Resumo de arquitetura de software Pt - II

Estilos Arquiteturais

Como definir o esstilo?

- Componentes -> blocos de construção de um sistema (partes do sftw, ou provedores)
- Serviços -> providos pelo componentes para os atores
- Estilo refere ao formato geral do sistema

Técnicas Auxiliares para a Definição da Arquitetura de Software

1. **Abstração**
   - Habilidade de extrair o que é comum e geral à dadas entidades
2. **Encapsulamento**
   - Agrupar informações para preservar as fronteiras da abstração
3. **Ocultação de Informação**
   - Esconder as informações que um cliente não precisa saber
4. **Modularização**
   - Gerência da complexidade quebrando o sistema em partes
5. ***Separation of Concerns (Separação de preocupações)***
   - Responsabilidades não relacionadas precisam ficar separadas
6. **Acoplamento e Coesão**
   - Acoplamento (-): como os diferentes módulos se relacionam
   - Coesão (+): a medida de quanto um módulo é autossuficiente

7. **Suficiência e Completude**
   - Componentes possuem características suficientes para uma interação útil e eficiente com os demais
8. **Separação das Políticas e Implementação**
   - Implementações não se prendem ao contexto → (+) reuso
9. **Separação das Interfaces da Implementação**
   - Clientes acessam interfaces separadas da implementação
10. **Único Ponto de Referência**
    - Definição única dos itens da arquitetura de software
11. **Dividir para Conquistar**
    - Dividir o problema em partes menores, simplificando a solução

---

Definição de arquitetura?

- Passo 1:
  - componente a ser refinado (Sistema completo)
- Passo 2:
  - identificar os requisitos do componente e os requisitos para as suas interações
- Passo 3:

    - Pesquise por um estilo arquitetural
- Passo 4:
    - Aplica o padrao

Pipers and filters?

- Cada componente possui uma serie de entradas e uma serie de saídas
- Fluxo de entrada
- Filtros devem ser entidades independentes
- Nao devem ter conhecimento do seu antecessor e do sucessor
- Se restringir ao que é recebido nos dutos de entrada e assegurar o que aparecerá nos dutos de saída

Vantagens

1. Permitem pensar um sistema complexo como uma composição de “filtros” específicos
2. Útil para aplicações de processamento de informação que interagem pouco com os usuários
3. Suportam o reuso – combinação de “filtros”
4. É fácil de adicionar, recombinar ou trocar (flexibilidade)
5. Permite algumas análises especializadas
    - Tais como: *throughput* e análise de *deadlock*
6. Suportam execução concorrente (vários filtros em paralelo)
7. Processamento eficiente

Desvantagens

1. Podem levar a uma organização do processamento em lote
2. Como os filtros são intrinsecamente independentes, são orientados a realizar uma transformação completa da entrada na saída
    - Não são adequados para sistemas interativos
3. Dificuldade de manter correspondência entre dois fluxos distintos, mas relacionados
4. Pode ter trabalho adicional para ajustar os dados às necessidades individuais de cada fluxo
5. Não existe compartilhamento de dados e gerenciamento de erros

Exemplo – Shell do Linux

```
$ ls | grep b | sort -r | tee arquivo.out | wc –l
```

No exemplo acima o comando "ls" lista o conteúdo do diretório, no entanto devido ao Pipe ele não envia o resultado para tela e sim para o comando "greb b", que por sua vez filtra os nomes de arquivos que contém a letra "b". O segundo Pipe envia a saída do comando "grep b" para "sort -r", que classifica os nomes em ordem crescente. A saída do comando "sort -r" é passada para o comando "tee", que divide os dados em dois, como uma conexão em T, fazendo com que as informações processadas por "sort -r" sejam escritas no arquivo "arquivo.out", e por fim o comando "wc -l" conta as linhas do arquivo "arquivo.out". Portanto, o resultado será a quantidade de arquivos que contém a letra b impressa na tela e o nomes desses arquivos em "arquivo.out".

Abstração de dados e organização a OO?

- Dados e operações são encapsuladas em um tipo abstrato de dados (objeto)
- Objetos interagem uns com os outros
-

Observações

- Ampla adoção e vantagens conhecidas

- Vantagens:
  - Objetos podem processar tarefas concorrentes
  - Objetos possuem múltiplas interfaces e são fracamente acoplados
  - Linguagens orientadas a objetos são amplamente usadas.
- Desvantagens:
  - Para que os objetos interajam, é preciso que ambos se conheça, o que gera dependência
  - A alteração de um objeto pode afetar todos os demais que dependem dele – pode gerar efeito em cascata

---

Baseado em eventos?

- Componentes interagem através da invocação explícita de métodos
- Componente registra interresse em um evento

- ## Vantagens e Desvantagens

  - Vantagens:
    - Forte suporte ao reuso
    - Facilita a evolução do sistema

  - Desvantagens:
    - Um componente que anuncia um evento não tem controle (ordem/tempo) sobre a execução associada
    - A troca de dados precisa ocorrer também de forma indireta (repositório compartilhado)

Sistema em camadas

- Organizados hierarquicamente
- Cada camada provê serviços para a camada superior e consume da inferior

- # Vantagens/Desvantagens

  - Vantagens:
    - Possibilita trabalhar em níveis crescentes de abstração
    - Suportam facilmente a aplicação de melhoramentos
    - Suportam o reuso

  - Desvantagens:
    - Nem todos os sistemas se adequam a essa organização
    - Requisitos de performance podem levar a quebra das regras de organização das camadas

---

Repositorios

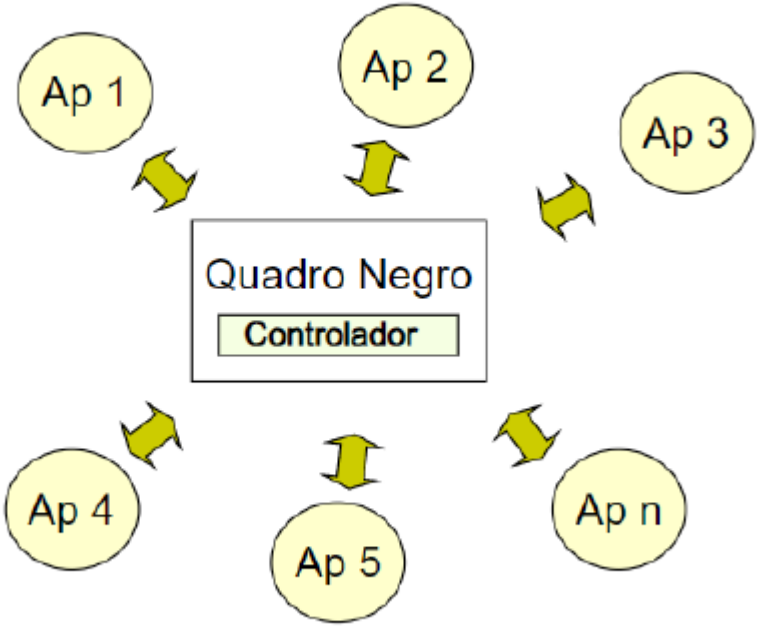
- Estrutura de dados central representado o estado

- # Componentes

1. ## Fontes de conhecimento
   - Fontes separadas e independentes – acessadas através do *blackboard* – cada qual resolvendo aspectos específicos do problema

2. ## A estrutura de dados blackboard
   - Armazena o estado do sistema (dados)

3. ## Controle
   - Monitora mudanças no *blackboard* e decide ações

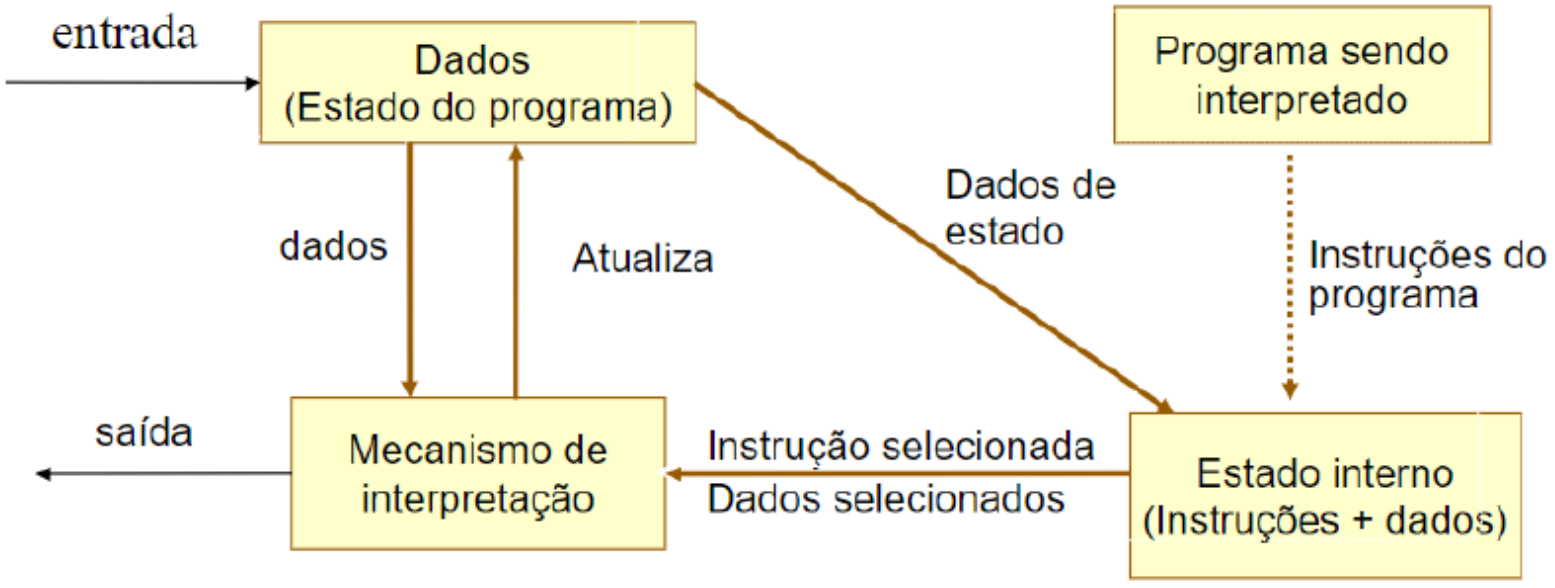

Ap 1
Ap 2
Ap 3
Quadro Negro
Controlador
Ap 4
Ap 5
Ap n

Vantagens

- Reúso de conhecimentos
- Suporta mudanças e manutenção
- Ajuda a resolver problemas de experimentação

- Desvantagens
  - Nenhuma boa solução é garantida
  - Dificuldade em estabelecer uma boa estratégia de controle
  - Baixa eficiência e alto esforço de desenvolvimento
  - Não suporta paralelismo

Interpretadores oritentados a tabela (Virtual machine)

- 

Cliente - Servidor

- Mostra como os dados e processamentos são distribuídos por uma variedade de componentes
- Se comunicam atraves de uma rede

- Cliente-Servidor

  - Vantagens:
    - Separação de interesses
    - Inerentemente distribuído: pode haver balanceamento de carga, tolerância a falhas
    - É fácil adicionar novos servidores ou atualizar servidores existentes
    - Escalabilidade: aumentando a capacidade computacional do servidor

- Cliente-Servidor

  - 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
    - Falhas no servidor

P2P

- Não ha distinção entre nos
- Cada no mantém seus proprios dados
- Cada no é cliente e servidor
- Vantagem:
  - Reduz falhas
- Desvantagem:

    - Aumenta tempo de consulta

Monilítica

- Funções de negócio implementadas em um unico processo
- Ao longo do tempo o sistema vai crescendo e ficando complexo
- Dificil manutenção

Microserviços:

- Desenvolver aplicação como um conjunto de pequenos serviços