2
Fundamentos teóricos e tecnologias utilizadas

Neste capítulo são apresentados fundamentos teóricos que embasam o trabalho da tese.

2.1.
Preâmbulo

O método proposto na tese é baseado em ontologias e se destina a prover meios à estruturação e análise de modelos de sistemas multiagentes (SMAs), descritos numa linguagem de modelagem orientada a agentes. Assim, para um melhor entendimento do método é importante algum conhecimento prévio sobre ontologias, lógica de descrição, SMAs e linguagens de modelagem para SMAs.

A seção 2.2 apresenta noções sobre o que são ontologias, como são desenvolvidas, linguagens de descrição adotadas e sobre o uso de ontologias em engenharia de software. A seção 2.3 apresenta noções sobre lógicas de descrição e sistemas de suporte à este tipo de lógica. Finalmente, na seção 2.4 são apresentadas noções básicas sobre SMAs e sobre a engenharia de SMAs, em especial sobre linguagens de modelagem para este tipo de sistemas.

2.2.
Ontologias

O termo Ontologia tem origem na Filosofia, e é considerado um dos ramos mais importantes da metafísica: Ontologia engloba o estudo da existência. Assim, ela consiste da pesquisa sobre as coisas que existem, a partir da descrição e classificação destas coisas em categorias. Além disso, também estabelece como estas coisas se relacionam.

Em Ciência da Computação uma das primeiras referências ao termo foi feita em 1991 por Neches e colaboradores (Neches et al, 1991). Partindo da idéia de componentes reutilizáveis de conhecimento como uma forma de facilitar

a construção de sistemas baseados em conhecimento, os autores chegam ao termo ontologia como uma forma de descrever o que seriam os tais componentes reutilizáveis.

"*Building knowledge-based systems today usually entails constructing new knowledge-bases from scratch. It could be done by assembling reusable components.... An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary"* (Neches et al, 1991).

Esta descrição se assemelha à descrição de um dicionário de dados, porém o diferencial está na existência de regras que permitem a combinação dos termos e relações a fim de inferir extensões do vocabulário descrito.

Gruber (1993) apresentou a definição mais citada na literatura: uma ontologia é uma especificação explícita de uma conceitualização. Posteriormente, Gómez-Perez (1999) fez um resumo das várias definições encontradas na literatura. Embora estas definições sejam, em geral, vagas, pode-se definir uma ontologia como uma representação formal para um domínio de conhecimento, representação esta que consiste das entidades que compõem este domínio e dos relacionamentos envolvendo estas entidades, acrescidos de regras que determinam restrições sobre estes relacionamentos e possibilitam inferir conhecimento novo sobre o domínio considerado.

A disseminação do uso de ontologias pela comunidade de engenharia de software foi alimentada, em grande parte, pela crescente demanda por sistemas baseados na *World Wide Web* (*www*). Esta demanda deveu-se, entre outros motivos, à globalização e à adoção da *www* como parte da vida diária de um número cada vez maior de usuários. A interoperabilidade entre vários destes sistemas, e a extensão da *www* para uma *www* semântica foram o gatilho para o uso de ontologias como artefatos de engenharia de software.

Para auxiliar no desenvolvimento de ontologias por não especialistas em engenharia de conhecimento, foram desenvolvidos vários métodos e metodologias para desenvolvimento de ontologias (Uschold & King, 1995; Bernaras et al, 1996 e Gómez-Pérez, 1996). Basicamente as metodologias dividem o desenvolvimento em etapas de definição de propósito (por quê e para quê), definição dos conceitos e relacionamentos, refinamento dos conceitos e relacionamentos e codificação dos conceitos e relacionamentos em alguma linguagem de descrição formal.

Qualquer linguagem formal pode ser utilizada para descrever ontologias. Porém linguagens concebidas para representação de conhecimento são mais adequadas, especialmente quando se pretende fazer uso do aparato dedutivo disponibilizado pelos sistemas de representação de conhecimento. Este aparato provê, em geral, a verificação de consistência da ontologia e a classificação de seus conceitos através da verificação de relacionamentos hierárquicos entre eles.

Linguagens formais para representação de conhecimento são baseadas em lógica, porém, dadas as características dos sistemas de representação de conhecimento, é desejável considerar subconjuntos da lógica que são decidíveis e capazes de expressar satisfatoriamente o conhecimento desejado. Estes subconjuntos da lógica são chamados lógicas de descrição ou *Description Logics* (DL), e são introduzidos na seção 2.3.1.

Atualmente existe uma proposta de padronização de uso da linguagem de descrição de ontologias OWL (*Ontology Web Language*) (W3C, 2004c), feita pelo *W3C consortium*. OWL é uma linguagem de marcação semântica, baseada em RDF (*Resource Description Framework*) (W3C, 2004d) e DL, que possui três sub-linguagens: OWL-Lite, OWL-DL e OWL-Full, cada uma com um grau de expressividade diferente. Para informações adicionais sobre o tema indica-se (W3C, 2004a, 2004b).

Nesta tese optou-se pelo uso da linguagem que descreve a lógica de descrição implementada por um sistema de representação de conhecimento, pois quando foram iniciados os testes relacionados ao método os principais sistemas de representação de conhecimento disponíveis (Powerloom, 2005; FaCT, 2003; RACER, 2004), apresentados na seção 2.3.2.1, ainda não dispunham de vários serviços de inferência para a linguagem OWL, especialmente os relacionados às consultas sobre instâncias das ontologias.

2.3.
Lógicas de descrição e sistemas baseados em lógica de descrição

Lógicas de descrição são formalismos para representação do conhecimento de um domínio de aplicação (o mundo), a partir da definição de conceitos relevantes do domínio (a terminologia) e usando estes conceitos para especificar propriedades dos objetos e indivíduos que fazem parte deste domínio

(a descrição do mundo) (Baader et al, 2003). Como formalismos de representação baseados em lógica, a semântica das DLs é baseada em lógica de primeira ordem. Outra característica importante das DLs é a ênfase dada aos serviços de inferência, que permitem inferir conhecimento implícito a partir de conhecimento explicitamente representado, além de classificar hierarquicamente os conceitos que compõem a terminologia.

Como a todo formalismo para representação de conhecimento deve estar associado um sistema de representação de conhecimento que fornece os serviços de inferência, é desejável que estes sistemas resolvam os problemas de inferência (consultas) retornando sempre uma resposta, seja ela positiva ou negativa. A questão da complexidade associada aos serviços de inferência destes sistemas é importante motivo de pesquisa, pois a garantia que o programa termina em tempo finito não significa que a resposta seja retornada em tempo razoável. A decidibilidade dos serviços de inferência depende da expressividade da DL considerada. Quanto mais expressiva a lógica de descrição, maior a complexidade associada aos problemas de inferência relacionados a ela.

A expressividade de uma lógica de descrição é medida pela família de linguagens de descrição que pode representar suas fórmulas. Assim, a família com menor expressividade é chamada AL (*attributive language*), e suporta descrições que usam conceitos atômicos, negação atômica, interseção de conceitos, quantificador universal (restrição de valor) e quantificador existencial limitado. Considere *C* e *D* conceitos descritos em AL, *R* uma propriedade atômica e A um conceito atômico. A Figura 1 mostra a sintaxe de descrição de conceitos usando AL.

$C, D \rightarrow$	$A \mid$	(conceito atômico)
	$\top \mid$	(conceito universal)
	$\bot \mid$	(conceito vazio)
	$\neg A \mid$	(negação atômica)
	$C \cap D \mid$	(interseção)
	$\forall R.C \mid$	(restrição de valor)
	$\exists R.\top$	(quantificador existencial limitado)

Figura 1. Sintaxe de descrição de conceitos para a família AL de lógicas de descrição.

Esta família pode ser estendida com a adição, por exemplo, da união de conceitos (ALU), ou com o quantificador existencial completo (ALE), com restrições numéricas (ALN), ou a negação de conceitos arbitrários (ALC), ou

qualquer combinação de AL com um conjunto formado pelas extensões descritas, dependendo do tipo de conhecimento que se pretende descrever.

2.3.1.
Formalismo básico

Um sistema de representação de conhecimento baseado em DL (doravante sistema de representação de conhecimento) possui componentes para criar bases de conhecimento, descritas numa linguagem baseada em DL. Além disso, também deve prover serviços de manipulação e consulta a estas bases de conhecimento. Uma base de conhecimento é formada por dois componentes: a terminologia (TBox) e a descrição do mundo (ABox).

A terminologia é formada pelos conceitos atômicos, que denotam conjuntos de indivíduos, as propriedades atômicas, que denotam relacionamentos binários entre indivíduos e, eventualmente, por axiomas que descrevem conceitos e propriedades complexas, que utilizam os conceitos e propriedades atômicas. Dada a definição para terminologia, pode-se dizer que uma terminologia é a representação formal de uma ontologia.

Cada sistema de representação de conhecimento implementa sua própria linguagem de descrição, porém suas fórmulas são facilmente identificáveis com fórmulas em lógica de primeira ordem. Nesta tese optou-se pela utilização da linguagem de descrição adotada pelo sistema RACER (RACER, 2004), o sistema escolhido como aparato dedutivo, definida como ALCHIQr+(D)-. A Tabela 1 descreve a sintaxe e a semântica das fórmulas usadas ao longo deste texto. Considere *C* e *D* conceitos e *R* uma propriedade (relacionamento entre conceitos). Dada a DL considerada, deve-se observar que *R* admite inversão e transitividade, além da possibilidade de definição de propriedades hierarquicamente.

Tabela 1. Sintaxe da linguagem de descrição do sistema RACER

Sintaxe	Semântica
`(some R C)`	Conjunto composto por indivíduos que estão relacionados através do relacionamento R a pelo menos um indivíduo denotado pelo conceito C.
`(all R C)`	Conjunto composto por indivíduos que estão relacionados através do relacionamento R a indivíduos denotados pelo conceito C, e só a eles.

(and C D)	Conjunto dos indivíduos obtidos através da interseção dos conjuntos de indivíduos denotados pelos conceitos C e D.
(or C D)	Conjunto composto por indivíduos denotados pelo conceito C ou pelo conceito D.
(not C)	Conjunto de indivíduos não denotados pelo conceito C.
(top)	Conceito universal (verdade).
(bottom)	Conceito vazio (falso).
(implies C D)	O conjunto de indivíduos denotados pelo conceito C é subconjunto do conjunto de indivíduos denotados pelo conceito D
(equivalent C D)	O conjunto de indivíduos denotados pelo conceito C é equivalente ao conjunto de indivíduos denotados pelo conceito D

Como exemplo de uso da terminologia para descrição de conceitos complexos, considere o domínio da família e das relações de parentesco, partindo dos conceitos atômicos homem (`man`), mulher (`woman`) e pessoa (`person`). A Figura 2 ilustra a descrição de homem e mulher como indivíduos que são pessoas, mãe (`mother`) como indivíduo que é mulher, pai (`father`) como indivíduo que é homem, pais (`parent`) como indivíduos que são mãe ou pai. Além disso, axiomas acrescentam semântica ao conceito mãe, informando que é um indivíduo relacionado a pelo menos uma pessoa através do relacionamento `has-child`.

```
(implies woman person)
(implies man person)
(implies mother woman)
(implies father man)
(equivalent parent (or mother father))
(implies mother (some has-child person))
(equivalent grandmother(and mother (some has-child parent)))
```

Figura 2 Descrição de uma terminologia (TBox) em DL.

A descrição do mundo (ABox) pode ser definida como a instanciação, parcial ou total, dos conceitos e relacionamentos definidos na terminologia. Como uma terminologia é a representação formal de uma ontologia, a descrição do mundo (ou ABox) é a representação formal de instâncias da ontologia. A Figura 3 ilustra exemplos de instanciação de conceitos e relacionamentos para a terminologia descrita na Figura 2.

```
(instance maria mother)
(instance joao father)
```

```
(instance ana woman)
(instance maria grandmother)
(related maria ana has-child)
```

Figura 3 Instanciação de TBox – descrição de mundo (ABox) em DL.

Na próxima seção são apresentados alguns sistemas de representação de conhecimento, com ênfase para o sistema RACER, juntamente com as funcionalidades que eles dispõem e que são utilizadas pelo método Observed-MAS.

2.3.2. Sistemas de representação de conhecimento

2.3.2.1. O sistema RACER

RACER (*Renamed Abox and Concept Expression Reasoner*) é um sistema de representação de conhecimento baseado em DL. A DL implementada no sistema é conhecida como SHIQ, a qual é, de fato, ALC estendida com restrições numéricas de qualificadores, hierarquia de propriedades, propriedades inversas, e propriedades transitivas (ALCHIQr+(D)-). A versão do sistema utilizada nesta tese foi a RACER 1.7.23, obtida gratuitamente em julho de 2005, a partir da URL http://www.sts.tu-harburg.de/~r.f.moeller/racer/. Atualmente o sistema só disponibiliza versões gratuitas para fins acadêmicos, e que podem ser solicitadas no endereço http://www.racer-systems.com/ .

O sistema RACER provê suporte para especificações de terminologias em geral, através do uso de axiomas, inclusive axiomas que descrevam a relação hierárquica entre conceitos, definições múltiplas ou cíclicas. Dentre os serviços de inferência fornecidos pelo sistema para as terminologias cita-se a verificação de consistência da terminologia (TBox) e a verificação da existência de ciclos na terminologia, pois apesar do sistema prover suporte à terminologias cíclicas, a inferência é sempre mais eficiente quando a base de conhecimento é estruturada por uma TBox acíclica. Relativamente à ABox, o sistema também provê a verificação de consistência baseada em alguma TBox, além de várias funções e serviços de recuperação de informação pré-programados. Para a recuperação de informação foi definida uma linguagem de consulta, chamada nRQL (*new RACER Query Language*) (Haarslev et al, 2004). Esta linguagem foi

utilizada para descrever as consultas usadas nas etapas de análise em ambas as fases do método Observed-MAS e é descrita na próxima subseção.

A linguagem de consulta nRQL

A linguagem nRQL pode ser considerada uma extensão dos mecanismos de consulta a ABoxes disponibilizado pelo RACER. Ela possibilita o uso de variáveis no corpo das consultas e também a definição de consultas complexas, as quais podem envolver interseção, união ou negação de conjuntos de indivíduos definidos por conceitos. As variáveis usadas nas consultas são, usualmente, determinadas pelos indivíduos na ABox que satisfazem a consulta, porém é possível usar indivíduos da ABox diretamente nas consultas.

nRQL foi a linguagem utilizada nesta tese para definir as consultas que analisam os modelos de SMAs transformados em instâncias de ontologias que, juntamente com as ontologias formam bases de conhecimento. A seguir apresentam-se exemplos de consultas feitas usando-se a linguagem nRQL.

Consultas sobre conceitos (Figura 4): recupera todas as instâncias do conceito determinado por `?x` na base de conhecimento.

```
(retrieve (?x) (?x woman))
```

Figura 4. Exemplo de consulta sobre conceitos.

Consultas sobre propriedades: recupera na base de conhecimento informações sobre conceitos relacionados através de uma propriedade. A Figura 5 retorna como resposta os pares mãe e filho que constam da base de conhecimento.

```
(retrieve (?mom ?kid) (?mon ?kid has-child))
```

Figura 5. Exemplo de consulta sobre propriedades I.

Também é possível recuperar informações sobre um determinado indivíduo da base de conhecimento. A Figura 6 ilustra um exemplo deste tipo de consulta.

```
(retrieve (?kid) (maria ?kid has-child))
```

Figura 6. Exemplo de consulta sobre propriedades II.

Consultas complexas: envolvem os operadores `AND`, `OR` ou `NOT`. A Figura 7 e a Figura 8 ilustram exemplos de consulta usando `AND`. A Figura 9 e a Figura 10 ilustram exemplos de consulta usando `OR`. Exemplos de utilização do `NOT` são ilustrados na Figura 11, na Figura 12 e na Figura 13.

```
(retrieve (?mom ?kid)
  (and (?mom mother) (?kid man) (?mom ?kid has-child))
```

Figura 7. Exemplo de consulta complexa usando `AND` - I.

A consulta da Figura 7 retorna como resposta pares de mãe e filho do sexo masculino, e a consulta da Figura 8 retorna triplas compostas por mãe e dois filhos de nomes diferentes, pois o sistema adota o critério de *Unique Name Assumption* (UNA) para variáveis. Para contornar o critério de UNA, basta adornar a variável com um cifrão, por exemplo, `$?kid1`.

```
(retrieve (?mom ?kid1 ?kid2)
  (and (?mom mother)
       (?kid1 person)
       (?kid2 person)
       (?mom ?kid1 has-child)
       (?mom ?kid2 has-child)))
```

Figura 8. Exemplo de consulta complexa usando `AND` - II.

Consultas envolvendo `OR` podem ser descritas de duas maneiras: uma usando apenas uma variável, que pode representar qualquer indivíduo denotado pelos conceitos participantes da união, e outra usando duas ou mais variáveis, cada uma representando indivíduos denotados por um dos conceitos participantes da união.

```
(retrieve (?x)
  (or (?x man) (?x woman)))
```

Figura 9. Exemplo de consulta complexa usando `OR` - I.

A Figura 9 ilustra um exemplo de consulta usando `OR` cuja resposta retornada engloba todos os indivíduos sabidamente pertencentes aos conceitos `man` e `woman`, na base de conhecimento. A Figura 10 ilustra um exemplo de consulta usando `OR` e a forma como ela é internamente processada pelo sistema. Sua resposta consiste de todas as combinações de pares (`man`, `woman`) presentes na base de conhecimento.

```
(retrieve (?x ?y)
  (or (?x man) (?y woman)))

(retrieve (?x ?y)
  (or (and (?x man) (?y top))
      (and (?x top) (?y woman))))
```

Figura 10. Exemplo de consulta complexa usando `OR` - II.

Consultas envolvendo negação também possuem suas particularidades. Se o interesse da consulta é recuperar indivíduos que o sistema não consegue provar que pertencem a determinado conceito, utiliza-se a negação conforme ilustrado na Figura 11.

```
(retrieve (?x) (not (?x grandmother)))
```

Figura 11. Exemplo de consulta complexa usando `NOT` - I.

Quando o interesse de uso da negação reside na informação sobre indivíduos que não foram explicitamente modelados na base de conhecimento

como pertencendo a determinado conceito, utiliza-se a negação conforme ilustrado na Figura 12.

```
(retrieve (?x) (not (?x (some has-child top))))
```

Figura 12. Exemplo de consulta complexa usando `NOT` - II.

Também é possível usar a negação para propriedades binárias, como ilustrado na Figura 13. Esta consulta retorna como resposta os pares (`man`,`woman`) tais que a mulher não é mãe do homem.

```
(retrieve (?x ?y)
  (and (?x man)
       (?y woman)
       (not (?y ?x has-child))))
```

Figura 13. Exemplo de consulta complexa usando `NOT` - III.

Combinações de consultas complexas também podem ser descritas pela linguagem. Durante a definição das consultas que auxiliaram nas etapas de análise do Observed-MAS foram feitas várias combinações de consultas complexas, gerando consultas mais complexas. Todas as consultas definidas são encontradas nos Apêndices B, D, E e H.

2.3.2.2. O sistema PowerLoom™

O sistema PowerLoom™ (Powerloom, 2005) é um sistema de representação de conhecimento que implementa uma linguagem de representação de conhecimento baseada em lógica (variante de KIF (Genesereth, 2004)) e provê um ambiente para construção de aplicações que requerem o uso de mecanismos de inferência. A máquina de inferência implementada pelo PowerLoom usa técnicas de dedução natural conhecidas como encadeamento para trás e para frente (*backward and forward chaining*), e pode tratar regras complexas, negação, igualdade, subsunção, dentre outros. O sistema tem sido desenvolvido no Instituto de Ciências da Informação, Divisão de Sistemas Inteligentes, da Universidade do Sul da Califórnia, nos Estados Unidos.

Este sistema foi preterido em relação ao RACER pois o PowerLoom não possuía *plugin* para o editor de ontologias Protégé (Protégé, 2004), usado durante a construção da ontologia que estruturava o método em sua primeira versão (Brandão et al, 2004).

2.3.2.3. O sistema FaCT

O sistema FaCT (*Fast Classification of Terminologies*) (FaCT, 2003) é um classificador de DL que também pode ser usado para teste de satisfabilidade em lógica modal. O sistema possui dois aparatos dedutivos, um para a DL SHF (ALC mais transitividade, propriedades funcionais e hierarquia de propriedades), e o outro para a DL SHIQ (SHF mais propriedades inversas e restrições numéricas de qualificador existencial). Além de implementar uma DL com alto grau de expressividade, o sistema provê suporte para raciocínio envolvendo bases de conhecimento que contenham axiomas gerais de inclusão de conceitos.

Trata-se de um sistema cujo código fonte é disponível (licença pública geral GNU) e pode ser obtido gratuitamente a partir da URL http://www.cs.man.ac.uk/~horrocks/FaCT/.

Este sistema foi preterido em relação ao RACER pois, apesar de possuir *plugin* para o Protégé, durante a realização dos primeiros testes envolvendo instâncias da ontologia o mesmo não dispunha de alguns serviços de inferência que seriam utilizados.

2.4. Sistemas Multiagentes – SMAs

Agentes de software são entidades autônomas, que possuem metas, se comunicam e se organizam enquanto inseridos num ambiente. Sistemas multiagentes são sistemas cujos componentes são agentes de software. Estes sistemas tiveram origem em Inteligência Artificial, mais especificamente na sub-área de inteligência artificial distribuída (Weiss, 1999), como uma metáfora para a resolução de problemas baseada no comportamento social do sistema, onde o conhecimento está distribuído entre os agentes, que se organizam e cooperam para atingir um objetivo comum. Neste contexto, a noção de SMAs e as teorias associadas a este tipo de sistemas trazem consigo noções sobre várias propriedades que seus componentes possuem, tais como autonomia, inteligência, capacidade de organização, cooperação e mobilidade, dentre outras. Tais propriedades acrescentam complexidade ao desenvolvimento de

SMAs e impulsionam a pesquisa relacionada à engenharia deste tipo de sistemas.

Do ponto de vista da Engenharia de Software, a pesquisa relacionada a SMAs, a qual será referida ao longo do texto como engenharia de software de sistemas multiagentes (ESSMA), está em franco desenvolvimento. Este desenvolvimento pode ser medido pela realização, nos últimos quatro anos, de *workshops* sobre engenharia destes sistemas (*Software Engineering for Large-Scale Multi-Agent-Systems* – SELMAS[1], e *Agent-Oriented Software Engineering* - AOSE[2]) em importantes conferências internacionais como a *International Conference on Software Engineering* (ICSE) e a *International Joint-Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS). No contexto nacional, a edição deste ano do Simpósio Brasileiro de Engenharia de Software (SBES 2005) contou com o primeiro workshop brasileiro na área, o *Software Engineering for Agent-oriented Systems* – SEAS 2005[3].

Grande parte da pesquisa produzida em ESSMA refere-se às atividades de projeto, como o desenvolvimento de linguagens de modelagem e metodologias de desenvolvimento de SMAs (Wooldridge et al, 2000; Bauer et al, 2001; Muller et al, 2001; Wood & Deloach, 2001; Wagner, 2003; SIlva & Lucena, 2004); à construção de padrões de arquitetura (Sycara et al, 2003; FIPA, 2004) e à produção de infra-estrutura básica de desenvolvimento (Collis et al, 1998; Bellifemine et al 1999; Bellifemine et al, 2003; Sycara et al, 2003). O método proposto nesta tese pode ser contextualizado como pertencente à pesquisa relacionada às atividades de projeto de SMAs, na medida em que provê suporte à estruturação e análise de modelos de SMAs descritos em linguagens de modelagem.

2.5.
Linguagens de modelagens orientadas a agentes

O desenvolvimento de SMAs envolve um grau de complexidade alto dadas as características específicas dos agentes de software que os compõem. Estas características podem ser explicadas pelas novas entidades adicionadas a estes sistemas, como, por exemplo, agentes, organizações, papéis e ambiente

[1] http://www.teccomm.les.inf.puc-rio.br/selmas2005/

[2] http://www.jamesodell.com/aose2004

[3] http://www.les.inf.puc-rio.br/seas2005/

(Shoham, 1993; Yu e Schimid, 1999; Wooldridge e Ciancarini, 2001; Zambonelli et al, 2001; Odell et al, 2003); pelos novos relacionamentos definidos entre estas entidades e pelas novas propriedades inerentes a estas entidades. Desta forma, linguagens de modelagem que possibilitem a descrição das especificidades inerentes a SMAs podem beneficiar o desenvolvimento destes sistemas ao definir diagramas capazes de representar todas as suas características. A seguir são apresentadas, brevemente, linguagens de modelagem que estendem UML para descrever, de forma mais adequada, modelos de SMAs.

2.5.1. MAS-ML

MAS-ML (*Multi-Agent Systems Modeling Language*) (Silva e Lucena, 2004) é uma linguagem de modelagem que estende UML através da introdução de novas metaclasses ao metamodelo de UML, que descrevem as novas entidades que compõem um SMA, assim como os novos relacionamentos definidos entre estas entidades, e entre estas entidades e objetos. O metamodelo da linguagem define dois novos diagramas estáticos (diagramas de organização e de papéis) e estende o diagrama de classes definido pelo metamodelo de UML, a fim de representar diferentes visões de um SMA. Além dos diagramas estáticos, também são estendidos os diagramas dinâmicos de seqüência e atividades de UML, a fim de descrever a interação entre as várias entidades que participam de um SMA, bem como descrever a estrutura de protocolos e planos. Esta linguagem foi usada como prova de conceito do método Observed-MAS e é tratada com mais detalhes no Capítulo 5, seção 5.2.

2.5.2. AUML

AUML (*Agent UML*) (Odell et al, 2000; Bauer, 2002; Bauer et al, 2002; Huget, 2002a; 2002b; Parunak & Odell, 2002; Odell et al, 2003) é uma linguagem de modelagem que estende UML a fim de descrever os aspectos relacionados às especificidades de agentes de software. A extensão de UML é obtida através da extensão de vários dos diagramas definidos pelo metamodelo de UML, tais como os diagramas de classe, de seqüência, de atividades e de colaboração. As entidades especificadas pela linguagem são agentes, papéis e organizações (ou

grupos). Diagramas de classe foram estendidos para descrever relacionamentos entre papéis; ou entre organizações e papéis; ou relacionamentos entre agentes (Bauer, 2002). Diagramas de seqüência foram estendidos para descrever protocolos definidos no contexto dos papéis e interações entre agentes desempenhando papéis. Parte dos diagramas definidos nesta linguagem foi utilizada para uma segunda aplicação do método Observed-MAS (Capítulo 7), a fim de analisar a dependência do mesmo à ontologia que define as abstrações relacionadas ao domínio de SMAs. Assim, no Capítulo 7 a linguagem será abordada em maiores detalhes.

2.5.3. AORML

A linguagem AORML (*Agent Oriented Rule Markup Language*) é baseada no metamodelo AOR (*Agent-Object-Relationship*) (Wagner, 2002 e 2003), e define como entidades agente, objeto, evento, ação, solicitação e compromisso. Os agentes são classificados em sete tipos diferentes. Os agentes institucionais representam organizações, as quais consistem de agentes internos que percebem eventos e executam ações enquanto desempenham papéis. Os agentes internos executam ações em prol do agente institucional, a fim de fazer uso de seus direitos e cumprir seus deveres. AORML estende os relacionamentos *association*, *generalization* e *composition* definido pelo metamodelo de UML e define novos relacionamentos chamados *does*, *perceives*, *sends*, *receives*, *hasClaim* e *hasCommitment*. Os novos relacionamentos relacionam as entidades ação, evento, compromisso, solicitação e agente.

Uma modelagem em AORML pode ser composta por dois tipos de modelos: um modelo externo e um modelo interno. O modelo externo corresponde a um modelo conceitual, voltado para a análise do domínio da aplicação. O modelo interno corresponde a um modelo de projeto da aplicação.

Modelos externos são classificados em diagramas de agentes, diagramas de estrutura de interação, diagrama de sequência de interação e diagrama de padrão de interação. Os diagramas de agentes são compostos por classes de agentes referentes ao domínio da aplicação, assim como classes de objetos que se relacionam com os agentes. Diagramas de estrutura de interação ilustram classes de eventos ou ações, assim como classes de compromissos/solicitações que descrevam possíveis interações entre dois agentes. Os diagramas de

seqüência de interação definem protótipos de instâncias de processos de interação.

Modelos internos são classificados em diagramas de estrutura de reação, diagramas de seqüência de reação e diagramas de padrão de reação. Os diagramas de estrutura de reação descrevem classes de agentes, ações, eventos, solicitações e compromissos que determinam possíveis interações entre os agentes. Os diagramas de seqüência de reação descrevem protótipos de instâncias de processos internos de interação. Finalmente, os diagramas de padrão de reação descrevem os padrões de reação dos agentes, os quais são descritos através de regras de reação.