Aula 8

Professor:

Márcio Barros

Projeto de Software - Modelagem Dinâmica

Conteúdo:

- Diagramas de Seqüência
- Modelagem Dinâmica de Classes

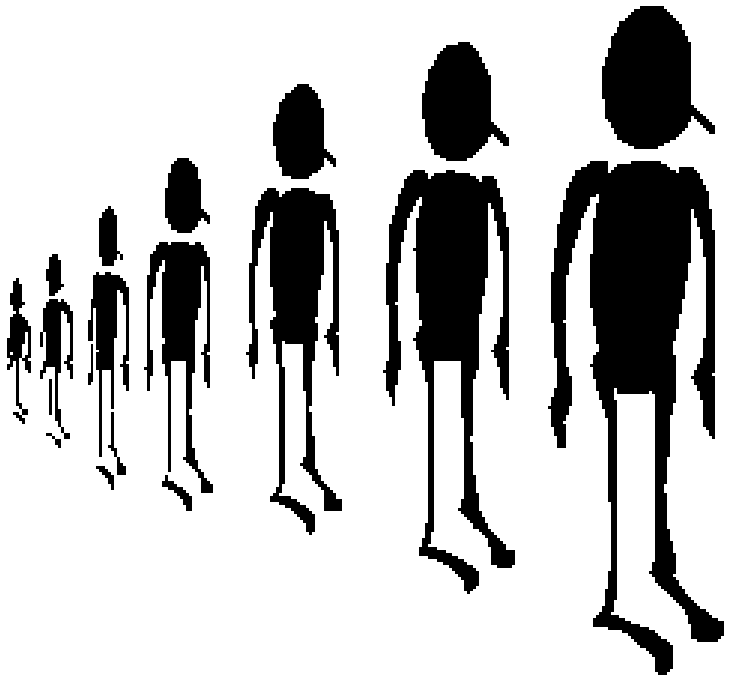
Fundação CECIERJ
Consórcio cederj

Diagramas Dinâmicos

- Existe uma forte ênfase na criação de diagramas estáticos
  - Em muitos casos, estes são os únicos diagramas criados durante um projeto de desenvolvimento de sistemas
  - Esta ênfase está associada ao uso de diagramas somente na etapa de análise de sistemas, onde a dinâmica relacionada com diversas classes pode não ficar clara (especialmente em sistemas de informação)
  - Esta ênfase não é apropriada, devendo-se lembrar que a orientação a objetos oferece mecanismos para o encapsulamento do comportamento das classes
  - Assim, a parte dinâmica é muito importante, especialmente na definição da arquitetura dos sistemas

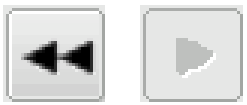
Diagramas de Seqüência

- Protocolos de Interação

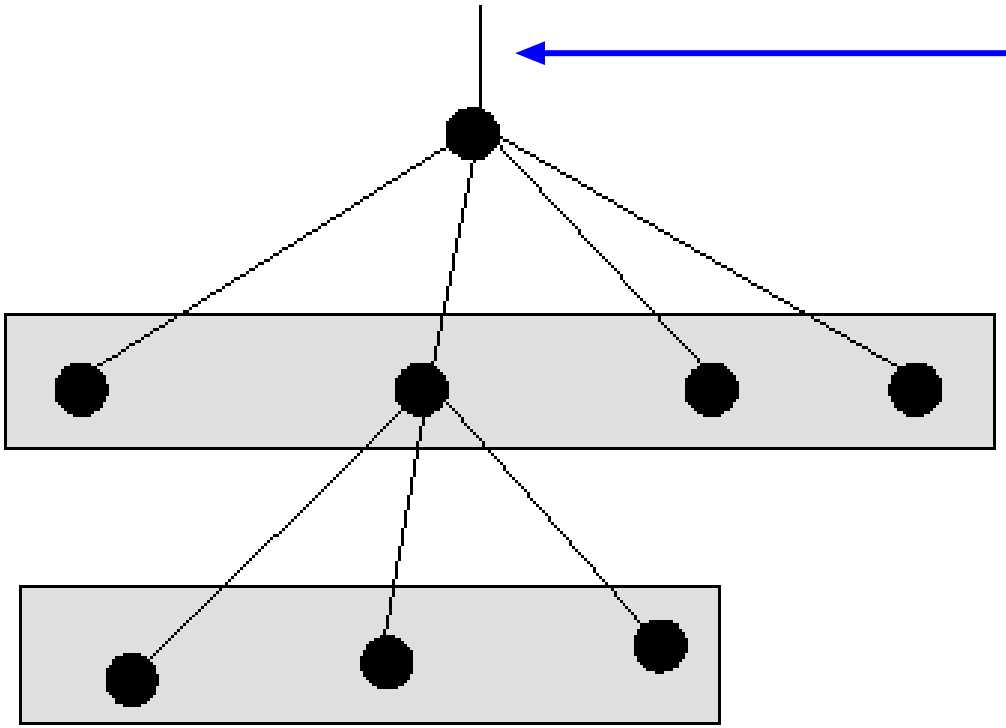


Cenários

- Um cenário é um caminho entre os fluxos de um use case
- Um use case é uma coleção de cenários
  - Um cenário é uma instância de um use case
  - Use cases são compostos de um fluxo principal e subfluxos
  - Subfluxos também podem se ramificar em subfluxos
  - As ramificações de fluxos e subfluxos formam uma árvore
  - Um cenário é um caminho da raiz até um nó folha da árvore de fluxos e subfluxos de um use case
  - O conjunto de cenários de um sistema é composto por todos os caminhos de todos os use cases



Descobrindo Atributos



Fluxo principal do use case

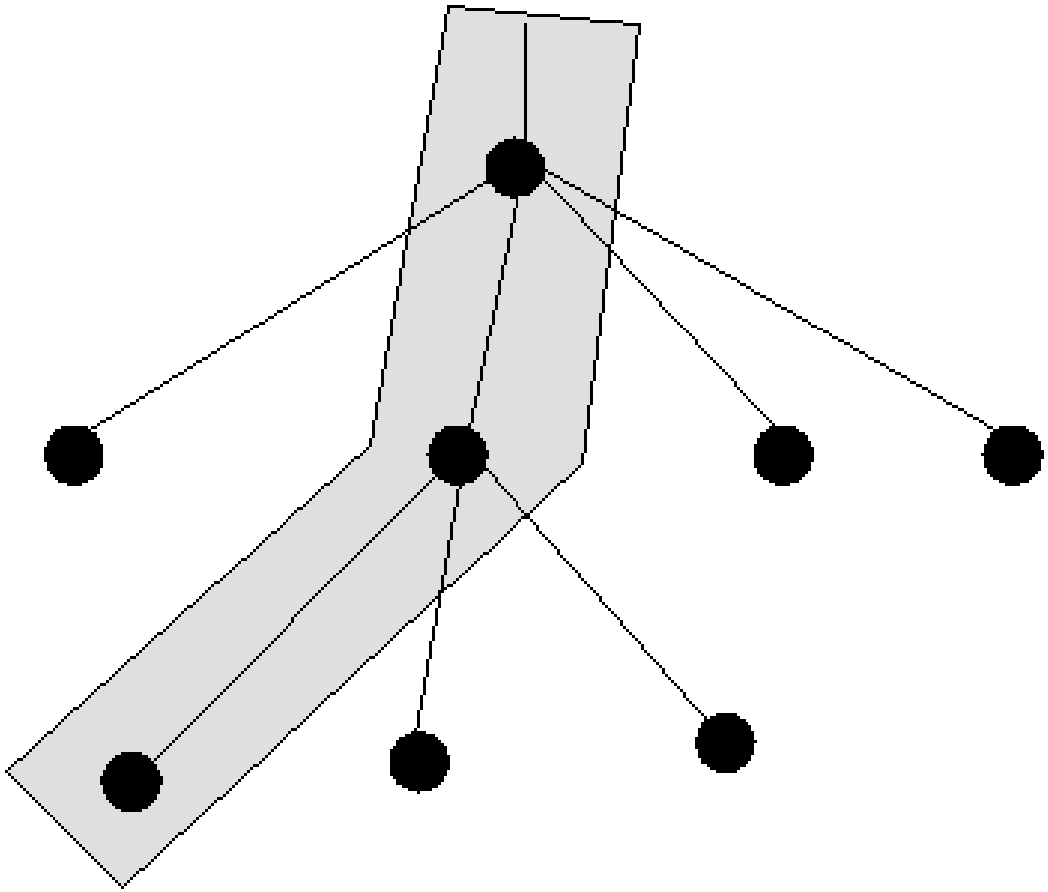
Subfluxos do fluxo principal

Ramificações de um subfluxo do use case

Subfluxos demonstram operações que podem ser selecionadas pelo usuário ou casos especiais de algumas operações.

Cenário

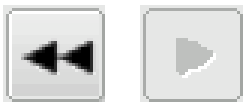
- Um cenário é um caminho da raiz até uma folha da árvore de ramificações de um use case



Objetivo

- Cenários tratam da linguagem do usuário
  - São utilizados na identificação dos elementos envolvidos nas operações do sistema no mundo real
  - Auxiliam na identificação das classes de objetos que devem interagir para concluir uma funcionalidade do sistema

- Cobertura
  - Cenários primários cobrem o fluxo principal dos use cases
  - Cenários secundários cobrem fluxos alternativos
  - Cenários de exceção cobrem fluxos de exceção e de erro



Cenários

- Desenvolvimento incremental
  - No desenvolvimento de um sistema, o analista deve começar sua investigação pelos cenários primários
  - Cenários secundários são progressivamente agregados
  - Cenários secundários relevantes devem ser estudados para evitar futuras alterações significativas nas principais classes

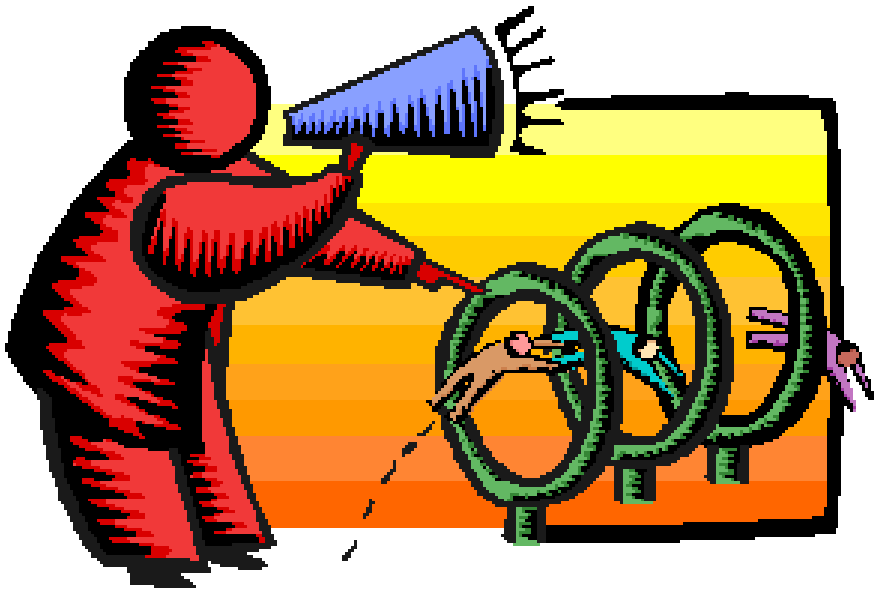
Diagramas de Seqüência

Objetivo

- Determinam os objetos responsáveis pela realização de um cenário e as mensagens que são trocadas entre eles
- O diagrama apresenta a ordem com que as mensagens são trocadas no tempo

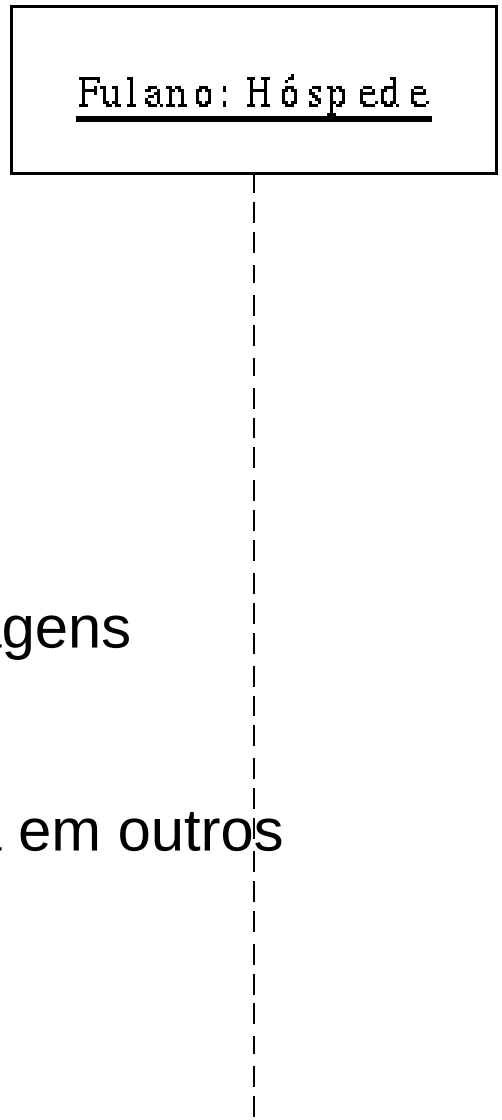
Composição

- Objetos
- Sequências



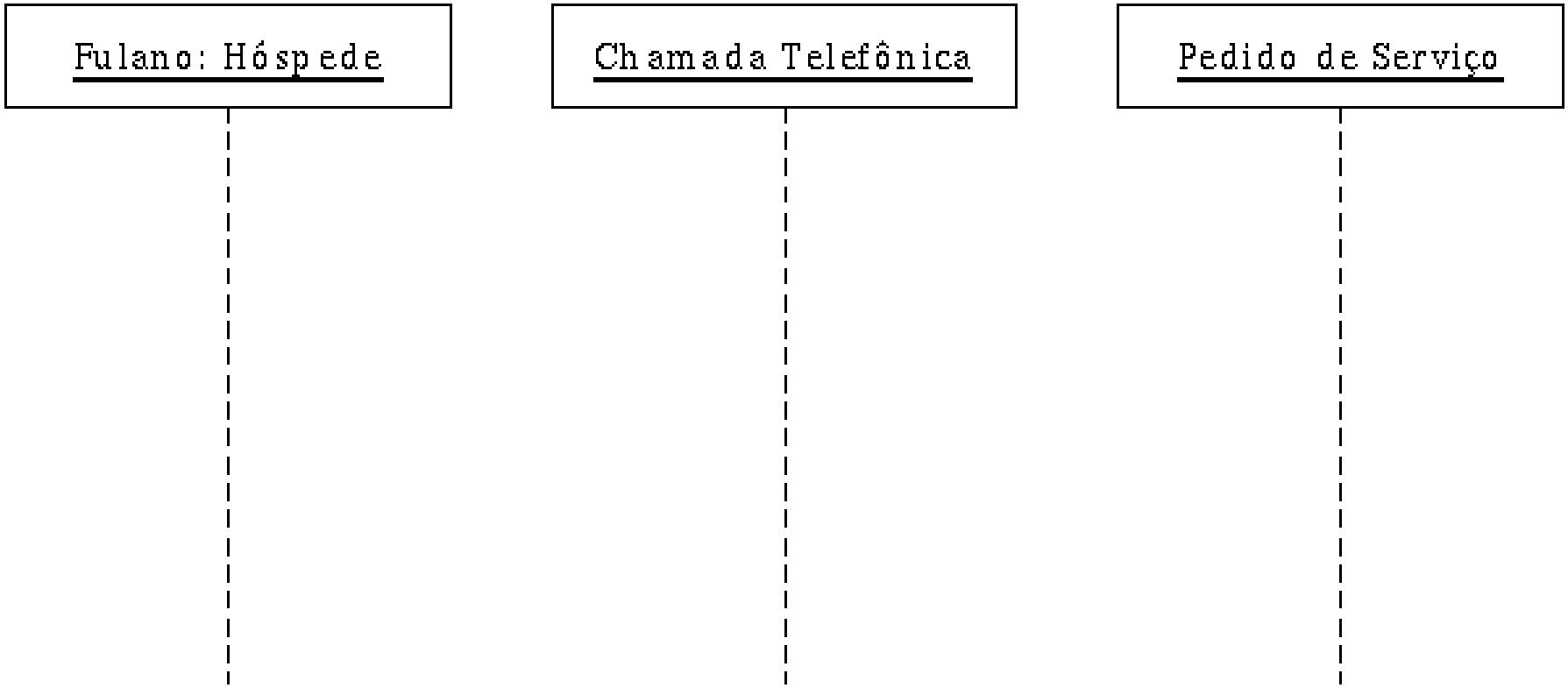
Objeto

- Representação
  - Objetos são representados por retângulos
  - Objetos devem ter um nome (sublinhado):
    - Nome de um objeto específico
    - Nome de objeto + nome da classe
    - Nome da classe (objeto anônimo)
  - A linha de vida define a seqüência de mensagens recebidas ou emitidas pelo objeto no tempo
  - Objetos podem aparecer sem a linha de vida em outros diagramas



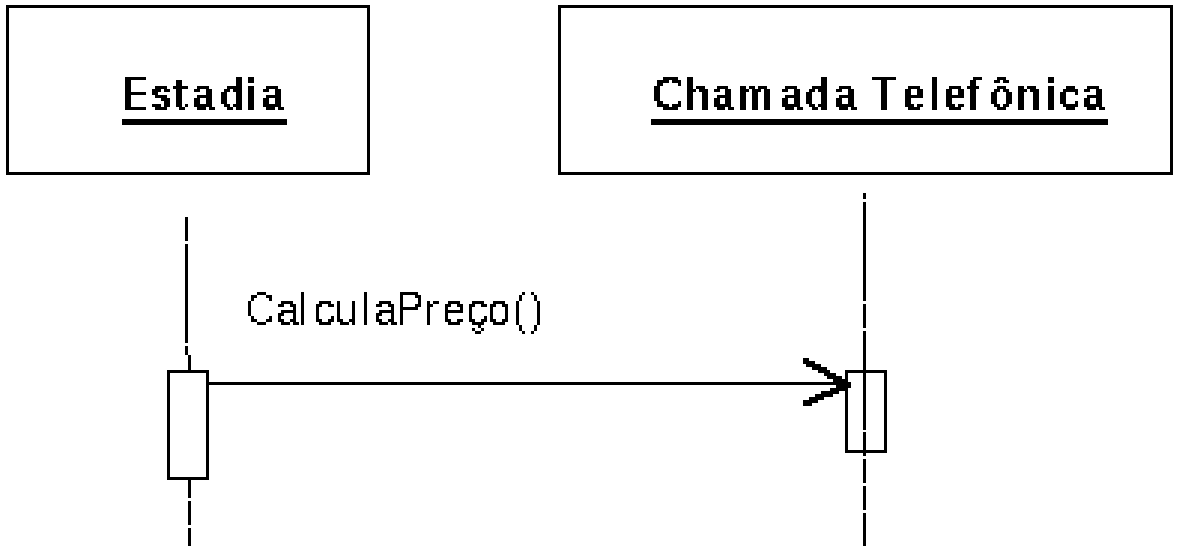
Objetos

- Um diagrama de seqüências contém diversos objetos
  - Os objetos são alinhados no topo do diagrama
  - Suas linhas de vida descem em direção à base do diagrama



Mensagens

- Representação
  - Mensagens são representadas como setas entre dois objetos
  - As setas devem conter nomes de métodos do objeto destino



Foco de Controle

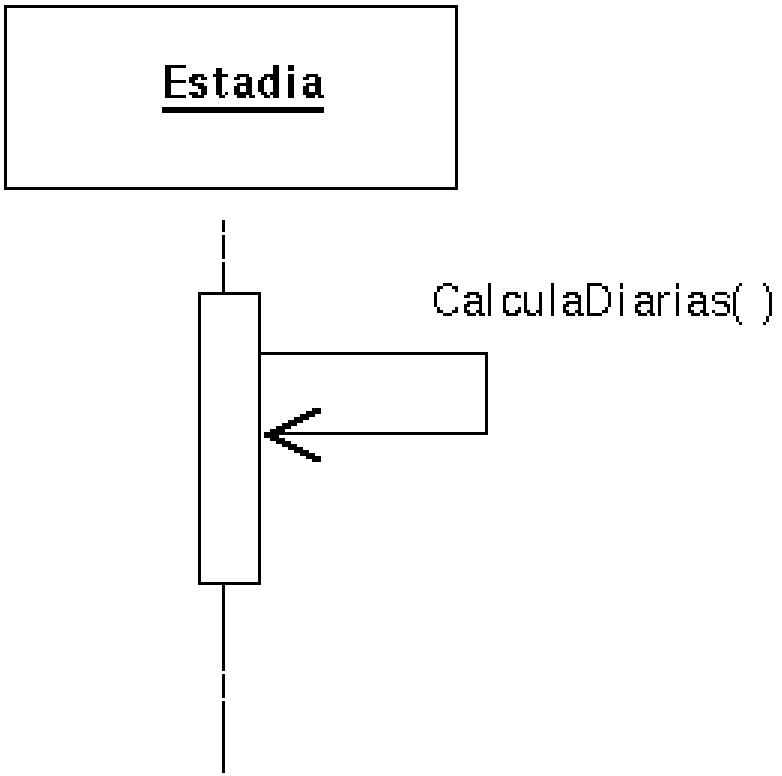
- Retângulos na linha de vida de um objeto
  - Representam o tempo em que um objeto está ativo na troca de mensagens
  - Representam o tempo em que o objeto é responsável pelo fluxo de controle do cenário
  - Focos de controle são opcionais nos diagramas de sequência

Complementos

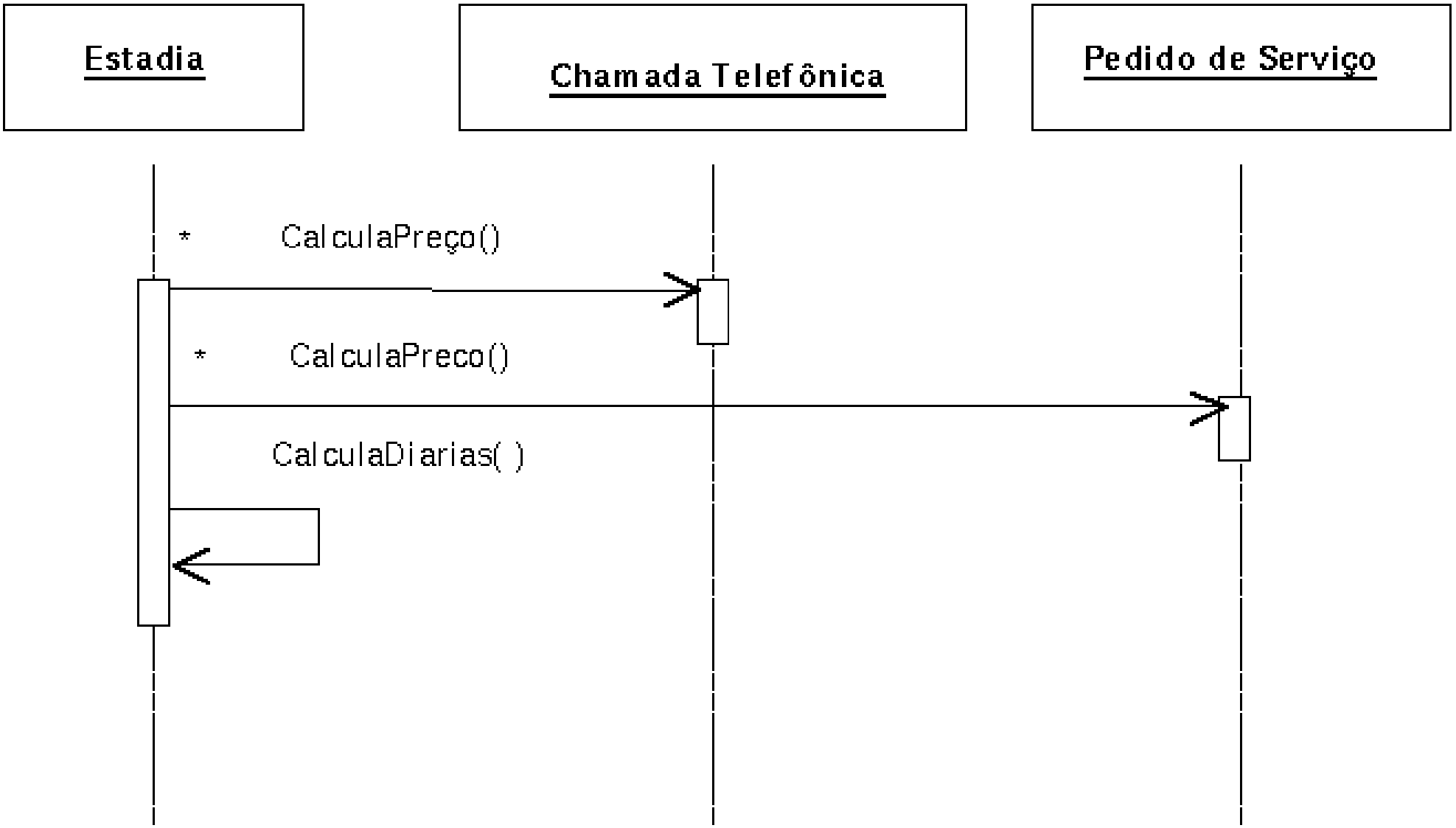
- Uma mensagem pode ter dois tipos de complementos
  - Condição:
    - A mensagem somente será emitida se a condição for verdadeira
    - A condição é expressa entre colchetes antes do nome do método
  - Repetição
    - A mensagem será emitida múltiplas vezes
    - Um asterisco é apresentado antes do nome do método

Auto-Mensagem

- Ocorre quando um objeto chama um método seu para realizar parte do cenário
  - Auto-mensagens são representadas por setas saindo e retornando para o próprio objeto



Exemplo


Estadia
Chamada Telefônica
Pedido de Serviço
* CalculaPreço()
* CalculaPreco()
CalculaDiarias( )

Descobrindo Diagramas de Seqüência

- Durante a análise ?
  - Devemos procurar por diagramas de seqüência que envolvam operações algoritmicamente complexas
  - Operações que criam, editam ou apenas consultam objetos devem ser postergadas para a fase de projeto

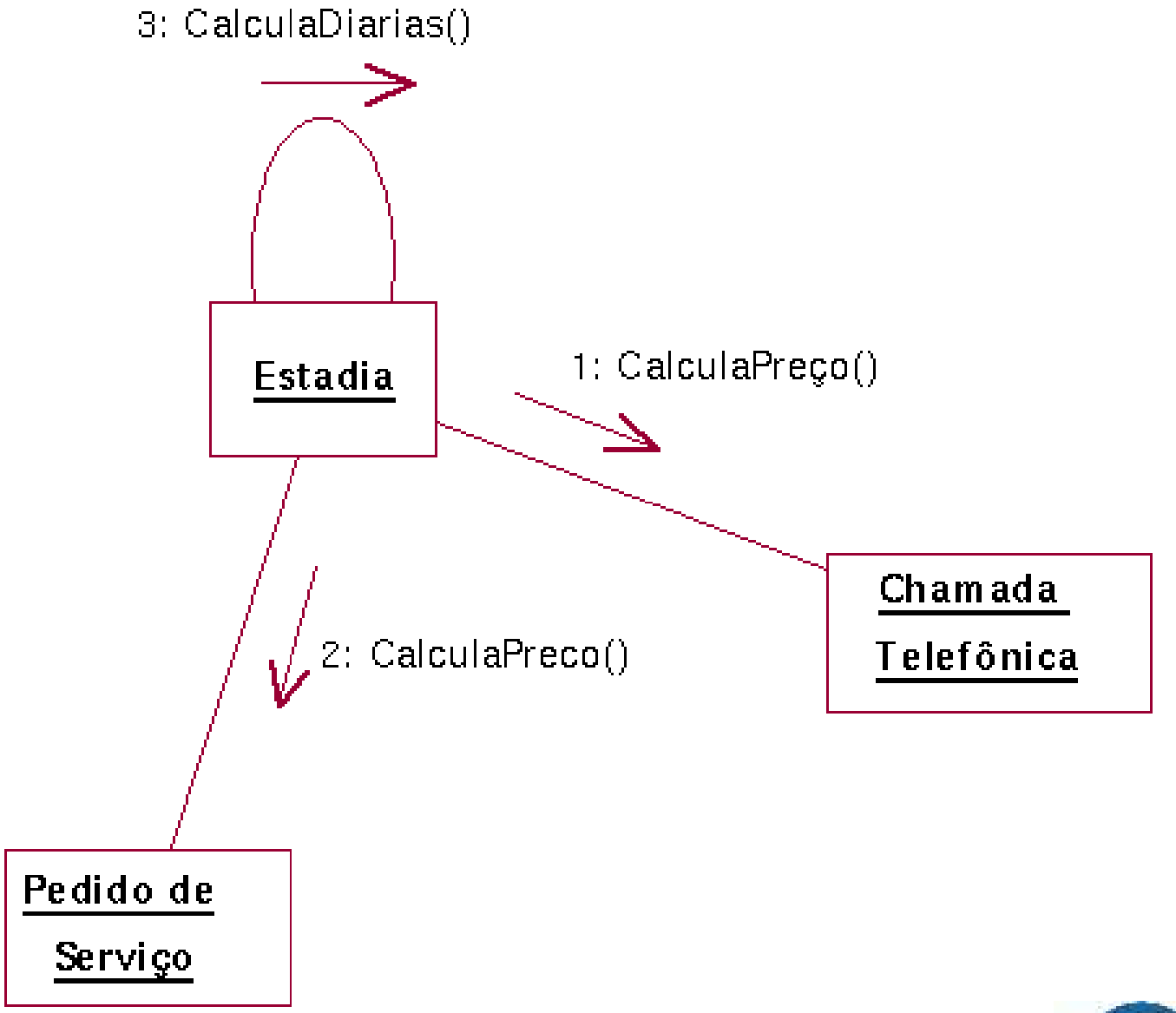
- Durante o projeto ?
  - Diagramas de seqüência devem demonstrar as relações entre as janelas da aplicação, os relatórios, o banco de dados e as operações realizadas pelos use cases

Diagramas de Colaboração

- Alternativa ao diagrama de seqüência

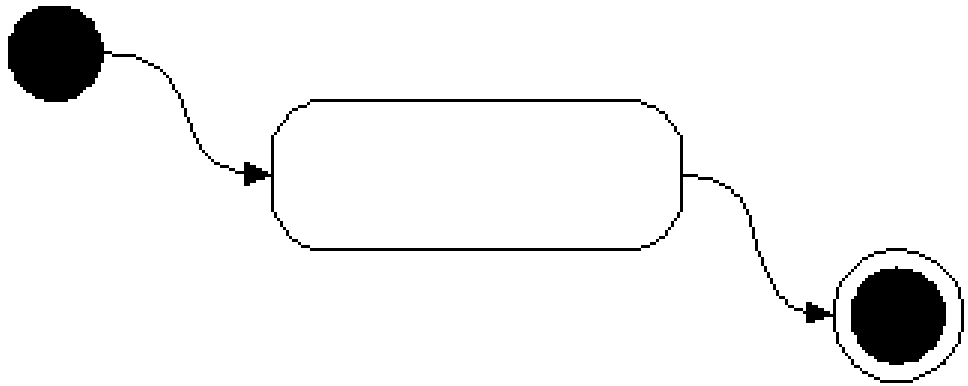
- Não apresenta a linha de tempo dos objetos
  - Objetos representados por retângulos
  - Mensagens representadas como setas entre os retângulos
  - Mensagens são numeradas (seqüência de tempo)

Exemplo (Encerramento de Estadia)



ESTADOS

- Modelagem Dinâmica de Classes



Diagramas de Estado

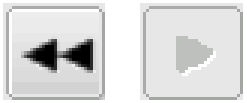
- Modelam comportamento dinâmica das classes
  - Diagramas de seqüência modelam o comportamento de um use case no tempo
  - Diagramas de estado modelam o comportamento de uma classe no tempo

- São utilizados apenas em classes com estados relevantes
  - Classes que monitoram eventos ao longo de sua vida
  - Serviços de monitoração algoritmicamente complexos
  - Estudar operações realizadas sobre as classes ou pelas classes

Diagramas de Estado

- São utilizados por diversos métodos de análise
  - Análise estruturada moderna
  - Análise essencial
  - Outros métodos de análise orientada a objetos

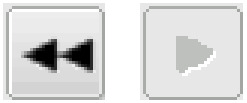
- Componentes
  - Estado
  - Transição
  - Estado Inicial
  - Estado Final



Estados

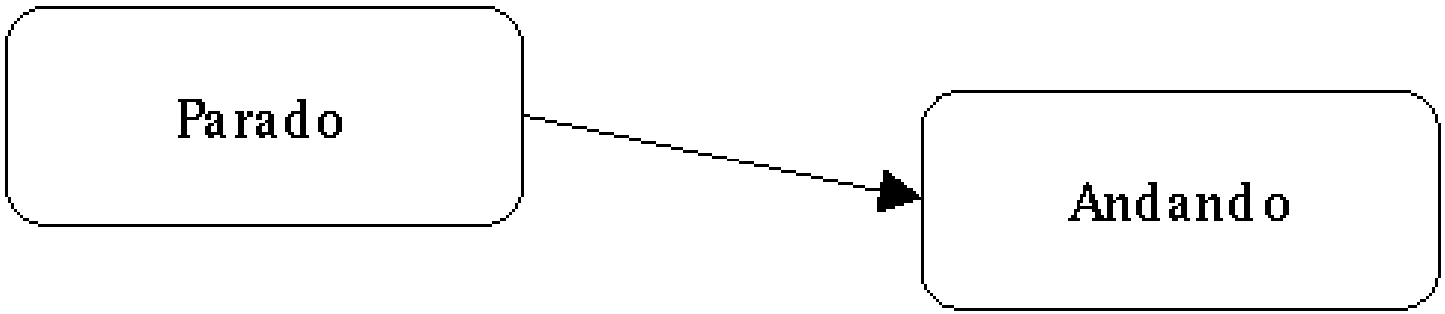
- Um estado representa uma condição em que o objeto pode se encontrar durante sua existência no sistema
  - O estado de um objeto muda ao longo do tempo
  - O estado é definido pelos valores dos atributos do objeto
  - Determina eventos, ações e condições possíveis no objeto

- Uma transição representa uma mudança de estado
  - Transições podem ser automáticas ou geradas por eventos
  - Transições não podem ser interrompidas e custam tempo zero.



Estados

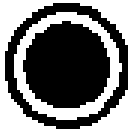
- Representação UML
  - Estado: representado por um retângulo de bordas arredondadas
  - Transição: representada por uma seta entre os estados
  - O nome do estado é apresentado no interior do retângulo
  - Nas transições geradas por eventos, a seta indica o nome do evento que provocou a mudança de estado



Estados Especiais

Um diagrama de estados tem no mínimo 2 estados:

Estado Inicial: um objeto recém criado no sistema se encontra neste estado.

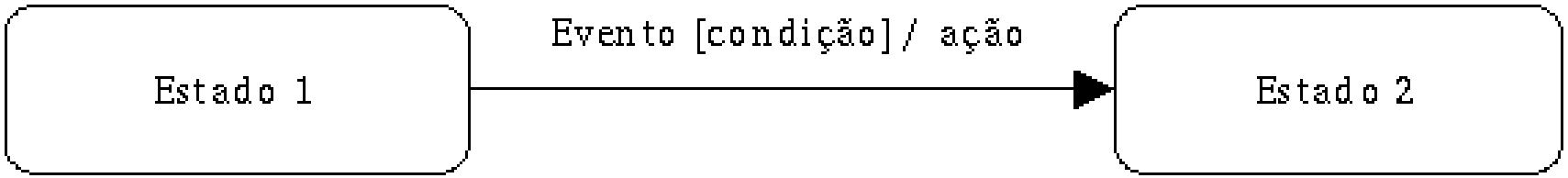


Estado Final: estado final na cadeia de troca de estados do objeto. O objeto não poderá trocar de estado após atingir seu estado final. Em um diagrama de estados podem existir diversos estados finais.

Transições

Uma transição pode estar associada a:

- Uma ação, que indica um método do objeto que será executado quando a transição de estado se realizar
- Uma condição, também conhecida como guarda, que indica quando a transição de estado deve ocorrer
- Ambos são apresentados junto ao nome do evento na transição, como na figura abaixo
- Ambos são opcionais



Detalhes de Estados

- Estados podem estar associado a ações
  - Uma ação de entrada (entry) indica uma operação realizada sempre que ocorre uma transição para dentro do estado
  - Uma ação de saída (exit) indica uma operação realizada sempre que ocorre uma transição para fora do estado
  - Ações são apresentadas no interior do estado, abaixo de seu nome

Descobrindo Estados

- Uma classe possui estados quando:
  - Possui diversas operações que são dependentes entre si
    - Uma venda somente pode ser entregue depois de registrada
    - Uma venda somente pode ser paga depois de entregue
    - Uma venda somente pode ser cancelada depois de registrada
    - Registrada, entregue, paga e cancelada são estados da venda
  - As características de um objeto influenciam o seu comportamento ou o comportamento de outra classe
    - Uma venda somente pode ser entregue depois de registrada
    - Vendas nunca são realizadas para clientes sem crédito
    - Um cliente devedor ou sem crédito não pode ser removido do sistema
    - "Não devedor", "devedor" ou "sem crédito" são estados do cliente

Descobrindo Estados

- Análise das operações do sistema
  - Classes que têm mais que as quatro operações básicas
    - Determine que operações precedem outras operações
    - Crie um estado para representar o objeto após cada operação
    - Determine as transições de estado
  - Verifique as restrições das operações
    - Em cada classe, verifique se suas operações dependem de condições dos atributos de seus objetos ou de objetos de outras classes
    - Crie estados nos respectivos objetos dependentes, representando as condições em que as operações da outra classe são aplicáveis