1
Introdução

Os sistemas multiagentes (SMAs) estão tendo cada vez mais aceitação no setor da engenharia de software e no meio acadêmico como um paradigma para o desenvolvimento e a criação de sistemas de software (Lind, 2001; Wooldridge et al., 2001). Junto com esse crescimento, estão sendo propostas novas tecnologias, métodos, linguagens de modelagem, plataformas de desenvolvimento, ferramentas e linguagens de programação. Os sistemas baseados em agentes requerem técnicas adequadas que explorem seus benefícios e suas características próprias (Zambonelli et al., 2002). No entanto, diferentes metodologias, linguagens e plataformas para SMAs propõem conjuntos muito variados de abstrações. Normalmente é muito difícil entender a definição de cada abstração e os relacionamentos entre elas. Nesse contexto, há a necessidade de criar um *framework conceitual* que defina as abstrações, seus relacionamentos e seu comportamento.

Como em qualquer novo paradigma para engenharia de software, o sucesso e a difusão de SMAs requerem *linguagens de modelagem*, entre outras tecnologias de software baseadas em agentes, que explorem o uso de abstrações relacionadas a agentes e promovam o refinamento dos modelos de design para código. As linguagens de modelagem devem representar os aspectos dinâmicos e estruturais (ou estáticos) dos SMAs expressando as características essenticais de todas as entidades. Os aspectos estruturais incorporam a definição das entidades, suas propriedades e seus relacionamentos. Os aspectos dinâmicos estão relacionados ao comportamento das entidades (UML, 2004).

Para reduzir o risco ao adotar uma nova tecnologia, convém apresentá-la como uma extensão incremental de métodos confiáveis e conhecidos e oferecer ferramentas de engenharia explícitas que ofereçam suporte a métodos aceitos para implementação da tecnologia (Odell et al., 2000; Bauer, 2002). Uma linguagem de modelagem para sistemas multiagentes deve ser, de preferência, uma extensão incremental de uma linguagem de modelagem confiável e conhecida.

Como agentes e objetos coexistem em SMAs (Caire et al, 2002; Castro et al., 2002), a linguagem de modelagem UML (UML, 2004) pode ser usada como uma base para o desenvolvimento de linguagens de modelagem para SMAs. Essa linguagem é um padrão *de facto* para a modelagem orientada a objetos. A UML é usada tanto no setor de engenharia de software quanto no meio acadêmico para modelar sistemas orientados a objetos. No entanto, em sua forma original, a UML oferece um suporte insuficiente para modelar SMAs. Entre outros problemas, o metamodelo de UML não oferece suporte à modelagem de agentes, papéis de agentes e organizações; entidades definidas em SMAs.

1.1. Declaração do Problema e Limitações dos Trabalhos Relacionados

Depois de exaustiva análise de teorias, metodologias e métodos para sistemas multiagentes, sentimos a necessidade de um *framework conceitual* que definisse as abstrações normalmente usadas e encontradas nos SMAs. Como nenhuma das linguagens de modelagem disponíveis para SMAs modelam os aspectos dinâmicos e estruturais dessas abstrações, há também a necessidade de uma *linguagem de modelagem* para SMAs.

1.1.1. Um Framework Conceitual para SMAs

Até então foram propostos na literatura poucos *frameworks* conceituais que descrevessem conceitos de SMAs (Yu et al., 1999; Dardenne et al., 1993; d'Inverno et al., 2001). Esses *frameworks* não definem muitos aspectos dinâmicos e estruturais normalmente descritos em SMAs (consulte o Capítulo 2 para obter mais detalhes).

A Necessidade de Definir Aspectos Estruturais de SMAs

Diferentes técnicas baseadas em agentes descrevem SMAs com base em vários tipos de entidades. Cada técnica descreve diferentes propriedades e associa diversos relacionamentos a cada entidade. Portanto, há a necessidade de definir os aspectos estruturais de SMAs ao descrever as entidades freqüentemente encontradas nesses sistemas. Enquanto as entidades são descritas, é necessário definir as propriedades associadas a elas e seus relacionamentos. Os relacionamentos entre as propriedades também devem ser descritos.

A Necessidade de Definir Aspectos Dinâmicos de SMAs

Os aspectos dinâmicos caracterizam-se pela execução interna das entidades (intra-ações) e pelas interações entre elas. As entidades podem ser executadas e interagir de formas diferentes. Como os SMAs são compostos por várias entidades, há a necessidade de descrever seus aspectos dinâmicos.

As intra-ações de uma entidade estão relacionadas às propriedades comportamentais definidas por ela. Por exemplo, as intra-ações de objetos estão relacionadas à execução de métodos, e as intra-ações de agentes, à execução de ações e planos que são as características comportamentais dos agentes. As interações entre uma entidade e outra são influenciadas pelos relacionamentos que as conectam. Apesar de os agentes interagirem enviando e recebendo mensagens, a seqüência de mensagens e o conteúdo de cada mensagem enviada e recebida pelos agentes são influenciados por seus relacionamentos. Logo, há a necessidade de descrever as interações entre as entidades com base nos relacionamentos que as conectam.

1.1.2. Uma Linguagem de Modelagem para SMAs

Foram propostas muitas linguagens de modelagem para SMAs que estendem o metamodelo de UML (Odell et al., 2000; Wagner, 2003; Depke et a. 2002). Contudo, ainda há a necessidade de uma linguagem de modelagem que (i) descreva os conceitos relacionados a agentes como abstrações de primeira classe, (ii) seja baseada em uma descrição explícita de um metamodelo de SMA, (iii) modele os aspectos dinâmicos e estruturais freqüentemente descritos em SMAs e (iv) promova o refinamento dos modelos de design para código.

A Necessidade de Representar Entidades de SMAs como Abstrações de Primeira Classe

Uma linguagem de modelagem para SMAs deve definir suas entidades como abstrações de primeira classe. Todas as linguagens propostas descrevem *agente* como uma abstração de primeira classe. Entretanto, entidades como *papel*, *organização* e *ambiente* não são definidas dessa forma em muitas delas. Devido à essa limitação, essas linguagens não podem ser usadas para modelar muitos aspectos dinâmicos e estruturais de SMAs. Não é possível modelar os relacionamentos e as interações entre agentes, objetos e outras entidades de SMAs.

A Necessidade de uma Descrição Explícita de um Metamodelo de SMA

Um metamodelo define uma linguagem para especificar modelos descrevendo a semântica de um conjunto de abstrações e definindo como essas abstrações são instanciadas (UML, 2004). Para cada abstração, o metamodelo descreve sua semântica, o metarelacionamento com outras abstrações e a representação gráfica dessa abstração em modelos.

Muitas linguagens de modelagem propostas que estendem UML não descrevem claramente as extensões aplicadas ao metamodelo de UML. Apesar de descreverem extensões para diagramas UML, essas linguagens normalmente não descrevem como o metamodelo de UML foi estendido a fim de modelar novas abstrações. As linguagens de modelagem descrevem a representação gráfica das novas abstrações, mas não descrevem bem sua semântica ou os relacionamentos entre elas.

As linguagens de modelagem que descrevem as extensões aplicadas ao metamodelo de UML usam estereótipos baseados em metaclasses. Um estereótipo é o elemento de um modelo que define valores adicionais (com base na definição das *tags*), outras restrições e, opcionalmente, uma nova representação gráfica. Eles oferecem uma forma de definir subclasses virtuais de metaclasses de UML com novos metaatributos e outra semântica. Uma metaclasse é uma classe cujas instâncias são classes. Em geral, são usadas para construir metamodelos. As linguagens de modelagem para SMAs publicadas na literatura usam a metaclasse *Class* (que representa classes de objeto) para definir agentes. Como agentes e objetos não compartilham as mesmas propriedades e relacionamentos, os agentes não devem ser descritos com base em objetos (consulte a Seção 3.1.2 e 4.2.1 para ter acesso a mais discussões).

A Necessidade de Modelar Aspectos Estruturais de SMAs

Uma linguagem de modelagem para SMAs deve descrever os diagramas estruturais para modelar os aspectos estruturais de SMAs. O conjunto de diagramas estruturais precisa ser capaz de modelar (i) as entidades normalmente definidas em SMAs, (ii) as propriedades dessas entidades associando as propriedades com as entidades e (iii) o relacionamento entre as entidades. As linguagens de modelagem propostas na literatura não modelam muitas entidades

de SMAs e, portanto, não definem os relacionamentos entre agentes e essas entidades.

A fim de modelar os relacionamentos, as propriedades e as entidades de SMAs, os diagramas estruturais de UML precisam ser estendidos. Elementos de diagrama[1] podem ser criados para representar relacionamentos, propriedades e entidades de SMAs. Eles facilitam a visualização e a modelagem dessas abstrações. Se a linguagem de modelagem define mais de um diagrama estrutural, cada diagrama deve descrever o conjunto de entidades, propriedades e relacionamentos que pode ser modelado. Também é importante especificar se os diagramas definem diferentes visualizações da mesma abstração ou se modelam vários conjuntos de abstrações.

A Necessidade de Modelar Aspectos Dinâmicos de SMAs

Os diagramas dinâmicos de SMAs devem ser definidos por uma linguagem de modelagem para SMAs a fim de modelar os aspectos dinâmicos desses sistemas. Eles devem ser capazes de modelar (i) as interações entre as entidades definidas nos diagramas estruturais e (ii) a execução interna dessas entidades. Eles podem ser definidos estendendo os diagramas dinâmicos de UML enquanto definem as interações e intra-ações de instâncias de entidades de SMAs.

Os diferentes tipos de interação precisam ser modelados nos diagramas dinâmicos para SMAs. As entidades que compões os SMAs interagem de várias formas. Os diagramas dinâmicos para SMAs também devem modelar os diferentes comportamentos internos das entidades do mesmo. Além disso, os elementos do diagrama devem ser criados para representar as instâncias da entidade do SMA.

Muitas linguagens de modelagem propostas não representam as interações das abstrações relacionadas a agentes e objetos. Ademais, algumas linguagens não modelam a execução interna das abstrações relacionadas a agentes.

A Necessidade do Refinamento de Modelos de Design de SMAs para Código

O desenvolvimento de abordagens apropriadas para implementar sistemas baseados em agentes é uma questão essencial para usar a tecnologia de agentes no

[1] Elementos de diagrama são elementos usados para representar graficamente abstrações em diagramas.

desenvolvimento de software. A fim de implementar os SMAs desenvolvidos usando uma linguagem de modelagem para SMAs, é necessário transformar os modelos de design de SMAs em código. Esses modelos são modelos de alto nível compostos por abstrações relacionadas a agentes. Para transformar modelos de SMAs em código, as abstrações relacionadas a agentes precisam ser mapeadas em abstrações definidas na linguagem de programação.

1.2. Solução Proposta

Esta tese contempla a definição de um *framework* conceitual e uma linguagem de modelagem para SMAs. Nossos objetivos são descrever os aspectos estruturais e dinâmicos das abstrações freqüentemente utilizadas em sistemas multiagentes definindo um *framework* conceitual e propor uma linguagem de modelagem que descreva diagramas estruturais e dinâmicos para modelar esses aspectos.

Demonstramos ter conseguido alcançá-los modelando um SMA referenciado na literatura como um *benchmark* apropriado. Além disso, outras duas aplicações de SMAs consideradas *benchmarks* são descritas ressaltando as diferenças entre elas. A análise das situações das aplicações é muito importantes para explorar todos os espectros das características dos sistemas multiagentes que podem ser modelados usando nossa linguagem de modelagem (Capítulo 6).

Nosso *framework* conceitual para SMAs (TAO) é avaliado (i) tornando explícito o conjunto de aspectos estruturais e dinâmicos descrito no TAO que não são apresentados em outros *frameworks* conceituais e (ii) modelando os aspectos dinâmicos e estruturais de um sistema multiagentes de acordo com as definições do TAO. Quase todos os aspectos dinâmicos e estruturais descritos no TAO são ilustrados modelando um *benchmark* para SMAs (Capítulo 6). Nossa linguagem de modelagem para SMAs (MAS-ML) é avaliada por meio de várias situações de modelagem difíceis e/ou impossíveis de representar nas linguagens existentes. Essas situações foram selecionadas a partir das características dos SMAs propostas na literatura. Os designs da MAS-ML também serão comparados aos designs fornecidos por outras linguagens de modelagem para sistemas multiagentes que estendem a UML.

1.2.1.
O *Framework* Conceitual TAO

O objetivo do *framework* conceitual TAO (Taming Agents and Objects) (Silva et al. 2003) é definir um conjunto central de abstrações de SMAs. Esse conjunto de abstrações usado no TAO foi desenvolvido com base em nossa investigação das metodologias orientadas a objetos e baseadas am agentes (Elammari et al., 1999; Yu et al., 1999; Lind, 2000; Wagner, 2000; Wooldridge et al., 2000; Caire et al., 2002), linguagens (Shoham, 1991; Weerasooriya, 1995; Kinny, 2002) e teorias (Shoham, 1993; Carley, 1999; Petrie, 2001) existentes. O TAO agrupa as abstrações freqüentemente descritas na literatura para os SMAs. O benefício desse *framework* é oferecer suporte ao desenvolvimento de novas metodologias, métodos e linguagens com base nos conceitos essenciais definidos nele e relacionados a ele. Cada conceito é visto como uma abstração candidata para linguagens de modelagem, metodologias e ambientes com suporte que pode ser aplicada em diferentes fases do desenvolvimento de SMAs.

O TAO define os aspectos dinâmicos e estruturais de SMAs. Enquanto descreve os aspectos estruturais dos SMAs, o TAO define as entidades que podem ser descritas, suas propriedades e os relacionamentos associados a elas. Ao descrever os aspectos dinâmicos de SMAs, o TAO define a criação e as destruições de entidades enfatizando sua interdependência e também define o comportamento independente do domínio (consulte a Seção 3.2.1 e 3.2.2).

1.2.2.
A Linguagem de Modelagem MAS-ML

O objetivo da MAS-ML (Multi-Agent System Modeling Language - Linguagem de Modelagem para Sistemas Multiagentes) (Silva et al., 2004a,b,c) é modelar todos os aspectos dinâmicos e estruturais definidos no TAO. Seu metamodelo é definido estendendo o metamodelo de UML de acordo com os conceitos definidos no TAO (consulte a Seção 4.2).

A MAS-ML define os digramas estruturais e dinâmicos para representar todos os aspectos do TAO. Esses diagramas podem ser usados durante as fases de design e análise. Os diagramas estruturais definidos pela MAS-ML são diagramas de papel, organização e classe (consulte a Seção 4.3). Com o uso de três

diagramas estruturais é possível modelar os aspectos estruturais de todas as entidades definidas no TAO.

O diagrama dinâmico definido pela MAS-ML é o diagrama de seqüência de UML estendido. Esse diagrama foi estendido (i) para modelar a interação entre as entidades, (ii) para modelar sua execução interna e (iii) para modelar protocolos de interação de agentes (consulte a Seção 4.4).

Com o objetivo de implementar os sistemas modelados usando a MAS-ML, foi desenvolvido um transformador para gerar código a partir dos diagramas estruturais dessa linguagem. Os modelos descritos no nível de abstração do agente são transformados em código orientado a objetos (consulte o Capítulo 5).

O processo de transformação (Silva et al., 2004d) baseia-se em uma arquitetura abstrata orientada a objetos para SMAs gerados a partir do metamodelo da MAS-ML. Essa arquitetura define um conjunto de módulos de componentes orientados a objetos e independentes do domínio e suas relações.

1.3. O Relacionamento entre UML, TAO e MAS-ML

Para explicar melhor o relacionamento entre UML, TAO e MAS-ML, usamos uma arquitetura de metadados de quatro camadas descrita na especificação MOF (MOF, 2004). As quatro camadas da arquitetura são: camada metametamodelo, camada metamodelo, camada modelo de domínio e camada instância. A camada instância foi suprimida na Figura 1.

A camada metametamodelo (primeira camada na Figura 1) é composta pela descrição da estrutura e da semântica dos metametadados. OMG especifica um metametamodelo chamado MOF que define um *framework* e uma linguagem abstrata para especificar, construir e gerenciar metamodelos neutros de tecnologia. No TAO, usamos o modelo ER (Entity-Relationship – Entidade-Relacionamento) (Chen, 1976) para descrever os metatadados de relacionamento e entidade que aparecem nessa camada. Esses metametadados oferecem as definições básicas para descrever as diferentes instâncias de entidade e relacionamento que aparecem na camada metamodelo.

A camada metamodelo (segunda camada na Figura 1) é composta pela descrição da estrutura e da semântica dos metadados que são informalmente agregados como metamodelos. Metamodelos são instâncias de metametamodelos,

e metadados são instâncias de metametadados. OMG define o metamodelo de UML sendo uma instância do metametamodelo de MOF. Definimos o metamodelo de TAO que é uma instância do metametamodelo de ER e o chamamos de *framework* conceitual em (Silva et al., 2003).

O metamodelo de UML especifica uma linguagem de modelagem que é consenso da comunidade de orientação a objetos em relação aos principais conceitos de modelagem. O metamodelo de TAO especifica os principais conceitos de SMAs (abstrações e seus relacionamentos) que incorporam os conceitos orientados a objetos apresentados na UML e os novos conceitos especificamente definidos para o desenvolvimento orientado a agentes com base em nossa experiência e em trabalhos descritos na literatura. Nossa proposta é criar um metamodelo da MAS-ML que estenda o metamodelo de UML de acordo com os conceitos descritos no metamodelo de TAO, conforme ilustrado na segunda camada da Figura 1. A MAS-ML especifica uma linguagem de modelagem que incorpora os conceitos orientados a agentes e objetos. A camada modelo de domínio (terceira camada na Figura 1) descreve os dados específicos ao domínio da aplicação. Os metadados na camada de metamodelo são instanciados em dados por meio dos modelos de domínio, usando informações do domínio. Os conceitos modelados usando a linguagem de modelagem MAS-ML são instanciados de acordo com as informações do domínio, criando os modelos de domínio.

A camada instância (informações ou implementação) (quarta camada na Figura 1) caracteriza as possíveis ocorrências do modelo de domínio. Essa camada descreve as instâncias específicas dos dados do modelo de domínio que podem ocorrer durante o ciclo de vida da aplicação modelada.

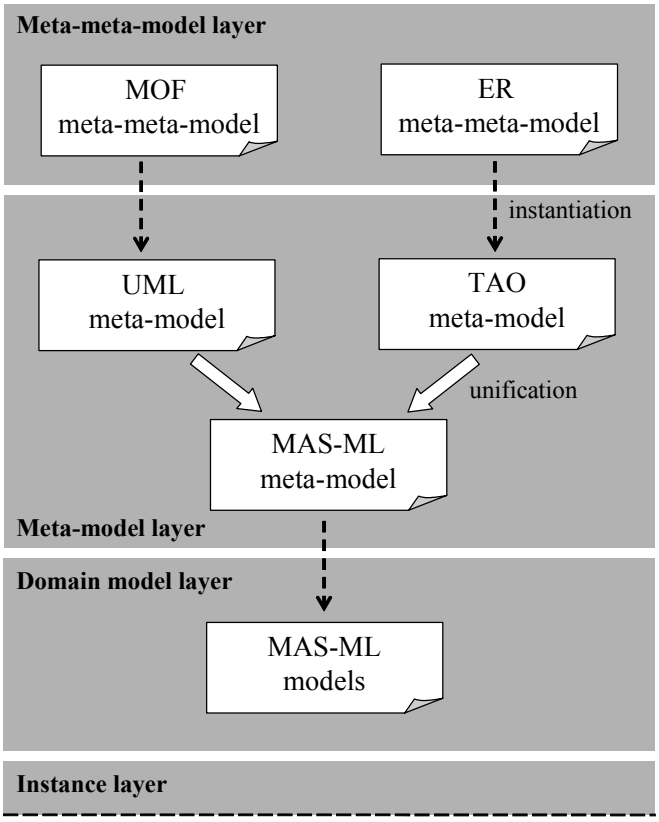

Figura 1 – Arquitetura de metadados com quatro camadas.

1.4. As Principais Contribuições

As principais contribuições desta tese são:

1. O *framework* conceitual TAO – ele define os aspectos estruturais dos SMAs, ou seja, o *framework* conceitual descreve as entidades que são normalmente apresentadas nos SMAs, suas propriedades e um conjunto de relacionamentos que podem ser usados para unir essas entidades.

2. As extensões para o *framework* conceitual TAO – o TAO foi estendido para definir as interações independentes do domínio que podem ocorrer entre as entidades apresentadas nos SMAs.

3. A linguagem de modelagem MAS-ML – ela trata das características particulares dos SMAs que não são tratadas na UML e não são tratadas de forma satisfatória nas propostas disponíveis na literatura. A UML é estendida de acordo com o *framework* conceitual TAO.

4. A abordagem de modelagem – ela ajuda o designer a usar a MAS-ML. Apesar de a MAS-ML ser independente de metodologias, foram fornecidas diretrizes para os designers a fim de ajudá-los a usar essa nova linguagem.

5. A arquitetura abstrata orientada a objetos para SMAs – ela é gerada a partir do metamodelo da MAS-ML. Então, pode ser usada para implementar os modelos desenvolvidos usando a linguagem de modelagem MAS-ML e para implementar qualquer sistema descrito com base no TAO.

6. A gramática da MAS-ML – ela formaliza a sintaxe dos aspectos estruturais da MAS-ML.

7. O transformador MAS-ML2Java – ele refina os diagramas estruturais da MAS-ML gerando código Java e seguindo a arquitetura abstrata OO.

1.5. Descrição da Tese

O Capítulo 2 apresenta os trabalhos relacionados a esta tese. Os *frameworks* conceituais e as linguagens de modelagem para SMAs são analisados, ressaltando seus pontos fracos e suas principais limitações. No Capítulo 3, o *framework* conceitual TAO é descrito definindo os aspectos dinâmicos e estruturais de SMAs. O Capítulo 4 apresenta a linguagem de modelagem MAS-ML. O metamodelo da MAS-ML é descrito junto com os diagramas estruturais e dinâmicos dessa linguagem. O Capítulo 5 descreve o transformador MAS-ML2Java. São apresentadas a gramática da MAS-ML e as regras usadas pelo transformador para gerar código a partir dos diagramas estruturais da MAS-ML. Esse Capítulo também apresenta uma arquitetura abstrata orientada a objetos para implementar SMAs. O Capítulo 6 ilustra o uso da MAS-ML e do MAS-ML2Java. Um exemplo de SMA é modelado usando a MAS-ML, e seus diagramas estruturais são refinados para o código aplicando o transformador MAS-ML2Java. Esse Capítulo também apresenta uma abordagem de modelagem para ajudar os designers no uso da MAS-ML. Finalmente, o Capítulo 7 apresenta as conclusões e os trabalhos futuros. As principais contribuições brevemente apresentadas na Seção 1.4 são melhor descritas no Capítulo 7.