9

Definindo Novos Pronomes

Neste capítulo, o *framework* abstraído da implementação dos pronomes explorados nos capítulos anteriores é apresentado. Na Seção 9.1, a motivação para a especificação do *framework* e o processo de abstração utilizado para obtê-lo são explicados. Na Seção 9.2, o *framework* é detalhadamente exposto. Nas Seções 9.3 a 9.8 os pronomes apresentados nos capítulos anteriores são obtidos a partir do *framework* e, finalmente, na Seção 9.9, este é instanciado para a obtenção de dois novos pronomes: CLOCK e SIBLINGS.

9.1

A Abstração do *Framework*

Os seis pronomes especificados nos capítulos anteriores provêem facilidades na implementação de um leque abrangente de arquiteturas e padrões de projetos além de validarem o conceito de pronomes, objetivo central deste trabalho. Este leque, no entanto, sempre pode ser estendido com a introdução de novos pronomes. Ou seja, a obtenção da completude do conjunto de pronomes é uma tarefa bastante difícil, senão impossível. A existência de um processo simples, o mais automático possível, para a definição e incorporação de novos pronomes em uma linguagem é, portanto, bastante desejável.

A implementação do conjunto de pronomes através de linguagens abertas torna-os acessíveis a qualquer linguagem reflexiva. Esta mesma implementação deixou também claras algumas semelhanças existentes entre as implementações dos diferentes pronomes. Uma prova disso é a diminuição de funções OpenC++ introduzidas a cada capítulo:

- 12 no Capítulo 3 (pronome SENDER);
- 11 no Capítulo 4 (pronome CREATOR);
- 5 no Capítulo 5 (pronome PARENT);
- 1 no Capítulo 6 (pronome MAIN);

- 0 nos Capítulos 7 e 8 (pronomes ALL e ANY respectivamente).

Ou seja, o conjunto de características de uma linguagem que precisam ser expostas através de reflexão para que um pronome possa nesta ser introduzido é restrito e, apesar de provavelmente não se ter atingido sua completude, conseguimos um subconjunto bastante representativo destas.

Além disso, as transformações implementadas para cada pronome, apesar de obviamente se diferenciarem bastante devido às diferentes características dos pronomes escolhidos, seguem um mesmo padrão. Por exemplo, no processo de inicialização, ocorrem, pelo menos, duas declarações: o pronome em si é declarado como palavra chave para um novo comando e a palavra chave *message* é declarada como um novo especificador. Outra característica que se repete: em toda árvore hierárquica é introduzida a raiz *CPronome*. Nesta classe, são introduzidos, com implementação vazia, tratadores para todas as mensagens enviadas através de pronomes.

A partir destas semelhanças, um *framework* para a definição de pronomes foi abstraído das implementações propostas nos capítulos anteriores e é apresentado a seguir.

9.2

O *Framework*

O *framework* pode ser dividido em duas partes. Na primeira, um *framework* composto apenas de metaclasses a serem estendidas para definição do nível Meta. Na segunda, um *framework* composto de classes a serem estendidas para criação de um suporte para a execução do sistema.

Estas duas partes são apresentadas, respectivamente, nas Figuras 9.1 e 9.2. Nestas, métodos em itálicos são abstratos, representando os pontos de extensão do *framework*.

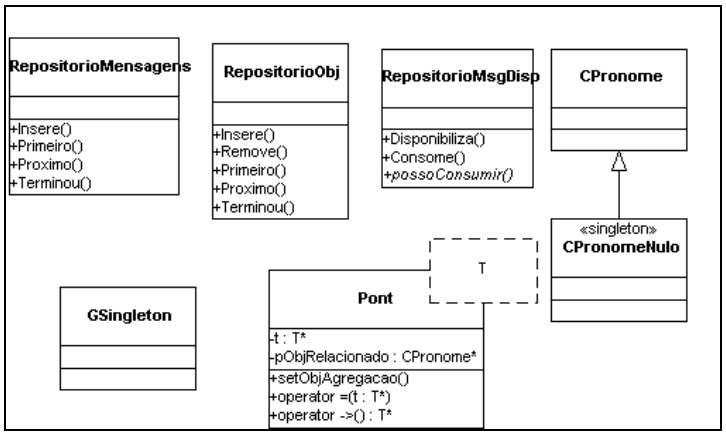

Figura 9.1 –*Framework* para suporte de tempo de execução

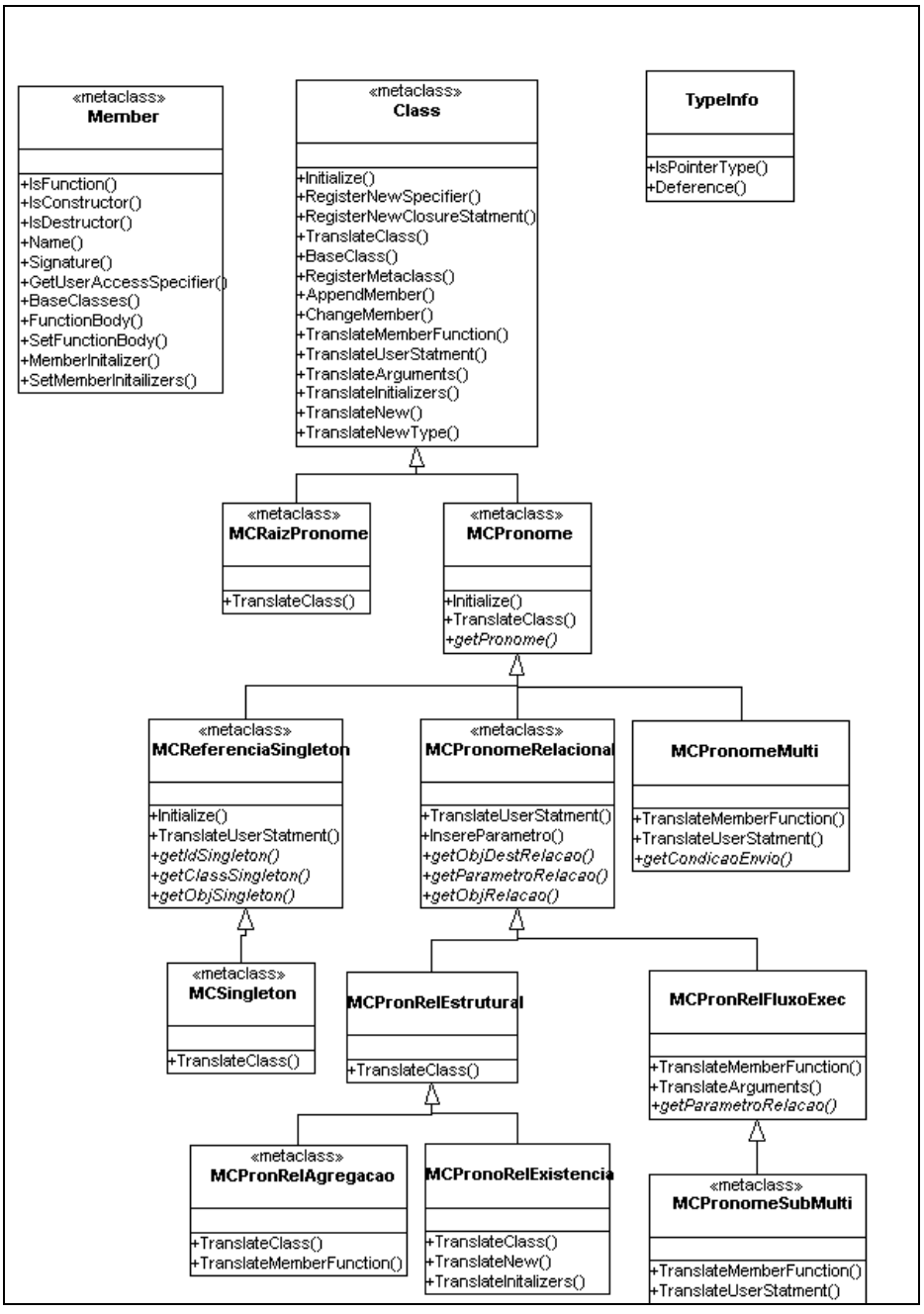

Figura 9.2 – Nível Meta do f*ramework*

Do modelo reflexivo disponibilizado pela linguagem aberta, são utilizadas as metaclasses que expõem as classes de nível base (*Class*), seus membros (*Member*) e os tipos destes (*TypeInfo*). Estas precisam disponibilizar os métodos citados na Figura 9.1 (já descritos nos capítulos anteriores), ou similares.

Nas subseções seguintes o *framework* é detalhadamente explicado.

9.2.1

Primeiro Nível de Abstração

Em um primeiro nível de abstração, como pode ser visto na Figura 9.3, a metaclasse que expõe as classes de nível base é estendida por duas metaclasses introduzidas no modelo: *MCPronome* e *MCRaizPronome*.

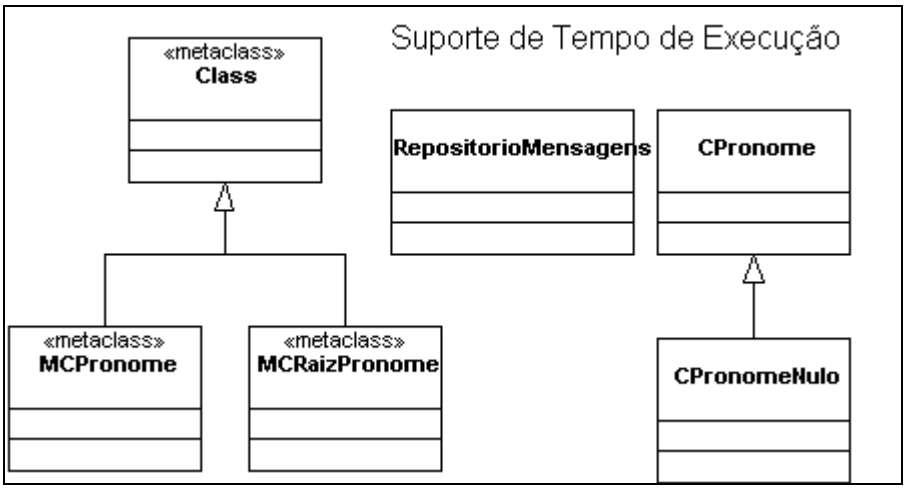

Figura 9.3 – Primeiro nível de abstração do *framework*

MCPronome é a superclasse de todas as metaclasses que promovem transformações associadas a pronomes, redefine os seguintes métodos:

- *Initialize()*: Neste método, são registrados o novo especificador *message* e o respectivo pronome como palavra-chave para o novo comando. Como se trata de uma metaclasse genérica para qualquer metaclasse de pronomes, é utilizado para este último registro o retorno da função virtual (e abstrata nesta metaclasse) *getPronome()*. Toda metaclasse de pronome deve, portanto, implementá-la, retornando o respectivo pronome.

- *TranslateClass()*: Neste método, a superclasse *CPronome* é introduzida, se a classe que está sendo transformada for raiz de uma árvore hierárquica. Além disso, toda função cujo especificador seja *message* é armazenada, depois de transformada, no *singleton* [G+95] *RepositorioMensagens*, provido como suporte de tempo de execução, e removida da classe.

RepositorioMensagens armazena as mensagens enviadas através de pronomes para que tratadores para estas, com implementações vazias, sejam posteriormente incluídos na classe *CPronome*, também provida como suporte de tempo de execução.

MCRaizPronome é a metaclasse da classe *CPronome* e redefine o método *TranslateClass()*, que insere nesta classe tratadores com implementações vazias para as mensagens coletadas pelo *singleton* descrito acima.

CPronomeNulo é uma herança de CPronome, também provida como suporte de tempo de execução, que é utilizada como o objeto representado por um pronome quando este não é válido em alguma situação, para evitar erros de execução. Um exemplo seria a representação do objeto que criou um objeto global.

9.2.2

Categorização de Pronomes

Os pronomes podem ser categorizados de três formas, sendo cada uma destas representada por uma metaclasse no *framework* já que suas implementações se assemelham. Esta configuração pode ser visualizada na Figura 9.4.

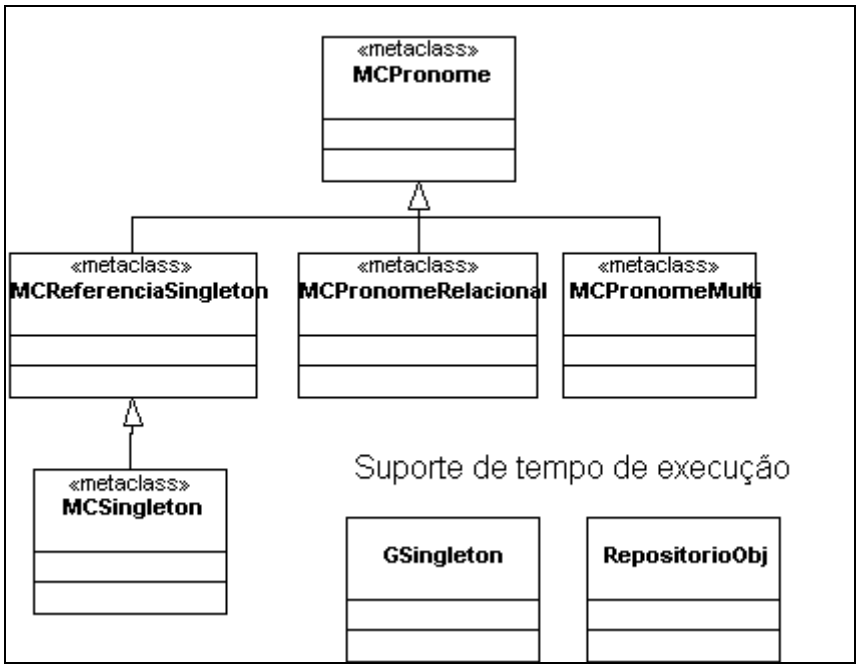

Figura 9.4 – Categorização dos pronomes segundo suas características de implementação

MCReferenciaSingleton é a superclasse de todas as metaclasses que se referenciam a pronomes que representam um objeto único no sistema. Redefine os métodos:

- *Initialize()*: Neste método, além dos registros feitos pelo mesmo método na superclasse *MCPronome*, o identificador do *singleton* é registrado como modificador e associado a uma metaclasse específica. Como se trata de uma metaclasse genérica para todas as metaclasses que se utilizam de pronomes que representam *singletons*, são utilizados para este último registro os retornos das funções virtuais (e abstratas nesta metaclasse) *getIdSingleton()*, que retorna o identificador do *singleton*, e *getClassSingleton()*, que retorna a respectiva metaclasse. Toda metaclasse de classe que se utilize de pronomes que representam um *singleton* deve, portanto, implementá-las, retornando os respectivos dados.
- *TranslateUserStatment()*: Neste método, o comando de envio de mensagem através do pronome é transformado, sendo substituído pelo objeto *singleton*. Novamente, por ser uma metaclasse genérica, é utilizado, para a transformação, o retorno da função virtual (abstrata nesta metaclasse) *getObjSingleton()*, que retorna a referência para o objeto *singleton* referenciado via pronome.

MCSingleton é a superclasse de todas as metaclasses que necessitem promover as transformações para que o objeto modelado por uma determinada classe seja um *singleton*, isto é, um objeto único no sistema. Redefine o método:

- *TranslateClass()* : Neste método, além das transformações definidas pelo mesmo método na superclasse *MCReferênciaSingleton*, em todo construtor, o objeto que está sendo criado é atribuído ao objeto que representará o *singleton*. Como esta metaclasse ainda é genérica, a referência ao objeto que representará o *singleton* é retornado pela função virtual *getObjSingleton()* (definida em *MCReferenciaSingleton* e ainda abstrata nesta metaclasse).

MCPronomeRelacional é a superclasse de todas as metaclasses que implementam pronomes que representam um objeto que se relaciona de alguma forma com o objeto corrente da execução. Redefine o método:

- *TranslateUserStatment()*: Neste, o comando de envio de mensagem através do pronome é transformado, sendo substituído pelo objeto relacionado com o objeto corrente através da relação definida pelo pronome. Por ser esta uma metaclasse genérica, é utilizado para a transformação o retorno da função virtual (abstrata nesta metaclasse) *getObjDestRelacao()*, que retorna o objeto associado ao pronome em questão.

A metaclasse implementa ainda o método *InsereParâmetro(m,i,parm)* que insere *parm* como *i-ésimo* parâmetro do membro *m* e será utilizado por suas subclasses. Além disso, define de forma abstrata os métodos:

- *getParametroRelacao()*: que deve retornar o parâmetro a ser inserido no cabeçalho de algum método para receber o objeto representado pelo pronome.
- *getObjRelacao()*: que deve retornar o objeto que será representado pelo pronome no momento em que a relação definida por este é estabelecida.

MCPronomeMulti é a superclasse de todas as metaclasses que implementam pronomes que representam um conjunto de objetos, podendo este ser formado de apenas um, ou até nenhum, objeto. Redefine os métodos:

- *TranslateClass()* : Neste método, os objetos são coletados quando criados e dispensados quando destruídos. Para tanto, em todo construtor é introduzido um comando que exporta o objeto que está sendo criado para o *singleton RepositorioObj*, provido como suporte de tempo de execução, que os armazena. Em todo destrutor, por outro lado, é introduzido um comando que indica ao mesmo *singleton*, que o objeto deve deixar de ser armazenado.
- *TranslateUserStatment()*: Neste método, o comando de envio de mensagem através do pronome é transformado, sendo substituído por uma iteração sobre os objetos armazenados pelo *singleton RepositórioObj*, com a mensagem sendo enviada a todos os objetos obtidos na iteração, caso uma determinada condição de envio esteja satisfeita. Esta condição de envio varia de pronome para pronome, sendo portanto dada pelo retorno do método virtual (e abstrato nesta metaclasse) *getCondicaoEnvio()*, que deve set implementado em toda metaclasse que implemente pronome que represente um conjunto de objetos.

RepositorioObj é um *singleton* [G+95], provido como suporte de tempo de execução, que é responsável pelo armazenamento das referências aos objetos do sistema. Além disso, disponibiliza uma interface de acesso a estes.

GSingleton é um *singleton* [G+95], provido como suporte de tempo de execução, superclasse de toda referência global a *singletons* introduzidos através de pronomes.

9.2.3

Subconjunto de Objetos do Sistema

Além da relação que define os objetos que fazem parte do conjunto, um pronome pode representar também um subconjunto deste, ou ainda um objeto único deste conjunto. Seria o caso, por exemplo, do pronome ANY que representa um único objeto do conjunto de todos os objetos do sistema. Esta restrição pode ser conseguida estendendo-se a metaclasse *MCPronomeMulti* e

introduzindo-se um novo *singleton*, *RepositorioMsgDisp*, como suporte de tempo de execução, como pode ser visto na Figura 9.5.

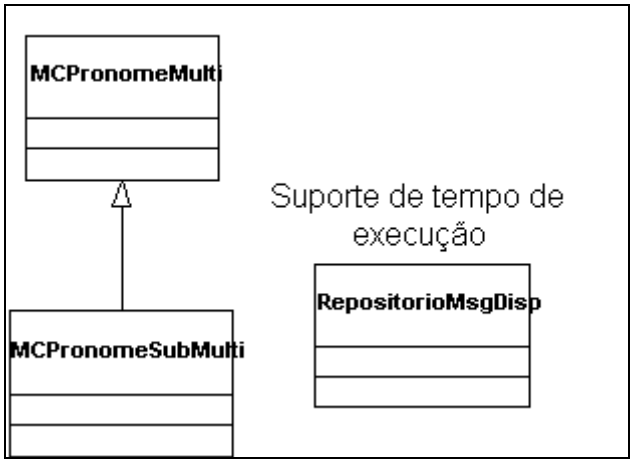

Figura 9.5 – Subconjuntos de objetos do sistema

MCPronomeSubMulti é, portanto, a superclasse de todas as metaclasses que implementam pronomes que representam um subconjunto de objetos do sistema. Redefine os métodos:

- *TranslateMemberFunction()*: Introduz, em toda função membro, a verificação, através do *singleton RepositorioMsgDisp* fornecido como suporte de tempo de execução, se a respectiva mensagem está disponível para o consumo. Em caso positivo, o restante do método é executado, efetivando seu tratamento. Em caso negativo, no entanto, o método retorna sem nada executar, uma vez que não há efetivamente nenhuma mensagem a consumir.
- *TranslateUserStatment()*: Este método, antes de efetivar as transformações promovidas pelo mesmo método na superclasse *MCPronomeMulti*, disponibiliza a mensagem que está sendo enviada através do *singleton RepositorioMsgDisp*. Com isto, quando as indicações de envio forem efetivamente realizadas, a mensagem estará disponível para consumo. É claro que, para que haja uma disputa justa pela mensagem disponibilizada, é necessária a utilização de um semáforo que garanta que a mensagem só estará disponível para consumo quando todos os objetos já tiverem sido avisados desta disponibilidade.

RepositorioMsgDisp é a superclasse de todo *singleton* provido como suporte de tempo de execução para gerenciar mensagens a subgrupos de objetos. Disponibiliza os métodos:

- *Disponibiliza()*: Coloca uma mensagem no estado de disponível para o resto do sistema.
- *Consome()*: Verifica se uma mensagem está disponível. Em caso negativo, retorna ao objeto chamador a indicação que a mensagem não foi consumida. Em caso positivo, a consome, caso isso possa ser feito, e retorna a indicação que a mensagem foi consumida e, conseqüentemente, precisa ser tratada. A verificação se a mensagem pode ser consumida depende do pronome através do qual ela foi enviada. É feita, portanto, pelo método virtual (e abstrato nesta classe) *possoConsumir()* que deve ser implementado em todo *singleton* que gerencia mensagens enviadas a subgrupos de objetos.

9.2.4

Categorização das relações com o objeto corrente da execução

Objetos podem se relacionar com o objeto corrente da execução de duas formas: estruturalmente, quando a relação é definida com base na forma como o sistema é construído; ou dinamicamente, quando a relação é definida com base em como o sistema é executado. Para prover estas duas formas, como pode ser conferido na Figura 9.6, o *framework* estende a metaclasse *MCPronomeRelacional*.

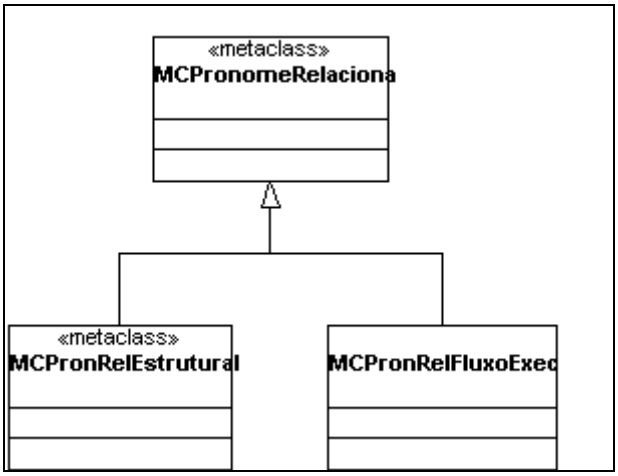

Figura 9.6 – Categorização de pronomes relacionados com o objeto corrente da execução

MCPronRelEstrutural é a superclasse de toda metaclasse que implementa pronomes que representam um objeto que se relaciona com o objeto corrente da execução através de alguma relação estrutural. Redefine o método:

- *TranslateClass()*: Além das transformações definidas na superclasse *MCPronome*, este método insere, caso a classe que está sendo transformada seja raiz de uma árvore hierárquica, um atributo onde será armazenado o objeto representado pelo pronome. Como esta é uma metaclasse genérica, o atributo a ser inserido é dado pelo retorno do método virtual *getObjDestRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse, que deve ser implementado por toda metaclasse que implemente pronomes que representem objetos relacionados estruturalmente com o objeto corrente da execução.

MCPronRelFluxoExec é a superclasse de toda metaclasse que implementa pronomes que representam objetos cujos relacionamentos com o objeto corrente da execução são definidos pelo fluxo de execução do sistema. Redefine os métodos:

- *TranslateMemberFuntion()*: Introduz, no cabeçalho de todo método, um parâmetro extra que representará o pronome no corpo deste método. Como esta é uma metaclasse genérica, o parâmetro a ser inserido é dado pelo retorno do método virtual *getParametroRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse.

- *TranslateArguments()*: Introduz, na lista de argumentos de um envio de mensagem, o objeto que será representado pelo pronome no corpo do método. Como esta é uma metaclasse genérica, o argumento a ser inserido é dado pelo retorno do método virtual *getObjRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse.

9.2.5

Categorização das Relações Estruturais

Relações estruturais ainda podem ser categorizadas em relações estáticas, que podem ser totalmente definidas quando o sistema inicia sua execução, ou dinâmicas, que só podem ser definidas durante a execução do sistema. Esta categorização pode ser visualizada na Figura 9.7.

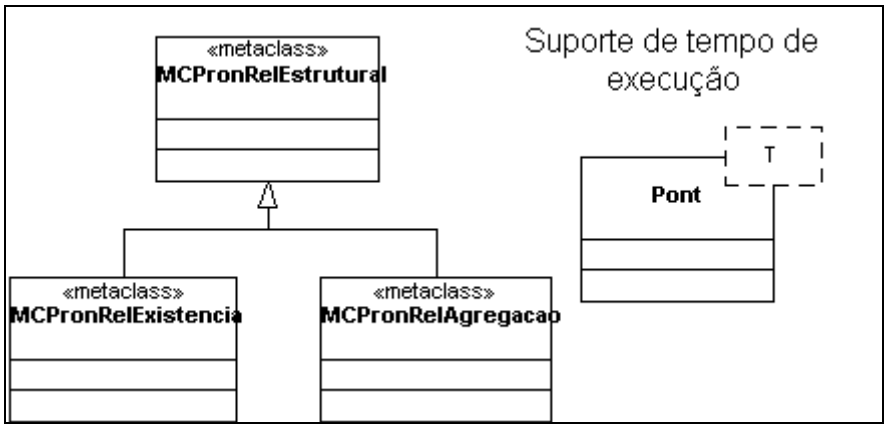

Figura 9.7 – Categorização dos pronomes que representam objetos relacionados estruturalmente com o objeto corrente da execução.

MCPronRelExistencia é a superclasse de toda metaclasse que implementa pronomes que representem um objeto que se relaciona, por construção, com o objeto corrente da execução. Redefine os métodos:

- *TranslateClass()*: Além das transformações definidas na superclasse *MCPronomeRelacional*, transforma os inicializadores de todo construtor e insere neste um parâmetro extra através do qual o objeto que será representado pelo pronome lhe será passado. Como esta é uma metaclasse genérica, o nome deste

parâmetro é dado pelo retorno do método virtual *getParametroRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse.

- *TranslateNew()*: Transforma um comando de construção de um objeto, inserindo o objeto que será representado pelo pronome como um argumento extra para o construtor. Como esta é uma metaclasse genérica, este argumento é dado pelo retorno do método virtual *getObjRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse.
- *TranslateInitializers()*: Transforma os inicializadores de um construtor, inserindo um argumento extra nas chamadas a construtores de atributos. Como estes estão sendo criados, o argumento extra será uma referência ao objeto que será representado pelo pronome no contexto dos atributos. Como esta é uma metaclasse genérica, este argumento é dado pelo retorno do método virtual *getObjRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse. Atributos não referenciados na lista de inicializadores são ali introduzidos já com o parâmetro extra inserido. Aos construtores de superclasses é repassado o parâmetro recebido como objeto destino da relação. Novamente, como esta é uma metaclasse genérica, este parâmetro é dado pelo retorno de outro método virtual *getParametroRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse. Superclasses que não estejam referenciadas na lista de inicializadores, são ali introduzidas já com o parâmetro extra inserido.

MCPronRelAgregacao é a superclasse de toda metaclasse que implementa pronomes que representem objetos que se relacionam de forma estática com o objeto corrente da execução. Redefine os métodos:

- TranslateClass() : Além das transformações já promovidas pelo método na superclasse *MCPronomeEstrutural*, este método transforma todo atributo que seja uma referência em um objeto modelado pela classe genérica *Pont<T>*, provida como suporte de tempo de execução. Introduz também os métodos *SetObjAgregacao()* e *GetObjAgregacao()* que dão acesso ao objeto que armazena o objeto destino da relação mapeada pelo pronome.

Como esta ainda é uma metaclasse genérica onde esta relação não é conhecida, este objeto é obtido pelo retorno do método *getObjDestRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse.

- *TranslateMemberFunction()*: Além das transformações já promovidas pelo método na superclasse *MCPronomeEstrutural*, este método inicializa o objeto que armazena o objeto destino da relação mapeada pelo pronome caso a função membro seja um construtor. Neste caso, também, invoca o método *SetObjAgregacao()* de todo atributo dinâmico, passando a referência para o objeto que será representado pelo pronome no momento em que a relação definida por este é estabelecida. Esta referência é obtida pelo método *getObjRelacao()*, definido em *MCPronomeRelacional* e ainda abstrato nesta metaclasse.

Como relações estáticas já estão definidas quando um sistema inicia sua execução, para objetos dinâmicos, que só são criados durante a execução do mesmo, a relação se define antes mesmo de suas criações. Neste caso, o objeto relacionado precisa ser guardado para, quando o objeto for efetivamente criado, a relação possa ser corretamente mapeada. Isto é feito utilizando-se a classe genérica *Pont<T>* provida como suporte de tempo de execução, que define os métodos:

- *SetObjAgregacao()* : Armazena o objeto que será representado pelo pronome de agregação.
- *operator = ()* : Disponibiliza um forma de se alterar o objeto efetivo, atribuindo a este a referência passada. Além disso, caso o objeto que será representado pelo pronome ainda não esteja inicializado no objeto efetivo, inicializa-o com a referência armazenada.
- operator -> () : Retorna o objeto efetivo.

9.3

Instanciação do Framework para a Obtenção do Pronome MAIN

A implementação de pronomes que representam um objeto único no sistema é feita estendendo-se as metaclasses *MCReferenciaSingleton* e

MCSingleton. A implementação através do *framework* do pronome MAIN apresentado anteriormente pode ser conferido na Figura 9.8.

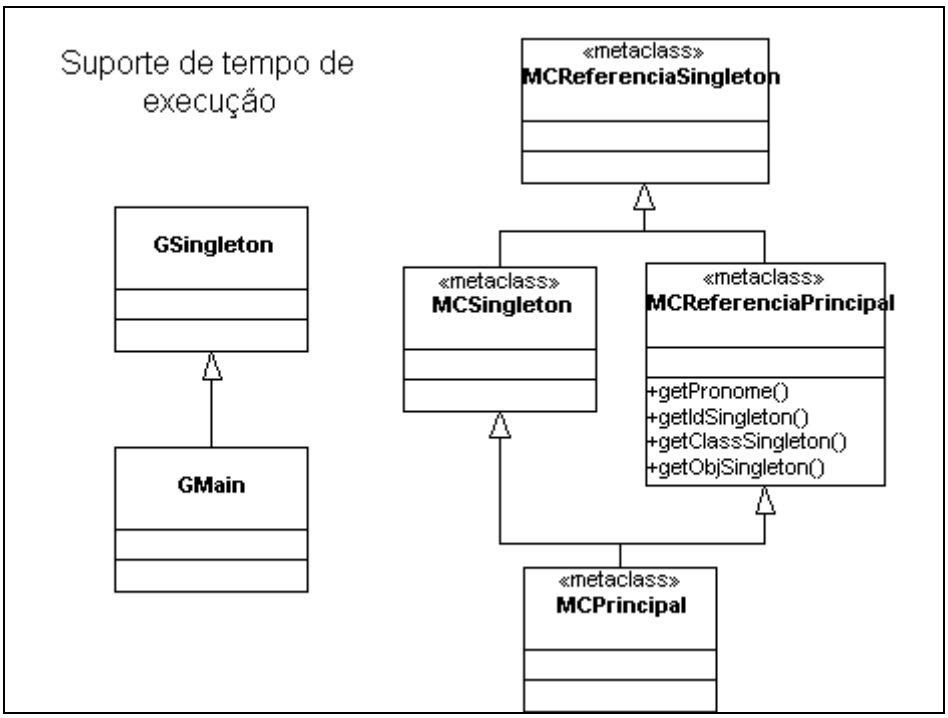

Figura 9.8 – Implementação do pronome MAIN através do *framework*

MCReferenciaPrincipal é a metaclasse que promove as transformações necessárias para que classes possam se referenciar ao objeto principal do sistema. Redefine os métodos:

- *getPronome()*: Definido de forma abstrata na metaclasse *MCPronome*, este método retorna o pronome a que esta metaclasse se refere, no caso, MAIN.
  - *getClassSingleton()*: Definido de forma abstrata na metaclasse *MCReferenciaSingleton*, este método retorna a metaclasse que implementa o pronome MAIN, no caso, *MCPrincipal*.
  - *getIdSingleton()*: Definido de forma abstrata na metaclasse *MCReferenciaSingleton*, este método retorna o modificador da classe que a identifica como *singleton,* no caso MAIN.
  - *getObjSingleton()*: Definido de forma abstrata na metaclasse *MCReferenciaSingleton*, este método retorna o objeto *singleton* referenciado via pronome, no caso, *gMain*.

MCPrincipal é a metaclasse que promove as transformações necessárias para a classe representar o objeto principal do sistema. Esta metaclasse não redefine nenhum método, herdando, no entanto, tanto de *MCSingleton* quanto de *MCReferenciaPrincipal*. Daquela herda o método *TranslateClass()* e desta os métodos que definem o pronome MAIN.

9.4

Instanciação do Framework para a Obtenção do Pronome ALL

A implementação de pronomes que representam um conjunto de objetos, eventualmente composto de um, ou até nenhum, objeto, é feita estendendo-se a metaclasse *MCPronomeMulti*. A implementação do pronome ALL através do *framework* pode ser conferida na Figura 9.9.

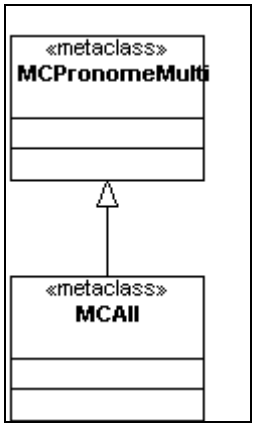

Figura 9.9 – Implementação do pronome ALL através do *framework*

MCAll é a metaclasse que promove as transformações necessárias para que as classes possam se referenciar a todos os objetos do sistema. Redefine os métodos:

- *getPronome()*: Definido de forma abstrata na metaclasse *MCPronome*, este método retorna o pronome a que esta metaclasse se refere, no caso, ALL.
- *getCondicaoEnvio()*: Definido de forma abstrata na metaclasse *MCPronomeMulti*, este método retorna se a mensagem pode ser enviada para objeto. No caso, como o pronome em questão se refere a todos os objetos sem restrições, a condição é sempre verdadeira.

9.5

Instanciação do Framework para a Obtenção do Pronome ANY

O pronome ANY, que representa um objeto qualquer do sistema, pode ser implementado estendo-se a metaclasse *MCPronomeSubMulti* e a classe *RepositorioMsgDisp*, como pode ser visto na Figura 9.10.

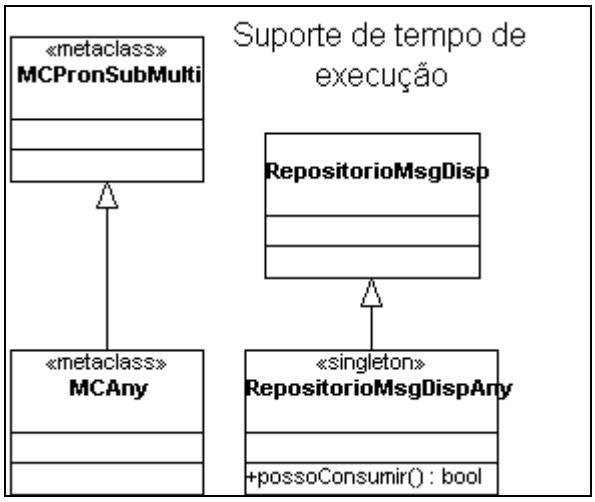

Figura 9.10 – Implementação do pronome ANY através do *framework*

MCAny é a metaclasse que promove as transformações necessárias para que as classes possam se referenciar a algum objeto do sistema. Redefine os métodos:

- *getPronome()*: Definido de forma abstrata na metaclasse *MCPronome*, este método retorna o pronome a que esta metaclasse se refere. No caso, ANY.
- *getCondicaoEnvio()*: Definido de forma abstrata na metaclasse *MCPronomeMulti*, este método retorna se a mensagem pode ser enviada ao objeto. No caso, como o pronome em questão se refere a qualquer o objeto sem restrições, a condição é sempre verdadeira.

RepositorioMsgDispAny é a classe do *singleton* provido como suporte de tempo de execução para gerenciar mensagens a um objeto qualquer do sistema. Disponibiliza o método:

- *possoConsumir()*: Definido de forma abstrata na classe *RepositorioMsg*, este método verifica se a mensagem que está

disponível pode ser consumida. No caso, esta pode ser consumida se ainda não o tiver sido.

9.6

Instanciação do Framework para a Obtenção do Pronome SENDER

O pronome SENDER pode ser implementado estendendo-se a metaclasse *MCPronRelFluxoExec*, como pode ser visto na Figura 9.11.

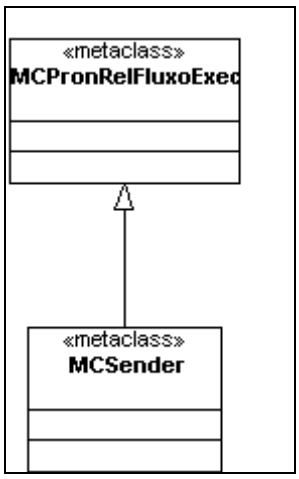

Figura 9.11 – Implementação do pronome SENDER através do *framework*

MCSender é a metaclasse que promove as transformações necessárias para que as classes possam se referenciar aos objetos que lhes enviam mensagens. Redefine os métodos:

- *getPronome()*: Definido de forma abstrata na metaclasse *MCPronome*, este método retorna o pronome a que esta metaclasse se refere, no caso, SENDER.
- *getParametroRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o parâmetro através do qual o objeto que está sendo representado pelo pronome é passado para o contexto do objeto que está se utilizando deste, no caso, *pSender*.
- *getObjDestRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que, no contexto

do objeto corrente da execução, armazena a referência ao objeto representado pelo pronome, no caso, também o parâmetro *pSender*.

- *getObjRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que será representado por este no momento em que a relação definida pelo pronome é estabelecida. No caso, a relação entre emissor e receptor é estabelecida no momento do envio da mensagem. Neste momento, o objeto corrente da execução é o objeto emissor e deve ser passado ao objeto receptor. Este método deve retornar, portanto, uma referência ao objeto corrente da execução para que seja inserido com argumento extra no envio da mensagem.

9.7

Instanciação do Framework para a Obtenção do Pronome CREATOR

O pronome CREATOR pode ser implementado estendendo-se a metaclasse *MCPronRelExistencia,* como mostra a Figura 9.12.

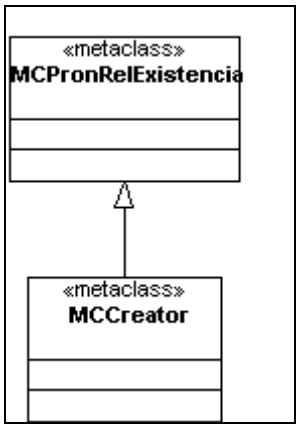

Figura 9.12 – Implementação do pronome CREATOR através do *framework*

MCCreator é a metaclasse que promove as transformações necessárias para que as classes possam se referenciar aos objetos que criaram suas instâncias. Redefine os métodos:

- *getPronome()*: Definido de forma abstrata na metaclasse *MCPronome*, este método retorna o pronome a que esta metaclasse se refere, no caso, CREATOR.
- *getParametroRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o parâmetro através do qual o objeto que está sendo representado pelo pronome é

passado para o contexto do objeto que está se utilizando deste, no caso, *pCreator*.

- *getObjDestRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que, no contexto do objeto corrente da execução, armazena a referência ao objeto representado pelo pronome, no caso, o atributo *m_pCreator*.
- *getObjRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que será representado pelo pronome no momento em que a relação definida por este é estabelecida. No caso, a relação entre criador e criatura é estabelecida no momento em que um objeto é criado. Neste momento, o objeto corrente da execução é o objeto criador e deve, portanto, ser passado ao objeto que está sendo criado. Este método deve retornar, portanto, a referência para o objeto corrente da execução para ser inserido com argumento extra na chamada ao construtor.

9.8

Instanciação do Framework para a Obtenção do Pronome PARENT

O pronome PARENT pode ser implementado estendendo-se a metaclasse MCPronRelAgregação como mostra a Figura 9.13.

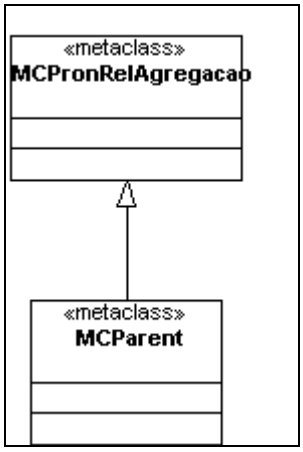

Figura 9.13 – Implementação do pronome PARENT através do *framework*

MCParent é a metaclasse que promove as transformações necessárias para que as classes possam se referenciar aos objetos que declararam suas instâncias. Redefine os métodos:

- *getPronome()*: Definido de forma abstrata na metaclasse *MCPronome*, este método retorna o pronome a que esta metaclasse se refere, no caso, PARENT.
- *getParametroRelacao()*: Definido de forma abstrata na metaclasse *MCPronRelacional*, este método retorna o parâmetro através do qual o objeto que está sendo representado pelo pronome é passado para o contexto do objeto que está se utilizando destes, no caso, *pParent*.
- *getObjDestRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que, no contexto do objeto corrente da execução, armazena a referência ao objeto representado pelo pronome, no caso, o atributo *m_pParent*.
- *getObjRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que será representado pelo pronome no momento em que a relação definida por este é estabelecida. No caso, a relação entre pai e filho é estabelecida no momento em que um objeto é declarado. Neste momento, portanto, a referência para o objeto que está sendo criado deve ser passada a todos os seus atributos. Este método deve retornar, portanto, a referência para o objeto corrente da execução para que seja inserida como argumento extra na chamada aos construtores dos atributos.

9.9

A Instanciação do Framework para a Obtenção de Novos Pronomes

Definido o *framework*, seguem dois exemplos de sua utilização na definição dos pronomes CLOCK e SIBLINGS. O primeiro representa um *singleton* [G+95] que marca de alguma forma, lógica ou física, o tempo do sistema. O segundo representa o conjunto dos objetos declarados pelo mesmo PARENT que o objeto corrente da execução.

9.9.1

O Pronome CLOCK

O pronome CLOCK pode ser utilizado na ordenação de eventos, como descrito em [Lam78]. Este utiliza relógios lógicos para ordenar eventos em programação distribuída. Esta ordenação é a base para muitos protocolos de sincronização descentralizados [And91].

Processos em um programa distribuído executam ações locais e ações de comunicação. Ações de comunicação são envio e recebimento de mensagens e efetam a execução de outros processos, uma vez que trocam informações, e são a base do mecanismo de sincronização.

A Figura 9.14 mostra três processos em um programa distribuído executando ações locais e ações de comunicação. Se estes processos executassem apenas ações locais, não haveria como se determinar a ordem relativa que as ações foram executadas. Se, no entanto, *A* envia uma mensagem para *B* e *B,* subseqüentemente, envia uma mensagem para *C*, como mostra a Figura 9.14, existe uma ordenação total entre as ações de comunicação: o envio por *A* acontece antes do recebimento por *B*, que acontece antes do envio por *B*, que acontece antes do recebimento por *C*.

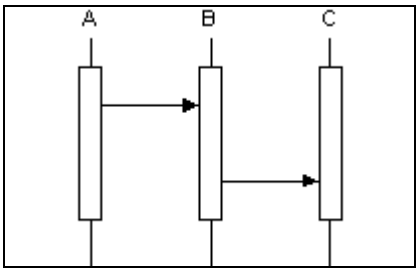

Figura 9.14 – Processos executando ações locais e de comunicação

Se existisse um relógio central único, ações de comunicação poderiam ser totalmente ordenadas associando-se a cada evento um *timestamp* único. Em particular, quando um processo envia uma mensagem, ele pode ler o relógio e anexar o valor deste à mensagem. Quando um processo recebe uma mensagem, ele pode ler o relógio e registrar o momento no qual cada evento de recepção ocorreu [And91].

Este relógio pode ser implementado como um relógio físico ou lógico (contador simples incrementado quando um evento acontece), não tendo esta decisão influência no ponto de interesse desta discussão.

9.9.2

A Obtenção do Pronome CLOCK a Partir do *Framework*

A implementação do pronome CLOCK a partir do *framework* apresentado é mostrada na Figura 9.15.

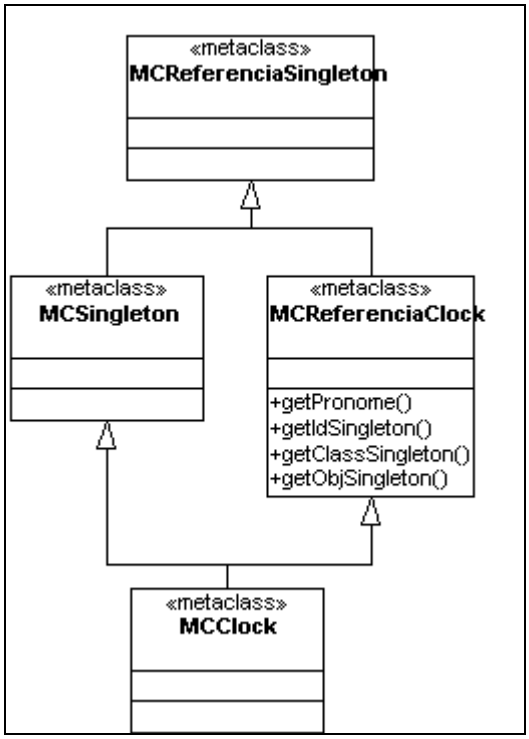

Figura 9.15 – Implementação do pronome CLOCK através do *framework*

Por ser um *singleton*, é necessária a criação de uma metaclasse que seja subclasse de *MCReferenciaSingleton* para que referências ao pronome CLOCK possam ser traduzidas. Esta metaclasse, denominada de *MCReferenciaClock*, redefine os mesmos métodos que a metaclasse *MCReferenciaPrincipal* definida anteriormente:

- *getPronome()* que retorna o pronome a que a metaclasse se refere, no caso, CLOCK.
- *getIdSingleton()* que retorna o modificador da classe que a identifica como *singleton*, no caso, CLOCK.

- *getClassSingleton()* que retorna a metaclasse que implementa o pronome, no caso, *MCClock*.
- *getObjSingleton()* que retorna o objeto singleton referenciado via pronome, no caso, *gClock*.

Além disso, é necessária a criação de uma metaclasse que faça as transformações necessárias na classe identificada como CLOCK. Esta metaclasse, denominada *MCClock* é equivalente à metaclasse *MCPrincipal* apresenta anteriormente e, da mesma forma que esta, não redefine método algum. Herda apenas, da metaclasse *MCReferenciaClock*, os métodos que definem o *singleton* e, da metaclasse *MCSingleton*, as transformações necessárias para que a classe, por ela transformada, represente um *singleton*.

9.9.3

O Pronome SIBLINGS

Já o pronome SIBLINGS, por sua vez, poderia ser utilizado, por exemplo, para otimizações que permitiriam a comunicação entre irmãos, sem que a mensagem passasse pelo pai.

Além da utilidade descrita acima, este pronome foi escolhido por ser de difícil categorização segundo os critérios impostos pelo *framework*. Por um lado, é um pronome que representa um conjunto de objetos: o que habilita sua metaclasse a herdar de *MCPronomeMulti*. Por outro lado, é um pronome que representa um conjunto de objetos que se relacionam de forma estática com o objeto corrente da execução: o que o habilita sua metaclasse a herdar de *MCPronRelAgregacao*. O aparente conflito não se confirma na realidade, já que todas as metaclasses que compõem o *framework* são interfaces, definindo apenas comportamentos. Sendo assim, não há problemas na metaclasse do pronome herdar de duas ou mais metaclasses do *framework*, herdando assim o comportamento de ambos. Conflitos de métodos devem ser resolvidos, redefinindo-se os métodos em questão e, nestas redefinições, comandar a execução dos métodos herdados de interesse. Esta solução é mostrada na Figura 9.16, com a metaclasse *MCSiblings*, responsável pela implementação do novo pronome, herdando de *MCPronRelAgregacao* e *MCPronomeMulti*. Toda a árvore hierárquica, com os métodos introduzidos a cada nível, é mostrada na Figura 9.16 para um melhor entendimento.

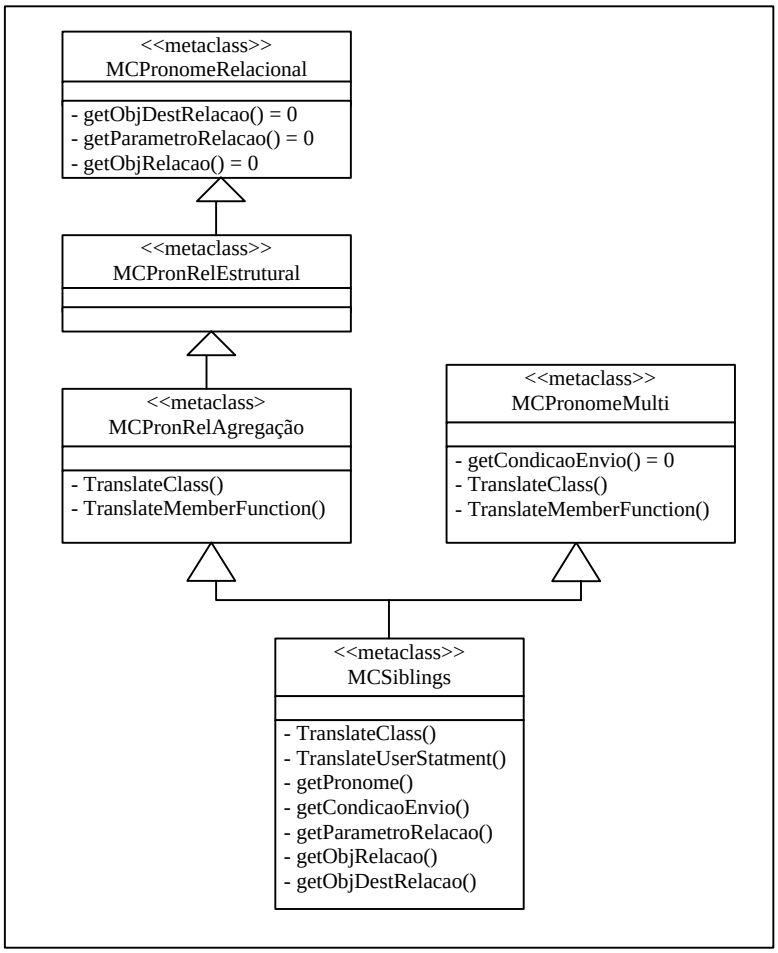

Figura 9.16 – Implementação do pronome SIBLINGS através do *framework*

A fim de garantir a execução correta do comportamento herdado, a metaclasse *MCSiblings* redefine os métodos:

- *TranslateClass()* que provoca a execução do mesmo método herdado de *MCPronRelAgregacao*, para que a relação de agregação se torne visível, e, subseqüentemente, do mesmo método de *MCPronomeMulti*, para que objetos possam ser colecionados.
- *TranslateUserStatment()* que provoca a execução do mesmo método herdado de *MCPronomeMulti*, para que o comando de envio seja transformado em uma coleção destes.

Além destes, a metaclasse implementa os seguintes métodos, deixados abstratos no *framework*:

- *getPronome()* que retorna o pronome a que a metaclasse se refere, no caso, SIBLINGS.

- *getCondicaoEnvio()*: Definido de forma abstrata na metaclasse *MCPronomeMulti*, este método retorna se a mensagem pode ser enviada ao objeto. No caso, a condição será satisfeita se o objeto para o qual a mensagem está para ser enviada for declarado pelo mesmo objeto que o objeto que está enviando a mensagem. Em outras palavras se os dois objetos envolvidos na operação possuírem o mesmo *parent*. Como esta informação está disponível nos objetos graças às transformações promovidas pela metaclasse *MCPronRelAgregacao*, a verificação pode ser executada sem problemas.
- *getObjRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que será representado pelo pronome no momento em que a relação definida por este é estabelecida. No caso, a relação entre irmãos é estabelecida no momento em que um objeto é declarado e é uma relação derivada da relação entre pai e filho. Manter a relação entre pai e filho, portanto, é suficiente para que a relação entre irmão esteja disponível. Desta forma, este método deve ser implementado da mesma forma que em *MCParent*, retornando a referência para o objeto corrente da execução para ser inserido com argumento extra na chamada aos construtores dos atributos.
- *getObjDestRelacao()*: Definido de forma abstrata na metaclasse *MCPronomeRelacional*, este método retorna o objeto que, no contexto do objeto corrente da execução, armazena a referência ao objeto utilizado para resolver o pronome, no caso, o atributo *m_pParent*.
- *getParametroRelacao()*: Definido de forma abstrata na metaclasse *MCPronRelFluxoExec*, este método retorna o parâmetro através do qual o objeto utilizado para resolver o pronome é passado para o contexto do objeto que está se utilizando deste, no caso, *pParent*.