8

O Pronome ANY

Neste capítulo, é explorado o pronome ANY. Na Seção 8.1, o conceito de objeto qualquer é explicado. Na Seção 8.2, são mostradas as transformações de código necessárias para que o pronome ANY seja corretamente executado. Na Seção 8.3, é apresentado o nível Meta necessário para a obtenção destas transformações. Finalmente, na Seção 8.4, a implementação do padrão estrutural *Broker* [BMR+96], descrito na Subseção 2.4.5, utilizando o pronome ANY é comparada com a implementação tradicional.

8.1

Um Objeto Qualquer

Em sistemas baseados em eventos, objetos que os anunciam não conhecem o objeto que os tratarão. Ou seja, uma mensagem é enviada para um objeto destino qualquer, ou melhor, para quem primeiro se prontificar a recebê-la. O pronome ANY pode ser utilizado para representar este objeto “qualquer” no contexto do objeto emissor.

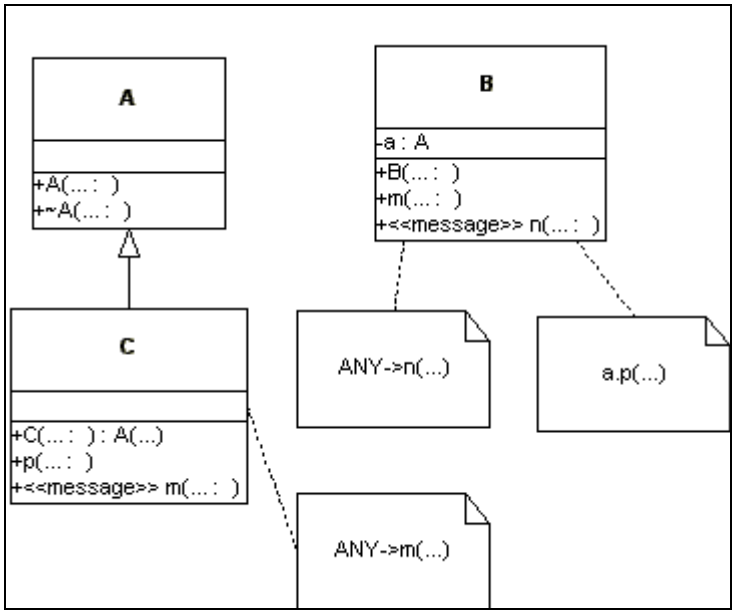

Figura 8.1 – Cenário de utilização do pronome ANY em nível base

A Figura 8.1 mostra o cenário no nível base de três classes, onde duas utilizam o pronome ANY. As classes *B* e *C*, em algum de seus métodos, enviam respectivamente as mensagens *n* e *m* através do pronome. As duas classes declaram as mensagens enviadas via pronome através do especificador <<message>> introduzido na linguagem.

8.2

Transformações de Código

Este cenário deve ser transformado de modo que:

1. as classes *A*, *B* e *C* possuam uma superclasse que implemente de forma vazia um tratador para *m()* e para *n()*; e
2. todos os objetos do sistema devem estar acessíveis no momento em que uma mensagem a um objeto qualquer é enviada.

Para a primeira transformação, pode ser adotada a mesma estratégia utilizada para os pronomes anteriores.

Para a segunda transformação, da mesma forma que para o pronome ALL, é necessário manter um repositório com todos os objetos do sistema para que a mensagem possa lhes ser oferecida. A diferença entre esta transformação e a transformação realizada para o pronome ALL está no momento do envio múltiplo das mensagens. Para ambos a mensagem é oferecida a todos os objetos contidos no repositório. Para o pronome ANY, no entanto, deve haver a garantia

que a mensagem será tratada por apenas um objeto. Este controle é feito pelo *singleton* [G+95] *RepositorioMsgDisp*, detalhado na Figura 8.3, introduzido como suporte de tempo de execução. Através deste, uma mensagem é disponibilizada antes de ser oferecida a todos os objetos e tem sua disponibilidade verificada antes de ser efetivamente tratada pelo objeto destino, como pode ser verificado na Figura 8.2. A disponibilização é feita através de um comando introduzido antes de todo envio de mensagem. Para envios de mensagens simples, uma mensagem é disponibilizada e um envio é feito. Para mensagens enviadas através do pronome ANY, apenas uma mensagem é disponibilizada para todos os envios. A verificação de disponibilidade é introduzida no início de cada tratador de mensagem. Sendo esta efetivamente tratada apenas se estiver disponível.

A observação de que o repositório retorne os objetos em uma ordem aleatória, colocada no pronome ALL como interessante, é de crucial importância para o pronome ANY, sob o risco de sempre o mesmo objeto tratar as mensagens enviadas através deste.

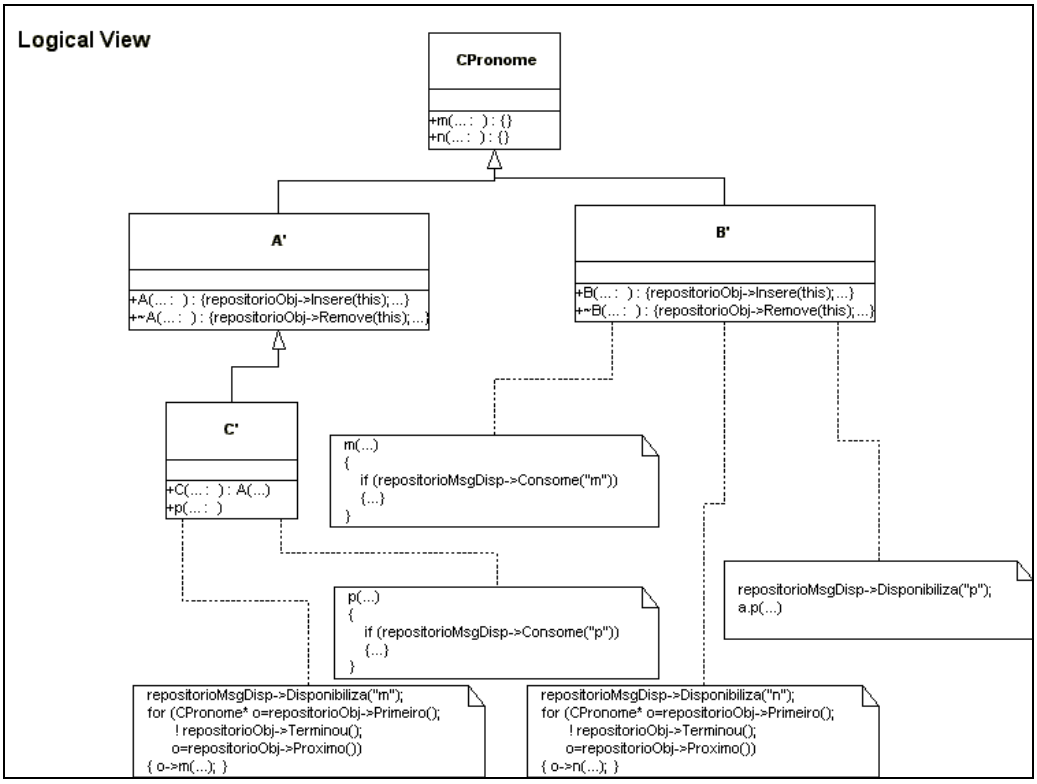

Figura 8.2 – Cenário de utilização do pronome ANY transformado

Figura 8.3 – Suporte de tempo de execução para utilização do pronome ANY

8.3

O Nível Meta

Para obter estas transformações, a todas as classes do nível base é associada a metaclasse *MCAny*, a menos a classe *CPronome* que é associada a uma metaclasse especial denominada *MCRaizPronome*. O nível meta e sua associação com o nível base mostrado anteriormente são detalhados na Figura 8.4.

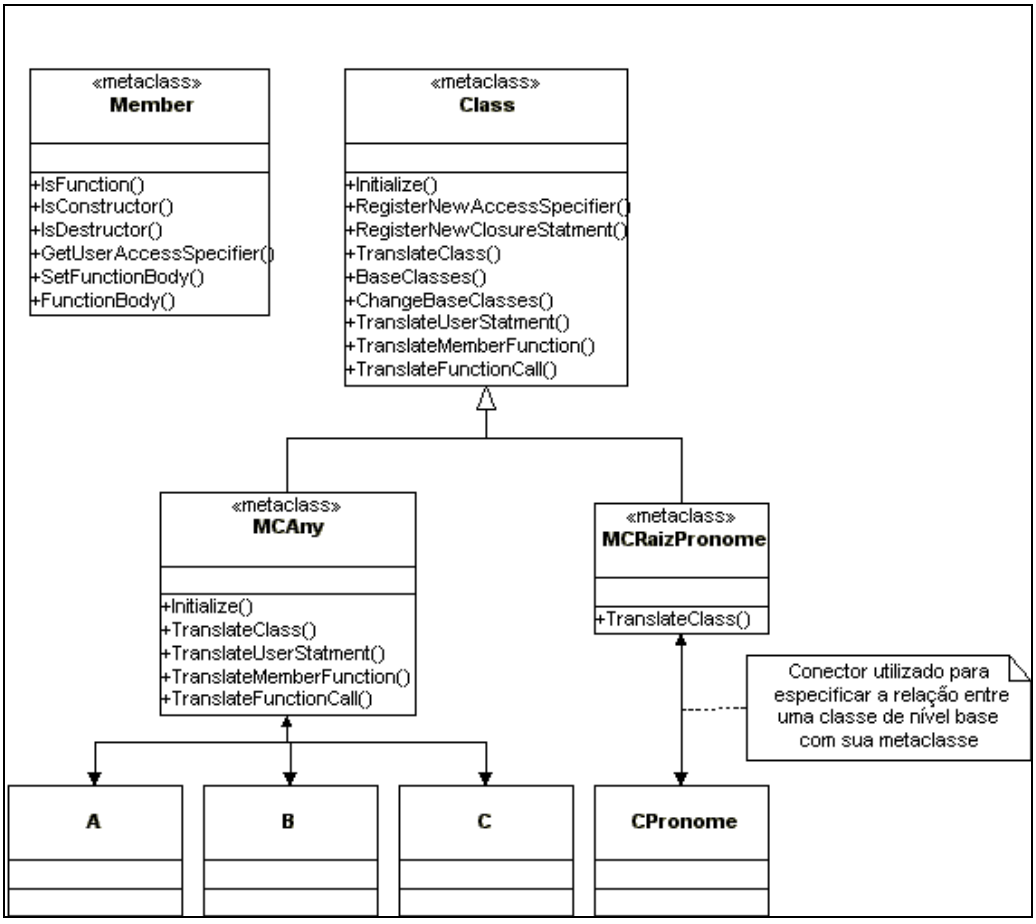

Figura 8.4 – Nível Meta para implementação do pronome ANY

Todos os métodos utilizados na implementação deste pronome já foram descritos nos capítulos anteriores não o sendo, portanto, novamente aqui.

Para que o especificador (message) e o novo comando (envio de mensagem para o ANY) seja identificado, a metaclasse *MCAny* redefine o método *Initialize()* e faz os dois registros:

- RegisterNewAcessSpecifier(“message”)
- RegisterNewClosureStatment(“ANY”)

As inserções da nova herança, a inserção dos objetos criados no repositório, feita nos construtores das classes raízes de árvore hierárquica, a remoção dos objetos do mesmo repositório, feita nos destrutores das mesmas classes, e o armazenamento das mensagens enviadas através do pronome são feitos na redefinição do método *TranslateClass()*.

Para o armazenamento das mensagens enviadas através do pronome, foi utilizado o objeto *Singleton* [G+95] *RepositorioMsg* que armazena as mensagens declaradas com o especificador <<message>> e as disponibiliza para ser introduzidas na classe *CPronome*.

A transformação dos tratadores de mensagens, que, antes de tratar efetivamente uma mensagem, devem verificar se esta está disponível, é feita redefinindo-se o método *TranslateMembemFunction()*. Este insere a verificação e só faz o tratamento caso o retorno desta seja positiva.

A transformação do envio efetivo da mensagem através do pronome ANY é feita redefinindo-se o método *TranslateUserStatment()*. Este substitui o envio único de mensagem original pela disponibilização da mensagem e posterior envio da mesma mensagem a todos os objetos contidos no repositório.

Ao término da transformação da última classe associada a *MCAny*, todas as mensagens enviadas através do pronome ANY estão armazenadas.

A metaclasse *MCRaizPronome* redefine o método *TranslateClass()* e neste, insere na classe *CPronome* uma função membro para cada mensagem armazenada no repositório, sempre com implementações vazias.

8.4

Padrão Arquitetural *Broker* Utilizando ALL

O padrão arquitetural *Broker* [BMR+96], descrito na Subseção 2.4.5, pode ter uma de suas funcionalidades implementada de forma bastante simples utilizando-se o pronome ANY. Esta funcionalidade seria o passo de localização de um servidor disponível para tratamento de alguma requisição, descrita da

seguinte forma no padrão : “O mediador deve manter um repositório para localizar outros mediadores ou *gateways* para os quais ele repassa mensagens (...) a fim de localizar servidores ou clientes”. Utilizando-se o pronome ANY, o mediador não precisa mais manter o repositório e requisições são disponibilizadas e oferecidas a todos os servidores. O primeiro que se desocupar, tratará a mensagem e todos os outros objetos a ignorarão.

Os tempos de execução da implementação tradicional da funcionalidade e da implementação utilizando o pronome ANY podem ser comparados na Tabela 8.1 e no gráfico mostrado na Figura 8.5.

Estes tempos foram conseguidos executando-se o padrão nas duas implementações e variando-se o número de objetos não mediadores. A variação foi escolhida porque, na implementação tradicional, objetos mediadores são também armazenados e mensagens são a eles oferecidos. Objetos não mediadores, por outro lado, são excluídos do controle de envio de mensagens na implementação tradicional e incluídos na implementação com o pronome, o que poderia causar uma baixa de desempenho.

Da mesma forma que nos pronomes descritos anteriormente, cada configuração do exemplo foi executada 1000 vezes, sendo os tempos mostrados na Tabela 8.1 a média destas execuções. Os desvios padrões das amostras consideradas são apresentados na Tabela 8.2. O ambiente de execução destes experimentos foi, também, o mesmo dos pronomes anteriores. Isto é, um computador Pentium4, com 256 MB de RAM, utilizando-se Linux Mandrake 8.1.

Número de objetos não mediadotes	**Implementação Tradiconal (μs)**	**Im plementação com ANY (μs)**	**Diferença (μs)**	**Aumento Percentual**
0	2343	2427	84	3,585147
1	2401	2508	107	4,456476
10	2524	2655	131	5,190174
100	3174	3349	175	5,513548
1000	7089	7473	384	5,416843
10000	68562	72359	3797	5,538053
100000	513632	537453	23821	4,63775621
1000000	3081692	3225627	143935	4,67064846
10000000	18489156	19383869	894713	4,839122997
100000000	110924939	116293214	5368275	4,839556414
1000000000	665449634	697659284	32209650	4,840283675
10000000000	3991697809	4184955704	193257895	4,841496131

Tabela 8.1 – Tempos de execução do padrão estrutural *Broker*

Número de objetos não mediadotes	Desvios Padrões Implementação Padrão	Desvios Padrões Implementação com ANY
0	0,2179	0,4969
1	0,3286	0,4129
10	0,2085	0,4063
100	0,1596	0,4187
1000	0,1971	0,3641
10000	0,3857	0,3050
100000	0,2971	0,2851
1000000	0,3064	0,2197
10000000	0,1950	0,2615
100000000	0,1863	0,2186
1000000000	0,2171	0,2015
10000000000	0,2053	0,2419

Tabela 8.2 – Desvios Padrões das amostras de tempo utilizadas para análise do padrão estrutural *Broker*

Os dados apresentados na Tabela 8.1 são melhor visualizados no gráfico da Figura 8.4.

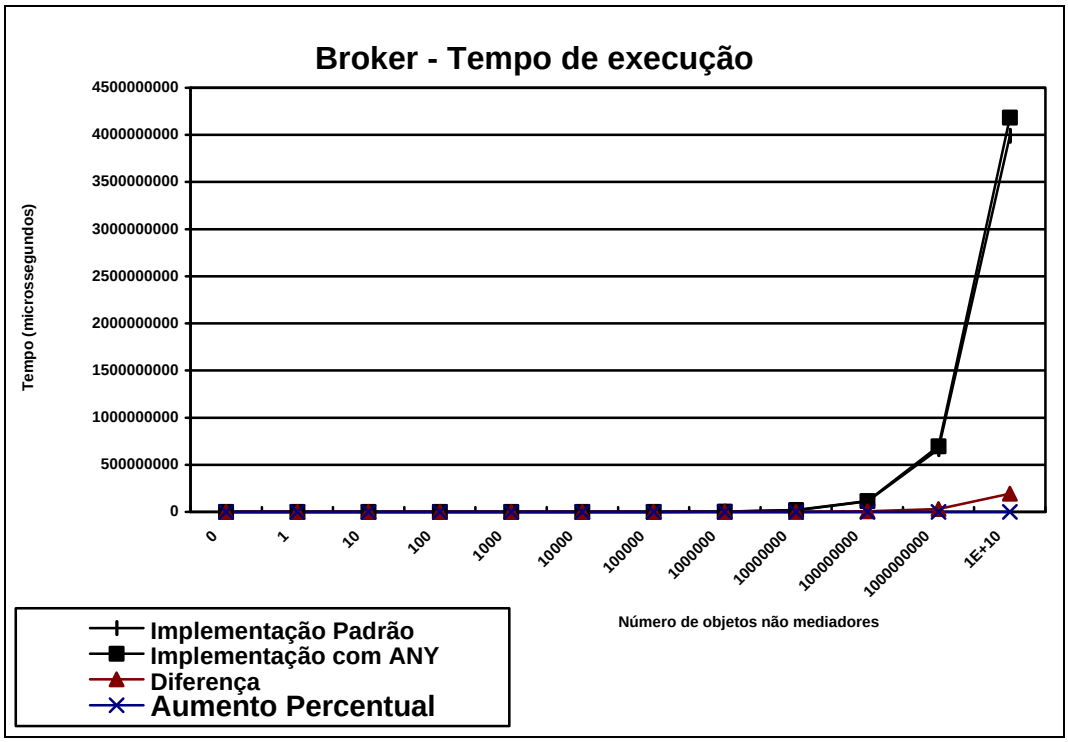

Figura 8.5 – Gráfico de comparação entre os tempos de execução do padrão *Broker*

Na Tabela 8.1 e no gráfico mostrado na Figura 8.5 pode-se verificar que o tempo de execução na implementação do padrão utilizando-se o pronome ANY é sempre um pouco maior, o que já era esperado. O aumento percentual máximo

obtido, no entanto, foi de 5%, evidenciando a não degradação do sistema com o uso do pronome.