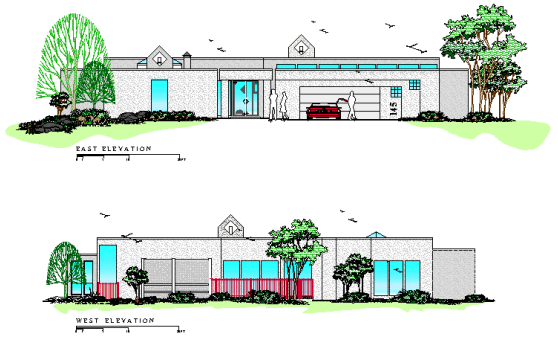
Arquitetura de Sistemas

Envolve os seguintes aspectos

- Composição / Decomposição do Sistema (Subsistemas / Módulos)
- Componentes e a Interação entre eles
- Níveis / Camadas e a Interação entre eles (Ordem / Estrutura)
- Organização das partes físicas do software a serem implementadas
- Restrições do software (naturais ou auto-impostas)
- Descrição geral do software
- Estrutura Estática / Dinâmica de um Sistema
- Estilo que orienta o desenvolvimento e a evolução do software
- Funcionalidade do software
- Outras considerações : reuso, desempenho, escalabilidade

Arquitetura de um Sistema



Definição: Representa a estrutura estática e comportamental de um sistema, compreendendo os serviços do sistema, realizados pelos componentes de software, que rodam nos processadores disponíveis

Está associada a detalhes internos do software na medida que esses detalhes manifestam-se externamente

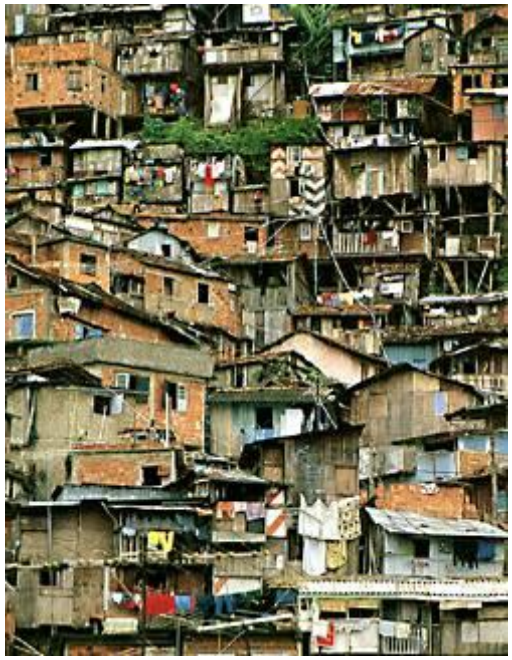
Não pode ser representada por um simples diagrama: é multifacetada

Arquitetura de um Sistema

Está associada a:

- Uso
- Funcionalidade
- Desempenho
- Resiliência
- Reuso
- Compreendibilidade
- Economia
- Restrições e compromissos tecnológicos
- Estética

Todo sistema já criado tem sua Arquitetura



Ela existe independente do seu conhecimento pelo projetista de sistemas

O Por Quê da Arquitetura

Uma abordagem adhoc leva a sistemas difíceis de mudar ou adaptar

Decomposição de sistemas em partes menores torna o software mais fácil de entender, administrar, desenvolver, manter

Auxilia no desenvolvimento baseado em componentes

Auxilia desde o início na administração do desempenho

Leva a um reuso melhor

Influencia a segurança, a testabilidade, a manutenbilidade e a administração do sistema

Conceitos Básicos da Arquitetura duma Aplicação Corporativa

Decomposição

Componentes

Frameworks

Patterns (Padrões)

Nívelamento

Camadas (Tiers)

Decomposição

Particionamento do sistema em partes menores, lógicas

Que tornam fácil a administração de sua complexidade

Auxilia a definição e descrição das interfaces entre as diferentes partes do sistema, de forma a facilitar a sua iteração

Facilita a distribuição do software através de múltiplos processadores

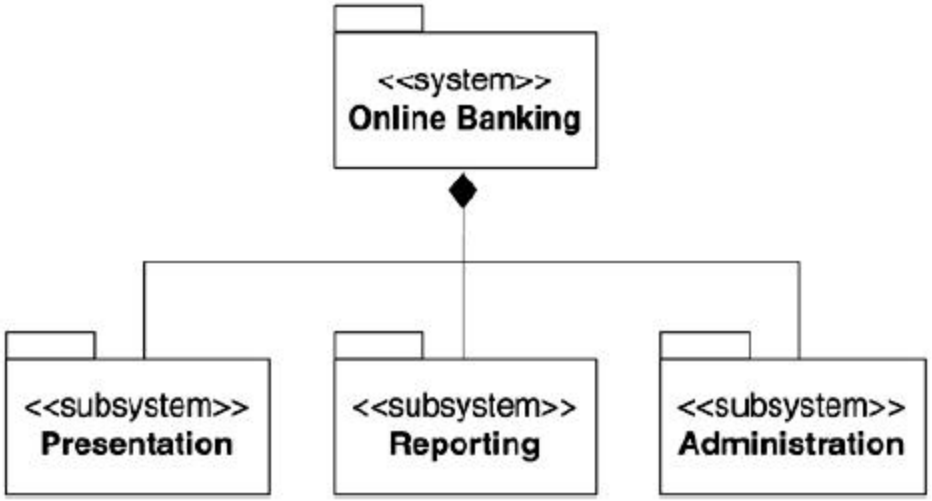
Provê uma partição natural das tarefas de desenvolvimento e facilita a distribuição das tarefas em equipes grandes

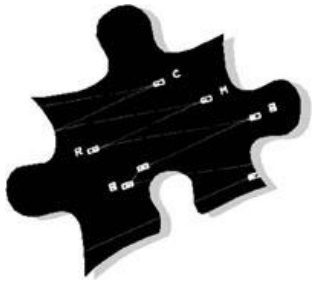
Decomposição com a UML

Pacotes

Subsistemas

Módulos





Componentes

Unidade coesa de software que provê um conjunto relacionado de serviços e funções

Útil somente no contexto de um modelo padrão (CORBA, EJB, DCOM)

Infraestrutura básica para composição e interação do componente

Tem uma interface (contrato entre o componente e a aplicação) bem definida que permite a sua interação com outros componentes

Debaixo de um mesmo modelo padrão e com a mesma interface podem ser substituídos

Podem conter outros componentes

Por que usar componentes

Em relação ao software tradicional são mais fáceis de manter e modificar

Tem o potencial de melhorar a produtividade com o reuso e componentes pré-fabricados

Desenvolvimento de aplicações mais flexíveis

Podem ser comprados e vendidos na medida que existir um mercado consolidado nesta área

Facilitam a partição mais natural do software em unidades coesas

Granulação dos Componentes

Componentes de alta granularidade / Subsistemas de alto nível

- Devem ter poucas dependências, bem definidas
- Disponibilizam capacidade de negócios discretas e complexas
- Ex.: Módulo de Controle de Estoques

Componentes de baixa granularidade

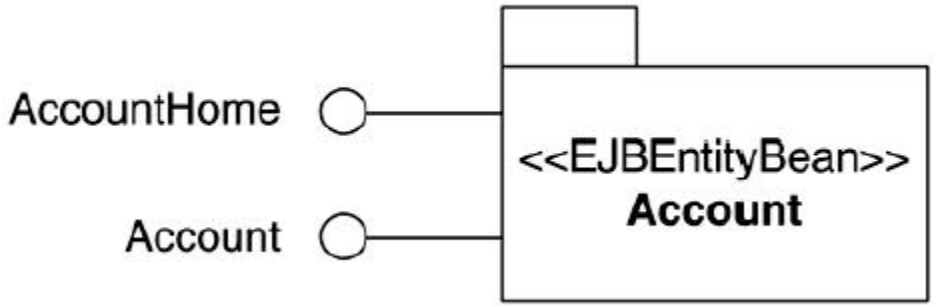
- Comparável com objetos tradicionais
- Tem maior número de dependências
- Ex.: Java Beans

Componentes em UML

Com suas interfaces

Com seu comportamento Interno

- Diagrama de Atividades
- Diagrama de Estado



Componente como um:

- subsistema
- módulo executável

Framework

Qualquer porção pré-definida de software desenvolvido e testado para ser reusado em vários projetos de desenvolvimento de aplicações num domínio específico

Tipo de micro arquitetura, abrangendo um conjunto de padrões que trabalham em conjunto para resolver um problema básico, num domínio comum

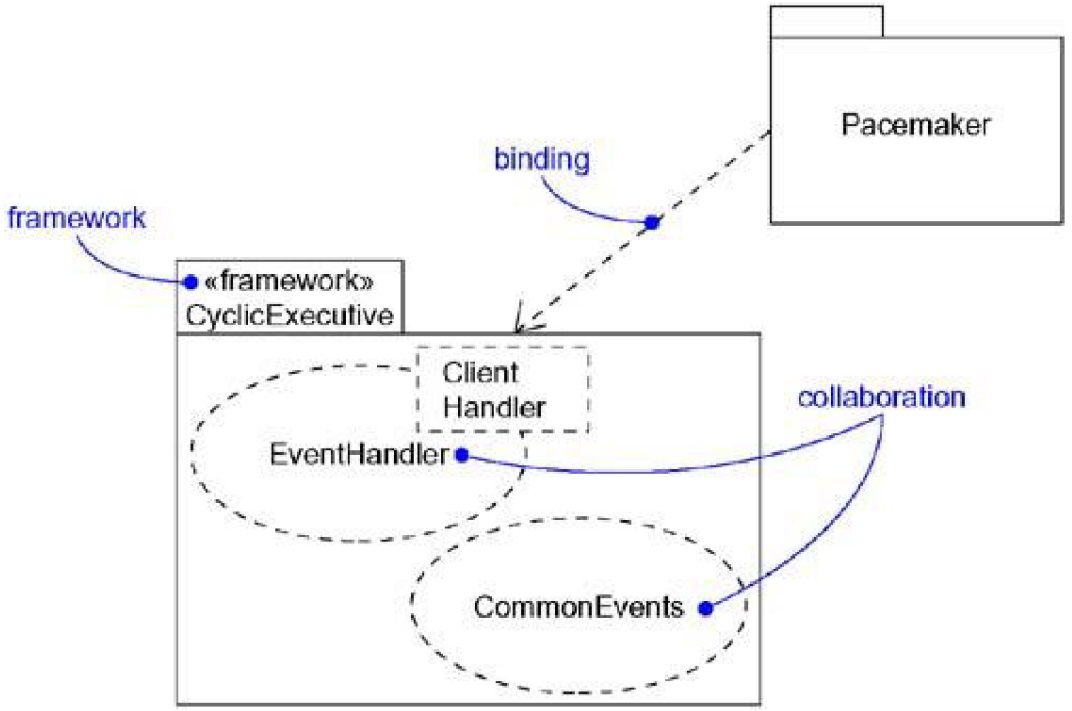
Permite a especificação, agrupamento e reuso de elementos para se construir efetivamente algum sistema

Framework na UML

Estereotipo de Pacote

Contendo

- Colaborações
- Diagramas



Framework – Modo de Utilização

Biblioteca

Conjunto de componentes reusáveis

Gabarito

Base para novas aplicações

Desenvolvimento de um Framework

ser simples de se entender

prover documentação adequada

Identificar mecanismos concretos de sua própria extensão

Padrões (Patterns)

Solução em um determinado contexto, que foi obtida através de experiência e se mostrou eficaz para resolver um problema básico nessa área.

Utilidade

Capta conhecimento comprovado obtido através de anos de experiência

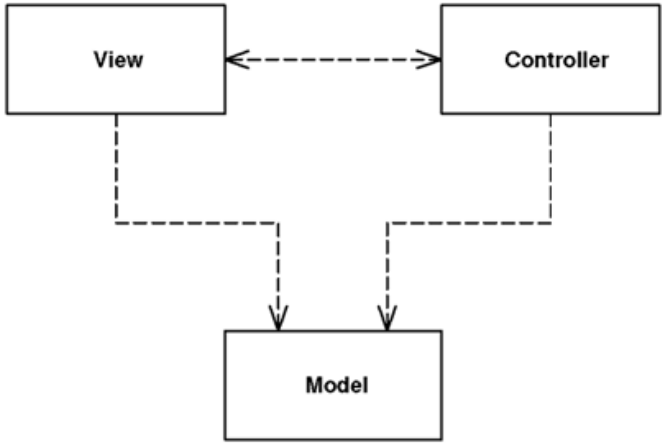
Ajuda na solução de problemas complexos encontrados em situações similares

Favorece a comunicação na equipe, provendo contexto básico para discussões

Exemplo

Padrão MVC (Design)

Como separar os objetos de entidade da aplicação (Model) da maneira como estes são apresentados para o usuário (View) e de como o usuário faz o controle sobre os mesmos (Controller)



Model

Sabe tudo sobre:

- **Os dados persistentes que deverão ser mostrados**
- **As operações que serão aplicadas para transformar os objetos**

Sabe nada sobre:

- **As interfaces do usuário**
- **Como os dados serão mostrados**
- **As ações das interfaces usadas para manipular os dados**

Representa os dados da aplicação e as regras de negócio que governam o acesso e a atualização desses dados

View

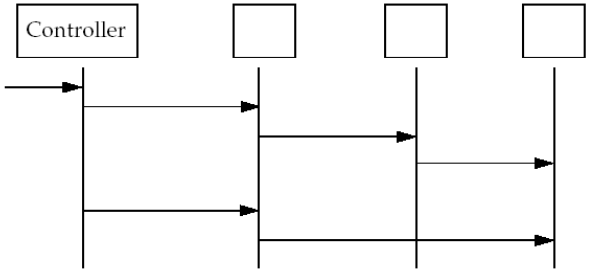
Refere-se ao objeto Model

Dispara as operações de consulta do Model para obter os dados e visualizá-los

Define como os dados serão visualizados pelo usuário

Mantém consistência na apresentação dos dados quando o Model muda

Controller



Sincroniza as ações do View com as ações realizadas pelo Model

Trabalha somente com sinais e não com os dados da aplicação

Sabe os meios físicos pelos quais os usuários manipulam os dados no Model

Vantagens da utilização do padrão MVC

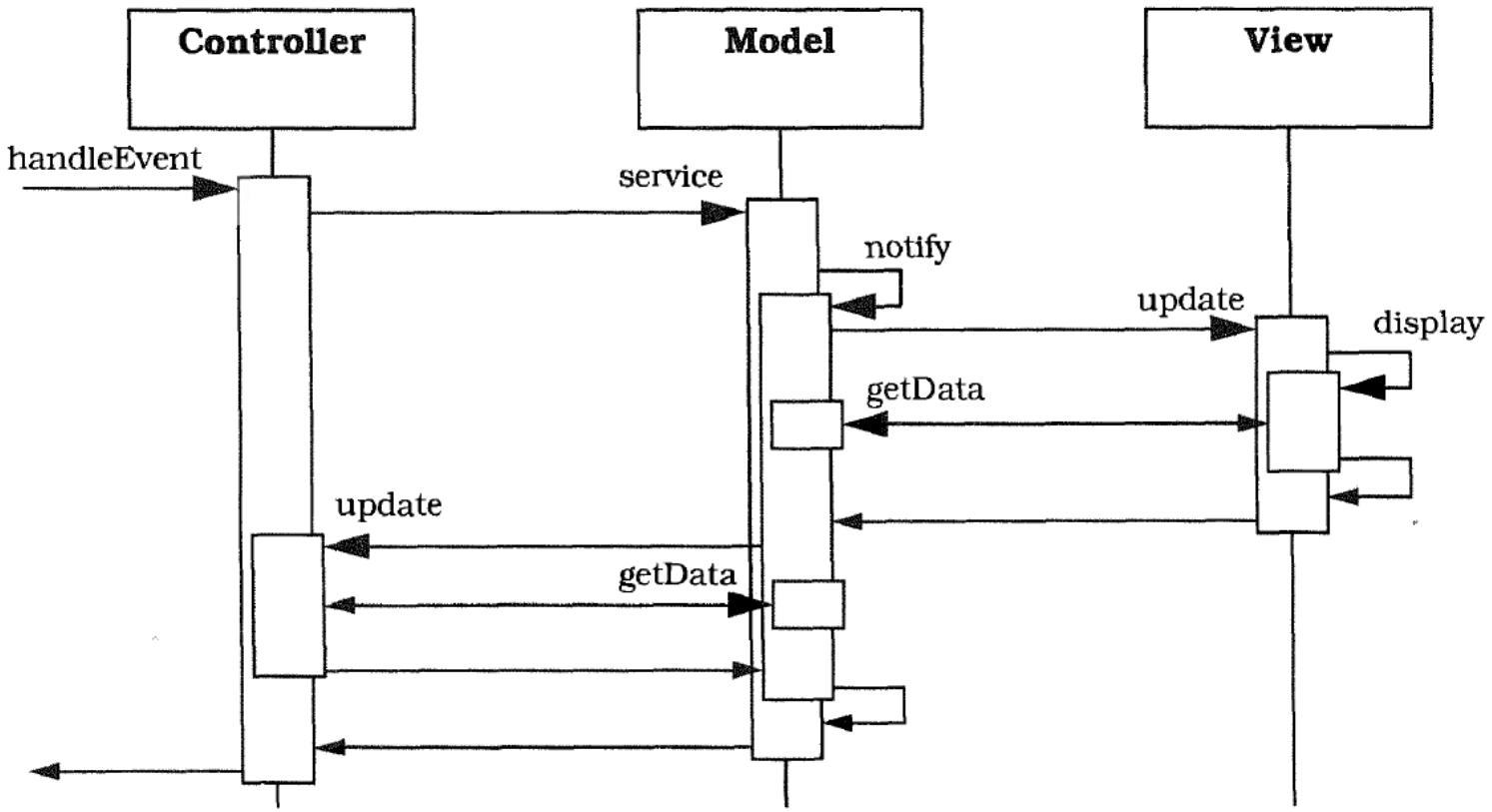
Múltiplos Views utilizando um mesmo Model

Suporte fácil para novos tipos de usuário do sistema

Projeto mais bem definido e modular

Facilidade de ampliação do sistema

Utilização em sistemas distribuídos



Dinâmica do sistema em UML

Classes Gerais de Padrões

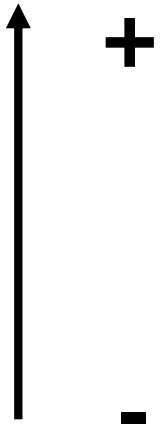
Referência

Análise (Negócios)

Arquitetônicos

Projeto (Design)

De Código



Nível de Abstração

Pattern Arquiteturas de Referência

Uma arquitetura geral e abstrata que pode ser instanciada para situações específicas:

- Completada, detalhada
- Parametrizada, adaptada
- Extendida, selecionada

Relacionada à um certo tipo de organização

- Telecomunicações
- Governo
- Bancos

Técnica

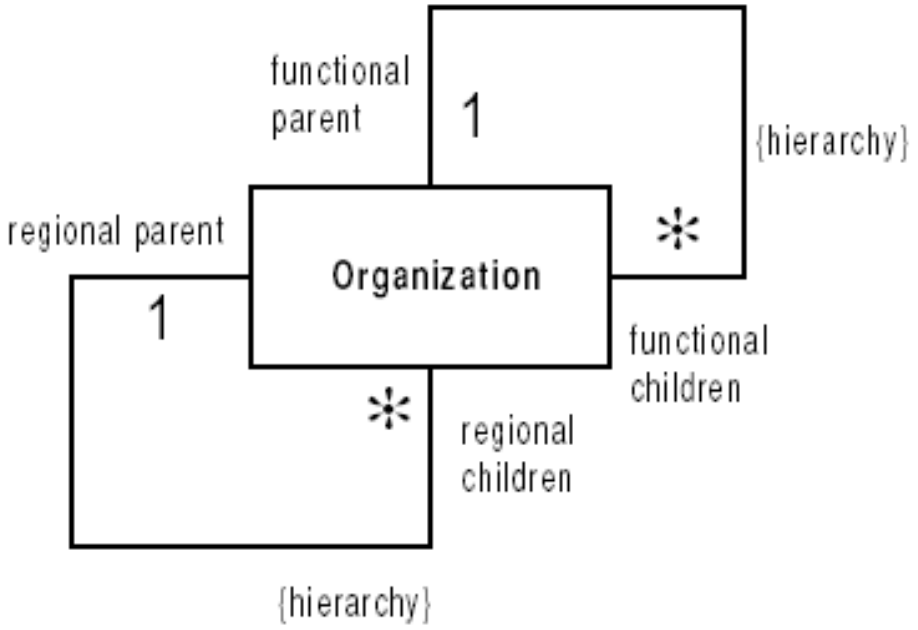
- SOA
- MDA

Padrões de Análise

Repositório - Fowler

Exemplo:

Hierarquia de uma Organização



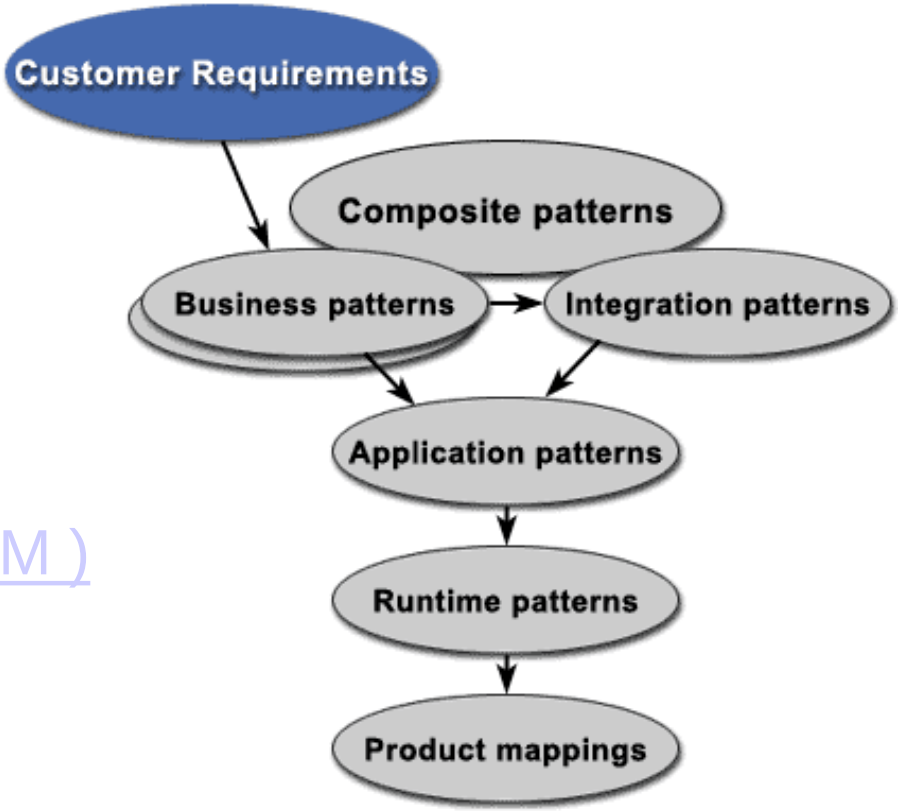
Modelos do domínio de sistemas (contas) em contextos específicos (finanças)

Padrões Arquitetônicos

Trata da estrutura dos sistemas , seus componentes, ambientes de processamento e como eles se relacionam entre si

Exemplo:

E-Business Patterns (IBM)



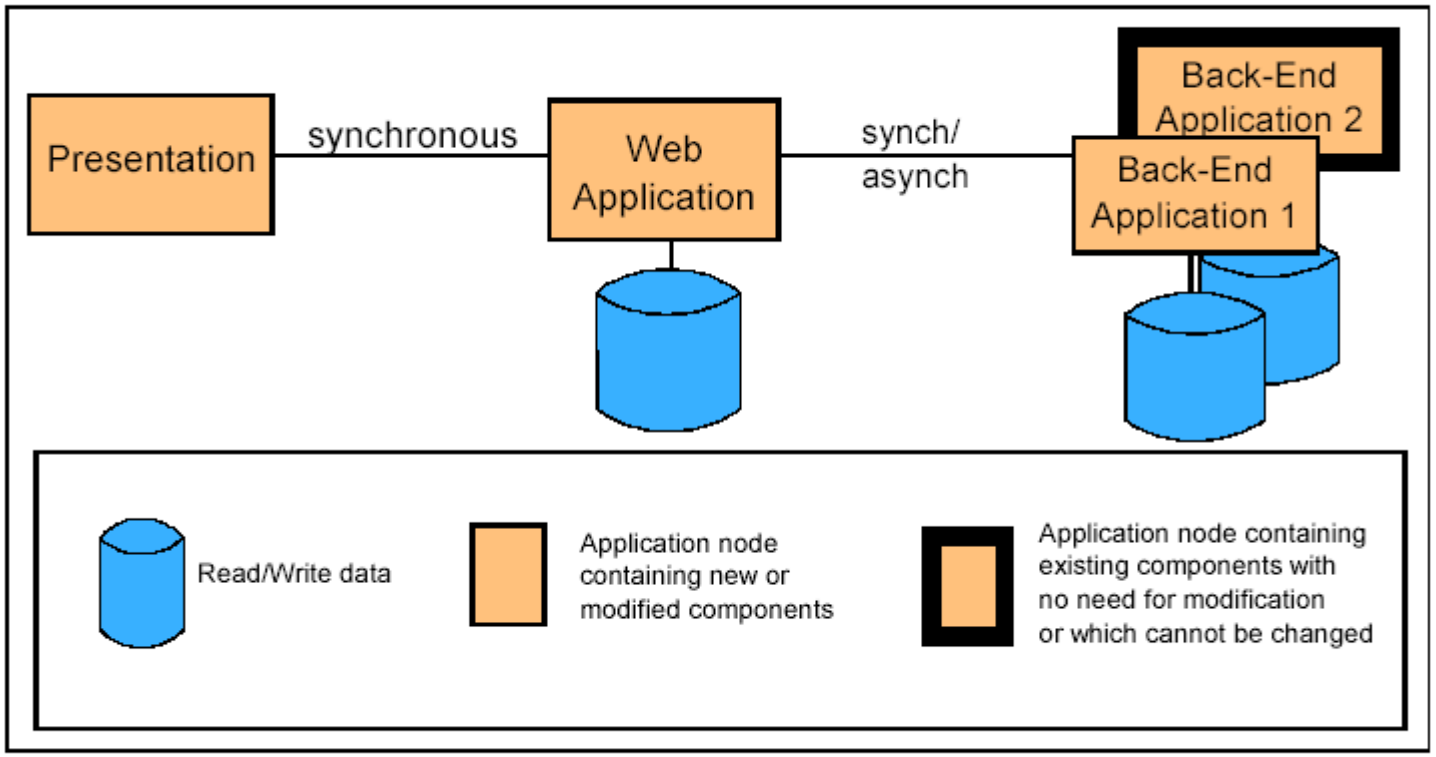
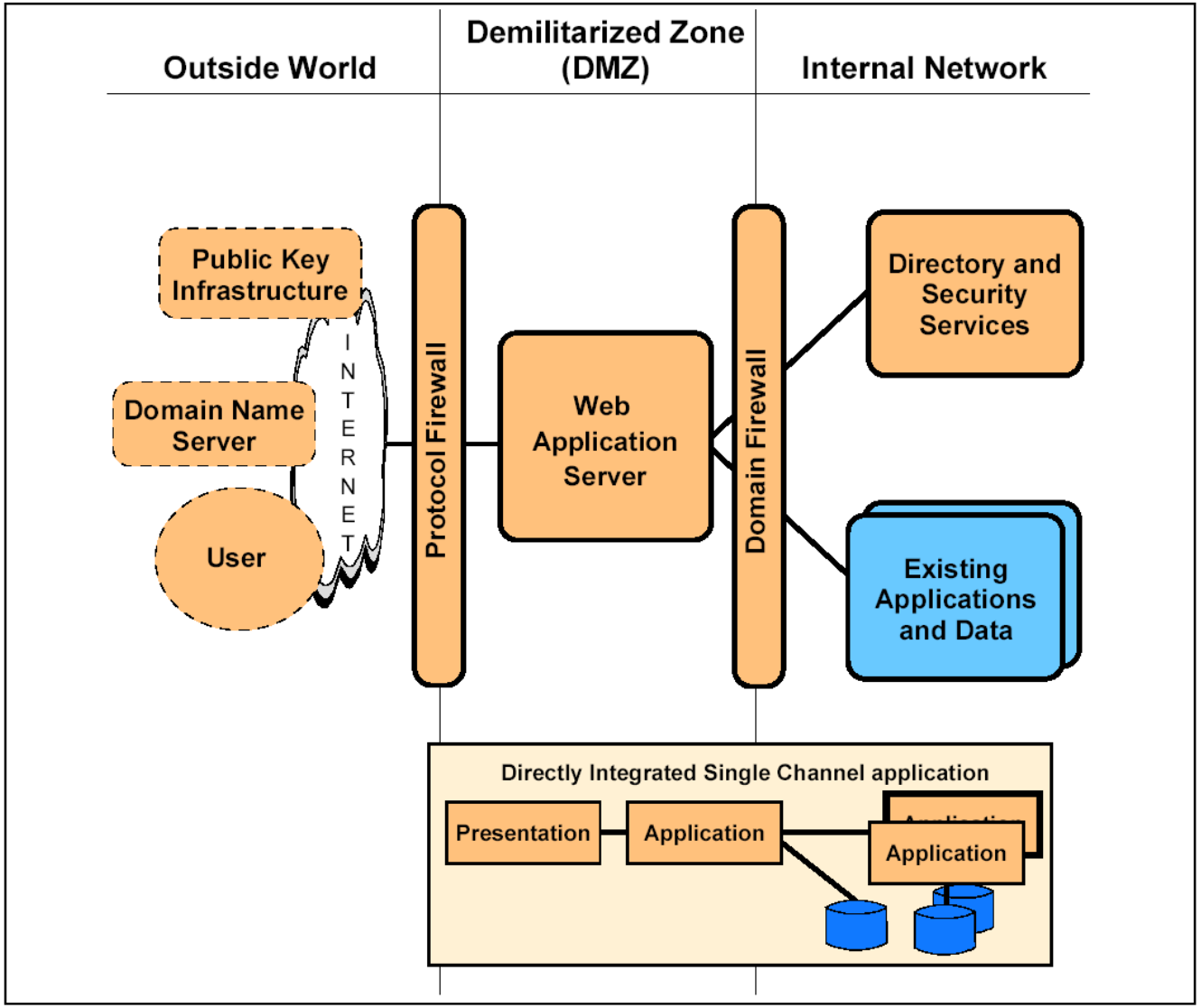


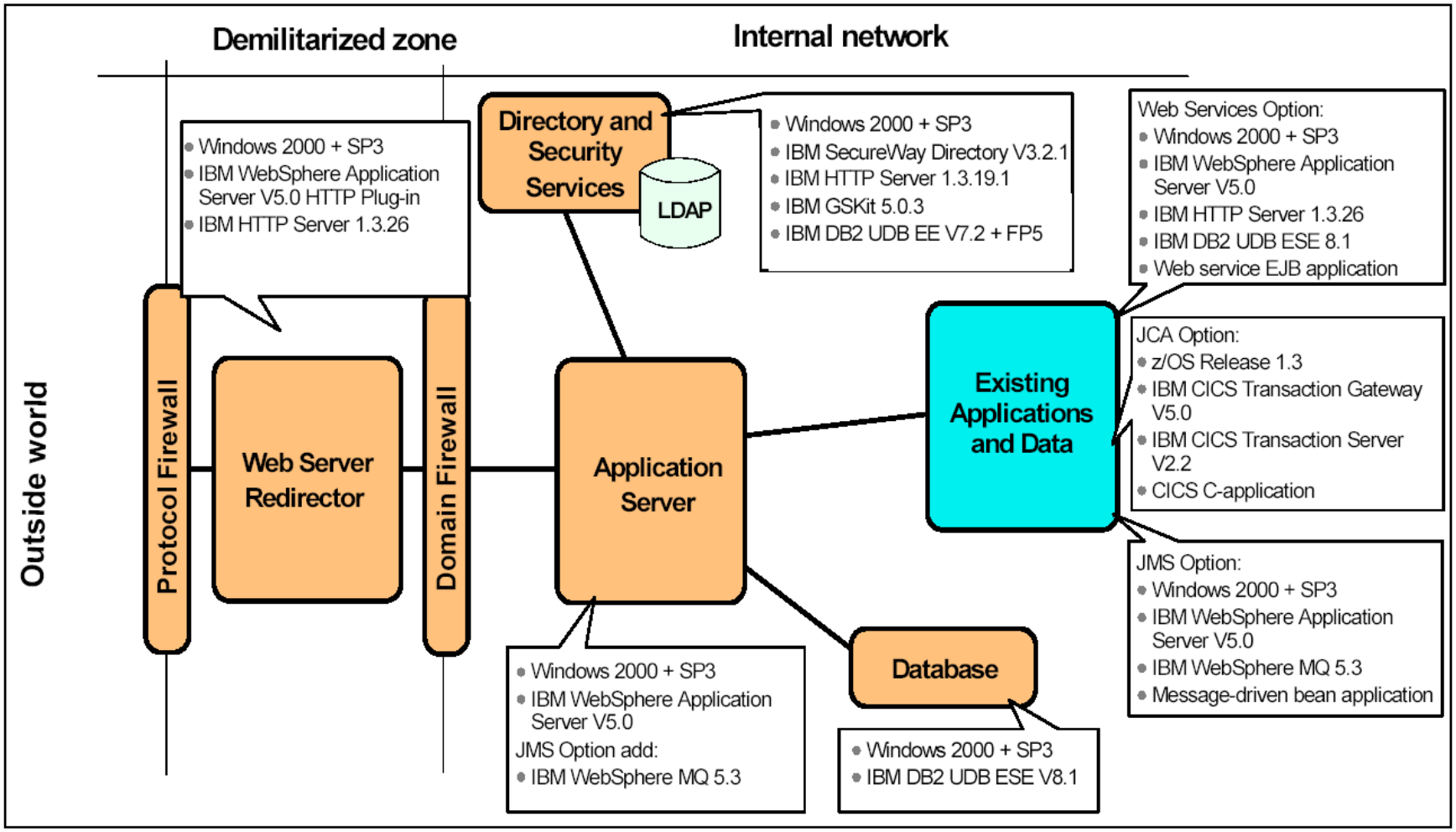
Figure 1-4 Self-Service::Directly Integrated Single Channel

Padrão de Aplicação **Directly Integrated Single Chanel** para o Padrão de Negócios **Self-Service**



Pattern Runtime para o Pattern de Aplicação

Direct Integrated Single Chanel



Mapeamento de produtos Windows 2000 para o Pattern de Aplicação **Direct Integrated Single Chanel**

Padrões de Design

Catálogo Java Patterns - Sun

Usados a nível de classes e objetos

Tipos:

De Criação: soluções para configuração e inicialização do design

Estrutural: estrutura de modo específico as interfaces e as relações das respectivas classes

Comportamental: identifica caminhos nos quais um grupo de classes interage com os demais, para alcançar um certo comportamento

Pattern de Criação

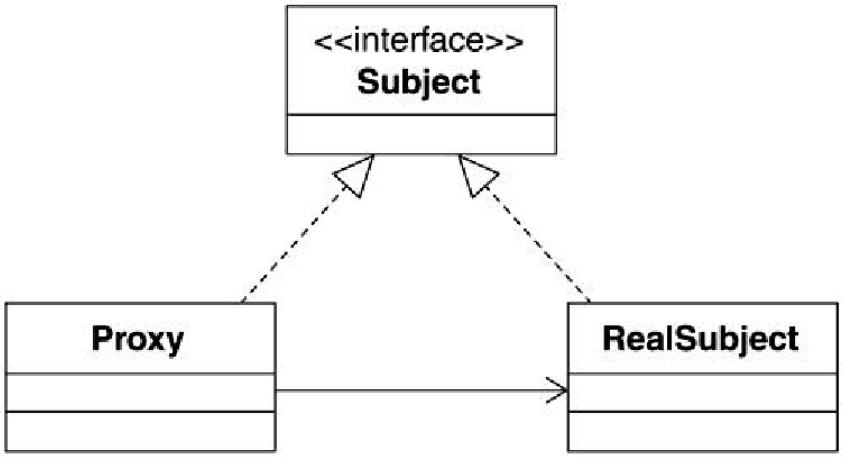
Exemplo

Pattern Singleton

Garante que uma classe tenha somente uma instância e provê um ponto global de acesso a ela

Pattern Estrutural

Exemplo



Pattern Proxy

O objeto Proxy provê um mecanismo indireto de acesso a outro objeto (RealSubject). Os dois objetos trabalham integrados através de uma mesma interface (Subject).

Vantagem: acesso mais fácil ao sistema (através de Proxy) quando existem restrições (por exemplo de segurança)

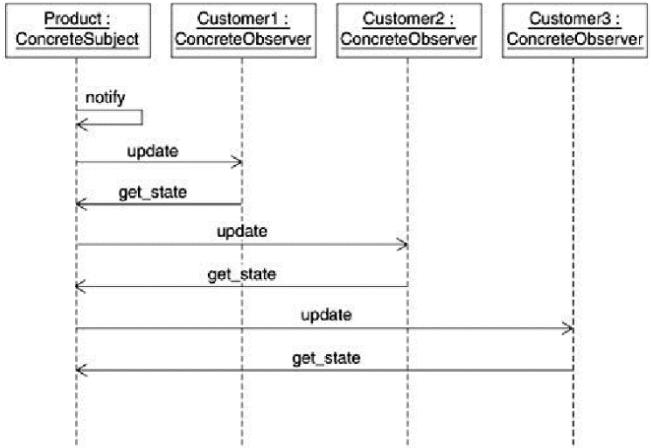
Pattern Comportamental

Exemplo: Pattern Subject-Observer

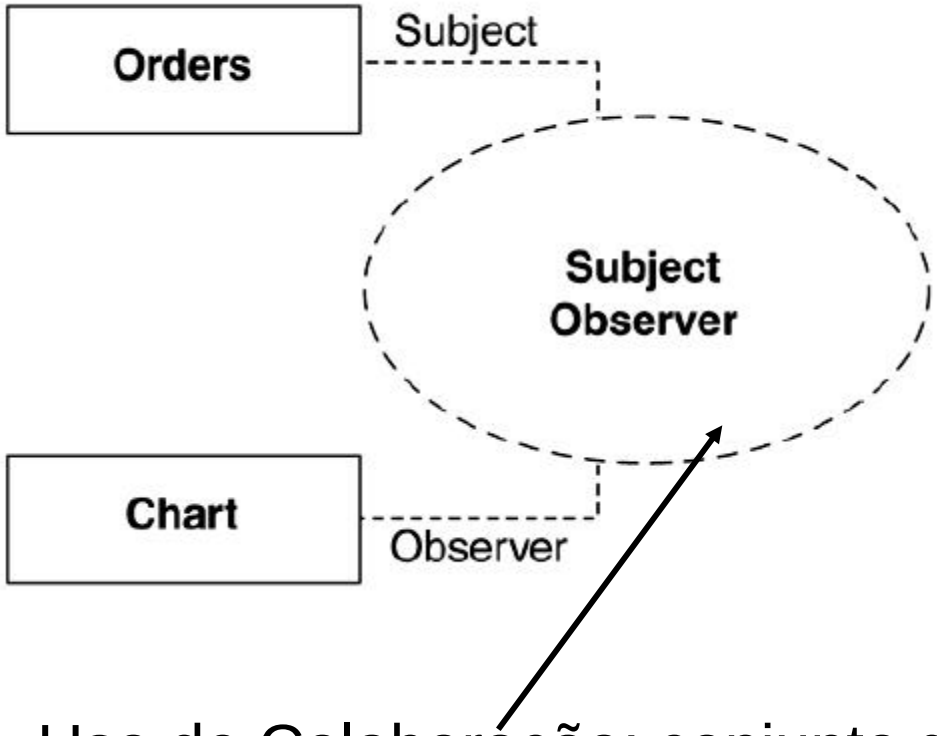
Uma pessoa (Observer) tem interesse em certo produto (Subject). Ela se registra esse interesse no sistema do fabricante, visando receber atualizações (update) sobre o produto.

Quando o produto é atualizado os Observer recebem notificações da mudança. Os Observer individualmente podem consultar o Subject específico para saber detalhes.

Especificação Comportamental

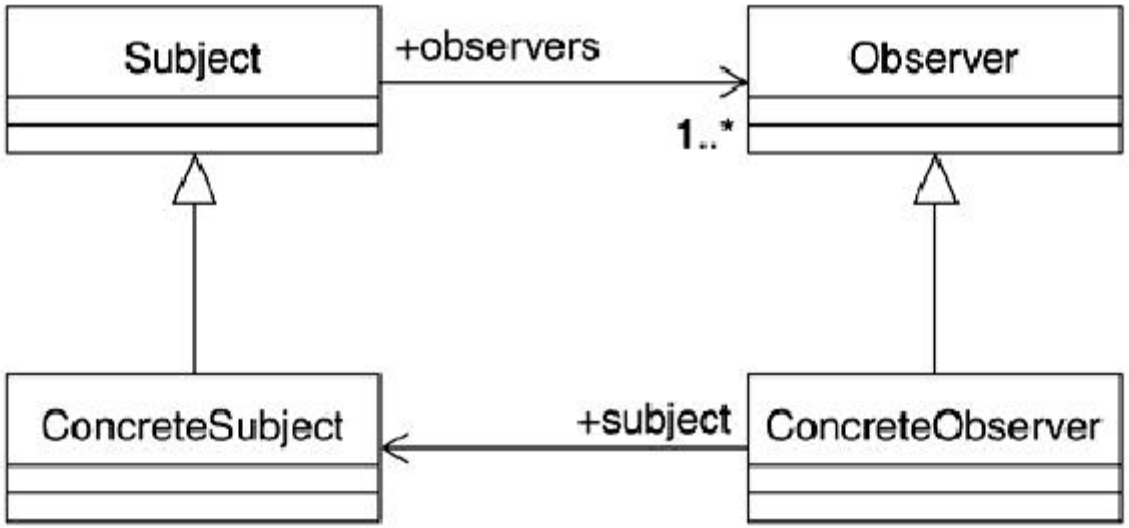


Patterns: **Representação em UML**



Uso de Colaboração: conjunto de objetos e seus vínculos, que interagem num certo contexto, para implementar certa estrutura (Diagrama de Classes) ou comportamento (Diagrama de Interação ou Estados)

Pattern Subject-Observer



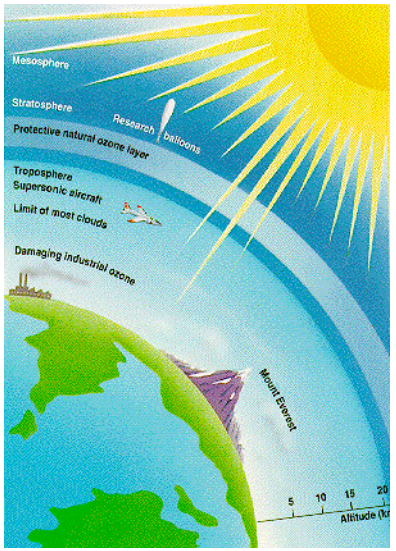
Especificação Estrutural

Nívelamento (Camadas)

Pattern para decomposição

Tratar complexidade

Partição lógica de sistemas
(subsistemas, módulos)



Abstrai tipos específicos de funcionalidades (camadas funcionais)

Provê limites conceituais (entre conjuntos de serviço)

Agrupa, separa, restringe o uso de itens do sistema

Tipos de Nivelamento

Por Responsabilidades Camadas

Distribuição do sistema em diversos processos colocados em múltiplos processadores ou num único

Ex.: Arquitetura Cliente-Servidor

Por Reuso

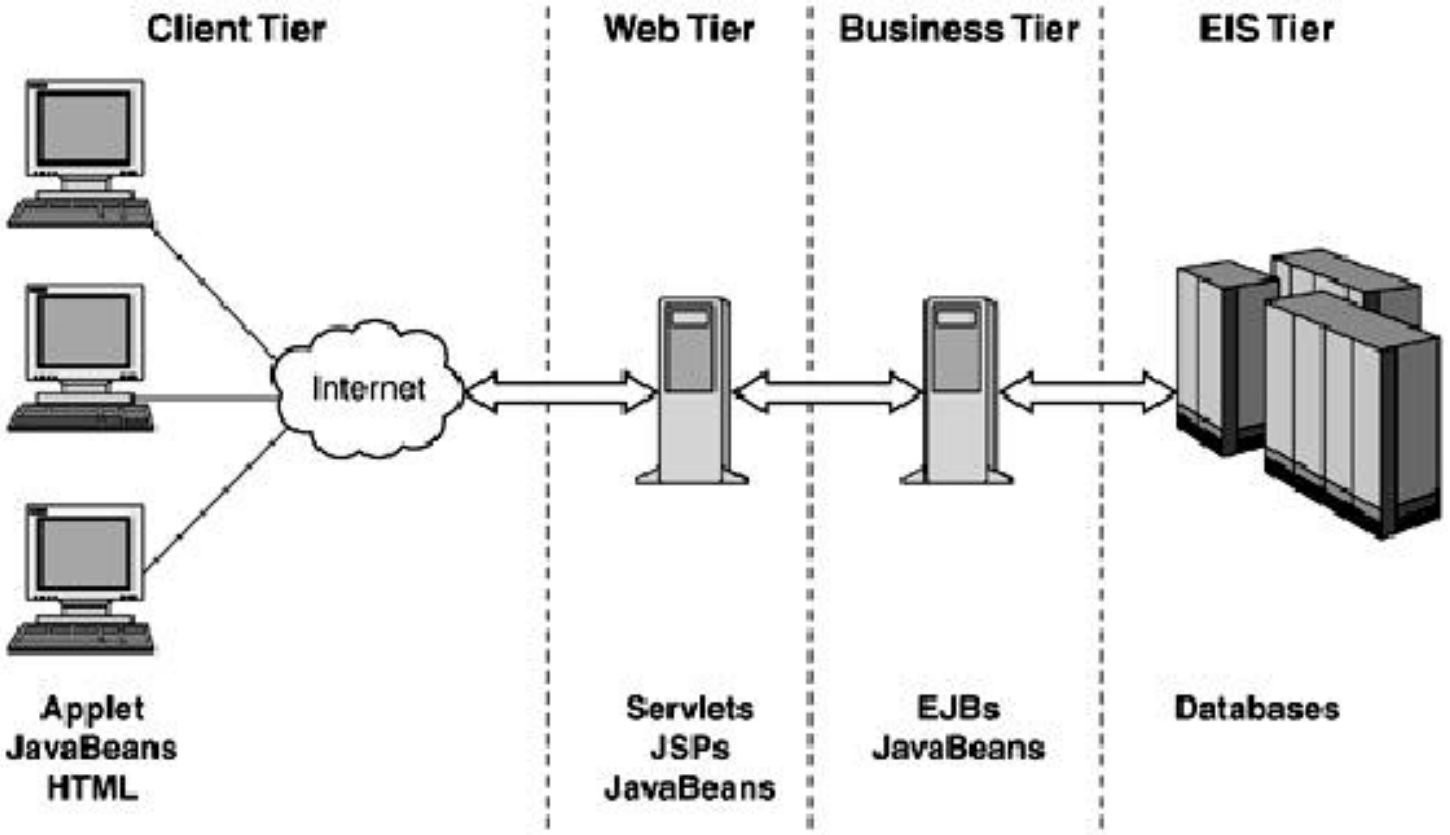
Desenvolvimento do sistema de forma que os diversos níveis prestam serviços aos outros níveis

Ex.:

Nível de fornecimento de informações ao usuário (Apresentação)

Nível de serviços gerais: log, tratamento de erros

Camadas (tiers) da Plataforma J2EE



Camadas

Componentes

Serviços

Características Básicas

Cada camada provê serviços para as outras camadas

Existe baixo acoplamento entre as camadas

O relacionamento entre as camadas é hierárquico por natureza

- Cada camada pode contar com a camada logo abaixo, não ao contrário
- Não deve existir dependência entre camadas que não sejam vizinhas

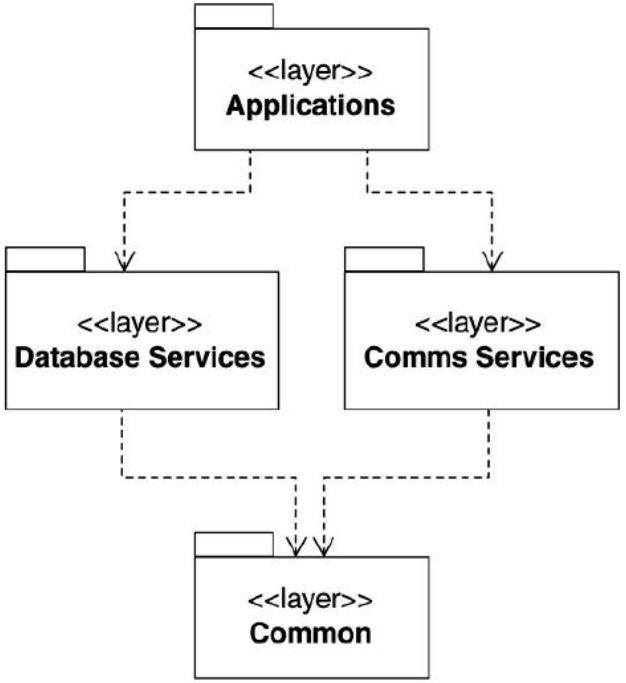
Camadas podem ter suas sub-camadas

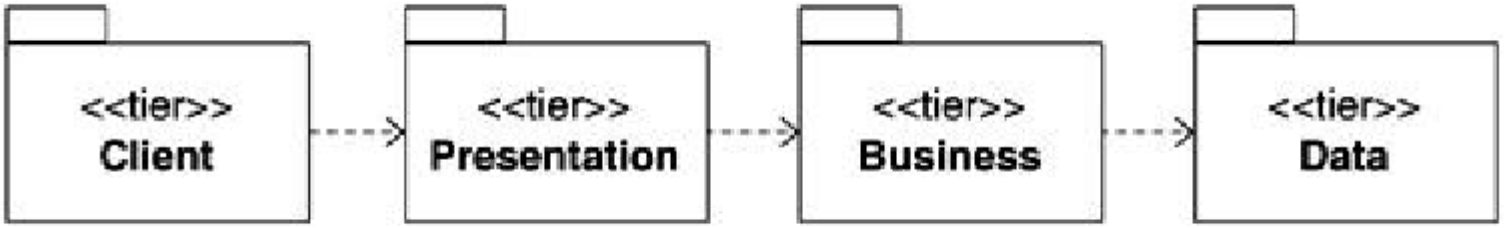
Normalmente as camadas

- Mais baixas: são mais fortemente ligadas ao hardware e ao sistema operacional (serviços básicos)
- Intermediárias: são a base para se construir diversos sistemas com capacidade similar (serviços gerais)
- Superiores: voltadas para as pecularidades do usuário

Nivelamento em UML

Arquitetura
Nivelada





Camadas

Cliente: interação do usuário

Apresentação: resultados das consultas de negócios

Negócios: principais regras de negócios

Dados: interface com o armazenamento dos dados persistentes

Arquitetura J2EE multi-camadas

Colocando tudo Junto

O que vem primeiro ?

Arquitetura do sistema

Análise do sistema

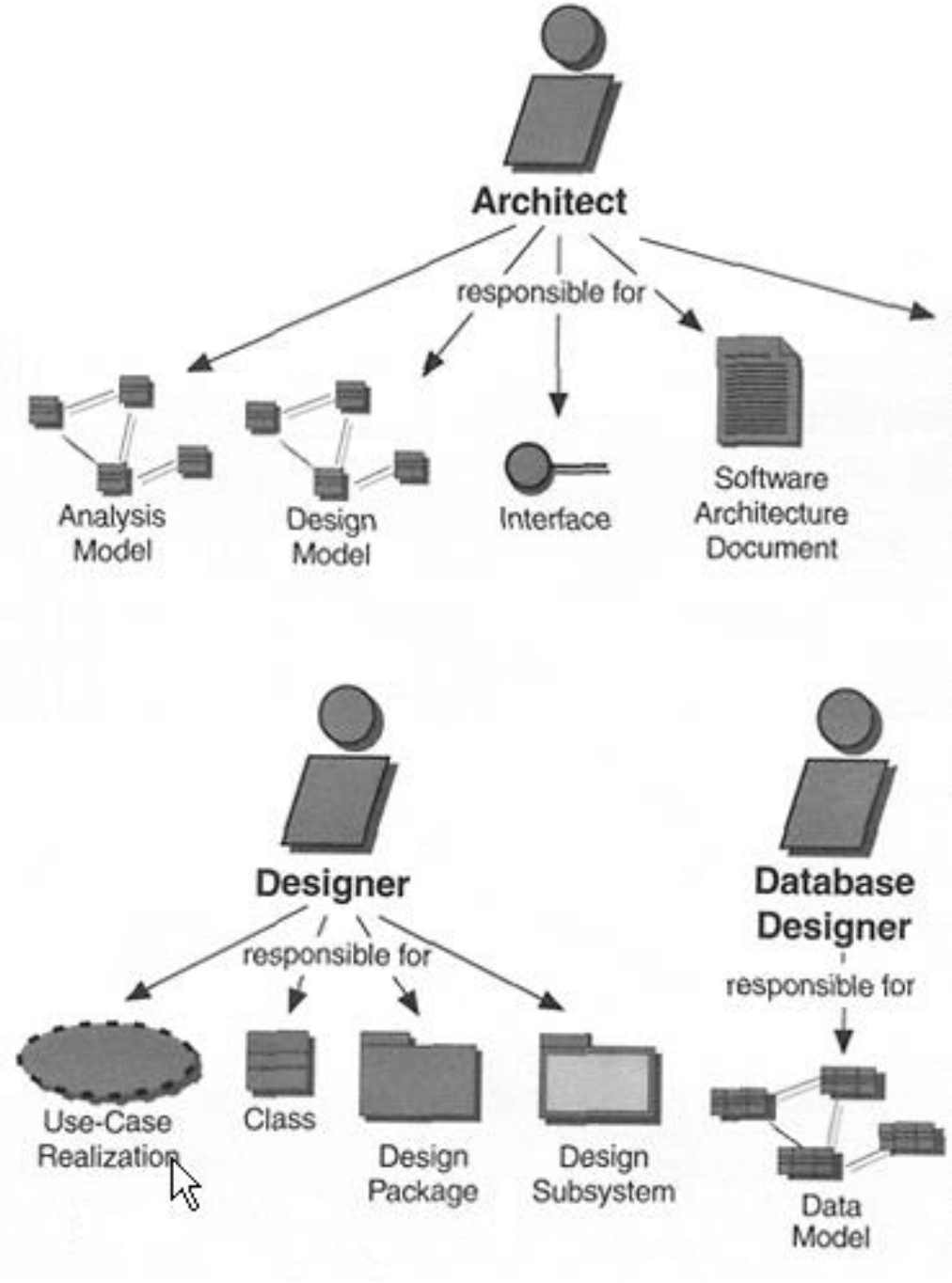
Processo Interativo

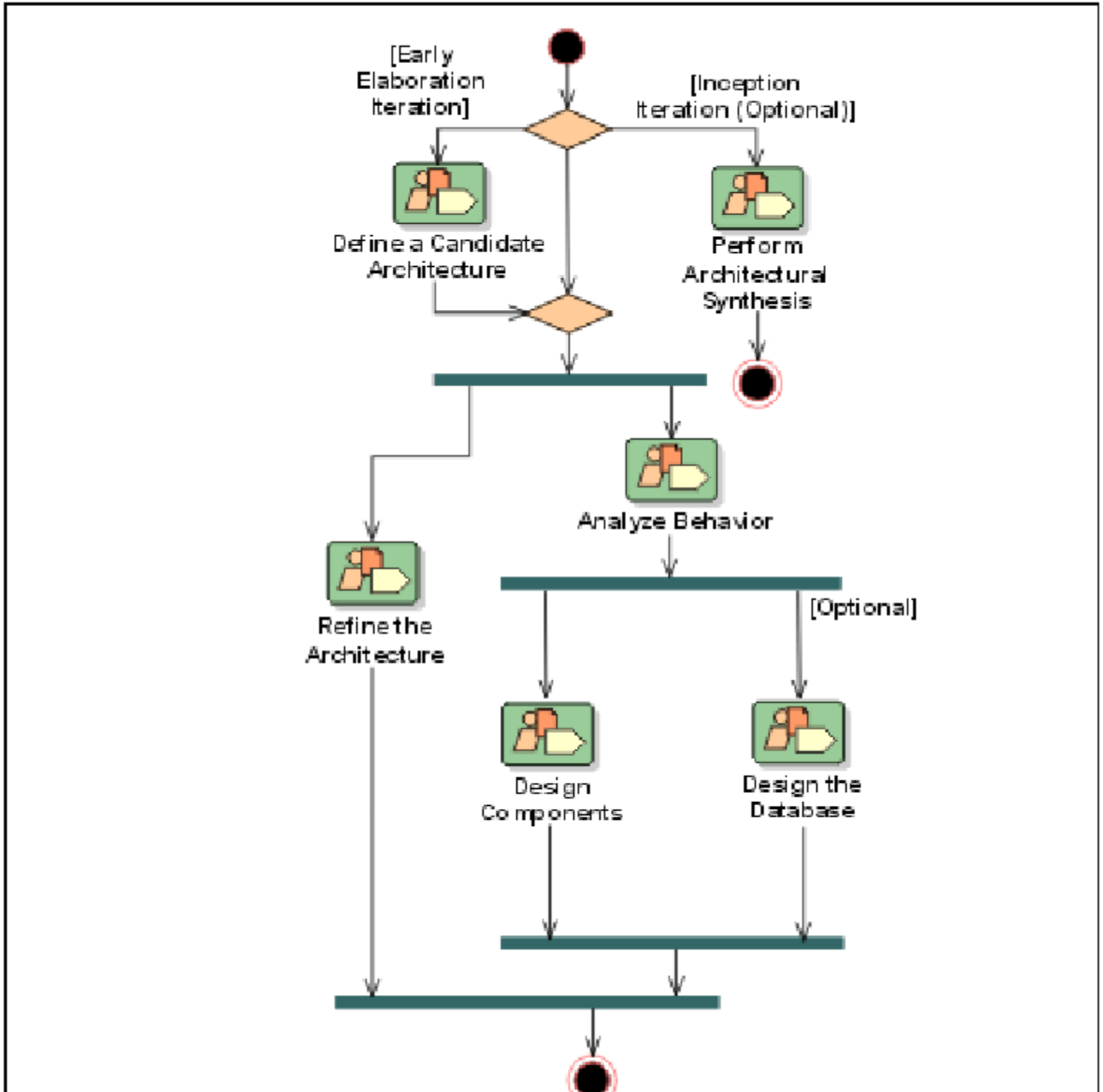
Ao requisitos são entradas importantes para se definir a arquitetura

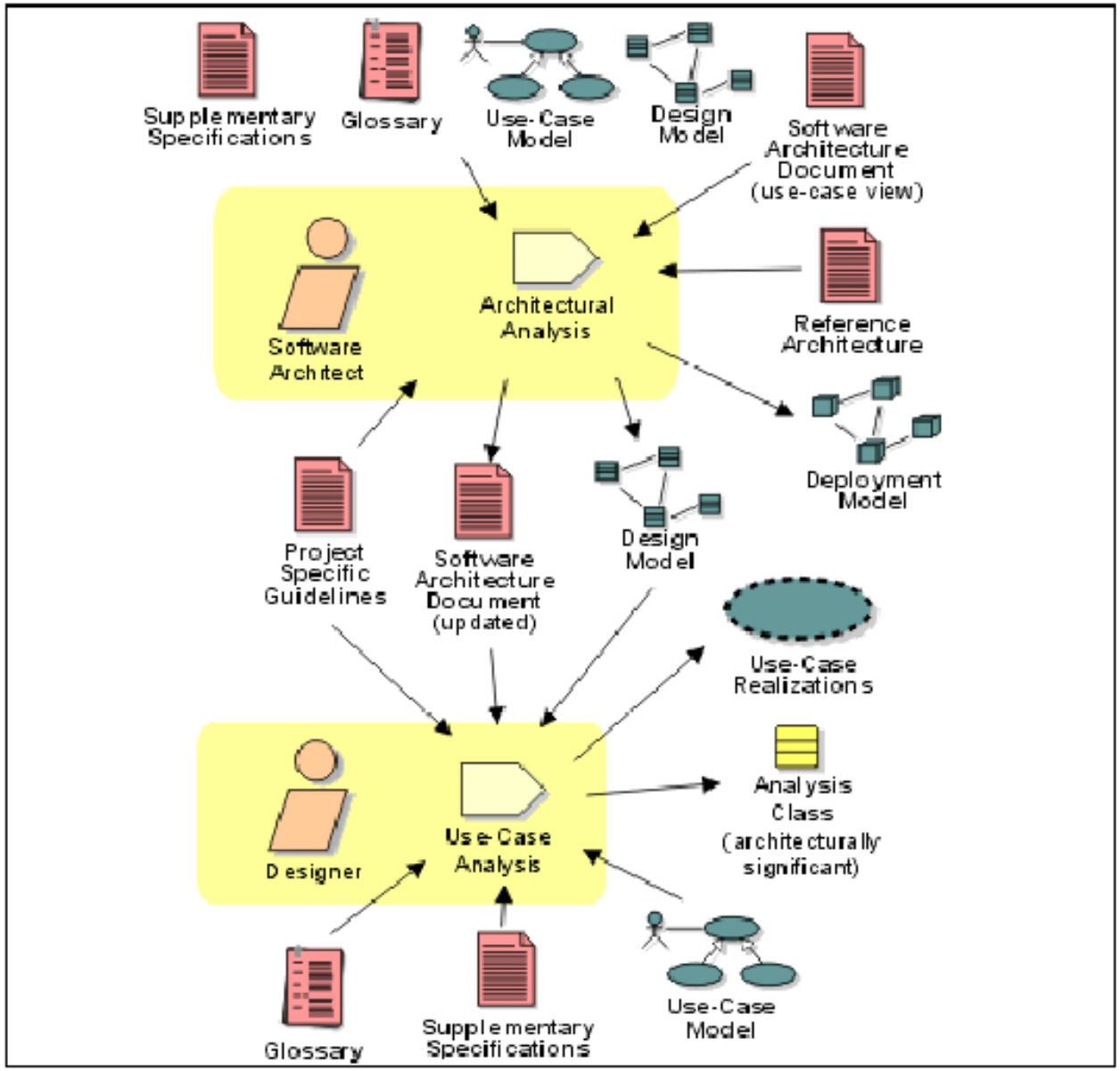
Na medida que se trabalha na arquitetura vê-se a necessidade de se ajustar ou clarear os requisitos

Definir a Arquitetura é muito mais um processo evolutivo

Ela toma forma na medida que as decisões são tomadas considerando-se os requisitos específicos e os compromissos do sistema


Architect
responsible for
Analysis Model
Design Model
Interface
Software Architecture Document
Designer
responsible for
Use-Case Realization
Class
Design Package
Design Subsystem
Database Designer
responsible for
Data Model


[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
[Optional]
Design Components
Design the Database


Supplementary Specifications
Glossary
Use-Case Model
Design Model
Software Architecture Document (use-case view)
Software Architect
Architectural Analysis
Reference Architecture
Deployment Model
Project Specific Guidelines
Software Architecture Document (updated)
Design Model
Use-Case Realizations
Designer
Use-Case Analysis
Analysis Class (architecturally significant)
Glossary
Supplementary Specifications
Use-Case Model