3 O Padrão Arquitetural Reflective Blackboard

Neste capítulo será apresentado o padrão Reflective Blackboard[4] [52,53,54] como uma proposta para arquitetura de sistemas multi-agentes. O padrão oferece uma abordagem para o tratamento de propriedades multi-agentes desde a fase arquitetural de seu desenvolvimento, baseando-se em dois padrões bem conhecidos: Blackboard e Reflection [7,49]. Nas seções a seguir, o padrão proposto será apresentado seguindo a estrutura usual para a definição de padrões [17,7].

3.1 Exemplo de Motivação

Consideremos uma aplicação de Marketplace onde compradores e vendedores negociam produtos e serviços. Vendedores anunciam seu desejo de vender produtos ou serviços submetendo ofertas para o Marketplace. Por outro lado os compradores acessam este mercado para submeter lances de compra para produtos ou serviços desejados, e simultaneamente encontrar os vendedores que atendam às suas demandas. Uma vez que um comprador encontra um vendedor apropriado eles comunicam-se indiretamente através do próprio Marketplace realizando propostas e contrapropostas. Alguns compradores eventualmente se juntam para comprar produtos e minimizar custos. O Marketplace é aberto, isto é, agentes podem se juntar a ele ou deixá-lo quando quiserem, e inicialmente não conhecem seus parceiros de negócio. Da mesma forma agentes compradores e vendedores podem visitar Marketplaces diferentes localizados na rede a fim de atingir seus objetivos individuais.

O padrão arquitetural Blackboard é uma solução natural para o problema de Marketplaces e já foi largamente usado na prática para desenvolver Marketplaces sofisticados [1,2,27,59]. Blackboards são os lugares comuns onde transações comerciais são conduzidas e onde produtos e serviços são negociados. Blackboards distintos representam diferentes Marketplaces (Figura) e funcionam como uma interface para troca de mensagens, utilizada na comunicação e coordenação das atividades realizadas pelos agentes. Agentes compradores e vendedores são as fontes de conhecimento que cooperam e competem a fim de processar transações comerciais para seus donos. Os agentes lêem e escrevem nos blackboards mensagens que encapsulam, lances, ofertas, propostas ou contrapropostas. Cada servidor pode hospedar um

[4] O nome em inglês é utilizado aqui para manter coerência com a literatura que em geral não traduz os nomes de padrões com o objetivo de facilitar a comunicação entre os engenheiros de software que os utilizam

ou mais Marketplaces distintos. O componente de controle é o responsável pela gerência do Marketplace garantindo suas políticas de controle.

Em aplicações de Marketplace deste tipo uma das estratégias de controle deve lidar com o controle da comunicação na presença de mobilidade. Marketplaces distintos podem estar espalhados na rede e, conseqüentemente, agentes compradores e vendedores movem-se para diferentes servidores para encontrar ou vender os produ-

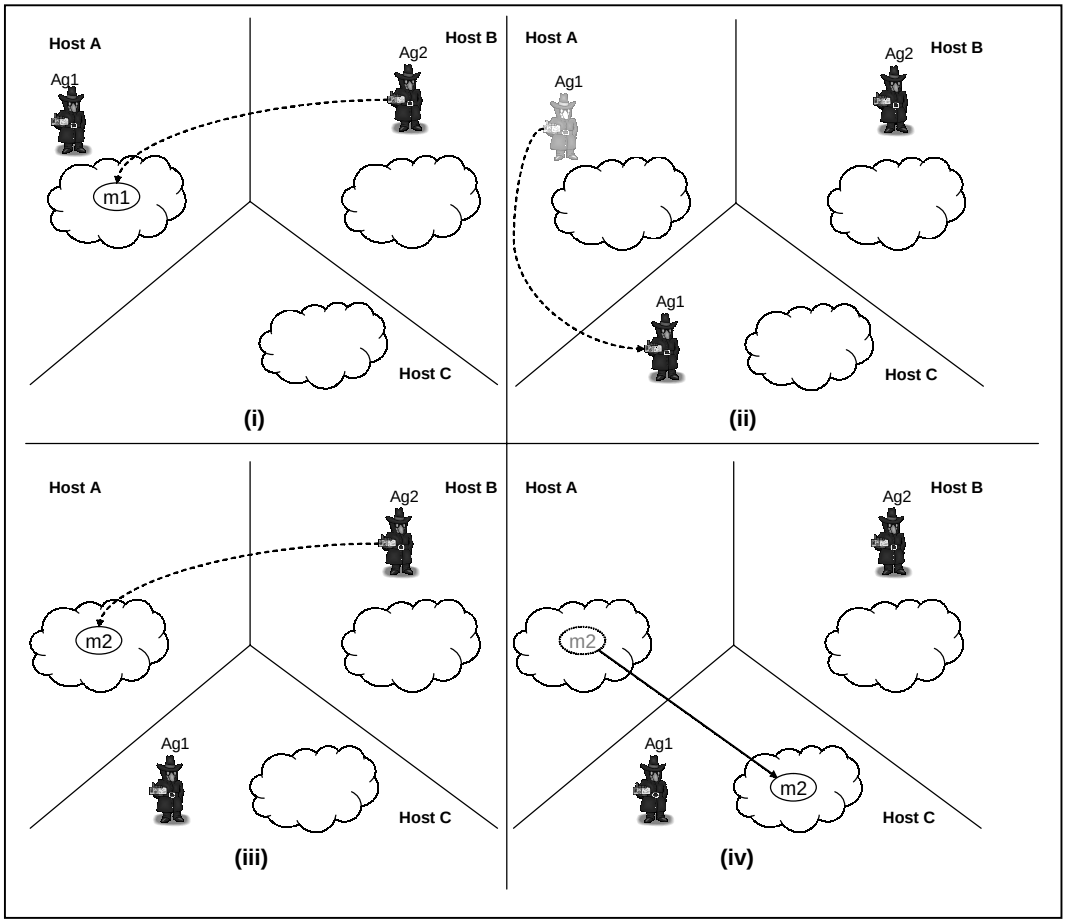

Figura 1 Exemplo de motivação ilustrado

tos ou serviços requisitados por seus donos.

Os agentes precisam visitar os servidores das próprias organizações para as quais trabalham de forma a armazenar dados relativos aos produtos que compraram ou venderam. Neste caso os servidores das empresas também contêm Blackboards que funcionam como interface de comunicação. Os Marketplaces e estes servidores das empresas dos agentes podem ser então os diferentes ambientes de execução dos agentes. Além disso, eles necessitam receber informações provenientes da própria empresa onde trabalham como, por exemplo, novas requisições de compras enviadas

para um agente comprador ou uma nova tabela de preços para um agente comprador. No início de seus trabalhos os agentes estão localizados nos servidores de suas empresas e todos os outros agentes que necessitam se comunicar com eles sabem sua localização e são capazes de lhes enviar mensagens de forma que eles possam lê-las e processá-las.

No entanto, uma vez que os agentes devem visitar deferentes Marketplaces espalhados na rede os servidores onde eles executam provavelmente irão mudar. Depois de se mover do servidor da sua empresa para um Marketplace e em seguida para outro Marketplace o agente deve continuar recebendo todas as mensagens que foram endereçadas a ele. Por outro lado, o agente que está enviando mensagens não sabe necessariamente que o agente destinatário mudou-se para um outro ambiente e pode continuar enviando mensagens para o ambiente de destino original. Uma vez que todas as mensagens devem alcançar o destinatário, elas devem ser encaminhadas para o novo ambiente do destinatário. Este processo de encaminhamento de mensagens deve também ser transparente para os agentes de forma que eles não precisem se preocupar com ele.

Este exemplo é ilustrado na Figura 1. Em (i) Ag1 e Ag2 são agentes que conhecem a localização uns dos outros. Ag2 pode então enviar mensagens diretamente ao blackboard que representa o ambiente de Ag1. Em (ii) Ag1 moveu-se para um ambiente diferente e em (iii) Ag2 enviou uma nova mensagem para o ambiente onde Ag1 estava. Desta forma deve existir uma estratégia de controle responsável pela composição da comunicação com a mobilidade dos agentes e que encaminhe a mensagem para o novo ambiente de Ag1. Esta estratégia de controle é ilustrada em (iv), e é denominada *estratégia de comunicação móvel*. Além desta estratégia marketplaces reais devem conter outras estratégias de controle para coordenar as atividades dos agentes, *gerenciando* as transações do Marketplace, garantindo a *segurança* destas transações, controlando a *persistência* dos agentes. A seguir a *estratégia de comunicação móvel* é utilizada para ilustrar a utilização do padrão proposto.

3.2 Problema

O padrão arquitetural blackboard já foi largamente utilizado para lidar com problemas que possuam soluções não-determinísticas [7]. No que se refere a sistemas multi-agentes [42], este padrão arquitetural é largamente aceito para implementar a comunicação [28,47] e coordenação[9]. Pesquisas recentes também conseguiram resultados positivos na utilização de blackboards na implementação da mobilidade e persistência dos agentes [51].

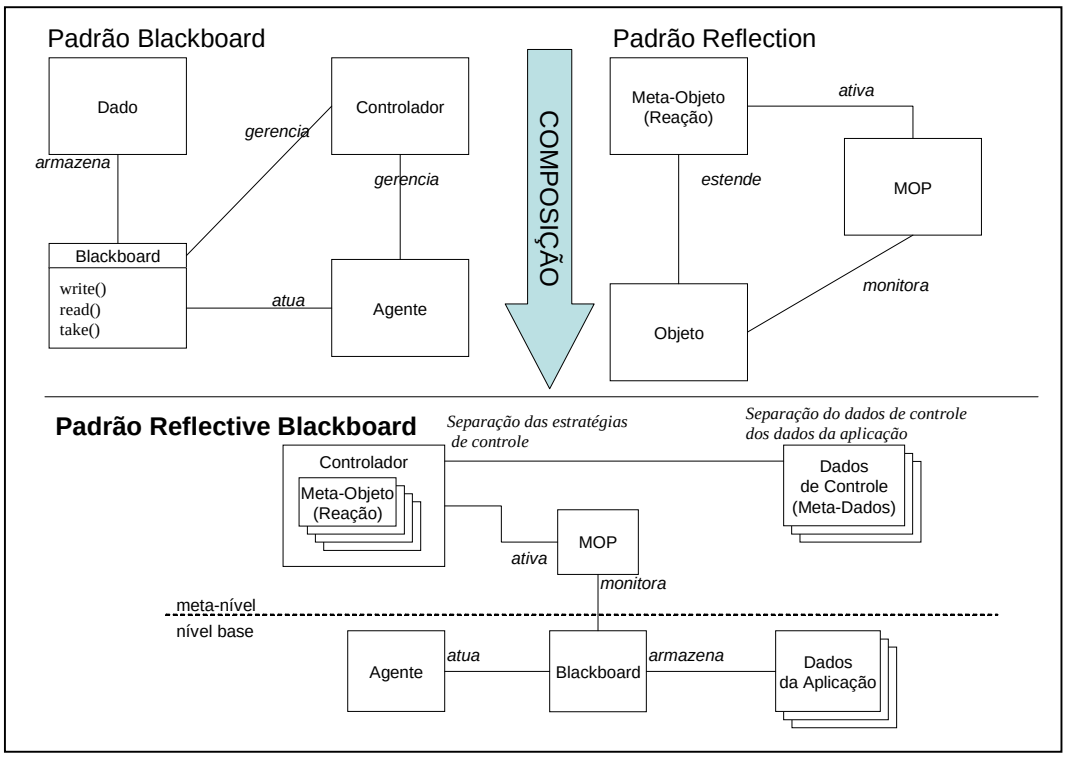

Figura 2 A Composição dos Padrões Blackboard e Reflection

Como descrito anteriormente, a estrutura do padrão arquitetural Blackboard é dividida em três subsistemas: o próprio blackboard, um grupo de fontes de conhecimento (ou agentes), e um componente de controle. O lado superior esquerdo da Figura 2 mostra os componentes do padrão Blackboard. O blackboard funciona como uma estrutura de armazenamento central de dados do sistema multi-agente. Os dados armazenados no blackboard podem ser dados da aplicação (como mensagens, informações, etc.) e dados de controle (ou meta-dados). O blackboard provê uma interface que possibilita que todos os agentes leiam, removam ou escrevam dados nele. Os agentes utilizam estas operações para se comunicar indiretamente uns com os outros e para coordenar suas atividades. Os agentes utilizam efetuadores para mo-

dificar o estado do blackboard e sensores para perceber mudanças neste estado (por motivo de simplicidade os sensores e efetuadores não são mostrados na Figura 2).

Embora a estrutura do blackboard tenha se mostrado uma boa interface de comunicação entre agentes de software, falta a ela uma especificação mais precisa do sue componente de controle. O componente de controle proposto na definição do padrão blackboard [7,49] é simplesmente definido como um "loop" que monitora as mudanças no blackboard e decide qual a próxima ação a ser executada. Entretanto sistemas multi-agentes reais incluem um conjunto de políticas de controle dependentes e independentes de aplicação e que são utilizadas no controle das diversas propriedades inter-agentes como mobilidade, comunicação, coordenação e persistência. O problema é que o principal ponto fraco do padrão Blackboard é a dificuldade em lidar com múltiplas estratégias de controle existentes em sistemas multi-agentes [7]. Isto porque o padrão não oferece suporte arquitetural para o tratamento de várias estratégias de controle simultaneamente. Finalmente o padrão Blackboard não provê nenhuma forma de separação explícita entre os dados da aplicação e os dados de controle, ficando o componente blackboard responsável pelo armazenamento de ambos. No entanto, o acesso a informações de controle deve ser restrito para alguns agentes.

➔ No que se refere ao exemplo de motivação apresentado anteriormente, o problema descrito acima está relacionado às dificuldades associadas à definição da *estratégia de comunicação móvel* de forma transparente aos agentes. Durante os processos de compra e venda, os agentes estão se movendo entre ambientes distintos (Marketplaces e ambientes internos de suas empresas) e não devem precisar controlar a localização de seus parceiros de comunicação. Além disso a utilização do padrão Blackboard mistura dados de controle (por exemplo, dados informando a localização dos agentes) e dados da aplicação (por exemplo, lances e ofertas).

Existem algumas forças[5]associadas a este problema:

- As políticas de controle para algumas propriedades do sistema em geral estão situadas em ambientes de execução distintos. Desta forma, a arquitetura de software deve ser suficientemente flexível para possibilitar a adaptação a mu-

[5] Do inglês *forces*

danças que venham a ocorrer nestes ambientes, bem como a mudanças nos requisitos da aplicação relacionados às estratégias de controle;

- A arquitetura de um sistema multi-agente deve ter uma alta capacidade de modificação, isto é, facilidade de incorporação de mudanças, uma vez que a natureza de uma determinada mudança foi determinada. Além disso, a arquitetura de software deve ser capaz de oferecer suporte a alteração, adição ou remoção de estratégias de controle em tempo de execução;

- A arquitetura do sistema multi-agente deve guiar seus projetistas e programadores na reutilização de estratégias em diferentes aplicações, nas quais múltiplas estratégias de controle são utilizadas.

3.3 Solução

Para solucionar o problema apresentado na seção anterior, propõe-se a composição do padrão arquitetural Blackboard com o padrão arquitetural Reflection [7]. O padrão arquitetural Reflection provê mecanismos para a mudança de estrutura e comportamento de um sistema dinamicamente [7]. O lado superior esquerdo da Figura 2 ilustra este padrão, que divide sistemas de software em dois níveis: o nível base e o meta-nível. O nível base contém a lógica da aplicação, implementada por seus agentes de software além dos dados da aplicação. Já o meta-nível, é composto por meta-objetos que encapsulam dados e comportamento. Os dados dos meta-objetos são chamados de meta-dados (ou dados de controle) e representam informações sobre os dados da aplicação armazenados no nível base. O comportamento associado aos meta-objetos pode ser visto como as reações a mudanças executadas no nível base [38]. A interface entre o nível-base e o meta-nível é realizada por um componente separado denominado protocolo de meta-objetos, ou MOP (Seção 2.3). O MOP é o responsável pelo redirecionamento do fluxo de controle do nível base para o meta-nível em pontos específicos de sistemas de software reflexivos.

A composição proposta resulta em três mudanças principais no padrão Blackboard: (i) o componente de controle é movido para o meta-nível enquanto o blackboard e os agentes e dados da aplicação são situados no nível base; (ii) o MOP intercepta transparentemente as operações efetuadas no blackboard; (iii) a semântica do componente de controle é dividida em meta-objetos distintos (meta-dados e reações). De acordo com estas mudanças, dados escritos no blackboard podem ser associados a meta-objetos localizados no meta-nível do sistema. Os meta-objetos comportam-se

como regras que dizem como o sistema deve se comportar quando operações específicas são efetuadas no blackboard. Por exemplo, um meta-objeto pode especificar que sempre que um dado específico for retirado do blackboard, o agente que o escreveu será notificado a respeito desta remoção. Desta forma, o controle da comunicação dos agentes, permite a inclusão transparente de propriedades inter-agentes no meta-nível.

➔ A solução apresentada acima pode ser aplicada ao exemplo de motivação apresentado na seção 3.1 através da implementação da *estratégia de comunicação móvel* no componente de controle do meta-nível de forma separada dos agentes que estão localizados no nível base. Isto é realizado através da criação de meta-objetos que especificam que um ponteiro de encaminhamento de mensagens é criado sempre que um agente move-se de um ambiente para outro. Estes ponteiros são então usados para o encaminhamento de mensagens endereçadas a agentes de seus ambientes originais, para os seus novos ambientes de destino.

Cada ponteiro de encaminhamento é também implementado através de uma regra, escrita no meta-nível (meta-dado), que especifica que toda mensagem endereçada ao agente ao qual ela está relacionada é encaminhada para o seu ambiente de destino. Esta estratégia de controle é baseada na mesma idéia implementada no protocolo Mobile IP [44], onde dados endereçados para dispositivos móveis são sempre endereçados para seu ambiente (sub-rede) original (*home-agent*), que é o responsável pelo encaminhamento dos dados ao ambiente (sub-rede) onde o dispositivo está realmente localizado. Mais detalhes acerca da estrutura e dinâmica da implementação desta estratégia de controle serão apresentados nas seções seguintes.

3.4 Estrutura

A estrutura do padrão arquitetural Reflective Blackboard pode ser dividida, assim como no padrão Blackboard, em três subsistemas distintos: o próprio blackboard, um grupo de fontes de conhecimento e o controlador (subsistema de controle). A Figura 3 ilustra, através da utilização de um diagrama de componentes UML [4], estes subsistemas, seus principais componentes bem como suas dependências.

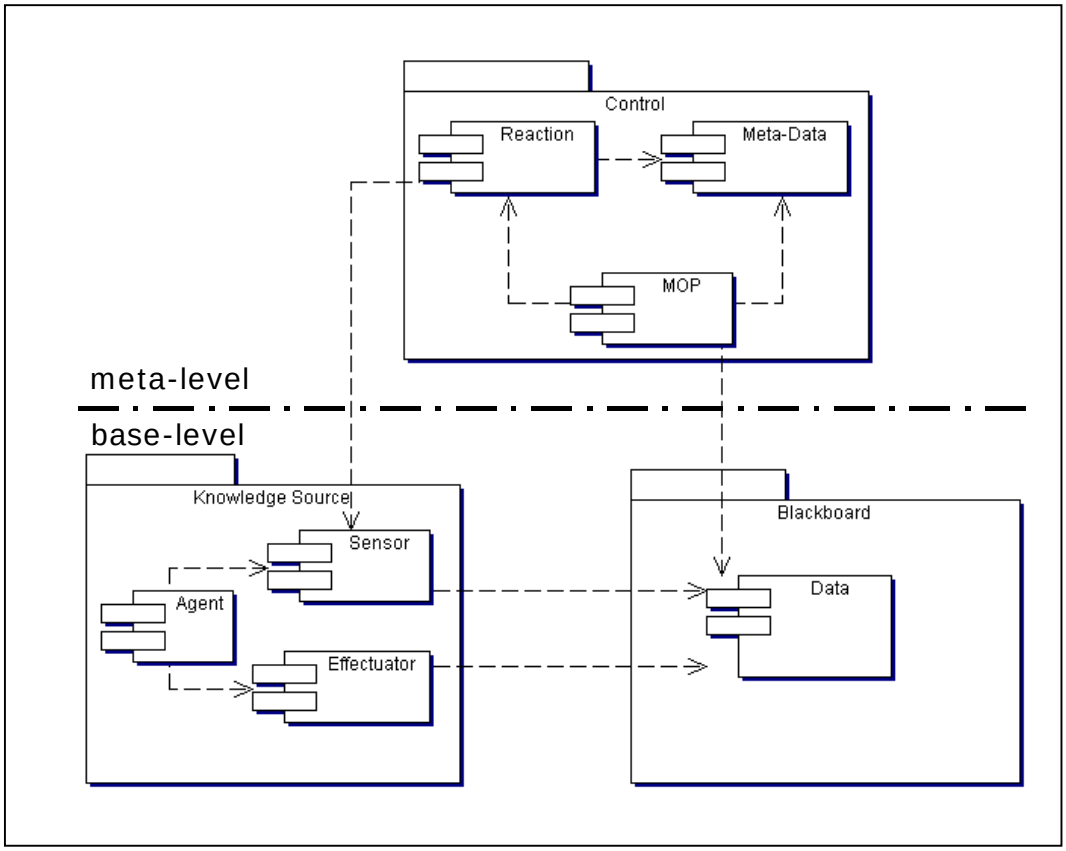

Figura 3 Estrutura do padrão Reflective Blackboard

O comportamento do subsistema que contém o blackboard é praticamente o mesmo que o proposto no padrão Blackboard. Ele funciona como uma unidade central de armazenamento de dados onde dados podem ser escritos, lidos ou retirados por diferentes agentes. A principal diferença agora, é que cada dado específico pode estar associado a um meta-objeto armazenado no subsistema de controle. Na verdade, através do uso da reflexão, a existência do componente de controle é transparente para a lógica e os dados básicos do sistema.

O subsistema de controlador é composto pelo protocolo de meta-objeto que em conjunto com uma coleção de meta-objetos implementam as estratégias de controle presentes no sistema multi-agente. Os meta-objetos são compostos por dados (meta-dados) e são os responsáveis por associar comportamento específico a operações executados sobre dados específicos. Estes meta-objetos podem então modificar de forma transparente o comportamento do sistema multi-agente.

Diferentes agentes podem atuar no blackboard através de seus sensores e atuadores que podem respectivamente sentir e realizar mudanças no blackboard que por

sua vez pode ser considerado seu ambiente. Os agentes não se comunicam diretamente. Na verdade eles apenas lêem e escrevem dados no blackboard.

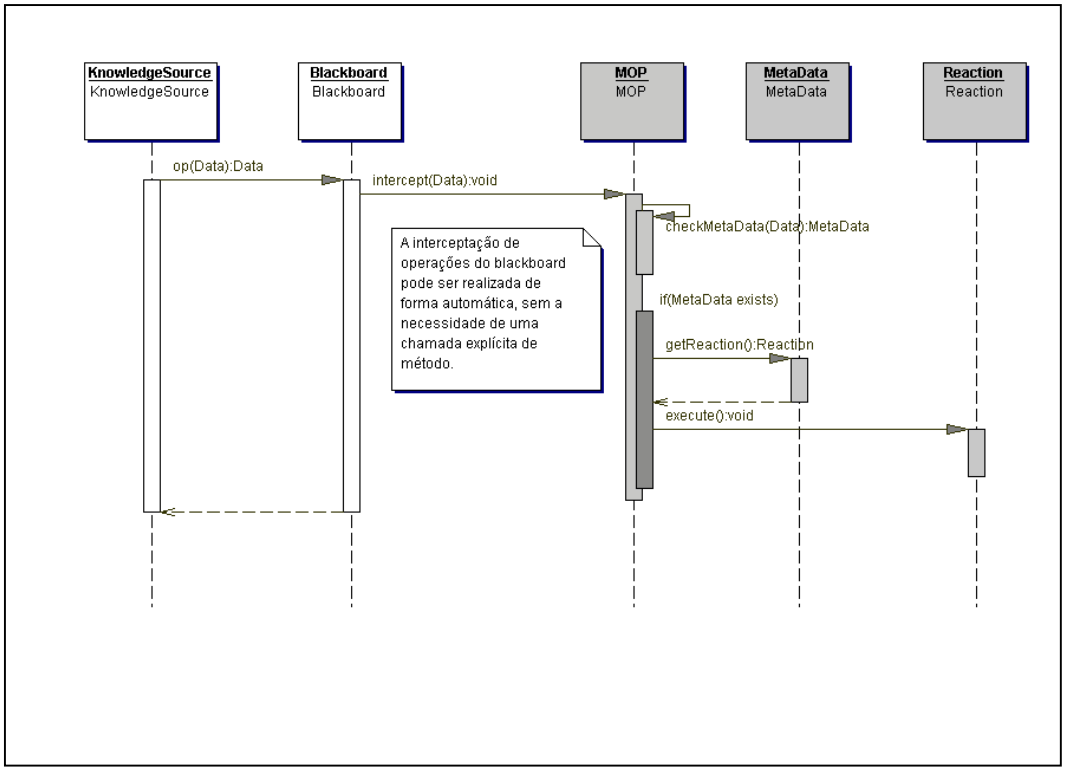

Figura 4 Dinâmica geral do padrão Reflective Blackboard

Sempre que um agente executa alguma operação sobre um dado específico armazenado no blackboard, o componente MOP verifica se existe algum meta-objeto associado à operação como um todo. Em caso positivo, ele executa a reação associada ao meta-objeto, isto é, seu comportamento. As reações podem acessar o próprio blackboard escrevendo, lendo e removendo dados. Desta forma a semântica de uma operação em um blackboard reflexivo é na verdade resultado das reações associadas a ela.

Meta-objetos também podem existir no subsistema de controle sem que de fato existam dados correspondentes no blackboard. Desta forma, o sistema multi-agente pode associar reações a dados que fazem parte do vocabulário do sistema [7] e que provavelmente serão escritos no blackboard em tempo de execução.

3.5 Dinâmica

A reflexão computacional é utilizada para interceptar e modificar os efeitos das operações realizadas no blackboard. Do ponto de vista dos agentes de software da aplicação, a reflexão é transparente: quando um agente escreve um dado no blackbo-

ard ele não tem conhecimento que esta operação de escrita será interceptada e redirecionada para o meta-nível. O cenário a seguir apresenta a dinâmica do comportamento genérico de uma arquitetura baseada no padrão Reflective Blackboard. A Figura 4 ilustra visualmente este cenário através de um diagrama de seqüência UML [4]. Na figura, os objetos em cinza representam componentes situados no meta-nível.

1. Uma fonte de conhecimento (ou agente) realiza uma operação sobre o blackboard (escrita, por exemplo);
2. Esta operação é interceptada pelo MOP que realizará, se especificado, atividades de controle sobre a operação executada;
3. O MOP verifica a existência de meta-objetos associados à operação executada. Se a operação executada for de escrita, o meta-objeto procurado será aquele relacionado ao dado escrito. Por outro lado, se a operação executada for de leitura ou retirada de dados, o meta-objeto procurado será relacionado ao dado lido ou removido do blackboard;
4. Se o meta-objeto procurado existir, seu comportamento, isto é, a reação associada a ele é executada. Os efeitos possíveis das reações devem ser especificados pela implementação do padrão (Seção 3.9). Dependendo da implementação, a reação pode modificar dados do blackboard ou ativar outros agentes, dentre outros tipos de atividades.

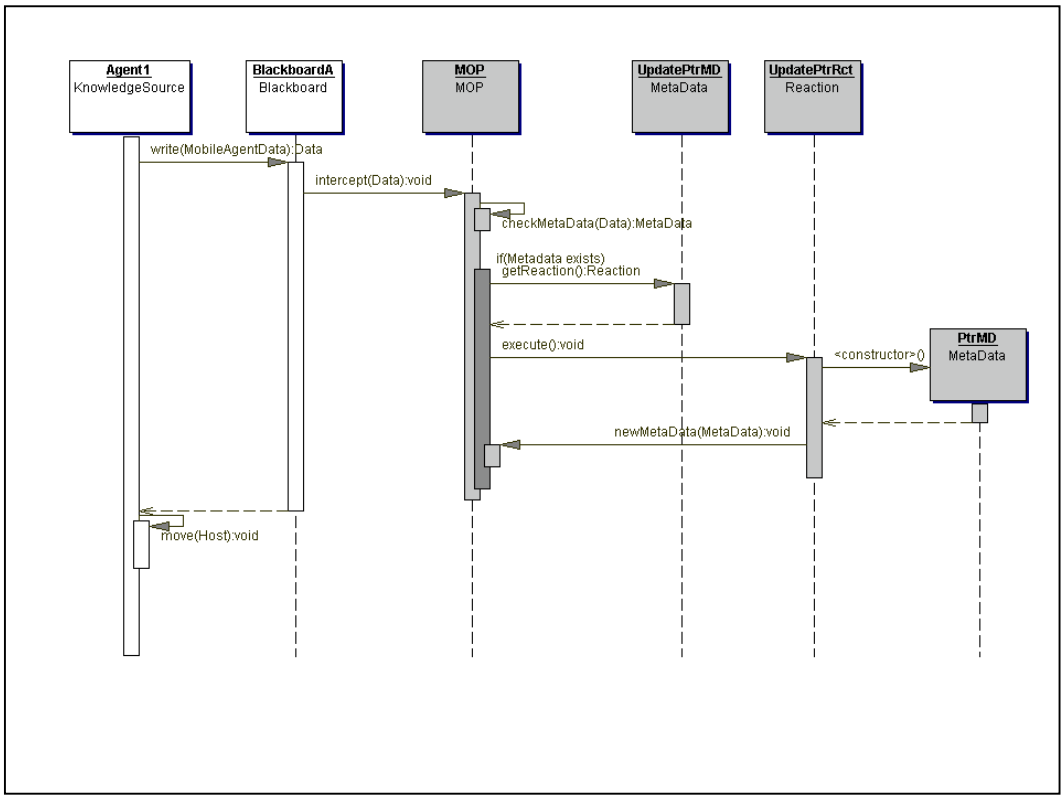

Figura 5 Dinâmica da estratégia de atualização de ponteiros

➔ No que se refere ao exemplo de motivação apresentado na seção 3.1, o cenário da dinâmica do padrão pode ser especializado em dois outros cenários. O primeiro deles, ilustrado através de um diagrama de seqüência na Figura 5, refere-se à atualização do ponteiro de encaminhamento de mensagens, enquanto o segundo está relacionado ao processo de encaminhamento das mensagens em si. O cenário de atualização de ponteiros é iniciado quando o Agente 1 decide se mover para outro ambiente e notifica seu ambiente, que é representado pelo Blackboard A, que ele vai partir. Esta notificação é representada pelo dado MobileAgentData que é escrito no blackboard. A operação de escrita é interceptada pelo MOP que verifica a existência de algum meta-dado associado a ela. Se tal meta-dado existir (UpdatePtrMetaData), ele será de fato o responsável, em conjunto com sua reação associada (UpdatePtrRct), pela execução da estratégia atualização de ponteiros, conforme especificado na seção 3.3. Desta forma, a reação associada é a responsável pela

criação de um novo ponteiro de mensagens (PtrMD) e, em seguida, por notificar o MOP que um novo meta-dado existe. Neste ponto a operação do meta-nível termina e o agente pode executar sua transferência para o ambiente de destino.

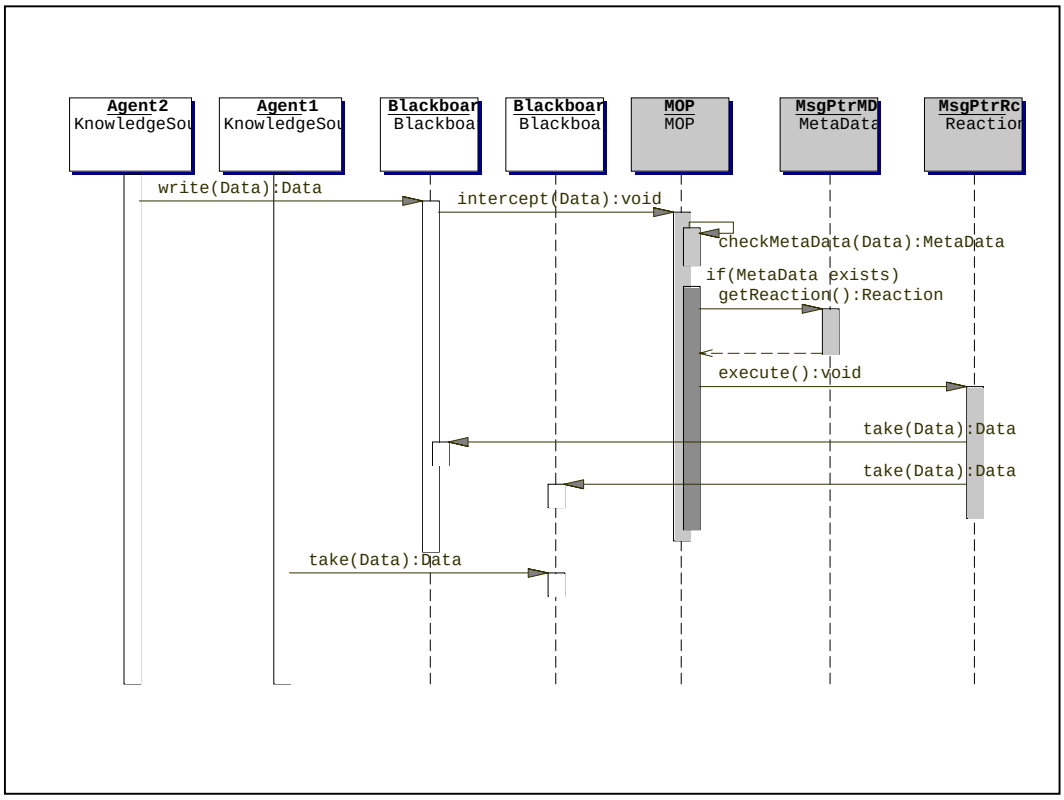

Figura 6 Dinâmica da estratégia de encaminhamento de mensagens

Após a atualização do ponteiro de encaminhamento mensagens, qualquer mensagem endereçada ao Agente 1 será encaminhada ao seu novo ambiente. No cenário do exemplo de motivação, o Agente 2 envia uma mensagem endereçada ao Agente 1 para o Blackboard A. Este processo é representado pela operação de escrita realizada pelo Agente 2 sobre o Blackboard A.

Esta operação é interceptada pelo MOP que verificará a existência de meta-dados associados à mensagem escrita. Tal meta-dado é na verdade o ponteiro de encaminhamento (PtrMD) que foi criado no cenário apresentado acima. Este meta-dado é o responsável pela associação da reação (MsgPtrRct) a mensagens endereçadas ao Agente 1. Esta reação, por sua vez, tem a responsabilidade de encaminhar a mensagem para o ambiente

onde o Agente 1 realmente está. A execução da reação vai então remover a mensagem do Blackboard A e escrevê-la no Blackboard B, que representa o ambiente onde o Agente 1 está localizado. Após a execução da reação a execução no meta-nível termina e o Agente 1 pode ler a mensagem enviada para ele. Este cenário é ilustrado através de um diagrama de seqüência na Figura 6.

3.6 Conseqüências

A utilização do padrão Reflective Blackboard possui os seguintes benefícios:

Tratamento separado das estratégias de controle. Implementar um sistema multi-agente utilizando uma arquitetura baseada no padrão Reflective Blackboard promove a separação das estratégias de controle do sistema das suas funcionalidades básicas. Além disso, a utilização do padrão possibilita a separação dos dados da aplicação dos dados de controle. Este tipo de separação leva a um melhor tratamento de diferentes aspectos de controle do sistema. Adicionalmente as diferentes políticas de controle podem ser compostas no meta-nível, de forma independente em relação à aplicação. Isto é particularmente importante quando são considerados sistemas multi-agentes de larga escala, uma vez que estes sistemas tendem a tornar-se sociedades organizadas de agentes, onde diferentes tipos de agentes cooperam com o objetivo de atingir um objetivo comum. Nestas sociedades as regras de controle e coordenação são em geral, bastante complexas e difíceis de gerenciar se estiverem misturadas com os dados básicos do sistema.

Tratamento separado de requisitos não funcionais. Além de seu uso para implementar estratégias de controle, o meta-nível do sistema multi-agente pode ser utilizado para implementar outros requisitos não funcionais como tolerância a falhas e segurança [50]. Esta abordagem é interessante porque é possível tratar estes requisitos separadamente dos dados básicos do sistema.

Maior facilidade de reutilização e manutenção. A reutilização e manutenção do sistema são melhoradas através do tratamento separado das estratégias de controle e requisitos não funcionais. Uma vez que o código dos agentes não está misturado com as invocações explícitas das estratégias de controle. O MOP é o responsável por estas invocações que são feitas de forma transparente para os agentes. Conseqüentemente a facilidade de leitura do sistema é melhorada, o que por sua vez

provê uma melhor reutilização e manutenção. Através desta separação de responsabilidades, a reutilização pode ser alcançada em diferentes níveis: (i) o nível dos agentes, (ii) o nível das estratégias de controle e (iii) o nível das propriedades ou funcionalidades disponibilizadas pelo sistema.

Modularidade. Embora a arquitetura resultante da aplicação do padrão Reflective Blackboard tenha naturalmente um número maior de componentes arquiteturais, ela melhora a modularidade do sistema, uma vez que o encapsulamento de informações é alcançado através destes componentes. Além disso, uma vez que a arquitetura está construída seus componentes podem ser reutilizados em diferentes aplicações multi-agentes.

Por outro lado, a utilização do padrão Reflective Blackboard possui os seguintes problemas:

Sobrecarga na performance. Software reflexivo é, em geral, mais lento que software não reflexivo. Isto se deve à existência do processamento adicional necessário para a busca de meta-dados no meta-nível e a subseqüente execução de reações

Acoplamento. Um outro ponto a respeito de sistemas baseados em blackboards é relacionado à forte dependência existente entre os agentes e a estrutura e interface do blackboard. Se engenheiros de software estiverem construindo uma aplicação baseada em agentes e desejarem modularizar as bases de conhecimento, então blackboards podem ser encarados como bons mecanismos de comunicação. Por outro lado, se são necessários ambientes onde vários agentes, com estruturas distintas e sem conhecimento de um blackboard centralizado, podem trabalhar em conjunto, são necessárias interfaces mais formais [39]. Entretanto, este problema pode ser superado através da utilização das arquiteturas baseadas em blackboards à la Linda [10,16] como JavaSpaces [14] e TSpaces [35].

3.7 Usos Conhecidos

TSpaces. TSpaces [35] é uma implementação bastante conhecida do padrão arquitetural Reflective Blackboard. TSpaces é uma arquitetura baseada em blackboards à la Linda [ref] para comunicações através da rede e que possui funcionalidades de bancos de dados. Esta implementação provê serviços de comunicação de grupo, serviços de banco de dados e serviços de notificação de eventos. A máquina de notificação de eventos de TSpaces é implementada como um Reflective Blackboard. As reações de TSpaces são chamadas de objetos de *callback* e seus meta-dados possuem

informações relativas à operação executada e ao dado monitorado pela máquina de notificação de eventos. Na implementação de sistemas multi-agentes os serviços de monitoração de eventos de TSpaces são usados para implementar as estratégias de controle do sistema. Desta forma os sistemas multi-agentes podem, por exemplo, utilizar esta funcionalidade para notificar os agentes que dados importantes foram escritos no blackboard [47].

MARS. Mars [9] é uma outra implementação do padrão Reflective Blackboard. Ele define blackboards à la Linda que podem ser programados para reagir com ações específicas aos acessos efetuados pelos agentes. MARS é implementado através da utilização da tecnologia de JavaSpaces [14] e foi criado com o objetivo de facilitar a tarefa de definição e implementação de estratégias de coordenação em sistemas baseados em agentes móveis. Os meta-dados de MARS são denominados meta-tuplas e contêm informações sobre o agente que realiza uma operação específica sobre um dado específico. O MOP é implementado através da utilização de buscas baseadas em casamentos de padrões em um blackboard de meta-nível, onde as meta-tuplas são armazenadas.

TuCSoN. TuCSoN [43] é um modelo de coordenação que pode ser visto como uma implementação do padrão Reflective Blackboard. Este modelo é baseado na noção de centros de tuplas que são, na verdade, blackboards programáveis. Estes centros de tuplas são programados através da associação de reações a dados e operações específicas. As reações são criadas através de uma linguagem de especificação proprietária e são tratadas de forma separada dos dados e lógica básicos da aplicação multi-agente.

T-Rex. T-Rex [51,45] também é uma implementação do padrão arquitetural Reflective Blackboard. T-Rex implementa um modelo reflexivo (MOP e meta-dados) que é bastante semelhante ao implementado por MARS. Por outro lado, enquanto MARS utiliza a estrutura reflexiva apenas para implementar a coordenação dos agentes, T-Rex também a utiliza na implementação da mobilidade, comunicação e persistência dos agentes, além de poder ser utilizado no tratamento de outros requisitos não funcionais como segurança e tolerância a falhas. No Capítulo 4 a estrutura e implementação de T-Rex serão apresentados em detalhes.

3.8 Padrões Relacionados

O padrão Reflective Blackboard é a base para a composição de múltiplos padrões conhecidos durante o refinamento de arquiteturas de software multi-agente. A seguir será mostrado como o padrão proposto está conectado a outros padrões, com quais outros padrões ele pode ser combinado e refinado, quais variantes ele pode incorporar bem como quais outros padrões podem ser utilizados para resolver o mesmo problema de outras formas. O relacionamento entre estes padrões é apresentado na Figura 7.

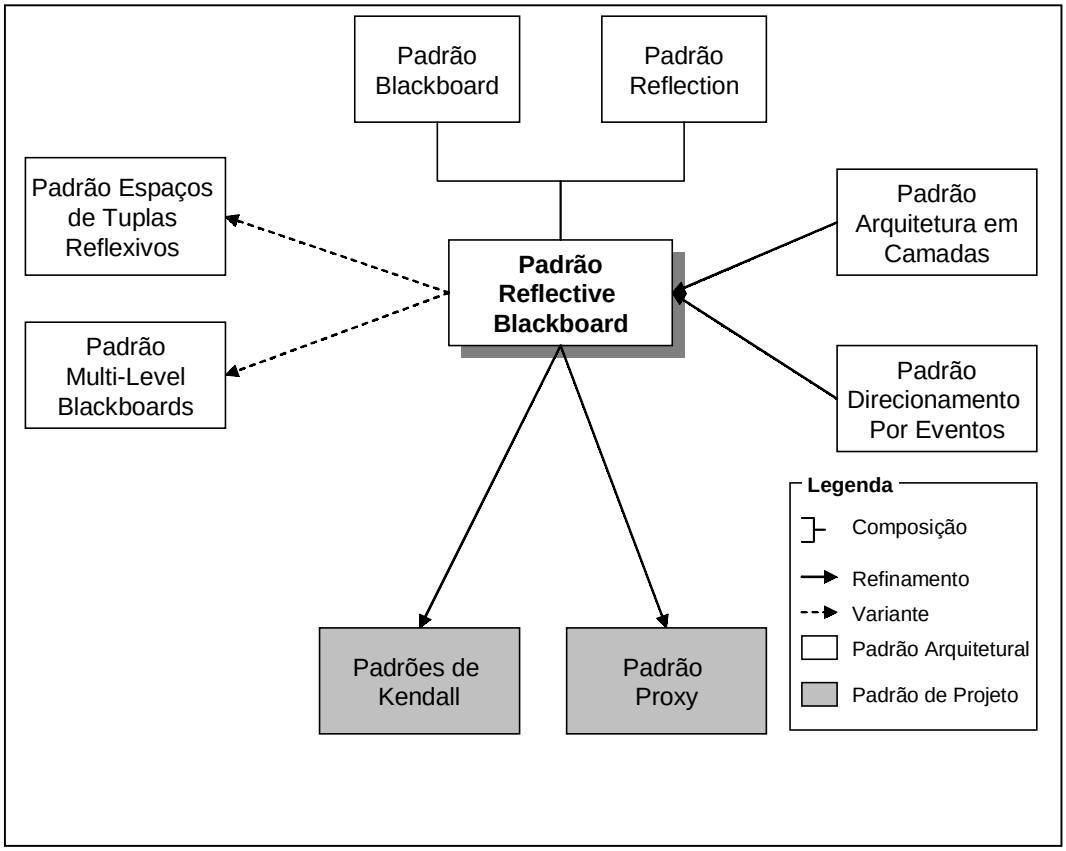

Figura 7 - Padrões Relacionados

Arquitetura Interna dos Agentes. A arquitetura de um agente de software é bastante complexa uma vez que ela encapsula um estado mental e diferentes funcionalidades comportamentais (propriedades intra-agentes) como autonomia, adaptação, colaboração e aprendizagem. Kendall et al. [33] examinam padrões de projetos para agentes através de uma arquitetura em camadas. Garcia et al. propõem um método orientado a aspectos para estruturar o projeto interno dos agentes de software [18], e o comparam com um método orientado a objetos. A composição desta abordagem orientada a aspectos com uma implementação de arquitetura baseada no padrão Reflective Blackboard foi proposta em [50].

Espaços de Tuplas Reflexivos. Nesta variante, o componente que representa o blackboard do padrão proposto é estruturado como um espaço de tuplas (Seção 2.2). Neste tipo de variante os componentes do meta-nível são estruturados como tuplas que são armazenadas em espaços de meta-nível ou meta-espaços. MARS, T-Rex e TuCSoN são alguns dos usos conhecidos desta variante do padrão. No Capítulo 1 é apresentada a implementação de T-Rex que é uma implementação desta variante do padrão Reflective Blackboard.

Blackboards Direcionados por Eventos. O padrão Reflective Blackboard pode ser combinado com o padrão Direcionamento por Eventos (Event-Driven) [14]. Um modelo de eventos é utilizado para sinalizar a ocorrência de mudanças no blackboard e para notificar os agentes a respeito destas modificações. O meta-nível poderia utilizar este mecanismo para ativar estratégias de controle baseado em eventos específicos.

Organização do Meta-Nível. A estrutura básica de uma arquitetura reflexiva é bastante parecida com o padrão de Arquitetura em Camadas (Layers) [7,49]. O nível base e o meta-nível são duas camadas, cada uma com seus próprios componentes. No entanto, ao contrário do que acontece em Arquiteturas em Camadas, existem dependências mútuas entre os dois níveis. Outro ponto importante refere-se à possibilidade de utilização do padrão Arquitetura em Camadas para refinar o meta-nível em múltiplos meta-níveis, criando a variante denominada Multi-Level Blackboards. Esta variante é composta por uma torre de meta-níveis, onde cada nível é capaz de incorporar diferentes níveis de controle.

Blackboards Distribuídos. Os meta-níveis de diferentes blackboards podem precisar se comunicar com o objetivo de implementar uma dada propriedade do nível do sistema. O padrão de projeto Proxy [17] é uma solução para esta comunicação remota. Este padrão de projeto provê uma referência para um objeto com objetivo de controlar acesso a este. O padrão Proxy é aplicável sempre que se necessita de uma maneira de referenciar um objeto que seja mais sofisticada e versátil que um simples ponteiro.

3.9 Questões de Implementação

Além das questões de implementação relativas ao padrão arquitetural Blackboard [7], as seguintes questões devem ser consideradas durante a implementação do padrão Reflective Blackboard:

Como os meta-dados serão estruturados? Uma questão fundamental com relação ao padrão é relativa à decisão sobre quais informações estarão disponíveis nos meta-dados. Na verdade, esta decisão depende da especificidade do componente de controle implementado pelo sistema multi-agente. Em geral os meta-dados possuem referências aos dados do nível base, identificação de agentes e operação realizada no blackboard [9,51]. No entanto, eles também podem possuir dados específicos à aplicação como, por exemplo, o grau de certeza de uma hipótese [7].

Como o protocolo de meta-objeto será implementado? Outra questão importante é a decisão sobre como o protocolo de meta-objeto irá agir sobre os dados do nível base e meta-nível. Uma possibilidade para esta implementação é a utilização de outro blackboard para armazenar os meta-dados, de forma que o MOP use operações padrão do próprio blackboard para escrever e buscar meta-dados. A utilização de espaços de tupla à la Linda [16,10] neste blackboard de meta-nível pode ajudar nesta tarefa uma vez que a utilização dos mecanismos de casamento de padrões pode facilitar as buscas de meta-dados. Uma outra abordagem é a utilização de arquiteturas reflexivas nativas como, por exemplo, Guaraná [41] para implementar as atividades do meta-nível.

Quais componentes terão acesso ao meta-nível? É bastante importante estabelecer políticas de controle de acesso ao meta-nível do sistema, onde suas estratégias de controle serão implementadas. Pode ser definido que os meta-dados serão escritos apenas durante a fase de implementação, permanecendo sem modificação em tempo de execução. Por outro lado, o administrador do sistema e até os próprios agentes poderiam inserir meta-dados em tempo de execução. A adoção desta abordagem pode requerer cuidados especiais no tratamento do meta-nível e a implementação de um meta-meta-nível pode ser útil para impedir mudanças maléficas ao componente de controle.

Quais componentes as reações poderão acessar e modificar? As reações são componentes que especificam mudanças no comportamento normal do sistema multi-agente. Desta forma, é uma decisão importante definir quais outros componentes do sistema elas poderão acessar e modificar. Em geral apenas através da modificação do blackboard é possível modificar o comportamento do sistema multi-agente como um todo, uma vez que seriam geradas alterações no ambiente dos agentes e agindo no seus sensores. Por outro lado, para implementar algumas estratégias de controle pode ser necessário permitir que as reações acessem outros componentes

específicos do sistema como, por exemplo, bancos de dados, sistemas de arquivos ou a rede.

Quais informações do nível base serão reificadas no meta-nível? Na execução das reações podem ser levadas em consideração informações relativas ao nível base, no momento da interceptação efetuada pelo MOP. Estas informações são materializadas no meta-nível e podem afetar a execução de uma reação específica. A decisão de quais e que tipo de informação reificar faz parte da implementação do padrão. Tipicamente são reficadas informações relativas ao dado afetado pela interceptação e à entidade que a causou [50,51,9,43]. No entanto, outras informações como estado geral do blackboard ou fase de execução do sistema, podem também ser materializadas no meta-nível.