Princípios de Análise e Projeto de Sistemas com UML

2ª edição

Eduardo Bezerra

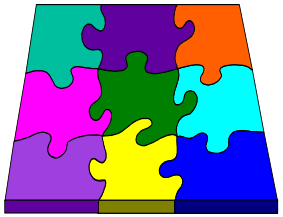
Editora Campus/Elsevier

Capítulo 7
Modelagem de Interações

“Somente após a construção de diagramas de interação para os cenários de um caso de uso, pode-se ter certeza de que todas as responsabilidades que os objetos devem cumprir foram identificadas”

-Ivar Jacobson.

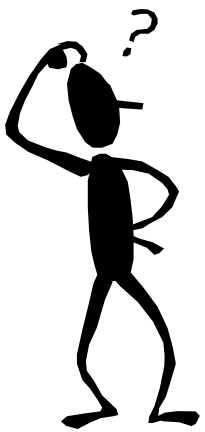
Tópicos



- Introdução
- Diagrama de seqüência
- Diagrama de comunicação
- Modularização de interações
- Construção do modelo de interações
- Modelo de interações em um processo iterativo

Introdução

- O objetivo dos modelos vistos até agora é fornecer um entendimento do problema correspondente ao SSOO a ser desenvolvido.
- Entretanto, esses modelos deixam algumas perguntas sem respostas.



- No modelo de casos de uso:
  - Quais são as operações que devem ser executadas internamente ao sistema?
  - A que classes estas operações pertencem?
  - Quais objetos participam da realização deste caso de uso?

Introdução

- No modelo de classes de análise:
  - De que forma os objetos colaboram para que um determinado caso de uso seja realizado?
  - Em que ordem as mensagens são enviadas durante esta realização?
  - Que informações precisam ser enviadas em uma mensagem de um objeto a outro?
  - Será que há responsabilidades ou mesmo classes que ainda não foram identificadas?
- Sessões CRC pode ajudar a identificar quais são as responsabilidades de cada objeto e com que outros objetos ele precisa colaborar.
  - Mas sessões CRC não fornecem um modo de documentar essas interações.

Introdução

- Para responder às questões anteriores, o ***modelo de interações*** deve ser criado.
- Esse modelo representa mensagens trocadas entre objetos para a execução de cenários dos casos de uso do sistema.
- A construção dos ***diagramas de interação*** é uma consolidação do entendimento dos aspectos dinâmicos do sistema, iniciado nas sessões CRC.
- A modelagem de interações é uma parte da ***modelagem dinâmica*** de um SSOO.

Diagramas de interação representam como o sistema age internamente para que um ator atinja seu objetivo na realização de um caso de uso. A modelagem de um SSOO normalmente contém diversos diagramas de interação. O conjunto de todos os diagramas de interação de um sistema constitui o seu ***modelo de interações***.

Introdução

- Os objetivos da construção do modelo de interação são:
  1. Obter informações adicionais para completar e aprimorar outros modelos (principalmente o modelo de classes)
     - Quais as operações de uma classe?
     - Quais os objetos participantes da realização de um caso de uso (ou cenário deste)?
     - Para cada operação, qual a sua assinatura?
     - Uma classe precisa de mais atributos?
  2. Fornecer aos programadores uma visão detalhada dos objetos e mensagens envolvidos na realização dos casos de uso.

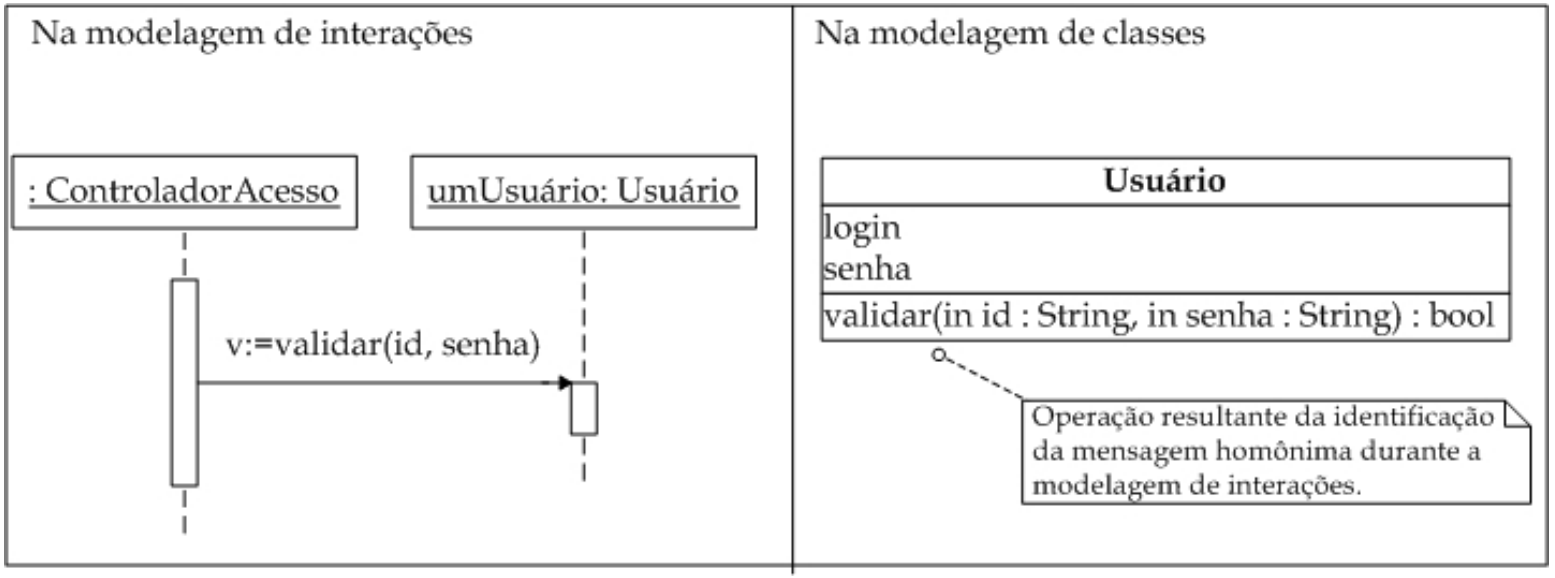
Mensagem

- O conceito básico da interação entre objetos é a ***mensagem***.
- Um sistema OO é uma rede de objetos que trocam mensagens.
  - Funcionalidades são realizadas pelos objetos, que só podem interagir através de mensagens.
  - Um objeto envia uma mensagem para outro objeto quando o primeiro deseja que o segundo realize alguma tarefa.
- O fato de um objeto "precisar de ajuda" indica a necessidade de este enviar mensagens.
- Na construção de diagramas de interação, mensagens de um objeto a outro implicam em operações que classes devem ter.

Uma mensagem representa a requisição de um objeto remetente a um objeto receptor para que este último execute alguma operação definida para sua classe. Essa mensagem deve conter informação suficiente para que a operação do objeto receptor possa ser executada.

Mensagens *versus* responsabilidades

- Qual o objetivo da construção dos diagramas de interação?
  - Identificar **mensagens** e, em última análise, **responsabilidades (operações e atributos)**



Uma mensagem implica na existência de uma operação no objeto receptor. A resposta do objeto receptor ao recebimento de uma mensagem é a execução da operação correspondente.

Sintaxe da UML para mensagens

- Na UML, o rótulo de uma mensagem deve seguir a seguinte sintaxe:

[[expressão-seqüência] controle:] [v :=] nome [(argumentos)]

- Onde o termo **controle** pode ser uma condição ou um iteração:

‘*’ ‘[’ cláusula-iteração ‘]’

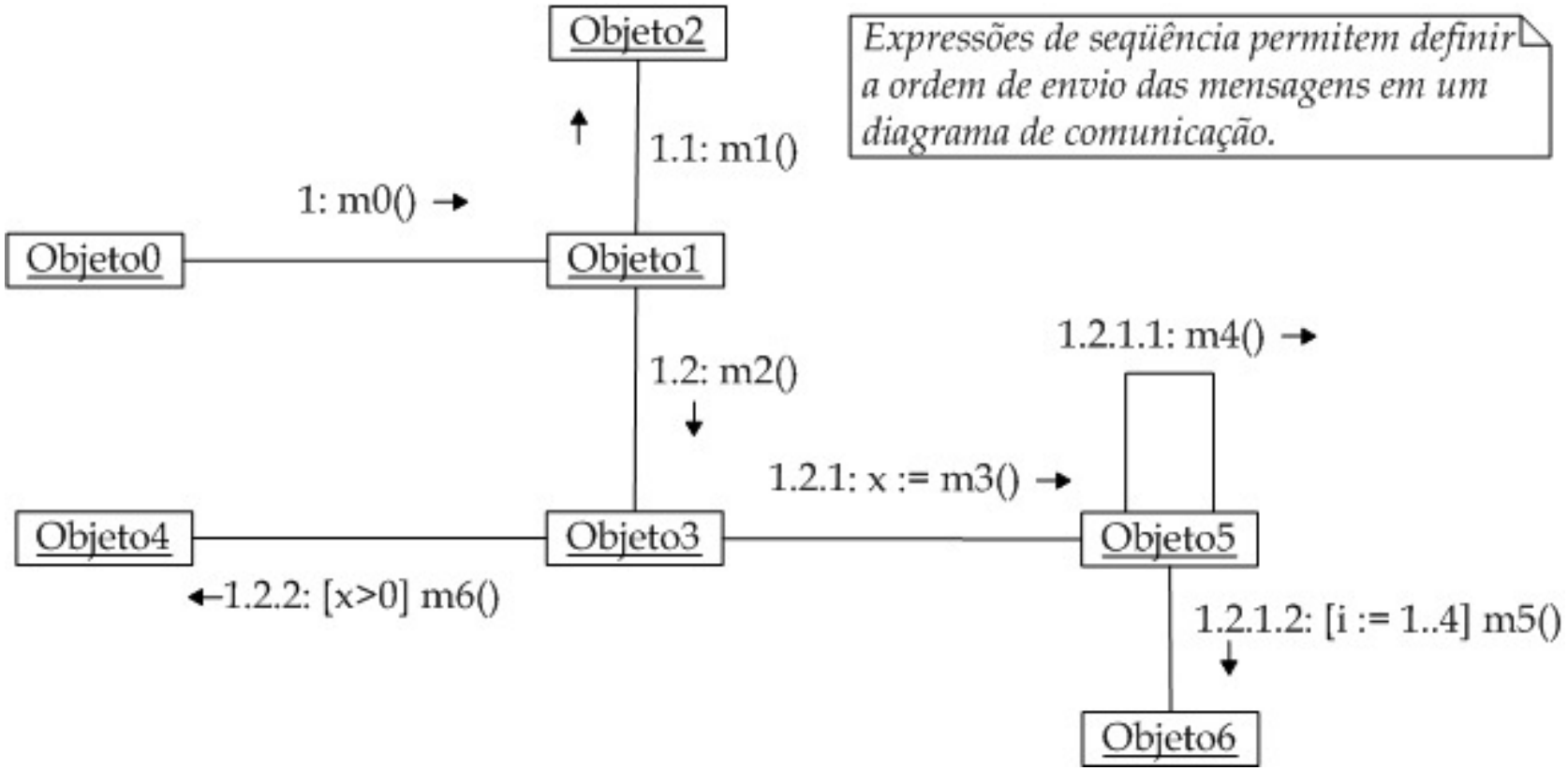
‘[’ cláusula-condição ‘]’

- O único termo obrigatório corresponde ao **nome** da mensagem.

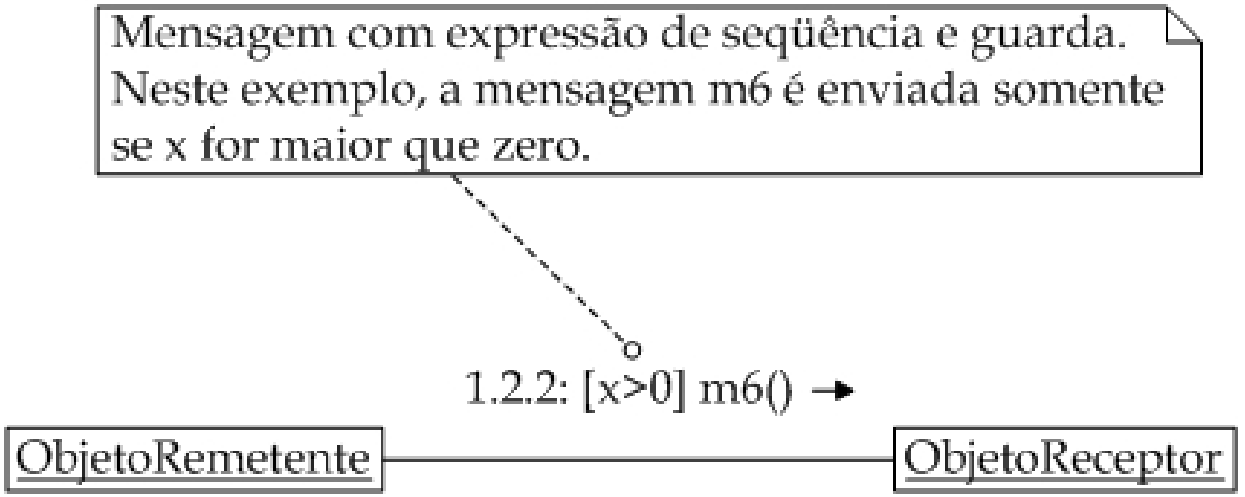
Exemplos (sintaxe UML para mensagens)

- Mensagem simples, sem cláusula alguma.
  1: adicionarItem(item)
- Mensagem com cláusula de condição.
  3 [a > b]: trocar(a, b)
- Mensagem com cláusula de iteração e com limites indefinidos.
  2 *: desenhar( )
- Mensagem com cláusula de iteração e com limites definidos.
  2 *[i := 1..10]: figuras[i].desenhar( )
- Mensagem aninhada com retorno armazenado na variável x.
  1.2.1: x := selecionar(e)

Exemplos (sintaxe UML para mensagens)

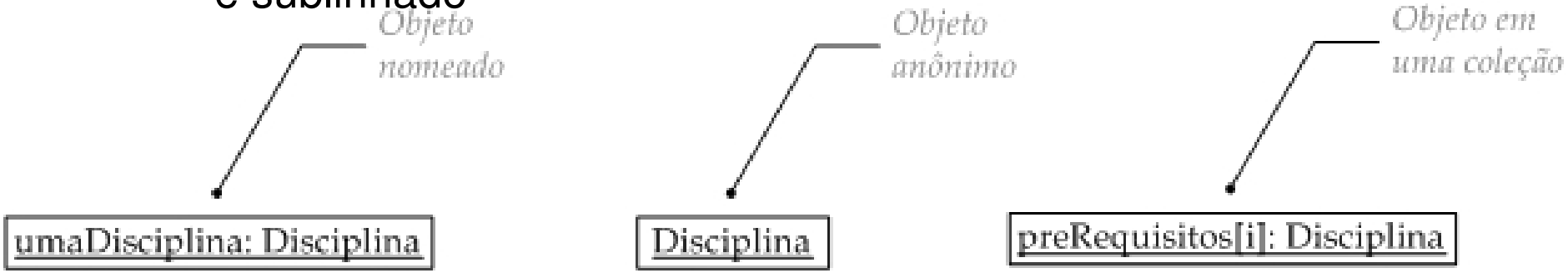


Exemplos (sintaxe UML para mensagens)

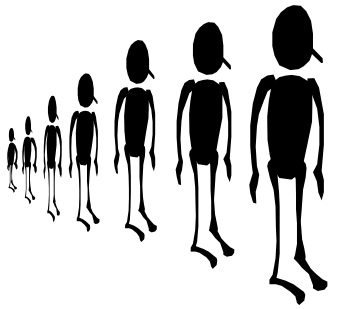


Notação para objetos

- Objetos são representados em um diagrama de interação utilizando-se a mesma notação do diagrama de objetos.
- Pode-se representar objetos anônimos ou objetos nomeados, dependendo da situação.
- Elementos de uma coleção também podem ser representados.
- Classes também podem ser representadas.
  - Para o caso de mensagens enviadas para a classe.
  - Uma mensagem para uma classe dispara a execução de uma *operação estática*.
  - A representação de uma classe em um diagrama de seqüência é a mesma utilizada para objetos, porém o nome da classe não é sublinhado



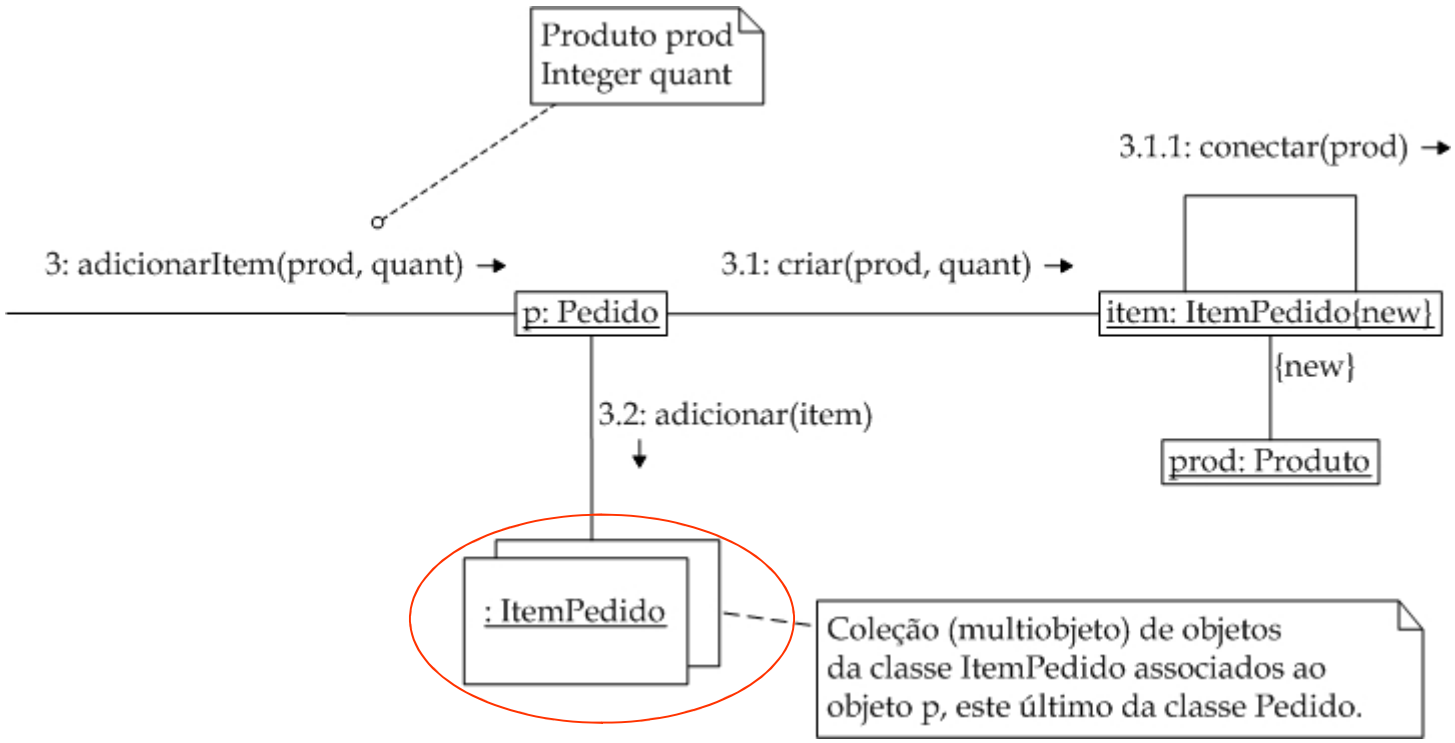
Multiobjetos



- Um **multiobjeto** é o nome que a UML dá para uma *coleção* de objetos de uma mesma classe. Pode ser utilizado para:
  - representar o lado muitos de uma associação de conectividade um para muitos.
  - representar uma lista (temporária ou não) de objetos sendo formada em uma colaboração.
- Um multiobjeto é representado na UML através de dois retângulos superpostos.
  - A superposição dos retângulos evita a confusão com a notação usada para objetos.
  - O nome do multiobjeto é apresentado no retângulo que fica por cima e segue a mesma nomenclatura utilizada para objetos.
  - Convenção: usar o nome da classe de seus elementos para nomear o multiobjeto.

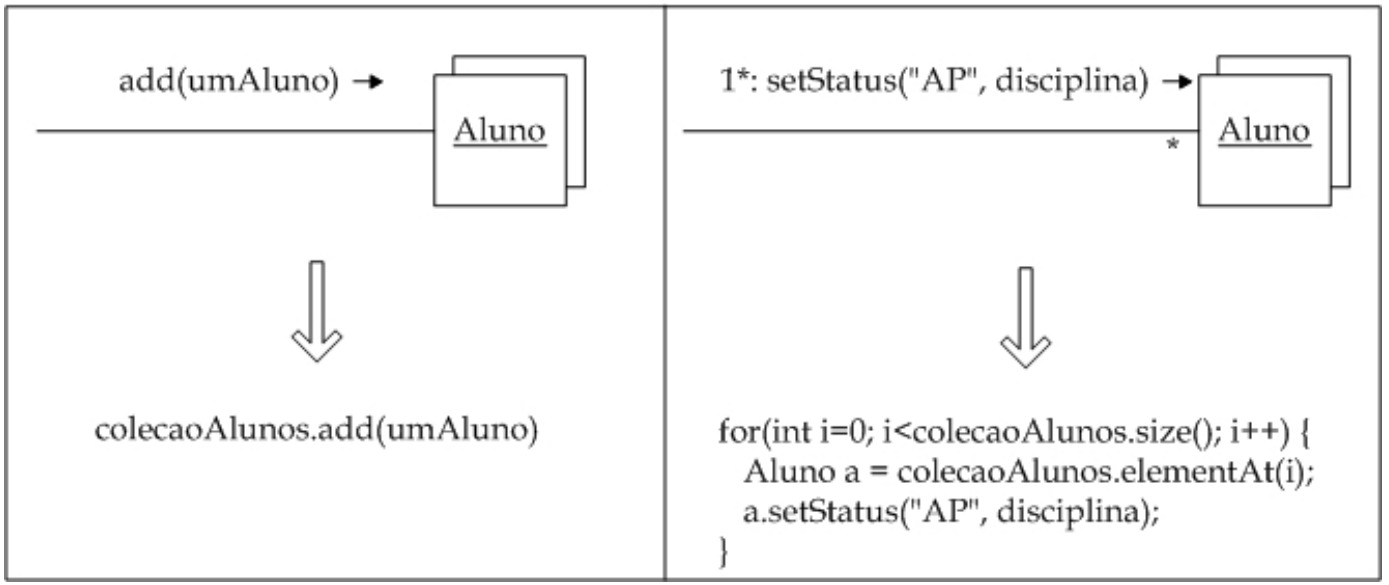
Notação para multiobjetos

- Uma multiobjeto é representado graficamente na UML através de dois retângulos superpostos.



Mensagens para Objetos/Coleção

- Uma mensagem pode ser enviada para um multiobjeto, ou pode ser enviada para um único objeto (elemento) do multiobjeto.
- Quando o símbolo de iteração não é usado, convenciona-se que a mensagem está sendo enviada para o próprio multiobjeto.
- Exemplo:



Implementação de multiobjetos

- Multiobjetos são normalmente implementados através de alguma estrutura de dados que manipule uma coleções.
- Portanto, algumas mensagens típicas que podemos esperar que um multiobjeto aceite são as seguintes:
  - Posicionar o cursor da coleção no primeiro elemento.
  - Retornar o i-ésimo objeto da coleção.
  - Retornar o próximo objeto da coleção.
  - Encontrar um objeto de acordo com um identificador único.
  - Adicionar um objeto na coleção.
  - Remover um objeto na coleção.
  - Obter a quantidade de objetos na coleção.
  - Retornar um valor lógico que indica se há mais objetos a serem considerados.

Implementação de multiobjetos (cont)

- A interface List da linguagem Java apresenta operações típicas de um multiobjeto.

```
public interface List<E> extends Collection<E> {
    E get(int index);
    E set(int index, E element);
    boolean add(E element);
    void add(int index, E element);
    E remove(int index);
    abstract boolean addAll(int index, Collection<? extends E> c);
    int indexOf(Object o);
    int lastIndexOf(Object o);
    ListIterator<E> listIterator();
    ListIterator<E> listIterator(int index);
    List<E> subList(int from, int to);
}
```

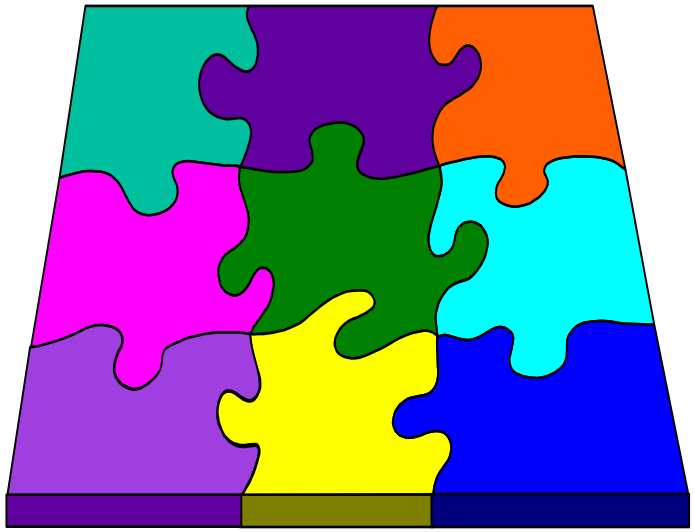
Tipos de diagrama de interação

- Há três tipos de diagrama de interação na UML 2.0: **diagrama de seqüência, diagrama de comunicação** e **diagrama de visão geral da interação**.
  - O diagrama de seqüência e o diagrama de comunicação são equivalentes.

Diagrama de seqüência: foco nas mensagens enviadas no decorrer do tempo.

Diagrama de comunicação: foco nas mensagens enviadas entre objetos que estão relacionados.

Diagrama de visão geral de interação. Pode ser utilizado para apresentar uma visão geral de diversas interações entre objetos, cada uma delas representada por um diagrama de interação. Diagrama é útil para ***modularizar*** a construção do diagramas de seqüência (ou de comunicação).



7.2 Diagrama de seqüência

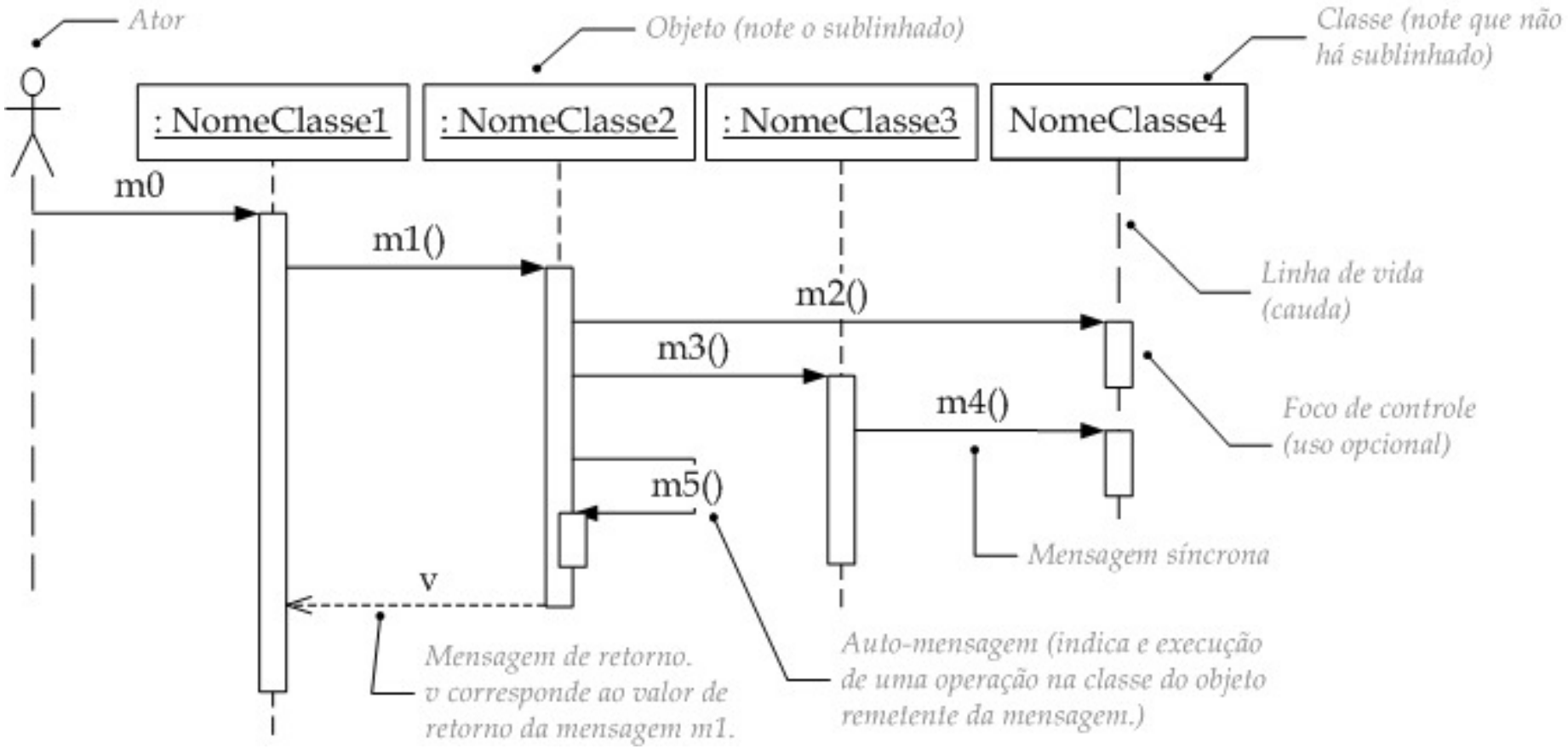
Diagrama de seqüência

- Os objetos participantes da interação são organizados na horizontal.
- Abaixo de cada objeto existe uma linha (linha de vida)
- Cada linha de vida possui o seu foco de controle.
  - Quando o objeto está fazendo algo.
- As mensagens entre objetos são representadas com linhas horizontais rotuladas partindo da linha de vida do objeto remetente e chegando a linha de vida do objeto receptor.
- A posição vertical das mensagens permite deduzir a ordem na qual elas são enviadas.
- Ordem de envio de mensagens em um diagrama de seqüência pode ser deduzida a partir das expressões de seqüência.
- Criação e destruição de objetos podem ser representadas.

Elementos gráficos de um DS

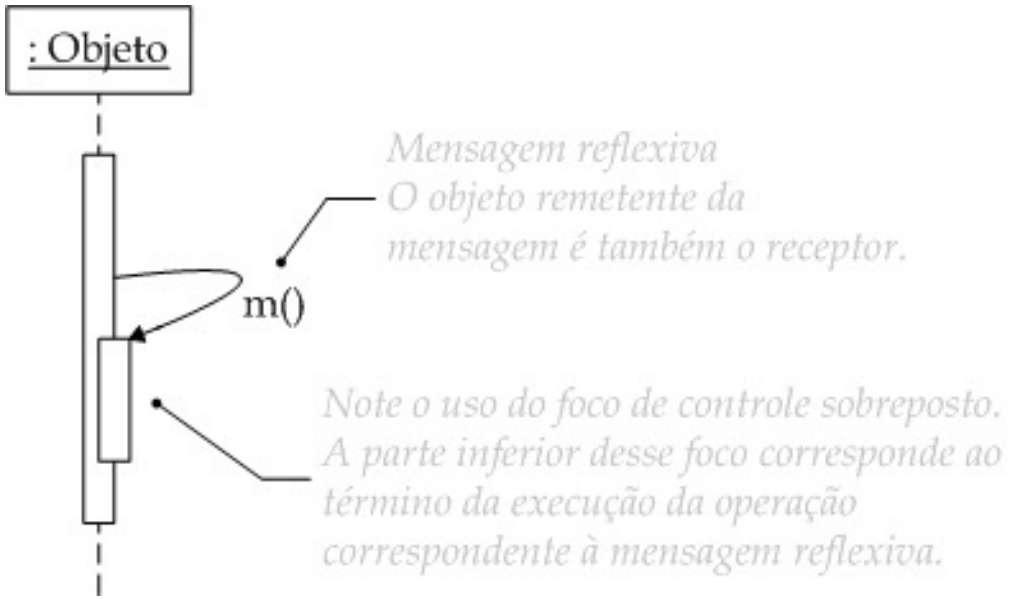
- Elementos básicos em um diagrama de seqüência:
  - Atores
  - Objetos, multiobjetos e classes
  - Mensagens
  - Linhas de vida e focos de controle
  - Criação e destruição de objetos
  - Iterações

Elementos gráficos de um DS

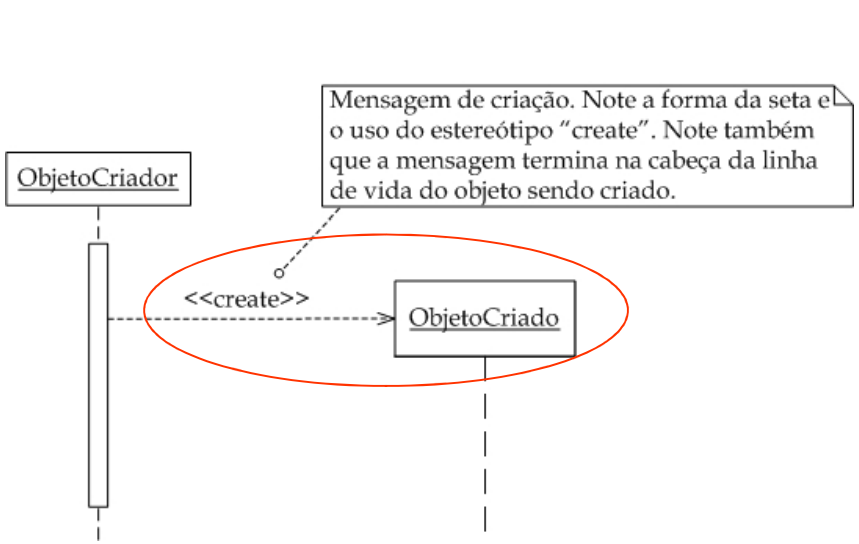


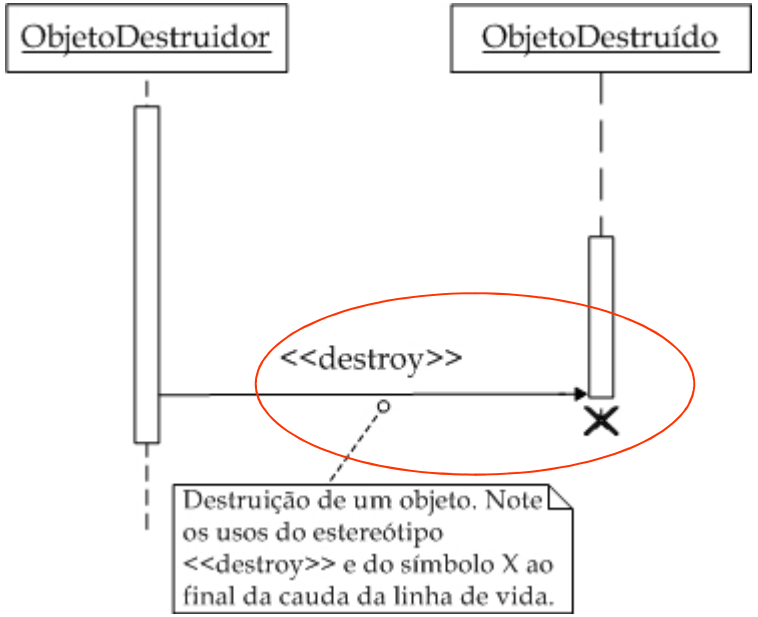
Mensagens reflexivas em um DS

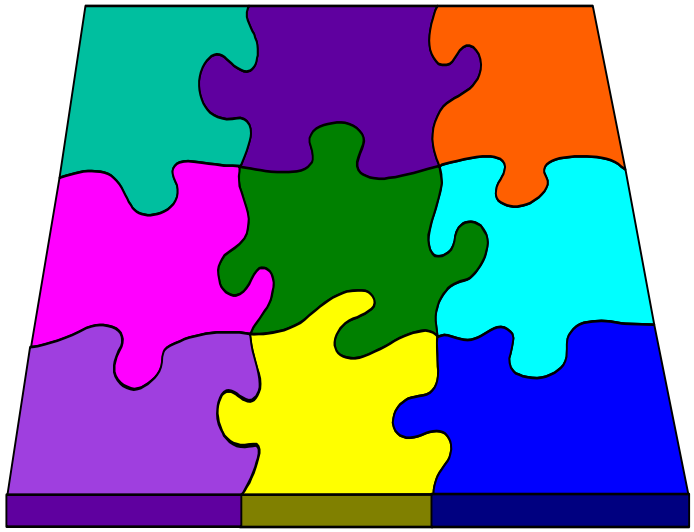
- Em uma mensagem reflexiva (ou auto-mensagem) o remetente é também o receptor.
  - Corresponde a uma mensagem para this (self).
  - O que isso significa na prática?



Criação/destruição de objetos em um DS







7.3 Diagrama de comunicação

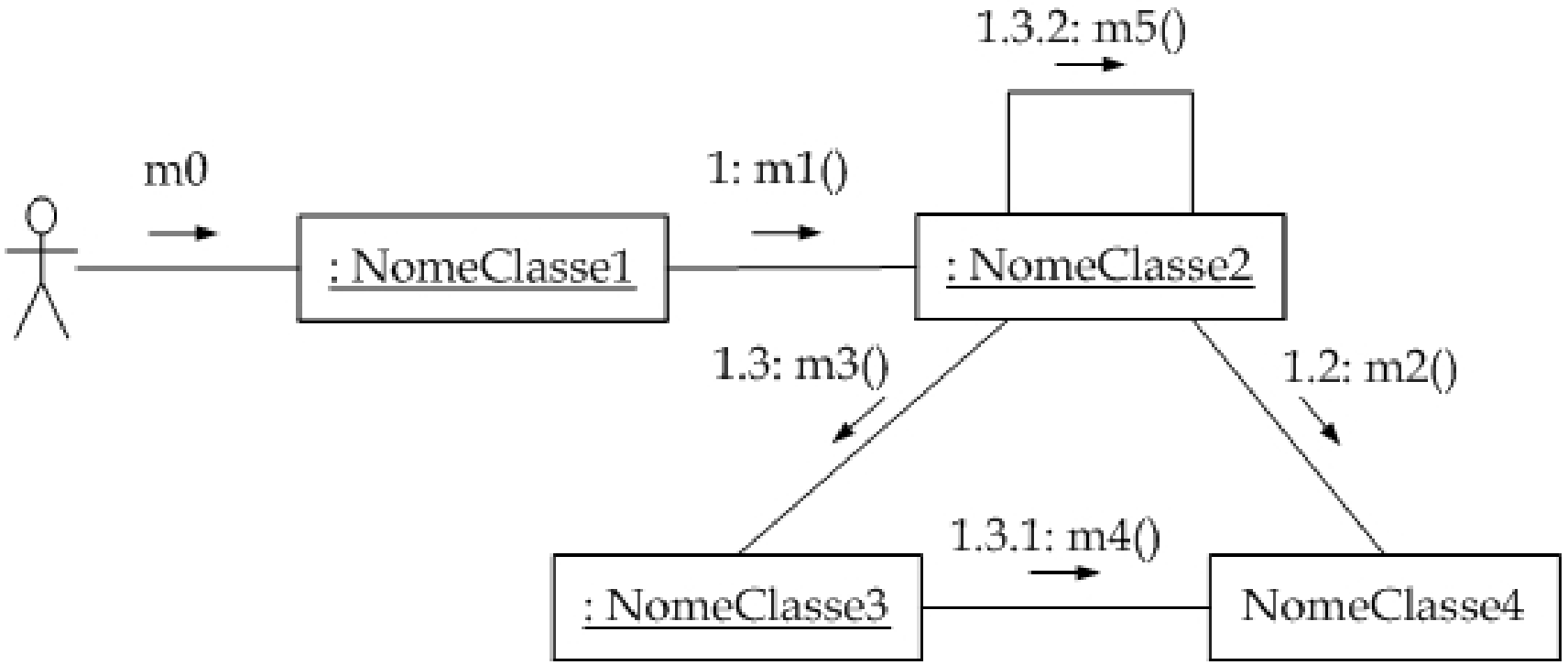
Diagrama de comunicação

- Chamado de **diagrama de colaboração** na UML 1.X.
- Estruturalmente, é bastante semelhante a um diagrama de objetos.
  - A diferença é que são adicionados setas e rótulos de mensagens nas ligações entre esses objetos.
- As ligações (linhas) entre objetos correspondem a relacionamentos existentes entre os objetos.
  - Deve haver consistência com o diagrama de classes...
- Os objetos estão distribuídos em duas dimensões
  - Vantagem: normalmente permite construir desenhos mais legíveis comparativamente aos diagramas de seqüência.
  - Desvantagem: não há como saber a ordem de envio das mensagens a não ser pelas expressões de seqüência.
- Direção de envio de mensagem é indicada por uma seta próxima ao rótulo da mensagem.

Elementos gráficos de um DC

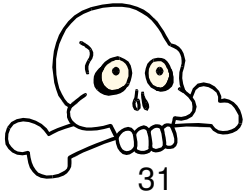
- Elementos básicos em um diagrama de comunicação:
  - Atores
  - Objetos, multiobjetos e classes
  - Mensagens
  - Ligações entre objetos
  - Criação e destruição de objetos
  - Iterações

Elementos gráficos de um DC

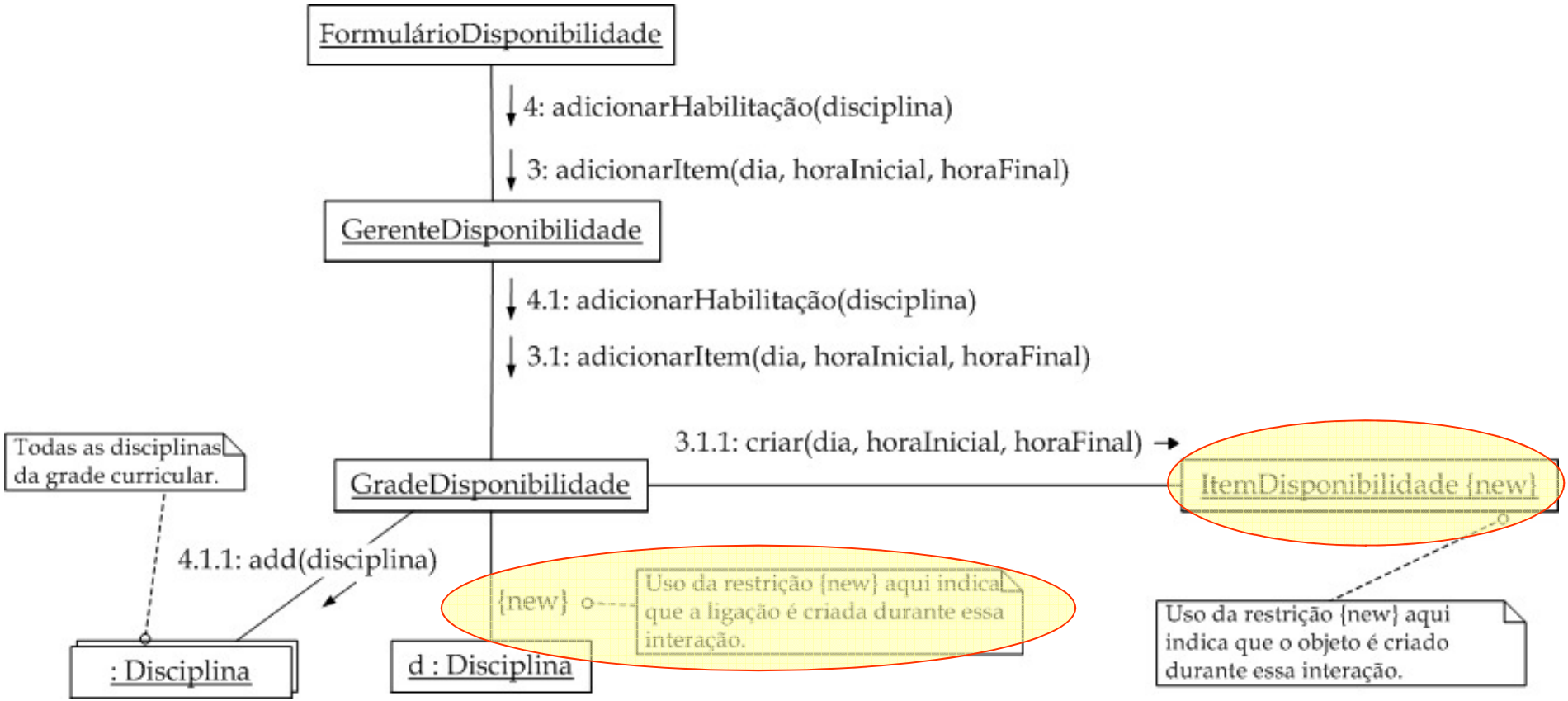


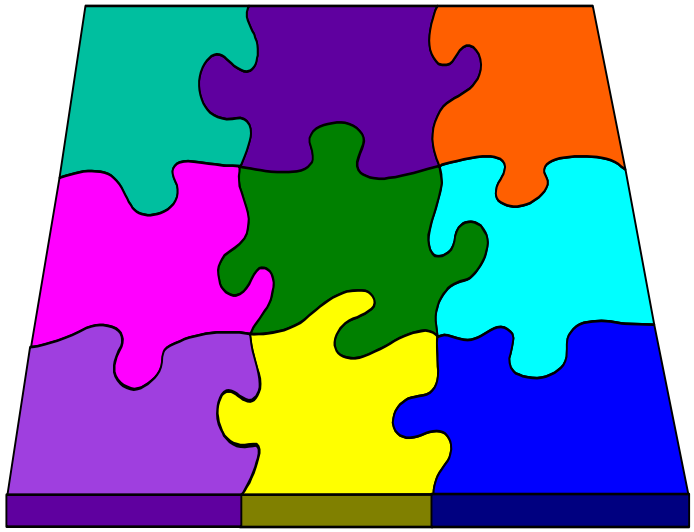
Criação de objetos em um DC

- Durante a execução de um cenário de caso de uso, objetos podem ser criados e outros objetos podem ser destruídos.
- Alguns objetos podem sobreviver à execução do caso de uso (se conectando a outro objetos); outros podem nascer e morrer durante essa execução.
- A UML define etiquetas (tags) para criação e destruição de objetos (ou de ligações entre objetos) no diagrama de comunicação.
  - **{new}**: objetos ou ligações criados durante a interação.
  - **{destroyed}**: objetos ou ligações destruídos durante a interação.
  - **{transient}**: objetos ou ligações destruídos e criados durante a interação.



Criação de objetos em um DC

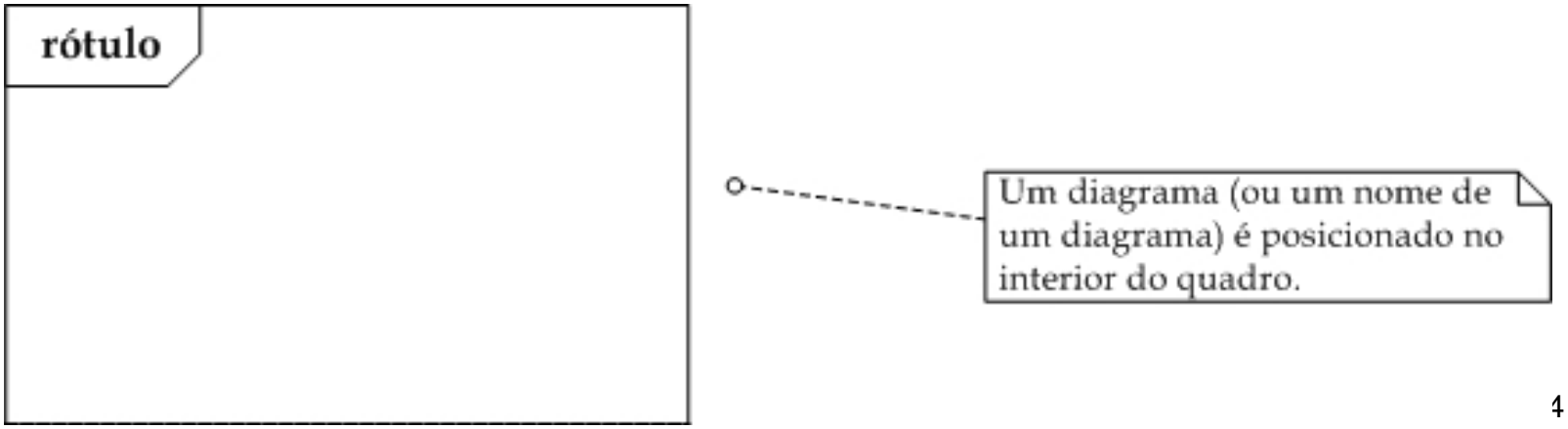




7.4 Modularização de interações

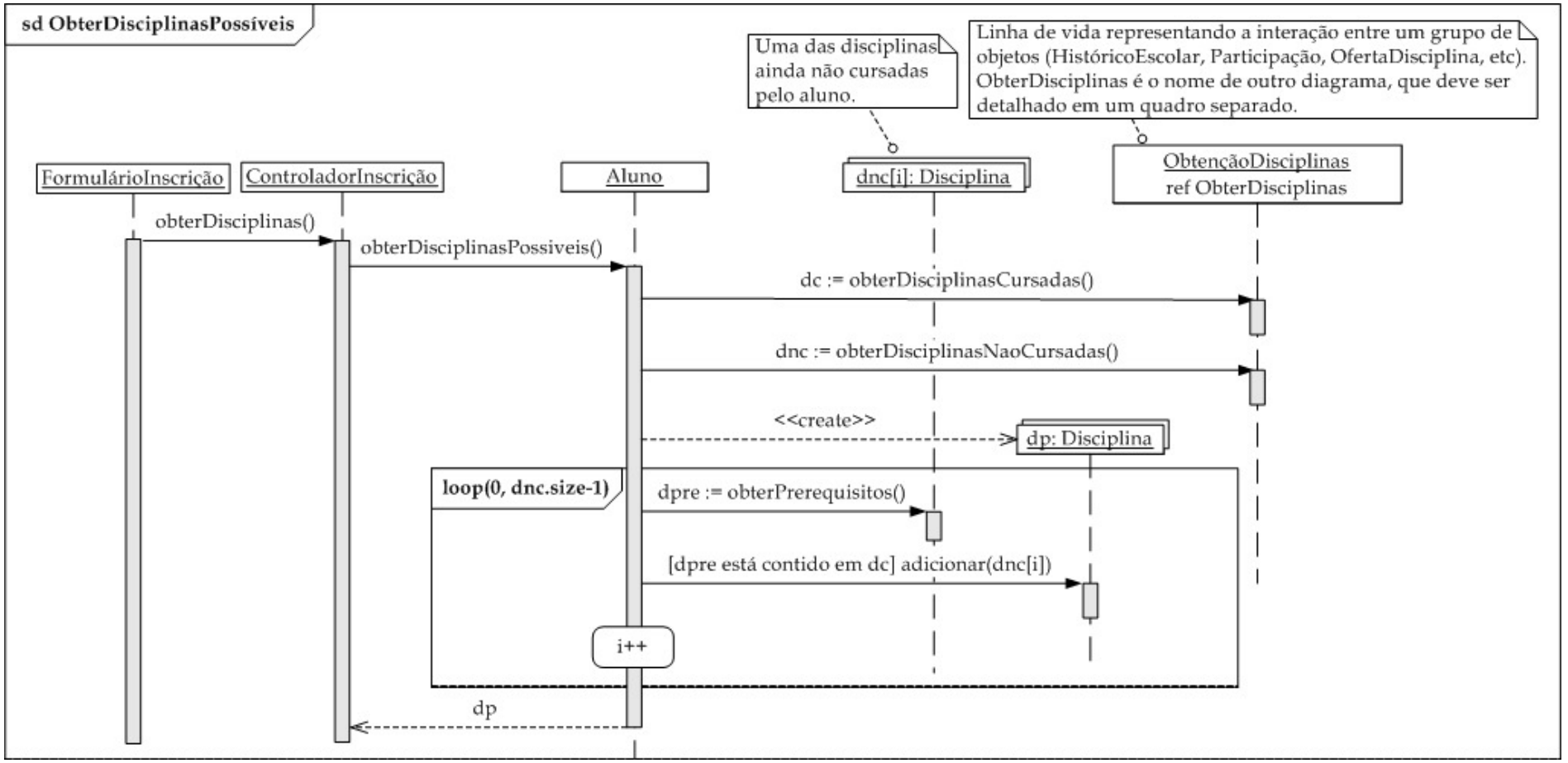
Quadros de interação

- Elemento gráfico, que serve para modularizar a construção de diagramas de seqüência (ou de comunicação).
- Objetivos específicos:
  - Dar um nome ao diagrama que aparece dentro do quadro;
  - Fazer referência a um diagrama definido separadamente;
  - Definir o fluxo de controle da interação.
- Notação:



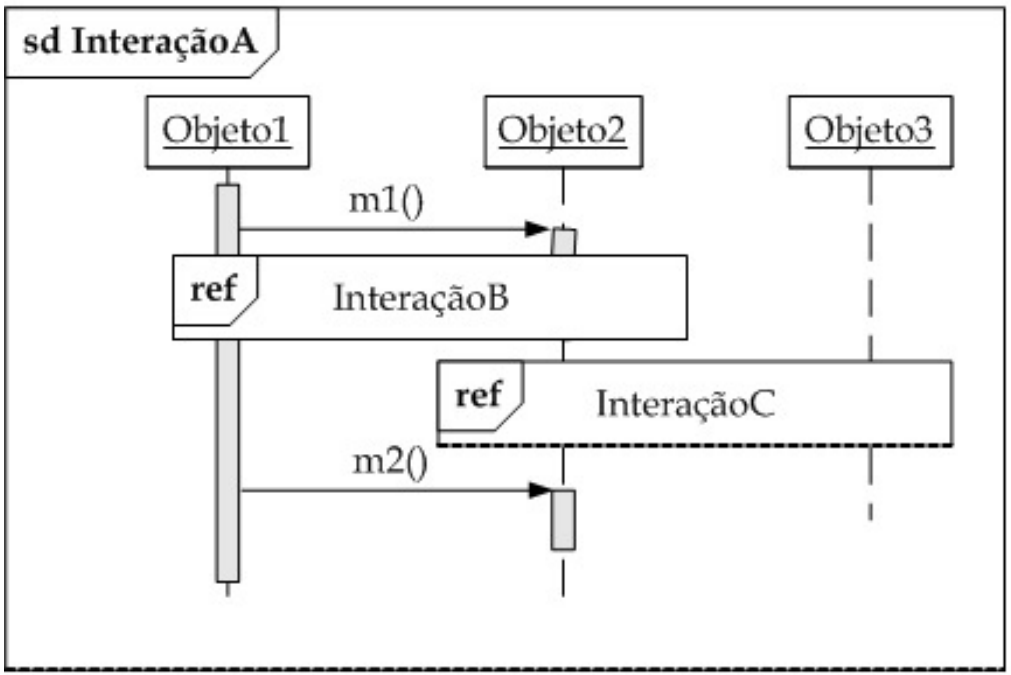
Diagramas nomeados

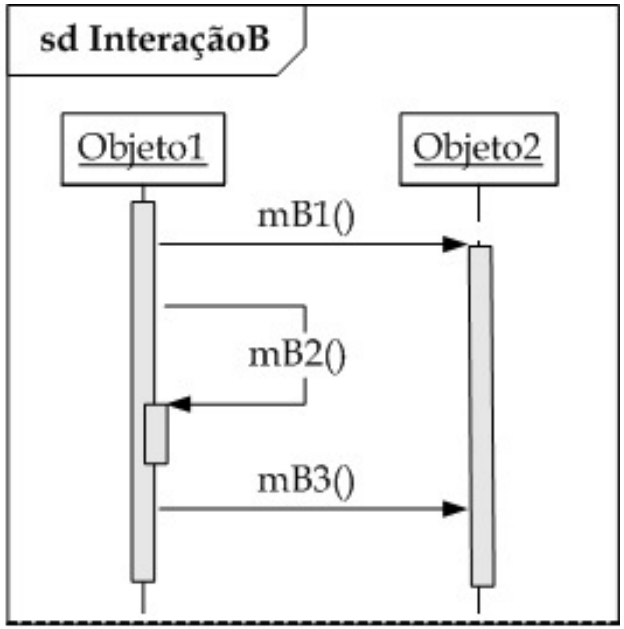
Dar um nome ao diagrama que aparece dentro do quadro



Diagramas referenciados

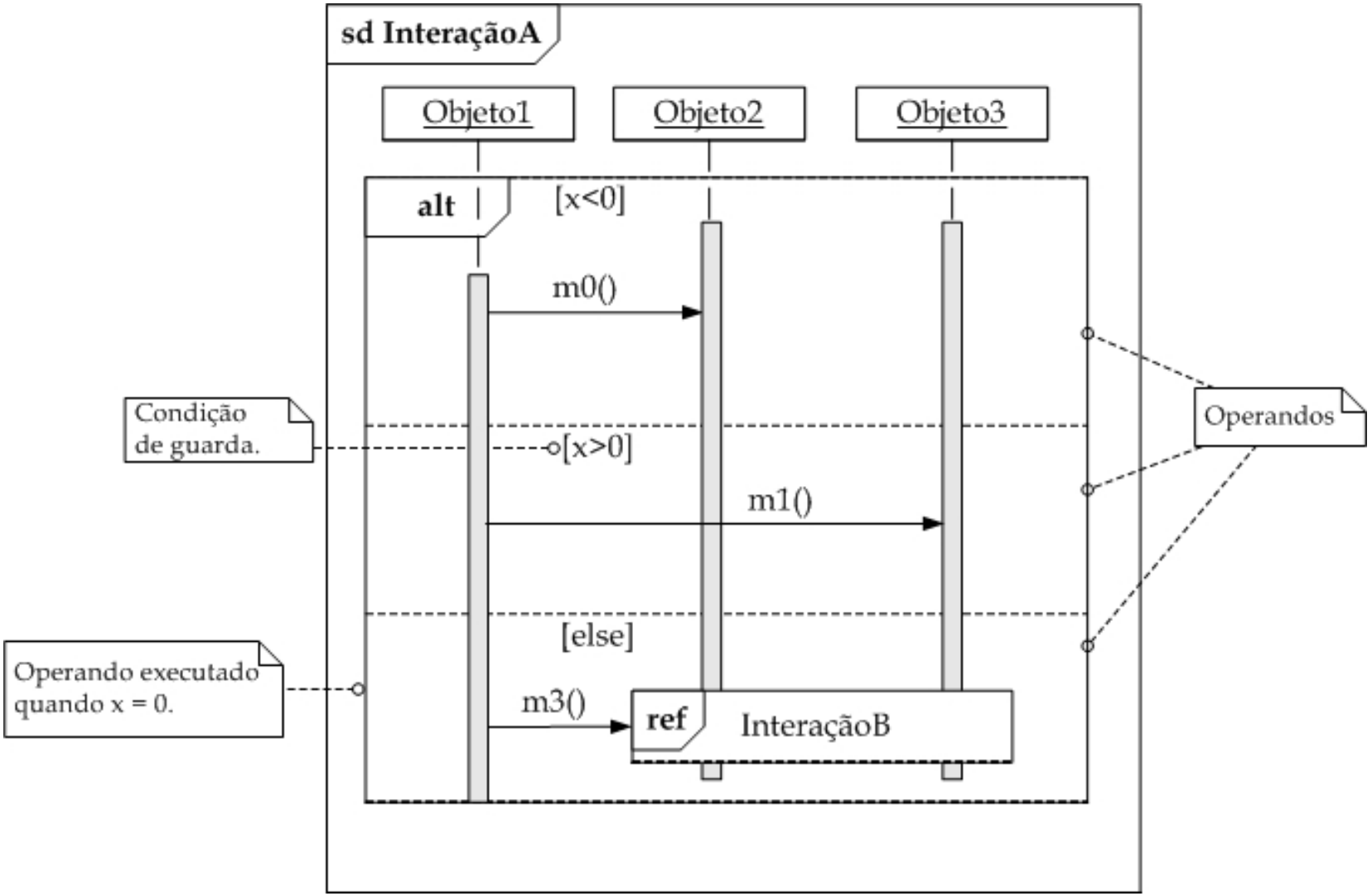
Fazer referência a um diagrama definido separadamente.



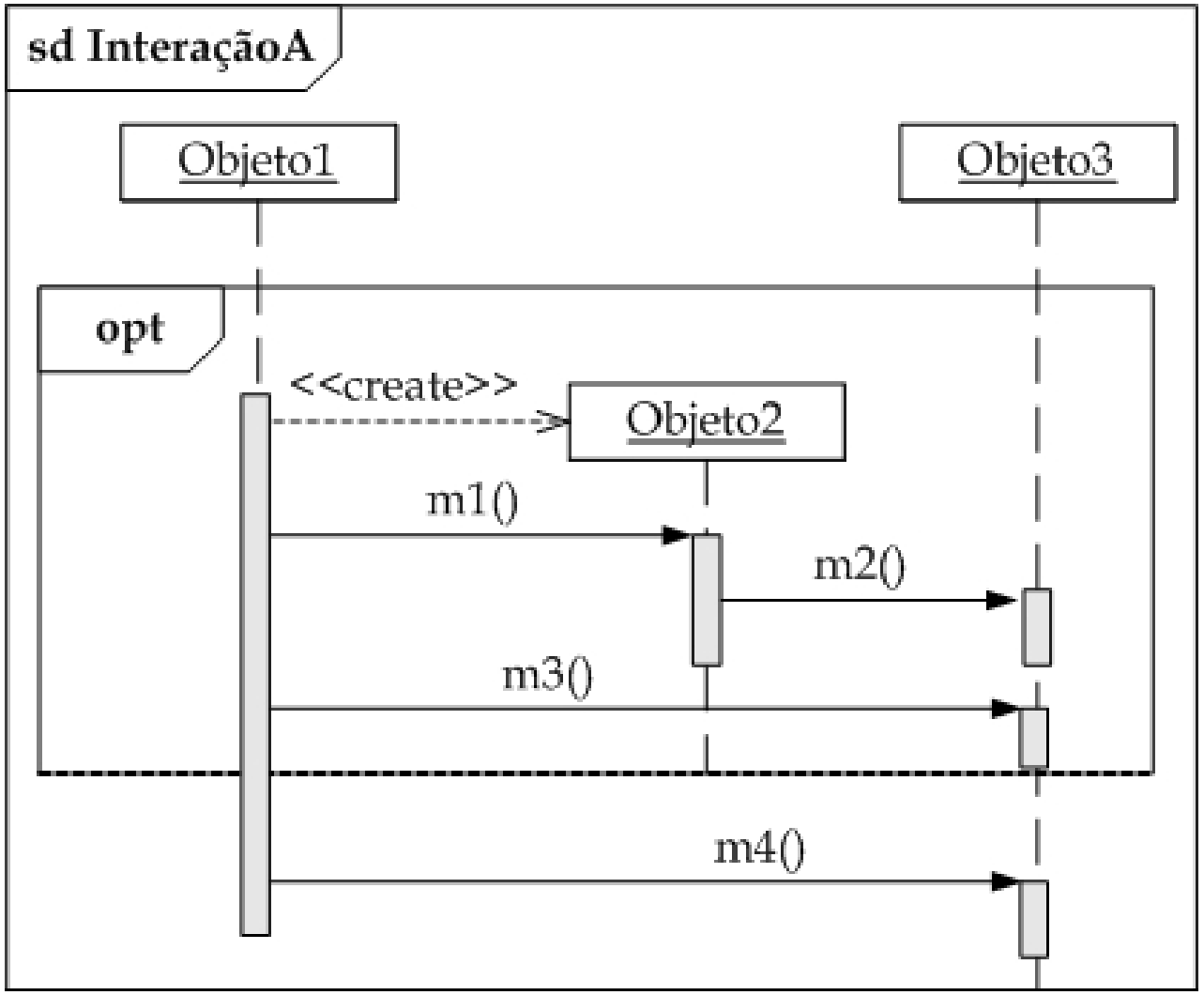


InteraçãoB e InteraçãoC são nomes de diagramas que apresentam mensagens trocadas entre os objetos Objeto1 e Objeto2. Note que os quadros correspondentes são rotulados com "ref" eposicionados sobre as linhas de vida dos objetos.

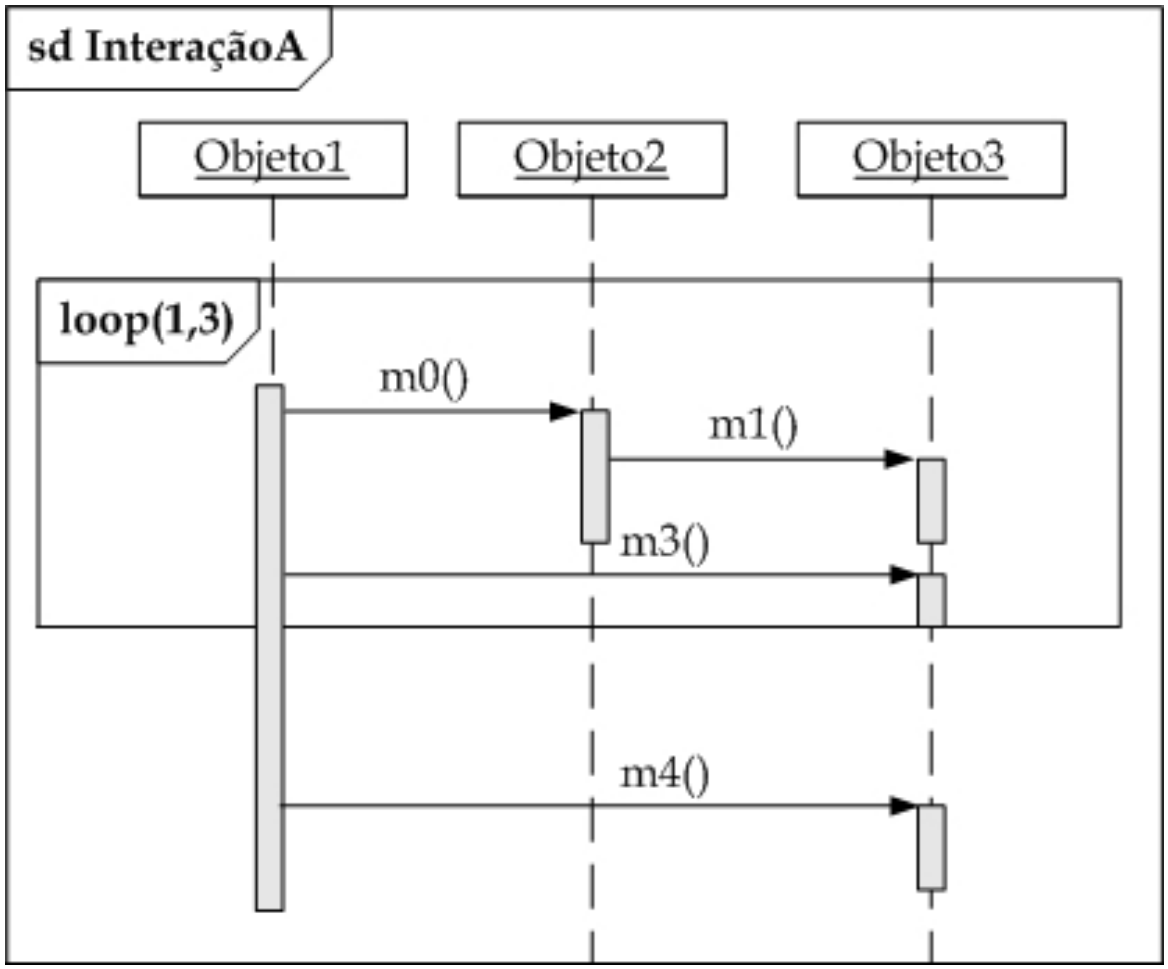
Fluxo de controle: alternativas

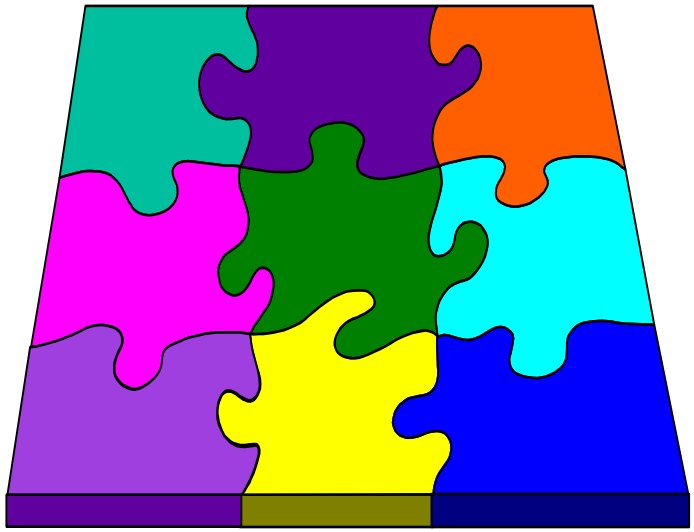


Fluxo de controle: opções



Fluxo de controle: iterações

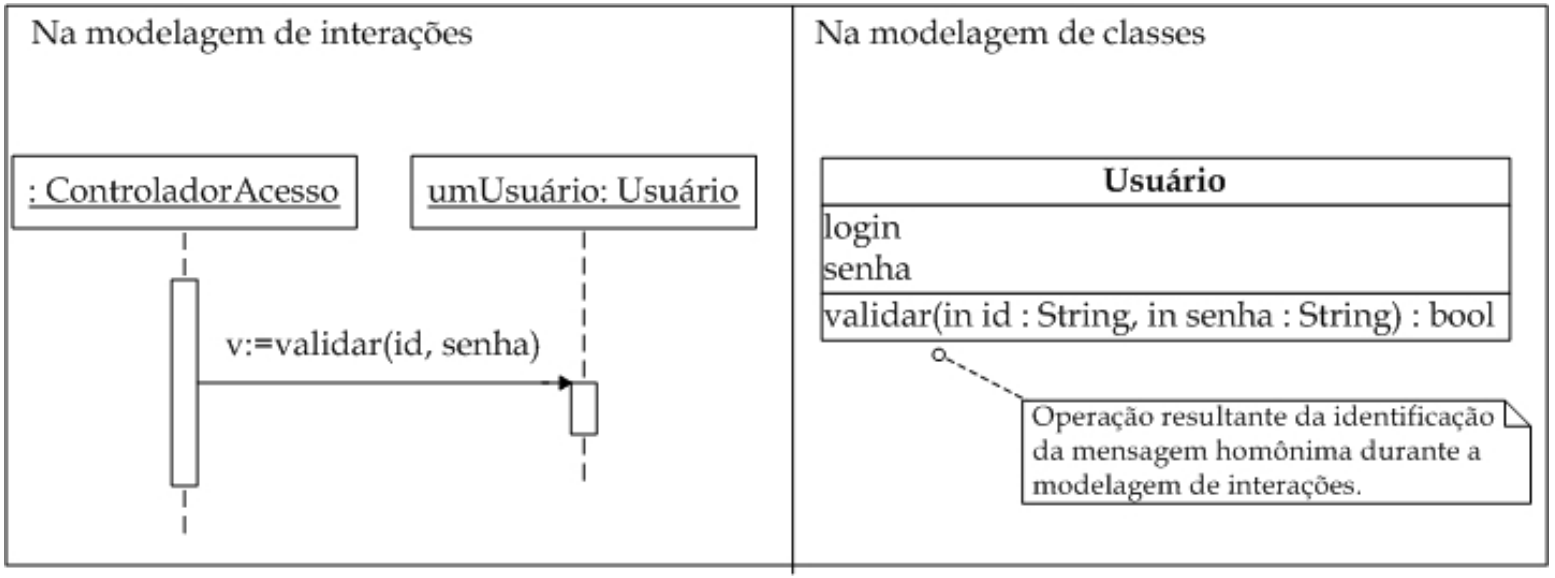




7.5 Construção do modelo de interações

Mensagens *versus* responsabilidades

- O objetivo da modelagem de interações é identificar **mensagens** e, em última análise, **responsabilidades**.



Uma mensagem implica na existência de uma operação no objeto receptor. A resposta do objeto receptor ao recebimento de uma mensagem é a execução da operação correspondente.

Alocação de responsabilidades

- Podemos então entender a modelagem de interações como um processo cujo objetivo final é decompor as responsabilidades do sistema e alocá-las a classes.
- Dado um conjunto de N responsabilidades, uma possibilidade é criar uma única classe no sistema para assumir com todas as N responsabilidades.
- Outra possibilidade é criar N classes no sistema, a cada um delas sendo atribuída uma das N responsabilidades.
- Certamente, as duas alternativas anteriores são absurdas do ponto de vista prático. Mas, entre as muitas maneiras possíveis de alocar responsabilidades, como podemos saber quais delas são melhores que outras?

Acoplamento e coesão

- A resposta à pergunta anterior não é nenhuma receita de bolo.
- De fato, para construirmos uma bom modelo de interações, devemos lançar mão de diversos princípios de projeto:
- Dois dos principais princípios são o acoplamento e a coesão.
- A ***coesão*** é uma medida do quão fortemente relacionadas e focalizadas são as responsabilidades de uma classe.
- É extremamente importante assegurar que as responsabilidades atribuídas a cada classe sejam altamente relacionadas.
  - Em outras palavras, o projetista deve definir classes de tal forma que cada uma delas tenha alta coesão.

Acoplamento e coesão

- O ***acoplamento*** é uma medida de quão fortemente uma classe está conectada a outras classes, tem conhecimento ou depende das mesmas.
- Uma classe com acoplamento fraco (baixo) não depende de muitas outras.
  - Por outro lado, uma classe com acoplamento forte é menos inteligível isoladamente e menos reutilizável.
- Além disso, uma classe com alto acoplamento é mais sensível a mudanças, quando é necessário modificar as classes da qual ela depende.
- Conclusão: criar modelos com ***alta coesão*** e ***baixo acoplamento*** deve ser um objetivo de qualquer projetista.

Dicas para construção do MI

- ***Identifique as <u>classes conceituais</u> que participam em cada caso de uso.***
  - Estas são as entidades do mundo real que estariam envolvidas na tarefa do caso do uso se este fosse executada manualmente.
    - Exemplos são: Aluno, OfertaDisciplina, Venda, Pagamento, etc.
  - Note que classes de fronteira também podem ser classes conceituais.
    - Por exemplo, FormulárioInscrição é um objeto de fronteira (para o caso de uso Realizar Inscrição) que também corresponde a um conceito existente no domínio do problema.

Dicas para construção do MI (cont)

- ***Identifique quaisquer classes de software que ajudem a organizar as tarefas a serem executadas.***
  - classes que não têm correspondente no mundo real
  - Essas classes normalmente são necessárias para manter a coesão das demais classes em um nível alto.
  - Segundo Craig Larman, essas classes são fabricações puras (*pure fabrications*).
  - Aqui, se encaixam algumas classes de fronteira, classes de controle.
  - Também: classes de acesso ao mecanismo de armazenamento, classes de autenticação, etc.

Dicas para construção do MI

- ***Defina também que objetos criam (destróem) outros objetos.***
  - Na realização de um caso de uso, objetos de entidade podem ser criados pelo objeto de controle, que recebe os dados necessários à instanciação a partir de objetos de fronteira.
  - Objetos de entidade também podem ser criados (destruídos) por outros objetos de entidade.
  - De uma forma geral, em uma agregação (ou composição), o objeto todo tem prioridade para criar (destruir) suas partes.
  - Portanto, em uma agregação (ou composição) entre objetos de entidade, é mais adequado que o objeto todo crie (destrua) suas partes quando requisitado por outros objetos.

Dicas para construção do MI (cont)

- ***Verifique a consistência dos diagramas de interação em relação ao MCU e ao modelo de classes.***
  - Verifique que cada cenário relevante para cada caso de uso foi considerado na modelagem de interações.
  - Se assegure de que as mensagens que um objeto recebe estão consistentes com as responsabilidades a ele atribuídas.
  - Alguns dos objetos necessários em uma interação já podem ter sido identificados durante a construção do modelo de classes de análise.
  - Durante a construção do diagrama de interação, o modelador pode identificar novas classes.
  - Atributos, associações e operações também surgem como subproduto da construção dos diagramas de interação.

Dicas para construção do MI (cont)

- ***Se certifique de que o objeto de controle realiza apenas a coordenação da realização do caso de uso.***
  - Como o controlador tem a responsabilidade de coordenação, todas as ações do ator resultam em alguma atividade realizada por esse objeto de controle.
  - Isso pode levar ao alto acoplamento; no pior caso, o controlador tem conhecimento de todas as classes participantes do caso de uso.
  - Responsabilidades específicas no domínio devem ser atribuídas aos objetos de domínio (entidades).
  - Sempre que for adequado, segundo os princípios de coesão e de acoplamento, devemos fazer com que as classes de domínio enviem mensagens entre si, aliviando o objeto de controle.

Dicas para construção do MI (cont)

- ***Faça o máximo para construir diagramas de interação o mais inteligíveis possível.***
  - Por exemplo, podemos utilizar notas explicativas para esclarecer algumas partes do diagrama de interação que esteja construindo.
    - Essas notas podem conter pseudocódigo ou mesmo texto livre.
  - Outra estratégia que ajuda a construir um modelo de interações mais inteligível é utilizar os recursos de modularização que a UML 2.0 acrescentou.
    - quadros de intereção, referências entre diagramas, etc.

Dicas para construção do MI (cont)

- ***Utilize o princípio de projeto conhecido como Lei de Demeter.***
  - Esse princípio está associado ao princípio do acoplamento e impõe restrições acerca de quais são os objetos para os quais devem ser enviadas mensagens na implementação de uma operação:
    - (a) ao próprio objeto da classe (ou self);
    - (b) a um objeto recebido como parâmetro do método;
    - (c) a um atributo da classe;
    - (d) a um objeto criado dentro do método;
    - (e) a um elemento de uma coleção que é atributo da classe.
  - A intenção é evitar acoplar excessivamente um objeto e também evitar que ele tenha conhecimento das associações entre outros objetos.

- Na modelagem de interações, quando definimos uma mensagem, estamos criando uma dependência entre os objetos envolvidos.
- Isso é mesmo que dizermos que estamos aumentando o acoplamento entre os objetos em questão.
- Portanto, é necessário que o modelador fique atento para apenas definir mensagens que são realmente necessárias.
  - Sempre que possível, devemos evitar o envio de mensagens que implique na criação de associações redundantes no modelo de classes.
  - Isso porque a adição de uma associação entre duas classes aumenta o acoplamento entre as mesmas.

Procedimento de construção

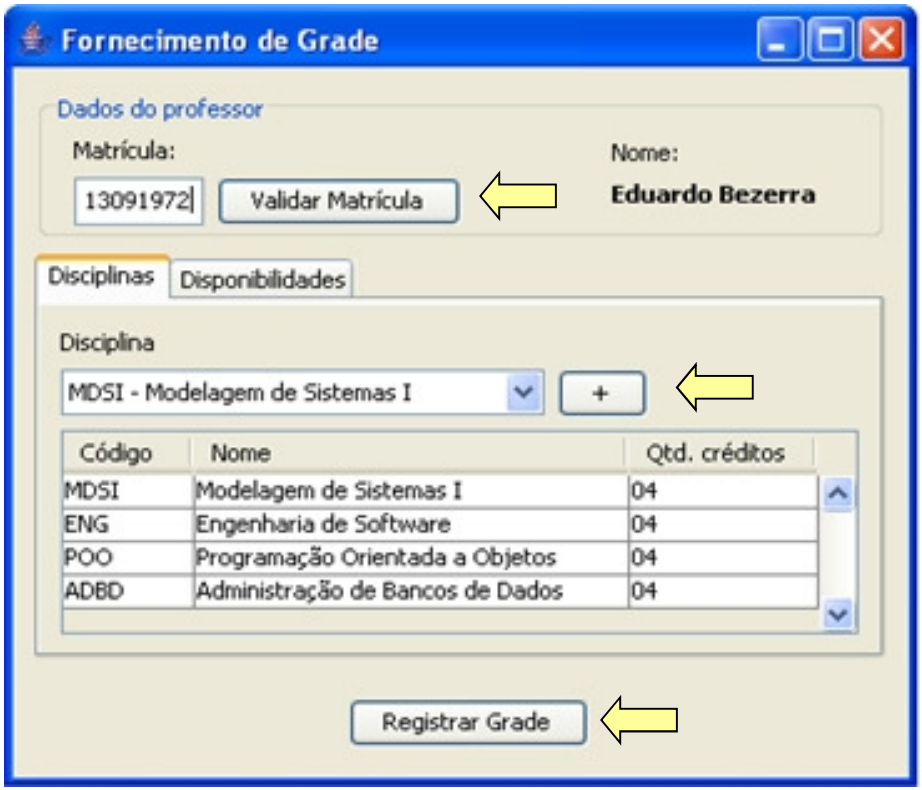
- Vamos agora descrever um procedimento para construção do modelo de interações.
- Esse procedimento genérico serve tanto para diagramas de seqüência quanto para diagramas de comunicação, resguardando-se as diferenças de notação entre os dois.
- Durante a aplicação desse procedimento, é recomendável considerar todas as dicas descritas anteriormente.
- Antes de descrevermos esse procedimento, é necessário que definamos o conceito de ***evento de sistema***…

Eventos de sistema

- Eventos de sistema correspondem às ações do ator no cenário de determinado caso de uso.
- Sendo assim, é relativamente fácil identificar eventos de sistemas em uma descrição de caso de uso: devemos procurar nessa descrição os eventos que correspondem a ***ações do ator***.
- *No caso particular em que o ator é um ser humano e existe uma interface gráfica para que o mesmo interaja com o sistema, os eventos do sistema são resultantes de ações desse ator sobre essa interface gráfica, que corresponde a objetos de fronteira.*

Eventos de sistema (cont)

- Considere o formulário a seguir, para o caso de uso (do SCA) denominado "Fornecer Grade de Disponibilidades“:



Eventos de sistema (cont)

- No formulário anterior, temos a seguinte lista de eventos de sistema:
  - solicitação de validação de matrícula de professor;
  - solicitação de adição de uma disciplina à grade;
  - solicitação de adição de um item de disponibilidade (dia, hora final e hora final) à grade;
  - solicitação de registro da grade.
- Importante: nem todo evento de sistema é originado em um objeto de fronteira correspondente a uma interface gráfica.
  - essa ocorrência pode ser gerada por um ator que não seja um ser humano (e.g., outro sistema ou um equipamento).

Eventos de sistema (cont)

- Mas, por que os eventos de sistema são importantes para a modelagem de interações?
- Por que as interações entre objetos de um sistema acontecem por conta do acontecimento daqueles.
  - Um evento de sistema é alguma ação tomada por um ator que resulta em uma seqüência de *mensagens* trocadas entre os objetos do sistema.
  - Portanto, o ponto de partida para a modelagem de interações é a identificação dos eventos do sistema.
  - Uma vez feita essa identificação, podemos desenhar diagramas de interação que modelam como os objetos colaboram entre si para produzir a resposta desejada a cada evento do sistema.

Procedimento de construção

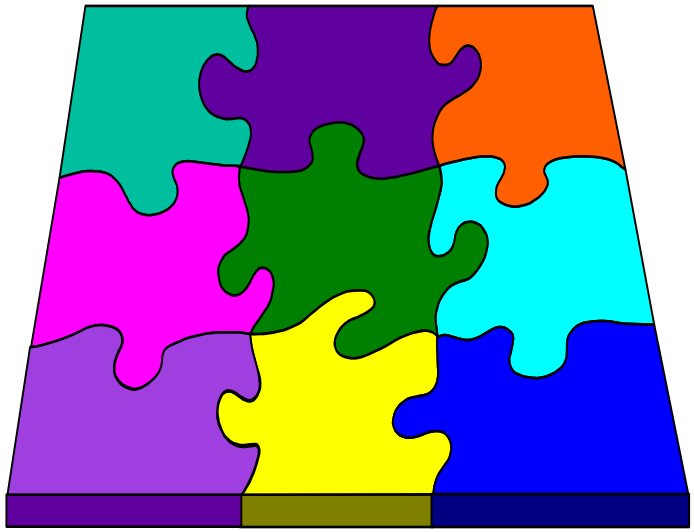
- Para cada caso de uso, selecione um conjunto de cenários relevantes.
  - O cenário correspondente ao fluxo principal do caso de uso deve ser incluído.
  - Considere também fluxos alternativos e de exceção que tenham potencial em demandar responsabilidades de uma ou mais classes.
- Para cada cenário selecionado, identifique os eventos de sistema:
  - Posicione o(s) ator(es), objeto de fronteira e objeto de controle no diagrama.
  - Para cada passo do cenário selecionado, defina as mensagens a serem enviadas de um objeto a outro.
  - Defina as cláusulas de condição e de iteração, se existirem, para as mensagens.
  - Adicione multiobjetos e objetos de entidade à medida que a sua participação se faça necessária no cenário selecionado.

Observações sobre o procedimento

- A definição das mensagens deve ser feita com base nas responsabilidades de cada objeto envolvido:
  - O nome da mensagem
  - Os argumentos de cada mensagem, se existirem.
  - O valor de retorno da operação correspondente, se existir.
  - Cláusulas de condição e de repetição, se existirem.
- A maioria dos objetos já devem ter sido identificados durante a construção do modelo de classes.
- Verificar as consistências:
  - Cada cenário relevante para cada caso de uso foi considerado?
  - A mensagens que um objeto recebe estão consistentes com suas responsabilidades?
- As mensagens de um ator a um objeto de fronteira normalmente são rotuladas com a informação fornecida
  - por exemplo, *item de pedido, id e senha*, etc.

Observações sobre o procedimento

- Mais de um controlador podem ser criados em um mesmo caso de uso, dependendo de sua complexidade.
  - O controlador pode mesmo ser suprimido, também em função da complexidade do caso de uso.
- Mensagens enviadas pelo objeto de fronteira por conta de um evento de sistema resultam na necessidade de definir ***operações de sistema*** no objeto controlador do caso de uso.
  - Por exemplo, no do formulário de fornecimento de disponibilidades, o controlador deve possuir as seguintes operações de sistema:
    - validarProfessor(matrícula);
    - adicionarDisciplina(nomeDisciplina);
    - adicionarItemDisponibilidade(dia, horaInicial, horaFinal).
    - registrarGrade()



7.6 Modelo de interações em um processo iterativo

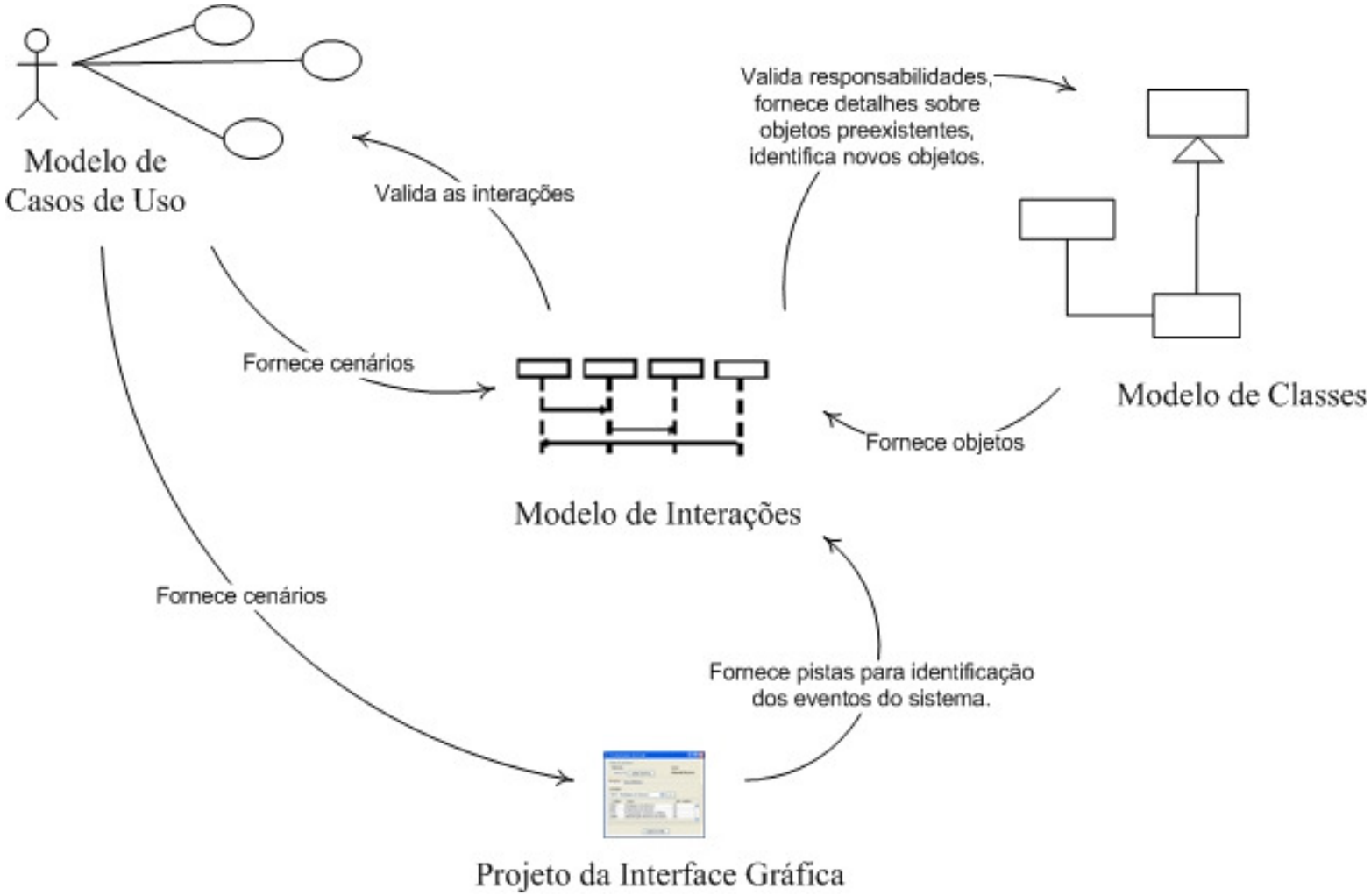
MI em um processo iterativo

- São utilizados na fase de construção de um ciclo de vida incremental e iterativo.
  - São construídos para os casos de uso alocados para uma iteração desta fase.
- Há controvérsias sobre o momento de início da utilização desse modelo (se na análise ou se no projeto).
- Inicialmente (+análise), pode exibir apenas os objetos participantes e mensagens exibindo somente o nome da operação (ou nome da responsabilidade).
- Posteriormente (+projeto), pode ser refinado.
  - criação e destruição de objetos, tipo e assinatura completa de cada mensagem, etc.

MI em um processo iterativo

- Embora modelos de um SSOO representem visões distintas, eles são *interdependentes e complementares*.
  - O MCU fornece cenários a serem considerados pelo MI.
  - O modelo de classes de análise fornece objetos iniciais para o MI.
  - A construção do MI fornece informações úteis para transformar o modelo de classes de análise no modelo de classes de especificação. Em particular, MI fornece os seguintes itens para refinar o modelo de classes de análise:
    - Detalhamento de operações
    - Detalhamento de associações
    - Operações para classes
    - Novos atributos para classes
    - Novas classes

MI em um processo iterativo



Discussão

- Como informações são passadas de um objeto a outro em um sistema OO?
- Quando utilizar diagramas de interações (seqüência ou comunicação)?
  - Há alternativas para esse momento?
- Qual é a conseqüência da construção dos DI's sobre os demais artefatos do sistema.
- Há possibilidade de geração de código a partir de um diagrama de interações?