Especificação de Sistemas de Tempo-Real utilizando Orientação a Objetos

Marco Aurélio Wehrmeister
mawehrmeister@inf.ufrgs.br

Roteiro

- Introdução
- Orientação a Objetos
- UML
- Real-Time UML
- Estudo de Caso: Automação de uma Cadeira de Roda

Introdução

- Complexidade dos sistemas aumenta de acordo com a evolução tecnológica
- Solução para o projeto de sistemas é aumentar o nível de abstração utilizado para representar os requisitos e funcionalidades do sistema

Introdução

- Uma solução encontrada foi utilizar diagramas para representar os requisitos e funcionalidades do sistema
- Duas abordagens para modelagem em alto-nível
  - Análise Estruturada (SA)
  - Orientação a Objetos (OO)

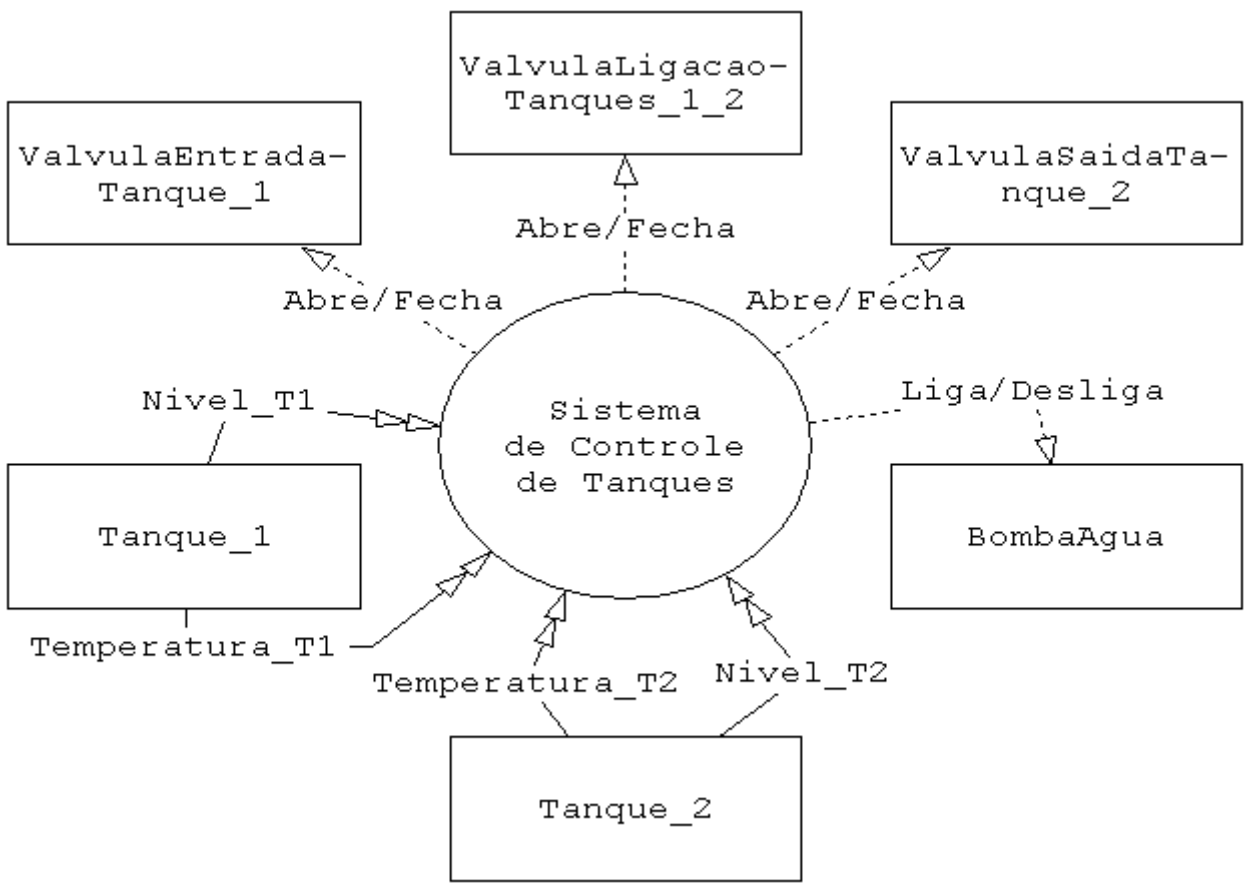
Análise Estruturada - Introdução

- Iniciou em 1970 com DeMarco e Yourdon
- Sistema modelado com base nas suas funcionalidades
- Conceitos de programação baseados em funções e procedimentos
- Estrutura do sistema é descrita através de Diagramas de Fluxo de Dado
- Comportamento é descrito através de Diagramas de Estado

SA/RT – Revisão

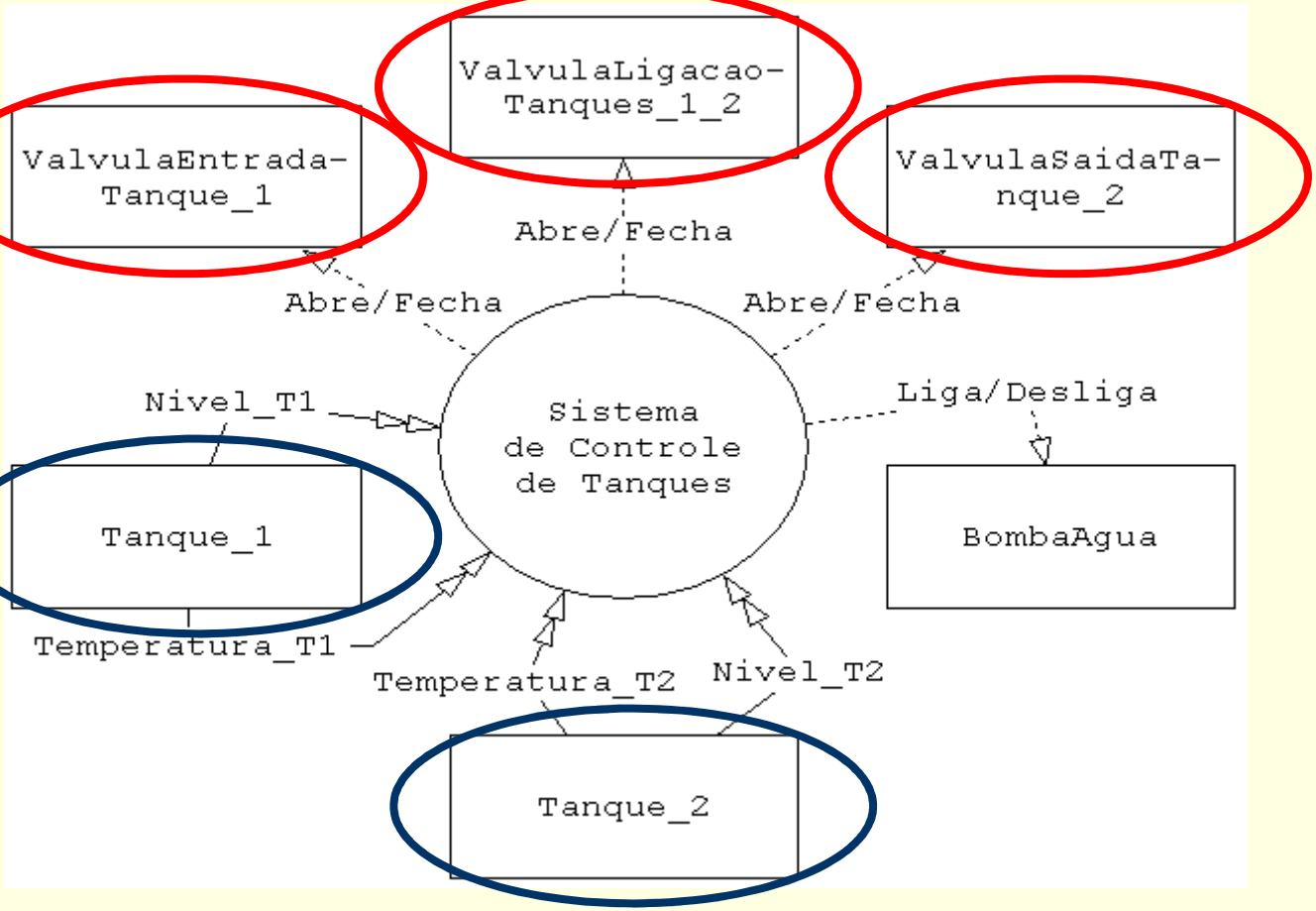
- Modelagem do Problema
  - Modelo do Ambiente
    - Diagrama de Contexto
    - Lista de Eventos
  - Modelo do Comportamento
    - Diagrama de Transformações
    - Diagrama de Informações
- Modelagem da Solução
  - Módulos, tarefas, processadores

SA/RT – Diagrama de Contexto



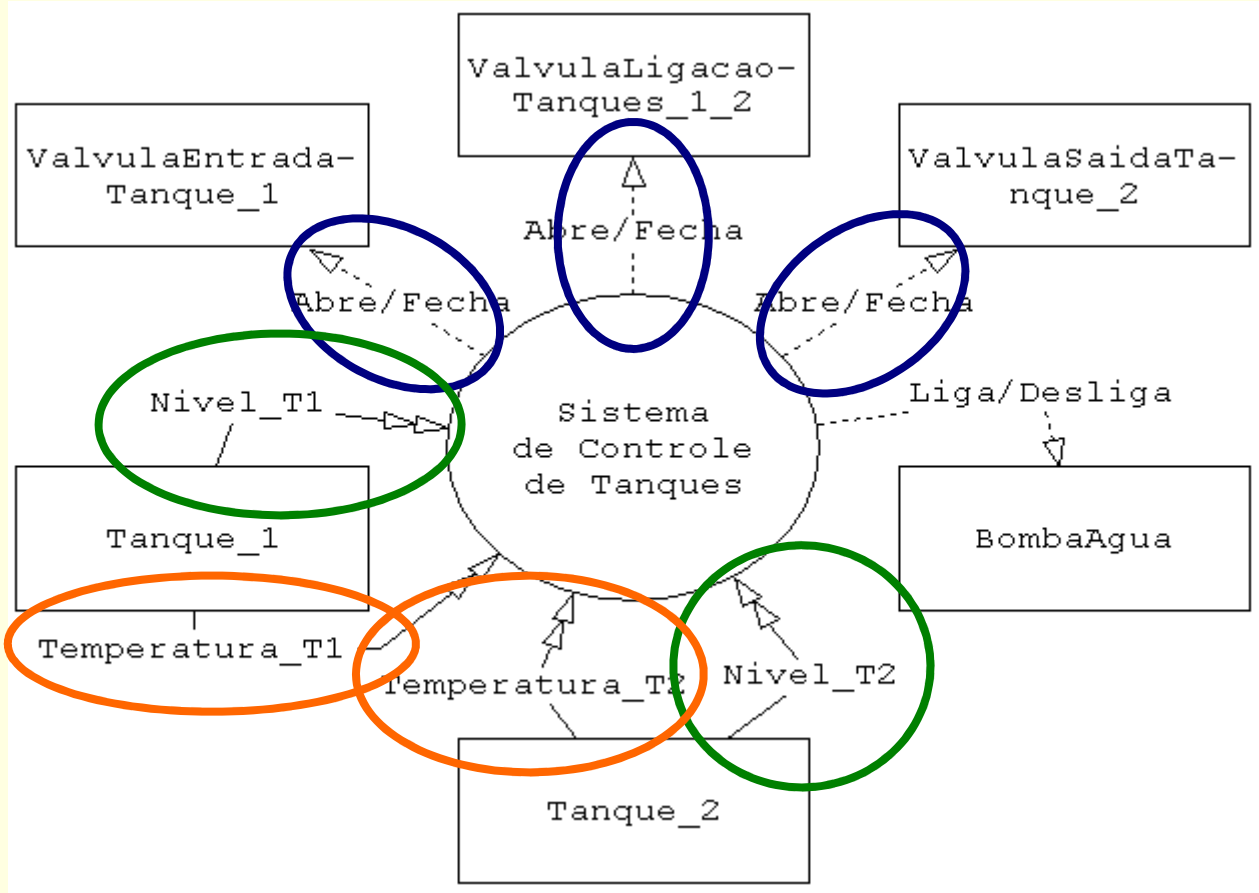
SA/RT – Problemas

Diversos terminadores do mesmo tipo



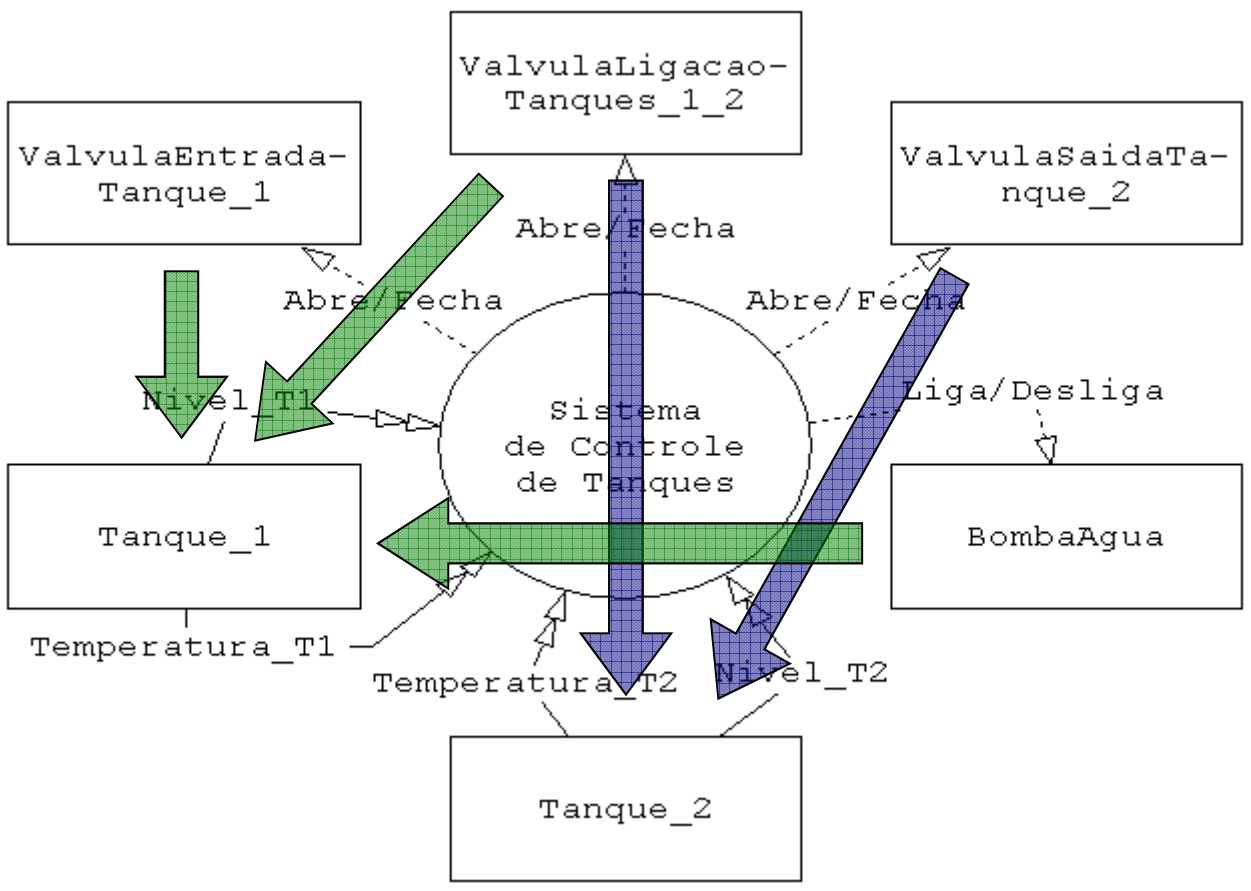
SA/RT – Problemas

Vários terminadores possuem
fluxos de dados iguais

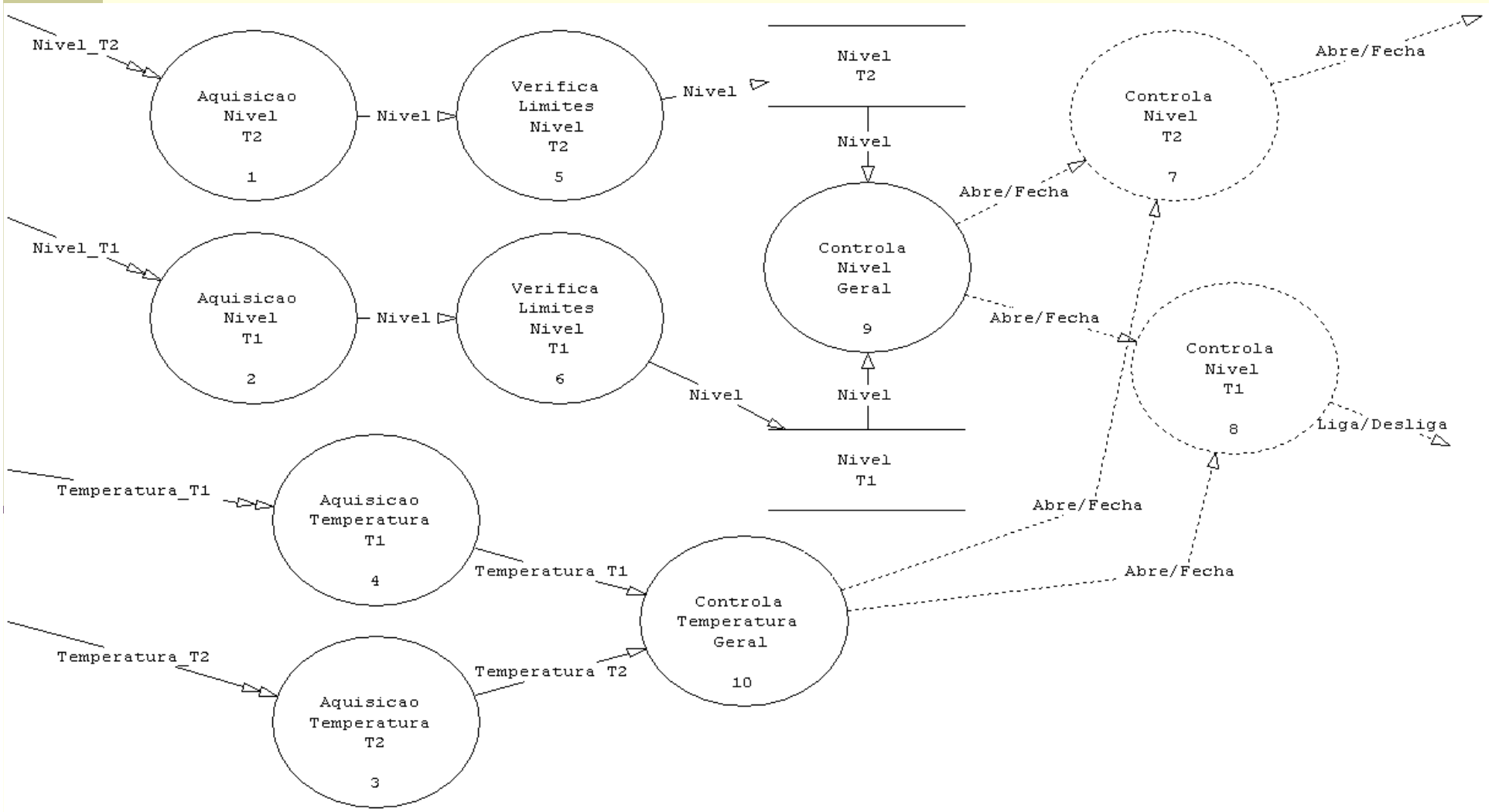


SA/RT – Problemas

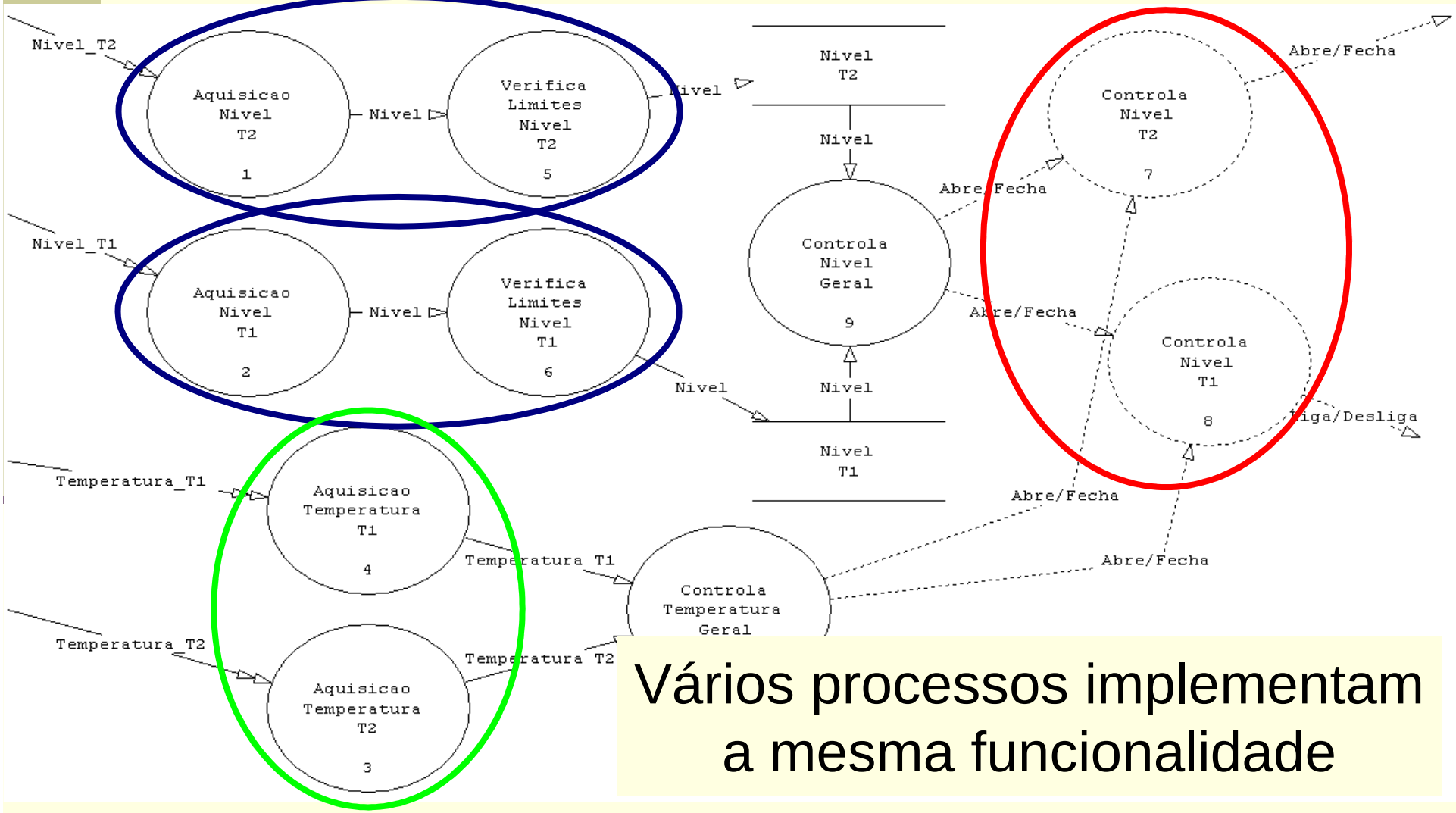
Relações entre elementos do sistema não são clara



SA/RT – Refinamento



SA/RT – Problemas



Vários processos implementam a mesma funcionalidade

SA/RT – Outras deficiências

- Dicionário de dados global
- A conexão entre fluxo e terminadores é semanticamente fraca
- Baixo reuso de funções
- Criação de hierarquias de elementos estruturais sistemas não é possível
- Definição dos requisitos temporais não é clara
- Modelos pouco intuitivos para pessoas não-técnicas

Orientação a Objetos – Introdução

- Iniciou no final dos anos 80
- Utiliza métodos para diminuir a diferença semântica entre a realidade e o modelo
- Entidades compostas
  - Estrutura
  - Comportamento

Orientação a Objetos – Mecanismos

- Mecanismos básicos
  - Objetos
  - Propriedades e Atributos
  - Mensagens e Métodos
  - Classes
  - Herança

Orientação a Objetos – Mecanismos

- Mecanismos básicos
  - Objetos
  - Propried Atributos
  - M
  - C
  - H

• Objetos no mundo real possuem características comuns e podem agrupados de acordo com suas características

• Uma classe representa um gabarito e descreve como os objetos estão estruturados internamente

• Objetos de mesma classe possuem a mesma definição

• Instância é um objeto criado a partir de uma classe

Orientação a Objetos – Mecanismos

- Mecanismos básicos
  - Objetos
  - Propriedades e Atributos
  - Mensagens e
  - Classe
  - Heranç

• Objetos no mundo real possuem propriedades e valores para estas propriedades
• Os valores das propriedades definem o estado do objeto

Orientação a Objetos – Mecanismos

- Mecanismos básicos
  - Objetos
  - Propriedades e Atributos
  - Mensagens e Métodos
  - Classes

• Um objeto exibe algum comportamento quando recebe um estímulo de outro objeto
• O estímulo é chamado de “envio de mensagem”
• Mensagem contém:
  • Nome do objeto receptor
  • Nome da mensagem
  • Argumentos (opcional)
• O comportamento executado recebe o nome de método

Orientação a Objetos – Mecanismos

- Mecanismos básicos
  - Objetos
  - Propriedades e Atributos
  - Mensagens e Métodos
  - Classes
  - Herança

• Estrutura que serve como base para a criação de objetos
• Descreve a estrutura e o comportamento que os objetos deste tipo devem possuir

Orientação a Objetos – Mecanismos

- Mecanismos básicos
  - Objetos
  - Propriedades e Atributos
  - Mensagens e Métodos
  - Classes
  - Herança

Mecanismo para compartilhar automaticamente métodos e atributos entre classes

Orientação a Objetos – Conceitos

- Conceitos chave
  - Abstração
  - Encapsulamen
  - Polimorfism
  - Persistência

• Trabalhar com classes sem a necessidade de conhecer sua implementação
• Diminui a complexidade no projeto de sistemas complexos

Orientação a Objetos – Conceitos

- Conceitos chave
  - Abstração
  - Encapsulamento
  - Polimorfismo
  - Persistên

• Empacotar dados de um objeto, permitindo acesso aos dados somente através dos métodos
• Objetos são “caixa-preta”
• Comunicação entre objetos ocorre exclusivamente através de mensagens pré-definidas

Orientação a Objetos – Conceitos

- Conceitos chave
  - Abstração
  - Encapsulamento
  - Polimorfismo
  - Persistência

• Capacidade de classes compatíveis ter comportamentos diferentes em resposta a uma mesma mensagem

Orientação a Objetos – Conceitos

- Conceitos chave
  - Abstração
  - Encapsulamento
  - Polimorfismo
  - Persistência

• Habilidade de um objeto existir além da execução que o criou

Orientação a Objetos – Exemplo

- Sensor de Distância
  - Atributos
    - Distância de um outro objeto físico
  - Comportamento
    - Sensorar ambiente
  - Subclasses
    - Laser, ultra-som, binocular, etc.

Orientação a Objetos – Comparativo

- Comparativo entre as abordagens

Análise Estruturada	Orientação a Objetos
Dados, Variáveis	Atributos
Chamadas à funções e procedimentos	Mensagens
Funções e procedimentos	Métodos
Estruturas	Classes
Ocorrência da estrutura	Instância, Objeto
	Herança

UML – Introdução

- Unified Modeling Language
  - Esforço em unificar a modelagem OO
    - Rumbaugh (OMT) + Jacobson (OOSE) + Booch (BOOCH)
  - Mantido pelo Object Management Group (OMG)
    - Versão atual: 2.0
- Ferramentas que suportam UML
  - Rational Rose
  - Artisan RT-Studio
  - Poseidon
  - Várias outras

UML – Objetivos

- Proporciona uma linguagem de modelagem visual expressiva
- Possui mecanismos de extensão e especialização para especializar os conceitos principais
- Ser independente de linguagens e processos particulares
- Proporcional uma formalização para compreensão da linguagem de modelagem
- Suporta conceitos de desenvolvimento em alto-nível como colaborações, frameworks, padrões e componentes
- Integra as melhores práticas de modelagem

UML – Tipos de Diagramas

- Diagramas Estruturais
  - Diagramas de Classe
  - Diagramas de Objetos
  - Diagramas de Componentes
  - Diagramas de Distribuição
- Diagramas Comportamentais
  - Diagramas de Casos de Uso
  - Diagramas de Seqüências
  - Diagramas de Colaboração
  - Diagramas de Estados
  - Diagramas de Atividade

UML – Tipos de Diagramas

- Diagramas Estruturais
  - Diagramas de Classe
  - Diagramas de Objetos
  - Diagramas de Componentes
  - Diagramas de Distribuição
- Diagramas Comportamentais
  - Diagramas de Casos de Uso
  - Diagramas de Seqüências
  - Diagramas de Colaboração(visto no caso de uso)
  - Diagramas de Estados
  - Diagramas de Atividade

Diagrama de Casos de Uso

- Descreve o comportamento funcional do sistema na visão do usuário
- Mostra as relações entre os atores e os casos de uso dentro do sistema
- Usado na fase de levantamentos dos requisitos do sistema
- É a descrição completa das funcionalidades do sistema e do seu ambiente

Diagrama de Casos de Uso – Exemplo

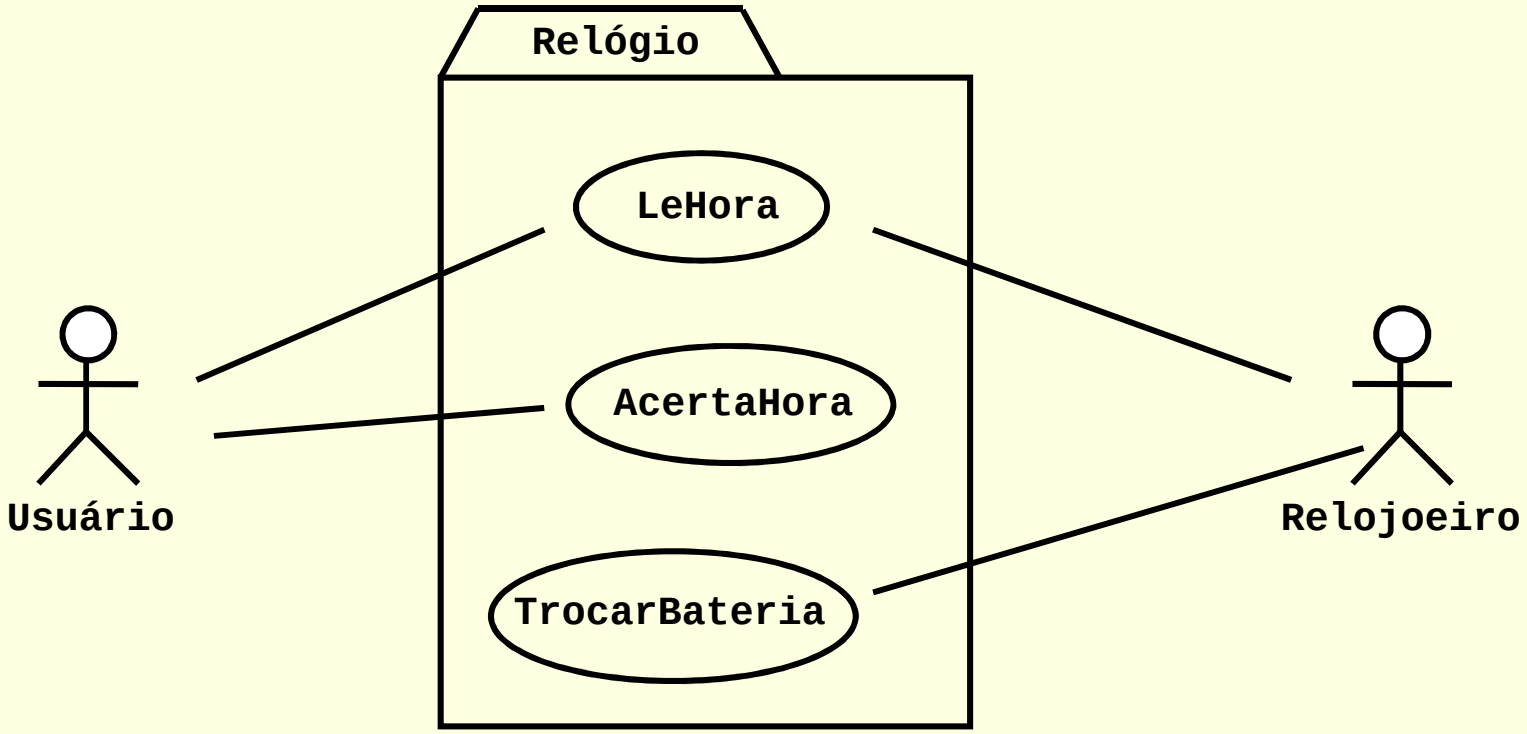


Diagrama de Casos de Uso – Exemplo

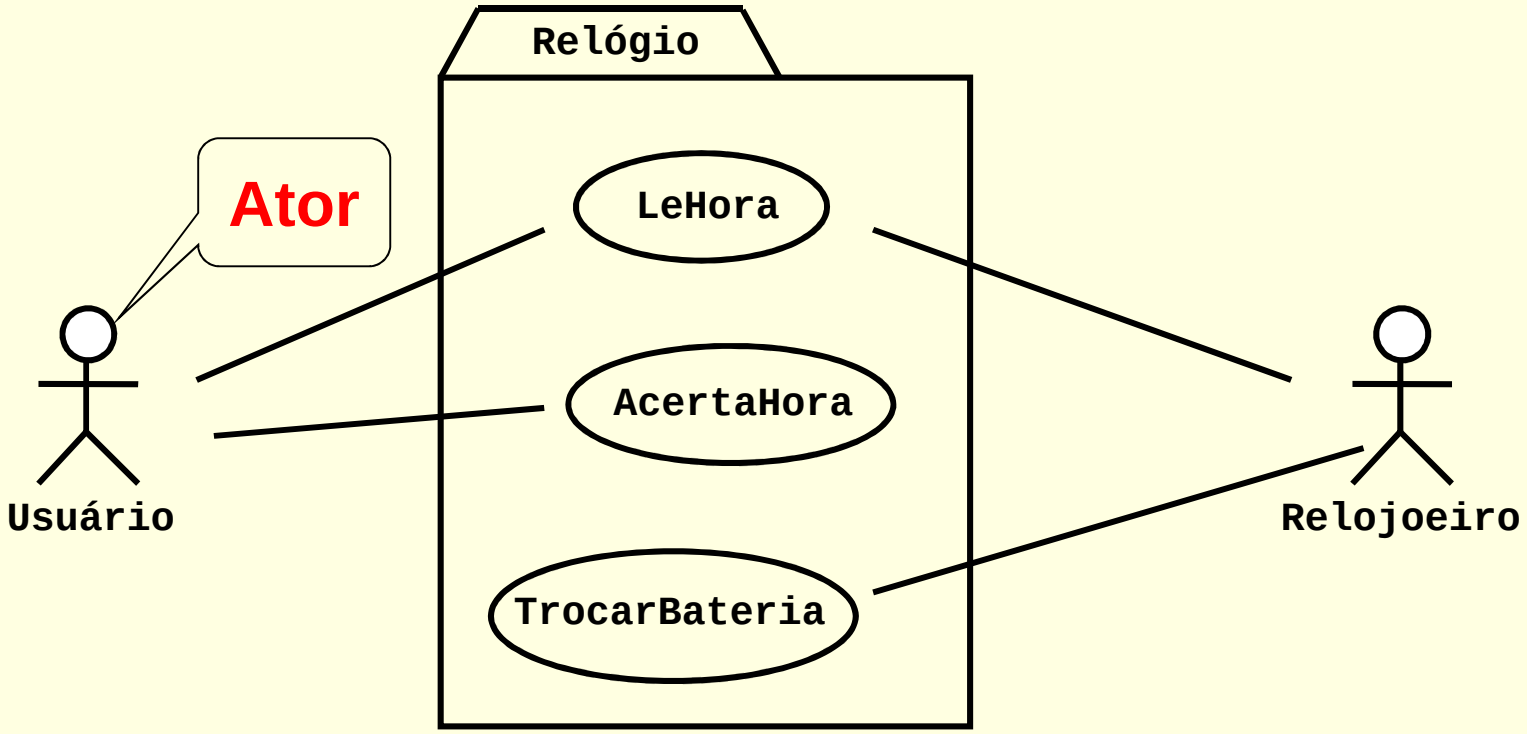


Diagrama de Casos de Uso – Exemplo

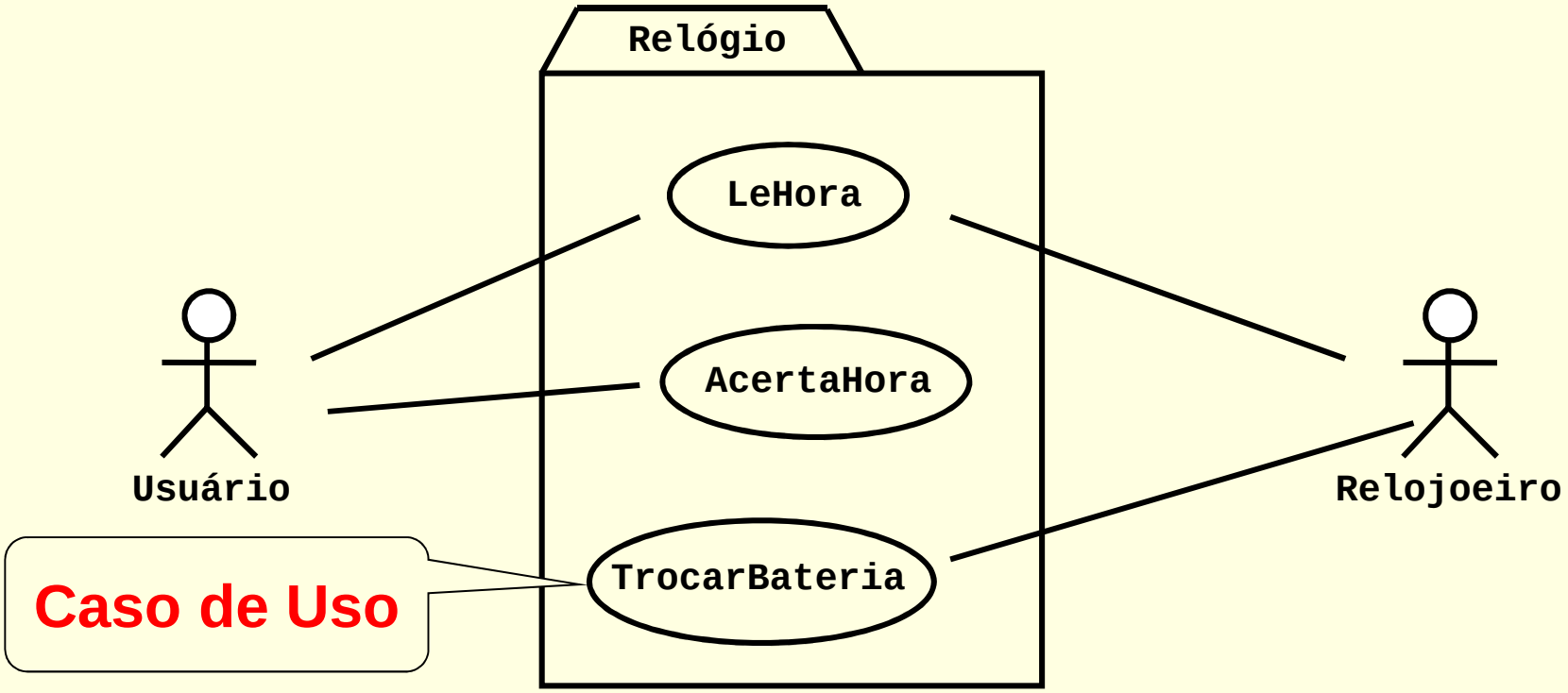


Diagrama de Casos de Uso – Exemplo

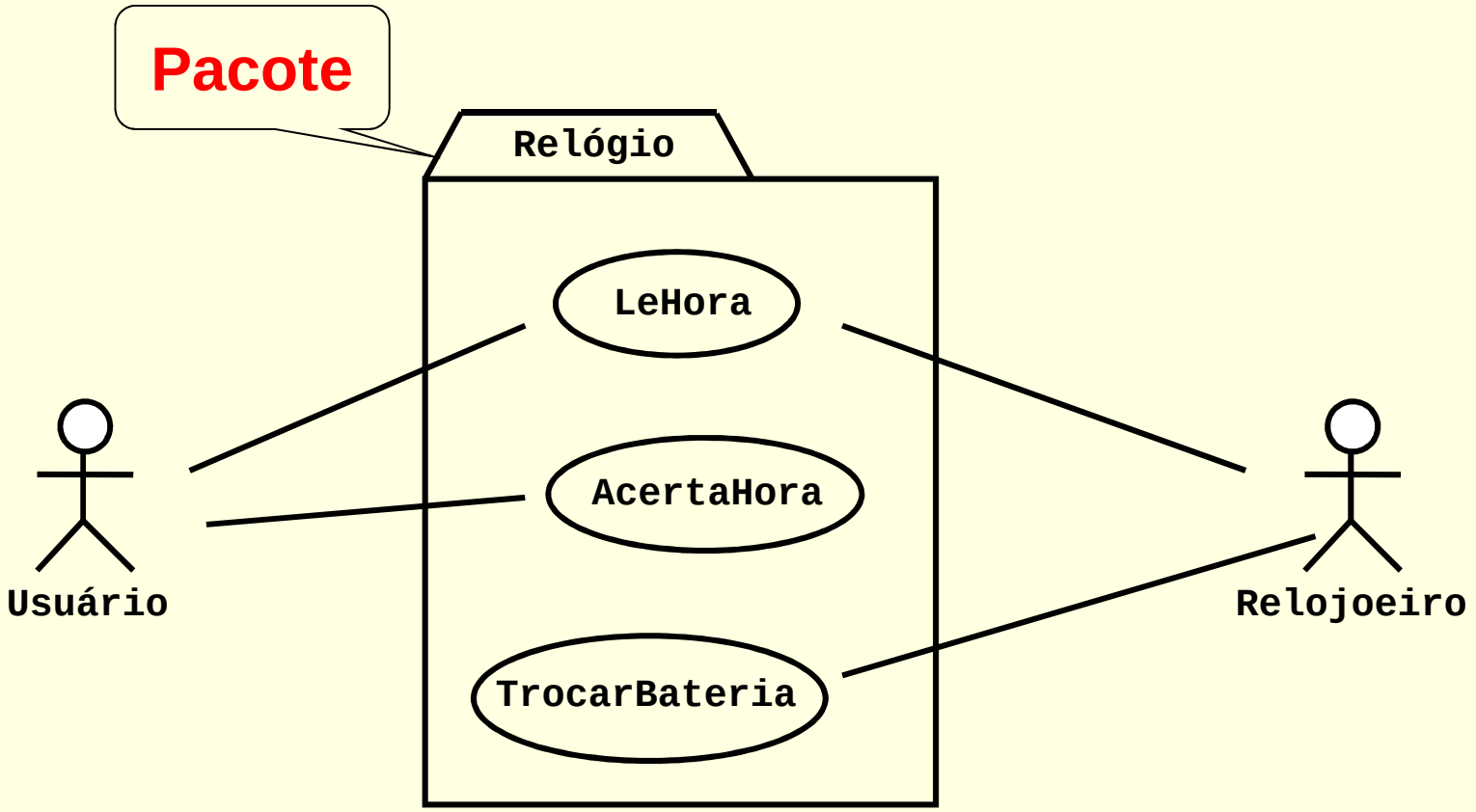


Diagrama de Casos de Uso

- Atores
  - Modela uma entidade externa que se comunica com o sistema
  - Pode ser: Usuário, Sistema Externo, Elemento físico do ambiente
  - Um ator possui um nome único e uma descrição opcional

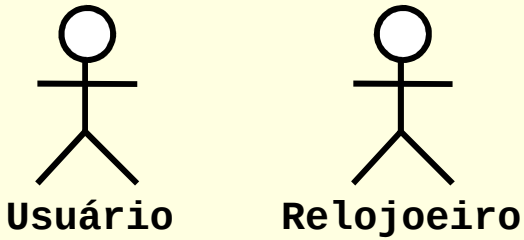


Diagrama de Casos de Uso

- Casos de Uso
  - Representa uma classe de funcionalidades fornecidas pelo sistema como uma seqüência de eventos
  - É composto por
    - Nome único
    - Atores participantes
    - Condições de entrada
    - Fluxo de eventos
    - Condições de saída
    - Requisitos especiais

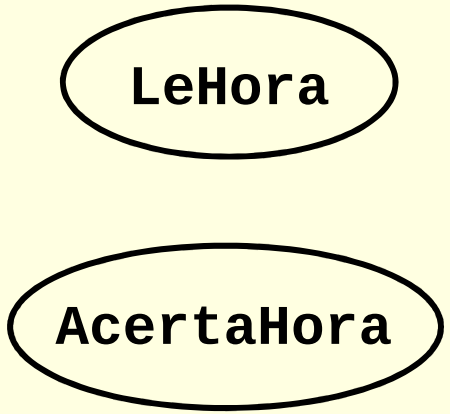
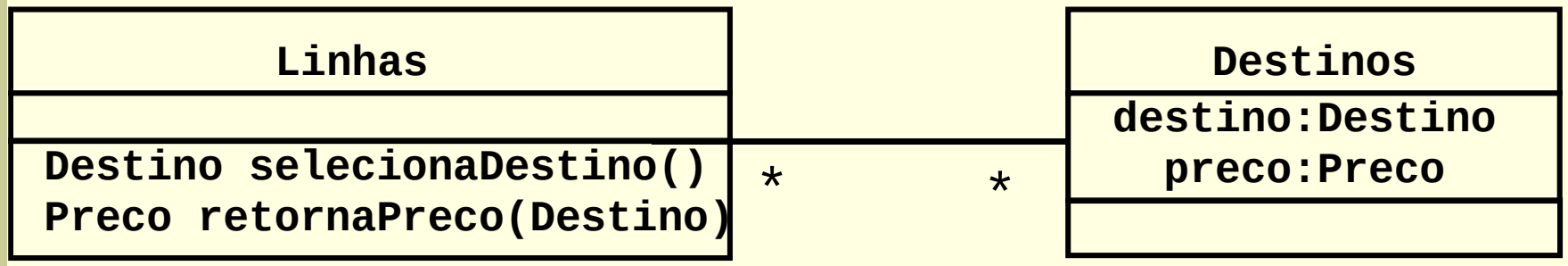
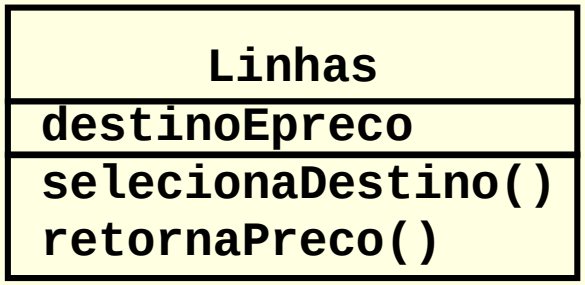


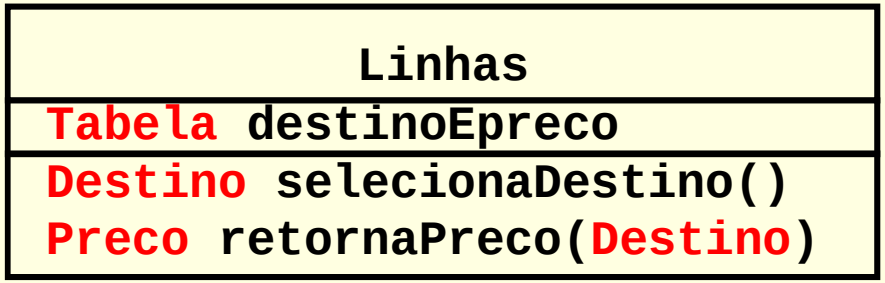
Diagrama de Classe



- Diagrama de classes representa a estrutura do sistema
- Diagramas de classes são usados
  - Durante a análise de requisitos para modelar os conceitos do domínio do problema
  - Durante o projeto do sistema para modelar os subsistemas e suas interfaces
  - Durante o projeto dos objetos para modelar as classes

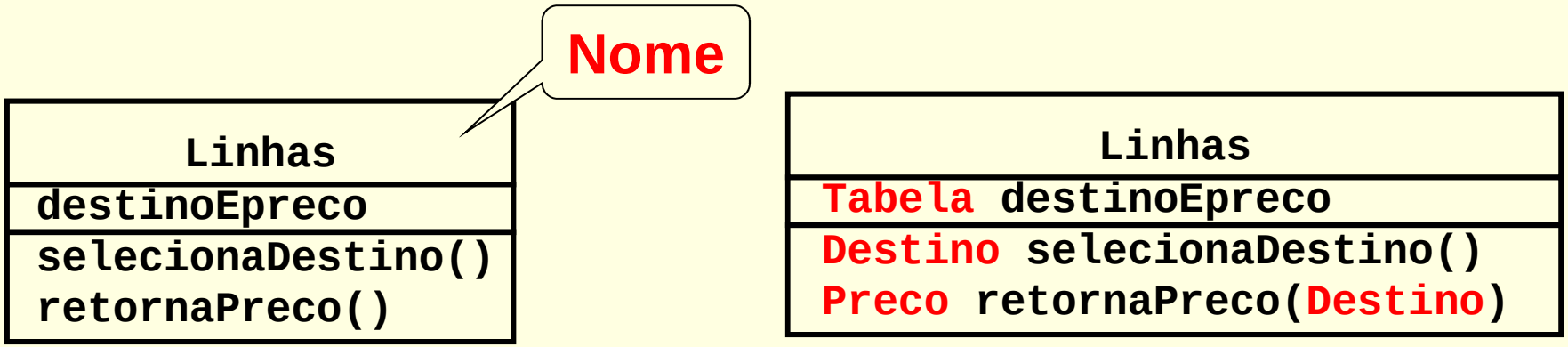
Classes





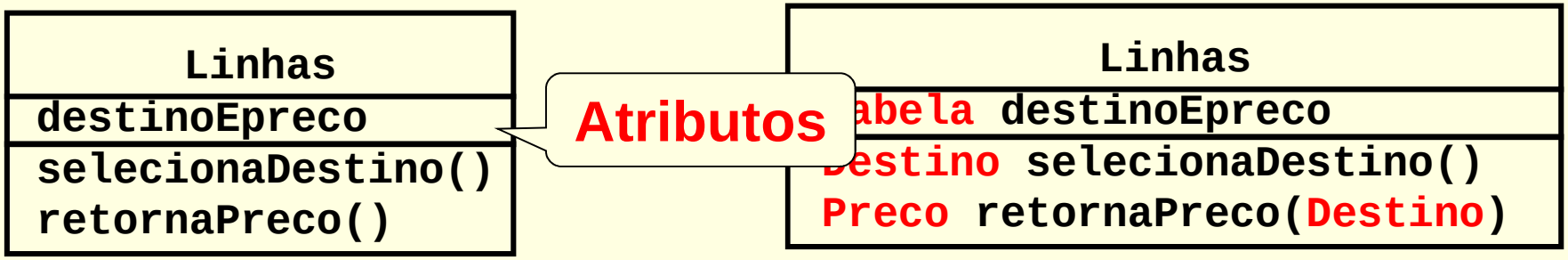
- Uma ***classe*** representa um conceito
- Uma classe encapsula estado ***(atributos)*** e comportamento ***(operações)***
- Cada atributo tem um ***tipo***
- Cada operação tem uma ***assinatura***
- O nome da classe é a única informação obrigatória

Classes



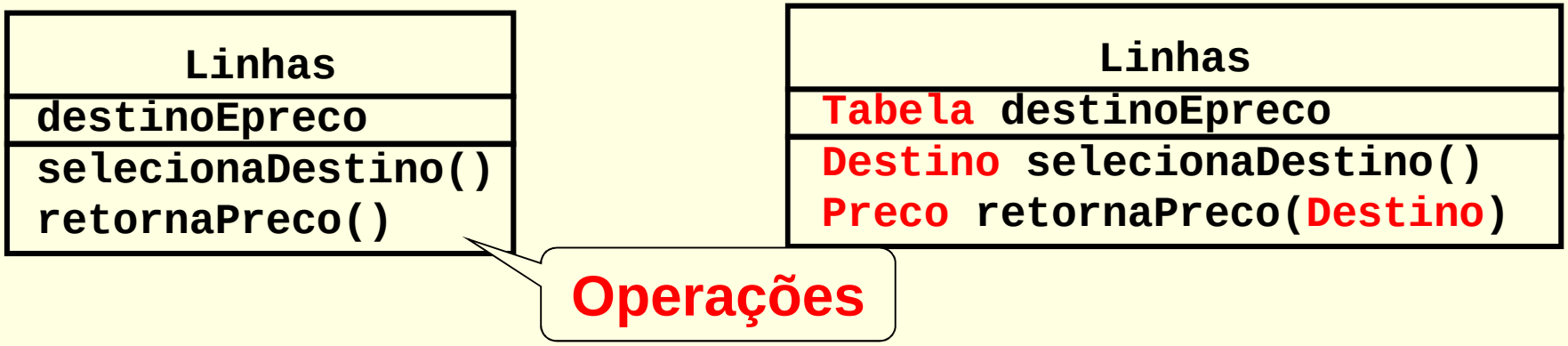
- Uma ***classe*** representa um conceito
- Uma classe encapsula estado ***(atributos)*** e comportamento ***(operações)***
- Cada atributo tem um ***tipo***
- Cada operação tem uma ***assinatura***
- O nome da classe é a única informação obrigatória

Classes



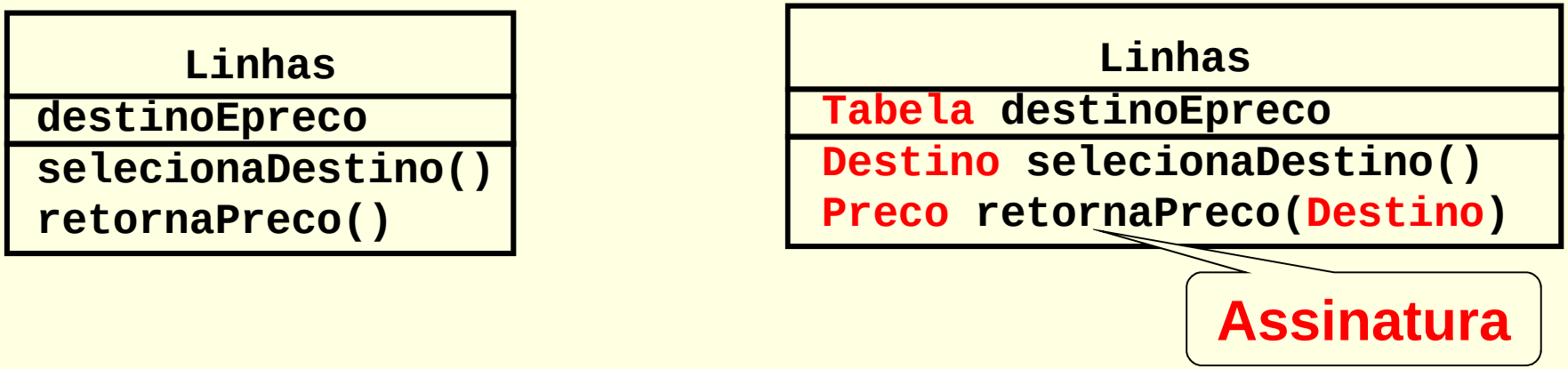
- Uma ***classe*** representa um conceito
- Uma classe encapsula estado ***(atributos)*** e comportamento ***(operações)***
- Cada atributo tem um ***tipo***
- Cada operação tem uma ***assinatura***
- O nome da classe é a única informação obrigatória

Classes



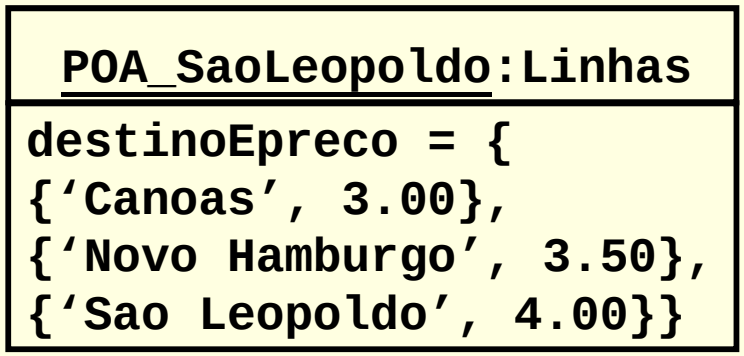
- Uma ***classe*** representa um conceito
- Uma classe encapsula estado ***(atributos)*** e comportamento ***(operações)***
- Cada atributo tem um ***tipo***
- Cada operação tem uma ***assinatura***
- O nome da classe é a única informação obrigatória

Classes



- Uma ***classe*** representa um conceito
- Uma classe encapsula estado ***(atributos)*** e comportamento ***(operações)***
- Cada atributo tem um ***tipo***
- Cada operação tem uma ***assinatura***
- O nome da classe é a única informação obrigatória

Instâncias



- Uma ***instância*** representa um ocorrência de uma classe
- O nome de uma instância é sublinhado e pode conter o nome da sua classe
- Os atributos são representados com seus ***valores***

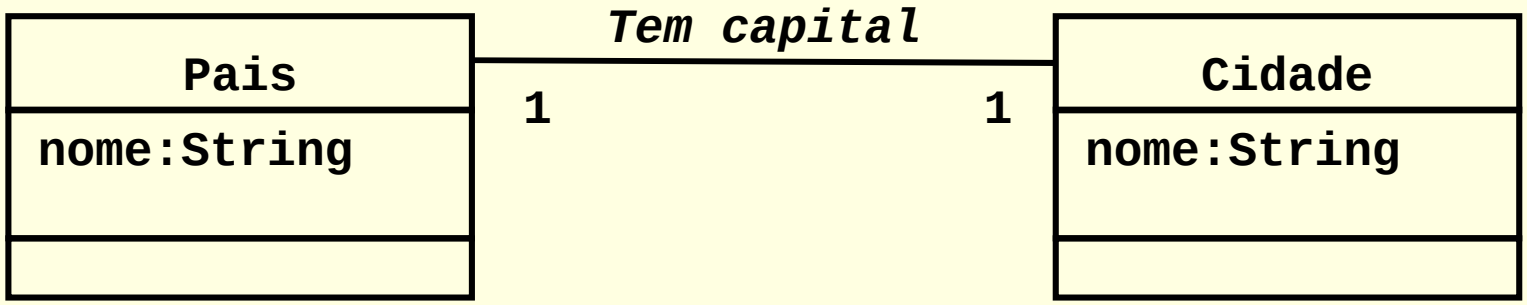
Atores vs. Instâncias

- **Qual a diferença entre um ator, uma classe e uma instância ?**
- Ator:
  - Uma entidade externa ao sistema que esta sendo modelado, que interage com o mesmo
- Classe:
  - Uma abstração para uma entidade no domínio do problema, que possui características comuns com outro elementos do sistema
- Objeto:
  - Uma instância específica de uma classe

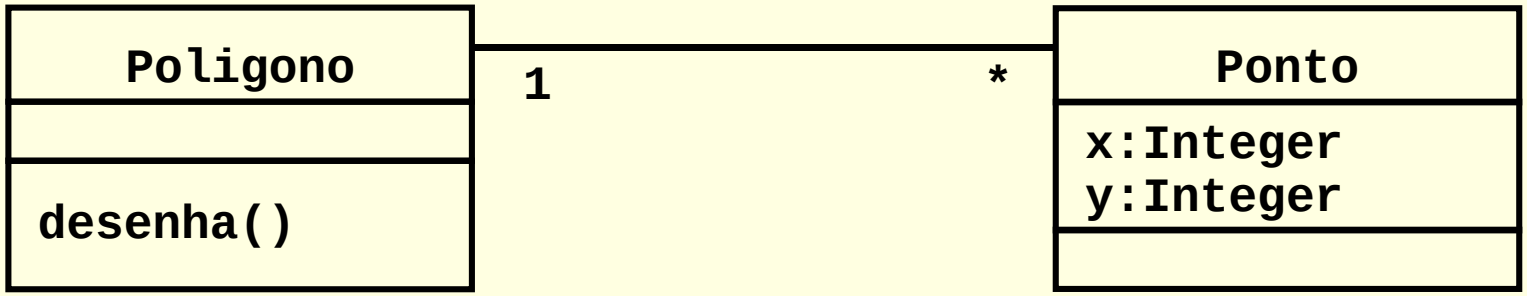
Associações

- Associações denotam relações entre classes
- A multiplicidade de uma associação mostra quantos objetos “fonte” podem referenciar objetos “destino”
  - Exemplos:
    - um-para-um
    - um-para-muitos
    - muitos-para-muitos

Associações



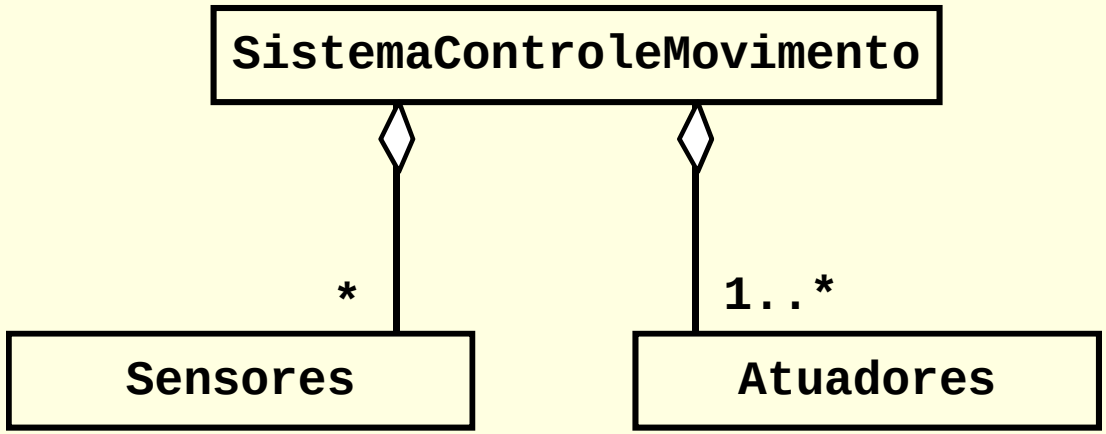
Associação 1-para-1



Associação 1-para-muitos

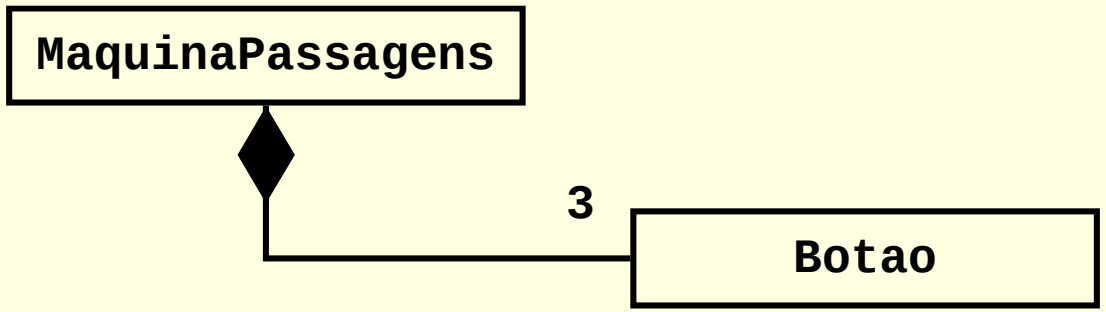
Agregação

- Uma ***agregação*** é um caso especial de associação que estipula uma hierarquia "composta por"
- O ***agregado*** é a classe pai, os ***componentes*** são classes filhas.

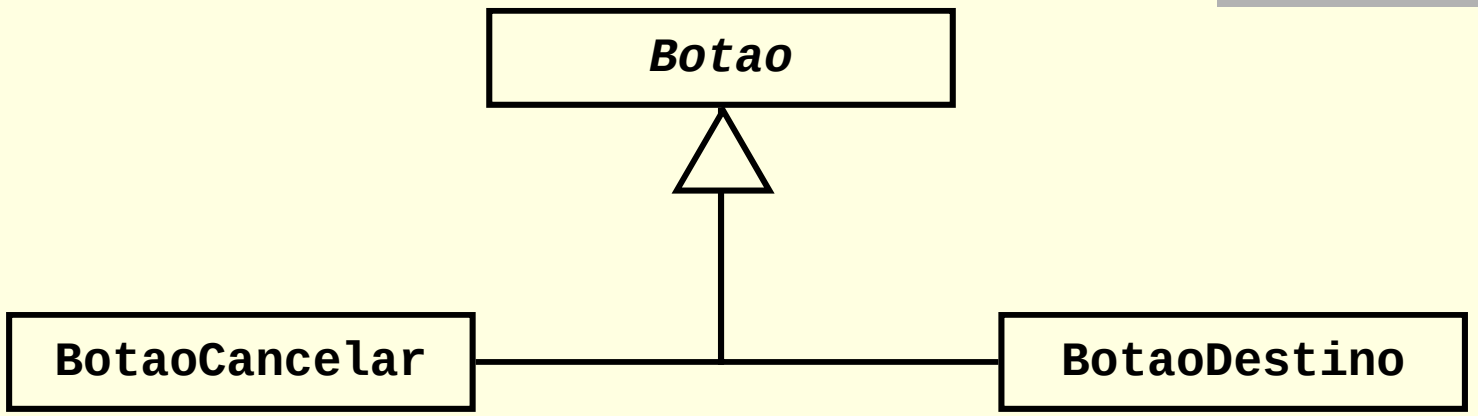


Composição

- Um losango preenchido indica ***composição***, uma forma “forte” de agregação onde os componentes não podem existir sem o agregado

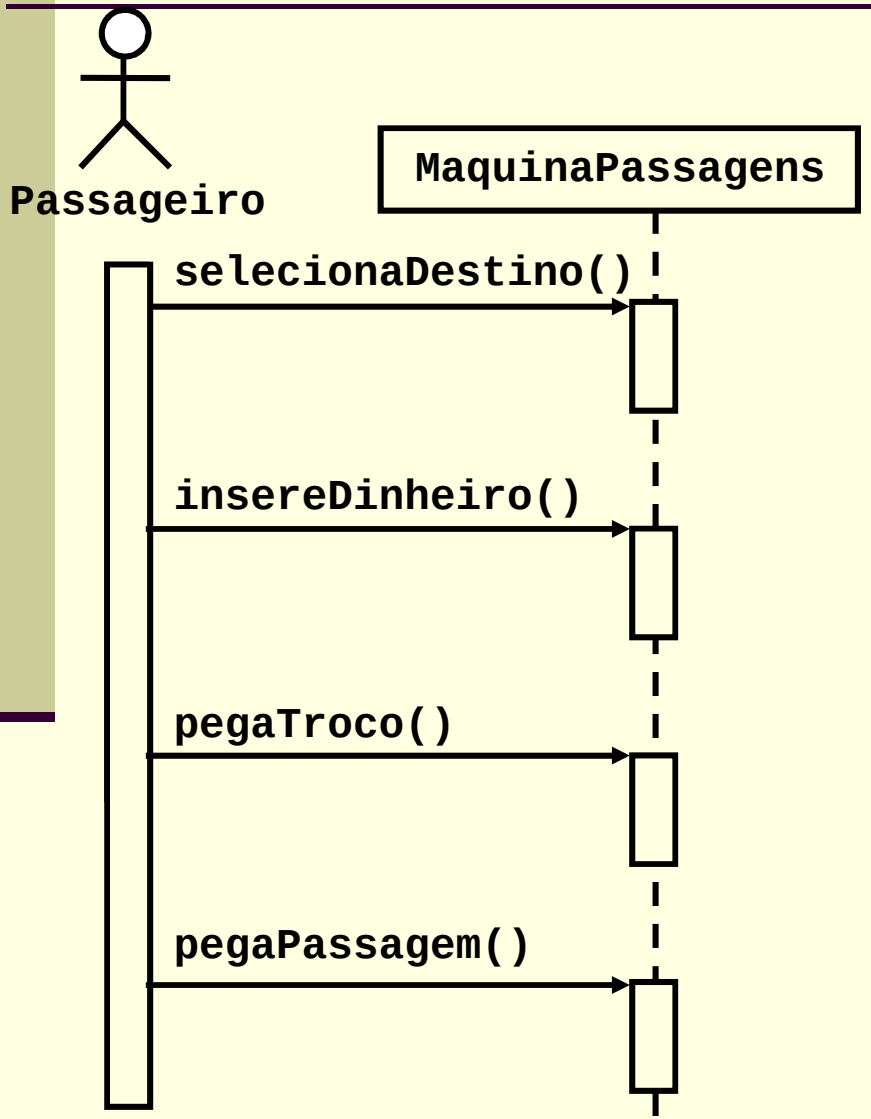


Generalização/Herança



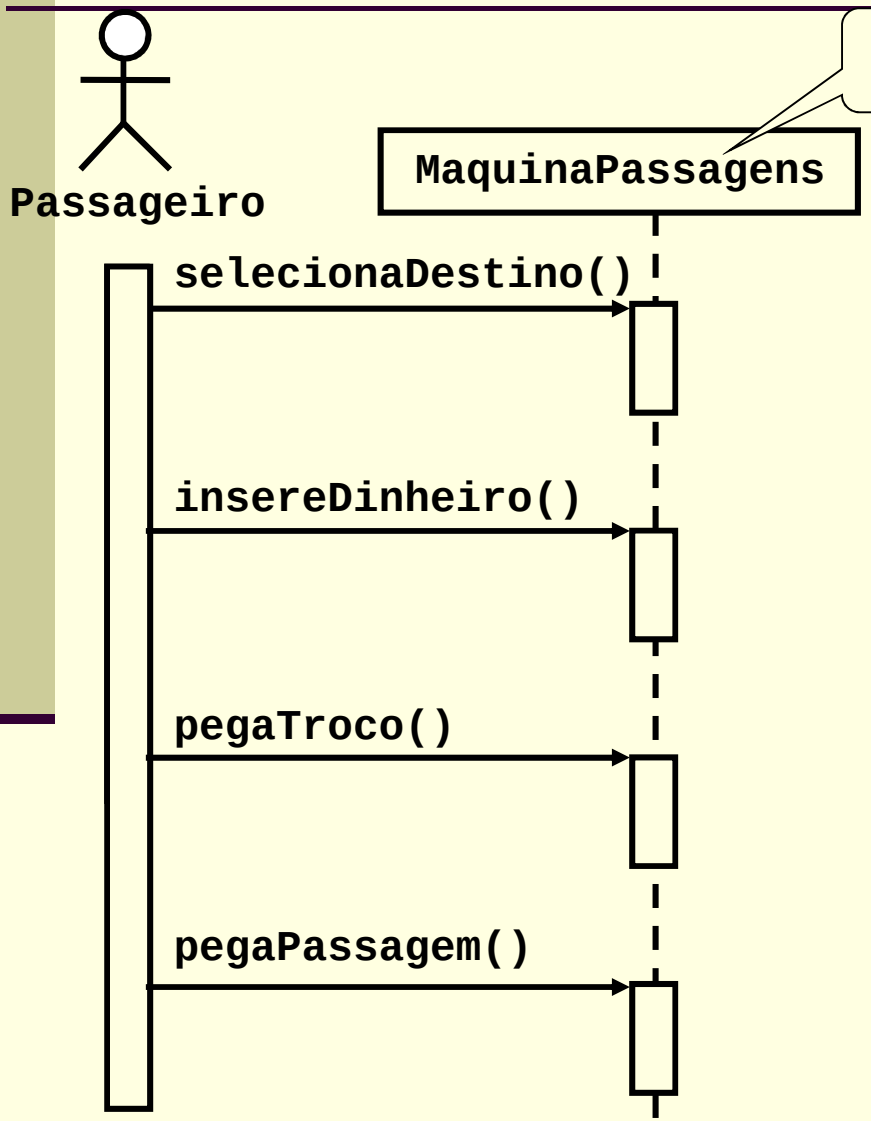
- A generalização mostra a relação de herança entre classes
- As classes filhas herdam os atributos e operações da classe pai
- A generalização simplifica o modelo eliminando as redundâncias

Diagrama de Seqüências



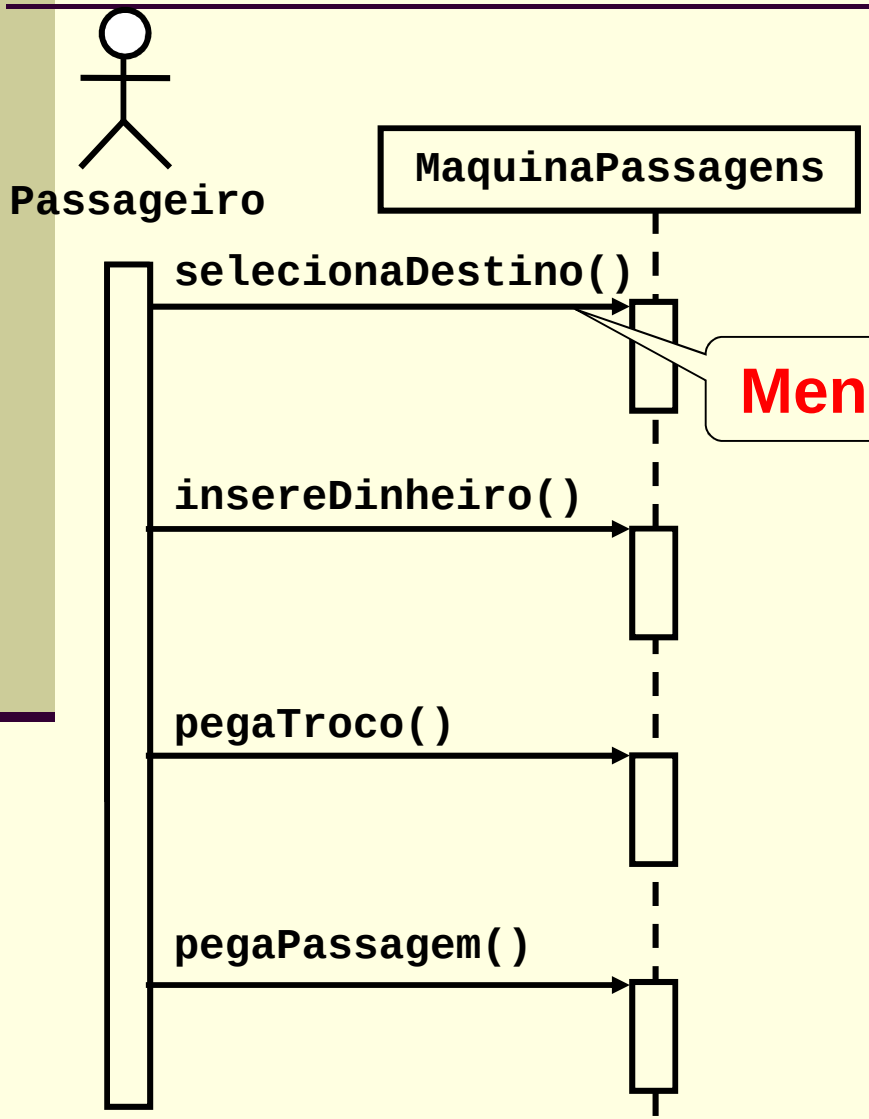
- Usado na análise de requisitos
  - Para refinar a descrição de casos de uso
  - Para encontrar objetos adicionais (“objetos participantes”)
- Usado no projeto do sistema
  - Para refinar as interfaces dos subsistemas

Diagrama de Seqüências



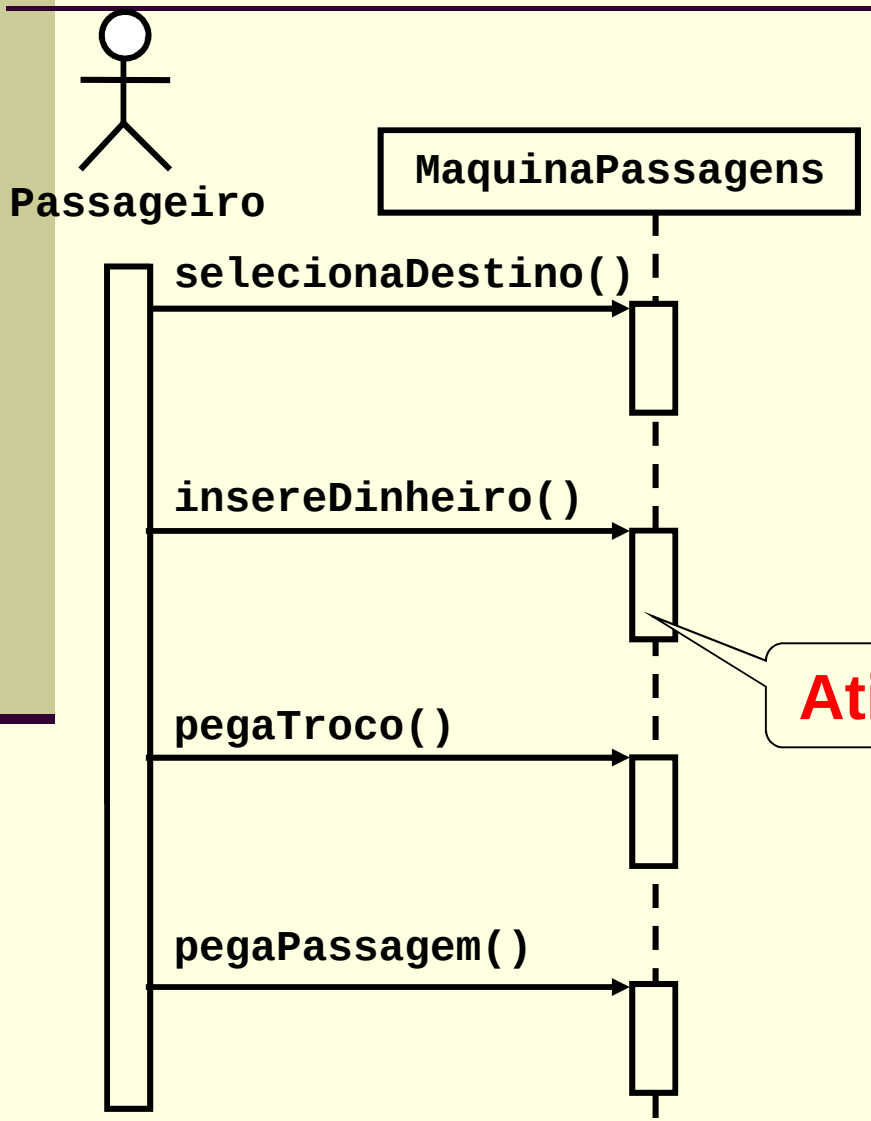
- Usado na análise de requisitos
  - Para refinar a descrição de casos de uso
  - Para encontrar objetos adicionais (“objetos participantes”)
- Usado no projeto do sistema
  - Para refinar as interfaces dos subsistemas

Diagrama de Seqüências



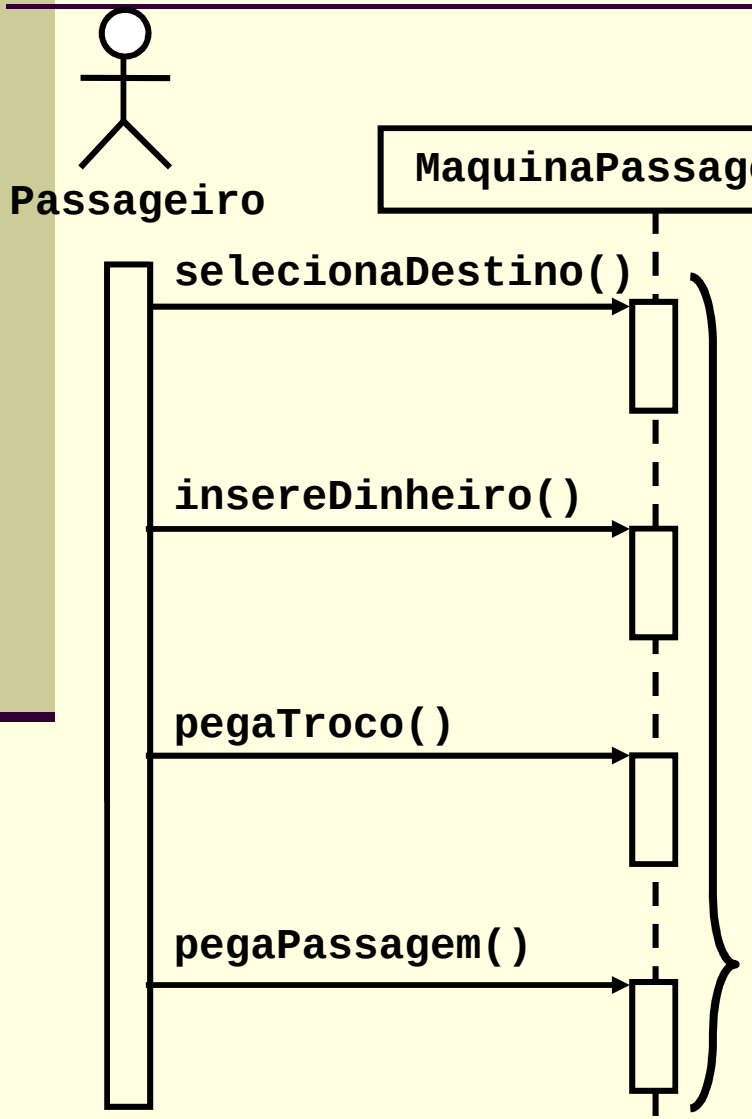
- Usado na análise de requisitos
  - Para refinar a descrição de casos de uso
  - Para encontrar objetos adicionais (“objetos participantes”)
- Usado no projeto do sistema
  - Para refinar as interfaces dos subsistemas

Diagrama de Seqüências



- Usado na análise de requisitos
  - Para refinar a descrição de casos de uso
  - Para encontrar objetos adicionais (“objetos participantes”)
  - o no projeto do sistema
  - Para refinar as interfaces dos subsistemas

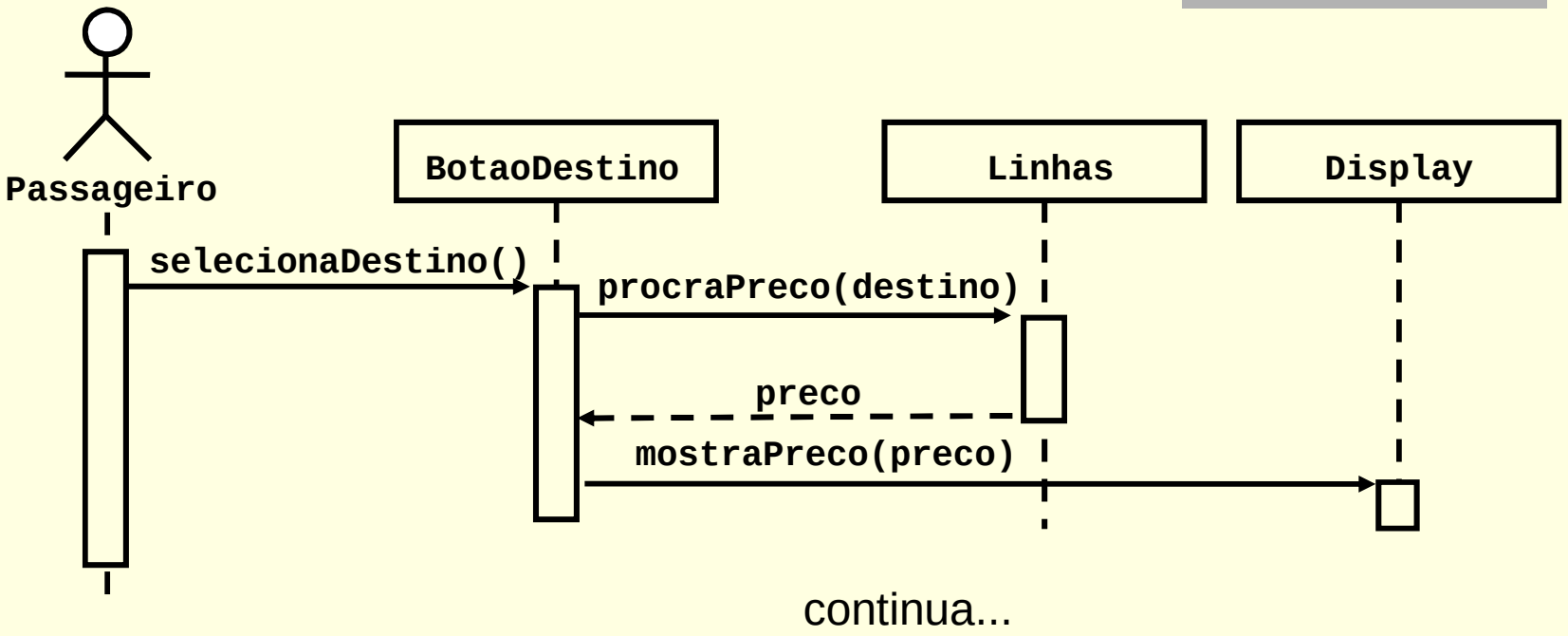
Diagrama de Seqüências



- Usado na análise de requisitos
  - Para refinar a descrição de casos de uso
  - Para encontrar objetos adicionais (“objetos participantes”)
- Usado no projeto do sistema
  - Para refinar as interfaces dos subsistemas

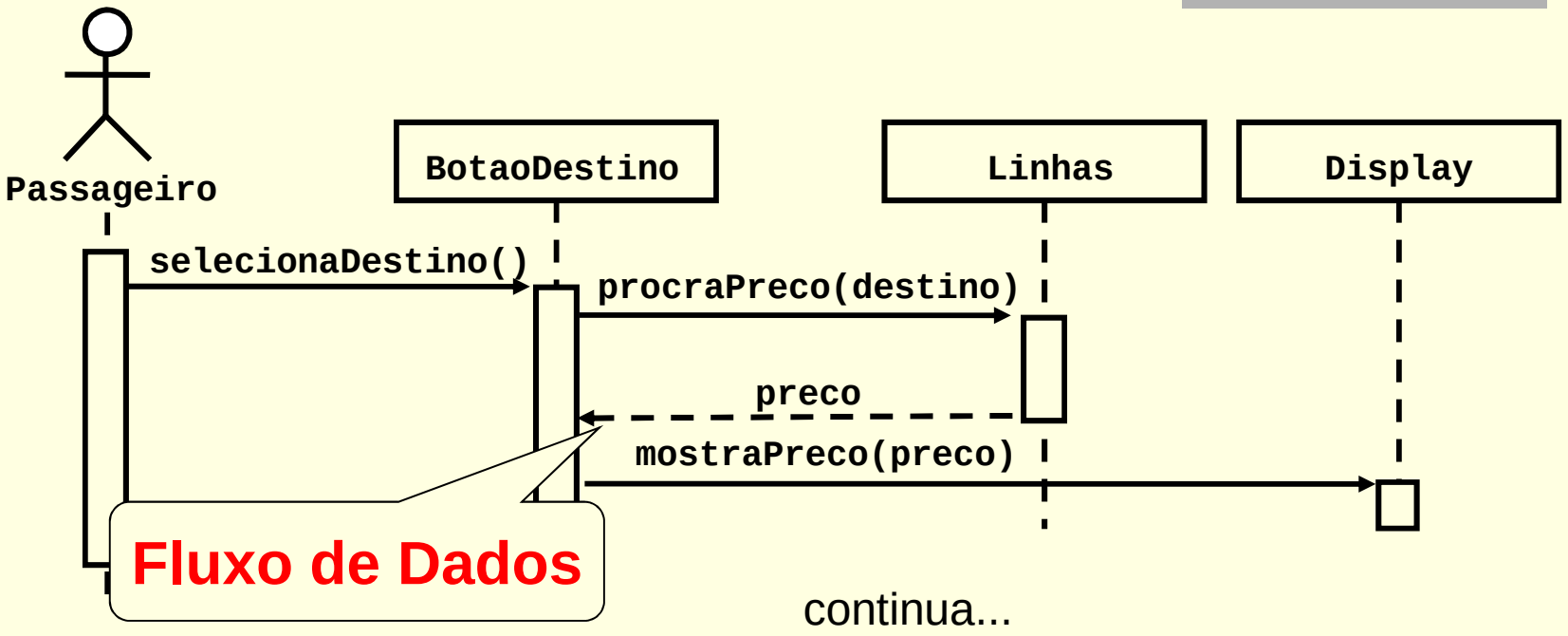
Tempo do Cenário

Diagrama de Seqüências: Mensagens Aninhadas



- A fonte de uma seta indica o estímulo que envia uma mensagem
- Um estímulo tem a duração da soma de todos os tempos de todas as mensagens aninhadas

Diagrama de Seqüências: Mensagens Aninhadas



- A fonte de uma seta indica o estímulo que envia uma mensagem
- Um estímulo tem a duração da soma de todos os tempos de todas as mensagens aninhadas

Observações sobre o Diagrama de Seqüências

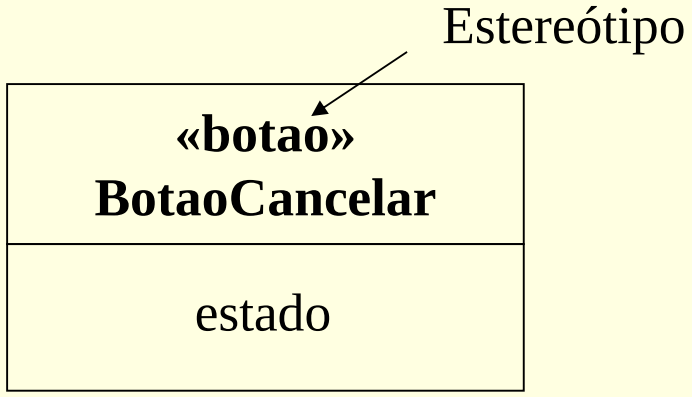
- O diagrama de seqüências da UML representa o comportamento em termos de interações
- Complementa o diagrama de classes que represente apenas a estrutura do sistema
- Útil para encontrar objetos participantes do sistema
- Consome tempo para a construção mas vale o investimento
- Semântica idêntica ao Diagrama de Colaboração

Mecanismos de Extensão

- Porque um mecanismo de extensão ?
  - Embora seja bem definida a UML precisa ser personalizada para domínios de problemas específicos
- Mecanismo de extensão permite
  - Adicionar novos elementos de modelagem
  - Criar novas propriedades
  - Especificar uma nova semântica
- Três tipos de mecanismos
  - Estereótipos (stereotypes)
  - Valores Anotados (tagged values)
  - Restrições (constraints)

Mecanismos de Extensão - Estereótipos

- Estereótipos são usados para criar novos elementos no modelo que podem ser usados em domínios específicos
- Exemplo: Modelando um sistema de controle de elevador
  - «hardware»
  - «software»



Mecanismos de Extensão – Valores Anotados

- Definem propriedades adicionais à qualquer elemento do modelo
- Podem ser definidos para estereótipos
- São apresentados como pares “etiqueta-valor” onde a “etiqueta” representa a propriedade e o “valor” representa o valor da propriedade
- São úteis para adicionar propriedades sobre
  - Geração de código
  - Controle de versão
  - Gerenciamento de configuração
  - Entre outros

Mecanismos de Extensão – Restrições

- As restrições são usadas para adicionar novas regras ou modificar alguma regra existente
- Também é usado para especificar condições que devem ser satisfeitas para os elementos do modelo
- Podem ser representadas através de linguagem natural ou OCL (Object Contraint Language)

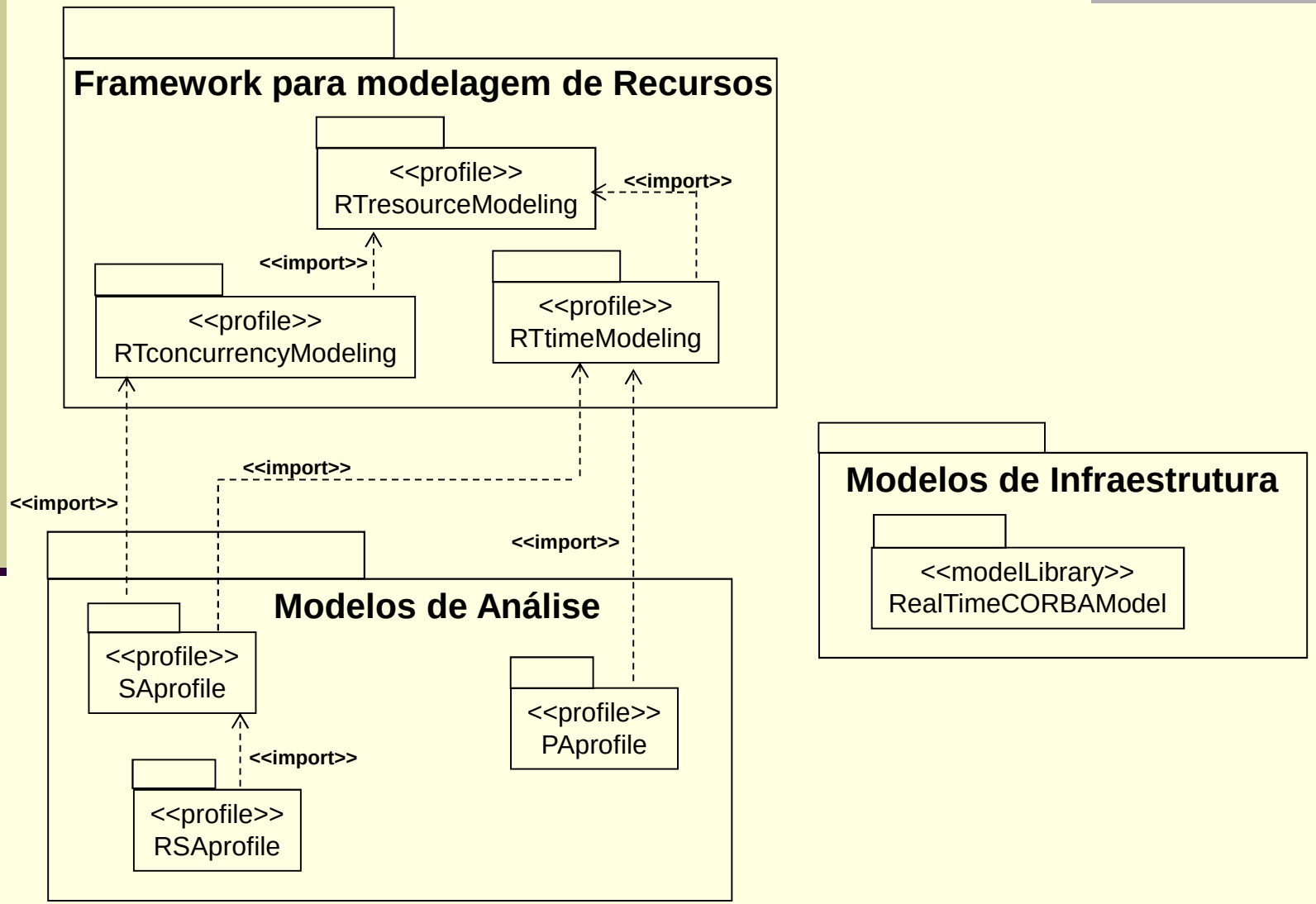
Mecanismos de Extensão – Perfis

- Os perfis UML são conjuntos de estereótipos, valores anotados e restrições para domínios de aplicação específico
  - Perfil UML Tempo-Real
  - Perfil de Tolerância a Falhas
  - Perfil de Plataformas

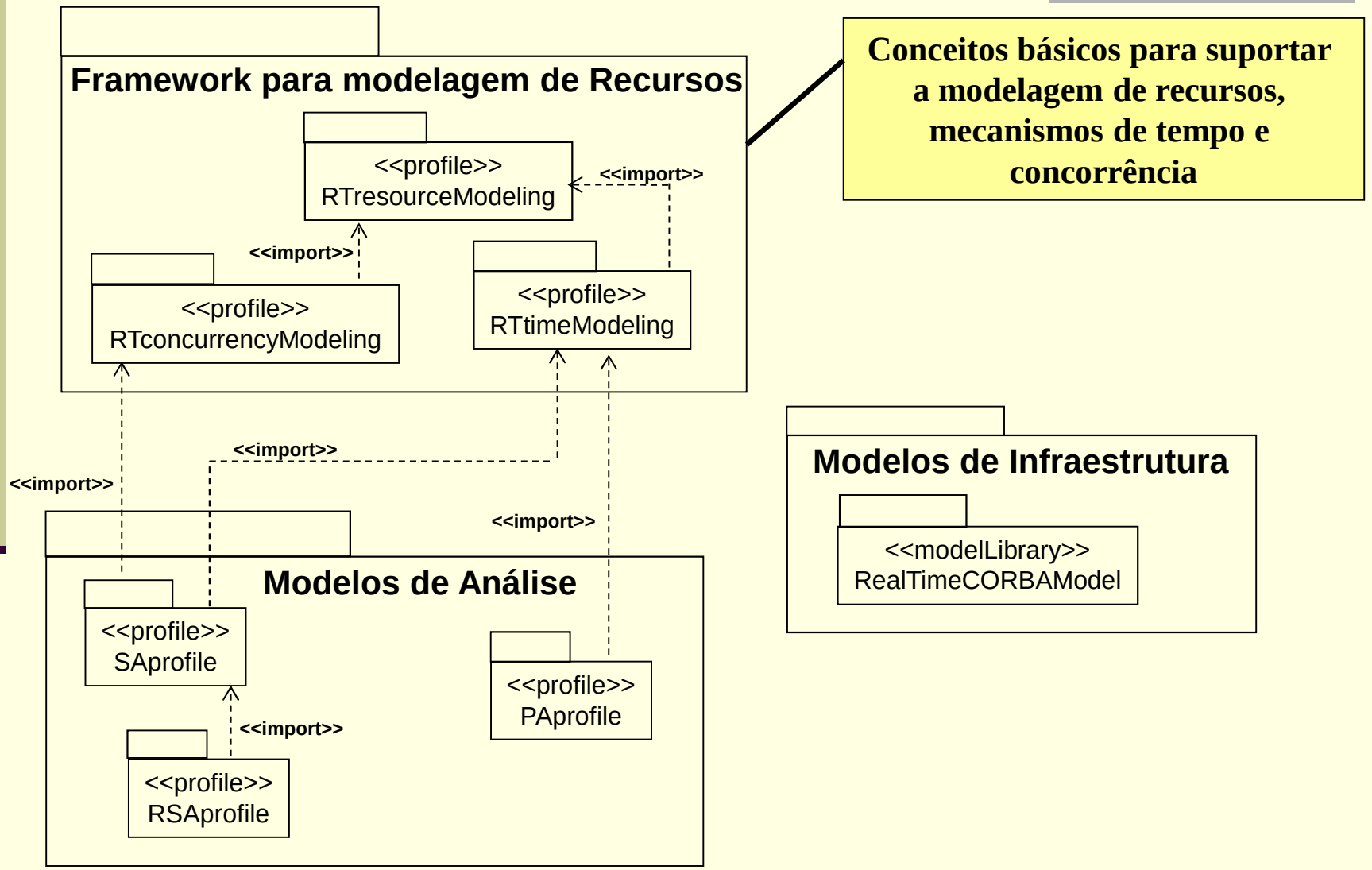
Real-Time UML

- UML Profile for Schedulability, Performance and Time
- Conjunto de abstrações normalmente utilizadas no projeto de um sistema tempo-real
- Visa facilitar
  - Especificação de restrições
  - Troca de informações entre as ferramentas de apoio

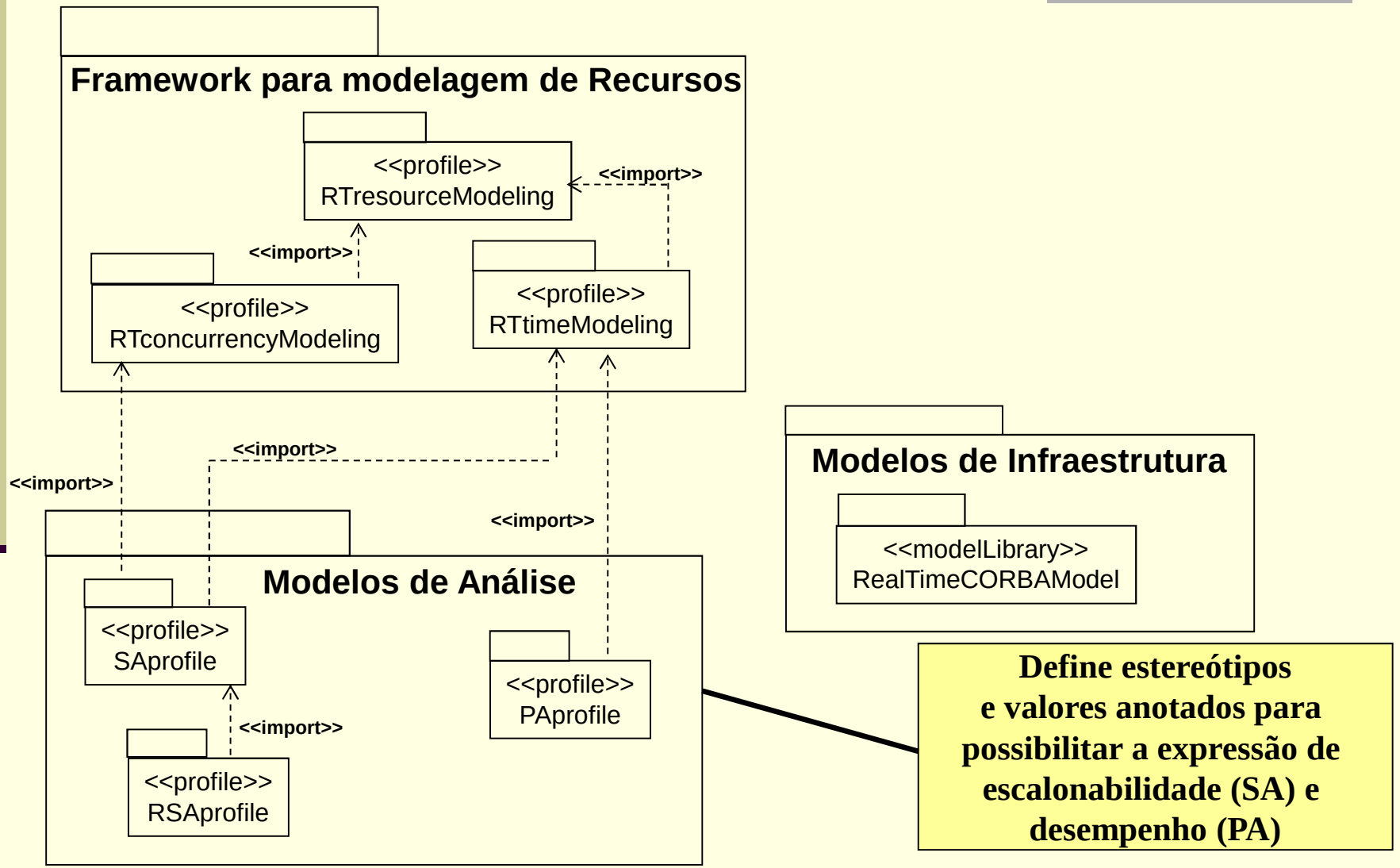
Real-Time UML – Estrutura do Perfil



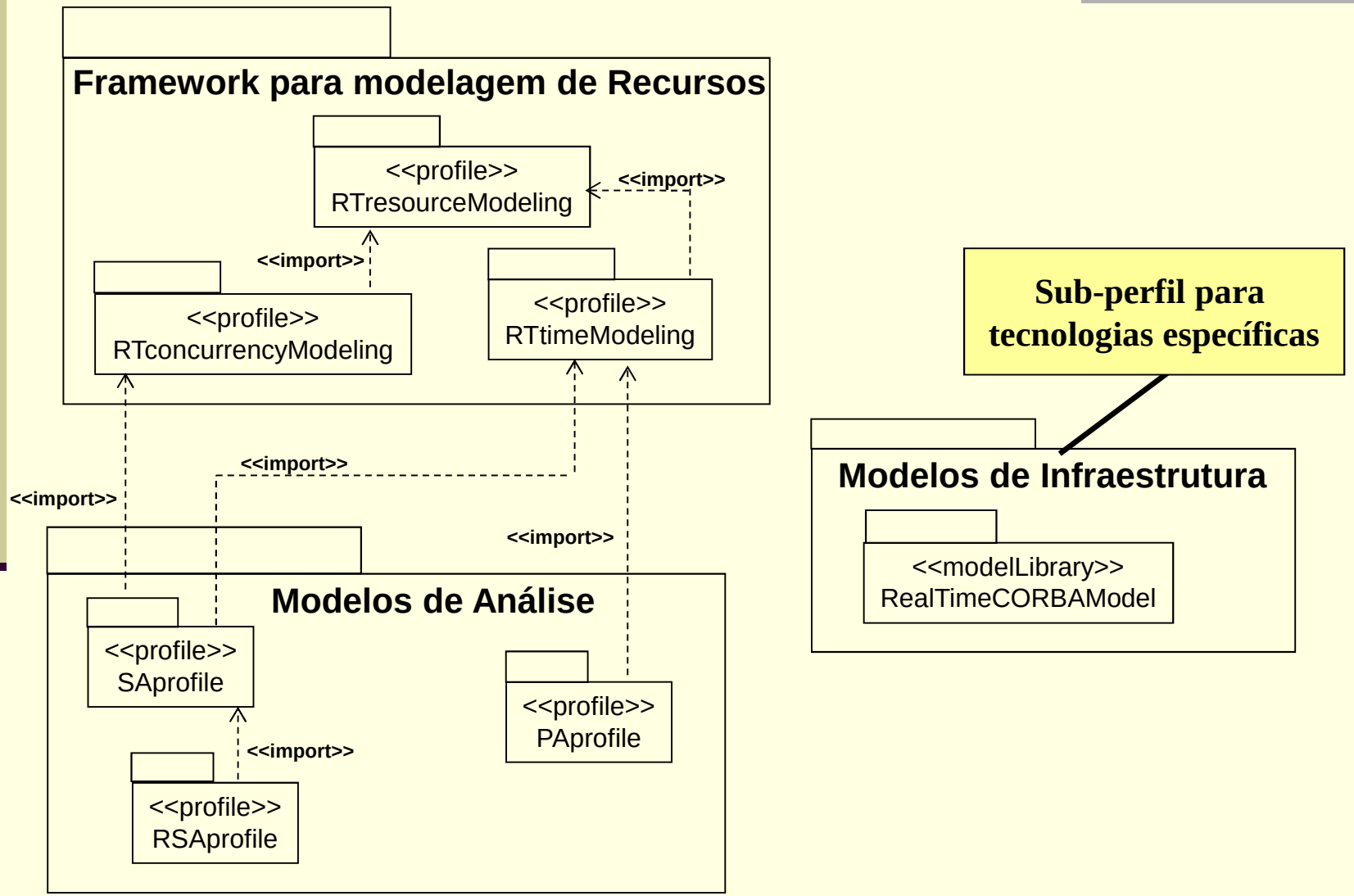
Real-Time UML – Estrutura do Perfil



Real-Time UML – Estrutura do Perfil



Real-Time UML – Estrutura do Perfil



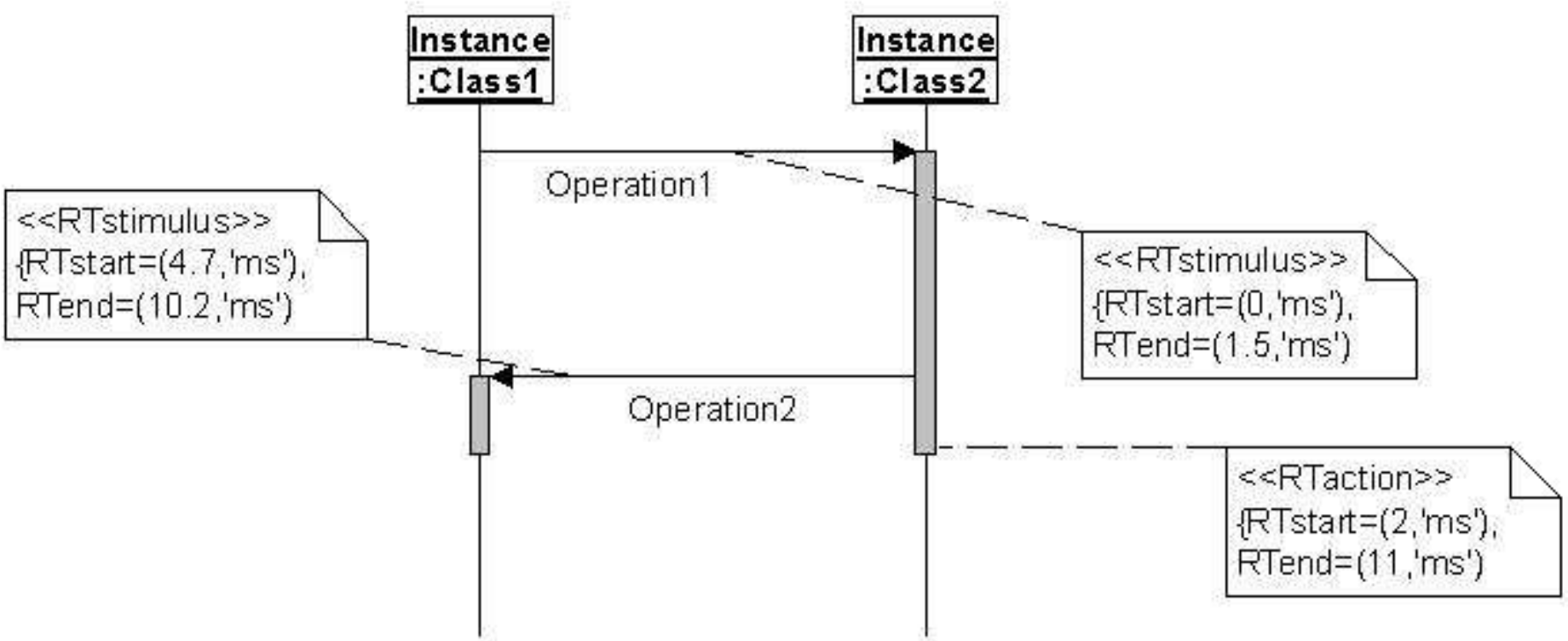
Real-Time UML – Modelagem de Recursos

- Especifica padrões presentes em muitos métodos de análise tempo-real
- Define uma terminologia comum e um framework conceitual voltados para reduzir as ambigüidades do modelo
- Expressa características de QoS (Quality of Service)

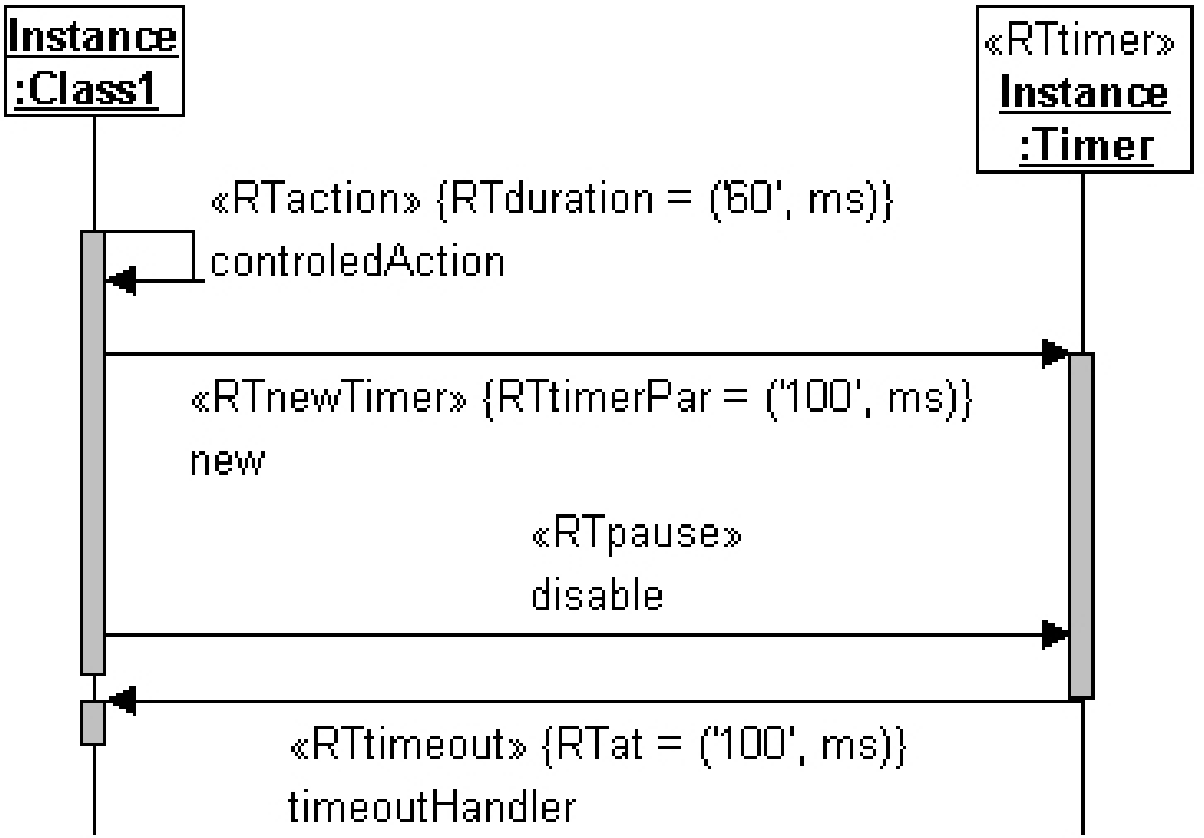
Real-Time UML – Modelagem de Tempo

- Cobre a especificação de medidas de tempo, como intervalos, delays e deadlines
- Define os serviços de tempo oferecidos pelos sistemas operacionais tempo real, como Timers e Clocks
- Também define padrões para funções de distribuição de probabilidades

Real-Time UML – Definição de Intervalos usando tempos relativos



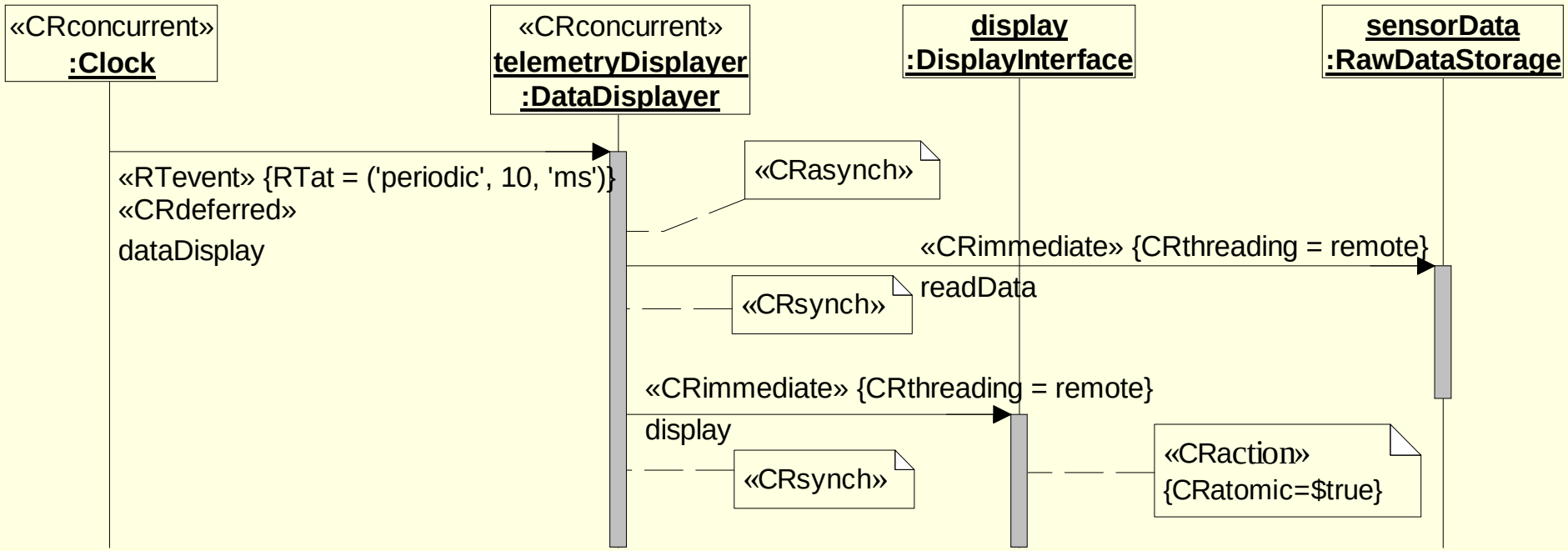
Real-Time UML – Execução de ação com controle de *deadline*



Real-Time UML – Modelagem de Concorrência

- Aborda os seguintes pontos:
  - **Recursos concorrentes**: mecanismos para um comportamento concorrente (tarefas, processos ou threads);
  - **Cenários concorrentes**: seqüências de ações geralmente interligadas efetuadas por recursos concorrentes;
  - **Serviços de recursos concorrentes**: serviços que possuem algum tipo de política de controle de acesso

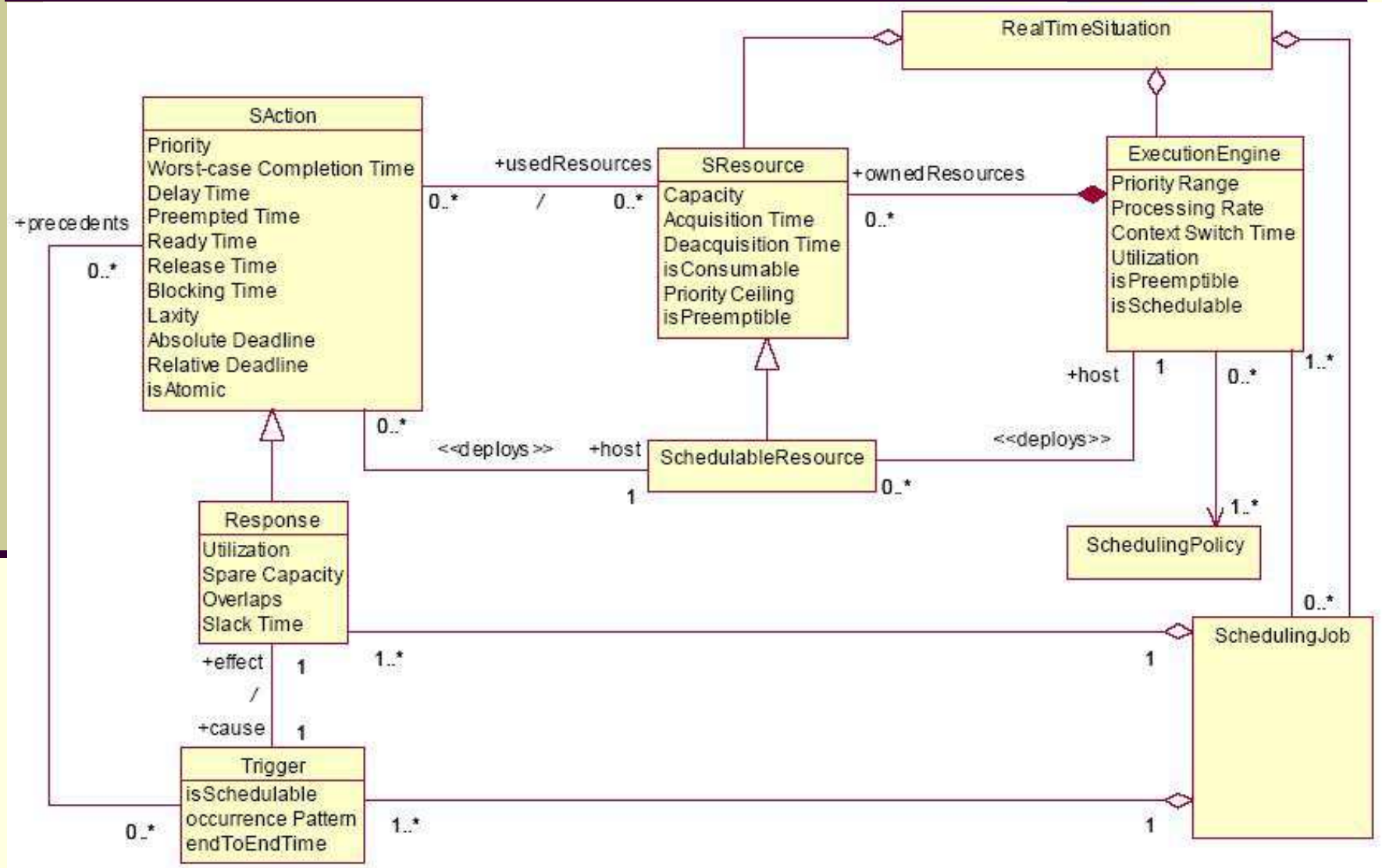
Real-Time UML – Exemplo de Modelagem de Concorrência



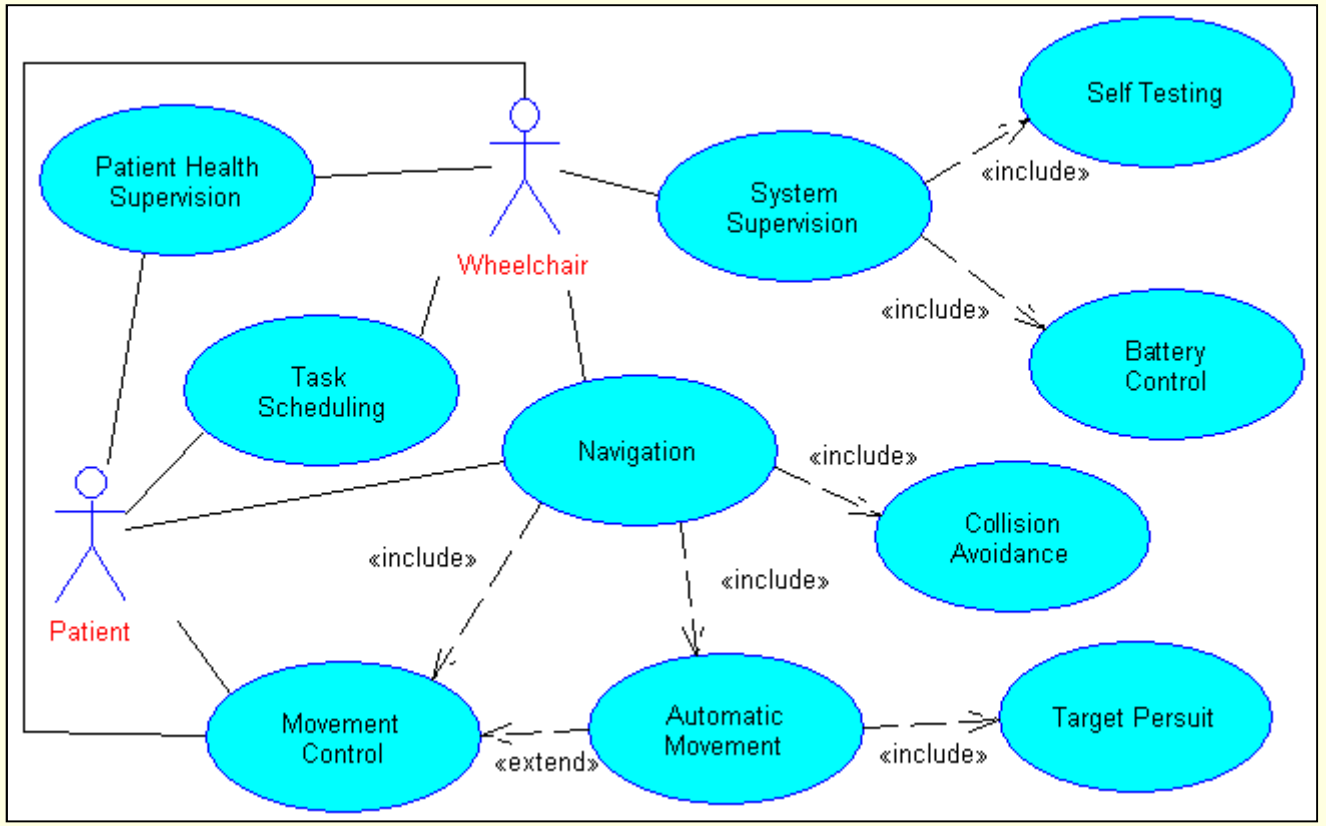
Real-Time UML – Modelo para Análise de Escalonabilidade

- Oferece alternativas para decorar o modelo com anotações, de modo a permitir a aplicação de diversas técnicas de análise de escalonabilidade
- Baseia-se na modelagem de situações, através de diagramas de seqüência e de colaboração

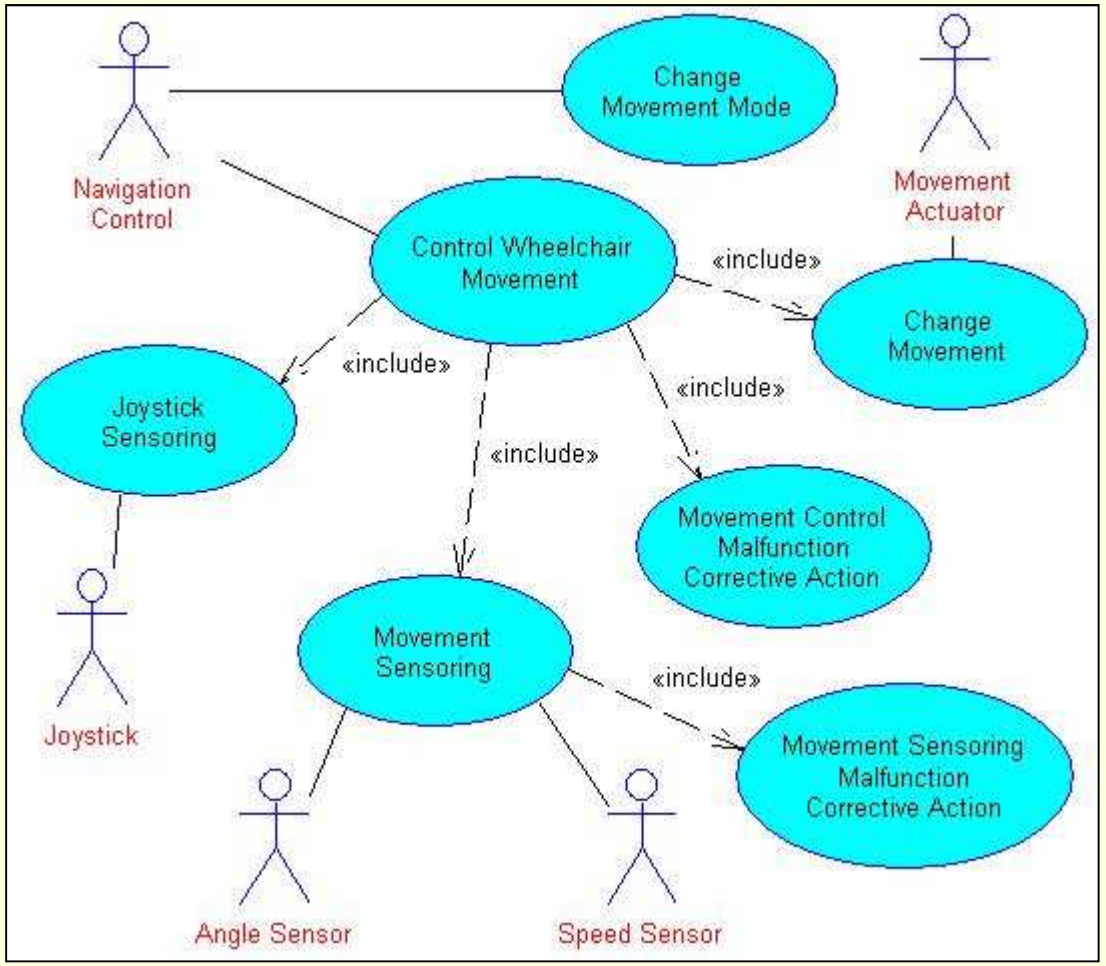
Real-Time UML – Modelo para Análise de Escalonabilidade



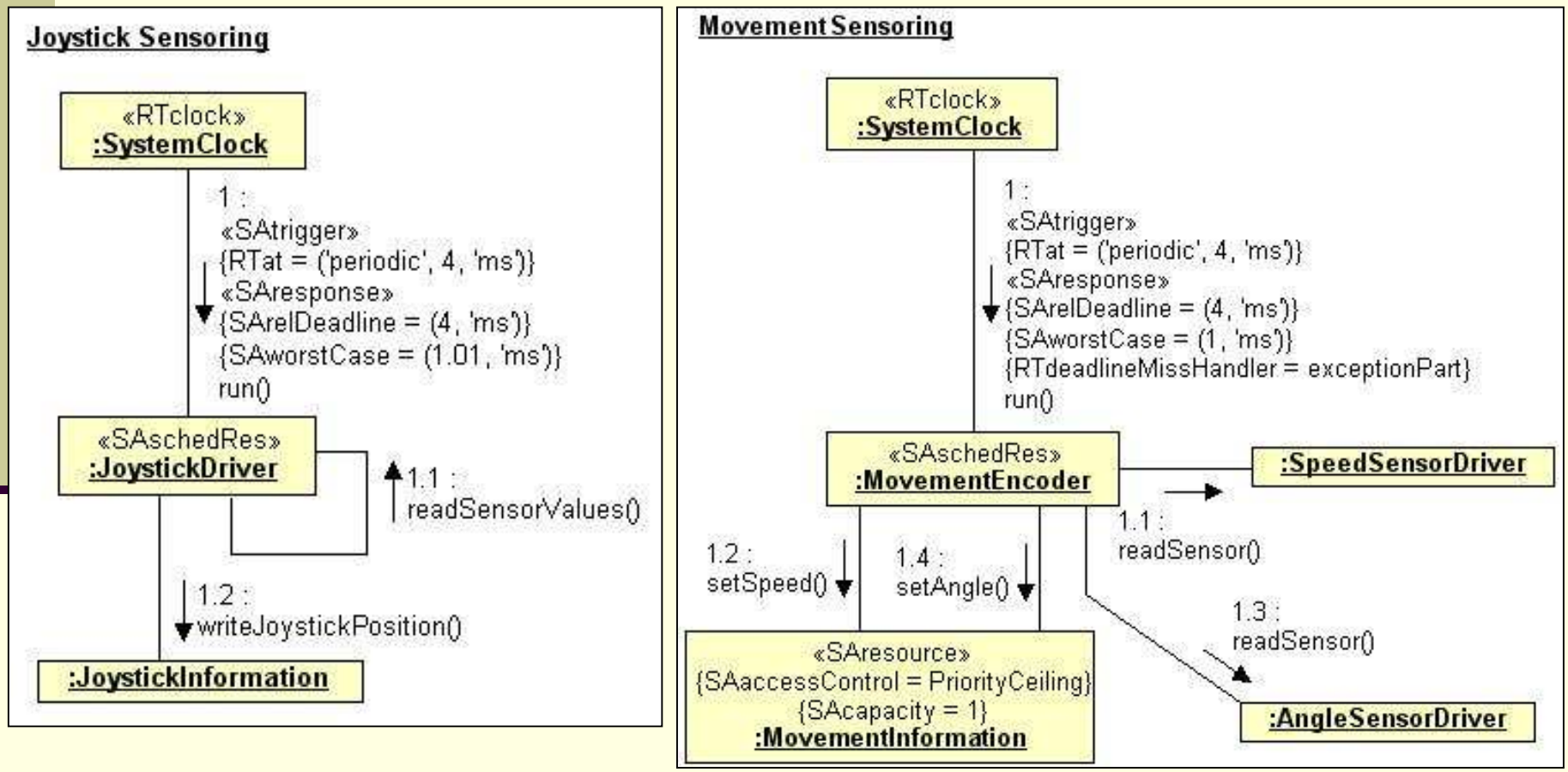
Estudo de Caso: Automação de uma Cadeira de Rodas



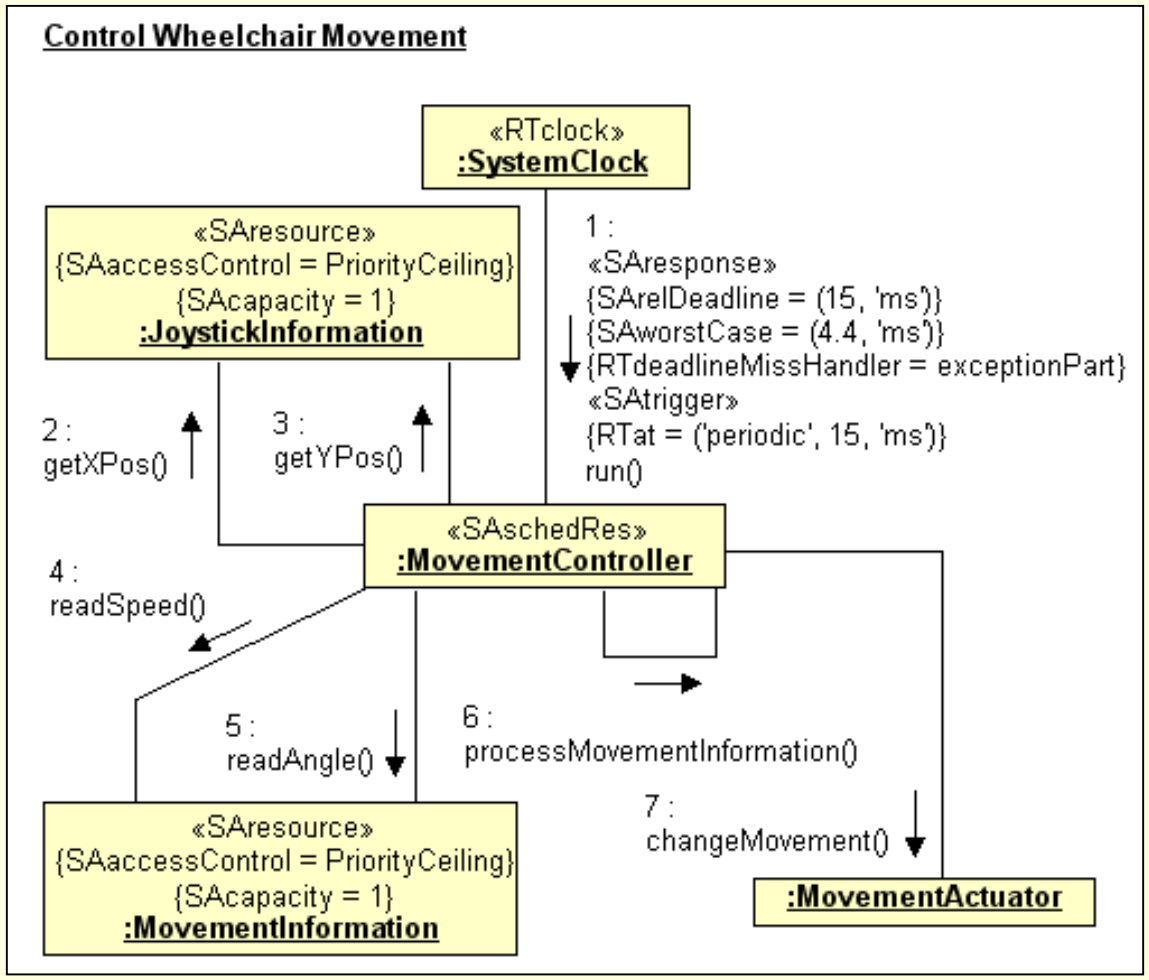
Controle de Movimento – Diagrama de Casos de Uso



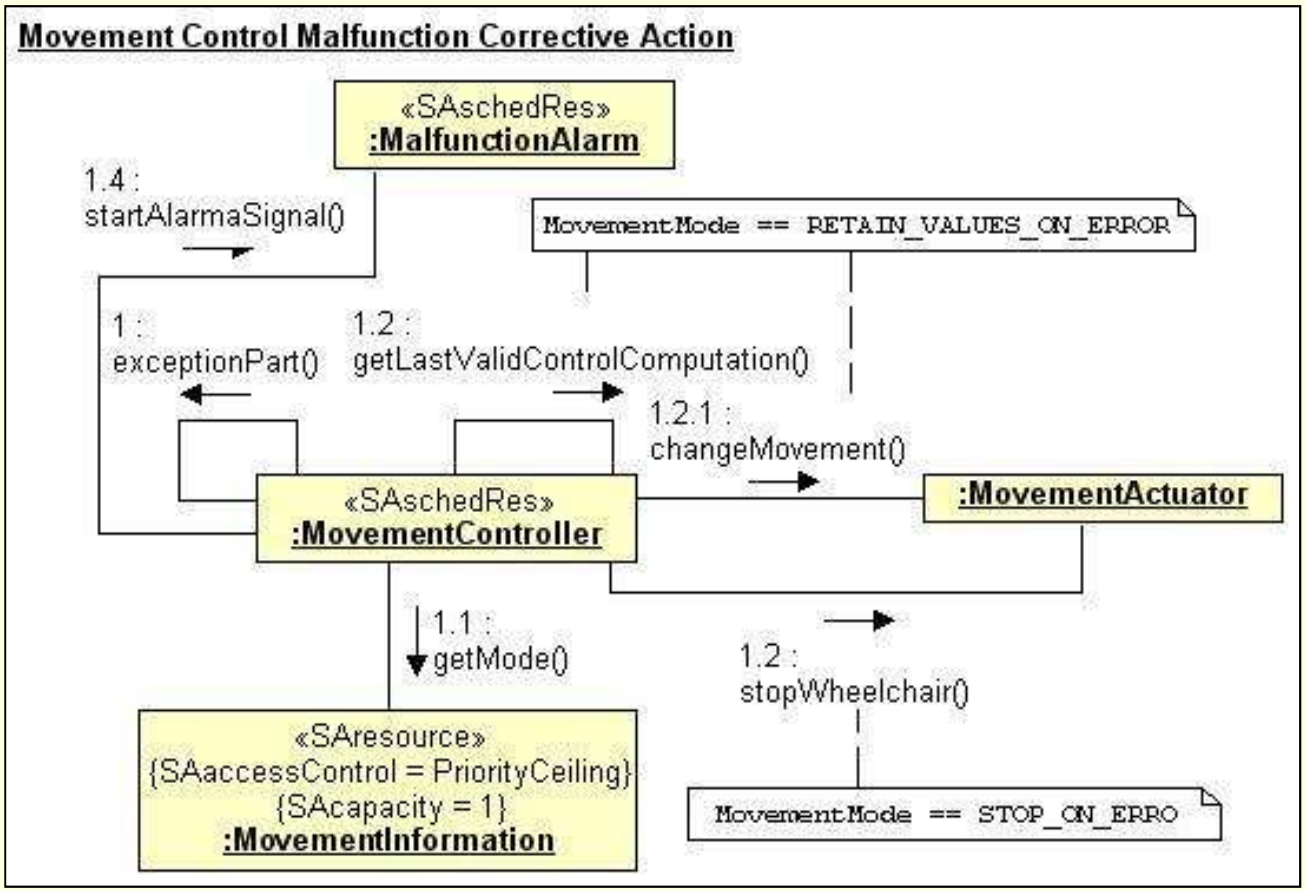
Leitura Sensores – Diagrama de Colaboração



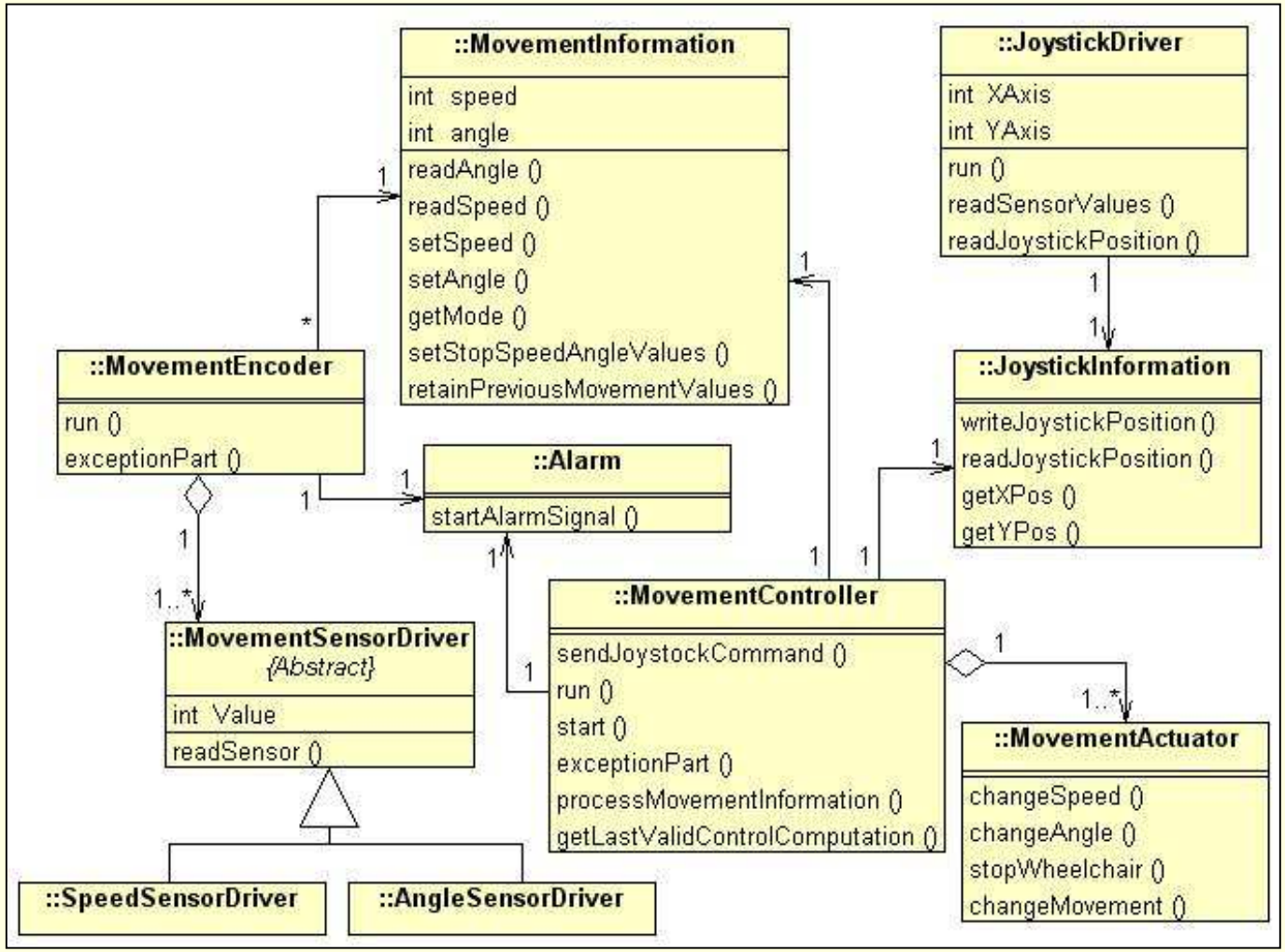
Controle/Atuação – Diagrama de Colaboração



Tolerância à Falhas – Diagrama de Colaboração

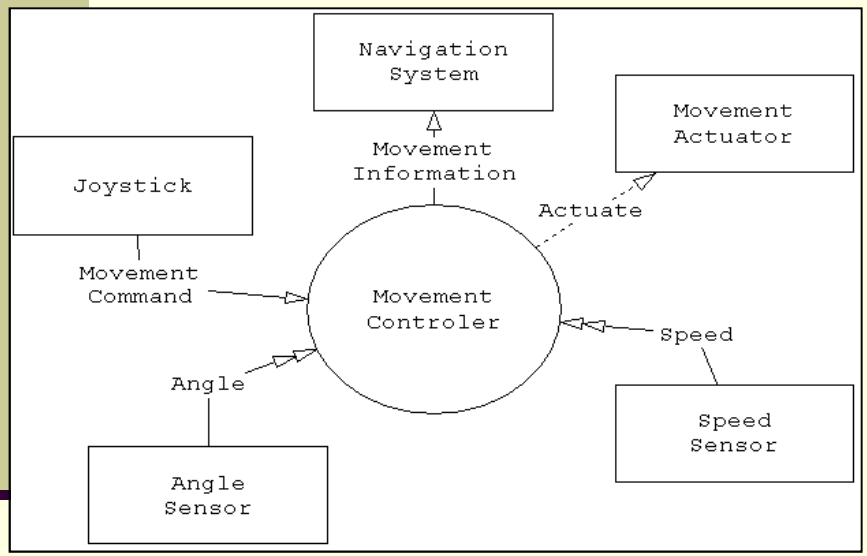


Estrutura do Controle de Movimento – Diagrama de Classes

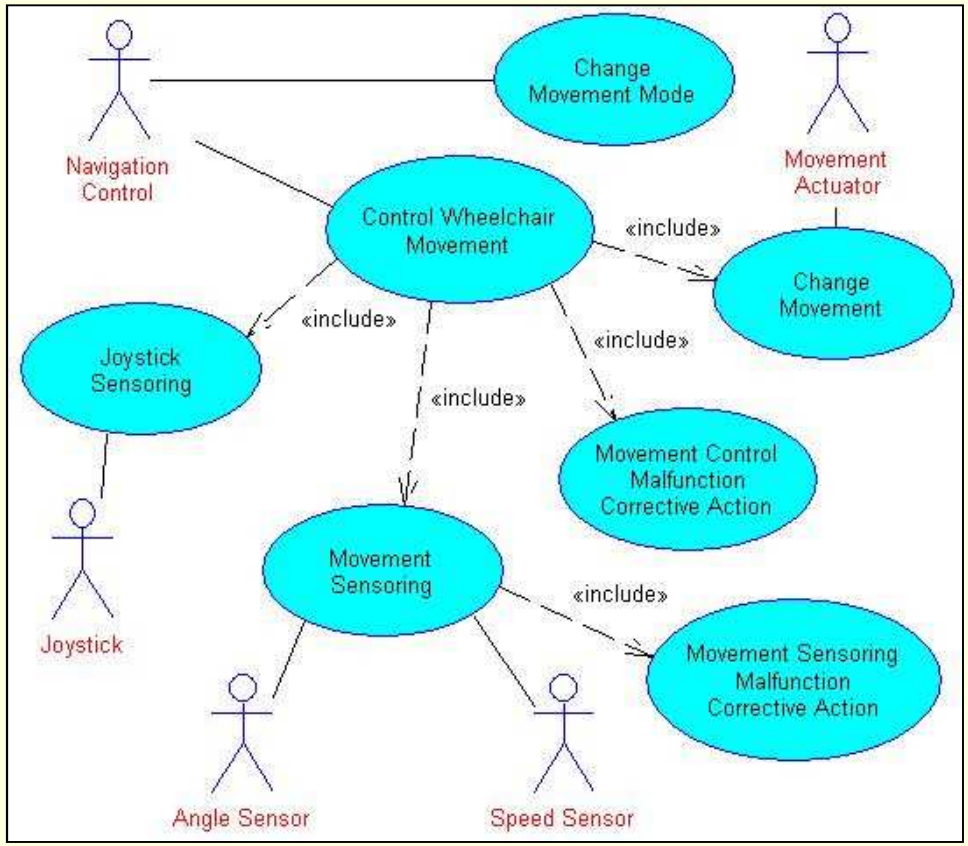


SA/RT vs. UML –
Visão Geral das Funcionalidades Sistema

SA/RT

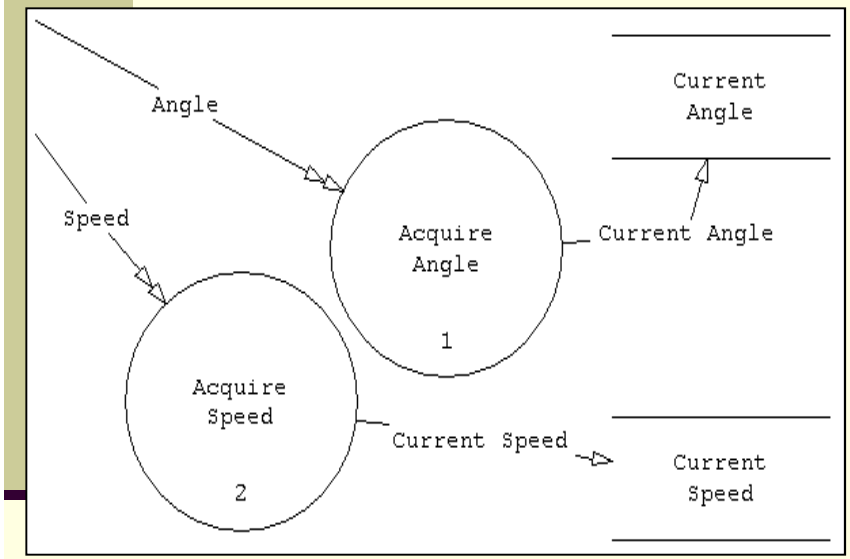


RT-UML

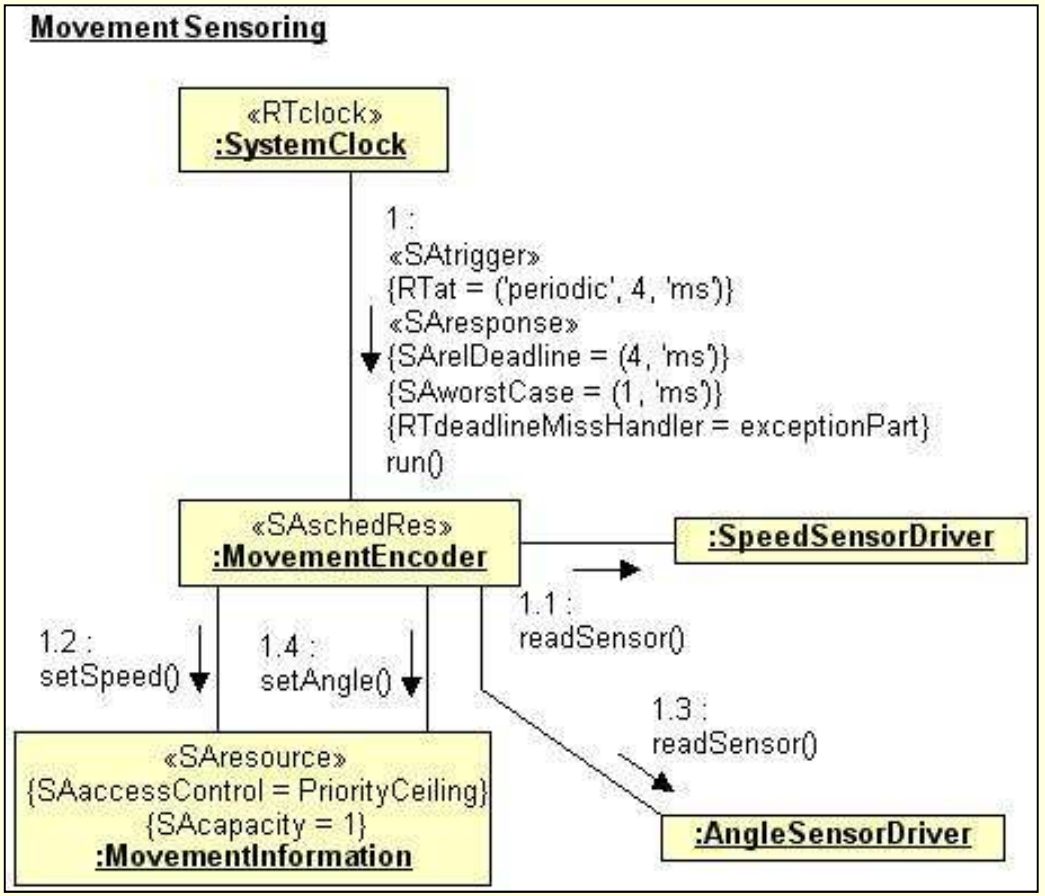


SA/RT vs. UML –
Modelagem do Comportamento do Sistema

SA/RT

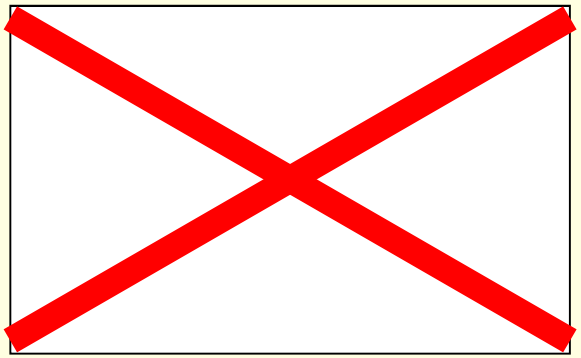


RT-UML

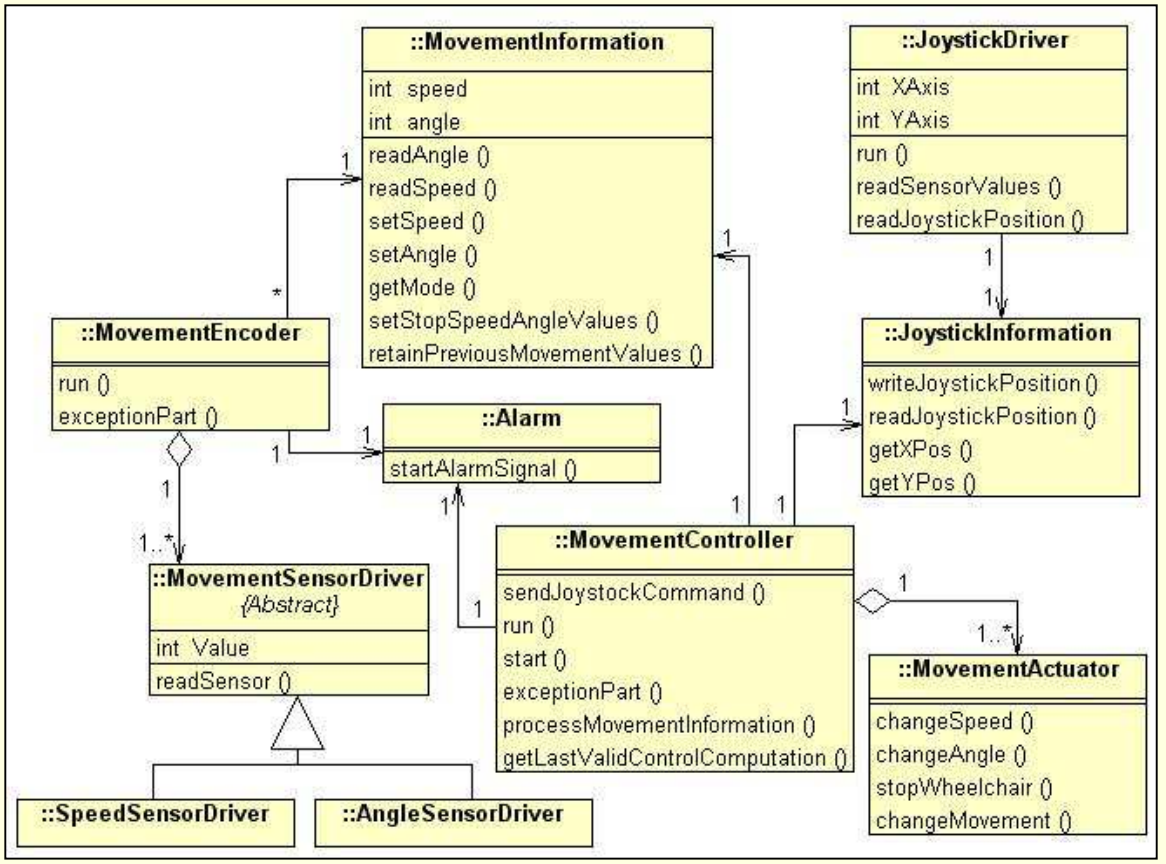


SA/RT vs. UML – Modelagem da Estrutura do Sistema

SA/RT



RT-UML



Perguntas ?!?