requisitos não funcionais
 - **Desempenho**: evitar comunicação excessiva entre componentes distribuídos
 - **Segurança**: ocultar características críticas de segurança em camadas mais internas
 - **Disponibilidade**: incluir componentes redundantes e tolerância à falhas, etc.

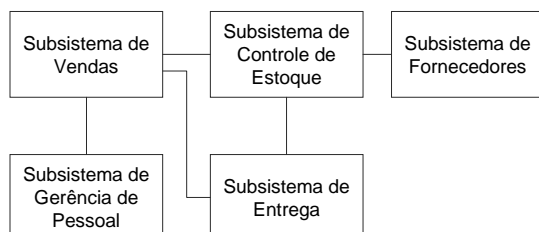
[Decisões de projeto arquitetural]

- Questões precisam ser respondidas
 - Como o sistema será distribuído em diferentes máquinas?
 - Como as funcionalidades serão decompostas em componentes?
 - Como avaliar se a arquitetura atende aos requisitos não funcionais?
 - Existe uma arquitetura genérica que possa ser usada?

[Diagrama de Componentes]

- Identifica os principais subsistemas que serão desenvolvidos
- Notação simplificada
 - Caixas representam os subsistemas (componentes)
 - Linhas representam alguma comunicação entre os subsistemas

[Exemplo de Arquitetura]



[Quando usar o diagrama?]

- Pontos positivos
 - Facilita comunicação com *stakeholders*
 - Úteis para planejamento de projeto
 - Facilita o desenvolvimento em paralelo
- Pontos negativos
 - Não mostra a natureza dos relacionamentos entre componentes
 - Não indica as propriedades internas dos subsistemas

Padrões Arquiteturais

Definição de Padrões

- Um padrão é uma descrição do problema e a essência da sua solução
- Documenta boas soluções para problemas recorrentes
- Deve ser suficientemente abstrato para ser reusado em aplicações diferentes

Arquiteturas de Referência

- Embora cada sistema seja único
 - É comum que sistemas do mesmo domínio tenham arquiteturas similares
- Um projeto pode ser baseado em um padrão arquitetural conhecido

Padrões Arquiteturais

- Padrões arquiteturais expressam formas de organizar a estrutura fundamental do sistema
 - Permitem a construção de uma arquitetura aderente a certas propriedades
- O conhecimento de padrões arquiteturais pode ajudar na definição da arquitetura do sistema

Elementos de um Padrão

- A escolha do padrão arquitetural pode ser o primeiro passo para a solução
- Padrões arquiteturais definem
 - Um conjunto de sub-sistemas
 - As responsabilidades de cada sub-sistema
 - Regras de relacionamentos entre os sub-sistemas

Da arquitetura a implementação

- Padrões arquiteturais representam os padrões mais abstratos
 - Padrões de projeto (Modelagem)
 - Idiomas (Programação)
- Os padrões arquiteturais definem a estrutura fundamental
 - Atividades subsequentes devem seguir esta estrutura

Padrões são abstratos

- Os padrões não definem completamente a arquitetura do sistema
 - Padrões são *templates* que devem ser refinados
- Exemplos de refinamentos
 - Incluir outros componentes e novos relacionamentos
 - Definir padrões de projeto e idiomas em fases subsequentes

Escolha do Padrão

- A escolha do padrão arquitetural deve estar associada ao tipo de sistema e seus requisitos não funcionais
- Algumas perguntas podem ajudar
 - O sistema é interativo?
 - Possui muitas variações?
 - Que requisitos não funcionais são importantes? Confiabilidade? Adaptabilidade?

Composição de Padrões

- Padrões diferentes levam a consequências diferentes
 - Mesmo quando os padrões abordam problemas semelhantes
- Assim como ocorre em padrões de projeto, sistemas complexos possuem mais de um padrão arquitetural

Exemplos de Padrões Arquiteturais

Padrões Arquiteturais

- Da desordem a estrutura
 - Layered Architecture
 - Pipes and Filters
 - Blackboard
- Sistemas distribuídos
 - Client-Server
 - Broker
- Sistemas interativos
 - Model-View-Controller (MVC)
 - Presentation-Abstraction-Control
- Sistemas adaptáveis
 - Microkernel
 - Reflection

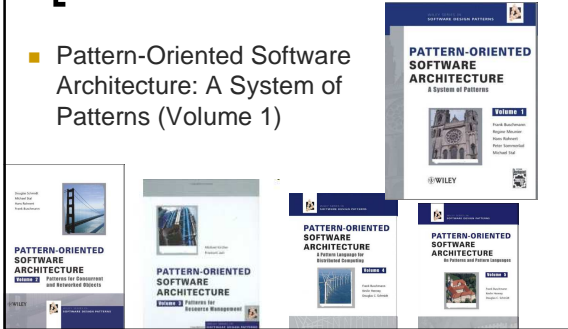
Padrões Arquiteturais

- Da desordem a estrutura
 - Layered Architecture (Arquitetura em Camadas)
 - Blackboard (Repositório Compartilhado)
 - Pipes and Filters (Dutos e Filtros)
- Sistemas distribuídos
 - Client-Server (Cliente-Servidor)
 - Broker
- Sistemas interativos
 - Model-View-Controller (MVC)
 - Presentation-Abstraction-Control
- Sistemas adaptáveis
 - Microkernel
 - Reflection

Discutidos no livro do Sommerville

Outros Padrões Arquiteturais

- Pattern-Oriented Software Architecture: A System of Patterns (Volume 1)

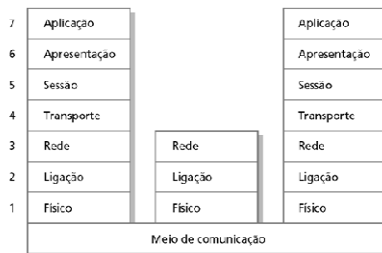


Arquitetura em Camadas

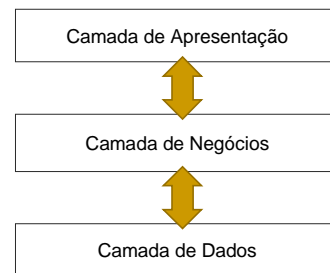
- Organiza o sistema em um conjunto de camadas
 - Cada camada oferece um conjunto de serviços
- Uma camada somente
 - Solicita serviços da camada inferior
 - Fornece serviços para a camada superior

Exemplo 1: Protocolos OSI

Modelo de camadas para sistemas de comunicação



Exemplo 2: Três Camadas



Vantagens

- Favorece o modelo de desenvolvimento incremental
- As camadas podem ser facilmente substituídas por equivalentes
 - Requer interfaces estáveis
- Mudanças em uma camada (teoricamente) só impacta a camada superior
- Camadas superiores podem ser independentes de plataforma/hardware

Desvantagens

- Estruturar o sistema em camadas não é trivial
 - Pode ser difícil identificar quais os serviços elementares das camadas inferiores
- Muitas camadas podem comprometer o desempenho do sistema
 - A requisição tem que trafegar pelas várias camadas até ser atendida

Dutos e Filtros

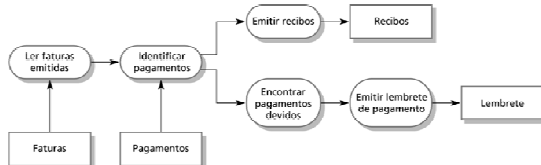
- Padrão de organização da dinâmica de um sistema
- Dois papéis principais
 - **Dutos:** componentes que conduzem ou distribuem os dados
 - **Filtros:** componentes que transformam os dados
- Usado principalmente em aplicações de processamento de dados

Dinâmica do Padrão

- Os dados de entrada se movem pelos dutos, sendo transformados pelos filtros, até serem convertidos em dados de saída
 - As transformações podem ocorrer em sequência ou em paralelo

Exemplo de Dutos e Filtros

- Entradas: Faturas e Pagamentos
- Saídas: Recibos e Lembretes



Vantagens

- O módulo de transformação (filtro) é bem modular
 - Facilmente reusável e substituível
- O estilo de workflow é aderente a muitos processos de negócios
- É simples evoluir o sistema pela adição de filtros
- Se aplica tanto a sistemas sequenciais quanto a sistemas concorrentes

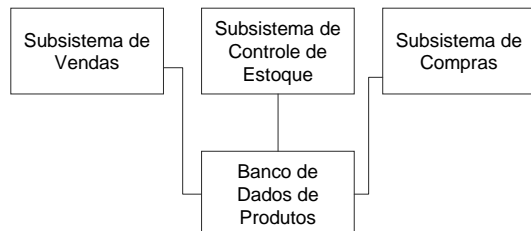
Desvantagens

- O formato dos dados trafegados devem ser acordados entre os módulos
- Pode haver um overhead causado pela padronização dos dados
- Incompatibilidade no formato dos dados pode dificultar o reuso de filtros

Repositório Compartilhado

- Também conhecido como *Blackboard*
- Os subsistemas manipulam a mesma base de dados
 - Um (ou mais) subsistema gera os dados
 - Vários subsistemas lêem os dados
- Adotado principalmente quando grandes quantidades de dados são compartilhadas

[Exemplo de Repositório]



[Vantagens]

- Maneira eficiente de compartilhar dados
- Backup é centralizado (mais fácil)
- Formas de proteção dos dados podem ser implementadas
- Os subsistemas que gravam dados não necessitam saber quem os usa
- Fácil integrar novos subsistemas

[Desvantagens]

- Os subsistemas devem entender o formato dos dados gravados
- Manter e evoluir grandes volumes de dados pode ser difícil / caro
- Subsistemas diferentes podem ter requisitos diferentes
 - Mais segurança ou maior disponibilidade
- Dificuldade para distribuir os dados
 - Dados redundantes ou inconsistentes

[Arquitetura Cliente-Servidor]

- Organizada como um conjunto de serviços
 - Servidores dos serviços
 - Clientes dos serviços
- Exemplos de servidores
 - Servidor de impressão
 - Servidor de arquivos, etc.

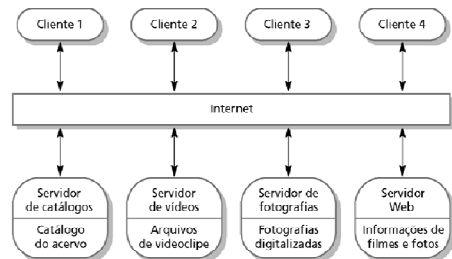
[Arquitetura Cliente-Servidor]

- Requer uma estrutura de rede para clientes acessarem os serviços
- Clientes sabem quais os serviços e servidores estão disponíveis
- Servidores não sabem quem são os clientes

[Arquitetura Cliente-Servidor]

- Essencialmente, é usado o protocolo *request-reply*
 1. Um cliente faz um pedido ao servidor e espera pela resposta
 2. O servidor executa o serviço e responde ao cliente

Exemplo de Cliente-Servidor



Vantagens

- A distribuição de dados é fácil e direta
- Faz uso efetivo dos recursos em rede
 - Possibilita hardware mais barato
- É fácil adicionar novos servidores ou atualizar servidores existentes

Desvantagens

- Não prevê modelo de dados compartilhado
 - Subsistemas usam diferentes organizações de dados
- Pode haver redundância de serviços em diferentes servidores
- Não prevê registro central de serviços
 - Difícil descobrir quais serviços estão disponíveis

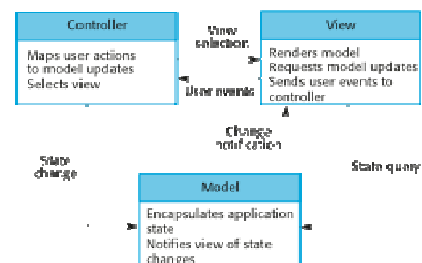
Padrão MVC

- Largamente utilizado em aplicações interativas Web
- Organiza o sistema em três componentes
 - **Modelo**: contém as funcionalidades e dados principais
 - **Visão**: responsável por apresentar os dados ao usuário
 - **Controlador**: trata os eventos de entrada

Descrição do MVC

- Separa a apresentação e a interação dos dados do sistema
 - Os três componentes tem responsabilidades distintas mas interagem entre si
- Quando é recomendado?
 - Quando existem várias maneiras de visualizar e interagir com os dados
 - São desconhecidos (ou são voláteis) os requisitos de interação com os dados

Representação dos Papéis



Vantagens

- Permite que os dados sejam alterados de forma independente de sua representação (e vice versa)
 - Apóia a apresentação dos mesmos dados de maneiras diferentes
- Facilita a distribuição do componente de visão
 - Os dados são mantidos centralizados e protegidos

Desvantagens

- Complexidade excessiva quando o modelo de dados e de interações é muito simples
 - A estrutura do padrão pode impor código adicional desnecessário

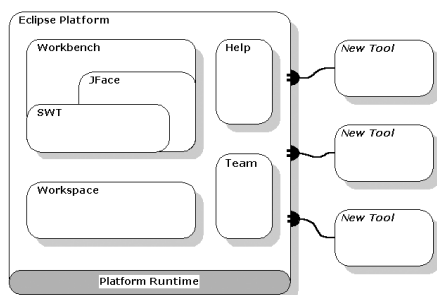
Microkernel

- Destinado a domínios de sistemas que possuem requisitos muito voláteis
 - Sistemas precisam ser capazes de se adaptar aos requisitos voláteis
- Este padrão de arquitetura separa a funcionalidade mínima em um núcleo
 - Novas funcionalidades são agregadas por extensões na forma de plug-in

Responsabilidades

- Microkernel
 - Provê as funcionalidades básicas
 - Oferece o meio de comunicação entre as extensões
 - Gerencia os recursos
- Extensões
 - Inclui novas funcionalidades
 - Fornece uma interface para interação com o microkernel e com outras extensões

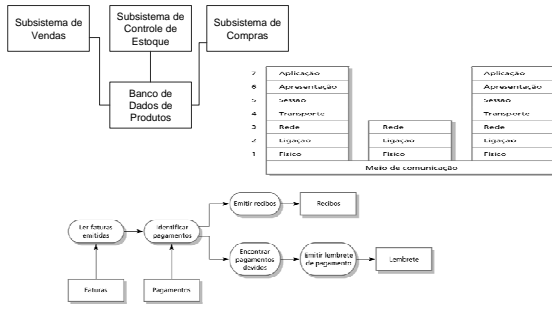
Exemplo de Microkernel



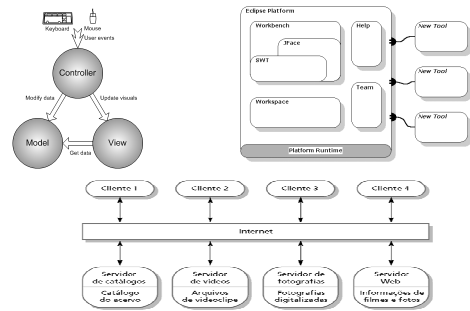
Resumo da Aula

- Arquitetura de software
 - Primeiras decisões no domínio da solução
 - Favorece o reuso em larga escala
- Padrões arquiteturais
 - Restringe as próximas fases de desenvolvimento
 - A escolha depende de características do sistema

Resumo dos Padrões



Resumo dos Padrões



Referências da Aula

- Ian Sommerville. **Engenharia de Software**, 9a. Edição. 2011.
 - Seção 6.1 Decisões de projeto de arquitetura
 - Seção 6.2 Visões da arquitetura
 - Seção 6.3 Padrões de arquitetura
- F. Buschmann et al. **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons, 1996.
 - Cap. 2 Architectural Patterns

Próxima Aula

- Laboratórios 2011 e 2012
- Tema:
 - Diagrama de Casos de Uso