UnB Departamento de Ciência da Computação

Arquitetura do Sistema

Nada que é visto, é visto de uma vez e por completo.
Euclides

Edison Ishikawa, D. Sc.

Introdução

- Objetivo
  - Apresentar métodos sistemáticos para obtenção do projeto de arquitetura do sistema (esquema preliminar por meio do qual o software é construído)

Sumário

- Introdução
- Desenvolvimento
- Referências

Introdução

- Sistemas OO são compostos de objetos que interagem entre si por meio do envio de mensagens com o objetivo de executar tarefas desse sistema.
- Sistemas também podem ser vistos como um conjunto de subsistemas que o compõem.
- A definição dos subsistemas é feita no **Projeto de Arquitetura**

Projeto de Arquitetura

- Define a forma com que o sistema se divide em partes e as interfaces entre elas
- Vantagens
  - Produz unidades menores de desenvolvimento
  - Maximiza o reuso no nível de subsistemas componentes
  - Ajuda a gerenciar a complexidade no desenvolvimento

Arquitetura de Software

- OMG
  - "É a estrutura organizacional do software. Uma arquitetura pode ser *recursivamente* decomposta em partes que interagem através de interfaces. Relacionamentos conectam as partes e restrições que se aplicam a grupos das partes."
- **Decisões tomadas para a definição da arquitetura de software influenciam diretamente na forma como um Sistema irá atender a seus requistos não-funcionais.**

Projeto de arquitetura

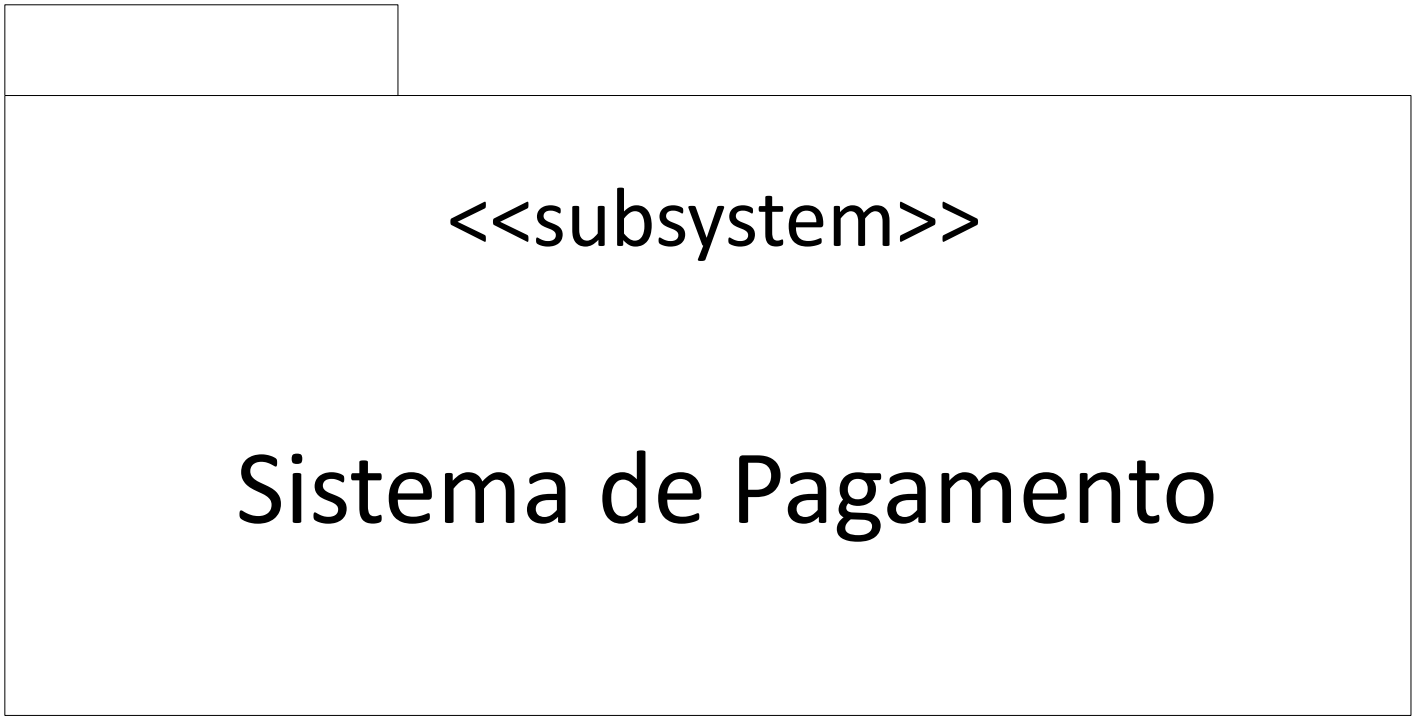
- Arquitetura lógica
  - Define como o sistema é decomposto em diversos subsistemas e como as suas classes são dispostas nestes subsistemas
- Arquitetura física
  - Define como os subsistemas são dispostos fisicamente quando o sistema tiver de ser implantado
    - i.e. , se houver diversos nós de processamento para o sistema ser executado, é importante definir em que nó cada subsistema estará posicionado e com que outros nós ele deve se comunicar
  - Tem a ver com distribuição física das partes do sistema (subsistemas, partições, camadas, sistemas distribuídos, concorrência, nós de processamento e componentes)

Arquitetura Lógica

- Definição
  - Organização de classes de um Sistema OO em subsistemas
    - Um subsistema provê serviços para outros por meio de sua interface
    - Cada subsistema provê ou utiliza serviços de utros subsistemas

Representação de subsistema em UML

- Representado por um pacote com o estereótipo <<subsystem>>



Arquitetura Lógica

- O fato de um subsistema utilizar os serviços fornecidos por outro é representado por um relacionamento de dependência (semelhante a dependência entre classes).
  - Uma classe A pode ter as seguintes relações de dependência em relação a uma classe B
    - Dependência por atributo
      - A possui um atributo cujo tipo é B (dependência estrutural)
    - Dependência por variável global
      - A utiliza uma variável global cujo tipo é B
    - Dependência por variável local
      - A possui alguma operação cuja implementação utiliza uma variável local do tipo B
    - Dependência por parâmetro
      - A possui pelo menos uma operação, e esta possui pelo menos um parâmetro, cujo tipo é B.

Arquitetura Lógica

- Dicas para alocar classes a subsistemas
  - 1. Comece pelo modelo de classes de domínio
    - Identificar as classes mais importantes
    - Para cada uma destas classes criar um subsistema
    - Outras classes menos importantes e relacionadas a uma classe considerada importante são posicionadas no subsistema da citada classe
    - Exemplo – Sistema de vendas pela Web
      - Classes importantes: Cliente, Pedido e Entrega
      - Outras classes: itemPedido fica no subsistema em que está Cliente, Transportadora no que está Entrega

Arquitetura Lógica

- Dicas para alocar classes a subsistemas
  - 2. Utilizar o princípio do baixo acoplamento entre subsistemas
    - Para minimizar a dependência (e o acoplamento), manter em um patamar mínimo possível a quantidade de associações entre classes que estão definidas em diferentes subsistemas

Arquitetura Lógica

- Dicas para alocar classes a subsistemas
  - 3. Utilizar o princípio de alta coesão dentro de cada subsistema
    - A dependência entre os elementos dentro de um sistema deve ser máxima, i.e., deve haver mais associações entre elementos dentro de um subsistema do que entre elementos pertencentes a subsistemas diferentes.

Arquitetura Lógica

- Dicas para alocar classes a subsistemas
  - 4. Evitar dependências cíclicas entre subsistemas
    - Uma dependência cíclica entre os subsistemas SS_1, SS_2 e SS_3 existe quando SS_1 depende de SS_2, que depende de SS_3, que depende de SS_1.
    - $SS_1 \rightarrow SS_2 \rightarrow SS_3 \rightarrow SS_1$

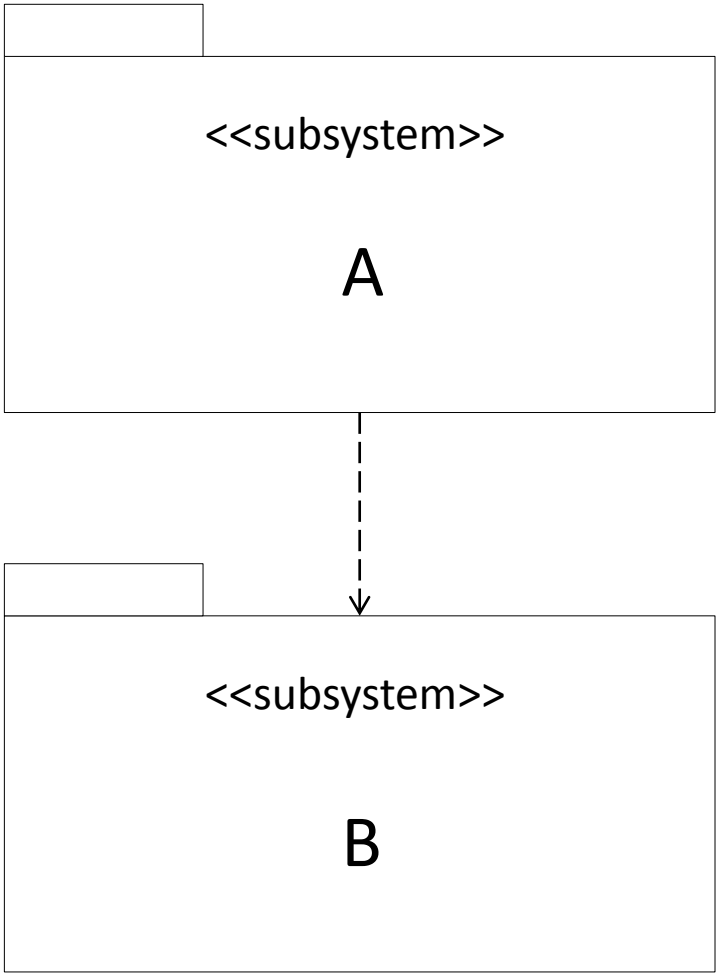
Arquitetura Lógica

- Dicas para alocar classes a subsistemas
  - 5. Um classe deve ser alocada em um único subsistema
    - O subsistema que define a classe deve mostrar todas as propriedades da mesma (nome, atributos, operações, etc...)
    - Outros subsistemas que fazem referência a essa classe podem utilizar a sua notação simplificada.
      - Isso evita que haja definições inconsistentes de uma mesma classe em diferentes subsistemas

Camadas de Software

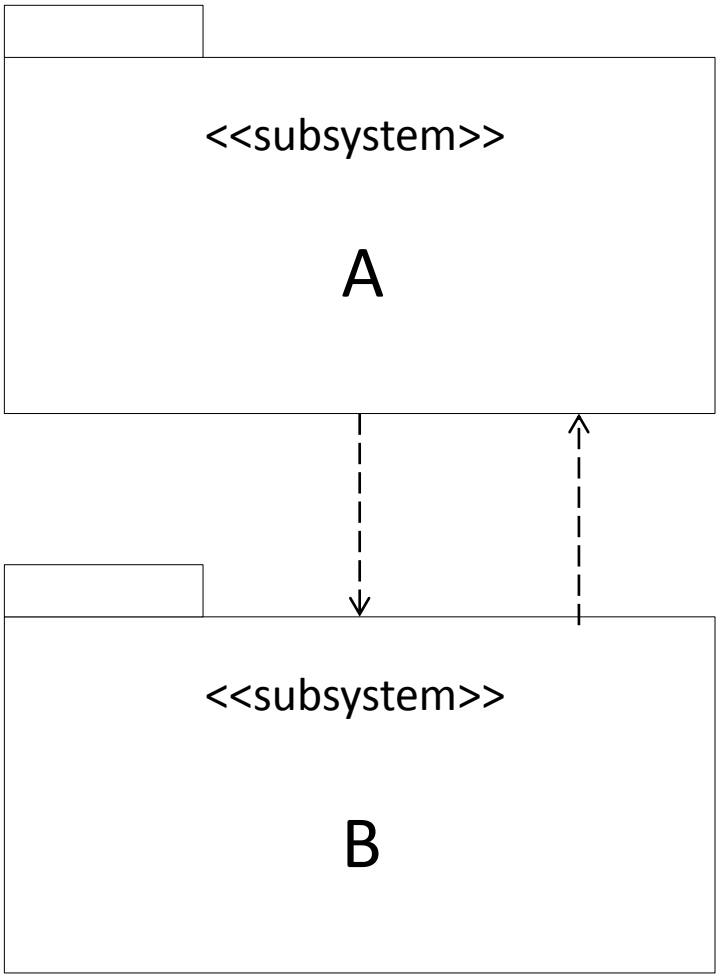
- Dois subsistemas interagem quando um precisa de serviços do outro
- Duas formas de interação
  - Cliente-servidor
  - Ponto-a-ponto
- Estas duas formas influenciam a forma pela qual esses sistemas são distribuídos fisicamente pelos nós de processamento

Arquitetura Cliente-Servidor



- Um subsistema assume o papel de servidor em relação ao outro subsistema, o cliente
- Chamamos de camadas os subsistemas envolvidos

Arquitetura Ponto-a-Ponto



- Os subsistemas assumem os papéis de cliente e de servidor simultaneamente

Camadas de Software

- Uma camada é uma coleção de unidades de software (tais como classes ou componentes) que podem ser executadas ou a cessadas.
- Camadas representam diferentes níveis de abstração.
- Dessa forma, Um SOO é representado por uma pilha de camada de software que se comunicam entre si

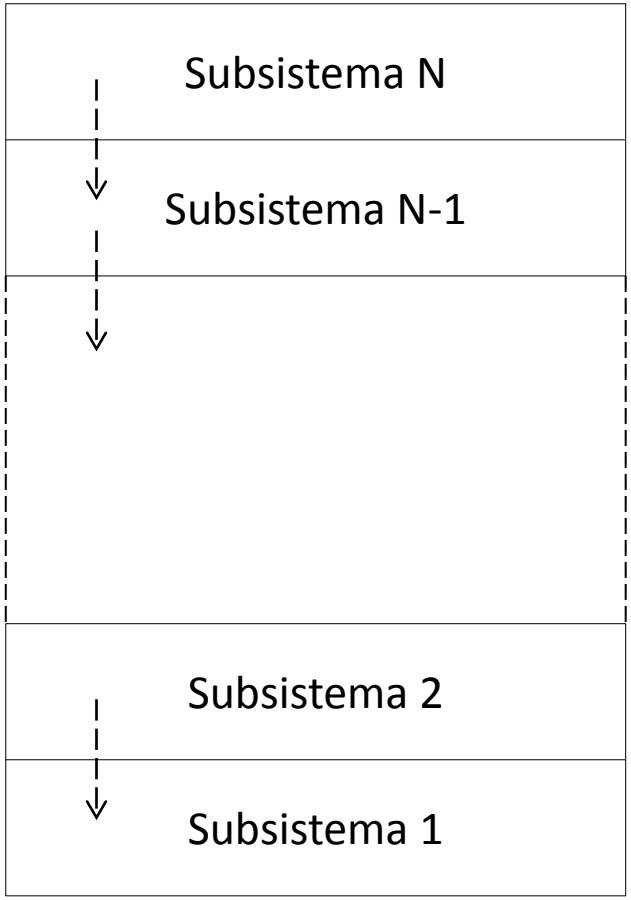
Camadas de Software

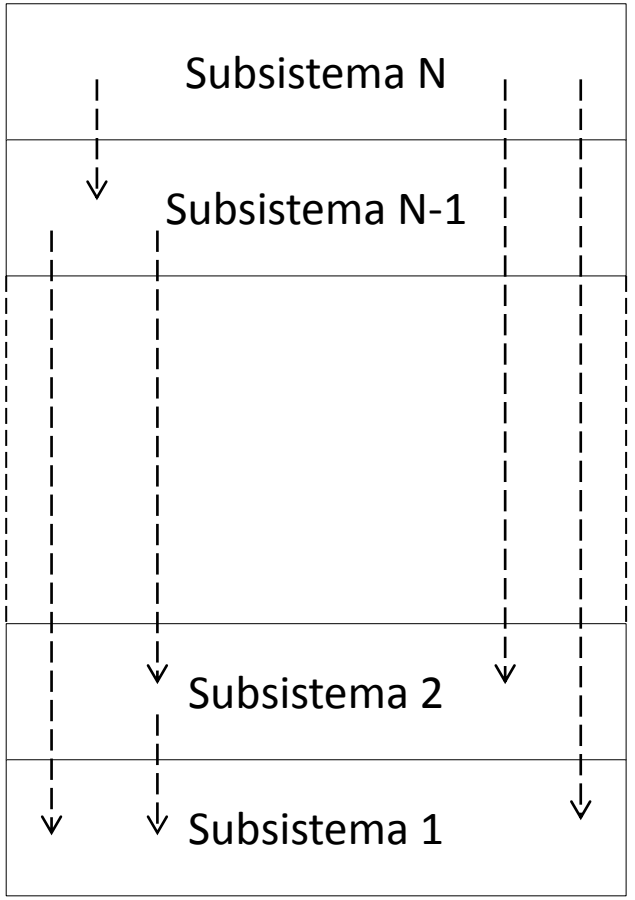
- Camadas inferiores representam serviços cada vez mais genéricos (que podem ser utilizados por diversos sistemas)
- Camadas superiores representam serviços cada vez mais especializados ao sistema em questão
- A divisão em camadas aumenta a portabilidade e a manutenibilidade (facilmente modificável)
  - Uma mudança em uma camada mais baixa (i.e., mais genérica) que não afete sua interface não implicará mudanças na camada mais alta (i.e., mais específica)
  - Uma mudança em uma camada mais alta que não implica na criação de um novo serviço na camada mais baixa não afetará esta última

Camadas de Software

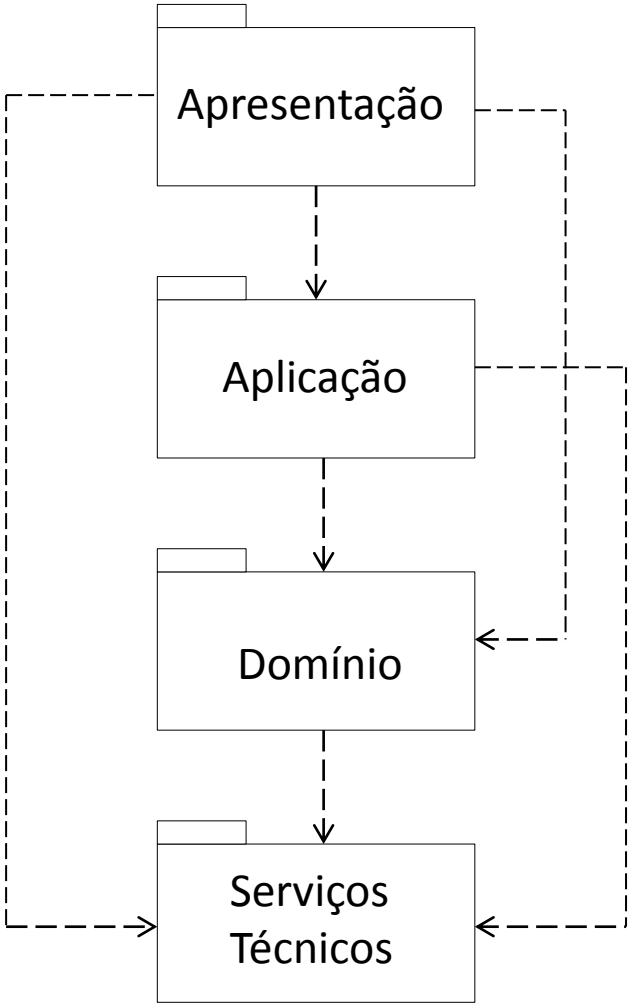
- Arquitetura fechada
  - Um componente de uma camada somente pode utilizar os serviços dos componentes de sua própria camada ou da camada imediatamente inferior
- Arquitetura aberta
  - Uma camada em certo nível pode utilizar os serviços de qualquer camada inferior
  - Também conhecida como relaxada ou transparente
  - É a mais utilizada na prática

Camadas de Software





Camada Lógicas em um App



Camada Lógicas em um App

- Camada de apresentação
  - Composta de classes que constituem a funcionalidade para visualização dos dados pelos usuários e interface com outros sistemas
  - Classes de fronteira se encontram nesta camada
  - Ex:
    - Sistema de menus baseados em texto
    - Interface gráfica construída em algum ambiente de programação
    - Interface de voz

Camada Lógicas em um App

- Camada de aplicação
  - Serve de intermediária entre os vários componentes da camada de apresentação e os objetos de negócio
  - Traduz as mensagens da camada de aplicação em mensagens compreendidas pelos objetos de domínio
  - Responsável pelo fluxo da aplicação e controla a navegação do usuário de uma janela a outra
  - Objetos de controle são alocados nesta camada
    - Direcionam as classes de negócio na realização das funcionalidades do sistema
    - Não contém regras de negócios, apenas delega tarefas para os objetos da camada de domínio
  - Conhecida também como camada de serviço

Camada Lógicas em um App

- Camada de domínio
  - É onde se encontram a maioria dos objetos da análise de domínio
  - Manipula requisitos da camada de aplicação
  - Objetos são normalmente independentes da aplicação
  - Camada responsável pela validação das regras de negócio, assim como da validação de dados provenientes da camada de apresentação (por meio da camada de aplicação)
  - Conhecida também por *camada da lógica do negócio*

Camada Lógicas em um App

- Camada de serviços técnicos
  - Local onde são encontrados serviços genéricos e úteis para uma gama bastante grande de aplicações.
  - Exemplos de serviços:
    - Segurança
    - Registro de operações do sistema (log)
    - Manipulação de arquivos
    - Estrutura de dados
    - Classes utilitárias
  - Contém classes cujo objetivo é permitir que o sistema se comunique com outros sistemas (SGBD ou Serviços Web)
  - Todas as camadas superiores são dependentes (requisitam serviços) desta camada

Implantação Física

- As definições da camada (layer) lógica servem como um guia para a alocação física dos subsistemas pelos nós de processamento existentes
  - A cada nó são alocadas uma ou mais camadas lógicas
- Vantagens
  - Divisão dos objetos permite um maior grau de manutenção e reutilização, uma vez que sistemas de software construídos em camadas podem ser mais facilmente estendidos
  - Também são mais adaptáveis a uma quantidade maior de usuários. Servidores novos ou mais potentes podem ser acrescentados para compensar um eventual crescimento no número de usuários do sistema
- Desvantagens
  - Pode diminuir o desempenho, uma vez que um evento precisa ser transmitido através das diversas camadas

Implantação Física

Sistemas em duas camadas – two tier

Camada de Apresentação

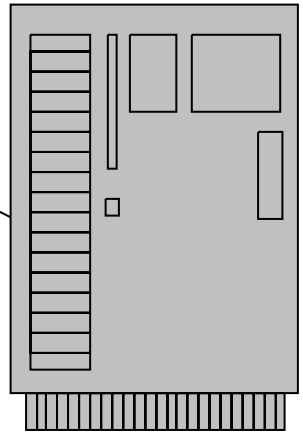
Máquina do Usuário

Thin Client

LAN

Demais Camadas

Repositório de Dados



Servidor

Implantação Física

Sistemas em duas camadas – two tier

Camada de Apresentação

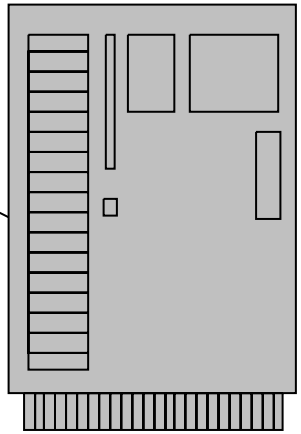
Demais Camadas

Máquina do Usuário

Fat/Rich Client

LAN

Repositório de Dados

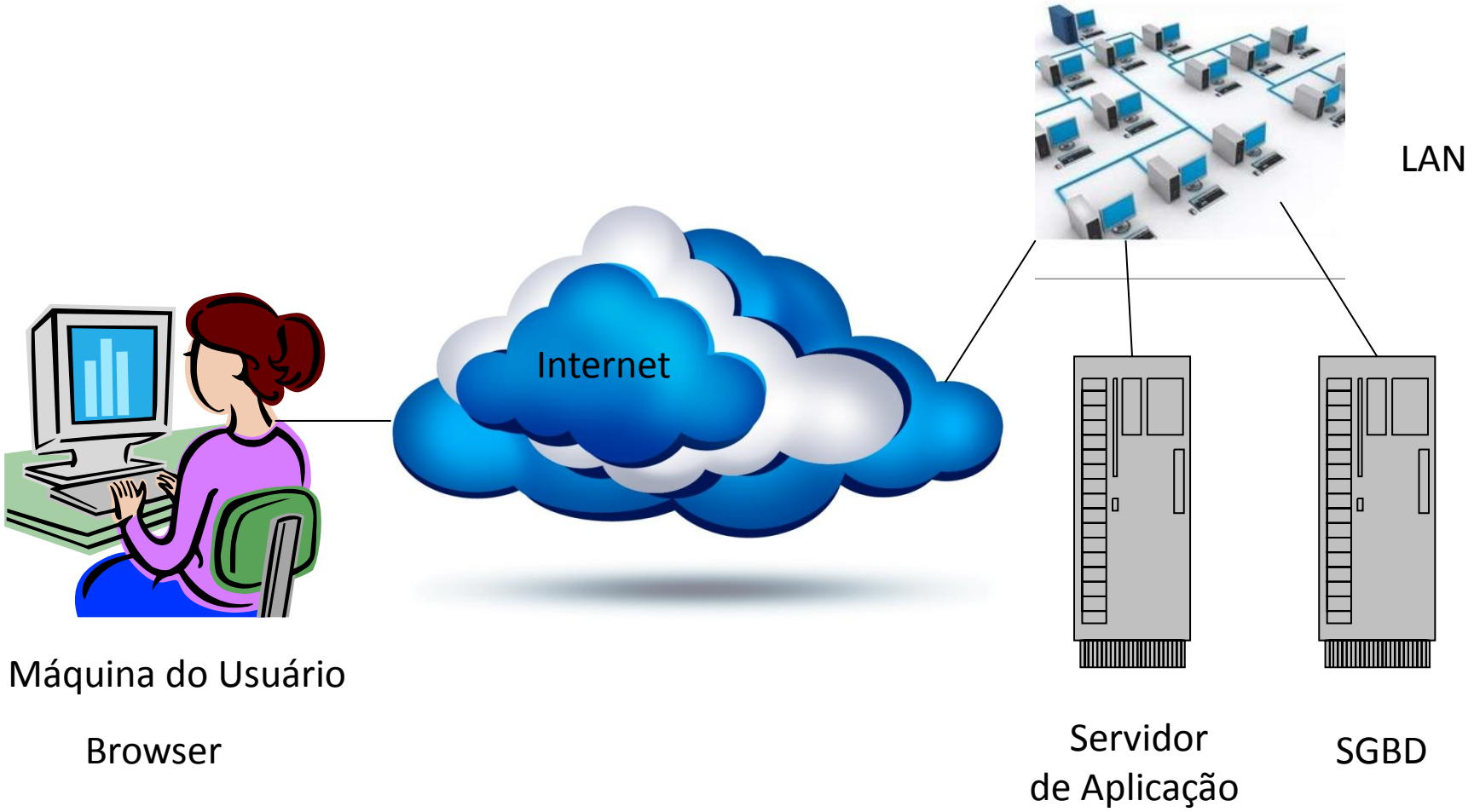


Servidor

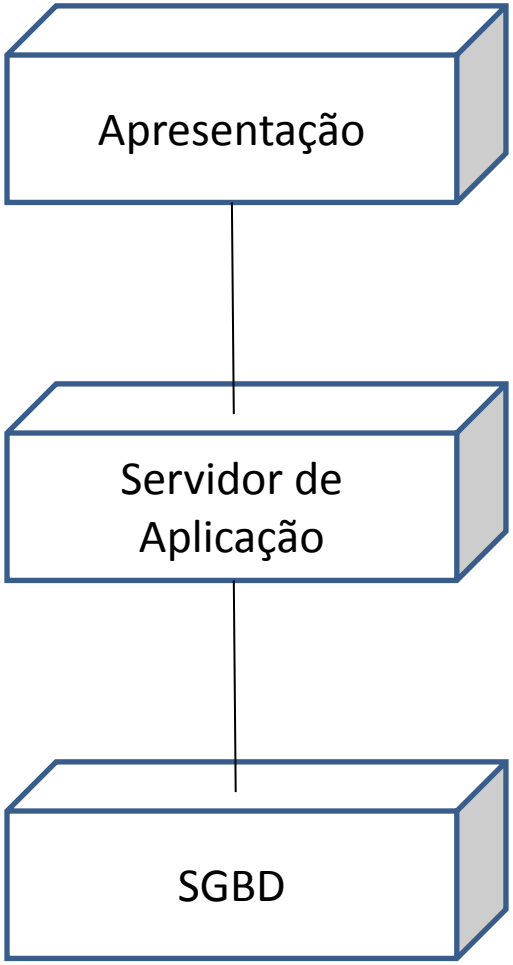
Two Tier

- Década de 90
- Vantajosa quando o número de cliente não é tão grande (uma centena de cliente interagindo com o servidor por uma rede local)
- Se tornou obsoleto com a popularização da Internet
  - Demanda para construção de sistemas que funcionassem na Internet
  - Browsers não foram feitos para clientes gordos

Three Tier, Four Tier ...



Three Tier, Four Tier ...



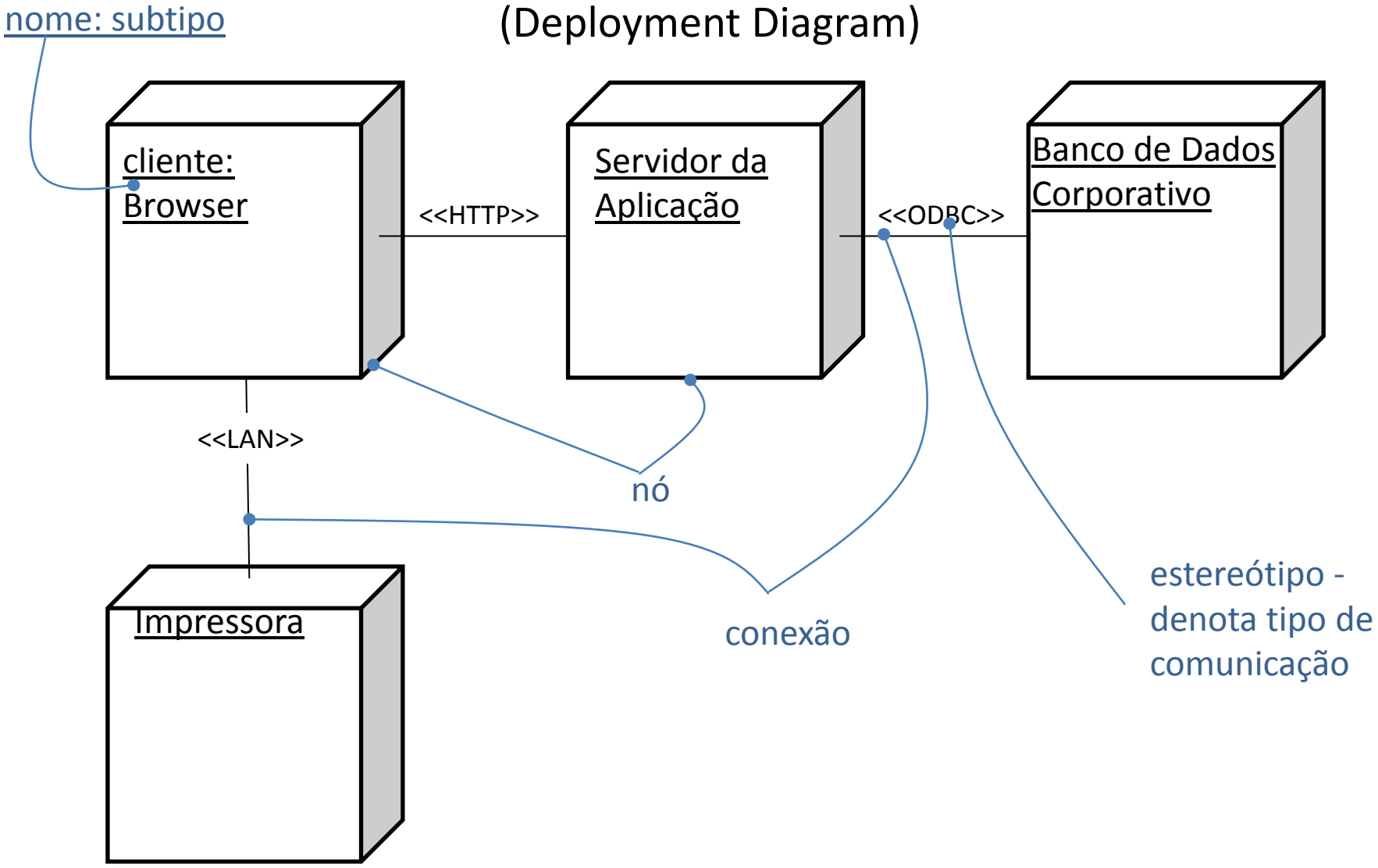
Fica no nó de processamento conhecido como **Presentation Tier**

Camadas lógicas de aplicação e domínio ficam no nó de processamento conhecido como **Middle Tier**

Data Tier

Diagrama de Implantação

(Deployment Diagram)



Alocação de Componentes

- Definida a alocação das camadas nos nós de processamento, falta definir os componentes de cada camada
- O que é um componente?
  - Analogia com a Engenharia Eletrônica
    - chip é um tipo de componente eletrônico de um computador
    - pode ser utilizado para construir um computador e pode ser substituído por outro chip mais poderoso que possua a mesma especificação (interface)
  - De forma semelhante, um componente de software é uma unidade que pode ser usada na construção de vários sistemas e que pode ser substituída por outra unidade que tenha a mesma funcionalidade

Tecnologias de Componentes

Microsoft **COM (Component Object Model) Technology**

CORBA™

OMG
OBJECT MANAGEMENT GROUP

Enterprise JavaBeans

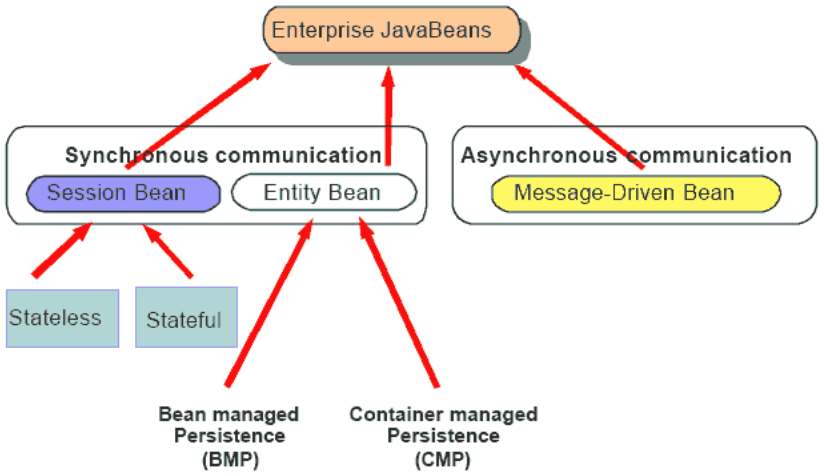
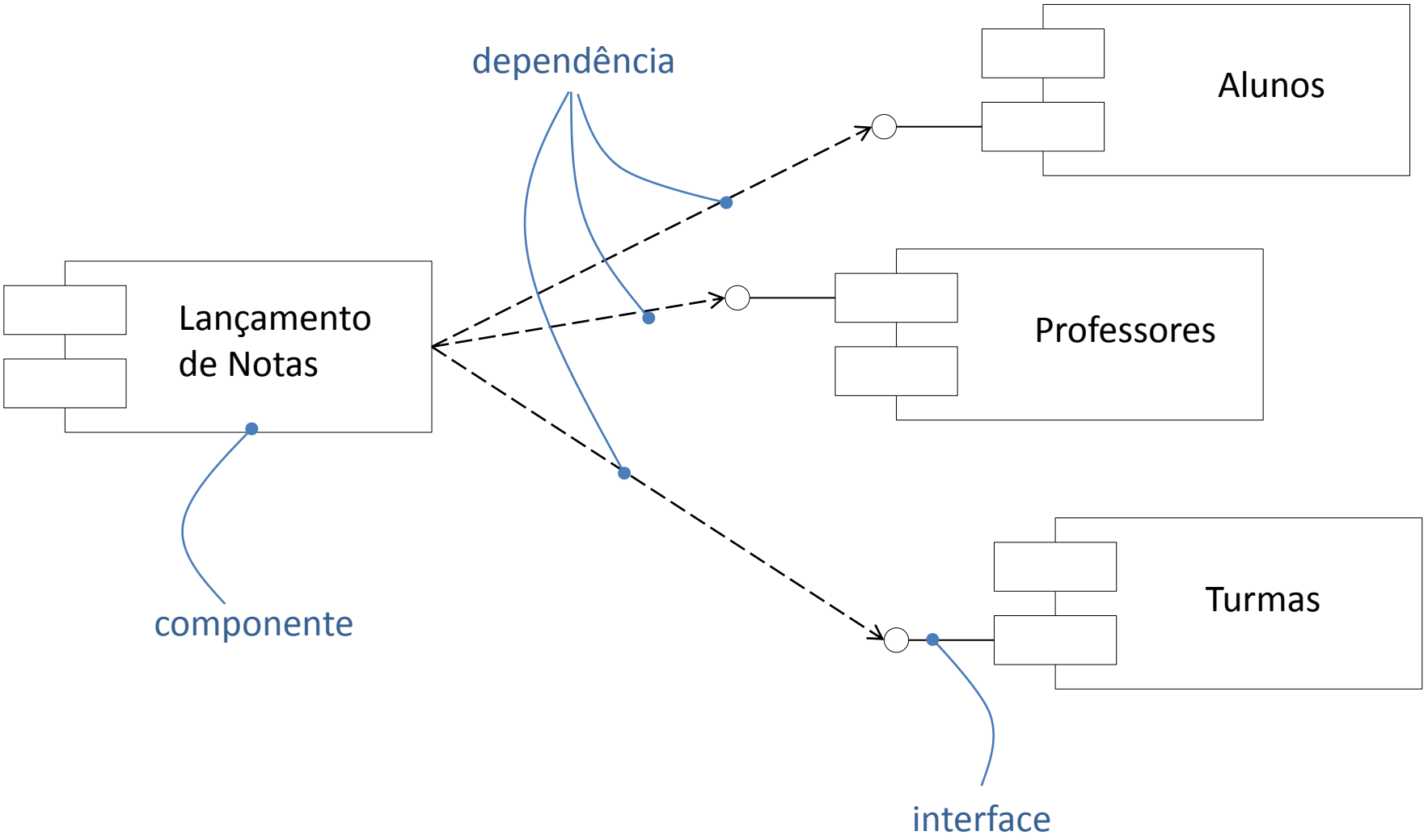
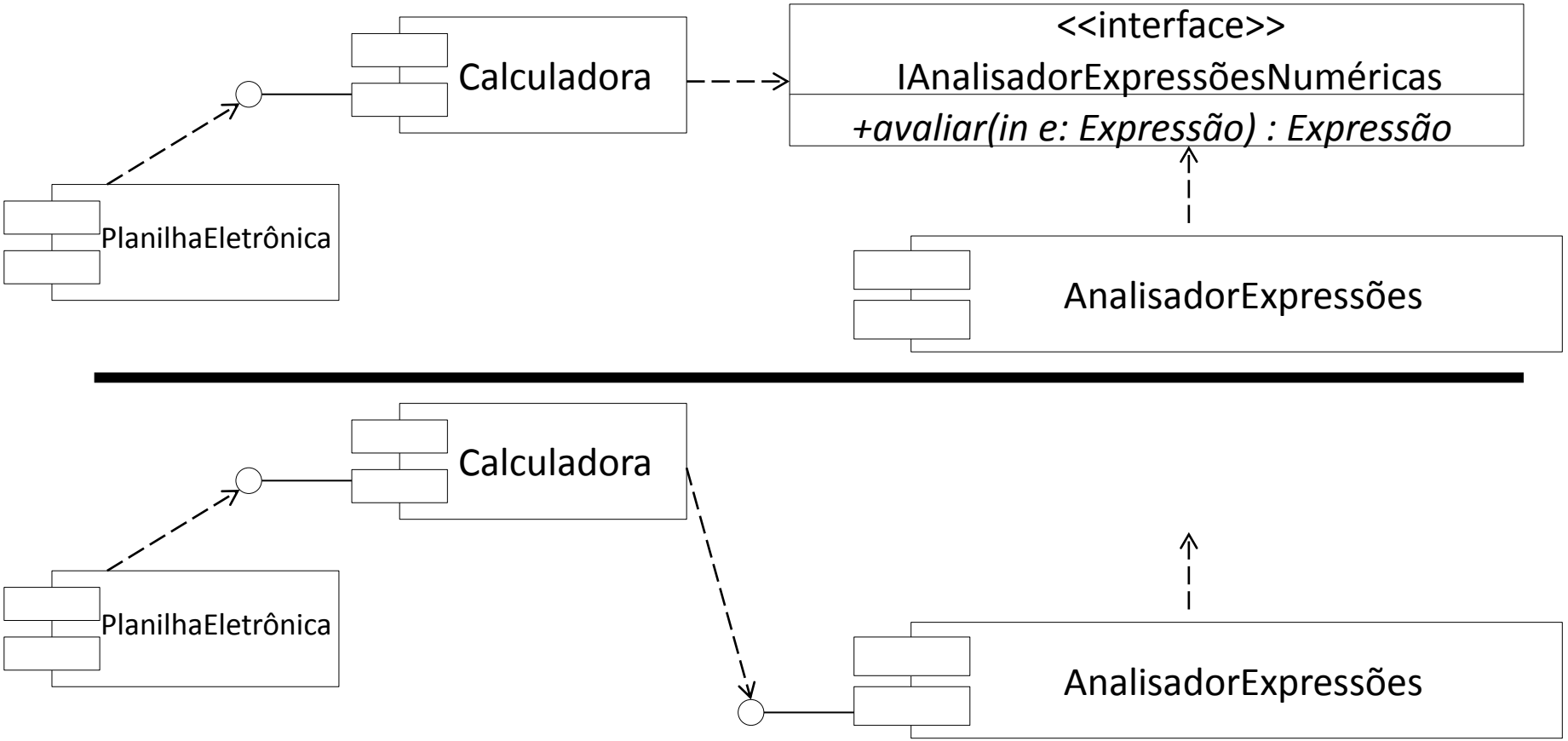


Diagrama de Componentes



Formas alternativas de representação de interfaces de componentes



Alocação dos componentes ao nó físico

- Objetivo
  - Distribuir a carga de processamento do sistema
  - Aumentar o desempenho do sistema
- Para isto é preciso levar em conta que:
  - Envio de mensagem de um objeto a outro
    - No mesmo processo é bastante rápido
    - Em processos diferentes leva até o dobro do tempo
    - Em nós diferentes é uma ou duas ordens de grandeza maior, dependendo do meio de comunicação

Outros fatores que afetam o desempenho

- Utilização de dispositivos de E/S, sua distribuição física e como os componentes o utilizam
- Carga computacional: necessidade de processamento simultâneo de dois ou mais componentes
- Capacidade de processamento dos nós: componentes com exigências mais urgentes de processamento devem ser alocados nos nós mais rápidos
- Realização de tarefas: quanto mais relacionados dois componentes estão, maior é a probabilidade deles serem alocados no mesmo nó
- Tempo de resposta: alocar componentes de forma que as dependências entre eles não cruzem as fronteiras de um nó. O objetivo é minimizar o tráfego pelos canais de comunicação entre os nós

Outros fatores não relacionados ao desempenho

- Necessidade de se comunicar com um sistema legado que se localiza em uma máquina específica e não pode ser realocado
- Segurança: alocar componentes de acordo com critérios de segurança (DMZ, firewall, ...)
- Diferenças de plataforma (hardware ou sistema operacional) dos componentes do sistema. Pode ser que um componente só seja executado em Linux ou Windows
- Características dos usuários do sistema: localização física dos usuários, máquinas que eles utilizam, como estão conectados.
- Redundância: mesmo componente pode ser alocado a mais de um nó. Aplicações tolerantes a falhas.

Implantação Física

- Hardware
  - Dispositivos de E/S
  - Prcessamento
  - Memória
  - Armazenamento secundário
- Telecomunicações e redes
  - Processamento distribuído
  - Internet, intranet e extranets
  - WWW
  - Exclusão digital
- Softwares de Sistema e Aplicação
  - S.O.
  - SW de App
  - Linguagens de programação
  - Tendências
    - Sw aberto
    - Open source
    - Licenças
    - Suporte

Implantação Física

- Necessidade de alta escalabilidade?
- Tipos de sistemas
  - VoD – Netflix, Youtube
  - Músicas - Napster
  - VoIP - Skype
  - Google
  - What´sUp
  - Tweeter
  - Facebook
  - Reservas de passagens aéreas
  - cnn.com

Implantação Física

- Um único servidor monoprocessado
- Um único servidor multiprocessado
- Virtualização
- Clusterização
- Distribuição geográfica da aplicação
- CDN
- P2P
- Cloud Computing

Referências

- Ferramentas de modelagem visual
  - Rational Rose (www.rational.com)
  - ASTAH Community (astah.net/editions/community)
- Livros
  - The Unified Modeling Language User Guide, Grady Booch et al
  - Engenharia de software – uma abordagem profissional, Roger S. Pressman
  - Princípios de análise e projetos de sistemas com UML, Eduardo Bezerra
- Especificações
  - www.omg.org

Dúvidas