UNISINOS

Engenharia de Software - Fundamentos

Fluxos e Processos de Negócio com BPMN

Prof. Guilherme Lacerda

guilhermeslacerda@gmail.com

Roteiro

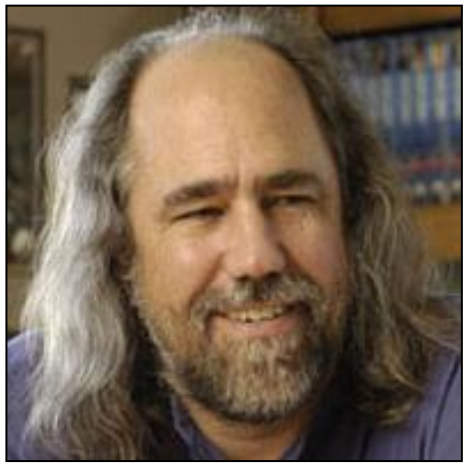
- Introdução
- Visões
- Diagramas UML
- Ferramentas

Introdução

Introdução

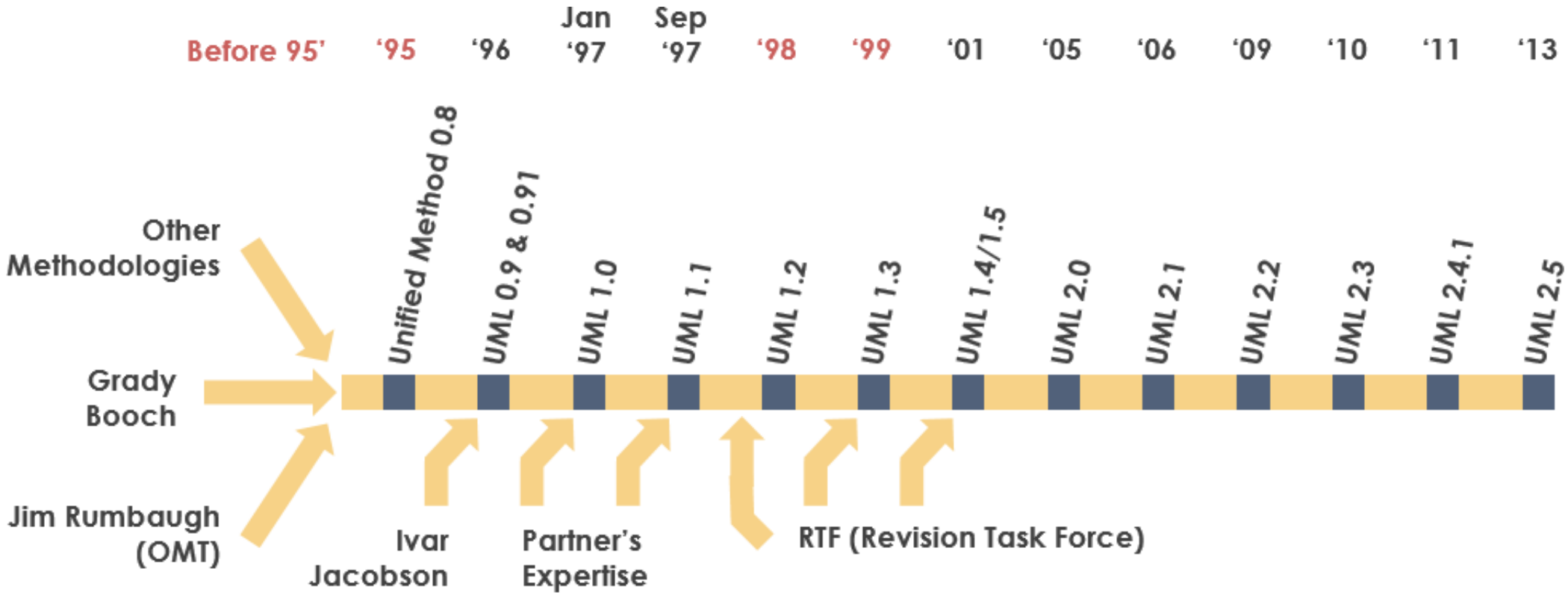
- É uma linguagem de modelagem para:
  - Visualizar, especificar, construir e documentar artefatos de sistemas de software
  - Proporcionar uma forma padrão para plano de arquitetura de projeto de software, incluindo regras de negócio e funções do sistema
  - Fornece múltiplas visões do SW, sob diferentes aspectos

Introdução



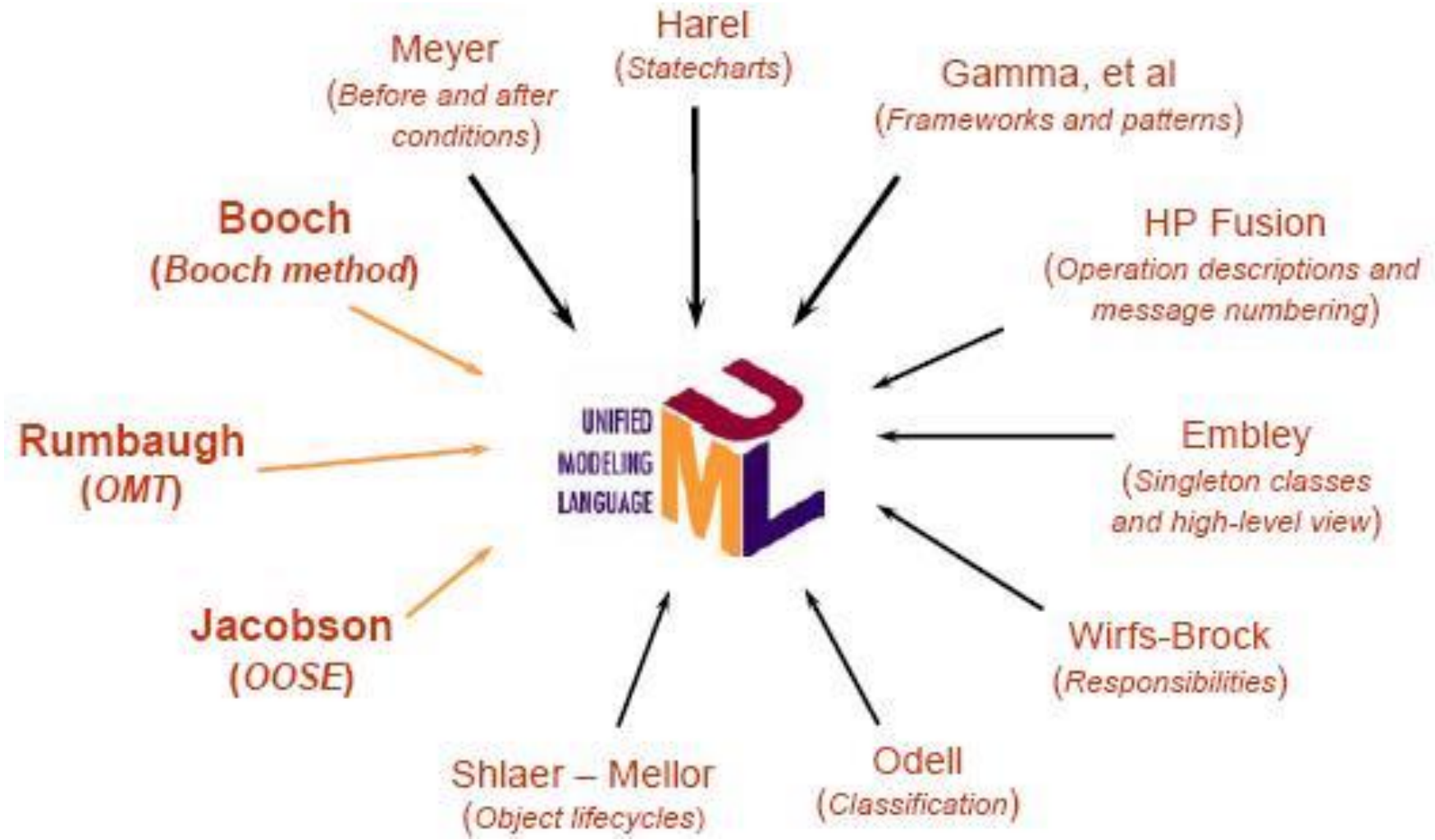
- Surgiu na Rational Software Corp. (1994)
  - Grady Booch, James Rumbaugh e Ivar Jacobson
- Padronização OMG (1997)

Introdução

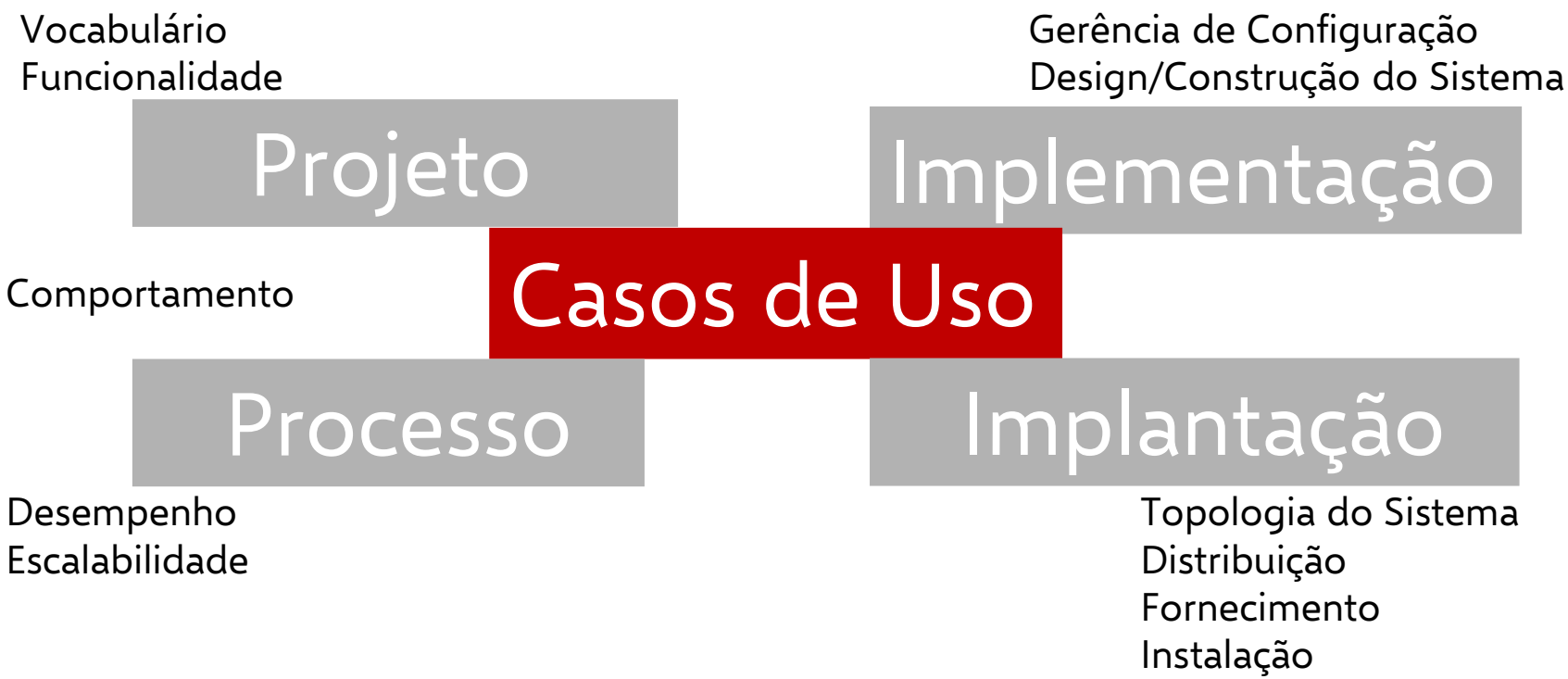


Before 95' - Fragmentation ▶ 95' - Unification ▶ 98' - Standardization ▶ 99' - Industrialization

Introdução


Meyer
(Before and after conditions)
Harel
(Statecharts)
Gamma, et al
(Frameworks and patterns)
Booch
(Booch method)
HP Fusion
(Operation descriptions and message numbering)
Rumbaugh
(OMT)
UNIFIED MODELING LANGUAGE
Embley
(Singleton classes and high-level view)
Jacobson
(OOSE)
Wirfs-Brock
(Responsibilities)
Shlaer – Mellor
(Object lifecycles)
Odell
(Classification)

Visões


Vocabulário
Funcionalidade
Gerência de Configuração
Design/Construção do Sistema
Projeto
Implementação
Comportamento
Casos de Uso
Processo
Implantação
Desempenho
Escalabilidade
Topologia do Sistema
Distribuição
Fornecimento
Instalação

Diagramas UML

Diagramas UML

- Estruturais
  - Classes, Objetos, Pacotes e Subsistemas, Estrutura Composta*, Componentes e Implantação

- Comportamentais
  - Máquina de Estados, Casos de Uso, Atividades, Seqüência, Comunicação, Interação Geral* e Tempo*

Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Casos de Uso

- Representam regras de negócio
  - Usa perspectiva de atores
- Capturam requisitos
  - Atores necessitam dos requisitos para atuar no negócio
- Diagrama mais abstrato, flexível e informal
  - Serve de base para outros diagramas
  - Embora representado como diagrama UML, é um artefato textual

Atores

- Usam/interagem com o Sistema
  - Qualquer elemento externo que interaja com o sistema
- Podem ser usuários ou outros sistemas
- Diferenças entre Atores e Usuários:
  - Uma única pessoa pode assumir papéis diferentes
  - Atores representam papéis e usuários podem representar o negócio

Associações

- Interações de atores com casos de uso

Cenários

- Descrição das atividades de um caso de uso (documentação)
- Descreve o que fazer, não como!
- Perspectiva do cliente

1. Cadastrar Usuários

Caso de Uso: 1.1 Incluir Usuário
Atores: Administrador do Sistema
Pré-Condições: Existir uma prefeitura cadastrada

Ações do Ator	Resposta do Sistema
1. Este caso de uso inicia quando o ator seleciona "Cadastrar Usuários" no menu	
	2. Sistema apresenta tela de cadastro de usuários
3. Ator preenche os dados do usuário (nome de usuário, login, senha, tipo de usuário – Administrador, Supervisor, Atendente) para cadastro e solicita gravação	
	4. Sistema registra gravação do usuário (sua Prefeitura é identificada)

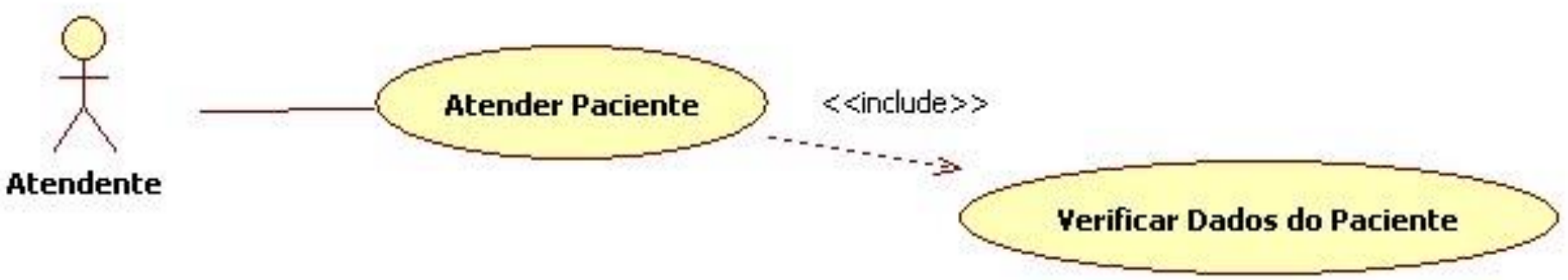
Alternativa 1: No passo 3, quando usuário solicita gravação, são validadas o preenchimento dos dados e a senha confirmada.
Alternativa 2: No passo 4, antes de efetuar gravação, são validados login e senha do usuário (não pode haver mais de um usuário com mesmo login e senha)
Pós-Condições: Usuário cadastrado no sistema

Caso de Uso: 1.2 Consultar Usuário
Atores: Administrador do Sistema
Pré-Condições: Possuir pelo menos um usuário cadastrado no sistema

Ações do Ator	Resposta do Sistema
1. Este caso de uso inicia quando o ator seleciona no "Cadastro de Usuários" a opção "Consultar"	
	2. Sistema lista todos os usuários cadastrados para a Prefeitura, identificando nome, login e tipo de

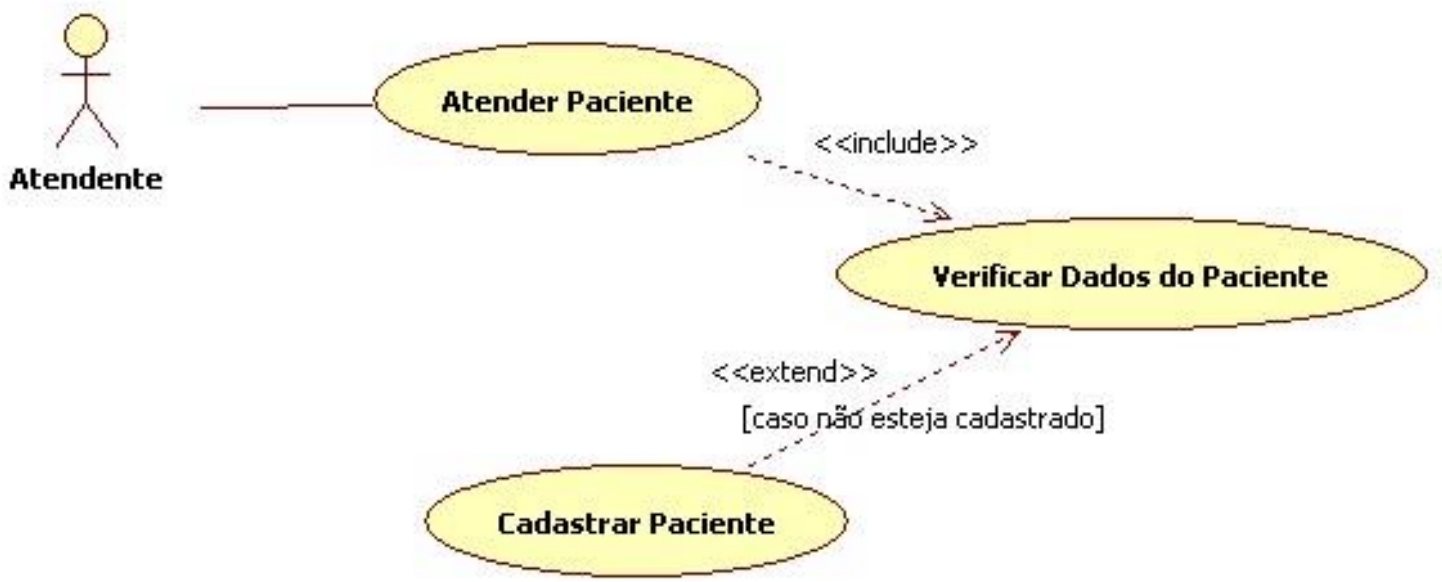
Associações <<include>>

- Relação de uso, quando um caso de uso usa/inclui funcionalidades de outro
- O caso de uso que é chamado não tem a referência de quem o incluiu
- Só é relevante esta associação se for compartilhado entre outros casos de uso



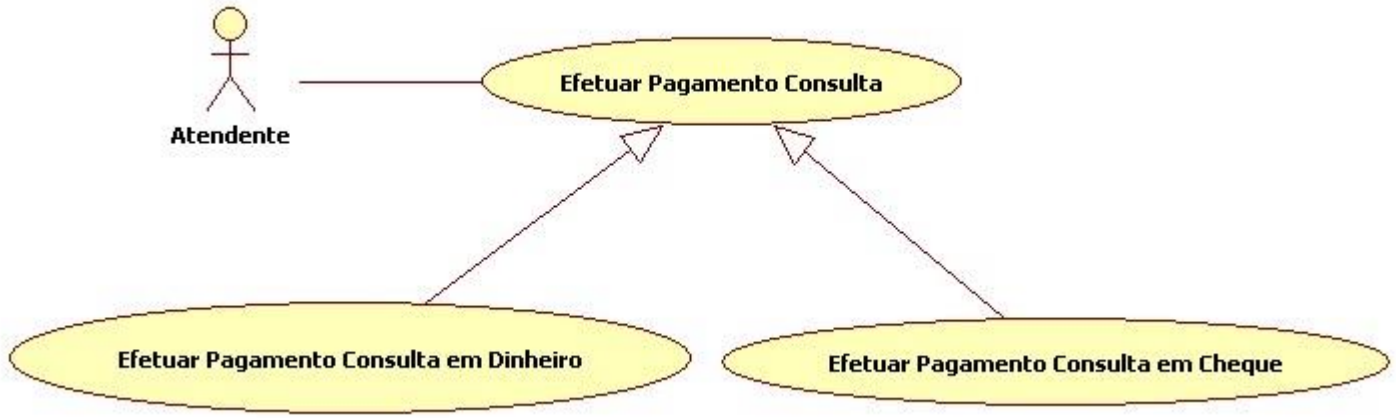
Associações <<extends>>

- Relação de uso, quando um caso de uso usa/inclui funcionalidades de outro, de forma opcional
- O caso de uso estendido sabe como inserir seu extensor no fluxo de eventos



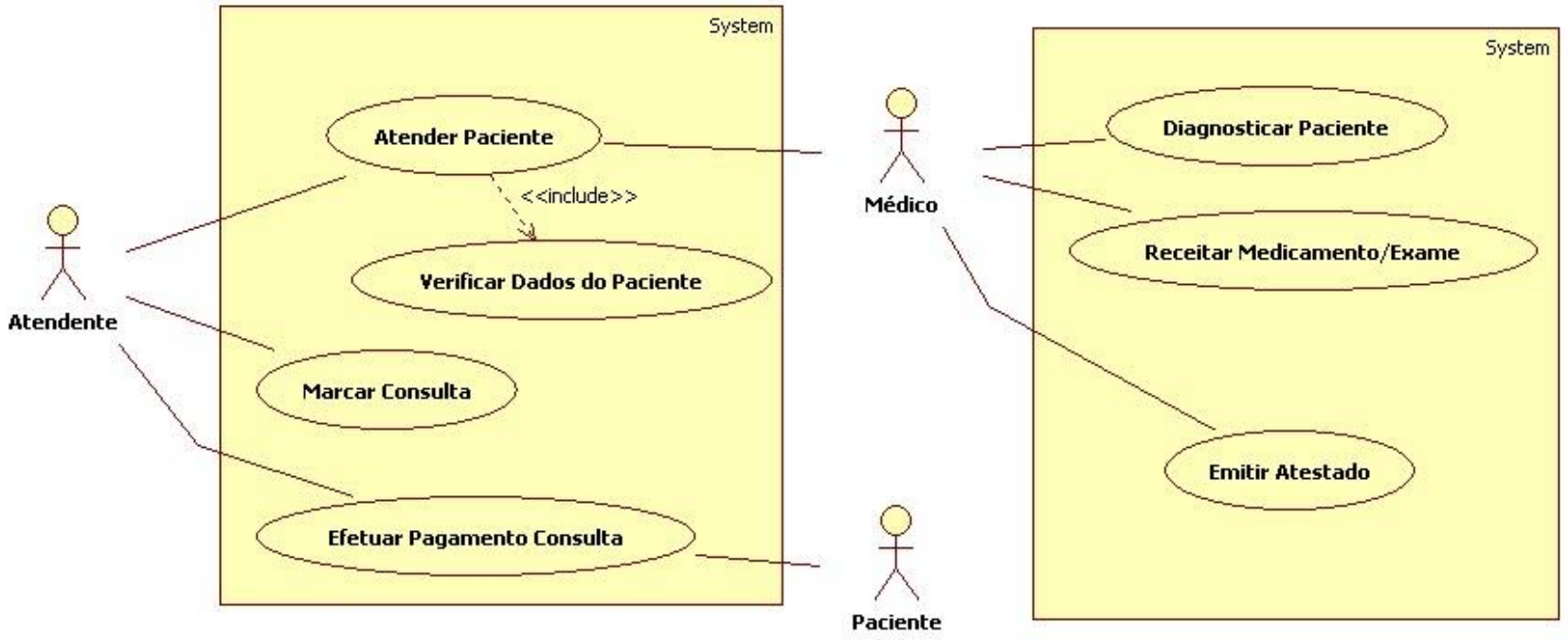
Associações do tipo Herança

- O caso de uso filho pode:
  - herdar todos os relacionamentos, atividades e todos os pontos de extensão do caso de uso pai
  - acrescentar suas próprias atividades, relacionamentos e pontos de extensão do caso de uso pai
  - Estender as atividades, relacionamentos e pontos de extensão do caso de uso pai
  - Nota: Pode-se ter herança de atores



Subsistemas

- “Dividir para conquistar”!
- Problemas complexos devem ser simplificados
- Produzir diagramas de caso de uso que ressaltem as fronteiras do negócio



Dicas para Casos de Uso

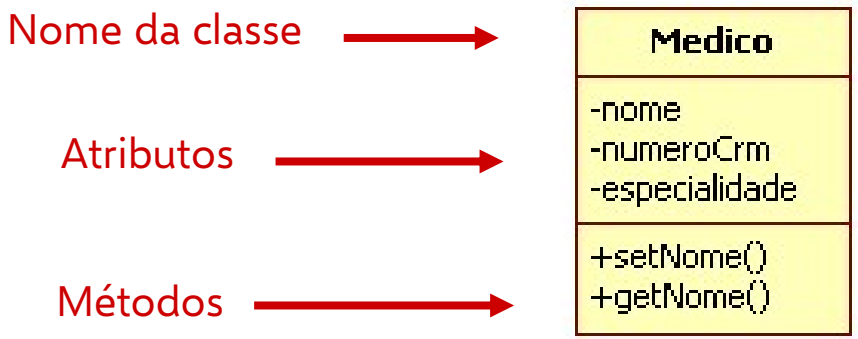
- Passos:
  - Identificar os atores
  - Identificar visões/responsabilidades
  - Identificar relações em comum (tipos de associações)
- Evite:
  - Abusar de <<include>> e <<extend>> (somente quando houver um compartilhamento)
  - Descrever requisitos físicos de implementação/detalhes de interface
  - Abusar da abstração

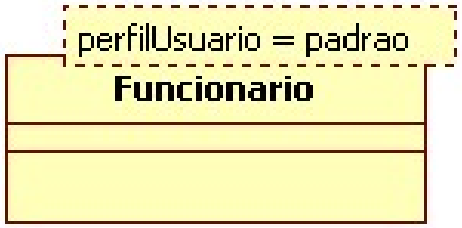
Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Classes e Objetos

- Representam a estrutura estática do sistema
- Ilustram os conceitos importantes do domínio do problema, suas características e comportamento
- Projetam uma solução de implementação do problema





Visibilidade

- Separa/concede o acesso a atributos/métodos de uma classe
  - (+) Public: Acesso sem restrição para atributos/métodos
  - (#) Protected: Objetos definidos a partir de subclasses de uma superclasse que possui o atributo/método têm acesso a ele
  - (-) Private: Somente objetos de mesma classe podem ter acesso/invocar um atributo/método privado
  - (~) Package: Somente objetos contidos em um mesmo pacote podem acessar um atributo/método neste nível

Atributos

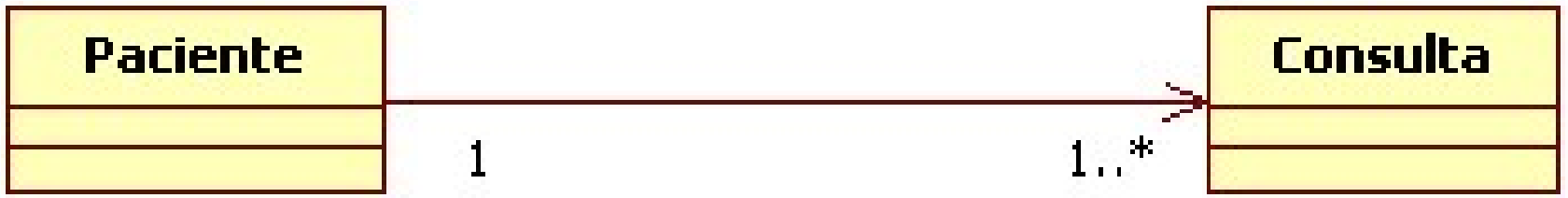
- Pertencem ao domínio do problema
- Possuem tipificação
- Dão características as classes
- Evoluem junto com o diagrama de classe, conforme a fase de projeto em que se encontra
- Atributos de classe: constantes, variáveis estáticas

Métodos

- Comportamento e responsabilidade das classes
- Também são registrados no diagrama de classes (porém são registrados em projeto)
- Podem executar operações ou retornar valores
- Dois tipos: de classe (referência) e de objeto (valor)

Associações

- Relacionamentos entre classes
- Aumentam a legibilidade do modelo



Papéis

- Nome dado ao relacionamento entre as classes, identificando sua forma de participação

Multiplicidade

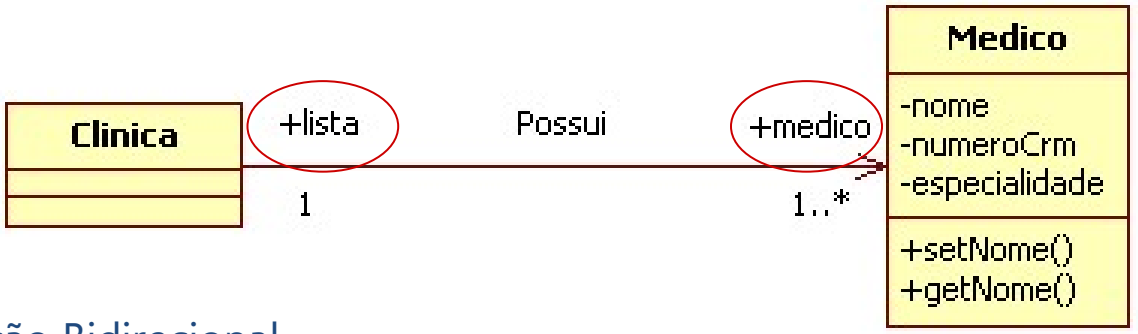
- * = zero ou mais
- 1..* = um ou mais
- 1..40 = um a quarenta
- 5 = exatamente cinco
- 2,4,6 = exatamente dois, quatro ou seis

Encontrando Associações

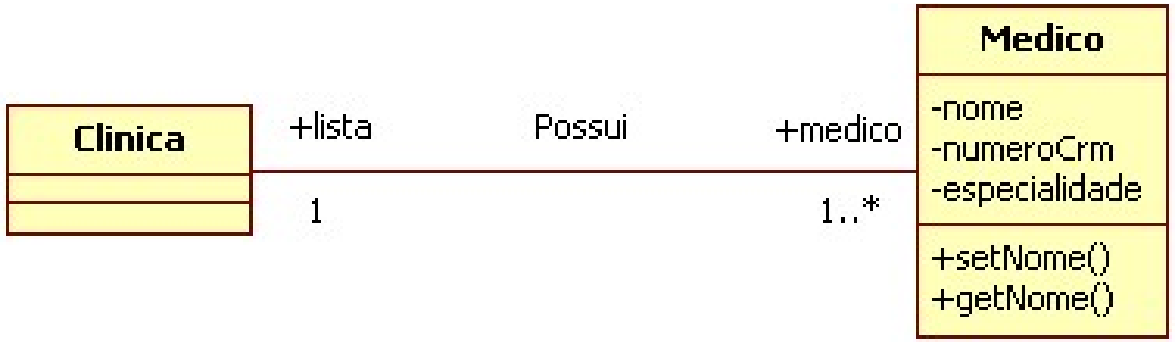
- A é parte de B
- A é um detalhe de B
- A usa serviços de B
- A se comunica com B
- A acessa B

Atributos e Associações

- Uma associação é representada por um atributo na classe de origem da associação
- Contém a identificação da classe destino

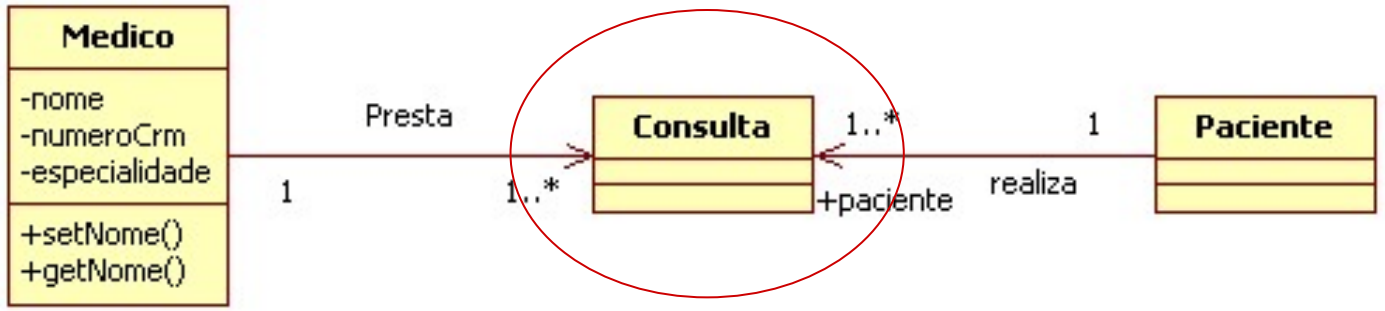


Associação Bidirecional



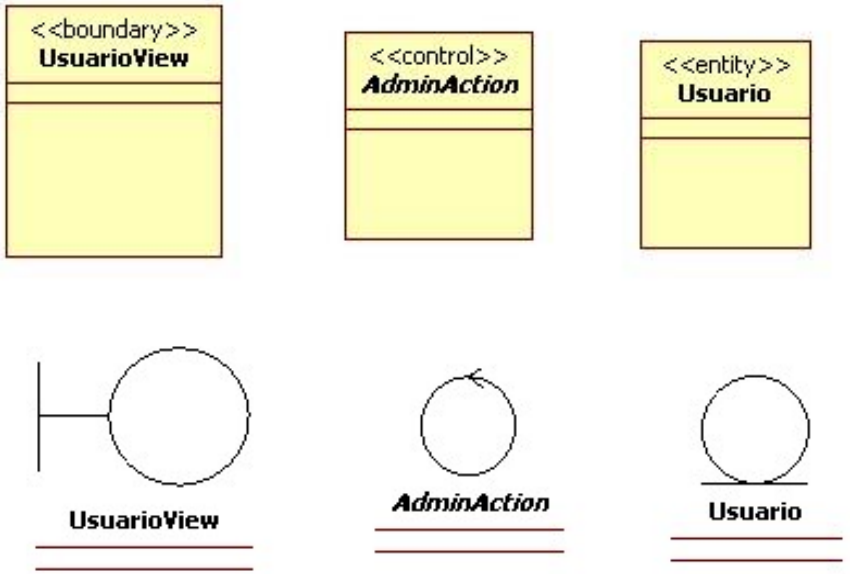
Classes e Associações

- Associações complexas entre classes podem necessitar guardar algumas informações sobre a associação



Esteriótipos/*Profiles*

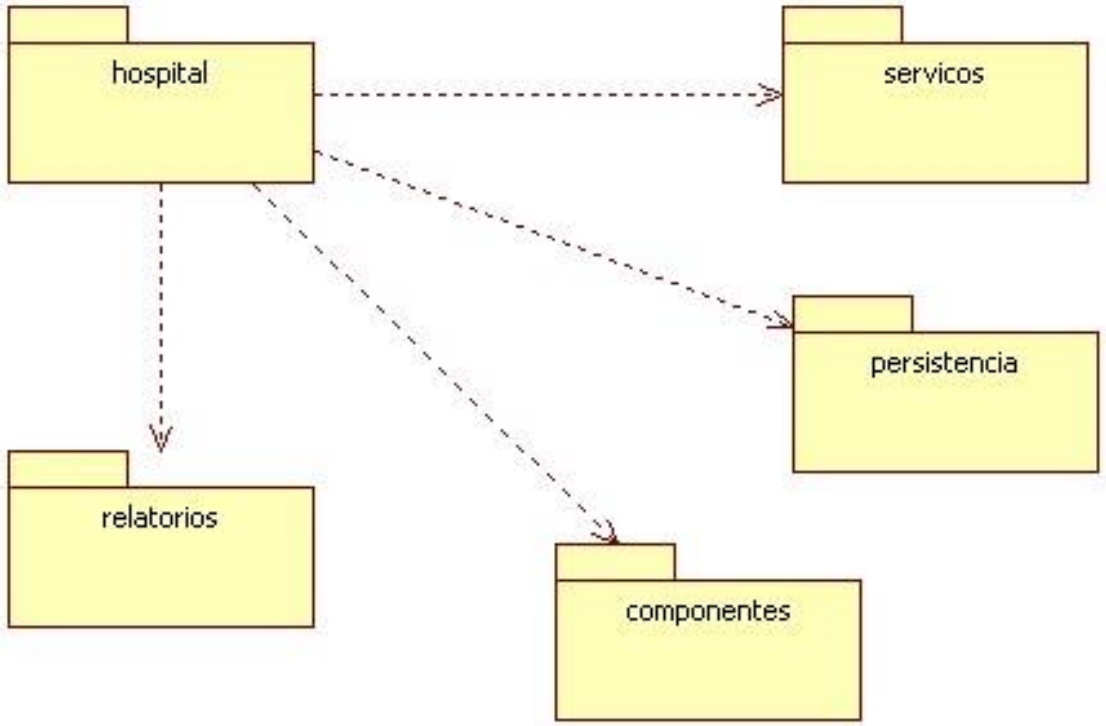
- São recursos da UML que indicam um comportamento específico
- É um metatipo
- Existem alguns esteriótipos conhecidos (<<include>>, <<extend>>, <<type>>, <<SessionBean>>, <<Http_Servlet>>...)



Dependências

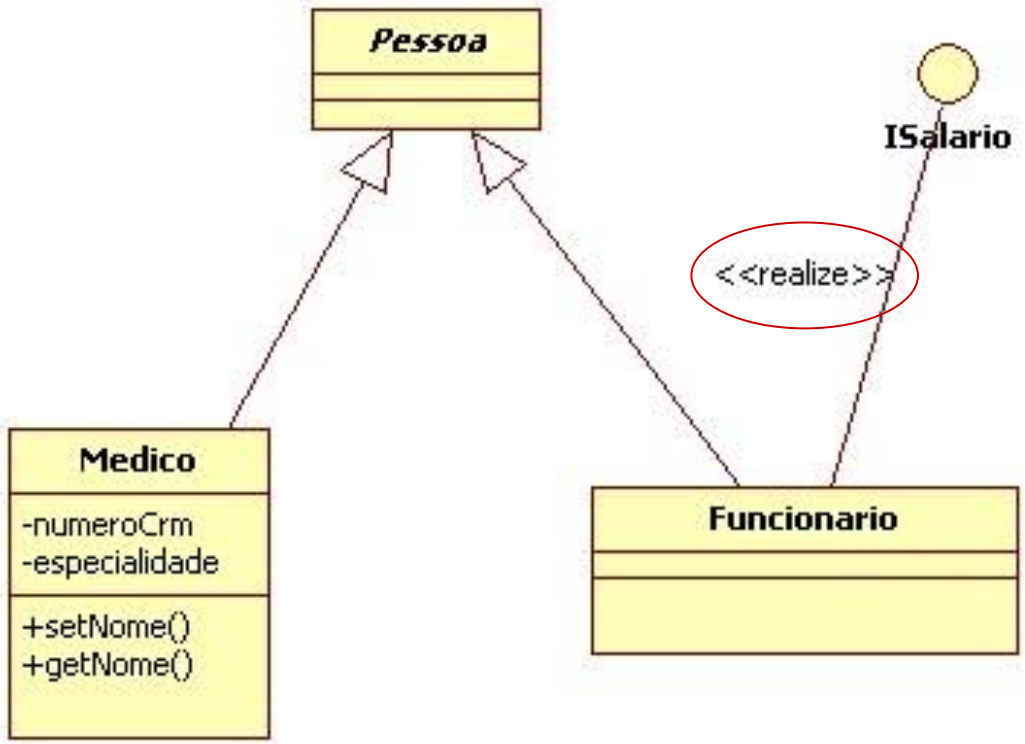
- É um outro tipo de associação, indicando que uma classe/pacote/subsistema utiliza serviços de outra (podem virar futuras associações)

- Acomplamento e Coesão



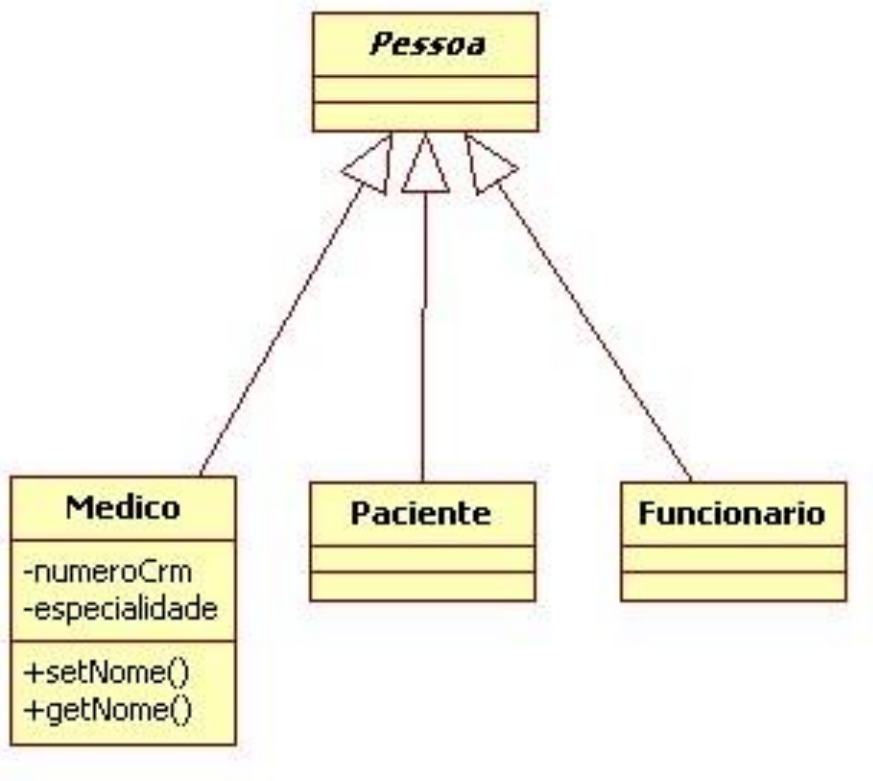
Realização/Implementação

- É uma implementação de interfaces
- Mistura relacionamentos de dependência e generalização
- Utiliza o esteriótipo <<realize>>



Herança

- Tipo de relação de classes onde uma classe filho (subclasse) herda características e comportamento da classe pai (superclasse)
- Importante: Não confundir herança com transmutação de objetos (papéis)



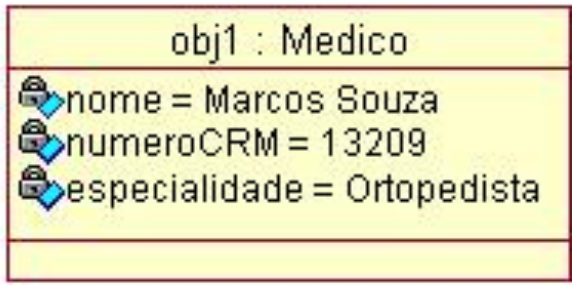
Agregação e Composição

- Agregação
- Tipo de associação todo-parte onde a existência da parte não está condicionada à existência do todo

- Composição
- Tipo de associação todo-parte onde a existência da parte está condicionada à existência do todo

Diagrama de Objetos

- Complemento do diagrama de classes
- Fornece uma visão dos valores armazenados pelos objetos, em um dado tempo de execução
- Utilizados para simular um estado e comportamento de objetos

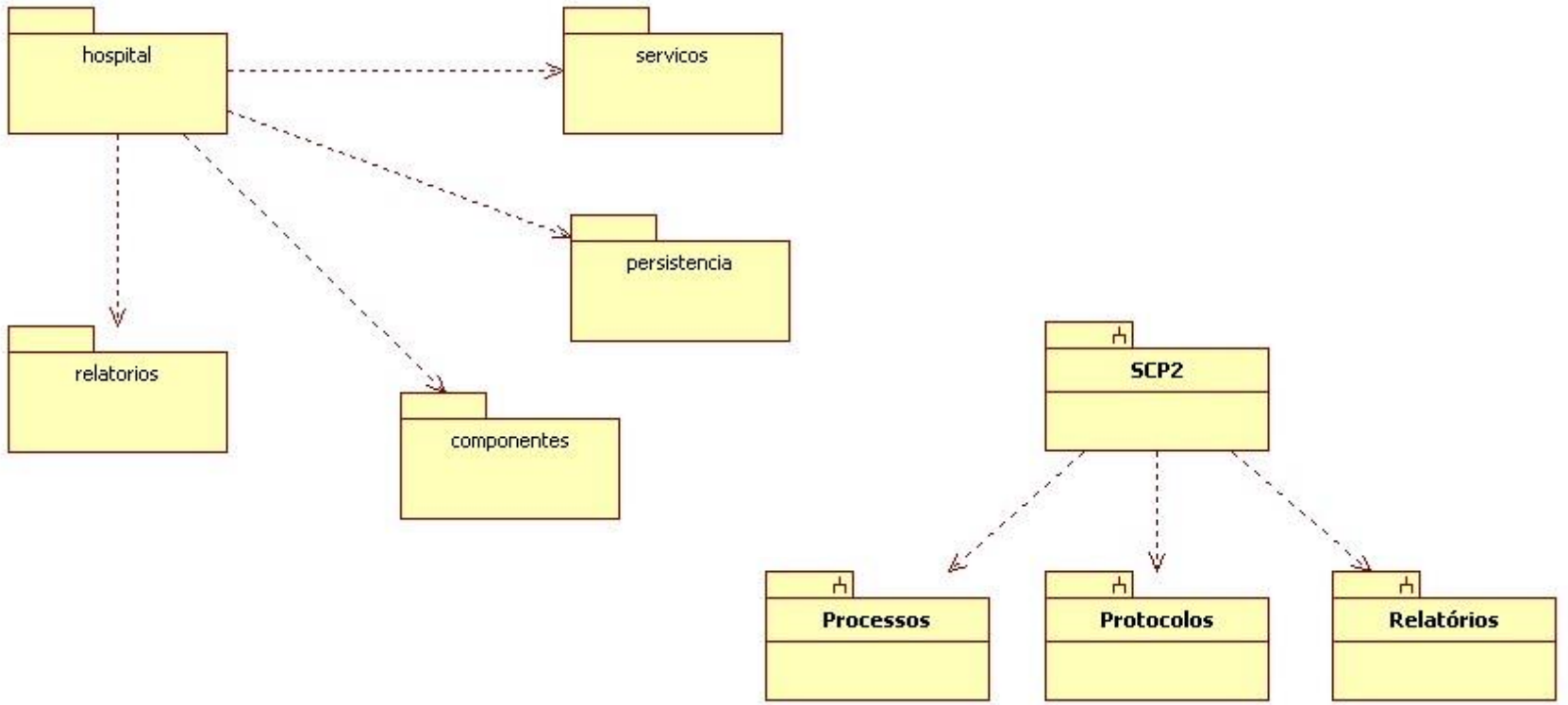


Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Pacotes e Subsistemas

- Aumentam a capacidade de compreensão do negócio, organizando o modelo
- Representam estruturas hierárquicas (subsistemas) e suas relações



Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Diagrama de Seqüência

- Apresenta duas dimensões
  - temporal (vertical) e objetos (horizontal)
- Muito utilizado para identificação de métodos das classes
- Nas interações usam-se “*” e condições são descritas entre colchetes
- Ajuda a mapear a interface do sistema
  - Elementos: Atores, objetos, linha de vida, Mensagens (auto-chamada, síncrona, assícrona e retorno)

Diagrama de Seqüência

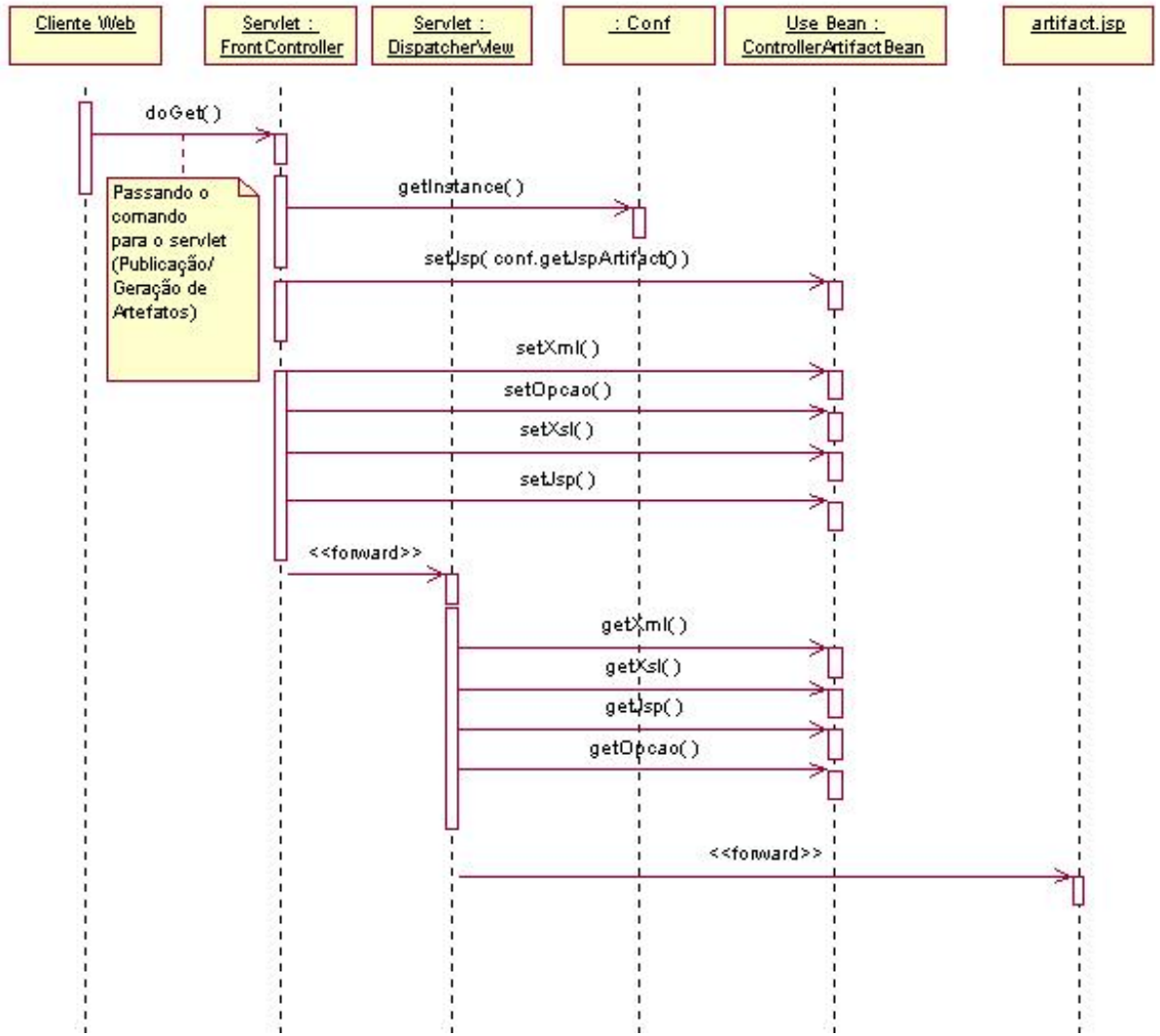


Diagrama de Seqüência

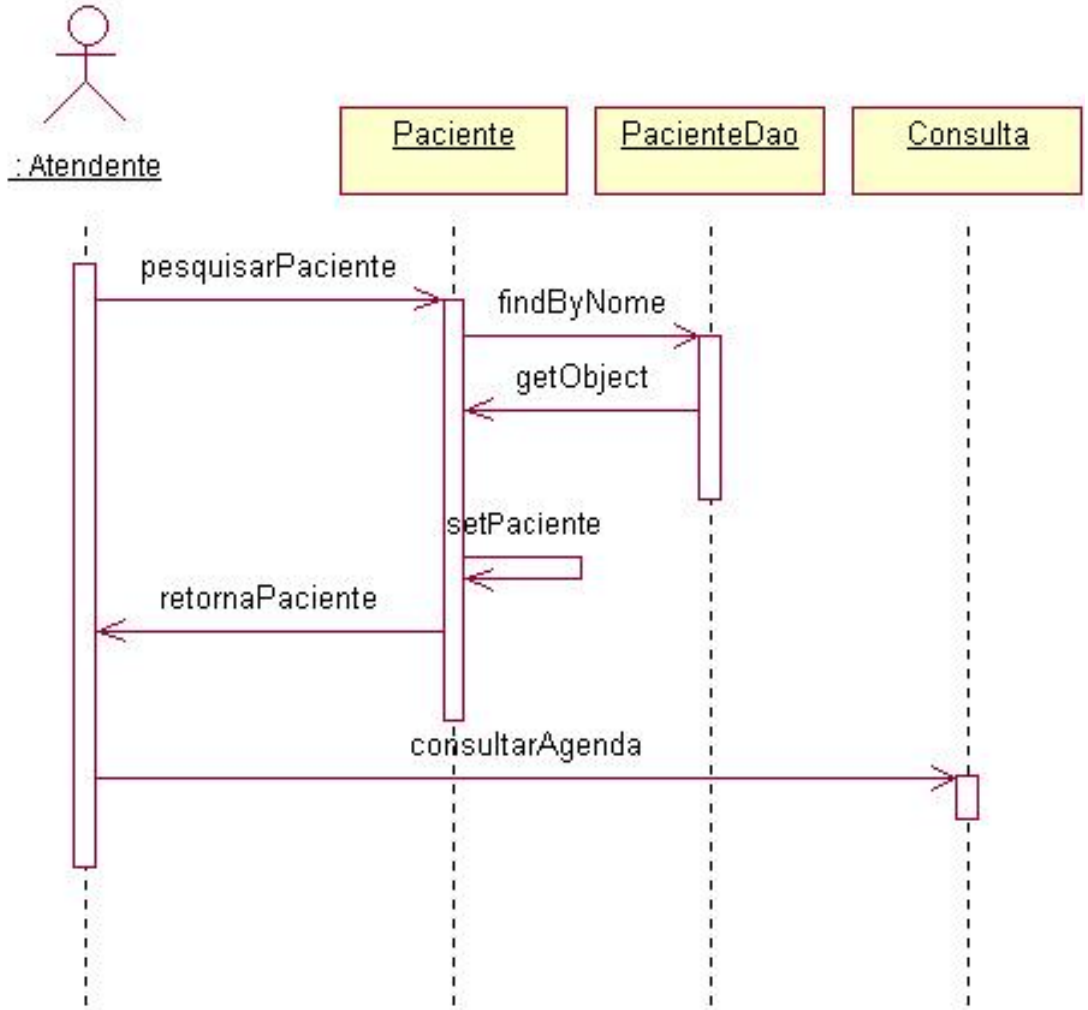


Diagrama de Seqüência (UML2)

- Fragmentos Combinados e Operadores de Interação
  - Problemas com testes, laços e processos paralelos
- Permite uma modelagem semi-independente, focado nos problemas relatados
- São delimitados por um retângulo definindo seu contexto de atuação dentro do diagrama
- Alguns exemplos de Operadores de Interação
  - Opt (Option), Alt (Alternatives), Par (Parallel), Loop (Looping), Break,...
- Representam com maior eficácia os laços e decisões que precisam ser expressas nos diagramas

Diagrama de Seqüência (UML2)

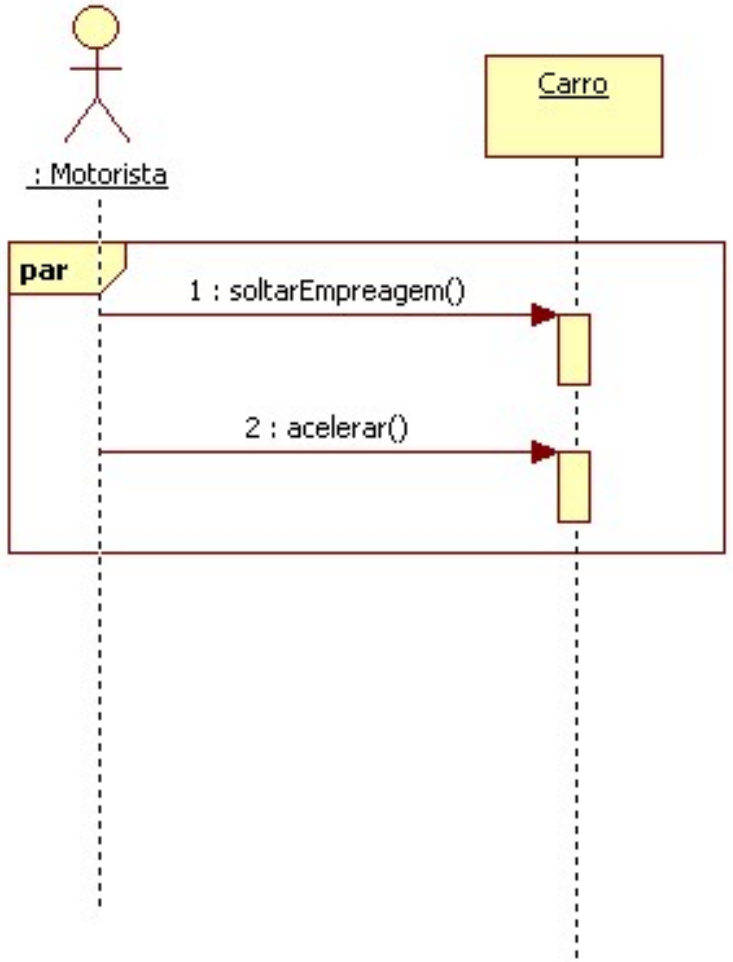


Diagrama de Seqüência (UML2)

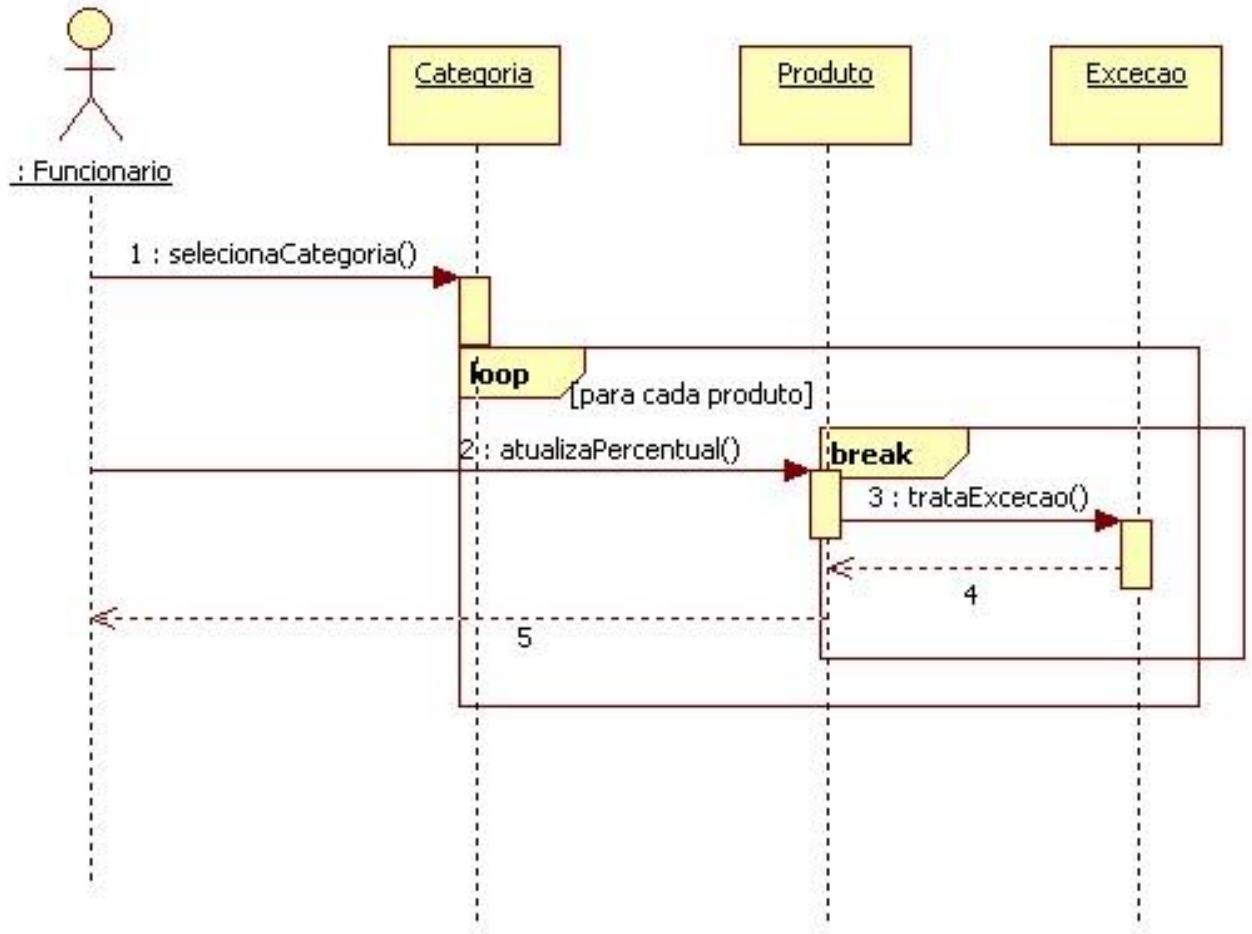
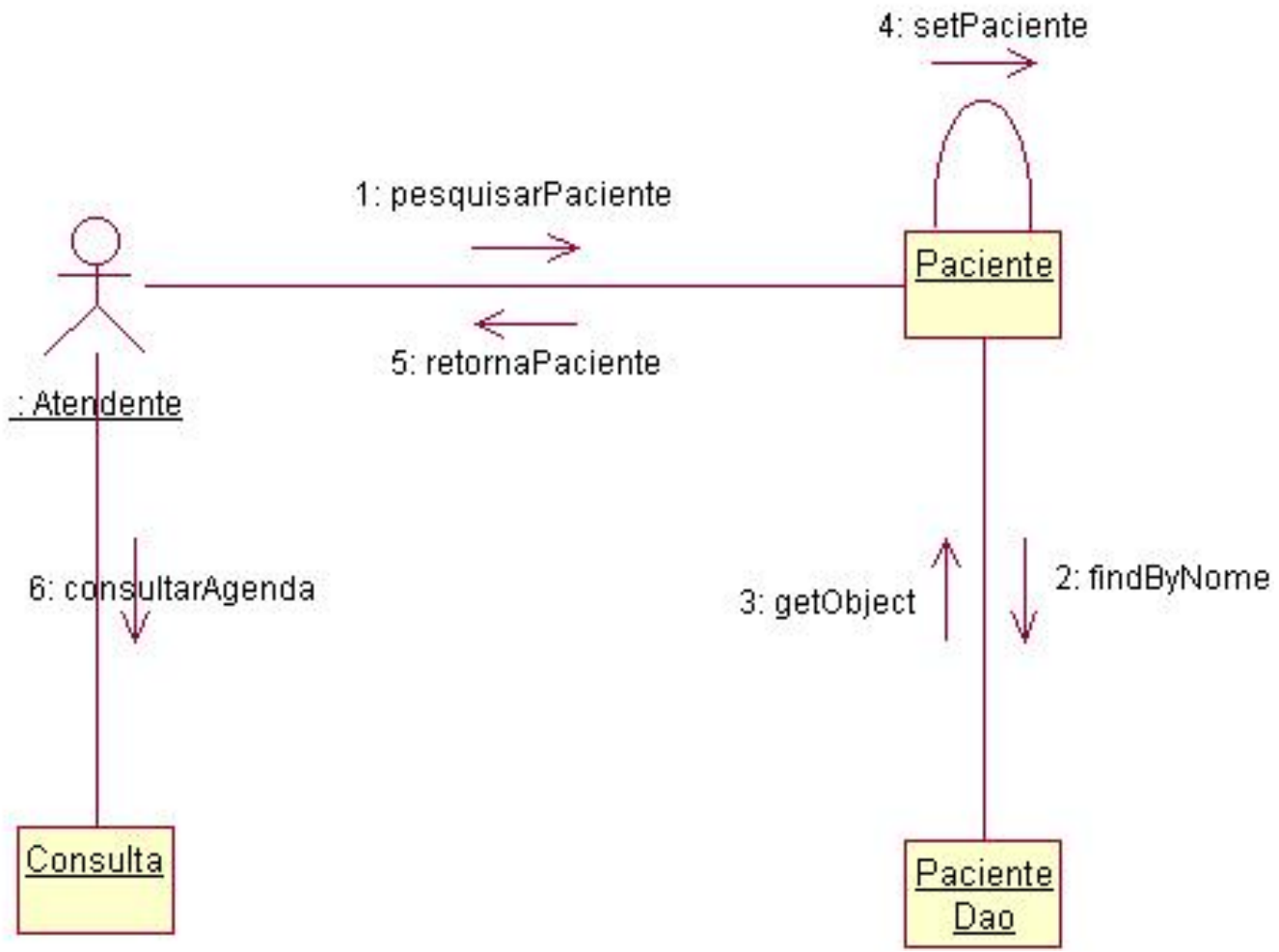


Diagrama de Comunicação

- Conhecido anteriormente como diagrama de colaboração (UML 1.5)
- Amplamente associado ao diagrama de seqüência
- Não se preocupam com a temporalidade do processo
- Não usam os fragmentos combinados e operadores de interação

Diagrama de Comunicação



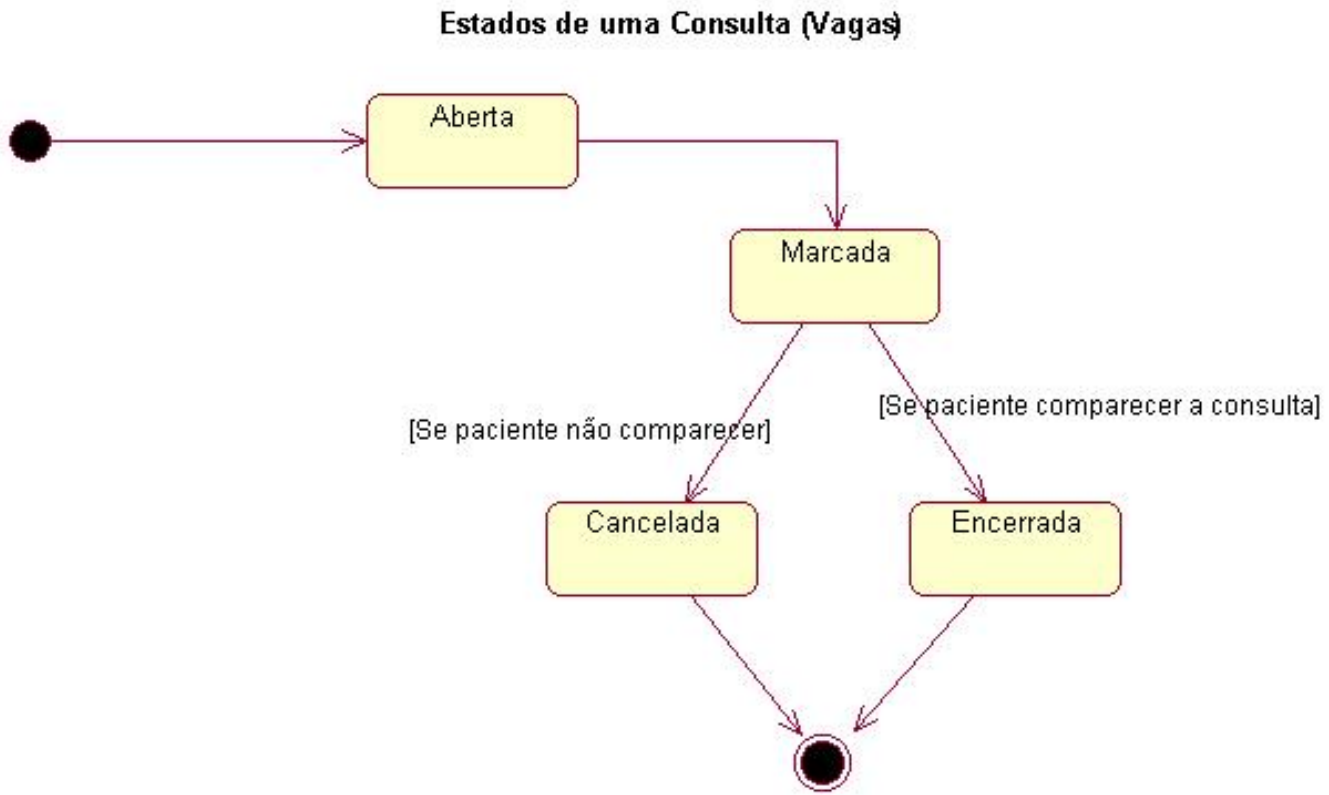
Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Máquina de Estados

- Era conhecido como diagrama de estados em versões anteriores da UML
- Demonstra o comportamento de um elemento através de transições de estado
- O elemento modelado pode ser variado (instância, caso de uso...)
  - Elementos: Início, estados, transições, Atividades internas (Entry, Do, Exit), Fim
- Barras de sincronização/junção/estados compostos

Máquina de Estados



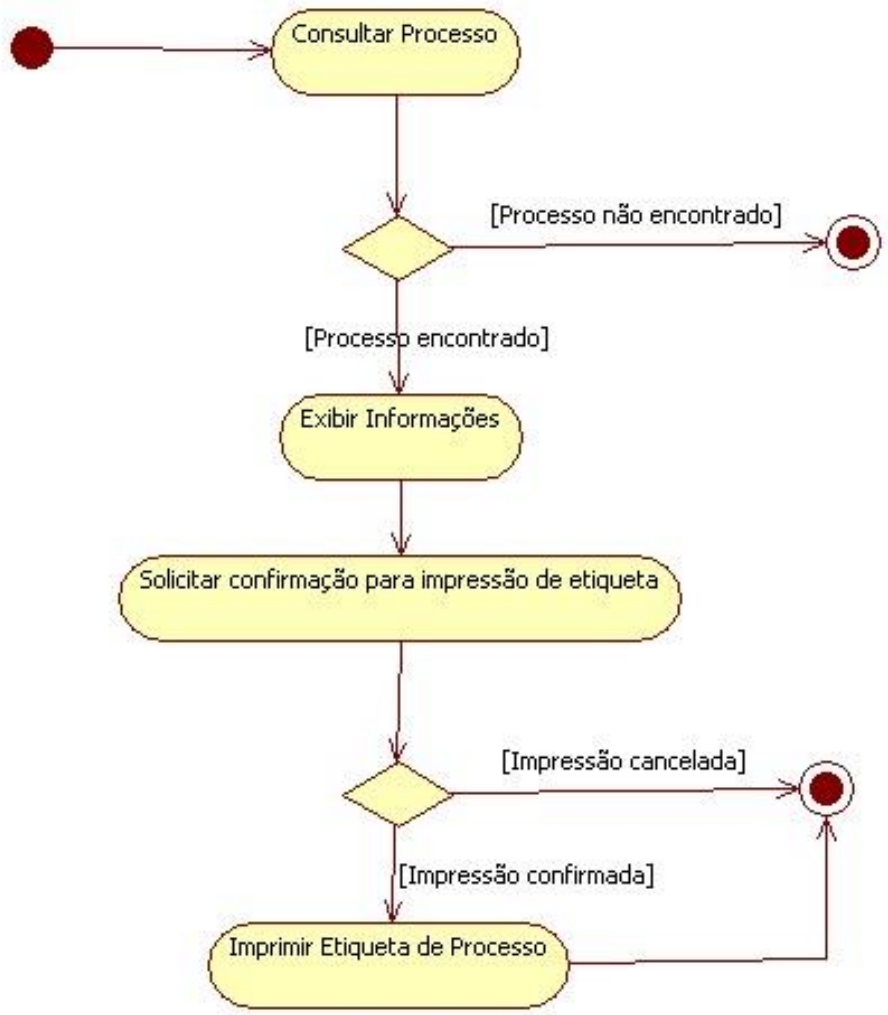
Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Diagrama de Atividades

- Tipo especial de diagrama de máquina de estados
- Exibe o fluxo de uma atividade para outra
- Abrange a visão dinâmica das atividades, dando ênfase no fluxo de controle dos objetos
- Ótimo recurso para modelagem de workflows
  - Elementos: Início, Nós de ação, decisão, controle de fluxo (setas), fim

Diagrama de Atividades



Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Diagrama de Componentes

- Componentes são partes físicas e substituíveis de um sistema
- Um SW abrange diferentes tipos de componentes (implantação, artefatos, código)
- Este diagrama serve para organizar as dependências existentes entre eles
- Abrange uma visão estática da implementação

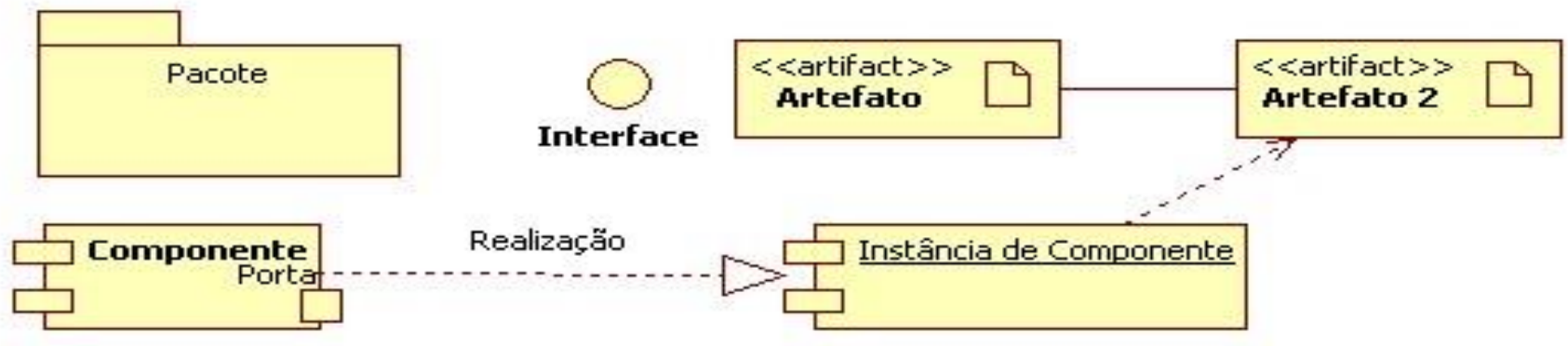


Diagrama de Componentes

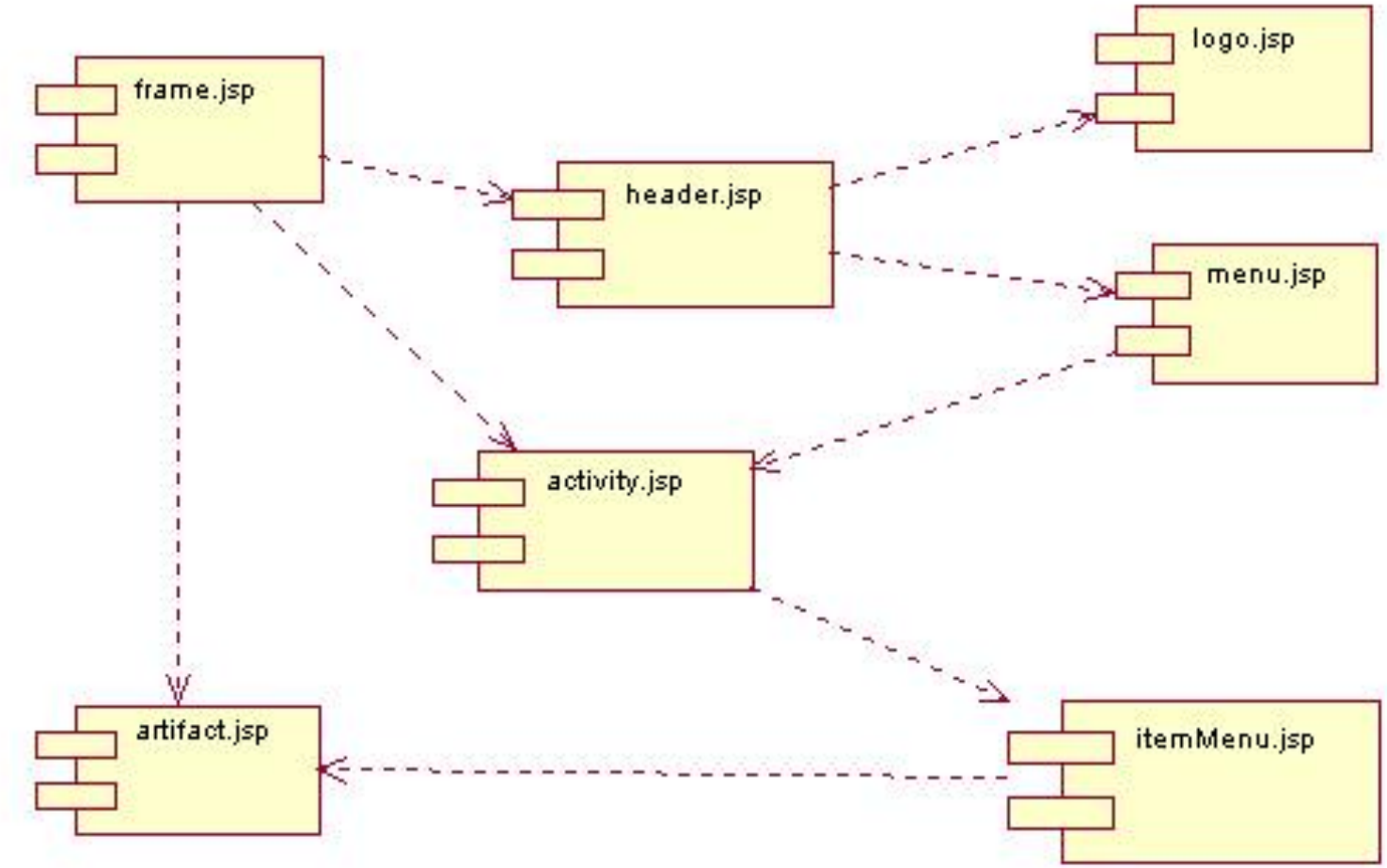
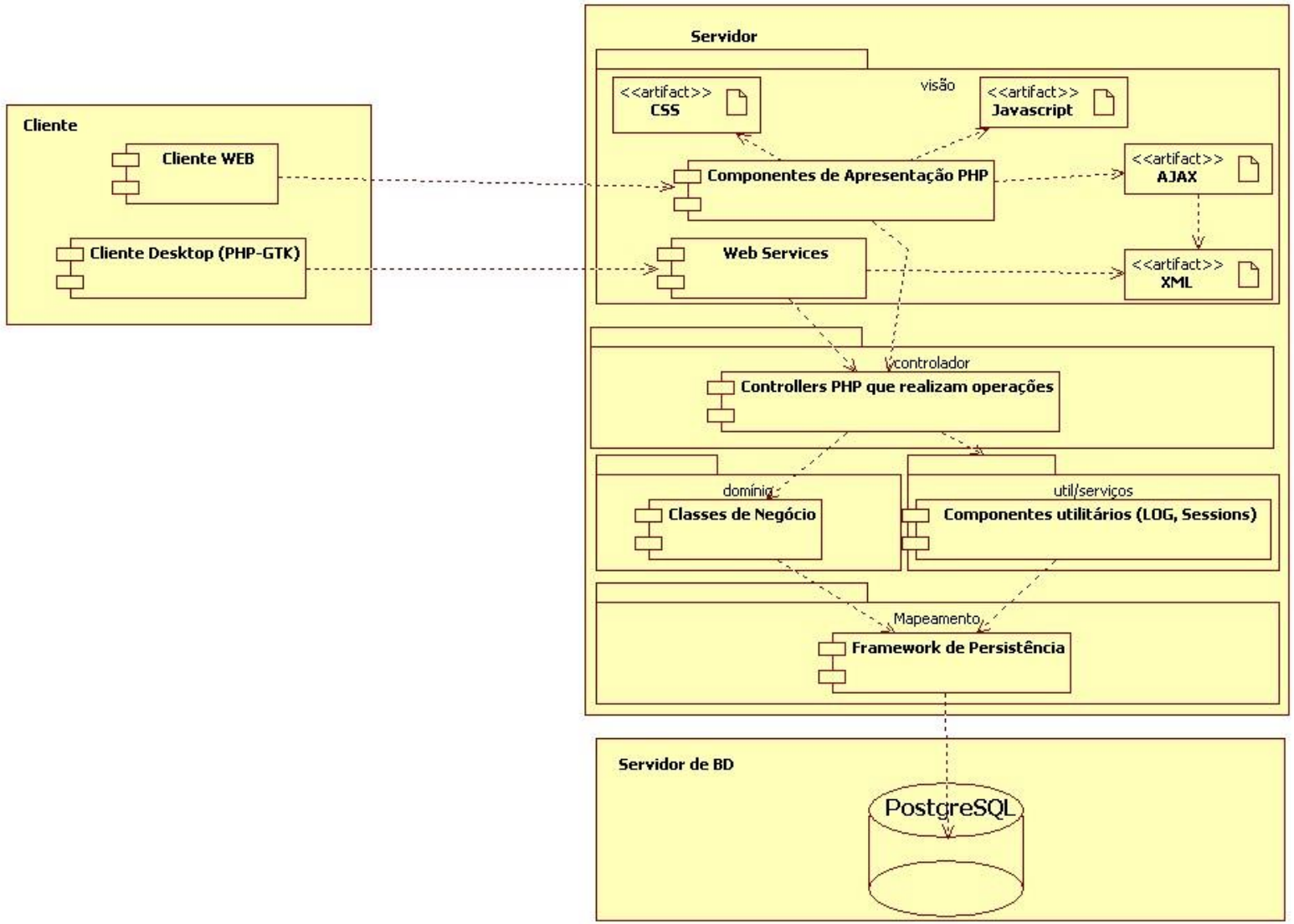


Diagrama de Componentes

Diagrama de Componentes



Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Diagrama de Implantação

- Expressa a organização física dos componentes
- Recurso para definição de arquitetura do SW
- Define hardware, protocolos, softwares...

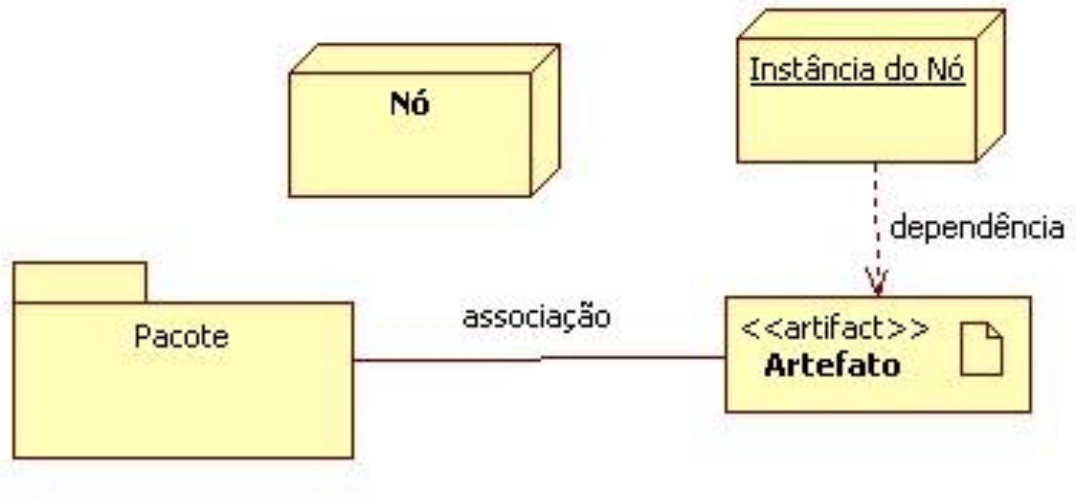
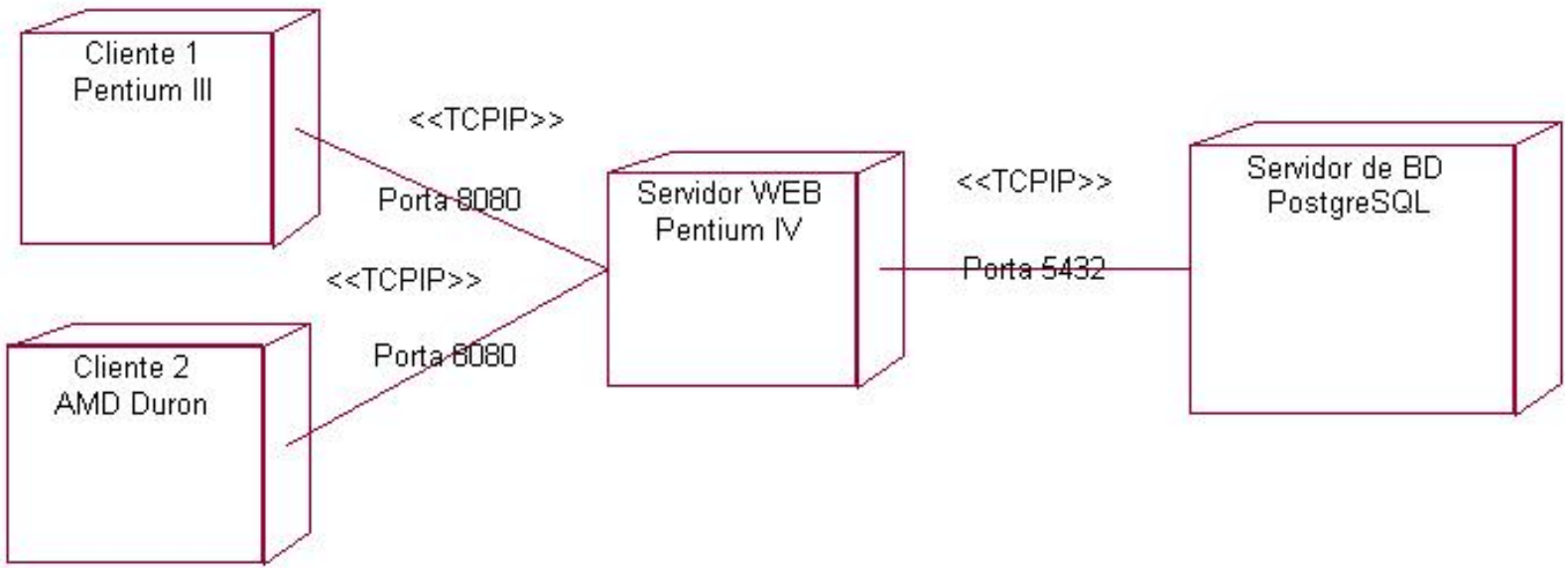


Diagrama de Implantação



Diagramas UML

- Casos de Uso
- Classes
- Pacotes e Subsistemas
- Seqüência e Comunicação
- Máquina de Estados
- Atividades
- Componentes
- Implantação
- Estrutura Composta, Interação Geral, Tempo*

Diagrama de Estrutura Composta

- Utilizado para modelar colaborações

- Expressa tempo de execução, padrões de uso e relacionamento dos elementos
  - A colaboração é representada por uma elipse
  - Os papéis das colaborações definem o uso das instâncias

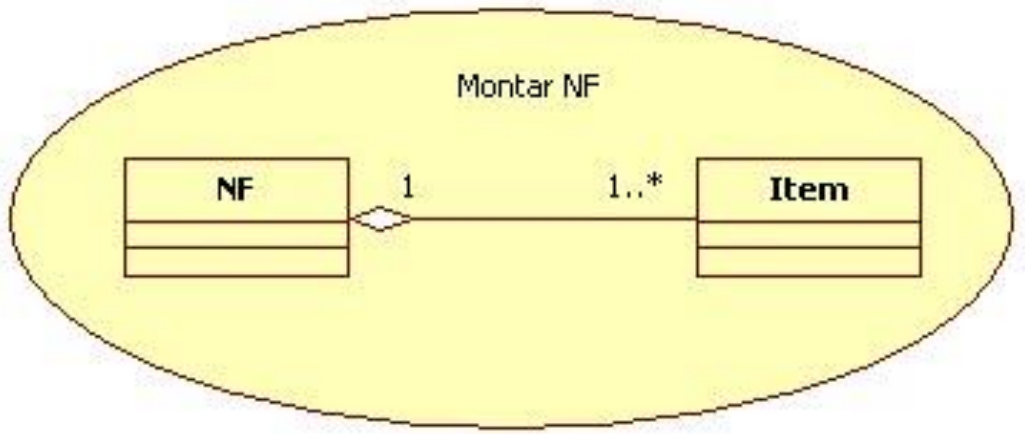


Diagrama de Interação Geral

- É uma variação do diagrama de atividades
- Fornece uma visão geral do controle de fluxo
- Pode ser uma combinação de diagramas
  - A partir do diagrama de atividades, pode-se fazer uma chamada para uma diagrama de seqüência

Diagrama de Interação Geral

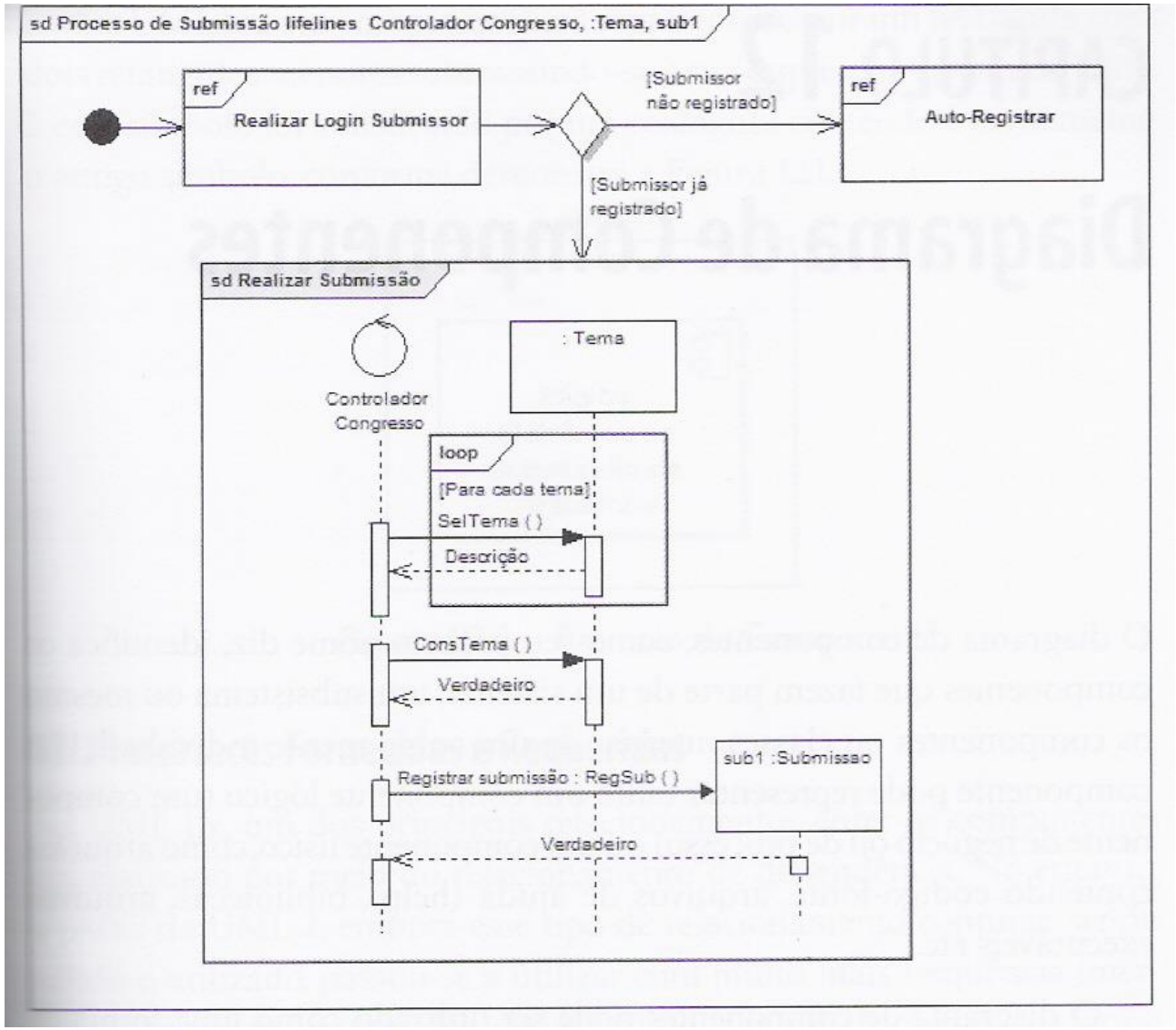
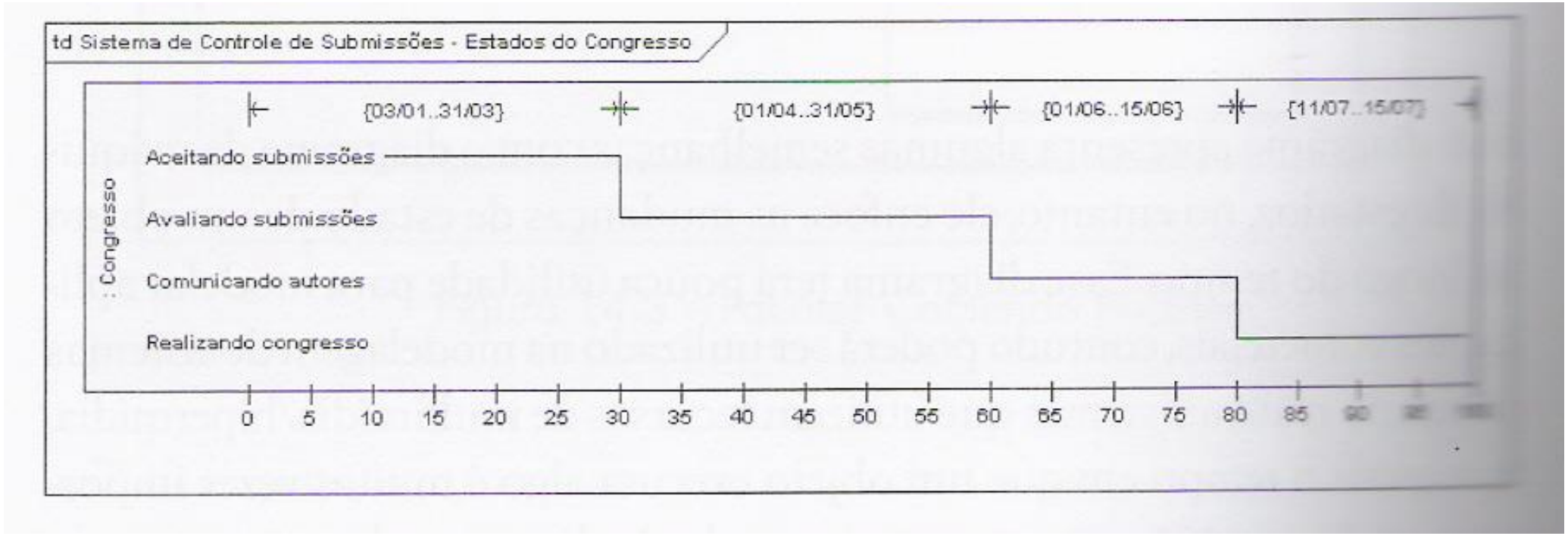


Diagrama de Tempo

- Apresenta semelhança com o diagrama de máquina de estados (seqüência + estados)
- Enfoca a mudança de estado de um objeto ao longo do tempo
- Tem pouca utilidade para aplicações comerciais
  - Mais utilizado em sistemas de tempo real ou que utilizem recursos multimídia/hipermídia

Diagrama de Tempo



Diagramas UML

Ferramentas

- StarUML
  - http://staruml.sourceforge.net/
- ArgoUML
  - http://argouml.tigris.org
- Enterprise Architect
  - http://www.sparxsystems.com.au
- Posseidon UML
  - http://www.gentleware.com
- Rational Rose
  - http://www-306.ibm.com/software/awdtools/developer/rosexde/
- Astah [Indicado!]
  - http://www.astah.net
- Visual Paradigm for UML
  - http://www.visual-paradigm.com
- Umbrello
  - http://www.umbrello.org

Principais Dicas

Dicas

- O que a UML não é...
  - Uma linguagem de programação
  - Metodologia ou Processo
  - Suficiente para modelar completamente um SW
  - Proprietária

- Use a UML...
  - Como base da modelagem
  - Como um subconjunto de notações
  - Para educar os desenvolvedores!

Dicas

- O Software é o objetivo principal
- Adote a simplicidade na modelagem
  - Modele com propósito
  - Modele para entender e comunicar
  - Use mais de um modelo
  - Modele incrementalmente
  - Conheça seus modelos
- O Conteúdo é mais importante que a forma
- Valorize a comunicação
- Modele com as outras pessoas