2
Trabalhos Relacionados

A análise dos trabalhos relacionados a esta tese cobre as propostas que definem os *frameworks* conceituais e as linguagens de modelagem para SMAs que estendem a UML. Algumas metodologias também foram analisadas pois definem, usam ou estendem linguagens de modelagem com base em UML. É importante ressaltar que não temos a intenção de avaliar as metodologias, apenas as linguagens de modelagem incorporadas nelas.

A fim de avaliar os *frameworks* conceituais e as linguagens de modelagem apresentados na literatura, propomos algumas questões. As questões foram definidas com base em problemas expressos na Seção 1.1. Os *frameworks* e as linguagens foram descritos enfatizando suas características enquanto as questões eram respondidas. Também foram descritos os pontos fracos dos *frameworks* e das linguagens de acordo com as questões propostas.

Depois de descrever as questões usadas para avaliar os *frameworks* conceituais e apresentar um conjunto de três *frameworks* conceituais para SMAs, apresentamos uma visão geral do TAO (Seção 2.2). Na Seção 2.3, apresentamos as questões usadas para avaliar as linguagens de modelagem. Além disso, descrevemos três linguagens de modelagem para SMAs e quatro outras metodologias de SMAs que incorporam linguagens de modelagem. Finalmente, na Seção 2.4, apresentamos uma visão geral da linguagem de modelagem MAS-ML.

2.1.
Avaliação de Frameworks Conceituais

Conforme mencionado na Seção 1.1.1, os *frameworks* conceituais disponíveis na literatura não definem os aspectos dinâmicos e estruturais das entidades freqüentemente descritas em SMAs. Nesta seção, avaliamos três *frameworks* conceituais, levando em consideração esses aspectos. Por aspectos estruturais, entendemos entidades, suas propriedades e relacionamentos. Por aspectos dinâmicos, entendemos o comportamento de entidades, ou seja, a

interação entre elas e sua execução interna (intra-ações). O comportamento de uma entidade é delimitado por sua criação e destruição. Portanto, a descrição da destruição e da criação das entidades é importante para compreender o seu comportamento. Nesse contexto, descrevemos as questões associadas a esses aspectos a fim de ajudar na avaliação dos *frameworks*. Essas questões foram organizadas em dois grupos. O grupo de questões de *aspectos estruturais* refere-se a entidades, propriedades e relacionamentos. Já o grupo de *aspectos dinâmicos* trata da definição do comportamento das entidades.

Aspectos estruturais:

[S1] Quais são as entidades definidas pelo *framework* conceitual?

[S2] Quais são as propriedades associadas a cada entidade?

[S3] Como as entidades estão relacionadas? Que tipos de relacionamentos ligam as entidades?

[S4] Como as propriedades estão relacionadas?

Aspectos dinâmicos:

[D1] Como as entidades são criadas e destruídas?

[D2] Como podem ser caracterizadas as intra-ações das entidades?

[D3] Como as entidades interagem?

2.1.1. KAoS

KAoS (Dardenne et al., 1993) é um *framework* conceitual que define abstrações, como entidades, relacionamentos e agente, como extensões de objeto [S1]. Uma entidade é um objeto autônomo independente de outros objetos. Um relacionamento é um objeto subordinado. Um agente é um objeto que tem escolha e comportamento. Ele define crenças, objetivos e ações [S2]. Alguns pontos fracos do KAoS são:

1. Ele não considera organizações, papéis e ambientes abstrações importantes [S1];
2. O KAoS não explica de forma satisfatória a distinção entre uma entidade e um agente [S1];
3. Ele não descreve as características de um objeto ou explica como ele é estendido por outras abstrações [S2];

4. Também não descreve os relacionamentos entre as abstrações definidas e por que um relacionamento deve ser descrito como um objeto [S3];
5. O KAoS não descreve qual é a relação das propriedades de um agente, ou seja, não descreve como um agente alcança um objetivo com base em suas ações e crenças [S4];
6. Ele não define nenhum aspecto dinâmico associado às entidades descritas [D1][D2][D3].

2.1.2. *Framework* Conceitual d'Inverno e Luck

O *framework* conceitual proposto em (d'Inverno et al., 2001) define uma hierarquia de entidades com base no elemento primitivo chamada *entidade*. A idéia básica por trás dessa hierarquia é que um ambiente consiste em entidades, algumas das quais são objetos. Nesse conjunto de objetos, alguns são agentes; e no conjunto de agentes, alguns são agentes autônomos. Todas as entidades têm atributos. Os objetos estendem entidades ao adicionar ações que definem suas habilidades. Como os agentes estendem objetos, eles têm atributos e ações e também definem objetivos. Agentes autônomos estendem agentes definindo motivações [S1][S2]. Ao definir objetivos, motivações, ações e atributos, o *framework* conceitual descreve como essas propriedades estão relacionadas [S4]. As principais limitações desse *framework* são:

1. Características comportamentais, como operações ou ações, não são associadas à entidade ambiente. O ambiente só é definido com base em seus atributos [S2];
2. O *framework* conceitual não define as entidades organizações e papéis [S1]. Portanto, não é possível descrever os agentes que estão exercendo papéis em uma organização [D2][D3];
3. O *framework* conceitual declara que algumas entidades são objetos. O conjunto de entidades que não são objetos não é definido claramente. Até hoje não foi fornecido nenhum exemplo que ilustre o que é uma entidade que não é um objeto [S1];
4. Os agentes são definidos adicionando objetivos à definição de objetos. Contudo, agentes não são objetos. Na Seção 3.2.1 mostramos a diferença

entre agentes e objetos e explicamos por que agentes não devem ser definidos com base em objetos [S1];

5. Não são definidos os relacionamentos entre as entidades [S3];
6. O *framework* conceitual não define nenhum aspecto dinâmico associado a entidades propostas [D1][D2][D3].

2.1.3. *Framework* Conceitual Yu e Schmid

Em (Yu et al., 1999), os autores propõem um *framework* conceitual para a modelagem com base em papéis e orientada a agentes. Eles se concentram na definição de papéis descrevendo suas propriedades e relacionamentos [S1][S2][S3]. Os papéis são definidos em termos de objetivos, qualificações, obrigações (ou deveres), permissões (ou direitos) e protocolos [S2]. Os agentes são apresentados como uma entidade que exerce papéis dentro de uma organização. Os papéis são atribuídos a agentes com base na avaliação da qualificação e das habilidades [D2][D3]. Os agentes são selecionados para exercer os papéis de acordo com as políticas organizacionais e suas habilidades [D2][D3]. Alguns pontos fracos dessa proposta são:

1. Apesar de os agentes serem definidos como uma entidade que exerce papéis, o *framework* conceitual não define as propriedades de agentes [S2] e os relacionamentos entre agentes e papéis [S3].
2. Esse *framework* não define a criação e a destruição de agentes e, portanto, não descreve a interdependência entre agentes e papéis [D1];
3. Apesar de os autores afirmarem que papéis são exercidos em organizações, a proposta não define as propriedades de organização e o relacionamento entre elas e papéis [S3].
4. Tampouco define a entidade ambiente em que estão os agentes e as organizações [S1].

2.2. Visão Geral do TAO

As entidades do TAO são objetos, agentes (OMG, 2004; Ishida et al., 1992; Shoham, 1993; Wooldridge et al., 2001), organizações (Shoham, 1993; Zambonelli et al., 2001; Caire et al., 2002; Odell et al., 2003), ambientes (Ishida et al., 1992; Yu at al., 1999; Lind, 2000; Odell et al., 2003), papéis do agente (Yu et

al, 1999; Zambonelli et al., 2001; Odell et al., 2003) e papéis de objeto (Kristensen, 1997; Jennings, 2000; Kuncak et al, 2002). [S1]. As propriedades dos objetos são atributos e métodos (ou operações) [S2]. As propriedades dos agentes são objetivos, crenças e planos [S2]. Como as organizações estendem agentes descrevendo axiomas, as propriedades das organizações são objetivos, crenças, ações, planos e axiomas [S2]. As propriedades de um ambiente dependem de suas características. Podemos definir como um objeto ou um agente [S2]. As propriedades de papéis de objeto são as mesmas relacionadas a objetos: atributos e métodos [S2]. As propriedades dos papéis do agente são objetivos, crenças, deveres, direitos e protocolos [S2]. As conexões entre as propriedades são descritas a medida que as propriedades são detalhadas [S4] (consulte a Seção 3.1).

Os relacionamentos definidos pelo TAO são *inhabit*, *ownership*, *control*, *play*, *inheritance*/*specialization*, *dependency*, *association* e *aggregation*. Os relacionamentos *inheritance*/*specialization*, *dependency*, *association* e *aggregation* estendem os relacionamentos de UML. Cada relacionamento pode ser aplicado a um conjunto de entidades [S3] (consulte a Seção 3.1.6).

Enquanto descreve os aspectos dinâmicos de SMAs, o TAO define a criação e a destruição de entidades enfatizando sua interdependência [D1] e também define o comportamento independente do domínio [D2][D3](consulte a Seção 3.2.1 e 3.2.2). O comportamento independente do domínio definido no TAO está relacionado a agentes e suborganizações que estejam entrando em organizações e agentes e suborganizações que estejam deixando as organizações e seus movimentos de um ambiente para outro. Durante a descrição do comportamento independente do domínio, são detalhadas as interações e a intra-ações das entidades [D2][D3].

2.3. Avaliação das Linguagens de Modelagem que Estendem UML

A fim de avaliar diferentes linguagens de modelagem, definimos um conjunto de questões com base nos problemas descritos na Seção 1.1.2. Essas linguagens são avaliadas de acordo com (i) os aspectos dinâmicos e estruturais definidos no modelo, (ii) as extensões propostas para o metamodelo de UML e (iii) as diretrizes propostas para implementar seu modelo.

As questões estão organizadas em quatro grupos: aspectos estruturais, aspectos dinâmicos, implementação e extensão. Cada grupo reúne questões relacionadas. O grupo de *aspectos estruturais* refere-se à definição de entidades, propriedades e relacionamentos e à modelagem desses aspectos nos diagramas estruturais propostos. O grupo de *aspectos dinâmicos* refere-se à definição de interações e intra-ações e à modelagem desses aspectos nos diagramas dinâmicos propostos. O grupo de *implementação* refere-se ao refinamento dos modelos no nível do design. O grupo de *extensão* trata das extensões propostas ao metamodelo de UML.

As questões descritas para avaliar o *framework* conceitual também foram consideradas ao avaliar as linguagens de modelagem, uma vez que cada linguagem de modelagem baseia-se em um *framework* conceitual (ou metamodelo). Também foram incluídas outras questões relacionadas a aspectos dinâmicos e estruturais de SMAs ao conjunto de questões definido na Seção 2.1 . Essas questões discutem a modelagem desses aspectos.

Aspectos estruturais:

[S1] Quais são as entidades definidas pelo *framework* conceitual?

[S2] Quais são as propriedades associadas a cada entidade?

[S3] Como as entidades estão relacionadas? Quais são os relacionamentos que ligam as entidades?

[S4] Como as propriedades estão relacionadas?

[S5] Quais são os diagramas estruturais?

[S6] Como as entidades são modeladas?

[S7] Como as propriedades são modeladas?

[S8] Como os relacionamentos são modelados?

Aspectos dinâmicos:

[D1] Como as entidades são criadas e destruídas?

[D2] Como podem ser caracterizadas as intra-ações das entidades?

[D3] Como as entidades interagem? A linguagem descreve a diferença entre as mensagens enviadas por agentes e aquelas enviadas por objetos?

[D4] Quais são os diagramas dinâmicos?

[D5] Como as intra-ações de entidades são modeladas?

[D6] Como as interações entre as entidades são modeladas?

Implementação:

[I1] A linguagem oferece diretrizes para implementar um sistema modelado usando seus diagramas?

[I2] A linguagem oferece o mapeamento dos conceitos definidos em seus modelos para os conceitos usados por uma linguagem de programação ou plataforma de desenvolvimento?

Extensões:

[E1]Como o metamodelo de UML foi estendido para incorporar abstrações relacionadas a agente?

2.3.1. Linguagens de Modelagem que Estendem UML

2.3.1.1. AUML

AUML (Odell et al., 2000; Parunak et al., 2002; Bauer et al., 2002; Bauer, 2002; Odell et al., 2003; Huget, 2002a,c) é uma linguagem de modelagem que estende UML para modelar os aspectos relacionados a agentes. Muitos autores contribuem para o desenvolvimento da AUML. A AUML propõe estender e usar vários diagramas de UML como os diagramas de implantação, atividade, colaboração, seqüência e classes [S5][D4]. Entretanto, poucos trabalhos foram desenvolvidos para estender os diagramas de implantação, atividade e colaboração. Os diagramas de atividade são usados para capturar o fluxo do processamento na comunidade de agentes e são estendidos para modelar os papéis associados a uma atividade. Os diagramas de colaboração também foram estendidos ao incluir estereótipos para descrever papéis. Os diagramas de implantação foram estendidos para ilustrar a propriedade de mobilidade de um agente ao definir um estereótipo que vincula dois ambientes diferentes.

As entidades definidas na AUML são agentes, papéis e organizações (ou grupos) [S1]. O diagrama de classes de UML [S5] foi estendido para modelar relacionamentos entre papéis, entre organizações e entre agentes (Bauer, 2002) [S3]. Ao usar o diagrama de classes para modelar os relacionamentos entre papéis, esses papéis são identificados [S6], o relacionamento de associação é usado para vinculá-los [S8] e *swim lanes* são aplicados para agrupar os papéis definidos por uma organização e para indicar qual instância de agente está exercendo o papel [S8]. Ao usar o diagrama de classes para modelar os relacionamentos entre

organizações e papéis, as organizações e os papéis são identificados [S6] e o relacionamento *aggregation* é usado para vinculá-los [S8]. Ao usar o diagrama de classes para modelar os relacionamentos entre agente, os agentes são identificados [S6], suas propriedades são descritas [S2][S7] e o relacionamento *association* é usado para vinculá-los [S8]. Os agentes são descritos com base em estados (ou atributos), ações, métodos (ou operações), protocolos (e os papéis exercidos em protocolos), habilidades e papéis exercidos na organização (Bauer, 2002; Huget 2002a) [S1][S2][S4]. Em (Huget, 2002a) os relacionamentos *association*, *generalization*, *aggregation* e *dependency* foram definidos para serem aplicados a classes do agente [S3]. Além disso, o relacionamento chamado *order* foi descrito para indicar que um agente pode pedir a outro agente que execute algumas tarefas [S3].

A AUML estende o diagrama de seqüência [D4] para modelar os protocolos descritos por papéis e para modelar as interações entre agentes enquanto os papéis são exercidos [D6]. As mensagens assíncronas rotuladas por uma ação de discurso (ou ação de comunicação) são usadas para indicar mensagens enviadas e recebidas pelos agentes [D3]. Os nomes dos caminhos associados a instâncias em diagramas de seqüência foram estendidos a fim de descrever uma instância de agente, o papel sendo exercido e a classe do agente [D6]. Em (Huget, 2002c), são propostas diretrizes para gerar código dos diagramas de seqüência ao modelar protocolos com base em um exemplo [I1].

Os pontos fracos da AUML são:

1. A AUML não define ambientes como uma abstração [S1]. Portanto, não é possível modelar um agente que esteja se movendo de um ambiente para outro usando o diagrama de seqüência proposto [D6];
2. As propriedades das organizações e dos papéis não são descritas [S2] e, portanto, não são modeladas [S7];
3. A criação e a destruição de agentes e papéis não são descritas. Portanto, não está claro se a criação de um papel depende da criação de um agente para exercer o papel [D1];
4. As organizações não são modeladas no diagrama de seqüência. Assim, não é possível modelar a interação entre os agentes e as organizações [D6];

5. O *pathname* de um agente em um diagrama de seqüência não descreve a organização em que o agente está exercendo o papel. Logo, não é possível modelar um agente que esteja se movendo de uma organização para outra [D6];
6. A linguagem não define ou modela as intra-ações das entidades definidas [D2][D5];
7. Não há mapeamento ou transformação dos modelos de design para o código [I2];
8. Na AUML, os agentes são definidos como um estereótipo da metaclasse *Class*. Na Seção 4.2.1 justificamos por que um agente não deve ser definido usando um estereótipo [E1];
9. Os papéis exercidos por agentes são definidos com base na metaclasse *ClassifierRole* usada para representar os papéis exercidos por objetos enquanto interagem com outros objetos. Na Seção 4.2.2 descrevemos a diferença entre papéis do agente e papéis de objeto. Os papéis do agente e os papéis de objeto possuem diferentes propriedades e, portanto, os papéis do agente não podem ser definidos com base na metaclasse *ClassifierRole* [E1];
10. A extensão feita ao metamodelo de UML não é definida a fim de descrever as organizações [E1];

Metamodelo da FIPA

Com base na AUML, a FIPA (FIPA Modeling, 2004) está no processo de desenvolvimento de um metamodelo de superestruturas da classe do agente. O metamodelo da FIPA estende o metamodelo de UML ao definir metaclasses para representar agentes, grupos (ou organizações) e papel do agente. Contudo, o grupo não é definido como uma entidade concreta, mas como uma entidade virtual que representa um conjunto de agentes relacionado via papéis.

Uma nova metaclasse chamada *AgentClassifier* foi definida pela FIPA para representar classes abstratas do agente. Essa metaclasse é definida estendendo a metaclasse *Classifier*. A metaclasse *AgentClassifier* foi estendida para representar classes concretas do agente (metaclasse *AgentPhysicalClassifier*) e classes de papéis do agente (metaclasse *AgentRoleClassifier*). A metaclasse *AgentPhysicalClassifier* descreve o conjunto de recursos básicos (ou seja,

primitivos, principais) que todos os agentes têm enquanto a metaclasse *AgentRoleClassifier* descreve as características dos papéis.

Como a proposta de modelagem da FIPA ainda está em desenvolvimento, algumas entidades, propriedades e relacionamentos importantes ainda não foram definidos, e outros ainda não foram bem definidos:

1. As propriedades de agentes, papéis e grupos não foram descritas;
2. O ambiente de entidade não foi definido;
3. Os relacionamentos entre agentes, papéis e grupos não foram ainda apresentados;
4. A diferença entre as metaclasses *AgentClassifier* e *AgentPhysicalClassifier* ainda não foi claramente descrita.

2.3.1.2. A Linguagem de Modelagem AOR e o Metamodelo de AOR

A AORML (AOR modeling language - linguagem de modelagem AOR) (Wagner, 2002) baseia-se no metamodelo AOR (Wagner, 2000; Wagner, 2003). As entidades definidas no AOR são evento, ação, alegação, compromisso, agente ou objeto [S1]. Todas as entidades são definidas como estereótipos com base na metaclasse *Class* definida pela UML [E1]. A AOR define sete tipos de agentes como o agente institucional. Um agente institucional representa organizações e consiste em alguns agentes internos que percebem eventos e executam ações em seu nome [D2], exercendo determinados papéis. Os agentes internos possuem determinados direitos e deveres [S2]. A AOR especializa os relacionamentos *association*, *generalization* e *composition* definidos na UML [S3]. Além disso, ela define os relacionamentos *does*, *perceives*, *sends*, *receives*, *hasClaim* e *hasCommitment*, vinculando eventos de ação, compromissos, alegações e agentes [S3]. Todos esses relacionamentos são definidos como estereótipos com base na metaclasse *Association* [E1].

Os modelos AOR consistem em um modelo AOR externo, correspondendo a um modelo de análise de domínio, e um modelo AOR interno, correspondendo a um modelo de design. O modelo AOR externo define o agente, o frame da interação, a seqüência da interação e os diagramas de padrões de interação. Os diagramas de agentes descrevem as classes do agente do domínio, determinadas classes de objetos relevantes e os relacionamentos entre eles [S5][S6][S7][S8]. Os

diagramas do frame de interação ilustram as classes do evento de ação e as classes de alegação/compromisso que determinam as possíveis interações entre duas classes do agente [D4][D6]. Os diagramas de padrões de interação enfatizam os padrões de interação gerais expressos por meio de um conjunto de regras de reação que definem um tipo de processo de interação [D4][D6]. Os diagramas de seqüência de Interação definem as instâncias prototípicas dos processos de interação [D4][D6]. AOR define as mensagens enviadas e recebidas pelos agentes ressaltando a diferença entre as mensagens enviadas e recebidas por objetos [D3].

Um modelo AOR interno define os frames de reação, as seqüências de reação e os diagramas de padrões de reação. O diagrama do frame de reação define os agentes e as classes de evento e ação, assim como as classes de alegação e compromisso que determinam as possíveis interações entre eles [D4][D6]. O diagrama de seqüência de reação define as instâncias prototípicas dos processos de interação dentro da perspectiva interna [D4][D5]. Os diagramas de padrões de reação enfatizam os padrões de reação do agente com as considerações expressas por meio das regras de reação [D4].

Os pontos fracos do metamodelo de AOR e AORML são:

1. A AOR não define ambientes [S1]. Portanto, não é possível modelar agentes que estejam se movendo de um ambiente para outro [D6];
2. As propriedades dos agentes institucionais (organizações) não são definidas e, portanto, não são modeladas [S2][S4][S7];
3. Não há distinção clara entre agentes externos, internos e papéis [S1];
4. A criação e a destruição de entidades da AOR não são especificadas [D1];
5. Usando modelos AOR, não é possível modelar agentes que estejam se movendo de uma organização para outra [D6];
6. AOR não modela a execução interna de agentes [D2][D5];
7. A proposta não define qualquer diretriz para implementar o sistema modelado usando seus diagramas. Não há mapeamentos ou transformações dos modelos de design para o código [I1][I2];
8. Os estereótipos com base na metaclasse *Class* são usados para definir todas as entidades do AOR. Na Seção 4.2.1 justificamos por que um agente não deve ser definido usando esse estereótipo [E1].

2.3.1.3. Transformação Gráfica e UML

Em (Depke et al., 2002), temos a descrição de uma técnica de modelagem para sistemas com base em agentes construídos com os conceitos e as notações da transformação gráfica e da UML. A técnica de modelagem é definida em três fases: requisito, análise e design. Os diagramas de UER (casos de uso estendidos para modelar agentes e objetivos) são usados na fase de especificação de requisitos a fim de representar a principal função externa do sistema, assim como as interações internas importantes entre agentes. As regras da transformação gráfica são usadas para descrever o sistema antes e depois da execução de um caso de uso ao representar a alteração de estado de objetos e agentes no sistema [S1][S4].

Os diagramas de classes [S5] são usados na análise para especificar os agentes e os objetos, além de suas mensagens [D3], atributos [S2] e associações [S3], identificados nos cenários de caso de uso [S6][S7][S8]. Também são usados na fase de design a fim de introduzir mais recursos, em particular, as assinaturas das operações autônomas do agente [S2][S7].

Para modelar os aspectos dinâmicos de um sistema, são usados diagramas de estado e seqüência [D4]. Os diagramas de seqüência são usados na fase de especificação de requisitos para especificar a *comunicação* entre os atores que participam de um caso de uso [D3][D6]. Além disso, são usados na fase de análise para descrever a comunicação necessária para executar um determinado protocolo [D3]. Os diagramas de estado são usados na fase de design a fim de determinar a ordem de execução local de uma operação do agente [D2][D5]. O efeito dessas operações no estado do sistema é descrito por regras de transformação gráficas locais.

Os pontos fracos dessa proposta são:

1. A proposta não modela outras entidades como papéis, organizações e ambientes e defende que a principal abstração dos SMAs são os agentes [S1]. Portanto, não é possível modelar agentes que estejam exercendo papéis em organizações ou se movendo de uma organização para outra. Ademais, não é possível modelar agentes que estejam se movendo de um ambiente para outro [D6];

2. Ela somente define um relacionamento que deverá ser usado para vincular agentes e objetos e não justifica por que outros relacionamentos não precisam ser descritos [S3];
3. O relacionamento entre os objetivos e as propriedades de outros agentes (atributos e mensagens) não é modelado [S4]. Como conseqüência, os relacionamentos entre os atributos e as mensagens definidos na fase de análise e os objetivos definidos na fase de especificação de requisitos não estão claros [S4];
4. A criação e a destruição de entidades não são especificadas [D1];
5. Os agentes são definidos como tendo operações (ou métodos) [D2];
6. As intra-ações de agentes não são modeladas em diagramas de seqüência [D5];
7. No diagrama de classes, as mensagens associadas a agentes são modeladas como métodos de objetos ativos, e nos diagramas de seqüência, são modelados como mensagens assíncronas OO. Na Seção 4.2.3, definimos a diferença entre mensagens de um objeto e mensagens de um agente e na Seção 4.4.2 propomos como modelar os novos tipos de mensagens [D3];
8. Os agentes são modelados como objetos ativos e, portanto, as classes são usadas a fim de modelar agentes. Na Seção 3.1.2 estabelecemos a diferença entre agentes e objetos ativos, ressaltando por que agentes não devem ser modelados como objetos ativos [S1];
9. A proposta não define qualquer diretriz para implementar o sistema modelado usando seus diagramas. Não há mapeamentos ou transformações dos modelos de design para o código [I1][I2].

2.3.2. Metodologias que Incorporam as Linguagens de Modelagem que Estendem a UML

Ao descrever as metodologias apresentadas nesta seção, concentrar-nos-emos nas características associadas às linguagens de modelagem usadas, definidas ou estendidas. Como nosso objetivo não é desenvolver uma metodologia, as características relacionadas à metodologia (orientadas a processo) não serão avaliadas. As linguagens de modelagem usadas, definidas ou estendidas pelas modelagens são discutidas seguindo as questões apresentadas na Seção 2.3.

2.3.2.1. MESSAGE

A metodologia MESSAGE (Caire et al., 2002) define agentes, organizações, papéis e características [S1]. Os agentes têm objetivos, habilidades (ou tarefas/ações) e crenças (ou conhecimento) e podem exercer papéis. Os papéis descrevem as características externas de um agente em um determinado contexto. Uma organização é um grupo de agentes os quais trabalham juntos para alcançar um propósito comum. As organizações são entidades virtuais no sentido em que o sistema não possui nenhuma entidade computacional individual que corresponda a uma organização. Os recursos representam entidades não-autônomas como bancos de dados ou programas externos usados por agentes [S2].

Essa metodologia introduz os diagramas de domínio, fluxograma, delegação, objetivo/tarefa e organização para modelar os aspectos estruturais [S5] e o diagrama de interação a fim de modelar os aspectos dinâmicos dos SMAs [D4]. Todos são extensões do diagrama de classes de UML, exceto o diagrama de tarefas, que estende o diagrama de atividade de UML. Um esquema também é introduzido para descrever as características de cada papel/agente [S6][S7].

O diagrama da organização mostra as entidades (agentes, organizações, papéis, características e classes de objetos) do sistema [S6] e os relacionamentos entre as entidades (relacionamentos *aggregation*, power e acquaintance) [S3][S8]. O diagrama de objetivo/tarefa mostra objetivos, tarefas, situações e a dependência entre eles [S4][S7]. O diagrama de delegação mostra como os subobjetivos obtidos decompondo um objetivo são atribuídos aos agentes/papéis incluídos na organização [S4]. O diagrama de fluxograma mostra os papéis em uma organização que devem realizar as tarefas necessárias para implementar um dado serviço fornecido pela organização [S4]. Os diagramas de domínio são diagramas de classes em que as classes representam conceitos de domínios específicos. Os diagramas de interação são centrados em interação (ou seja, há um diagrama desses para cada interação) e mostram o iniciador, os respondedores, o motivador (normalmente um objetivo do iniciador) de uma interação e outras informações relevantes como as informações alcançadas e fornecidas por cada participante [D4][D6].

O diagrama de seqüência proposto pela AUML também pode ser usado para representar os protocolos de interação e as mensagens trocadas entre os papéis [D2][D3]. Em MESSAGE, uma mensagem é especificada por um emissor, destinatário, ação de discurso e o conteúdo[D4].

Os pontos fracos da linguagem de modelagem proposta por MESSAGE são:

1. A linguagem não descreve claramente o conjunto de relacionamentos definidos. Não explica a diferença entre o relacionamento *acquaintance* e o relacionamento *association* de UML. Além disso, não define qualquer relacionamento que vincula o agente, o papel que exerce e a organização em que exerce o papel [S3][S8];
2. Não há distinção clara entre agentes e papéis. A metodologia especifica que os agentes são identificados com base em papéis, mas não define diretrizes para descobrir os agentes. Ademais, o diagrama da organização não define qualquer mecanismo para modelar os papéis exercidos por um agente [S1];
3. As propriedades atribuídas a papéis não são definidas. A linguagem não define como as características de um agente podem ser expressas por meio de seus papéis [S2];
4. A proposta usa o diagrama de seqüência apresentado pela AUML a fim de detalhar os protocolos de interação entre agentes/papéis [D4];
5. A criação e a destruição de entidades não são especificadas [D1];
6. A metodologia não descreve se um agente pode alterar papéis ou se pode se comprometer com outro papel [D3];
7. A proposta não descreve nem modela a execução interna das entidades definidas [D2][D5];
8. Ela também não descreve como o metamodelo de UML foi estendido a fim de incorporar conceitos relacionados a agentes [E1];
9. MESSAGE não define qualquer diretriz para implementar o sistema modelado usando os diagramas propostos. Não há mapeamento ou transformação dos modelos de design para o código [I1][I2].

2.3.2.2. Tropos

A metodologia Tropos (Mylopoulos et al., 2001; Castro et al., 2002) define conceitos como ator (um ator pode ser um agente ou um papel) [S1], assim como dependências sociais entre atores, incluindo dependências de recursos, tarefa, *softgoals* e objetivos [S3][S4]. O diagrama proposto pela metodologia para modelar os aspectos estruturais do sistema é o diagrama de atores [S5]. Os aspectos dinâmicos são modelados por meio dos diagramas de interação de agentes, plano e habilidade [D4]. Todos os diagramas usados para modelar aspectos dinâmicos baseiam-se em diagramas de AUML. Um diagrama de atores define os atores [S6], seus objetivos [S2] e os relacionamentos de dependência entre os atores [S7][S8]. Os diagramas de habilidade usam o diagrama de atividades de AUML a fim de modelar habilidades a partir do ponto de vista de um determinado agente. Os eventos externos definidos no estado inicial de um diagrama de habilidade, planos de modelo de nós de atividades, eventos de modelo de arcos de transição e crenças são modelados como objetos [S4][S7]. O diagrama de plano descreve a estrutura interna de um plano, resumida como um nó simples em um diagrama de habilidade [S4][S7]. Os diagramas de plano são modelados usando diagramas de ação de AUML. O diagrama de interação de agentes modela a interação entre agentes e objetos. Esse diagrama usa o diagrama de seqüência de AUML [D4] em que os agentes interagem enviando mensagens que são definidas de forma diferente das mensagens de objeto [D3][D6]. Tropos também usa o diagrama de classes de AUML para modelar os aspectos estruturais dos atores e de outras entidades [S6].

A metodologia Tropos usa a plataforma de programação de agentes Jack (Howden et al., 2001) a fim de implementar seus agentes [I1]. Ela define como os conceitos podem ser mapeados no modelo BDI e nos construtos de Jack e como cada conceito está relacionado aos outros dentro do mesmo modelo [I2].

Os pontos fracos da linguagem de modelagem proposta por Tropos são:

1. As propriedades de atores não são explicitamente descritas. Contudo, o diagrama de habilidade e atores indica que os atores têm objetivos, habilidade/planos e crenças [S2];

2. Tropos não define a diferença entre atores e papéis [S1]. Portanto, não é possível modelar o relacionamento entre atores (ou agentes) e papéis [S3], tampouco é possível modelar atores que estejam se comprometendo com papéis ou alterando papéis [D6];
3. Tropos não define organizações e ambientes [S1]. Portanto, não é possível modelar agentes que estejam exercendo papéis em organizações ou se movendo de um ambiente para outro [D6];
4. Tropos não descreve como instâncias de agente/atores são criadas ou destruídas [D1];
5. A proposta não modela as intra-ações das entidades definidas [D2][D5];
6. Como quase todos os diagramas de Tropos baseiam-se nos diagramas de AUML, todos os pontos fracos indicados em relação a diagramas de AUML também se aplicam aos diagramas de Tropos [S5][D4];
7. Ela também não descreve como o metamodelo de UML foi estendido a fim de incorporar conceitos relacionados a agentes [E1].

2.3.2.3. Prometheus

Prometheus (Padgham et al, 2002) define um agente com base em seus objetivos, crenças, planos e eventos [S1][S2]. A metodologia consiste em três fases: fases de especificação do sistema, design da arquitetura e design detalhado. A fase de especificação do sistema usa cenários de caso de uso para descrever seqüências das etapas de operações.

A fase de design da arquitetura atribui as funcionalidades definidas a agentes na fase anterior. Essa fase usa os diagramas de interação [D4], sistema [S5] e conhecimento [S5]. O diagrama de conhecimento simplesmente vincula cada agente a outros com os quais interage [S3][S6][S8]. As propriedades dos agentes não são descritas nesse diagrama, mas são descritas na forma da descrição de um agente [S7]. O diagrama de sistema une objetos de dados compartilhados, agentes e eventos [S8]. O diagrama de interação mostra interação entre agentes em vez de objetos [D6]. Esse diagrama veio diretamente do design orientado a objetos e usa a notação da AUML para especificar protocolos [D3][D6].

A fase de design detalhado define os diagramas de habilidade e agente. O diagrama de visão geral de agente é muito semelhante ao diagrama de sistema,

mas oferece uma visão de alto nível da parte interna do agente concentrando-se nas habilidades [D2] dentro de um agente em vez de um sistema [D5]. Esse diagrama mostra as habilidades e os fluxos de tarefas e eventos entre essas habilidades, assim como os dados internos do agente [D5]. Para cada habilidade definida nos diagramas de agentes, há um diagrama de habilidade que descreve a parte interna mostrando o refinamento desde as habilidades até os planos [S4]. A linguagem define sua própria notação a fim de descrever agentes, habilidades, planos, eventos e dados/crenças. Além disso, os descritores de dados, eventos e planos individuais podem ser usados para descrever essas propriedades [S7]. Essas descrições podem fornecer os detalhes necessários para mover para a implementação [I1]. Ademais, uma ferramenta de suporte à metodologia permite que os diagramas de design sejam traçados e gera um código esqueleto correspondente (em Jack) [I2].

Os pontos fracos da linguagem de modelagem Prometheus são:

1. Prometheus não define papéis, organizações e ambientes [S1]. Portanto, não é possível modelar agentes que estejam exercendo papéis em organizações ou se movendo de uma organização para outra. Ademais, não é possível modelar agentes que estejam se movendo de um ambiente para outro [D6];
2. A linguagem somente define um relacionamento que deverá ser usado para vincular agentes e não justifica por que outros relacionamentos não são necessários [S3];
3. Prometheus não descreve como instâncias de agente são criadas ou destruídas [D1];
4. A linguagem usa o diagrama de seqüência de AUML a fim de descrever as interações entre agentes. Assim, todos os problemas relacionados a esse diagrama também se aplicam ao diagrama de interação de Prometheus [D4];
5. Ela também não descreve como o metamodelo de UML foi estendido a fim de incorporar conceitos relacionados a agentes [E1].

2.3.2.4. MaSE

A metodologia MaSE (DeLoach, 1999) usa a linguagem de modelagem AgML, que é semelhante à UML. No entanto, foram adicionadas outras características à linguagem, e outras semânticas orientadas a objetos tradicionais foram modificadas a fim de capturar a noção de agentes e seu comportamento cooperativo. A linguagem apenas modela a entidade de agente [S1] que possui objetivos e serviços [S2]. Os agentes se comunicam enviando e recebendo mensagens [D3] descritas usando uma linguagem comum de troca de mensagem. Os agentes são modelados como objetos ativos [S6].

A AgML usa quatro diagramas para definir características de alto nível de sistemas multiagentes. Os diagramas de implantação e agentes descrevem os aspectos estruturais [S5] do sistema enquanto a hierarquia da comunicação e os diagramas de classes de comunicação definem os aspectos dinâmicos [D4]. Os diagramas de agentes definem as classes do agente [S6] e seus relacionamentos com base em conversas [S3][S8]. As conversas descrevem os protocolos de coordenação entre agentes [D6]. Uma conversa é identificada por um nome e pelos papéis exercidos pelos agentes nela.

Um diagrama de hierarquia da comunicação define os relacionamentos entre as diferentes conversas dentro de um sistema [S4]. Os diagramas de classes de comunicação são um conjunto de diagramas de máquina de estado finita que definem os estados das conversas em que os agentes podem estar, conforme definido pelo papel do agente nas conversas. Cada diagrama define a seqüência legal das mensagens que podem ser enviadas entre dois agentes envolvidos em uma conversa (ou seja protocolos) [D3][D6]. Os diagramas de implantação definem os parâmetros do sistema como locais, tipos e números reais dos agentes no sistema.

Os pontos fracos da AgML são:

1. A MaSe não define organizações nem ambientes [S1]. Portanto, não é possível modelar agentes que estejam exercendo papéis em organizações ou se movendo de uma organização para outra. Ademais, não é possível modelar agentes que estejam se movendo de um ambiente para outro [D6];

2. Não é possível modelar agentes comprometidos com papéis ou que estejam mudando de papéis. [D6];
3. A linguagem não define qualquer diagrama que descreva os relacionamentos entre serviços, objetivos e conversas [S4];
4. MaSe não descreve como instâncias de agente são criadas ou destruídas [D1];
5. Não é possível modelar agentes que estejam mudando de uma conversa para outra. Cada diagrama de classes de comunicação define uma conversa [D3][D6];
6. A linguagem não define ou modela as intra-ações associadas aos agentes [D2][D5];
7. A linguagem não descreve o conjunto de conversas disponíveis (ou relacionamentos) que vinculam agentes. Também não especifica se o designer pode definir livremente conversas [S3];
8. Os agentes são modelados como objetos ativos e, portanto, as classes são usadas a fim de modelar agentes. Na Seção 3.1.2 estabelecemos a diferença entre agentes e objetos ativos e defendemos que os agentes não devem ser modelados como objetos ativos [E1].

2.4. Visão Geral da MAS-ML

Ao descrever a linguagem de modelagem MAS-ML, o metamodelo de MAS-ML [E1] é apresentado junto com os diagramas estruturais e dinâmicos da MAS-ML. Além disso, foi desenvolvido um transformador para gerar código dos diagramas estruturais da MAS-ML [I1][I2].

Como essa linguagem se baseia no TAO, os aspectos dinâmicos e estruturais modelados que usam a MAS-ML também são definidos no TAO [S1][S2][S3][S4][D1][D2][D3]. A MAS-ML define os diagramas estruturais de papel, organização e classe [S5] e os diagramas dinâmicos de seqüência [D4] (consulte a Seção 4.3 e 4.4). O diagrama de classe de MAS-ML estende o diagrama de classe de UML a fim de modelar agentes, organizações e ambientes. Os relacionamentos do TAO usados nesse diagrama são relacionamentos *inhabit*, *association* e *specialization*. O diagrama de organização modela organizações, agentes, papéis do agente, papéis de objetos e ambientes. Os relacionamentos

usados nesse diagrama são relacionamentos *ownership*, *play* e *inhabit*. O diagrama de papel modela os papéis do agente, papéis de objetos e as classes. O conjunto de relacionamentos usados nesse diagrama são relacionamentos *control*, *dependency*, *association*, *aggregation* e *specialization*. A fim de representar graficamente as entidades relacionadas a agentes e suas propriedades, foram propostos novos elementos de diagrama [S6][S7][S8].

O diagrama de seqüência de UML [D4] foi estendido (i) para modelar a interação entre agentes, organizações, ambientes e objetos, (ii) para modelar a execução de planos e ações associados a agentes, organizações e ambientes ativos [D2][D5] e (iii) para modelar protocolos definidos por papéis (consulte a Seção 4.4) [D6]. A fim de representa graficamente essas entidades em diagramas de seqüência, foram definidos novos *pathnames* e novos elementos de diagrama. Além disso, o conceito de mensagens enviadas e recebidas foi estendido. Dois tipos diferentes de mensagens podem ser representados em diagramas de seqüência: as mensagens enviadas e recebidas por objetos e aquelas enviadas e recebidas por agentes (ou organizações ou ambientes ativos) [D3]. Ademais, alguns estereótipos associados a mensagens foram definidos a fim de representar agentes e organizações que estejam comprometidos com papéis, estejam mudando, ativando, desativando e cancelando papéis [D3].