Diagramas de Interação

Ricardo R. Gudwin
DCA-FEEC-UNICAMP
17/10/2010

Diagramas de Interação

Diagramas de interação mostram padrões de interação entre instâncias de objetos. Na verdade, de acordo com a norma UML, os diagramas de interação podem aparecer em duas formas diferentes, os diagramas de sequência e os diagramas de comunicação (também chamados de diagramas de colaboração, na versão 1 da norma UML). As informações em ambos os diagramas é equivalente, mas cada tipo de diagrama enfatiza um aspecto particular da interação. Os diagramas de sequência mostram a sequência explícita de estímulos entre objetos e são melhores para especificações de tempo real e para cenários complexos. Os diagramas de comunicação mostram o relacionamento entre as instâncias e são melhores para o entendimento de todos os efeitos sobre uma determinada instância, bem como para um design procedural. Uma regra prática para sabermos quando devemos utilizar diagramas de sequência ou diagramas de comunicação é a seguinte: se o número de objetos interagindo é grande, e o número de mensagens sendo trocado entre cada objeto é pequeno, devemos dar preferência aos diagramas de comunicação. Se, ao contrário, o número de objetos é pequeno, mas o número de mensagens sendo trocadas entre eles é grande, então o mais adequado é a utilização de um diagrama de sequência

Diagramas de Sequência

Um diagrama de sequência mostra uma interação na forma de uma sequência temporal de mensagens sendo enviadas entre instâncias. Em particular, mostra as instâncias participando de uma interação por meio de *lifelines* e o estímulo que trocam entre si arranjadados na forma de uma sequência temporal. Assim, um diagrama de sequência não explicita as associações entre objetos, embora possamos inferir que havendo uma mensagem de um objeto a outro no diagrama, é necessário que haja uma associação entre eles (pois caso contrário, o objeto não seria capaz de enviar a mensagem).

A idéia de ***lifeline*** é a idéia de representar temporalmente o ciclo de vida de um objeto executando um papel bem específico em uma interação. Assim, cada objeto participando da interação possui sua própria *lifeline*. Representa-se uma *lifeline* por meio de uma linha vertical saindo de baixo do objeto e varrendo todo o espaço do diagrama. Dessa forma, entende-se que o eixo vertical representa o passar do tempo. As setas entre diferentes *lifelines* denotam uma comunicação ocorrendo entre os objetos envolvidos em um instante do tempo. A existência e a duração do objeto em um papel pode ser mostrada, mas o relacionamento entre os objetos não é mostrado explicitamente como no caso dos diagramas de comunicação. Uma *lifeline* pode se dividir em duas ou mais *lifelines* concorrentes para expressar uma condicionalidade, ou seja, fluxos de sequência alternativos, dependentes de condições que são indicadas no diagrama. Um exemplo de diagrama de sequência pode ser visto na figura 1 a seguir:

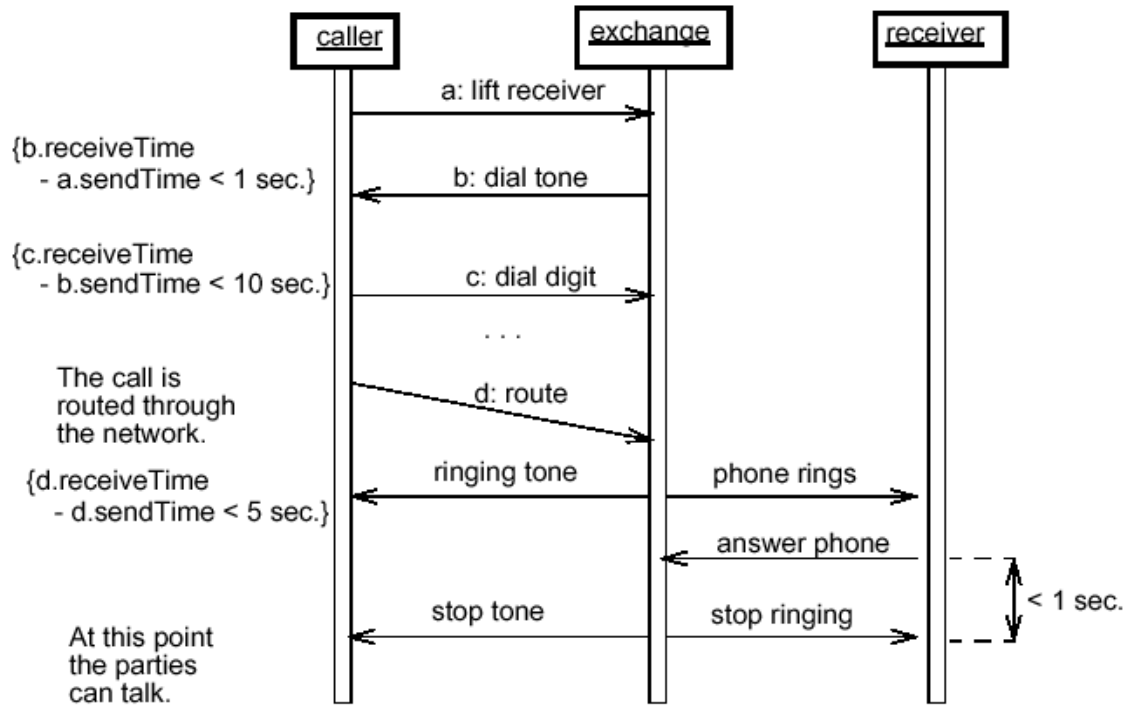

Figura 1: Exemplo de Diagrama de Sequência

Observe nesse exemplo que todos os objetos estão ativos (ou com o chamado foco de controle) durante todos os instantes de tempo. Esse fato é representado pelas linhas cheias percorrendo todo o *lifeline*. Esse tipo de representação pode também ser utilizado quando não se deseja representar explicitamente o momento em que um objeto está ativo, quando é criado ou destruído. Entretanto, todas essas informações podem ser representadas. Um exemplo mostrando a criação de objetos, o momento em que estão ativos e o momento em que são destruídos pode ser visto na figura 2.

Nesta figura, diversos recursos de representação dos diagramas de sequência são representados. Por exemplo, o **foco de controle** ou ativação, mostra o período no qual um objeto está executando uma ação. Durante o foco de controle, a lifeline apresenta uma linha cheia e dupla. Quando o objeto não está com o foco de controle, a lifeline se transforma em uma linha tracejada.

Observe também um exemplo de **mensagens condicionais** entre **ob1** e **ob3**. Dependendo do valor de x, o objeto **ob1** emite a mensagem **foo(x)** para **ob2** ou a mensagem **bar(x)** para **ob3**.

Um exemplo de **recursão** é mostrado também na figura. Uma recursão acontece quando um objeto manda uma mensagem para si próprio, como faz o objeto **ob1** com a mensagem **more()**.

No diagrama vemos também a representação da **criação de objetos**. Os objetos que existiam previamente encontram-se alinhados ao topo do diagrama. No caso os objetos **ob3** e **ob4** já existiam ao início do processo. Os objetos **ob1** e **ob2**, entretanto, foram criados durante a interação. Essa criação é representada com o deslocamento do objeto para o instante de tempo em que este foi criado. Assim, o objeto **ob1** foi criado pelo usuário por meio da mensagem **op()** e o objeto **ob2** foi criado por meio da mensagem **foo(x)**.

Do mesmo modo, representamos a **destruição de um objeto** por meio da marca X em algum ponto da lifeline do objeto. Assim, os objetos **ob1** e **ob2** são destruídos nos pontos marcados por X no diagrama.

Devemos ainda observar que diferentes tipos de setas ocorrem no diagrama. Na verdade, o UML

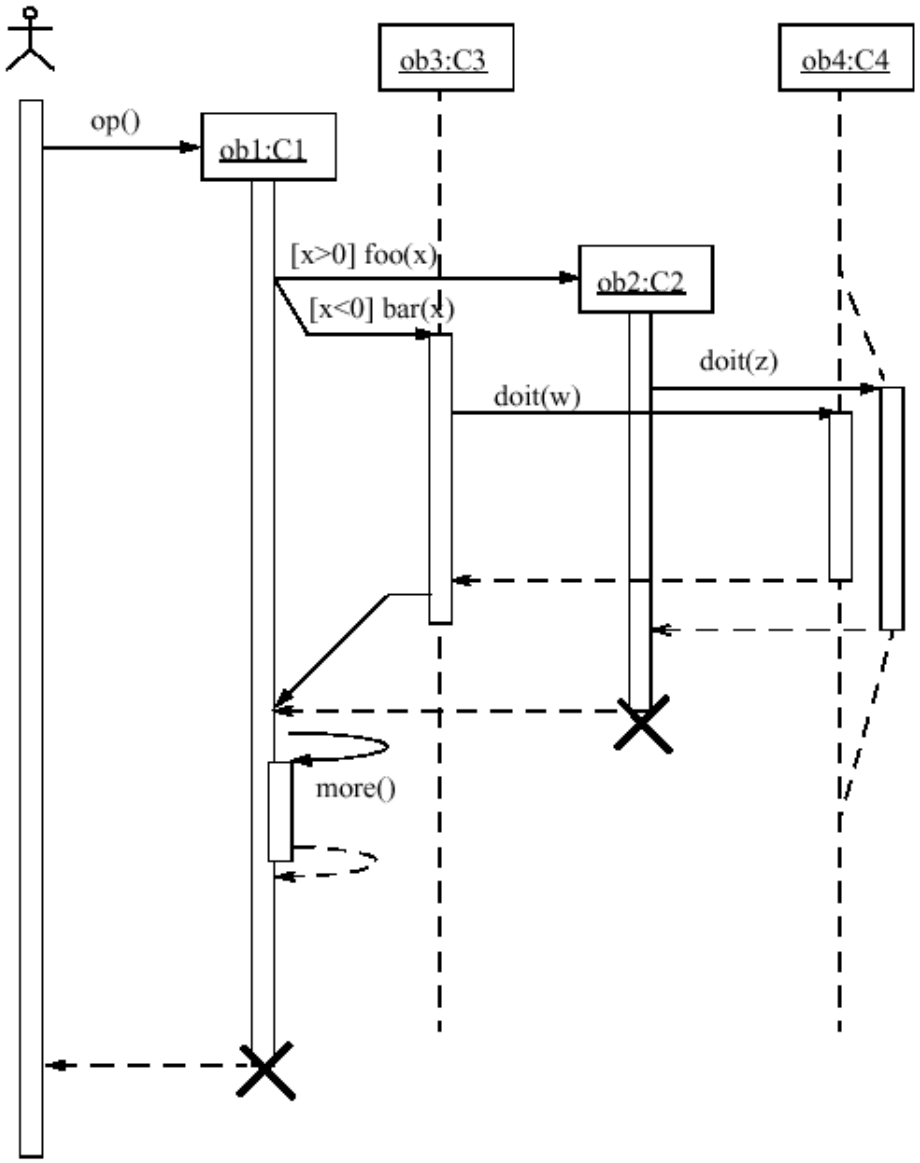

Figura 2: Exemplo de Diagrama de Sequência com Representação da Ativação dos Objetos

estabelece diferentes signficados para diferentes tipos de setas. Cada tipo de seta representará um diferente tipo de comunicação.

As setas do tipo → são utilizadas para representar **chamadas de procedimento ou outro tipo de fluxo de controle**. Toda uma sequência aninhada é completada antes que a sequência mais externa termine. Este tipo de seta pode ser usada em chamadas de procedimento ordinárias, mas pode também ser usada em objetos concorrentes quando um deles manda um sinal e espera uma sequência de comportamentos ser completada.

As setas do tipo → são utilizadas para representar um **fluxo de controle fraco**. Assim, cada seta desse tipo mostra a progressão do próximo passo na sequência. Normalmente as mensagens desse tipo são assíncronas.

As setas do tipo → são utilizadas para representar **estímulos assíncronos**. Este tipo de seta é utilizado no lugar do anterior quando se quer mostrar explicitamente uma comunicação assíncrona entre dois objetos.

Por fim, as setas do tipo - -> (tracejadas) são utilizadas para representar um **retorno de uma chamada de procedimento**. Assim, são usadas em conjunto com as setas do primeiro tipo, ao final de cada procedimento. Setas desse tipo são utilizadas no diagrama da figura 2.

Diagramas de Comunicação

Um diagrama de comunicação mostra a interação entre objetos organizada de acordo com os papéis de cada objeto na interação, e sua ligação entre si. Ao contrário de um diagrama de sequência, mostra explicitamente o relacionamento entre objetos executando os diversos papéis. Da mesma maneira, não mostra o tempo como uma dimensão separada, ou seja, de forma explícita como nos diagramas de comunicação. Entretanto, a sequência de mensagens é mostrada na forma de números atribuídos a cada mensagem, que indicam a ordem em que as mensagens são enviadas. Assim, a informação temporal, apesar de não estar colocada de forma explícita, também está presente nos diagramas de comunicação.

Os diagramas de comunicação podem ser desenvolvidos em duas formas diferentes: ao **nível de especificação** (papéis de classifier, papéis de associação e mensagens) ou ao **nível de instância** (objetos, links e estímulos). No primeiro caso, os diagramas modelam papéis e a estrutura de colaboração entre esses papéis, sendo que no segundo caso, mostra-se os objetos que assumem esses papéis.

Um exemplo de diagrama de comunicação é apresentado na figura 3 a seguir:

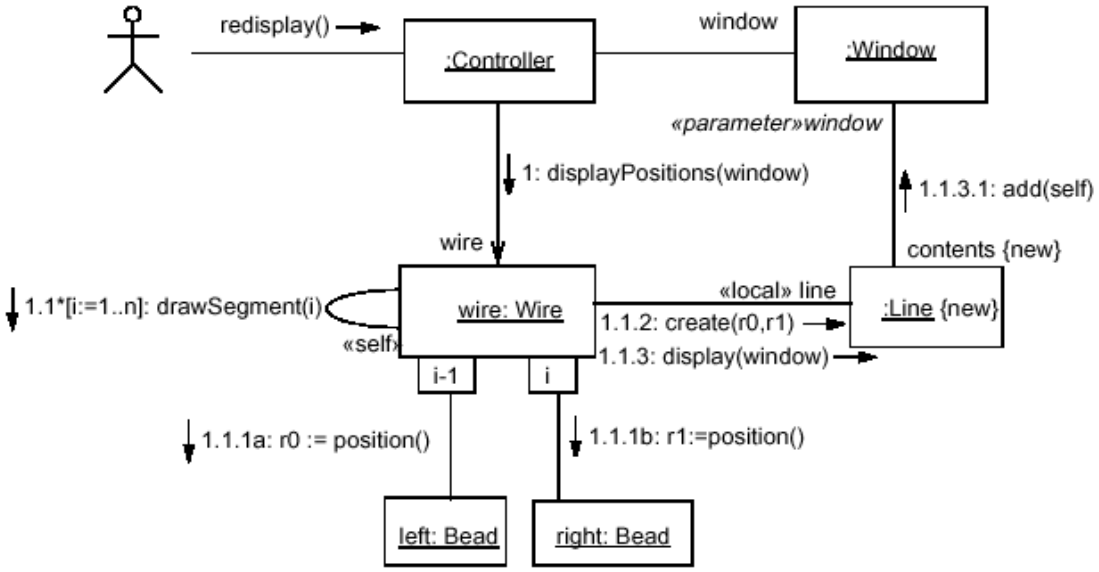

Figura 3: Exemplo de Diagrama de Comunicação

Observe alguns detalhes desse exemplo. Em primeiro, observe que todos os nomes nas caixas estão grifados. Isso indica que se trata de instâncias das classes e não das classes por si só. Alguns objetos possuem nomes, ao passo que outros apenas indicam a classe da instância. Observe que os relacionamentos entre as classes estão indicados, da mesma forma que apareceriam em um diagrama de classes. Entretanto, sobre cada relacionamento, existe uma pequena seta representando uma mensagem que é enviada de um objeto a outro. Observe especialmente o exemplo de recursão, onde o objeto **wire** manda um conjunto de mensagens para si próprio. Observe a sintaxe dessa mensagem. O caracter * indica que se trata de um conjunto de mensagens, e não de uma única mensagem. Em seguida, o string [i :=1..n] indica que se trata de uma recursão de n mensagens

indexadas por i. Observe ainda o tráfego de objetos. A mensagem 1.1.1a obtém como retorno um objeto **r0**, que é enviado posteriormente na mensagem 1.1.2 como um parâmetro para o objeto **:Line**. Um outro recurso muito utilizado também em diagramas de comunicação é o dos multi-objetos, ilustrado na figura a seguir:

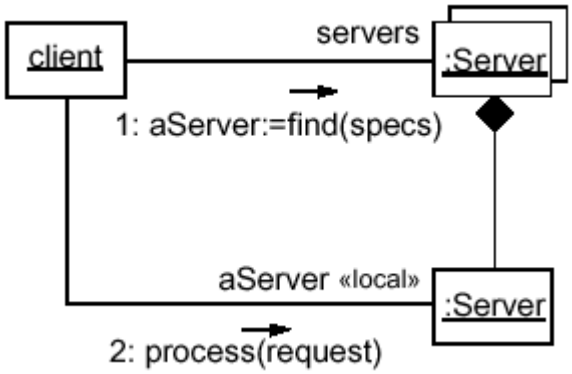

Figura 4: Exemplo de Multi-objetos

Um **multi-objeto** representa um conjunto de objetos posicionados em uma extremidade de uma associação que contenha multiplicidade maior que 1. Assim, os multi-objetos são utilizados para representar mensagens que são enviadas à coleção inteira de objetos ao invés de um único objeto dentro da coleção. Multi-objetos podem ter diferentes implementações em linguagens de programação. A mais conhecida delas corresponde aos chamados arrays ou vetores. Entretanto, linguagens mais avançadas como a linguagem Java possuem diversas outras implementações de multi-objetos, tais como os objetos do package Collections, que permitem a representação de listas, conjuntos, e outros tipos de coleções mais sofisticadas.