UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIA E TECNOLOGIA
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

EMMANUEL SÁVIO SILVA FREIRE

NorMAS-ML: SUPORTE À MODELAGEM DE SISTEMAS MULTI-AGENTE NORMATIVOS

FORTALEZA – CEARÁ
2012

EMMANUEL SÁVIO SILVA FREIRE

NorMAS-ML: SUPORTE À MODELAGEM DE SISTEMAS MULTI-AGENTE NORMATIVOS

Dissertação submetida à Comissão Examinadora do Programa de Pós-Graduação Acadêmica em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientação:
Professora Dra. Mariela Inés Cortés

FORTALEZA - CEARÁ
2012

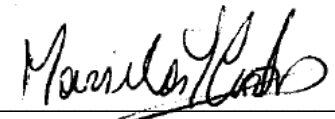
EMMANUEL SÁVIO SILVA FREIRE

NorMAS-ML: SUPORTE À MODELAGEM DE SISTEMAS MULTI-AGENTE NORMATIVOS

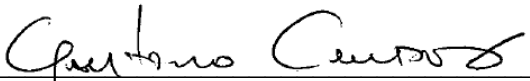
Dissertação submetida à Comissão Examinadora do Programa de Pós-Graduação do Mestrado Acadêmico em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovada em ____ / ____ / _____

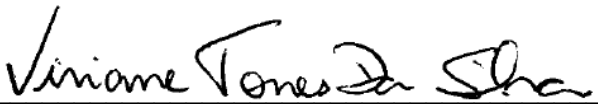
BANCA EXAMINADORA



Professora Dra. Mariela Inés Cortés (Orientadora)
Universidade Estadual do Ceará – UECE



Professor Dr. Gustavo Augusto de Lima Campos
Universidade Estadual do Ceará – UECE



Professora Dra. Viviane Torres da Silva
Universidade Federal Fluminense – UFF

Aos meus pais, Aderbal e Francisca, por serem meus maiores exemplos e incentivadores.

AGRADECIMENTOS

Inicialmente a Deus, por sempre se fazer presente em todos os momentos da minha vida, me amparando e sempre mostrando o caminho certo a seguir.

A todos os meus familiares, especialmente aos meus pais e a minha irmã Cibely Freire, pela paciência, por entenderem a minha ausência, pelo incentivo e pela ajuda durante o tempo da realização deste trabalho.

A minha orientadora Profª Mariela Cortés, por acreditar no meu trabalho, guiar meus estudos e principalmente pela paciência na correção de artigos e desta dissertação. Muito obrigado por ser tão presente durante o mestrado.

Aos Professores Enyo Gonçalves e Gustavo Campos, agradeço a colaboração, a paciência na correção de artigos e pelas valiosas discussões que contribuíram muito para a realização desta pesquisa.

A profª Viviane Torres, agradeço pela contribuição na definição do tema da minha pesquisa, pela disponibilidade em tirar dúvidas e pela correção de artigos.

A todos os meus colegas do grupo de pesquisa GESSI e do LAPAQ, especialmente a Állan Ribeiro, Robson Oliveira e Robert Marinho. Obrigado pela ajuda na extensão da Ferramenta MAS-ML *tool*.

Ao meu colega de mestrado Yrleyjânder Lopes, agradeço a parceria nas disciplinas e na escrita de artigos e na disponibilidade em tirar dúvidas.

Aos meus grandes amigos, Laysa Cunha, Riselha Marques, Paulo Augusto, Juliana Viana, Geovane Freire, Alana Joyce, Daniele Delmiro, Shandy Cardoso, Paula Sousa, Fábio Mendes e Lilian Segundo. Obrigado por fazerem parte da minha vida.

Aos meus amigos que trabalharam comigo no Centro de Gestão e Desenvolvimento Tecnológico, especialmente a Grasielle Queiroz, Deborah Pontes, Ygor Sampaio, Grazielly Moura, Carlos Andros, Daniele Damasceno, Adriana Lacerda e Thiago Leite. Obrigado pelos ensinamentos e conselhos dados durante o meu período de aluno especial.

Aos Professores Antônio Serra e Hairon Gonçalves, agradeço pelo apoio e confiança depositada desde o tempo da minha graduação no IFCE.

A todos os funcionários e professores que fazem parte do Mestrado Acadêmico em Ciência da Computação da UECE, pelo suporte dado durante o mestrado, principalmente a Darcia e Marcos.

À Fundação Cearense de Amparo a Pesquisa (Funcap) e a CAPES, pelo apoio financeiro parcial.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis."

José de Alencar

RESUMO

A modelagem integrada dos diversos aspectos que constituem um sistema complexo é fundamental para o profundo entendimento da sua estrutura e comportamento. Em Sistemas Multi-Agente (SMAs), o comportamento de agentes é governado por normas, cujos elementos que as compõem precisam ser modelados adequadamente em tempo de *design*. Neste contexto, destacamos MAS-ML, uma linguagem de modelagem para SMAs baseada no *framework* conceitual TAO que fornece diversos mecanismos para a modelagem das propriedades das entidades que tipicamente compõem um SMA. Entretanto, o suporte aos conceitos relativos a normas é limitado. Esta dissertação aborda a extensão do *framework* TAO e da linguagem MAS-ML para dar suporte à modelagem de SMA considerando os elementos estáticos das normas. A nova versão de MAS-ML será denominada NorMAS-ML e permitirá uma visão mais completa do sistema através de uma única linguagem de modelagem. Adicionalmente, a ferramenta de suporte MAS-ML *tool* será estendida para possibilitar a modelagem dos novos elementos e do novo diagrama definidos na extensão.

Palavras–Chave: Sistema Multi-agente; Normas; Linguagem de Modelagem; MAS-ML; NorMAS-ML.

ABSTRACT

The integrated modeling of the different aspects that constitute a complex system is essential for a deep understanding of its structure and behavior. In Multi-Agent Systems (MAS), the behavior of agents is governed by norms, which the elements within them need to be modeled properly at design time. In this context, we highlight MAS-ML, a modeling language for MAS based on the conceptual framework TAO that provides several mechanisms to modeling the properties of the entities that tipically compose a SMA. However, the support for the concepts related to norms is limited. This dissertation involves the extension of the framework TAO and the MAS-ML language to support the modeling of SMA considering the static elements of norms. The new version of MAS-ML will be called NorMAS-ML and it will allow a more complete view of the system through a single modeling language. Additionally, the tool support MAS-ML tool will be extended to enable the modeling of the new elements and diagram defined in the extension.

Keywords: Multi-agent System; Norms; Modeling Language; MAS-ML; NorMAS-ML.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AML	*Agent Modeling Language*
AOR	*Agent-Object-Relationship*
AORML	*Agent-Object Relationship Modeling Language*
ASF	*Agent Society Framework*
AUML	*Agent Unified Modeling Language*
CASE	*Computer-Aided Software Engineering*
EMF	*Eclipse Modeling Framework*
EMOF	*Essential Meta-Object Facility*
ES	Engenharia de Software
GMF	*Graphical Modeling Framework*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
MAS-ML	*Multi-agent System Modeling Language*
MAS-ML 2.0	*Multi-Agent System Modeling Language estendida*
MDA	*Model Driven Architecture*
MDD	*Model Driven Development*
Moise+	*Model of Organization for Multi-Agent Systems*
NorMAS-ML	*Normative Multi-Agent System Modeling Language*
NormML	*Normative Modeling Language*
OCL	*Object Constraint Language*
OperA	*Organizations per Agents*
OMG	*Object Management Group*
SMA	Sistema Multi-Agente
SMAs	Sistemas Multi-Agente
SODA	*Societies in Open and Distributed Agent spaces*
TAO	*Taming Agents and Objects*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentados os principais fatores que motivaram o desenvolvimento desta pesquisa, assim como os objetivos a serem alcançados e estabelece como o trabalho está organizado.

1.1 Motivação

O desafio de lidar com a complexidade faz com que engenheiros e desenvolvedores procurem por mecanismos que tornem seus programas cada vez mais "inteligentes", capazes de detectar mudanças no ambiente no qual estão inseridos e decidir pela ação mais adequada (RUSSELL; NORVIG, 2004). Neste contexto, os sistemas centrados em agentes (PADILHA; JÁCOME, 2002) (CASILLO, 2008) vêem sendo amplamente explorados pela comunidade científica como uma abordagem adequada para o desenvolvimento de sistemas computacionais cada vez mais complexos. No entanto, o desenvolvimento deste tipo de sistemas não é trivial, requerendo de um esforço de engenharia no intuito de dar suporte adequado às diversas atividades de desenvolvimento.

Segundo RUSSELL e NORVIG (2004), um agente é uma entidade capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. Os agentes inteligentes podem ser classificados de acordo com a maneira que eles coletam informações e agem no ambiente. No caso de vários agentes cooperando ou disputando entre si, inseridos em um mesmo ambiente e trocando informações, chamamos esse sistema de multi-agente (SMA) (RUSSELL; NORVIG, 2004).

Assim como qualquer sistema de computação, o desenvolvimento de SMA precisa seguir um processo bem definido para a execução das atividades envolvidas de forma a atingir o nível de qualidade e expectativas idealizadas por usuários e projetistas. O processo de desenvolvimento possui suas etapas muito bem definidas, estabelecendo as atividades que serão executadas em cada uma delas, assim como também métodos e ferramentas a serem utilizados de forma a atingir maior produtividade na execução e qualidade no produto (SOMMERVILLE, 2007).

A Engenharia de Software propicia boas práticas para o desenvolvimento de software por meio da utilização de técnicas e ferramentas, através das quais se consegue ter um melhor aproveitamento em cada etapa do processo de desenvolvimento (SOMMERVILLE, 2007) (PRESSMAN, 2009). No contexto do paradigma orientado a objetos, um conjunto de métodos e técnicas auxiliam o seu desenvolvimento e implementação,

como por exemplo, as linguagens de modelagem. A utilização de linguagens de modelagem na fase de análise e projeto é fundamental para a geração dos modelos representando as diferentes visões do sistema a ser desenvolvido (BEZERRA, 2007). Neste contexto, a pesquisa por metodologias, linguagens de modelagem, plataformas de desenvolvimento e linguagens de programação para sistemas multi-agente é de fundamental importância.

De forma geral, agentes são inseridos em ambientes e, se relacionam com outros agentes para alcançar seus objetivos. Neste complexo cenário, o comportamento dos agentes é governado por um conjunto de normas especificado para restringir suas ações. Com isso, a pesquisa em normas para SMAs tem como foco a restrição do comportamento das entidades que fazem parte de um SMA, definindo o que é obrigado, permitido ou proibido de fazer (FIGUEIREDO; SILVA, 2010). Assim sendo, a modelagem das entidades em conjunção com a modelagem das normas que as governam pode ser de valioso auxilio para um entendimento mais abrangente e aprofundado da estrutura e comportamento do sistema.

As normas podem ser definidas em tempo de modelagem ou em tempo de execução. No entanto, a consideração de normas em tempo de modelagem propicia uma visão mais completa do sistema e pode influenciar na modelagem das entidades que integram o sistema. Adicionalmente, algumas verificações e checagens podem ser realizadas ainda durante a fase de definição das normas, e problemas detectados antes mesmo do sistema ser implementado podem ser solucionados (LÓPEZ Y LÓPEZ, 2003).

Dada a importância da modelagem de normas em forma concomitante com a modelagem das demais entidades que participam de um SMA, se torna indispensável à existência de linguagens de modelagem que possibilitem o projeto de SMAs com normas.

Um reduzido número de linguagens de modelagem fornece suporte à modelagem de normas em associação com as entidades previstas em um SMA. Dentre elas, podemos destacar as linguagens MAS-ML (*Multi-agent System Modeling Language*) (SILVA, 2004) e NormML (*Normative Modeling Language*) (FIGUEIREDO, 2011). A primeira apresenta suporte parcial à modelagem dessas entidades em associação a normas e a segunda apresenta suporte à modelagem dos elementos que compõem as normas.

MAS-ML permite a modelagem de todas as entidades previstas em um SMA a partir dos conceitos definidos no *framework* conceitual TAO (*Taming Agents and Objects*) (SILVA et al., 2003). Entretanto, o suporte à modelagem dos conceitos referentes a normas ainda é limitado. Por outro lado, a linguagem NormML (*Normative Modeling Language*) possibilita a modelagem dos principais elementos relacionados às normas e fornece mecanismos para resolver os conflitos entre normas em tempo de *design*. Entretanto, a

linguagem não permite a modelagem dos aspectos estruturais e comportamentais das entidades tipicamente encontradas em um SMA, como agente, organização, papel, ambiente, etc. No entanto, devido à interdependência entre a modelagem de normas e a modelagem das entidades do sistema, a eficácia de NormML depende da integração com uma outra linguagem de modelagem para SMA para apoiar a modelagem de todo o sistema.

Além da existência de uma linguagem de modelagem que permita a modelagem de todos os aspectos estruturais e comportamentais das entidades que compõem os sistemas multi-agente normativos, a existência de uma ferramenta CASE que possibilite a modelagem de SMAs utilizando uma linguagem de modelagem especifica torna-se essencial para a validação dos modelos criados.

1.2 Objetivos

A partir da motivação descrita, este trabalho tem como principal objetivo tornar a linguagem MAS-ML uma linguagem normativa capaz de modelar os elementos que compõem as normas previstos em FIGUEREIDO (2010) através da extensão do *framework* TAO, da linguagem MAS-ML e da ferramenta MAS-ML *tool*. Com a linguagem estendida, NorMAS-ML, será possível a modelagem das normas juntamente com a modelagem das entidades que compõem um SMA.

De forma complementar, tem-se como objetivos específicos:

- Estender o *framework* TAO adicionando novas entidades e possibilitando que entidades preexistentes possam ser adequadamente restringidas pelas normas;
- Estender a sintaxe abstrata da linguagem MAS-ML utilizando como base o reuso e adequação de metaclasses e relacionamentos definidos em NormML;
- Estender a sintaxe concreta da linguagem MAS-ML para definir os elementos gráficos utilizados para representar as novas entidades definidas na sintaxe abstrata;
- Definir um novo diagrama estático para MAS-ML de forma a possibilitar a modelagem das normas em associação com as entidades definidas na linguagem;
- Estender a ferramenta MAS-ML *tool* para possibilitar o suporte à modelagem do novo diagrama estrutural;
- Ilustrar a adequação da extensão realizada com base na modelagem de estudos de caso.

1.3 Organização do Trabalho

O conteúdo deste trabalho é apresentado ao longo de sete capítulos, incluindo a presente introdução, os quais são descritos sucintamente a seguir.

Capítulo 2 – Fundamentação Teórica: Discorre sobre os conceitos relacionados com agentes e sistemas multi-agente. Adicionalmente, são apresentados o *framework* TAO e as linguagens de modelagem MAS-ML e NormML.

Capítulo 3 – Trabalhos Relacionados: Apresenta uma revisão bibliográfica acerca de metodologias, linguagens de modelagem e modelos organizacionais que consideram os conceitos de normas para a restrição do comportamento das entidades que compõem um SMA, destacando suas principais características e limitações.

Capítulo 4 – Estendendo o *Framework* TAO e a Linguagem MAS-ML: Descreve em detalhes a extensão realizada no *framework* conceitual TAO e na linguagem de modelagem MAS-ML, ressaltando o novos elementos que foram definidos e as alterações realizadas nos elementos existentes. Com a extensão de MAS-ML é originada a sua nova versão denominada de NorMAS-ML.

Capítulo 5 – Extensão da Ferramenta de Modelagem: Descreve em detalhes a extensão da ferramenta de modelagem MAS-ML *tool* em consistência com as extensões na linguagem propostas no capitulo anterior.

Capítulo 6 – Estudos de Caso: Apresenta os estudos de caso que ilustram a utilização dos novos elementos definidos na extensão da linguagem MAS-ML.

Capítulo 7 – Considerações Finais: Relaciona as principais contribuições desta pesquisa, assim como as suas limitações. Apresenta também oportunidades para trabalhos futuros e as conclusões deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo fornece a fundamentação teórica sobre os conceitos relacionados ao escopo deste trabalho. Inicialmente, os conceitos relacionados com agentes e sistemas multi-agente são apresentados. Em seguida, o *framework* TAO e as linguagens de modelagem MAS-ML e NormML são descritos. Alguns dos conceitos aqui apresentados são necessários também para o entendimento dos trabalhos relacionados descritos posteriormente.

2.1 Agentes e Sistemas Multi-agente

Os sistemas centrados em agentes (PADILHA; JACOME 2002) (CASILHO, 2008) vêem sendo amplamente explorados pela comunidade científica como uma abordagem adequada para o desenvolvimento de sistemas computacionais complexos. Segundo RUSSELL e NORVIG (2004), um agente de software é uma entidade capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. A Figura 1 representa esquematicamente a interação entre um agente e seu ambiente.

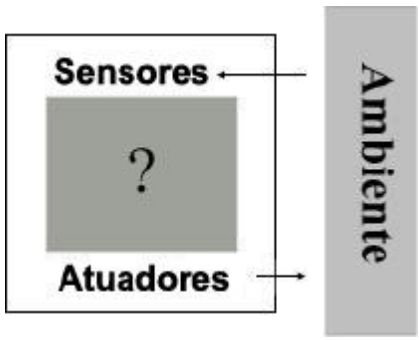


Figura 1 – Agente com seus sensores e atuadores interagindo com o ambiente (GONÇALVES, 2009)

BRADSHAW (1997) estabeleceu o seguinte conjunto de propriedades que os agentes possuem e que são comumente aceitos pelos pesquisadores da área:

- **Autonomia:** a capacidade de executar suas próprias tarefas e alcançar seus próprios objetivos sem necessariamente influência do utilizador;
- **Comportamento colaborativo**: a capacidade de trabalhar em conjunto com outros agentes (por meio da cooperação, negociação, coordenação e delegação de tarefas) para atingir um objetivo comum, ou seja, o objetivo do sistema no qual eles estão interagindo;
- **Reatividade**: a capacidade de perceber os eventos em tempo real e agir em consequência;

- **Comunicabilidade**: a capacidade de se comunicar com os humanos, outros agentes, sistemas legados, e fontes de informação;
- **Personalidade:** a capacidade de manifestar os atributos críveis como as emoções;
- **Adaptabilidade**: a capacidade de aprender e evoluir com base em suas experiências, as experiências de outros agentes e as mudanças do local de acolhimento; e
- **Mobilidade**: a capacidade de se mover de um lugar hospedeiro para outro.

Os agentes inteligentes podem ser classificados de acordo com a maneira através da qual coletam informações e agem no ambiente, ou seja, de acordo com a sua arquitetura. RUSSELL e NORVIG (2004) destacam quatro arquiteturas de agentes inteligentes, a saber: (i) agentes reativos simples, (ii) agentes reativos baseados em conhecimento, (iii) agentes baseados em objetivo e (iv) agentes baseados em utilidade. É possível observar na Figura 1 um símbolo de '?' situado entre as percepções do agente e suas ações, o qual representa a função agente que é diferente para cada arquitetura interna.

No caso de vários agentes cooperando ou disputando entre si, inseridos em um mesmo ambiente e trocando informações, temos um sistema multi-agente (SMAs). Os agentes que compõem esses sistemas podem possuir diferentes arquiteturas internas, porém ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes exibem duas características fundamentais: (i) serem capazes de agir de forma autônoma tomando decisões levando à satisfação dos seus objetivos e (ii) serem capazes de interagir com outros agentes utilizando protocolos de interação social.

2.2 *Taming Agents and Objects* – TAO

O *framework* TAO tem como principal função oferecer uma ontologia que englobe os fundamentos da Engenharia de Software baseada em agentes e objetos. Com isso, é possível oferecer suporte ao desenvolvimento de SMAs de larga escala (SILVA et al., 2003). Os aspectos estáticos e dinâmicos do TAO são descritos a seguir.

2.2.1 Aspectos Estáticos do TAO

A Figura 2 apresenta as entidades definidas pelo metamodelo TAO. A seguir, as mesmas são descritas (SILVA, 2004).

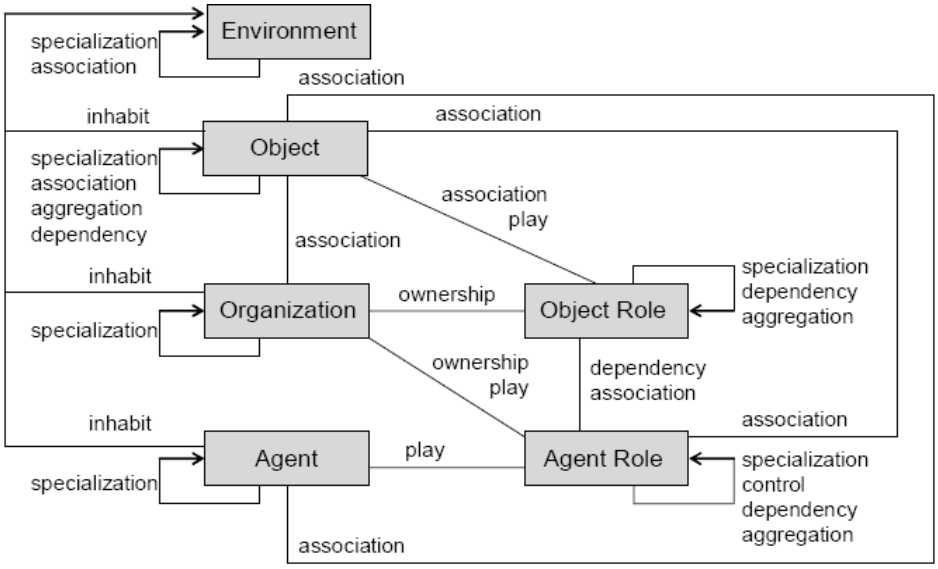


Figura 2 – Os relacionamentos e as entidades do TAO (SILVA, 2004)

- **Objeto**: É um elemento passivo que possui um estado e um comportamento. Durante seu ciclo de vida, ele executa operações que podem modificar seu estado, porém, essas operações podem ser executadas somente quando forem solicitadas por outra entidade.
- **Agente**: É um elemento autônomo, adaptativo e interativo, que possui os seguintes componentes mentais: crenças (tudo que o agente conhece), objetivos (estados futuros que o agente deseja alcançar), planos (seqüências de ações que alcançam um objetivo) e ações.
- **Organização**: É responsável por agrupar agentes, objetos e suborganizações. Uma organização possui objetivos, crenças (como agentes) e axiomas e está situada em um ambiente. Os axiomas caracterizam as restrições globais da organização às quais os agentes e as suborganizações devem obedecer. Uma organização também define os papéis que devem ser exercidos pelos agentes e pelas suborganizações dentro dela e os papéis que devem ser exercidos pelos objetos.
- **Papel de Objeto**: Orienta e restringe o comportamento de um objeto limitando as informações e o comportamento que outras entidades podem acessar. Um papel de objeto pode adicionar informação, comportamento e relacionamentos a instância do objeto que o executa.
- **Papel de Agente**: Orienta e restringe o comportamento de um agente descrevendo seus objetivos ao exercer o papel, definindo as ações que deve exercer e as ações que pode executar ao exercer o papel. Um papel de agente define deveres, direitos

e protocolos. Um dever define uma ação que deve ser executada por um agente; um direito define uma ação que pode ser executada por um agente; e um protocolo define uma interação entre um papel de agente e outros elementos.

- **Ambiente**: É um elemento no qual residem agentes, objetos e organizações.

Adicionalmente, os seguintes relacionamentos foram definidos no TAO (SILVA, 2004):

- ***Inhabit*:** O relacionamento *inhabit* especifica que a instância de entidade que reside – o cidadão – é criada e destruída no habitat e, portanto, pode entrar e sair dele, respeitando suas permissões. Um cidadão não pode residir em dois habitats ao mesmo tempo. O habitat conhece todos os cidadãos que residem nele, e cada cidadão conhece seu habitat. Esse relacionamento aplica-se a ambientes e agentes, a ambientes e objetos e a ambientes e organizações.
- ***Ownership*:** Alguns elementos precisam ser membros de outros elementos. O relacionamento *ownership* especifica que um elemento – o membro – é definido no escopo de outro elemento – o proprietário – e que o membro deve obedecer a um conjunto de restrições globais definidas pelo proprietário. Os membros podem ser criados e destruídos dinamicamente por seu proprietário.
- ***Play*:** Objetos, agentes e suborganizações precisam estar relacionados com papéis. O relacionamento *play* define quais objetos, agentes ou suborganizações estão relacionados com um papel e que devem assumir as propriedades e os relacionamentos definidos por esse papel. O comportamento dessas entidades é guiado e restringido pelo escopo do papel.
- ***Specialization/Generalization*:** O relacionamento *specialization* define que o sub-elemento que especializa o super-elemento pode adicionar e redefinir as propriedades e o comportamento associado com o super-elemento.
- ***Control*:** O relacionamento *control* define que a entidade controlada deve fazer tudo que a entidade do controlador pedir. A entidade do controlador conhece as entidades controladas, e cada entidade controlada conhece as entidades que a controlam. O relacionamento *control* pode ser usado entre dois papéis de agente, porém não pode ser utilizado entre dois papéis de objeto.
- ***Dependency*:** Um elemento – o cliente – pode ser definido para depender de outro – o fornecedor – para realizar seu trabalho. O relacionamento *dependency* define

que o cliente não pode completamente fazer o seu trabalho sem a ajuda do fornecedor.

- ***Association*:** Se um elemento está associado com outro, ele sabe que esse outro elemento existe. O relacionamento *association* deve definir como um elemento interage com outro.
- ***Aggregation*/*Composition*:** Se um elemento é agregado a outro elemento, dizemos que ele é parte de um agregador. O agregador pode utilizar as funcionalidades disponíveis em suas partes. As partes não precisam saber que eles estão sendo agregadas a um agregador, mas o agregador conhece cada uma de suas partes. Dependendo da força de agregação, a parte não pode existir sem o agregador.

2.2.2 Aspectos Dinâmicos do TAO

O TAO também descreve a interação e a execução interna das entidades, destacando os processos comportamentais independentes do domínio. Os processos independentes do domínio estão divididos em dois grupos: processos primitivos e de alto nível. Os aspectos dinâmicos primitivos são compostos pelos processos de criação e destruição. Os processos de alto nível são compostos por (SILVA, 2004): (i) processo de compromisso com um papel, (ii) processo de ativação de um papel, (iii) processo de cancelamento do compromisso com um papel, (iv) processo de desativação de um papel (v) e processo de movimento de um ambiente para outro. Eles descrevem os padrões de comportamento derivados das características dos relacionamentos Habita, Posse e Exerce entre as entidades de SMAs. Também abrangem processos para um agente entrando ou saindo da organização, uma organização entrando ou saindo de outra organização, e um agente ou organização entrando ou saindo de um ambiente (GONÇALVES, 2009).

2.3 MAS-ML

MAS-ML é uma linguagem de modelagem que estende a UML (UML, 2011) para permitir a modelagem de SMAs, com base no *framework* conceitual TAO (SILVA et al., 2003). O processo de extensão utilizado por SILVA (2004) baseou-se na criação de novas metaclasses e estereótipos para representar as entidades que fazem parte dos SMAs. A partir da ontologia definida no TAO, a linguagem MAS-ML é capaz de oferecer suporte ao desenvolvimento de SMAs de larga escala. A Figura 3 apresenta parte do metamodelo da linguagem.

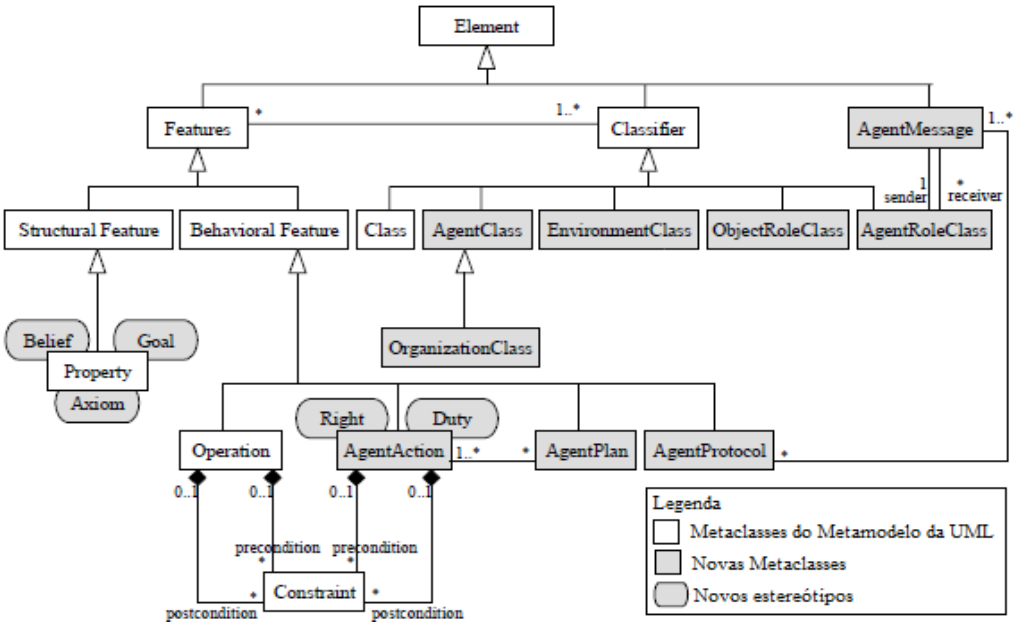


Figura 3 – O metamodelo de MAS-ML e suas propriedades (SILVA, 2004)

2.3.1 Aspectos Estáticos de MAS-ML

A seguir são descritos os diagramas estáticos definidos em MAS-ML.

- **Diagrama de Classes**: MAS-ML estende o diagrama de classes de UML a fim de representar os relacionamentos entre as classes e outras entidades do SMA. O diagrama de classes estendido representa os relacionamentos entre as classes e os ambientes, classes e agentes e classes e organizações. Ademais, ele também foi estendido para representar os relacionamentos entre agentes, entre ambientes e entre organizações.
- **Diagrama de Organização**: Tem como objetivo modelar todas as organizações de um sistema. O diagrama de organização é responsável por modelar uma organização, ou seja, por modelar as propriedades da organização (objetivos, crenças, planos, ações e axiomas), os papéis definidos pela organização, as entidades (agentes, classes e suborganizações) que exercem esses papéis e o ambiente em que ela reside (SILVA, 2004). Neste diagrama são exibidos os relacionamentos Posse, Exerce e Habita.
- **Diagrama de Papéis**: O diagrama de papel é responsável pela ilustração dos relacionamentos entre os papéis do agente e os papéis de objeto identificados nos diagramas de organização. Esse diagrama também identifica as classes acessadas pelos papéis de objeto e papéis do agente. As interações entre os agentes e as

organizações do sistema são descritas com base nos relacionamentos entre os papéis ilustrados nos diagramas de papel (SILVA; CHOREN; LUCENA, 2007). Neste diagrama são exibidos os relacionamentos Controle, Dependência, Associação, Agregação e Especialização.

2.3.2 Aspectos Dinâmicos de MAS-ML

Os aspectos dinâmicos de MAS-ML são representados através de uma extensão dos diagramas de seqüência e de atividade da UML para representar os aspectos dinâmicos de SMAs, ou seja, para representar as interações entre as instâncias do SMA e as ações de cada instância.

A extensão do diagrama de sequência (SILVA, 2004) inclui a definição de novos *pathnames*[1] e ícones para as instâncias dos SMAs (agentes, organizações e ambiente). O conceito de mensagem usado em UML foi estendido para representar entidades que estão enviando e recebendo mensagens e não estão chamando métodos de outras entidades. Além disso, foram criados estereótipos para representar a criação e a destruição de instâncias de SMAs e para representar a interação entre agentes, organizações, objetos e seus papéis, alguns estereótipos associados a mensagens foram redefinidos e outros foram criados por SILVA (2004). A partir das adequações em MAS-ML, no diagrama de atividades é possível a modelagem de planos e ações dos agentes e das organizações, e dos conceitos relacionados com a linguagem (SILVA; CHOREN; LUCENA, 2005). Com isso, cada atividade é representada por um retângulo de bordas arredondadas. As crenças do agente são representadas por um quadrado com a identificação das crenças utilizadas pelo agente e os objetivos são representados no canto superior direito através de uma descrição textual com o estereótipo <<*goal*>>.

2.4 NormML

NormML (FIGUEIREDO, 2011) é uma linguagem de modelagem baseada na UML que permite a restrição do comportamento das entidades que compõem um SMA. O metamodelo de NormML é resultado da extensão do metamodelo SecureUML (BASIN; DOSER; LODDERSTEDT, 2006), que fornece uma linguagem para a modelagem de funções, permissões, ações, recursos e restrições de autorização, juntamente com as relações entre permissões e papéis, ações e permissões, recursos e ações e restrições e permissões.

[1] Na UML, estrutura de nomes para identificar uma entidade em um diagrama de seqüência.

A extensão para NormML é feita nos elementos básicos: *norm*, *agent* e *AgentAction* e também inclui um conjunto de invariantes que garante a boa formação de uma norma e várias operações que são usadas para identificar possíveis conflitos entre duas normas definidas. Com isso, NormML permite a modelagem das normas que são utilizadas para restringir o comportamento de agentes, organizações e suborganizações durante um período de tempo, e definir as sanções aplicadas quando violadas ou cumpridas (SILVA; BRAGA; FIGUEIREDO, 2010). A seguir, os principais elementos que compõem uma norma são descritos (FIGUEIREDO, 2011):

- **Conceitos deônticos:** a lógica deôntica refere-se à lógica de pedidos, ordens, regras, leis, princípios morais e julgamentos (MEYER; WIERINGA, 1993). Em SMAs, tais conceitos têm sido utilizados para descrever as restrições de comportamento para os agentes na forma de obrigações (o que o agente deve executar), permissões (o que o agente pode executar) e proibições (o que o agente não pode executar). Assim, uma das principais propriedades de uma norma é a identificação do seu tipo de restrição, ou seja, a identificação do conceito deôntico associado à norma.
- **Entidades Envolvidas:** desde que as normas são sempre definidas para restringir o comportamento das entidades, a identificação dessas entidades cujo comportamento está sendo restringido é fundamental. A norma pode regular o comportamento dos indivíduos (ou seja, de um determinado agente, ou de um agente enquanto está desempenhando um papel determinado) ou o comportamento de um grupo de indivíduos (ou seja, todos os agentes desempenhando um determinado papel, grupos de agentes, grupos de agentes desempenhando papéis ou todos os agentes do sistema).
- **Ações:** uma vez que uma norma é definida para restringir a execução das entidades é importante que a ação a ser envolvida seja claramente regulamentada. Tais ações podem ser de comunicação, normalmente representadas pelo envio e recebimento de uma mensagem, ou ações não-comunicativas (como acessar e modificar um recurso, entrar em uma organização, se deslocar para outro ambiente, etc.).
- **Restrições de Ativação:** As normas têm um período em que suas restrições devem ser cumpridas, porém somente quando elas, as normas, estiverem ativas. As normas podem ser ativadas por uma restrição ou um conjunto de restrições que

podem ser: a execução das ações, a especificação de intervalos de tempo (antes, depois, entre elas), a realização de estados do sistema ou aspectos temporais (como datas), e também a ativação / desativação de outra norma e o cumprimento / violação de uma norma.

- **Sanções:** quando uma norma é violada, a entidade que violou essa norma pode sofrer uma punição e quando for cumprida uma norma, a entidade que tem acompanhado a norma poderá receber uma recompensa. Essas recompensas e punições são chamadas de sanções e devem ser descritas juntamente com a especificação norma.
- **Contexto:** as normas são geralmente definidas em um determinado contexto que determina a área da sua aplicação. A norma pode, por exemplo, ser descrita no contexto de um determinado ambiente e deve ser preenchida apenas pelos agentes de execução no ambiente ou pode ser definida no contexto de uma organização e cumprida apenas pelos agentes que desempenham um papel na organização.

A Figura 4 apresenta parte do metamodelo[2] de NormML, no qual uma norma corresponde a um conjunto de metaclasses instanciadas juntamente com seus relacionamentos. A norma pode ser tanto uma permissão (por instanciar a metaclasse *NormPermission*), uma proibição (por instanciar a metaclasse *NormProhibition*) ou uma obrigação (por instanciar a metaclasse *NormObligation*).

A norma pode restringir o comportamento dos agentes em si, o comportamento de todos os agentes desempenhando um determinado papel, ou o comportamento de um agente específico que está desempenhando um determinado papel, definido pela relação entre agente e papel.

NormML considera como recurso qualquer propriedade das entidades que fazem parte do SMA. Esta linguagem tem quatro tipos de recursos: Atributo, Método, Entidade e Associação. Ela estende o conjunto de recursos com as ações de agentes e papéis que são representadas pela metaclasse *AgentAction*. Assim, é possível descrever as normas para controlar o acesso a atributos, métodos, objetos e associações, e também para controlar a execução das ações de agentes e papéis. Cada tipo de recurso é associado a um conjunto de ações que pode ser usado para controlar o acesso ao recurso. Por exemplo, os atributos são associados com as ações de leitura, atualização e acesso completo (leitura e atualização). No

[2] O metamodelo completo de NormML está disponível em http://www.ic.uff.br/~kfigueiredo/normML/abstractmetamodel.pdf

caso de restrições aplicadas às ações de agentes e papéis (metaclasse *AgentAction*), o comportamento que deve ser usado é a execução da ação (*ActionExecute*). Note que *AgentAction* é o recurso e *ActionExecute* é a ação que está sendo usado para controlar ou restringir o acesso ao recurso.

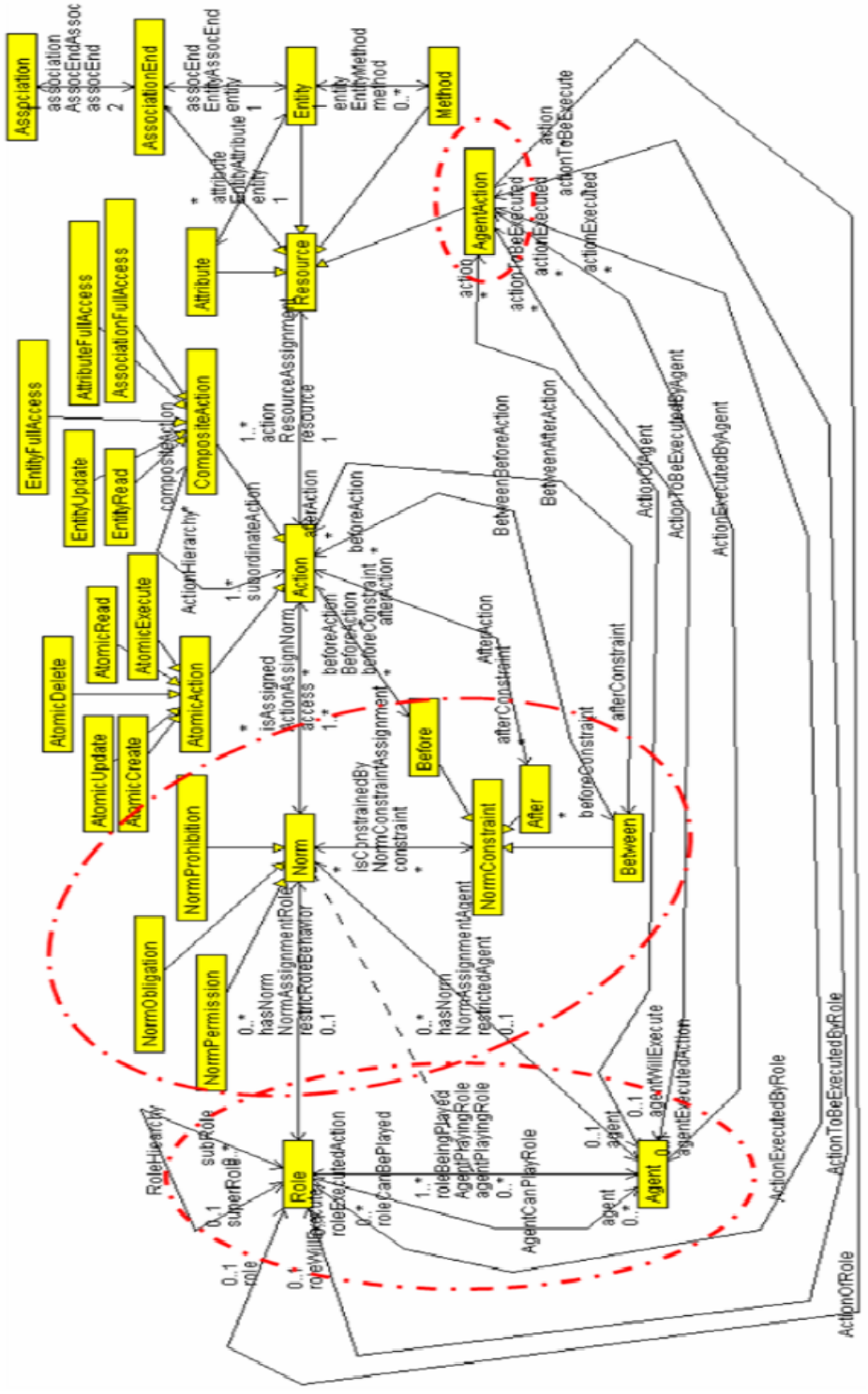


Figura 4 – Metamodelo Parcial de NormML (FIGUEIREDO, 2011)

Adicionalmente, NormML permite a especificação do período de tempo durante o qual uma norma é ativa, que é representada pela metaclasse *NormConstrain*. Com isso, se uma norma é condicionada por uma cláusula *Before*, isso significa que a norma é ativa somente antes da execução da(s) ação(ões) descrita(s) na cláusula *Before*. Analogamente para a cláusula *After*. No caso de uma cláusula *Between*, a norma só é ativa durante o período delimitado por dois grupos de ações.

A linguagem também possui uma ferramenta de suporte que permite a modelagem dos elementos descritos anteriormente e a verificação de conflitos entre normas.

2.5 Conclusão

Este capítulo abordou a teoria relacionada com os elementos que compõem os sistemas multi-agente normativos juntamente com o *framework* TAO e as linguagens de modelagem MAS-ML e NormML. Com isso, a fundamentação teórica relacionada com o escopo deste trabalho foi apresentada.

3 TRABALHOS RELACIONADOS

Diversas abordagens utilizando normas para agentes e SMA são propostas em (DENNIS; TINNEMEIER; MEYER, 2009), (VECHT et al., 2009), (VRIES et al., 2009) (GARCÍA-CAMINO et al., 2006) e (VASCONCELOS; KOLLINGBAUM; NORMAN, 2007). Esses autores focam na (i) utilização de normas para restringir o comportamento de entidades, (ii) modelagem dos elementos que compõem uma norma, (iii) verificação de conflito entre normas e (iv) implementação de normas. Entretanto, este capítulo apresenta a análise dos trabalhos relacionados que definem metodologias, modelos organizacionais e linguagens de modelagens para SMAs juntamente com normas.

Essa análise tem como objetivo verificar se cada trabalho analisado suporta a modelagem das entidades típicas dos SMAs assim como também, dos principais elementos que compõem as normas (apresentados na Seção 2.4). Adicionalmente, foi levada em consideração a existência de ferramenta CASE para suporte à modelagem das entidades e diagramas definidos para cada trabalho analisado.

A partir do levantamento bibliográfico realizado, foram analisados as metodologias Prometheus e SODA, os modelos de organização Moise+ e OperA, e as linguagens de modelagem AORML, AML, NormML e MAS-ML. Foram escolhidos esses trabalhos pelo fato deles possibilitarem a modelagem das entidades que compõem os sistemas multi-agente normativos e também pela afinidade com a proposta aqui apresentada.

3.1 Prometheus

A metodologia Prometheus (PADGHAM; WINIKOFF, 2004) define um agente com base em seus objetivos, crenças, planos e eventos. A metodologia consiste em três fases:

- Especificação do sistema, que utiliza cenários de caso de uso para descrever as seqüências das etapas de operações;
- *Design* da arquitetura, que atribui às funcionalidades definidas aos agentes na fase anterior. Essa fase utiliza os diagramas de (i) interação, que apresenta a interação entre agentes em vez de objetos sistema; (ii) conhecimento, que vincula cada agente a outros com os quais interage; e (iii) sistema, que une objetos de dados compartilhados, agentes e eventos; e
- *Design* detalhado, que define os diagramas de (i) agente, que é muito semelhante ao diagrama de sistema, mas oferece uma visão de alto nível das propriedades do

agente concentrando-se nas habilidades dentro de um agente em vez de um sistema; e (ii) habilidade, que descreve as propriedades do agente, apresentando o refinamento desde as habilidades definidas no diagrama de agente até os planos.

A metodologia estende o metamodelo da UML para a inclusão das suas fases e permite a definição de normas para as entidades do sistema como também a ativação das mesmas. Possui uma ferramenta CASE para dar suporte à metodologia por meio da modelagem dos diagramas definidos, juntamente com o código esqueleto correspondente na linguagem Jack (HOWDEN et al., 2001). Entretanto, Prometheus:

- Não define papéis, organizações e ambientes;
- Não descreve como o metamodelo de UML foi estendido a fim de incorporar conceitos relacionados a agentes;
- Permite a definição de normas de obrigação e permissão, e considera que o que não é permitido como proibido;
- Não permite a definição de sanções.

3.2 SODA

SODA (OMICINI, 2001) é uma metodologia para a análise e projeto de aplicações baseadas na Internet como SMAs. Tem como objetivo possibilitar a análise e o projeto de agentes individuais, sociedades de agentes e ambientes de agentes. A definição de agentes é feita em termos do seu comportamento observável e do seu papel no SMA. Portanto, a metodologia não aborda as questões intra-agentes, ou seja, não permite a definição da estrutura interna dos agentes.

A metodologia SODA consiste na fase de análise e de *design* para a descrição dos aspectos inter-agentes: definição das sociedades e da infra-estrutura para SMAs. Na fase de análise, o domínio da aplicação é estudado e modelado, levando em consideração, a avaliação dos recursos computacionais e as restrições da tecnologia a ser utilizada. Nesta fase são explorados os três modelos:

- Modelo de Papel: os objetivos da aplicação são modelados como tarefas que devem ser cumpridas, que são associadas a papéis e grupos de agentes;
- Modelo de Recurso: as características do ambiente da aplicação são modeladas como serviços disponíveis, que são associados a recursos abstratos; e
- Modelo de interação: a interação envolvendo papéis, grupos de agentes e recursos é modelada como (i) protocolos de interação, definidos como uma informação

requerida para papéis e recursos, e (ii) regras de interação, governando a interação entre grupos de agentes.

Como resultado da fase de análise, temos as seguintes entidades modeladas: (i) papel, que é definido de acordo com sua tarefa individual, suas permissões de acesso a recursos e seu protocolo de interação, (ii) grupo de agentes, que é definido de acordo com suas tarefas sociais, suas permissões de acesso a recursos, sua participação nas regras sociais e suas regras de interação, e (iii) interação, que é definido de acordo com os serviços que presta, seus modos de acesso, as permissões concedidas aos papéis e grupos para acessar seus serviços e seus protocolos de interação.

A fase de *design* consiste na representação dos modelos abstratos definidos na fase de análise. Nesta fase são explorados os três modelos:

- Modelo de agente: indivíduos e regras sociais são modelados como classes de agente;
- Modelo de sociedade: grupos são modelados como sociedades de agentes; e
- Modelo de ambiente: recursos são modelados como classes de infra-estrutura.

Como resultado da fase de *design*, temos as seguintes entidades definidas: (i) classe de agente, definida de acordo com seus papéis individuais e sociais, (ii) sociedade de agentes, definida de acordo com seus grupos e (iii) classe de infra-estrutura, definida de acordo com seus recursos.

A metodologia permite a definição de normas de permissão e obrigação para agentes, papéis e grupos de agentes no contexto de uma determinada organização. Entretanto, SODA:

- Não permite a modelagem dos aspectos estruturais de agentes;
- Não permite a modelagem de objetos, papéis de objetos e agentes associados a um determinado papel;
- Permite a definição de normas de obrigação e permissão, e considera que tudo o não é permitido como proibido;
- Não permite a definição de normas no contexto de ambiente;
- Não permite a definição de restrições de ativação para normas;
- Não permite a definição de sanções;
- Não possui ferramenta de suporte.

3.3 Moise+

O modelo organizacional Moise+ (HÜBNER; SICHMAN; OLIVIER, 2002) é fortemente baseado no modelo Moise (HANNOUN, 2002), que apresenta uma visão centrada na organização considerando três formas de representar as restrições organizacionais (papéis, planos e normas). O modelo Moise+, como é focado em uma visão centrada na organização, é baseado em duas noções centrais: (i) uma especificação organizacional, que um grupo de agentes adota para formar (ii) uma entidade organizacional, cuja ação se destina a atingir uma finalidade.

Para a organização foram definidos três aspectos: (i) estrutural, que possui os papéis da organização, que são os componentes elementares da organização, e como eles estão relacionados (relacionamento entre papéis, grupos de papéis e hierarquia); (ii) funcional, especifica como os planos globais, missões e metas podem ser atingidos; e (iii) deôntico, que liga os outros dois aspectos, indicando quais as responsabilidades dos papéis nos planos globais.

Moise+ permite apenas a descrição de normas de permissão e proibição para papéis no contexto de uma organização. Entretanto, somente ações não-comunicativas podem ser restringidas pelas normas. Adicionalmente, a ferramenta CASE *Moise API and Platform* (HÜBNER; SICHMAN; OLIVIER, 2002) permite criar a especificação da organização e gerenciar suas entidades. O modelo apresenta os seguintes pontos fracos:

- Não permite a especificação de ambientes. Logo, não é possível a modelagem de agentes se movendo de um ambiente para outro;
- Não define as propriedades dos agentes;
- Não possibilita a especificação de normas para agentes e ambientes;
- Não oferece suporte para a definição de sanções.

3.4 OperA

O modelo organizacional OperA (DIGNUM, 2004) é um *framework* que permite a especificação de SMAs por meio da distinção entre as características (estrutura e comportamento) do modelo de organização e do comportamento dos agentes que fazem parte desse modelo. Esse *framework* possui os modelos organizacional, social e de interação para a modelagem de organizações e seus componentes.

O modelo organizacional descreve a estrutura organizacional da sociedade, juntamente com os papéis e as interações definidos pelas partes interessadas da organização. Esse modelo possui quatro estruturas:

- Estrutura Social: descreve os papéis participantes da organização, incluindo seus objetivos, direitos e obrigações, os grupos possíveis de papéis e as relações entre os papéis;
- Estrutura de Interação: permite a descrição da interação entre as cenas que compõem a organização;
- Estrutura Normativa: permite a definição de normas que regem a organização;
- Estrutura Comunicativa: descreve a linguagem de domínio específica para a sociedade, para ser usada na comunicação entre os agentes e papéis.

O modelo social define quais os papéis, descritos no modelo organizacional, que serão exercidos pelos agentes da sociedade, e o modelo de interação define as possíveis interações entre os agentes na sociedade.

OperA permite a descrição de normas de obrigação, permissão e proibição para agentes, papéis de agentes e para grupos de agentes no contexto de organização, interação e transição de cena. Adicionalmente, permite a definição de restrição para a ativação das normas. Entretanto, esse modelo organizacional:

- Não permite a modelagem dos aspectos estruturais de agentes.
- Não permite a especificação de ambientes. Logo, não é possível a modelagem de agentes se movendo de um ambiente para outro;
- Não permite a definição de normas para agentes associados a um determinado papel;
- Não oferece suporte para a definição de sanções de recompensa, apenas de punição;
- Não possui ferramenta de suporte.

3.5 AORML

AORML (WAGNER, 2003) é uma linguagem de modelagem para SMAs baseada no metamodelo AOR (WAGNER, 2003). As entidades definidas em AOR são: agente, evento, ação, reinvidicação, comprometimento e objeto comum. As organizações são representadas através do agente do tipo institucional. Todas as entidades são definidas como estereótipos com base na metaclasse *Class* definida pela UML (UML, 2011).

Os diagramas de AORML dividem-se em externos e internos. No modelo externo temos os seguintes diagramas: Diagrama de agentes, Diagrama de interação de quadros, Diagrama de interação de sequência e Diagrama de interação de padrões. Enquanto no modelo interno temos o Diagrama de Quadros da Reação e o Diagrama de Seqüência de Reação.

AORML permite a descrição de normas para um papel específico e para sua aplicação em grupo de indivíduos, a definição do período de ativação das normas, a restrição de ações não-comunicativas (como por exemplo, acessar e modificar um recurso) (FIGUEIREDO; SILVA, 2010).

Alguns pontos fracos de AORML são:

- Não define ambientes. Portanto, não é possível modelar agentes que estejam se movendo de um ambiente para outro;
- Não é possível modelar agentes que estejam se movendo de uma organização para outra;
- Não permite a definição de normas para grupos de agentes;
- Não permite a modelagem de sanções;
- Não possui ferramenta de suporte para a modelagem dos conceitos definidos pela linguagem.

3.6 AML

A linguagem de modelagem para agentes AML (DANC, 2008) é uma linguagem de modelagem visual semi-formal para especificação, modelagem e documentação de sistemas que incorpora os conceitos da teoria de SMAs.

Esta linguagem de modelagem é baseada em um metamodelo que estende a UML sem introduzir explicitamente novos conceitos ao metamodelo da UML. As entidades definidas por AML são agentes, recursos e ambientes. Devido à necessidade de modelar aspectos sociais, AML utiliza conceitos de unidades organizacionais, relacionamento social, papéis e propriedades de papéis.

Devido à necessidade de modelar aspectos sociais, esta linguagem utiliza conceitos de unidades organizacionais, relacionamento social, papéis e propriedades de papéis. A linguagem possibilita descrever normas para um papel específico e para um grupo de indivíduos. Proporciona também, uma forma de restringir tanto ações comunicativas como não-comunicativas e a definição de um período no qual as normas ficarão ativas (SILVA; BRAGA; FIGUEIREDO, 2010). Algumas limitações de AML são descritas a seguir:

- Nem mesmo utilizando os mecanismos de extensão fornecidos pela UML ou simplesmente mudando alguns diagramas, a linguagem consegue dar apoio suficiente à complexidade de dinâmica de SMAs (GONÇALVES, 2009);
- Não introduz explicitamente novos conceitos no metamodelo UML;
- Não oferece suporte para a definição de sanções;
- Não permite a definição de normas para um ambiente;
- Não prevê a ativação de restrições pelo cumprimento ou violação de normas;
- Não possui ferramenta de suporte capaz de modelar os conceitos da linguagem.

3.7 NormML

NormML é uma linguagem de modelagem baseada na UML para a especificação das normas de forma a restringir o comportamento das entidades dos SMA. A escolha da UML como metalinguagem permite uma fácil integração de NormML com outras linguagens de modelagem de SMA também baseada na UML, como AUML (ODELL, 2000) e MAS-ML (SILVA; CHOREN; LUCENA, 2008).

O metamodelo de NormML estende o metamodelo SecureUML (BASIN; DOSER; LODDERSTEDT, 2006), que fornece uma linguagem para a modelagem de funções, permissões, ações, recursos e restrições de autorização, juntamente com as relações entre permissões e papéis, ações e permissões, recursos e ações e restrições e permissões. A extensão é feita nos elementos básicos: *norm*, *agent* e *agentAction* e também, inclui um conjunto de invariantes que garante a boa formação de uma norma e várias operações que são usadas para identificar possíveis conflitos entre duas normas definidas.

Com a linguagem NormML é possível a modelagem de todos os elementos que compõem as normas: (i) conceitos deônticos, (ii) entidades envolvidas, (iii) ações, (iv) restrições de ativação, (v) sanções e (vi) contexto. Assim, as normas podem ser definidas para agentes, organizações, suborganizações, papéis de agente e ambiente, juntamente com seus períodos de ativação e sanções associadas. NormML possui uma ferramenta de suporte para a modelagem de normas nas entidades definidas na linguagem e para a checagem de conflitos entre as normas. Como a linguagem foca somente na modelagem dos elementos que compõem as normas, algumas deficiências podem ser levantadas:

- Não é possível modelar agentes que estejam se movendo de uma organização para outra;
- Não define a troca de papéis por um agente;

- Não permite a modelagem das entidades que fazem parte de um SMA, como por exemplo, do agente e suas propriedades.

3.8 MAS-ML

A linguagem MAS-ML é resultado de uma extensão conservativa da UML juntamente com a inclusão de todas as entidades e suas propriedades definidas no *framework* TAO (SILVA et al., 2003). A extensão realizada baseou-se na definição de novas metaclasses e estereótipos.

MAS-ML possibilita a modelagem de todas as entidades, suas propriedades e relacionamentos tipicamente encontrados em SMAs. As entidades definidas originalmente incluem: (i) agente, (ii) papel de agente, (iii) papel de objeto, (iv) objeto, (v) organização e (vi) ambiente. A mudança de papel e de organização por um determinado agente é prevista na linguagem, como também, a interação entre todas as entidades definidas.

A linguagem: (i) possibilita a modelagem dos conceitos deônticos de permissão e obrigação representados pelos estereótipos <<*duty*>> e <<*right*>>, os quais são definidos no papel de agente. Com isso, a restrição das ações de um agente, de uma organização e/ou de uma suborganização é aplicada de acordo com o papel ativo ao qual a referida entidade se encontra associada; (ii) permite a definição de normas de obrigação no contexto de organizações e suborganizações por meio do estereótipo <<*axiom*>>. Essas normas são restrições globais dentro de uma determinada organização ou suborganização. Com isso, todos os agentes e suborganizações vinculados a uma organização ou suborganização, têm seu comportamento restringido pelas normas definidas por esse estereótipo; adicionalmente, a linguagem (iii) possui ferramenta de suporte para a modelagem dos conceitos definidos na linguagem.

Por outro lado, a linguagem apresenta algumas deficiências, tais como:

- Não permite a definição de normas aplicadas no contexto de um ambiente, que devem ser válidas para todas as entidades que habitam este ambiente;
- Não permite a definição de normas de proibição para papéis de agente;
- Não permite a definição de normas de proibição e permissão para agentes, organizações e suborganizações;
- Não prevê a modelagem de sanções nem de ativações das normas.

3.9 Comparativo entre os Trabalhos Relacionados

A Tabela 1 apresenta o comparativo entre os trabalhos relacionados apresentados nesta seção. Para a comparação, foram levados em consideração: (i) a capacidade de modelar as entidades típicas dos SMAs, (ii) a possibilidade de modelar os principais elementos que compõem as normas e (iii) a existência de ferramenta CASE para suporte a modelagem.

Tabela 1 – Comparativo entre os Trabalhos Relacionados

Trabalhos Relacionados	Principais Elementos que Compõem as Normas para SMAs																				Modelagem das Entidades Típicas de um SMA com suas Propriedades						CASE
	Conceitos Deônticos			Entidades Envolvidas					Ações		Restrições de Ativação					Sanções		Contexto										
	Permissão	Proibição	Obrigação	Agente	Papel	Agente vinculado com Papel	Grupos de indivíduos	Todos no sistema	Comunicativas	Não-Comunicativas	Execução de Ações	Intervalo de Tempo	Obtenção de estados	Aspectos Temporais	Cumprimento e Violação de Normas	Punição	Recompensa	Ambiente	Organização	Interação	Transição de Cena	Agente	Papel de Agente	Objeto	Papel de Objeto	Organização	Ambiente	
Prometheus	•		•		•		•	•	•	•	•	•	•	•					•	•		•		•				•
SODA	•		•	•	•		•	•		•									•			•	•			•	•	
Moise+	•		•		•					•				•					•			•	•			•		
OperA	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•			•	•	•	•	•			•		
AORML	•	•	•	•	•		•			•				•					•			•	•	•				
AML	•	•	•	•	•	•	•		•	•	•	•	•	•					•	•		•	•			•	•	
NormML	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•										•
MAS-ML	•		•		•	•	•		•	•									•			•	•	•	•	•	•	•

3.10 Conclusão

A partir da análise apresentada e considerando a necessidade de uma linguagem de modelagem que possibilite a modelagem de normas em associação às entidades típicas de um SMA, destacamos MAS-ML por (i) ser baseada na UML, (ii) possuir uma ontologia adequada para a modelagem das entidades típicas de SMA e seus relacionamentos, (iii) prever suporte à modelagem de alguns dos elementos que compõem normas para SMAs, porém de forma parcial, e (iv) possuir ferramenta de suporte. Com isso, o metamodelo de MAS-ML se apresenta como o mais adequado a receber as entidades e relacionamentos vinculados a normas. A extensão em MAS-ML é baseada nos conceitos definidos em NormML, uma vez que dada a compatibilidade entre os metamodelos, ambos derivados da UML, as metaclasses e relacionamentos em NormML podem ser reutilizados mais facilmente, com as devidas adequações.

4 ESTENDENDO O *FRAMEWORK* TAO E A LINGUAGEM MAS-ML

Este capítulo apresenta a extensão do *framework* TAO e da linguagem MAS-ML, para dar origem a NorMAS-ML. A extensão do TAO consiste na definição de uma nova entidade para representar a norma, suas propriedades e relacionamentos. A extensão de MAS-ML ocorre em três estágios: na sintaxe abstrata, na qual as metaclasses e relacionamentos definidos em NormML são adequados e incorporados, e na sintaxe concreta, na qual novos elementos gráficos a serem utilizados nos diagramas são definidos para representar as entidades criadas na sintaxe abstrata. Adicionalmente, um novo diagrama estrutural é definido para a modelagem das normas. Tanto a extensão de MAS-ML como a definição do diagrama estrutural foram apresentados por (FREIRE et al., 2012a). Portanto, esse capítulo apresenta uma versão revisada e estendida dessa extensão.

4.1 Novas Características no TAO

A extensão do *framework* TAO se baseia na inclusão dos elementos que compõem as normas apresentados na Seção 2.4. Estes elementos foram estabelecidos a partir do levantamento apresentado em (FIGUEIREDO, 2011) que analisou especificações e linguagens utilizadas para descrever e implementar normas.

Para possibilitar a inclusão desses elementos no *framework* TAO, foram definidos a entidade *Norm* e os quatro relacionamentos: (i) *Context*, (ii) *Restrict,* (iii) *SanctionReward* e (iv) *SanctionPunishment*. Foi necessária a definição dessa nova entidade, pois a mesma possui um estado, propriedades comportamentais e relacionamentos específicos. Adicionalmente, a entidade *Norm* possibilita que as entidades definidas em TAO possam ter seu comportamento restringido, juntamente com a aplicação de sanções e a definição dos períodos nos quais as normas estarão ativas.

Para a associação das normas com as demais entidades que fazem parte da sua definição, foram criados os relacionamentos (i) *Context*, responsável por identificar o contexto no qual a norma será aplicada; (ii) *Restrict*, responsável por identificar quais as entidades que serão restringidas pela norma; (iii) *SanctionReward*, responsável por identificar quais as recompensas de uma determinada norma e (iv) *SanctionPunishment*, responsável por identificar quais as punições de uma determinada norma. A Figura 5 apresenta a extensão do metamodelo do *framework* TAO juntamente com o novo elemento *Norm* e seus relacionamentos.

A seguir, tanto a entidade *Norm* como os quatro relacionamentos *Context*, *Restrict, SanctionReward* e *SanctionPunishment* serão definidos por meio da utilização de *templates* (SILVA et al., 2003).

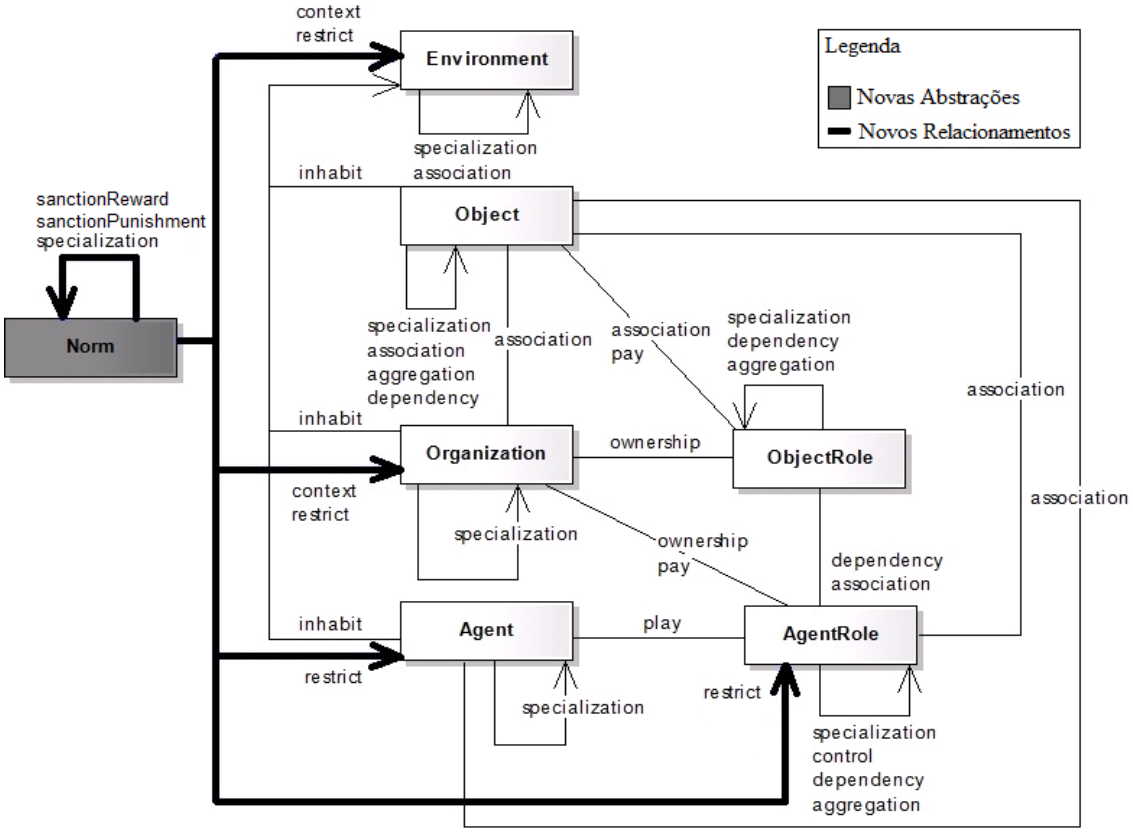


Figura 5 – Metamodelo estendido do TAO

4.1.1 Entidade *Norm*

Segundo FIGUEIREDO e SILVA (2010), a norma é um elemento que restringe durante um período de tempo o comportamento de agentes, organizações e suborganizações, e aplica sanções quando violadas ou cumpridas.

A norma é um elemento que possui um estado, propriedades comportamentais e relacionamentos. O estado da norma armazena o recurso a ser restringido. Segundo SILVA, BRAGA e FIGUEIREDO (2010), o recurso que pode ser restringido pela norma pode ser uma entidade ou uma propriedade de uma entidade. As entidades que podem ser governadas pelas normas são: (i) um agente, (ii) um papel de agente, (iii) uma organização e (iv) um ambiente. Adicionalmente, as propriedades que podem ser restringidas são: (i) um objetivo, (ii) uma crença, (iii) um atributo, (iv) um método, (v) uma ação, (vi) um plano, (vii) um protocolo, (viii) uma associação ou (ix) uma mensagem.

O comportamento da norma é definido nas suas características, as quais são baseadas nos conceitos deônticos e nas restrições de ativação. Os conceitos deônticos definem

o tipo de restrição da norma. Portanto, a norma pode ser uma obrigação (o que o agente *deve* executar), uma permissão (o que o agente *pode* executar) ou uma proibição (o que o agente *não pode* executar). As restrições de ativação definem o período no qual a norma estará ativa. Uma norma pode ser ativada por uma restrição ou por um conjunto de restrição que podem ser: (i) a execução de ações, (ii) específicos intervalos de tempo (antes, depois ou entre), (iii) a realização de estados do sistema ou aspectos temporais (como datas) e também (iv) a ativação ou desativação de outras normas ou o cumprimento ou violação de uma norma.

Os relacionamentos da norma descrevem (i) o contexto que determina a aplicação da norma, (ii) a entidade que terá o seu comportamento restringido e (iii) a recompensa ou a punição que será recebida pela entidade que cumpriu ou violou a norma. Esses relacionamentos serão descritos nas próximas seções.

O *template* da norma apresenta a classe da norma que define seu estado como o recurso que será restringido, o comportamento de suas instâncias como um conjunto de suas propriedades, que são baseadas nos conceitos deônticos e nas restrições de ativação, e outro conjunto formado pelos relacionamentos que são comuns a todas as instâncias da norma.

_____Norm_____

Norm_Class Norm_Class_Name

Restriction_Type Deontic_Concept_Name
Resource <Element_Class_Name.property >
Activation_Constraint setOf{Constraint_Type Constraint_Type_Name : (<Element_Class_Name_First> and/or <Element_Class_Name_Second>) or <date> or <Element_Class_Name.property : Operator = (Element_Class_Name.property) or value)>}
Relationships setOf {Relationship_Name}

end Norm_Class

4.1.2 Relacionamentos da Norma

Esta seção apresenta os quatro relacionamentos definidos no TAO para possibilitar a associação entre as entidades do TAO e o novo elemento *Norm* definido na seção anterior. Estes novos relacionamentos foram introduzidos para representar as outras propriedades das normas.

Seja **A** um conjunto de agentes, **a** ∈ **A**, **E** seja um conjunto de ambientes, **e** ∈ **E**, **N** seja um conjunto de normas, **n** ∈ **N** e **O** seja um conjunto de objetos, **o** ∈ **O**. Seja **Org** um conjunto de organizações, **org**, **subOrg** ∈ **Org** e **subOrg** sempre representa uma suborganização. Seja **R** um conjunto de papéis, **R** = **RObj** U **RAg**, onde **RObj** é um conjunto de papéis de objeto e **RAg** é um conjunto de papéis de agente, **r** ∈ **R**, **ro** ∈ **Robj** e **ra** ∈ **RAg**. Para cada um dos relacionamentos apresentados a seguir, será representada a sua definição, sua classificação e os elementos que podem ser ligados por ele.

- **Context (C): C(context, norm): C(e, n), C(org, n), C(subOrg, n)**

O relacionamento *context* especifica que uma entidade está relacionada a uma norma. Quando uma classe de entidade está relacionada a uma classe da norma pelo relacionamento *context*, isso significa que a instância de entidade é o contexto que determina a área de aplicação da norma. Uma instância da norma deve estar vinculada a pelo menos uma instância de (i) organização, (ii) suborganização e/ou (iii) ambiente.

Quando uma norma está vinculada com uma organização, suborganização e/ou ambiente, todos os seus habitantes (organizações, suborganizações e agentes) terão seu comportamento restringido por essa norma.

- **Restrict (R): R(element, norm): R(a, n), R(e, n), R(org, n), R(subOrg, n), R(ra, n)**

O relacionamento *restrict* define qual a entidade que será restringida pela norma. Quando uma classe de entidade está relacionada a uma classe da norma por esse relacionamento, isso significa que a instância da entidade estará sendo restringida pela norma e caso viole ou siga a norma, ela receberá a sanção associada a esta norma.

As entidades que podem ser restringidas pela norma são: (i) agente, (ii) ambiente, (iii) organização, (iv) suborganização e (v) papel de agente.

- **SanctionReward (SR): SR(reward, norm): SR(n, n)**

O relacionamento *sanctionReward* especifica a sanção de recompensa que pode ser recebida pela entidade que cumpriu a norma. Esse relacionamento só pode ser vinculado a classes da norma. Com isso, o relacionamento identifica a norma que contém a recompensa e o recurso de outra norma que será a recompensa.

- **SanctionPunishment (SP): SP(punishment, norm): SP(n, n)**

O relacionamento *sanctionPunishment* especifica a sanção de punição que pode ser recebida pela entidade que violou a norma. Esse relacionamento só pode ser vinculado a classes da norma. Com isso, o relacionamento identifica a norma que contém a punição e o recurso de outra norma que será a punição.

O *template* que representa os novos relacionamentos definidos para TAO é apresentado abaixo. Para cada relacionamento, são identificados seus elementos e os seus respectivos papéis.

_____Relationship_____

Relationship Relationship_Name

CONTEXT : context, norm
| RESTRICT : element, norm
| SANCTION_REWARD : reward, norm
| SANCTION_PUNISHMENT : punishment, norm

end Relationship

4.2 Adequações nas Entidades do TAO

Além de definir o novo elemento *Norm* e seus relacionamentos, adaptações nos elementos já existentes são necessárias visando à consistência do metamodelo e evitando redundâncias. De acordo com SILVA et al. (2003), um papel de agente guia e restringe o comportamento de um agente, uma vez que os objetivos, as crenças, os deveres, os direitos, os protocolos e os compromissos associados ao papel caracterizam o agente enquanto está associado a esse papel. Na concepção original, os conceitos de direito (*right*) e dever (*duty*) são usados para definir as ações que *podem* e *devem* ser executados por agentes. Visto que esses conceitos são utilizados para restringir o comportamento dos agentes que estão associados a esse papel, eles podem ser considerados semanticamente equivalentes aos conceitos deônticos de permissão e obrigação definidos pelas normas. Consequentemente, os conceitos de direito e dever foram substituídos pelos conceitos de norma no *template* de papel de agente. Com isso, além de definir as ações que podem e devem ser executadas, o papel de agente pode definir também as ações que não podem ser executadas ou proibidas. Consequentemente, no *template* do papel de agente também foi incluída a lista de ações que podem ser restringidas pelas normas. O novo *template* para a representação do papel do agente é apresentado a seguir.

_____Agent_Role_____

Agent_Role_Class Agent_Role_Class_Name

Goals setOf{Goal_Name}
Beliefs setOf{Belief_Name}
Actions setOf{Action_Name}
Norms setOf{Norm_Name}
Protocols setOf{Interaction_Class_Name}U setOf{Rule_Name}
Commitments setOf{Action_Name}
Relationships setOf{Relationship_Name}

end Agent_Role_Class

Uma organização define um conjunto de regras (*rules*) e leis (*laws*) que os agentes e as suborganizações associados a ela devem obedecer. As regras e as leis são usadas para caracterizar as restrições globais de uma organização (SILVA et al., 2003). De acordo com MEYER e WIERINGA (1993), o conceito de normas inclui as características das regras e das leis. Portanto, na extensão, estes conceitos foram removidos e substituídos pela associação de normas à organização. Similarmente ao caso do papel de agente, o conjunto de ações que poderão ser restringidas pelas normas é definido no *template* da organização. Com isso, todos os agentes que estiverem associados com uma organização terão que seguir essas normas, caso não queiram receber sanções de penalidade. O novo *template* da organização é apresentado a seguir.

_____Organization_____
Organization_Class Organization_Class_Name

Norms setOf{Norm_Name}
Actions setOf{Action_Name}
Relationships setOf{Relationship_Name}

end Organization_Class

No TAO, um ambiente incorpora as restrições de acesso associadas aos seus serviços e recursos (SILVA et al., 2003). Por outro lado, o conceito de norma contempla a definição de restrições de acesso relacionadas com serviços e recursos no ambiente. Portanto, os conceitos de serviços (*services*) e recursos (*resources*) foram removidos e substituídos pelo conceito de norma no ambiente. O novo *template* para a representação ambiente é apresentado a seguir.

_____Environment_____
Environment_Class Environment_Class_Name

Norms setOf{Norm_Name}
Behavior setOf{Properties}
Relationships setOf{Relationship_Name}
Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}

end Environment_Class

4.3 Extensão da Sintaxe Abstrata de MAS-ML

Nesta seção é apresentada a extensão da sintaxe abstrata da linguagem MAS-ML de modo que as entidades que fazem parte de um SMA possam ser modeladas considerando os principais elementos que compõem as normas a elas associadas. Em (FREIRE et al., 2011) e (FREIRE et al., 2012b) é apresentada uma versão detalhada dessa extensão. Esta seção descreve uma versão revisada e estendida dessas publicações.

A extensão do metamodelo é resultado da análise realizada considerando os conceitos, as metaclasses e os relacionamentos definidos em NormML, e em consistência com as alterações realizadas no *framework* TAO apresentadas nas seções 4.1 e 4.2 do presente trabalho. A estratégia utilizada para a extensão é baseada nos mecanismos de extensão conservativa da UML, isto é, por meio da definição de novas metaclasses e estereótipos, e conservação da estruturação das metaclasses originárias da UML, sem perda de semântica.

4.3.1 Norma

As normas são baseadas nos conceitos deônticos que têm sido utilizados para descrever as restrições de comportamento para os agentes. São eles: obrigações, permissões e proibições (FIGUEIREDO, 2011).

Considerando a complexidade dos elementos envolvidos na definição de normas e dos relacionamentos requeridos entre normas e as outras entidades envolvidas na sua definição (metaclasses), foi necessária a adequação da metaclasse *Norm,* originária de NormML. Esta metaclasse foi incluída no metamodelo de MAS-ML e estende a metaclasse *Element*, originaria da UML, para representar as normas. Com esta metaclasse podemos definir normas para: (i) agentes, (ii) organizações, (iii) suborganizações, (iv) ambiente e (v) papel de agente.

Na sua concepção original, MAS-ML contempla a definição de obrigações e direitos (ou permissões) do agente enquanto executa um papel. Estes conceitos em MAS-ML são denotados pelos estereótipos <<*duty*>> e <<*right*>> associados à metaclasse *AgentAction*. Porém, de acordo com a análise apresentada na Seção 4.2, estes conceitos foram eliminados do TAO e substituídos por normas. Essa alteração reflete na sintaxe abstrata de MAS-ML uma vez que os estereótipos <<*duty*>> e <<*right*>> definidos originalmente na metaclasse *AgentAction* se tornam desnecessários e foram eliminados. A partir da inclusão do conceito de norma no TAO e considerando que além de direitos e permissões, uma norma pode identificar também proibições, três novos estereótipos foram criados na sintaxe abstrata da linguagem em associação à metaclasse *Norm*, são eles: <<*obligation*>>, <<*permission*>> e <<*prohibition*>>,

que representam as obrigações (o que o agente *deve* executar), as permissões (o que o agente *pode* executar) e as proibições (o que o agente *não pode* executar), respectivamente. A Figura 6 apresenta a metaclasse *Norm* e seus estereótipos.

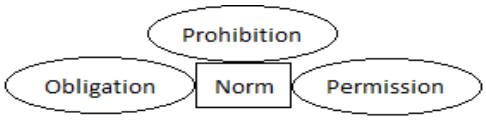


Figura 6 – Estereótipos da metaclasse *Norm*

Diferentemente do metamodelo de NormML, onde estes conceitos são representados por metaclasses, no presente trabalho foram modelados como estereótipos. Essa decisão de projeto foi tomada, pois esses conceitos possuem sua estrutura semelhante à estrutura da metaclasse *Norm*. Assim sendo, a modelagem através de estereótipos é suficiente para diferenciar os três conceitos deônticos que estão associados às normas.

Neste novo cenário, normas são vinculadas ao papel de agente de forma a manter as restrições no comportamento dos agentes e suborganizações associados a esse papel. Com isso, além de definir as ações que podem e devem ser executadas pelos agentes e suborganizações, um papel de agente pode definir as ações que serão proibidas de serem executadas pelos agentes e suborganizações que estão associados a esse papel.

Outro conceito contemplado na definição original da linguagem MAS-ML que pode ser associado à teoria de normas é o de axioma. Os axiomas caracterizam as restrições globais da organização às quais os agentes e as suborganizações devem obedecer. Este conceito na sintaxe abstrata de MAS-ML é denotado pelo estereótipo <<*axiom*>> associado à metaclasse *Property*, e representa um conjunto de leis e regras que os agentes e suborganizações vinculadas a uma determinada organização devem obedecer (SILVA, 2004). Porém, a partir da inclusão do conceito de normas e de acordo com a análise apresentada na Seção 4.2, os conceitos de leis e regras foram eliminados do *framework* TAO. Desta forma, para garantir a consistência entre a sintaxe abstrata de MAS-ML e o *framework* TAO, o estereótipo <<*axiom*>> foi eliminado. No entanto, uma organização ou suborganização deve ser associada a uma norma ou a um conjunto de normas para que possa continuar a restringir o comportamento dos agentes ou das suborganizações que executam papéis nesta organização ou suborganização. Neste contexto, uma organização ou uma suborganização é capaz de definir as ações que podem, devem e/ou não podem ser executadas pelos agentes e suborganizações que estão associadas a elas.

Segundo FIGUEIREDO (2011), uma norma pode regular o comportamento de (i) um único agente, (ii) um conjunto de agentes que estão exercendo um determinado papel, (iii)

um agente específico que está exercendo um determinado papel, (iv) todos os agentes que estão exercendo papéis em uma organização, (v) todos os agentes que estão exercendo papéis em uma suborganização enquanto tal suborganização está exercendo um papel na sua super-organização e (vi) todos os agentes que habitam um determinado ambiente.

Com isso, podemos perceber que uma norma pode estar associada a uma organização, um agente, um papel de agente e a um ambiente. Por meio da associação entre a metaclasse *Norm* e *Classifier*, através da herança, consegue-se vincular uma norma a todas as especificações de *Classifier*. Porém nem todas as especificações de *Classifier* podem ser vinculadas a normas, como Objeto (*Class*) e Papel de Objeto (*ObjectRoleClass*). A Figura 7 apresenta os relacionamentos definidos entre a metaclasse *Norm* e as especializações de *Classifier*.

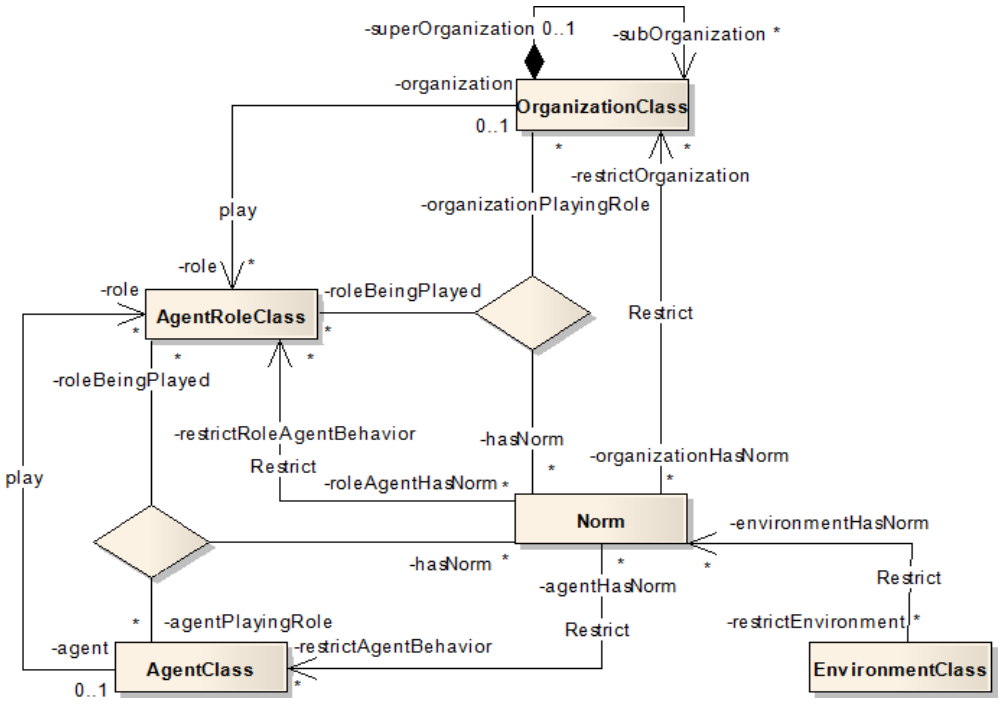


Figura 7 – Associações entre *Norm* e outras metaclasses (adaptado) (FIGUEIREDO, 2011)

4.3.2 Ação

Uma norma é definida para restringir a execução das entidades ou acesso aos recursos do sistema, através de um conjunto de ações. A partir dessa definição, as ações que controlam recursos são vinculadas a normas. Com isso, as ações a serem regulamentadas precisam ser claramente definidas. Para atender a essas especificações, em MAS-ML foi criada a metaclasse *NormAction,* equivalente a *Action* de NormML. Essa metaclasse tem como objetivo definir as ações sobre os recursos que serão restringidos pelas normas,

diferentemente da metaclasse *AgentAction*, definida em MAS-ML, que tem como objetivo representar as ações executadas por agentes.

A metaclasse *NormAction* foi definida em MAS-ML como extensão da metaclasse *BehavioralFeature* da UML, e é especializada através das metaclasses *AtomicAction* e *CompositeAction* (ambas originárias de NormML). A metaclasse *AtomicAction* tem como objetivo mapear diretamente as operações do sistema modelado (excluir, atualizar, ler, criar e executar), enquanto a metaclasse *CompositeAction* mapeia as operações das entidades do sistema e mantém a hierarquia tanto de *AtomicActions* como *CompositeActions* (BASIN; DOSER; LODDERSTEDT, 2006).

Segundo BASIN, DOSER e LODDERSTEDT (2006), as ações atômicas podem ser classificadas em ações de (i) atualização, (ii) envio, (iii) deleção, (iv) leitura, (v) recebimento, (vi) cancelamento, (vii) criação, (viii) execução, (ix) confirmação e (x) alcance; já a ação composta pode ser classificada em ação de (i) atualização, (ii) leitura, (iii) total acesso, (iv) envio, (v) recebimento, e (vi) execução em relação a recursos específicos do sistema. A incorporação destes conceitos no metamodelo de MAS-ML é realizado através da definição de novos estereótipos associados à metaclasse *AtomicAction* e *CompositeAction*, respectivamente, através dos quais pode ser diferenciado cada tipo de ação (Figura 8).

Figura 8 – Estereótipos das novas metaclasses *AtomicAction* e *CompositeAction* em MAS-ML

Em NormML, todos os tipos de ações definidos para as metaclasses *AtomicAction* e *CompositeAction* são definidos em seu metamodelo como metaclasses. Entretanto, utilizamos estereótipos para representá-los. Essa decisão de projeto foi tomada, pois todas essas metaclasses de NormML possuíam a mesma estrutura da classe *Action*. Portanto, a diferenciação de semântica pode ser feita por meio de estereótipos (UML, 2011). A Tabela 2 apresenta as ações que podem ser utilizadas para restringir o acesso a um determinado recurso. As ações sublinhadas representam ações compostas.

Tabela 2 – Recursos e suas ações (FIGUEIREDO, 2011)

Recurso	Ações
Entidade	create, read, update, delete, full access
Atributo	read, update, achieve, full access
Método	execute
Associação	read, update, full access
Agente	create, delete, update, full access
Papel de Agente	create, delete, commit, cancel, update, full access
Organização	create, delete, enter, leave, update, full access
Ambiente	create, delete, enter, leave, update, full access
Ação de Agente	execute
Mensagem	receive, send, full access
Protocolo	create, delete, receive, send, full access
Crença	create, delete, update, full access
Objetivo	achieve, commit, cancel, full access
Plano	create, delete, update, execute, full access

Em MAS-ML foram reaproveitadas todas as associações da ação definidas em NormML. A Figura 9 apresenta essas associações, juntamente com as especializações da metaclasse *NormAction*.

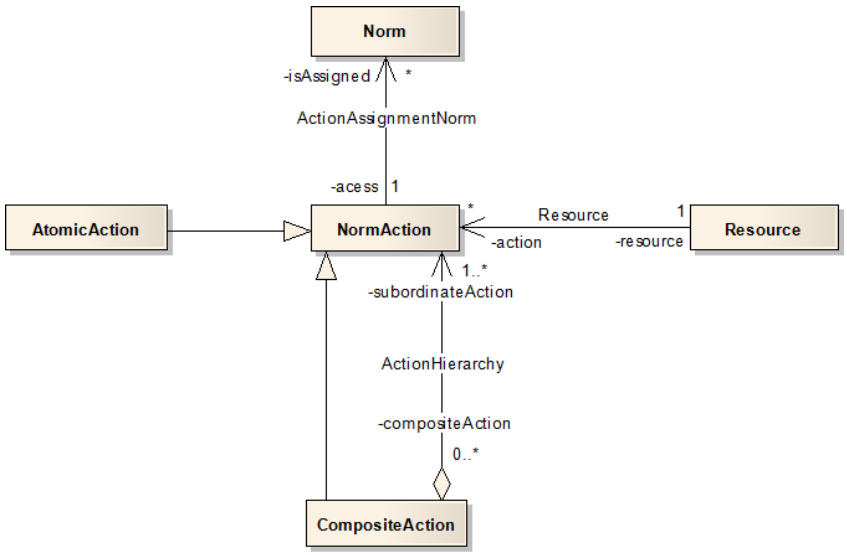


Figura 9 – Associações entre *NormAction* e outras metaclasses (adaptado) (FIGUEIREDO, 2011)

Por meio da associação *ActionAssignmentNorm* (Figura 9) uma ação pode ser vinculada a uma ou mais normas. Com o relacionamento *Resource*, um recurso se vincula a uma ação. Com isso, é possível modelar normas que definem diferentes restrições de acesso para diferentes recursos. Finalmente, por meio da associação *ActionHierarchy*, é possível criar uma hierarquia de ações.

4.3.3 Restrição de Ativação

Segundo FIGUEIREDO (2011), as normas têm um período em que suas restrições devem ser cumpridas, porém somente quando elas, as normas, estiverem ativas. As normas podem ser ativadas por uma restrição ou um conjunto de restrições que podem ser: (i) a execução das ações, (ii) a especificação de intervalos de tempo (antes, depois, entre elas), (iii) a realização de estados do sistema ou aspectos temporais (como datas), e também (iv) a ativação / desativação de outra norma e (v) cumprimento / violação de uma norma.

De forma a modelar estes conceitos em MAS-ML e devido à complexidade apresentada nos diversos tipos de restrição de ativação, foi adequada a metaclasse *NormConstraint*, responsável por definir o período de restrição de uma norma, juntamente com as suas especializações: (i) *Before*, indica que uma norma está ativa antes da execução da ação e/ou da realização a partir de uma determinada data; (ii) *After*, indica que uma norma está ativa depois da execução da ação e/ou da realização a partir de uma determinada data; (iii) *Between*, indica que uma norma está ativa entre a execução de duas determinadas ações e/ou da realização entre duas datas definidas; (iv) *If*, ativa a norma de acordo com a comparação entre dois operandos ou por data. Um operando pode ser um objetivo, uma crença, um atributo ou um valor.

Por meio das associações da metaclasse *NormAction*, conseguimos definir o período de restrição de ativação de uma norma. O relacionamento *BeforeAction* significa que a norma está ativa antes da execução da ação e/ou da realização da data descrita pelo relacionamento *BeforeDate*. O relacionamento *AfterAction* significa que a norma está ativa depois da execução da ação e/ou da realização da data descrita pelo relacionamento *AfterDate*.

No caso dos relacionamentos *BetweenBeforeAction* e *BetweenAfterAction*, a norma está ativa durante o período definido por esses dois relacionamentos ou por datas, definidos pelos relacionamentos *beforeBetweenDate* e *afterBetweenDate*. O relacionamento *CondicionalConstraint* ativa a norma de acordo com a comparação entre dois operandos ou por data. A metaclasse *Date* foi reutilizada para armazenar as datas de ativação das normas.

Utilizando o relacionamento *condicionalConstraint*, uma norma estará ativa quando (i) uma data definida na cláusula *if* for alcançada ou (ii) os operandos definidos no relacionamento *condicionalOperand* juntamente com o operador definido na associação *restrictOperand* forem verdadeiros. Por exemplo, uma norma só será ativa caso um objetivo a ser alcançado seja igual ao objetivo definido na cláusula *If*. Um operando pode ser um

objetivo, uma crença, um atributo ou um valor. A metaclasse *Operator* de NormML foi reutilizada para indicar a comparação a ser feita na cláusula *if*.

A Figura 10 apresenta os relacionamentos das especializações da metaclasse *NormConstraint*. Todos eles foram reutilizados de NormML.

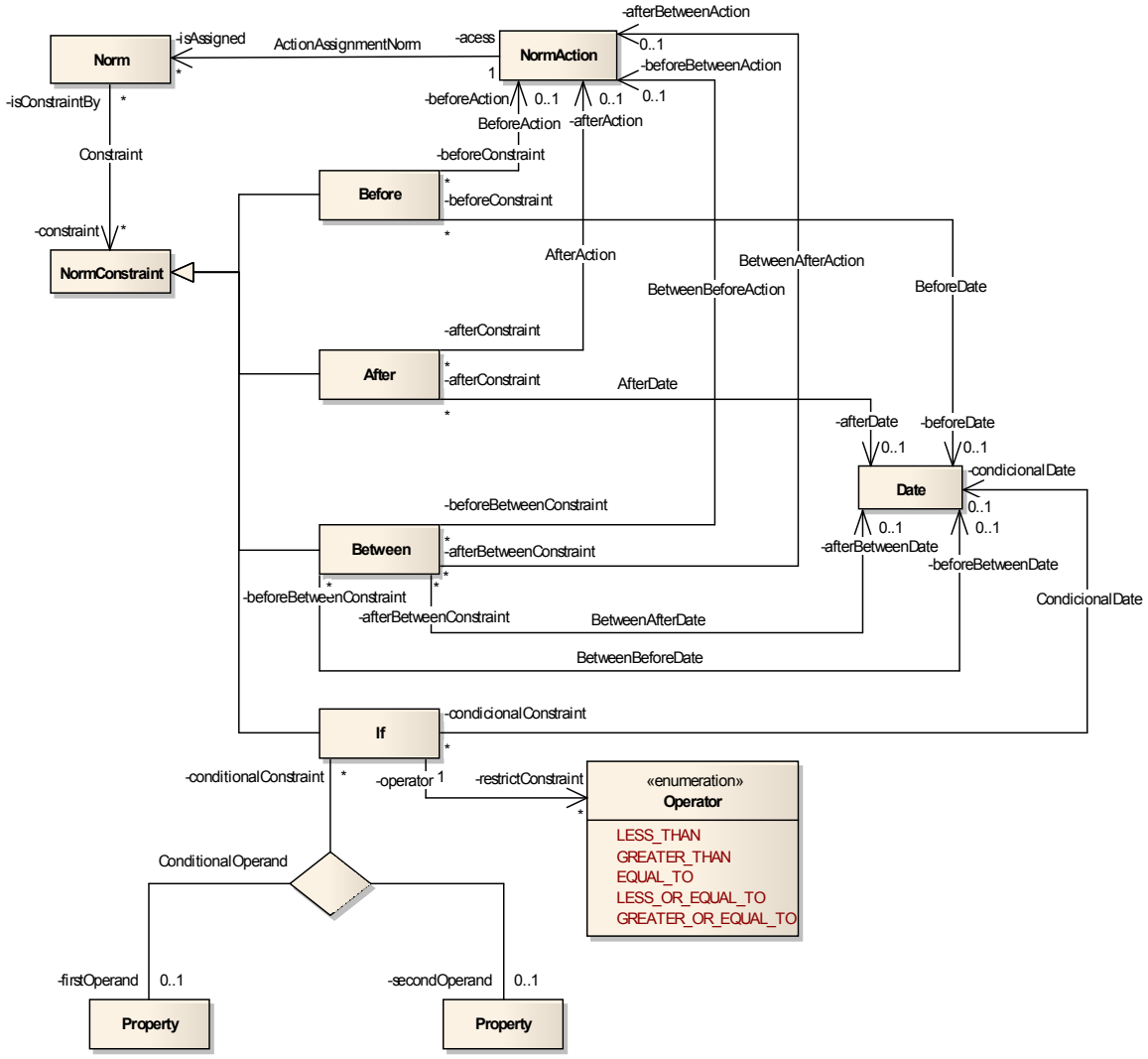


Figura 10 – Associações entre *NormConstraint* e outras metaclasses de MAS-ML (adaptado) (FIGUEIREDO, 2011)

4.3.4 Recurso

O acesso aos recursos do sistema pode ser controlado através de um conjunto de ações, definido no contexto de normas. De acordo com FIGUEIREDO (2011), um recurso pode ser (i) um atributo, (ii) um método, (iii) uma associação, (iv) um agente, (v) um papel, (vi) uma organização, (vii) um ambiente, (viii) uma ação, (ix) uma mensagem, (x) um

protocolo, (xi) uma crença, (xii) um objetivo e (xiii) um plano. Neste contexto, a metaclasse *Resource* (originária do SecureUML) foi incorporada para possibilitar a modelagem de recursos. A nova metaclasse em MAS-ML foi definida como uma extensão da metaclasse *Element* originária da UML.

Embora a definição em FIGUEIREDO (2011) sugira um relacionamento de herança entre um recurso e as entidades que podem ser consideradas recursos no sistema, a inserção deste relacionamento no metamodelo de MAS-ML foi definido por meio da associação entre as metaclasses *Classifier*, *Feature*, *Association* e *AgentMessage* com a metaclasse *Resource*. Esta decisão de projeto foi tomada no intuito de conservar a estruturação das metaclasses originarias da UML, sem perda de semântica. Com isso, a partir da extensão proposta, um recurso pode estar vinculado a:

- Uma característica estrutural ou comportamental, por meio da associação entre as metaclasses *Resource* e *Feature*. Desta forma, tanto as características estruturais (objetivos, crenças, atributos), quanto comportamentais (métodos, ações, planos e protocolos) se tornam recursos;
- Um classificador, por meio da associação entre as metaclasses *Resource* e *Classifier*. Com isso, as diversas entidades que compõem um SMA são definidas como recurso (um agente, uma organização, um papel de agente ou um ambiente);
- Um relacionamento, por meio da associação entre as metaclasses *Resource* e *Association*, de forma a restringir o acesso à leitura e atualização; e
- Uma mensagem, por meio da associação entre as metaclasses *Resource* e *AgentMessage* estabelece que mensagens de agente também sejam recursos do sistema que podem ter seu acesso restringido.

A Figura 11 apresenta a metaclasse *Resource* e os novos relacionamentos definidos no metamodelo de MAS-ML.

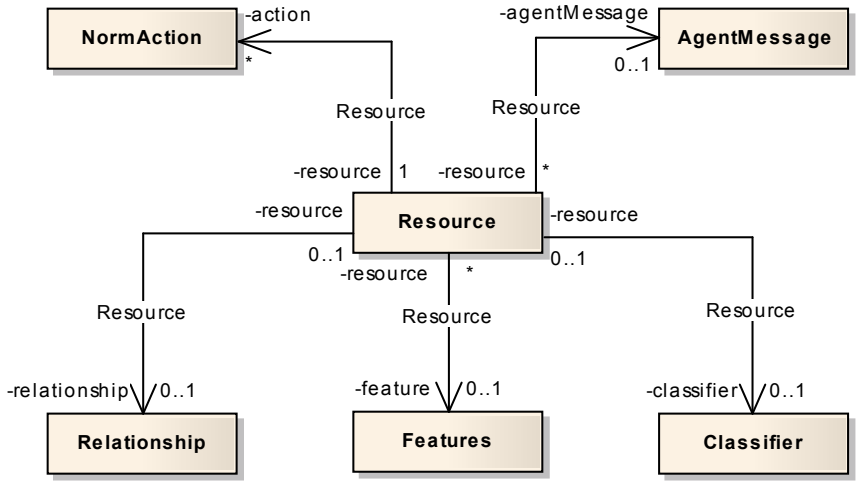


Figura 11 – Associações entre *Resource* e outras metaclasses em MAS-ML

4.3.5 O Relacionamento *Sanction*

Quando uma norma é violada/cumprida, a entidade que violou/cumpriu essa norma pode receber uma sanção (FIGUEIREDO, 2011). A introdução destes conceitos em MAS-ML levou à incorporação da metaclasse *Sanction*, originária de NormML. Essa metaclasse foi adequada ao metamodelo de MAS-ML e passou a estender a metaclasse *DirectedRelationship* de MAS-ML. A adequação dessa metaclasse foi necessária, pois no metamodelo da MAS-ML não existia nenhuma metaclasse que representasse esse conceito, que corresponde aos relacionamentos *SanctionReward* e *SanctionPunishment* definidos na extensão realizada no *framework* TAO (Seção 4.1.2). Estes relacionamentos só podem ser definidos entre normas, pois as sanções são utilizadas para recompensar e/ou punir as entidades que estão sendo restringidas pelas normas.

Diferentemente de NormML, onde as sanções de punição e de recompensa são modeladas através de duas metaclasses, em MAS-ML estes conceitos foram definidos como estereótipos de *Sanction,* uma vez que compartilham a mesma estrutura que a referida metaclasse. Com isso, os estereótipos <<*punishment*>> e <<*reward*>> foram criados para representar, respectivamente, a punição e a recompensa relativas a uma determinada norma. A Figura 12 apresenta os novos estereótipos definidos.

Figura 12 – Estereótipos da metaclasse *Sanction*

4.3.6 O Relacionamento *Context*

Uma norma precisa ter definido o seu contexto ou domínio de aplicação, de forma que todas as entidades que fizerem parte desse contexto terão o seu comportamento

restringido. A partir das propriedades definidas na extensão realizada no *framework* TAO, a introdução deste conceito em MAS-ML levou à criação da metaclasse *Context*, que estende a metaclasse *DirectedRelationship*. A criação se fez necessária, pois no metamodelo da NormML, esse conceito era representado por meio de uma associação e em MAS-ML, não existia nenhuma entidade que representasse esse conceito.

Este relacionamento pode ser aplicado entre uma Norma e as seguintes entidades: (i) Organização, (ii) Suborganização e (iii) Ambiente.

4.3.7 O Relacionamento *Restrict*

A nova metaclasse *Restrict*, que estende a metaclasse *DirectedRelationship*, foi criada para representar o relacionamento *restrict* definido na extensão realizada no *framework* TAO. Não foi possível reutilizar nenhuma entidade de NormML, pois no seu metamodelo, o conceito de restrição era representado por meio de uma associação.

Através desse relacionamento é estabelecida qual a entidade que será restringida pela norma. Este relacionamento pode ser aplicado entre uma norma e as seguintes entidades: (i) Agente, (ii) Papel de Agente, (iii) Organização, (iv) Suborganização e (v) Ambiente.

A Figura 13 apresenta um subconjunto de metaclasses do metamodelo de MAS-ML e as extensões feitas para a incorporação dos elementos das normas definidos em (FIGUEIREDO, 2011). Essa figura apresenta as novas metaclasses na extensão realizada em MAS-ML.

4.4 Extensão da Sintaxe Concreta de MAS-ML

Ao introduzir novas abstrações no metamodelo de MAS-ML, precisamos criar novos elementos gráficos para representar nos diagramas as entidades e os relacionamentos definidos na sintaxe abstrata de MAS-ML (Seção 4.3). Adicionalmente, as entidades *AgentRoleClass* e *OrganizationClass* tiveram sua estrutura alterada devido á inclusão dos conceitos das normas em MAS-ML. Com isso, os seus respectivos elementos de diagrama tiveram que ser alterados. Entretanto, mesmo com a alteração realizada na entidade *EnvironmentClass*, não foi necessária a alteração na sua representação gráfica no diagrama, pois o mesmo tratava os recursos e serviços como seus atributos e métodos. Com isso, é possível utilizar esses atributos e métodos em associação com normas para restringir o acesso por outras entidades.

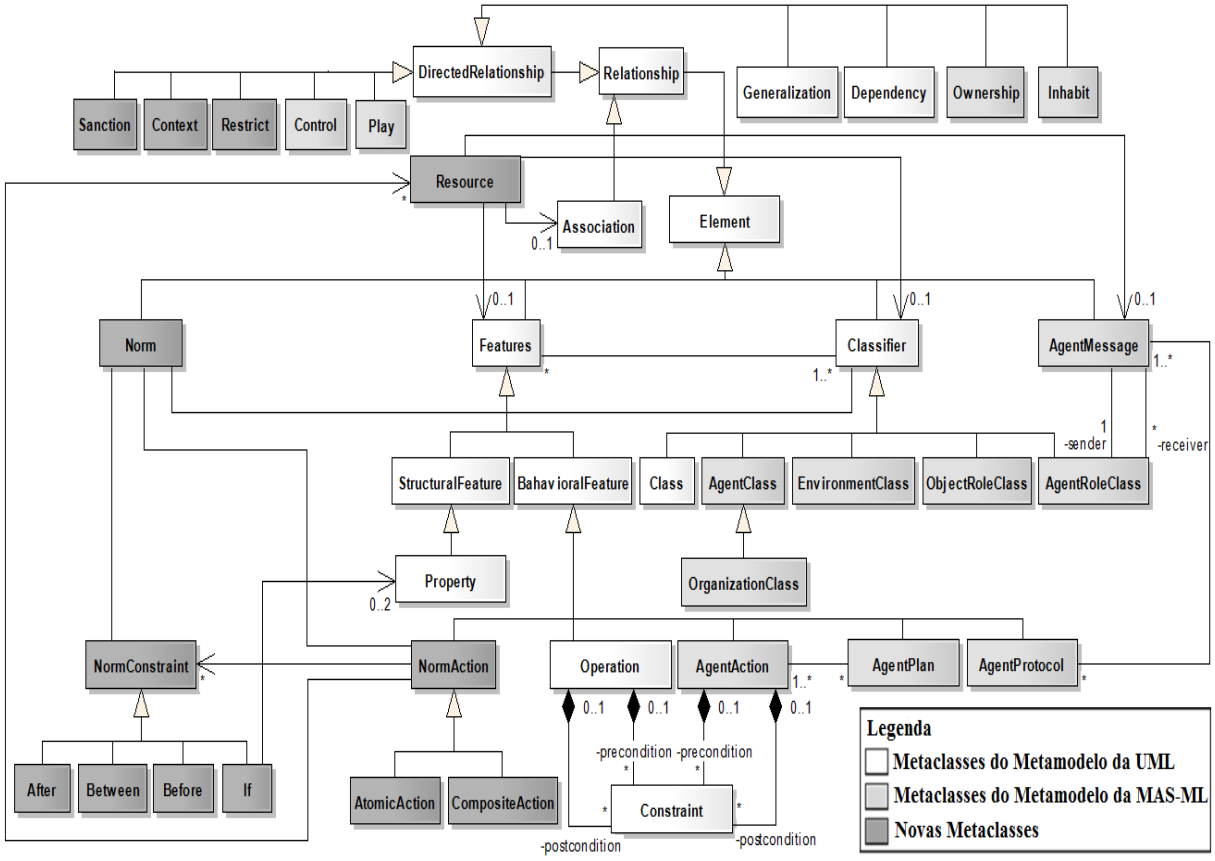


Figura 13 – O metamodelo de MAS-ML estendido para incorporar as entidades definidas para normas e suas propriedades

SILVA (2004) propôs um conjunto de diagramas estruturais que possibilita a modelagem das entidades e dos relacionamentos definidos no TAO. Entretanto, esses diagramas não possibilitam a modelagem dos aspectos relacionados com as normas. Por isso, além da criação dos elementos gráficos, foi criado um novo diagrama estrutural, o diagrama de normas.

Nessa Seção são apresentados (i) os novos elementos gráficos para a entidade *Norm* e os relacionamentos *Context*, *Restrict* e *Sanction*; (ii) os novos elementos gráficos para as entidades Papel de agente (*AgentRoleClass*) e Organização (*OrganizationClass*) e (iii) o novo diagrama estrutural de Normas.

4.4.1 Novos Elementos de Diagramas Estruturais

Com o objetivo de representar os conceitos incluídos na sintaxe abstrata, novos elementos de diagrama são necessários. A representação gráfica da entidade *Norm* e dos relacionamentos *Context*, *Restrict* e *Sanction* são apresentados a seguir.

4.4.1.1 *Norm*

Uma norma é apresentada como um retângulo sólido com um ângulo no canto superior direito e outro no canto inferior esquerdo (ver Figura 14). O compartimento superior contém o nome da classe da norma, que deve ser único em seu espaço de nome. Adicionalmente, deve ser informado o conceito deôntico que a norma está representando utilizando um dos estereótipos da metaclasse *Norm*: *<<permission>>*, *<<obligation>>* ou *<<prohibition>>*. *O* compartimento intermediário possui o recurso que será restringido pela norma, que pode ser uma entidade ou uma propriedade de uma entidade. As entidades que podem ser restringidas são: (i) agente, (ii) papel de agente, (iii) organização, ou (iv) ambiente. Uma propriedade restringida pode ser: (i) objetivo, (ii) crença, (iii) atributo, (iv) método, (v) ação, (vi) plano, (vii) protocolo, (viii) associação ou (ix) mensagem. Adicionalmente, deve-se definir o tipo de restrição através dos estereótipos definidos para a metaclasse *NormAction* (Figura 8).

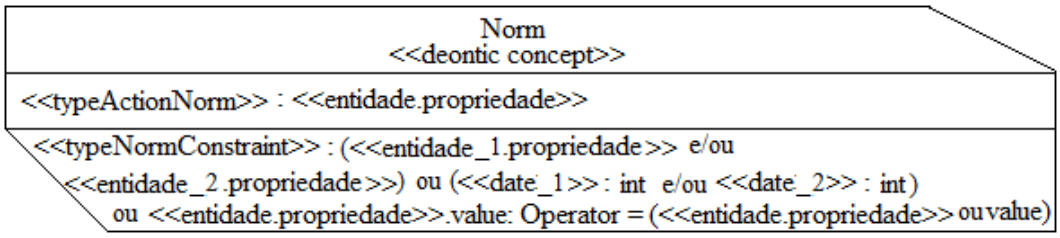


Figura 14 – Elemento Gráfico de *Norm*

O compartimento inferior possui as restrições de ativação da norma. Para definirmos uma restrição, temos que informar o tipo de restrição de ativação. Para isso, utilizamos um dos estereótipos definidos na metaclasse *NormConstraint*. Caso a restrição seja baseada em uma propriedade de outra entidade ou em datas, em ambos os casos, deve-se fazer a sua identificação neste compartimento. Nas restrições condicionais, além de informar os operandos que fazem parte da condição de ativação da norma, deve-se informar o tipo de comparação que será feita entre eles, para isso, utilizamos um dos atributos definidos na metaclasse *Operator*.

4.4.1.2 O Relacionamento *Context*

O relacionamento *Context* liga a classe *Norm* a uma (i) *OrganizationClass* ou (ii) *EnvironmentClass*. Esse relacionamento é apresentado como uma linha simples com um triângulo invertido em uma extremidade. O triângulo indica o contexto no qual a norma será aplicada. A Figura 15 ilustra esse relacionamento.

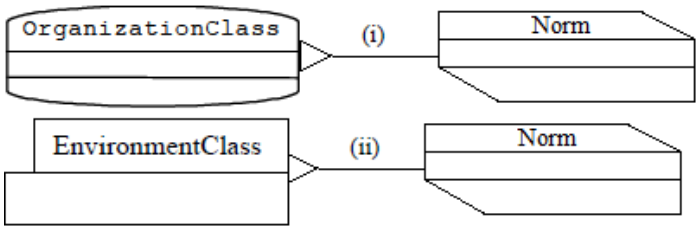


Figura 15 – O relacionamento *Context* pode se associar a uma (i) *OrganizationClass* ou (ii) *EnvironmentClass*

4.4.1.3 O Relacionamento *Restrict*

O relacionamento *Restrict* liga a classe *Norm* a uma *AgentClass*, *OrganizationClass*, *AgenteRoleClass* ou *EnvironmentClass*. Esse relacionamento é apresentado como uma linha simples com um quadrado preenchido em uma extremidade. O quadrado preenchido indica a entidade que terá o comportamento restringido pela norma. A Figura 16 apresenta esse relacionamento.

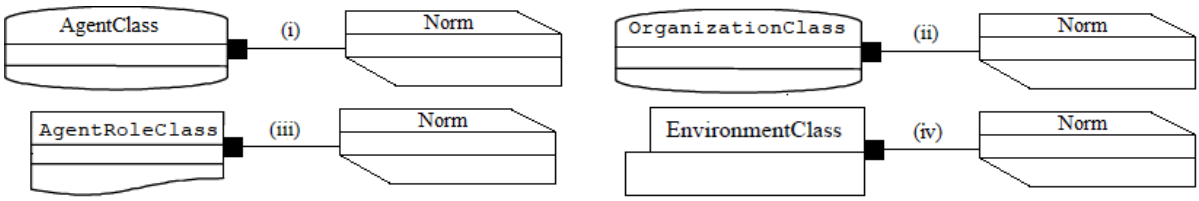


Figura 16 – O relacionamento *Restrict* pode se associar a uma (i) *AgentClass*, (ii) *OrganizationClass*, (iii) *AgentRoleClass* ou (iv) *EnvironmentClass*

4.4.1.4 O Relacionamento *Sanction*

O relacionamento *Sanction* liga à classe *Norm* a outra classe *Norm*. Esse relacionamento é apresentado como uma linha simples com um pentágono em uma extremidade. O pentágono indica a sanção da norma. Caso o pentágono esteja preenchido, isso significa que a sanção da norma é uma punição. Caso contrário, isso significa que a sanção da norma é uma recompensa. A Figura 17 apresenta esse relacionamento.

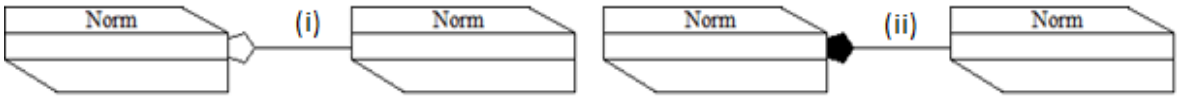


Figura 17 – O relacionamento *Sanction* de (i) recompensa (*reward*) e (ii) punição (*punishment*)

4.4.2 Alteração nos Elementos de Diagramas Estruturais

Como apresentado na Seção 4.3.1, a estrutura das entidades *OrganizationClass* e *AgentRoleClass* foram modificadas devido à inclusão dos conceitos das normas no

metamodelo de MAS-ML. Portanto, os elementos de diagramas que representam essas entidades também precisam ser modificados para refletir a nova estrutura dessas entidades. A seguir, essas alterações serão apresentadas.

4.4.2.1 *OrganizationClass*

Segundo SILVA (2004), uma organização é apresentada na forma de um losango com três compartimentos separados por linhas horizontais. O compartimento superior contém o nome da classe da organização que deve ser único em seu espaço de nome incluído. O compartimento intermediário contém a lista de objetivos, crenças e axiomas da organização, e o inferior, uma lista de ações e planos da mesma. Esses dois compartimentos podem ser omitidos, quando necessário.

Devido à eliminação do conceito de axioma na sintaxe abstrata, que correspondia ao estereótipo <<*axiom*>>, e a substituição desse conceito por normas, o compartimento intermediário teve que ser alterado por meio da eliminação da lista de axiomas. Agora, a partir das adequações propostas, uma organização pode estar associada a uma ou mais normas e restringir o comportamento dos agentes e suborganizações associadas a ela por meio do relacionamento *Context.* A Figura 18 apresenta o elemento de diagrama da organização modificado.

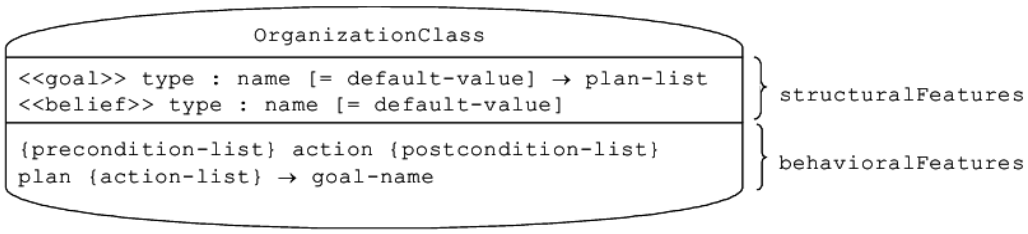


Figura 18 – O elemento de diagrama *OrganizationClass* alterado

4.4.2.2 *AgentRoleClass*

Segundo SILVA (2004), um papel de agente é representado por um retângulo sólido com uma curva na parte inferior, que possui três compartimentos separados por linhas horizontais. O compartimento superior contém o nome de papel do agente que deve ser único em seu espaço de nome incluído. O compartimento intermediário de lista contém a lista de objetivos e crenças associados ao papel, e o inferior, uma lista de deveres, direitos e protocolos.

Devido à eliminação dos conceitos de direito e dever na sintaxe abstrata da linguagem que correspondiam aos estereótipos <<*duty*>> e <<*right*>>, e a substituição desses

conceitos por normas, o compartimento inferior na representação de *AgentRoleClass* precisou ser alterado por causa da eliminação da lista de deveres e direitos. Em compensação, foi incluída uma lista de ações que deve ser utilizada para indicar quais ações do papel de agente, uma determina norma irá restringir. Com isso, é possível restringir o comportamento de agentes e suborganizações que estão associadas a um determinado papel por meio de normas. A Figura 19 apresenta o elemento de diagrama do papel de agente modificado.

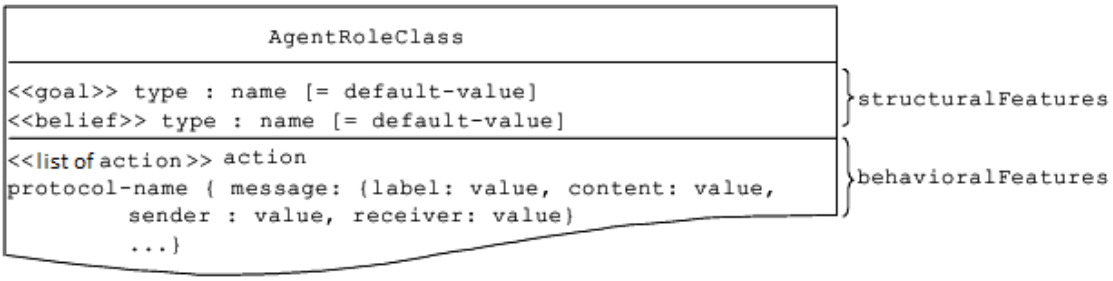


Figura 19 – O elemento de diagrama *AgentRoleClass* alterado

4.4.3 Diagrama de Normas

A partir dos novos conceitos definidos em MAS-ML, o diagrama estático de normas é proposto com o objetivo de modelar todas as normas que governam um SMA, e suas propriedades, tais como: (i) o contexto, (ii) o recurso que será restringido, (iii) entidade que terá o comportamento restringido, (iv) as restrições de ativação, e (v) as sanções.

As entidades que podem participar do diagrama de normas são: (i) classe de norma, (ii) classe, (iii) classe de papel de objeto, (iv) classe de agente, (v) classe de papel de agente, (vi) classe de organização e (vii) classe de ambiente. Os relacionamentos definidos originalmente em MAS-ML que podem ser usados no novo diagrama são:

- *ownership*: usado entre classes de organização e as classes de papel definidas pela organização;
- *play*: usado entre classes de agente e classes de papel de agente, entre classes de suborganização e classes de papel de agente, e entre classes e classes de papel de objeto; e
- *inhabit*: usado entre as classes de ambiente e as classes de organização, de papel de agente, de papel de objeto, de objeto, de agente e de norma.

Adicionalmente, os novos relacionamentos definidos na extensão podem ser utilizados:

- *context*: usado entre classes de norma e as classes de ambiente e organização;

- *restrict*: usado entre a classe de norma e as classes de ambiente, organização, papel de agente e agente; e
- *sanction*: usado entre classes de norma.

Exemplos do diagrama de normas podem ser encontrados no Capítulo 6, no qual, dois estudos de estudos de caso são apresentados e modelados utilizando esse novo diagrama.

4.5 Conclusão

Considerando o limitado suporte fornecido pelo *framework* TAO e a linguagem MAS-ML para a modelagem de normas para SMAs, este capítulo apresentou a extensão (i) do *framework* TAO de forma a possibilitar a inclusão dos elementos que compõem as normas definidos na pesquisa de FIGUEIREDO (2011), (ii) da sintaxe abstrata de MAS-ML a partir da reutilização e adequação de algumas metaclasses e relacionamentos definidos em NormML. Adicionalmente novas metaclasses e estereótipos foram criados com base na extensão realizada no TAO, (iii) da sintaxe concreta de MAS-ML, na qual foram definidos novos elementos gráficos para a representação das novas metaclasses incluídas na sintaxe abstrata. Além disso, um novo diagrama estático é proposto de forma a possibilitar a modelagem das normas que regem as entidades de um SMA. A extensão da linguagem MAS-ML aqui proposta é denominada de NorMAS-ML.

A extensão realizada no TAO envolve a definição do novo elemento *Norm* e dos relacionamentos *Context*, *Restrict*, *SanctionReward* e *SanctionPunishment*. Os conceitos de dever (*duty*) e direito (*right*) do elemento *AgentRole*, de regra e lei do elemento *Organization* e as restrições de serviços e recursos do elemento *Environment* se tornaram desnecessários devido à inclusão dos elementos que compõem as normas. Por isso, esses conceitos foram eliminados.

A extensão da sintaxe abstrata de MAS-ML envolve a adequação de 13 metaclasses provenientes de NormML: (i) *Norm*, (ii) *Resource*, (iii) *NormConstraint*, (iv) *After*, (v) *Between*, (vi) *Before*, (vii) *If*, (viii) *Date*, (ix) *Operator*, (x) *NormAction*, (xi) *AtomicAction*, (xii) *CompositeAction* e (xiii) *Sanction*. Adicionalmente, foram definidas as metaclasses *Context* e *Restrict* juntamente com os novos estereótipos para as metaclasses *Norm*, *Sanction*, *AtomicAction* e *CompositeAction*. Estes estereótipos possibilitam a diferenciação entre normas de permissão, obrigação e proibição, sanções de recompensa e punição e os tipos de ações atômicas e de composição. Devido à inclusão dos conceitos de

normas na sintaxe abstrata, foi necessário eliminar os estereótipos <<*duty*>> e <<*right*>>, ambos de *AgentRoleClass*, e <<*axiom*>> de *OrganizationClass*.

Na extensão da sintaxe concreta, foram definidos novos elementos gráficos para representar a metaclasse *Norm* e os relacionamentos *Context*, *Restrict* e *Sanction* nos respectivos diagramas. Adicionalmente, os elementos de diagrama das entidades *AgentRoleClass* e *OrganizationClass* foram alterados devido à eliminação dos estereótipos <<*duty*>>, <<*right*>> e <<*axiom*>> da sintaxe abstrata. Com isso, a lista de direitos e deveres da entidade *AgentRoleClass* e a lista de axiomas da entidade *OrganizationClass* foram removidas. Adicionalmente, foi incluída a lista de ações na entidade *AgentRoleClass*.

A partir da extensão realizada nas sintaxes abstrata e concreta de MAS-ML, um novo diagrama estrutural foi definido: o diagrama de normas, que tem como objetivo modelar todas as normas de um SMA. Portanto, a nova versão denominada NorMAS-ML possibilita a modelagem de todos os elementos que compõem as normas juntamente com as entidades típicas de um SMA.

5 EXTENSÃO DA FERRAMENTA DE MODELAGEM

Neste capítulo é apresentado o processo de extensão da ferramenta MAS-ML *tool* para dar apoio à modelagem de Sistemas Multi-Agente Normativos com base nas sintaxes abstrata e concreta de NorMAS-ML definidas no capítulo anterior.

5.1 Escolha da Ferramenta

Atualmente, existem duas ferramentas para modelagem de SMAs utilizando MAS-ML: *VisualAgent* (DE MARIA et al., 2005) e MAS-ML *tool* (FARIAS et al., 2009) (GONÇALVES et al., 2011). *VisualAgent* é um ambiente de desenvolvimento de software que tem a finalidade de auxiliar desenvolvedores na especificação, projeto e implementação de SMAs. Ela também propõe uma abordagem dirigida por modelo para o desenvolvimento de SMAs fazendo uso do ASF (*Agent Society Framework*) (SILVA; CORTÉS; LUCENA, 2004), o que representa um ponto relevante da abordagem.

Por outro lado, MAS-ML *tool* é um ambiente de modelagem específico de domínio que atende à modelagem de SMAs, de acordo com a especificação do metamodelo definido na concepção original de MAS-ML. Posteriormente a ferramenta foi evoluída por GONÇALVES (2009) de forma a possibilitar a modelagem de agentes com diferentes arquiteturas internas definidas por RUSSELL e NORVIG (2004). O objetivo de MAS-ML *tool* é fornecer um ambiente de modelagem no qual os desenvolvedores possam trabalhar com os conceitos do domínio do problema, ao mesmo tempo em que utilizam explicitamente conceitos definidos no domínio da solução, neste caso os conceitos e abstrações definidos no paradigma de SMAs.

Uma das principais vantagens da MAS-ML *tool* em relação à ferramenta *VisualAgent* é a capacidade de realizar verificação do modelo em relação ao metamodelo da MAS-ML. A ausência desta funcionalidade na *VisualAgent* pode comprometer as sucessivas transformações dos modelos presentes na abordagem MDD (MELLOR; CLARK; FUTAGAMI, 2003) proposta e levar a um mau entendimento dos modelos criados, além de que um modelo inconsistente pode gerar código com erros. Adicionalmente, a falta de documentação e acesso ao código fonte levaram à escolha da ferramenta MAS-ML *tool* para hospedar a implementação da extensão proposta à linguagem MAS-ML.

5.2 MAS-ML *Tool*

MAS-ML *tool* é um ambiente de modelagem específico de domínio que atende à modelagem de SMAs. Por meio deste ambiente, os desenvolvedores podem trabalhar com os conceitos do domínio do problema, ao mesmo tempo em que utilizam explicitamente conceitos definidos no domínio da solução, neste caso os conceitos e abstrações relativos ao paradigma de SMAs.

Atualmente, a ferramenta possibilita a modelagem de todos os diagramas estáticos: (i) de classe (FARIAS et al, 2009), (ii) de organização (GONÇALVES et al, 2011) e de papel (FEIJÓ, 2012). Na sua versão mais atual, somente o diagrama dinâmico de sequência é suportado pela ferramenta (FEIJÓ, 2012).

O ambiente foi desenvolvido como um *plug-in* da plataforma Eclipse (ECLIPSE, 2011). Isso implica que os usuários podem trabalhar com modelagem de SMAs ao mesmo tempo em que fazem uso dos recursos oferecidos pela plataforma Eclipse. Dado que muitas plataformas de agentes são implementadas em Java, tais como Jade (BELLIFEMINE; CAIRE; GREENWOOD, 2007), Jadex (POKAHR; BRAUBACH; LAMERSDORF, 2003), Jason (BORDINI; WOOLDRIDGE; HÜBNER, 2007), o uso da plataforma Eclipse também facilita uma possível geração de código dentro do mesmo ambiente de desenvolvimento.

A ferramenta foi desenvolvida utilizando uma abordagem dirigida por modelos, onde o modelo central e de maior abstração é o próprio metamodelo da linguagem MAS-ML. A Figura 20 apresenta uma visão geral da MAS-ML *tool* juntamente com alguns dos seus componentes principais. As letras representam os seguintes recursos:

(A) ***Package Explorer*** - Tem como funcionalidade central permitir a organização dos arquivos em uma estrutura de árvore a fim de que seja possível um melhor gerenciamento e manipulação dos arquivos;

(B) ***Modeling View*** - Os modelos que são criados precisam necessariamente ser visualizados com o objetivo de atender dois requisitos básicos quando se usa modelos: compreensibilidade e comunicação. Diante dessa necessidade, a *modeling view* permite os desenvolvedores visualizar e editar os modelos de forma interativa;

(C) ***Nodes Palette*** - Os construtores que podem fazer parte dos diagramas encontram-se na *nodes palette*. Sendo assim, os desenvolvedores podem criar instâncias desses construtores, as quais são visualizadas na *modeling view*;

(D) ***Relationship Palette*** - Os relacionamentos que podem ser estabelecidos entre os construtores presentes na *nodes palette* são disponibilizados na *relationship palette*;

(E) ***Properties View*** - Permite manipular de forma precisa as propriedades dos modelos. Exibe as propriedades definidas no metamodelo da MAS-ML quando o modelo é exibido e selecionado na *modeling view*;

(F) ***Problems View*** - Caso exista alguma inconsistência no modelo, ela será apresentada em *problems view*. Esta funcionalidade é particularmente importante para viabilizar o uso de modelagem de SMAs dentro do contexto do desenvolvimento dirigido por modelos, no qual modelos são vistos como artefatos de primeira ordem;

(G) ***Outline View*** **-** Uma vez que um modelo tenha sido criado, um *overview* da distribuição dos elementos presentes no mesmo poderá ser visualizado através da *outline view*.

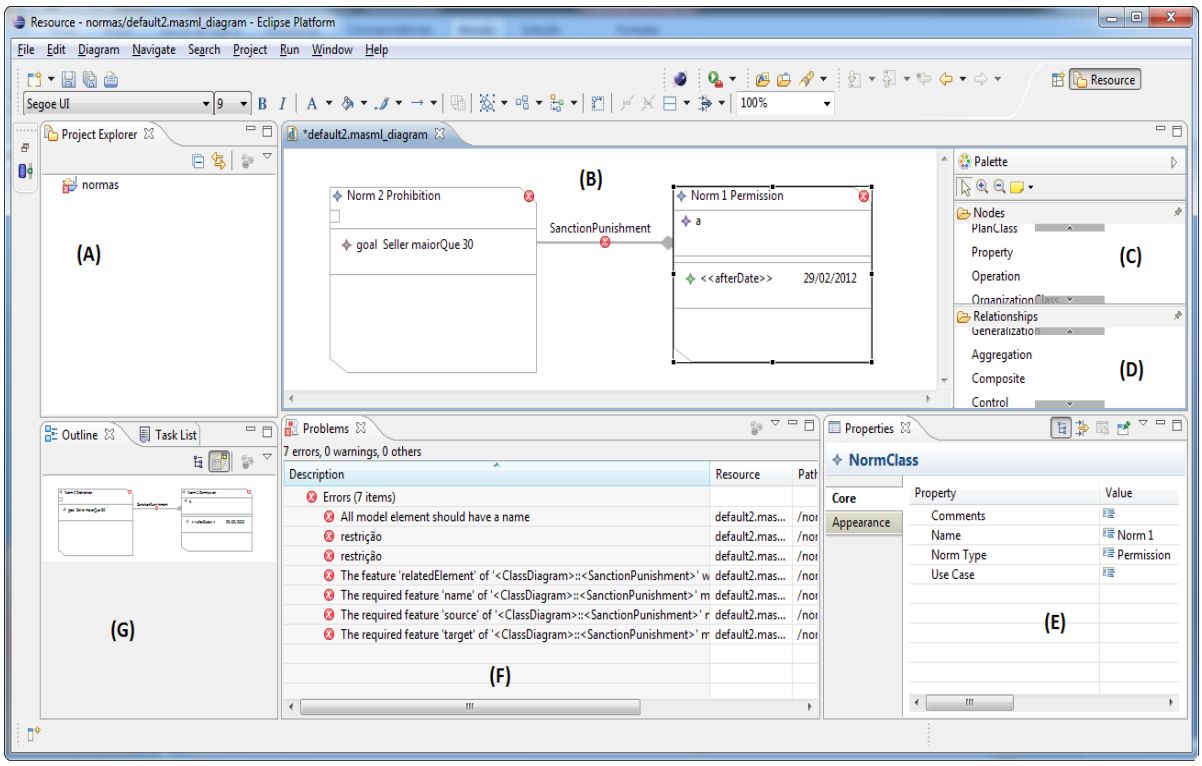


Figura 20 – Visão geral da ferramenta MAS-ML *tool*

5.3 Extensão da Ferramenta MAS-ML *Tool*

A versão da ferramenta (FEIJÓ, 2012) utilizada como base para a sua extensão disponibiliza a modelagem dos três diagramas estáticos definido em MAS-ML: diagrama de classes, de organização e de papéis. No contexto da extensão proposta no presente trabalho, os diagramas de organização e de papéis precisaram ser modificados. Adicionalmente, o diagrama de normas proposto neste trabalho foi criado. Com isso, os modelos criados através desses três diagramas conseguem expressar as alterações realizadas nas entidades de MAS-

ML juntamente com a inclusão das novas entidades que representam os elementos que compõem as normas.

A estratégia adotada para implementar as extensões propostas segue a abordagem dirigida por modelos que foi utilizada originalmente para desenvolver a ferramenta, utilizando, neste caso, o metamodelo da linguagem NorMAS-ML apresentado na Figura 13. A seguir, são descritas, de forma resumida, as seis etapas realizadas no processo de evolução da ferramenta.

5.3.1 Extensão do Modelo de Domínio

Originalmente, o metamodelo foi especificado usando o EMOF[3] (*Essencial Meta-Object Facility*), uma linguagem de definição de metamodelo usada para definição da UML. Em relação à evolução, foram incluídas as metaclasses: (i) *NormClass*, (ii) *ConstraintClass*, (iii) *DateConstraint*, (iv) *CondicionalConstraint*, (v) *NormResource*, (vi) *Context*, (vii) *SanctionPunishment*, (viii) *SanctionReward* e (ix) *Restrict*. A Figura 21 ilustra a criação das novas metaclasses.

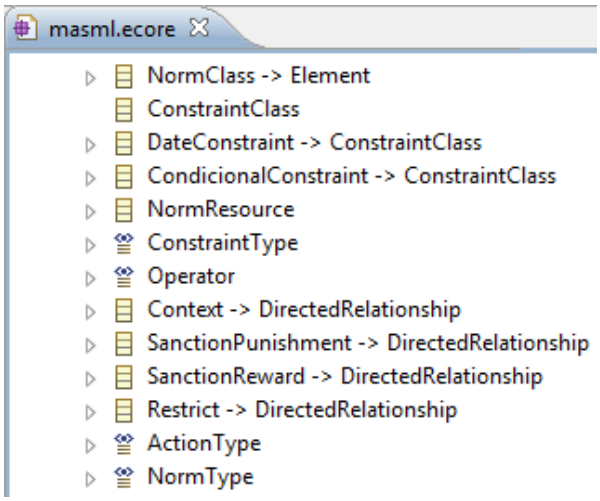


Figura 21 – As novas metaclasses incluídas no ecore da ferramenta

É importante ressaltar que as metaclasses *CondicionalConstraint* e *DateConstraint* definidas na extensão da MAS-ML *tool* equivalem à metaclasse *NormContraint* definida na sintaxe abstrata de NorMAS-ML. Essa divisão foi feita para facilitar a validação dos modelos. Adicionalmente, a metaclasse DateConstraint representa tanto as restrições por data como por ações por meio das cláusulas Before, After e Between.

[3] O EMOF é parte integrante do *plug-in* EMF (*Eclipse Metamodel Framework*) que consiste em um *framework* de modelagem e de geração de código para construir aplicações baseadas em modelos (EMF, 2011).

Adicionalmente, foram definidas as seguintes semânticas para representar os estereótipos que foram definidos na da sintaxe abstrata de NorMAS-ML:

- *NormType* está associada à metaclasse *NormClass* e representa os conceitos deônticos da norma por meio das opções: 0- *Permission*, 1- *Obligation* e 2-*Prohibition*;
- *ActionType* está associada à metaclasse *DateConstraint* e representa os estereótipos definidos para as especializações da metaclasse *NormAction* (Figura 8);
- *Operator* está associada à metaclasse *CondicionalConstraint* e representa as opções: 0- *lessThan*, 1- *greaterThan*, 2- *equalTo*, 3- *lessOrEqualTo* e 4- *greaterOrEqualTo*;
- *ConstraintType* está associada a metaclasse *DateConstraint* e representa as seguintes opções: 0- *afterResource*, 1- *afterDate*, 2- *beforeResource*, 3- *beforeDate*, 4- *betweenResource* e 5- *BetweenDate*.

5.3.2 Extensão do Modelo Gráfico

O Modelo Gráfico contém a definição das entidades e de suas propriedades, assim como de seus relacionamentos, com base no metamodelo da linguagem. No contexto da evolução, as novas metaclasses apresentadas na Seção 4.3 foram adicionadas na Definição do Modelo Gráfico da ferramenta e seus relacionamentos com outras entidades foram criados, como pode ser observado na Figura 22.

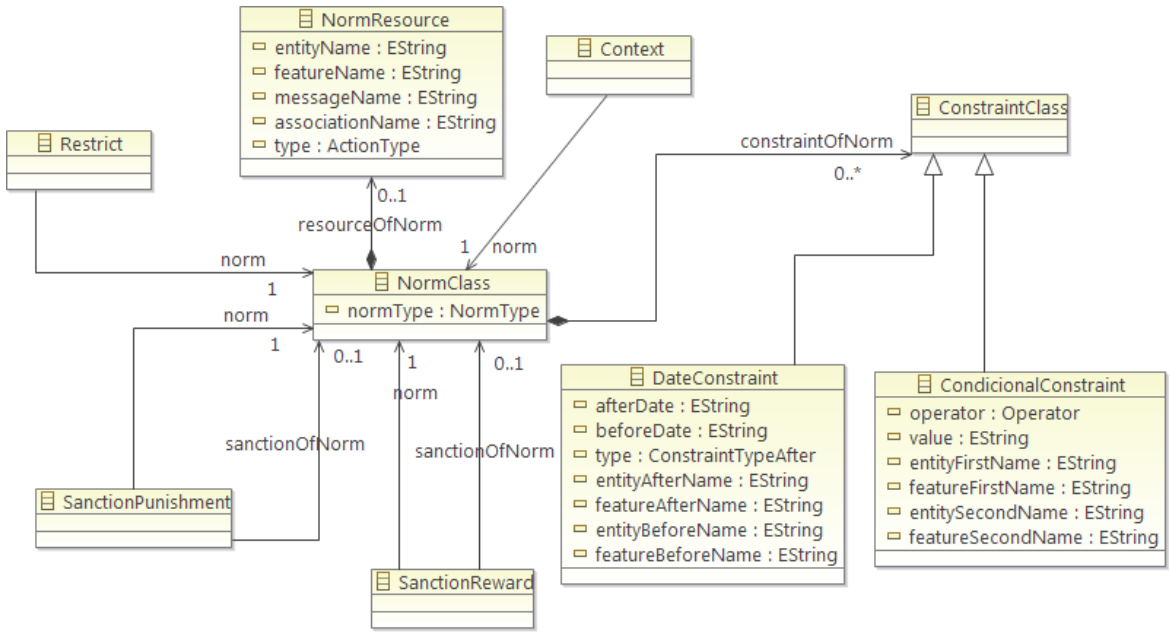


Figura 22 – Representação das novas metaclasses e seus relacionamentos

5.3.3 Extensão do Modelo de Ferramenta

Esta etapa especifica quais elementos fazem parte da paleta da ferramenta com base no modelo de domínio e o modelo gráfico determinados anteriormente. Adicionalmente, para o diagrama de normas foram criados novos elementos na paleta da ferramenta para criar normas e seus relacionamentos. A Figura 23 ilustra os novos elementos adicionados à paleta.

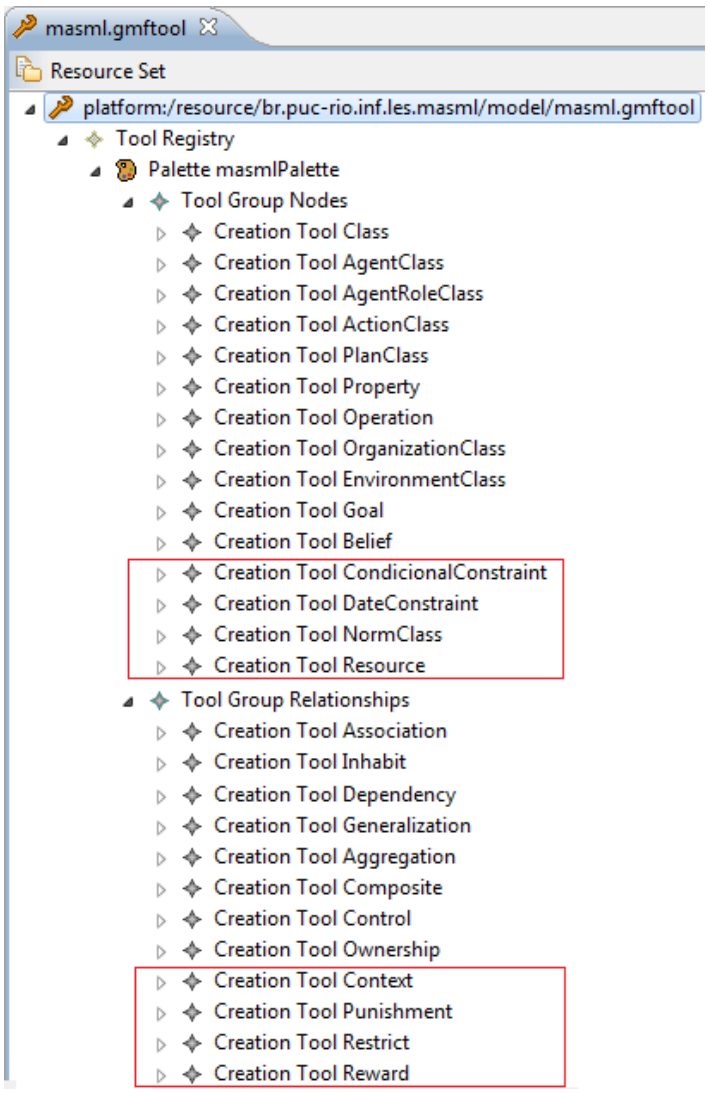


Figura 23 – Novos elementos da paleta no modelo da ferramenta

5.3.4 Extensão da Definição do Modelo Gráfico

Esta etapa está relacionada com a representação dos elementos no respectivo diagrama. O modelo Gráfico, que tem como entrada o modelo de domínio, é utilizado na combinação para gerar o modelo de mapeamento. Para esta etapa, foram criados compartimentos para a representação das normas e seus relacionamentos, além da

representação dos nós (nodes) e conexões provenientes da etapa anterior. Na Figura 24 podem ser vistos os nós, as conexões e os compartimentos do diagrama de normas.

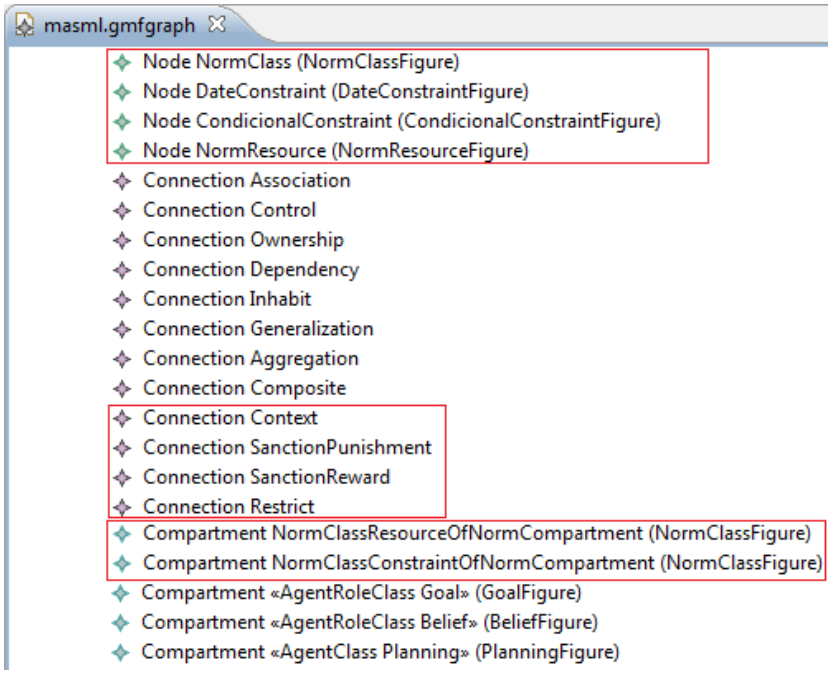


Figura 24 – Elementos do diagrama de normas no modelo gráfico

5.3.5 Extensão do Modelo de Mapeamento

Este modelo é criado a partir de uma combinação dos modelos definidos nas subseções anteriores apresentadas neste capítulo, ou seja, se concentra na construção de um mapeamento entre os três modelos: modelo de domínio, modelo gráfico e o modelo de ferramenta. O mapeamento gerado é usado como entrada em um processo de transformação que tem como objetivo criar um modelo específico de plataforma. As regras de validação para a checagem da corretude dos modelos gerados foram implementadas através da multiplicidade das metaclasses na Definição do Modelo Gráfico da ferramenta e através de restrições em OCL (*Object Constraint Language*) (OCL, 2011). A representação das regras OCL no Eclipse é apresentada na Figura 25.

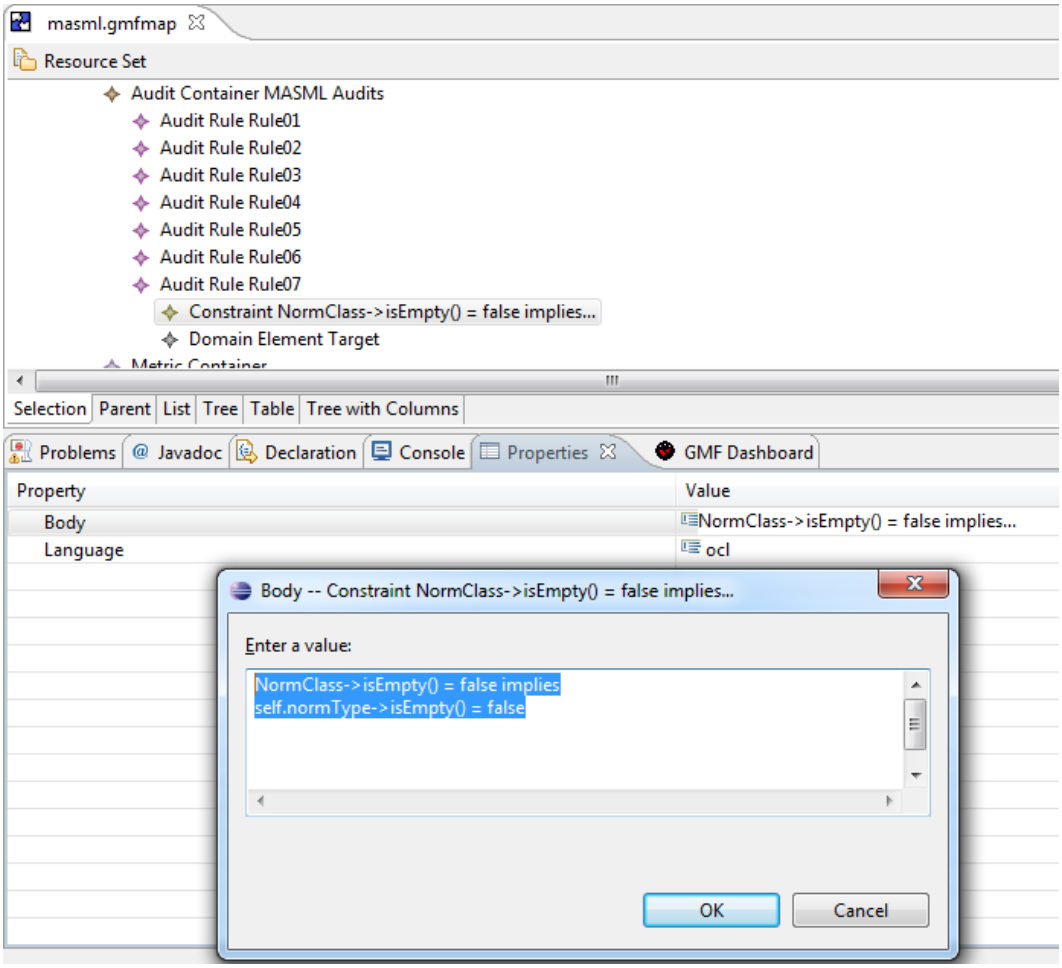


Figura 25 – Representação das regras OCL na ferramenta

A Tabela 3 apresenta a descrição de cada regra definida na ferramenta juntamente com a sua definição em OCL.

Tabela 3 – Regras de validação dos modelos

Regra #	**Descrição**	**Definição em OCL**
01	Toda norma deve possuir um conceito deôntico associado	self.normType->isEmpty() = false
02	Toda norma precisa restringir o comportamento de alguma entidade	self.constraintOfNorm->isEmpty() = false
03	Todo recurso da norma deve possuir um tipo	self.type->isEmpty() = false
04	Um recurso da norma deve ser uma entidade, um atributo de uma entidade, uma mensagem ou uma associação	self.entityName.size() > 0 or self.messageName.size() > 0 or self.associationName.size() > 0 or self.featureName.size() > 0
05	Se o tipo de recurso for uma entidade, os atributos *entity* e *entityName* devem ser informados	self.entityName->isEmpty() = false implies self.entity->isEmpty() = false
06	Se o tipo de recurso for um atributo, os atributos *entity*, *entityName*, *feature* e *featureName* devem ser informados	self.featureName->isEmpty() = false implies self.feature->isEmpty() = false
07	Se o tipo de recurso for uma mensagem, os atributos *message* e *messageName* devem ser informados	self.messageName->isEmpty() = false implies self.message->isEmpty() = false
08	Se o tipo de recurso for uma associação, os atributos *association* e	self.associationName->isEmpty() = false

	associationName devem ser informados	implies self.association->isEmpty() = false
09	Uma restrição condicional de ativação da norma deve possuir um operador	self.operator->isEmpty() = false
10	Uma restrição condicional de ativação deve possuir a comparação entre duas entidades, dois atributos ou um atributo e um valor determinado	(self.entityFirst->isEmpty() = false and self.operator->isEmpty() = false and self.entitySecond->isEmpty() = false) or (self.entityFirst->isEmpty() = false and self.featureFirst->isEmpty() = false and self.operator->isEmpty() = false and self.entitySecond->isEmpty() = false and self.featureSecond->isEmpty() = false) or (self.entityFirst->isEmpty() = false and self.featureFirst->isEmpty() = false and self.operator->isEmpty() = false and self.value->isEmpty() = false)
11	Uma restrição de ativação do tipo data deve possuir um tipo	self.type->isEmpty() = false
12	Se o tipo de restrição for *AfterDate*, deve ser informada uma data	self.type = ConstraintTypeAfter::afterDate implies self.afterDate->isEmpty() = false
13	Se o tipo de restrição for *AfterResource*, deve ser informada uma ação de uma determinada entidade	self.type = ConstraintTypeAfter::afterResource implies self.entityAfter->isEmpty() = false and self.featureAfter->isEmpty() = false and self.typeAfter->isEmpty() = false
14	Se o tipo de restrição for *BeforeDate*, deve ser informada uma data	self.type = ConstraintTypeAfter::beforeDate implies self.beforeDate->isEmpty() = false
15	Se o tipo de restrição for *BeforeResource*, deve ser informada uma ação de uma determinada entidade	self.type = ConstraintTypeAfter::beforeResource implies self.entityBefore->isEmpty() = false and self.featureBefore->isEmpty() = false self.typeBefore->isEmpty() = false
16	Se o tipo de restrição for *BetweenDate*, devem ser informadas duas datas distintas	self.type = ConstraintTypeAfter::betweenDate implies self.beforeDate->isEmpty() = false and self.afterDate->isEmpty() = false
17	Se o tipo de restrição for *BetweenResource*, devem ser informadas duas ações da mesma entidade ou de entidades diferentes	self.type = ConstraintTypeAfter::betweenResource implies self.entityBefore->isEmpty() = false and self.featureBefore->isEmpty() = false and self.typeBefore->isEmpty() = false self.entityAfter->isEmpty() = false and self.featureAfter->isEmpty() = false and self.typeAfter->isEmpty() = false
18	Se o tipo de restrição for *BetweenDate*, a data informada no atributo *beforeDate* não deve ser igual nem superior a data informada no atributo *afterDate*	self.type = ConstraintTypeAfter::betweenDate implies self.beforeDate < self.afterDate

5.3.6 Geração da Ferramenta

Finalmente, seguindo a abordagem generativa (CZARNECKI; EISENECKER, 2000), o próximo passo consiste na geração do código da ferramenta levando em consideração o modelo específico de plataforma que foi estendido na etapa anterior.

Na Figura 26 é apresentado o *Dashboard*, recurso visual do Eclipse utilizado no desenvolvimento seguindo a abordagem generativa com GMF (GMF, 2011), no qual as fases descritas anteriormente podem ser percebidas.

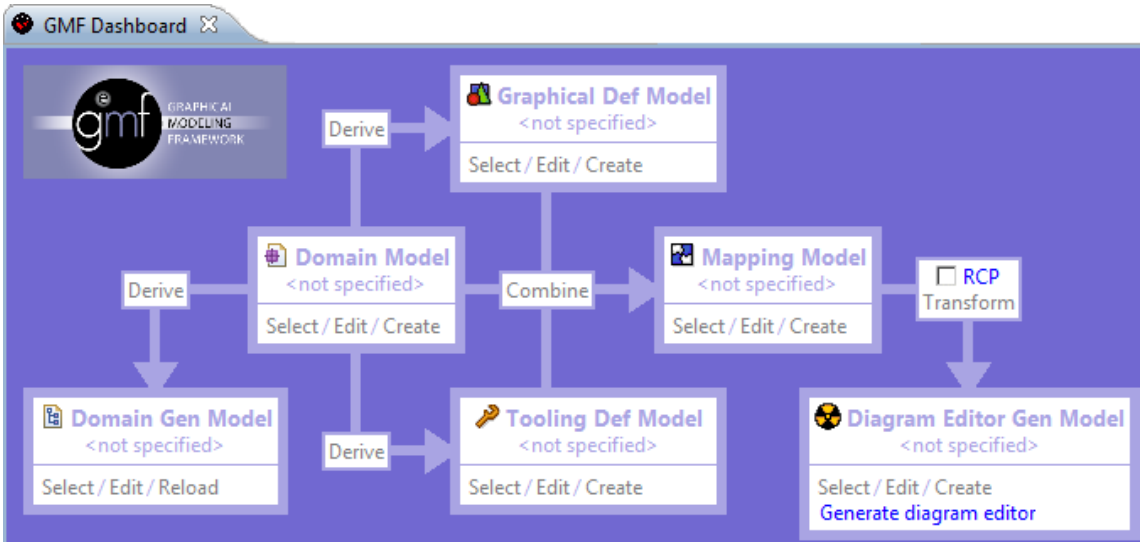


Figura 26 – Elementos do diagrama de normas no modelo gráfico

Após a extensão realizada na ferramenta MAS-ML *tool*, é possível a modelagem de todos os elementos que compõem uma norma apresentados na Seção 2.4. A Figura 27 apresenta os elementos gráficos que representam a entidade *Norm* e os relacionamentos *Context*, *Restrict* e *Sanction (Reward* e *Punishment)* definidos em MAS-ML *tool* na extensão e a Figura 28 apresenta a visão geral do editor do diagrama de normas no qual esses elementos podem ser utilizados.

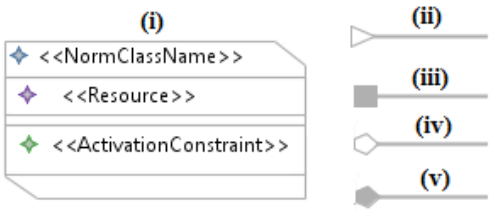


Figura 27 – Os elementos de diagrama que representam a entidade (i) *Norm* e os relacionamentos (ii) *Context*, (iii) *Restrict*, (iv) *Sanction – Reward* e (v) *Sanction – Punishment*

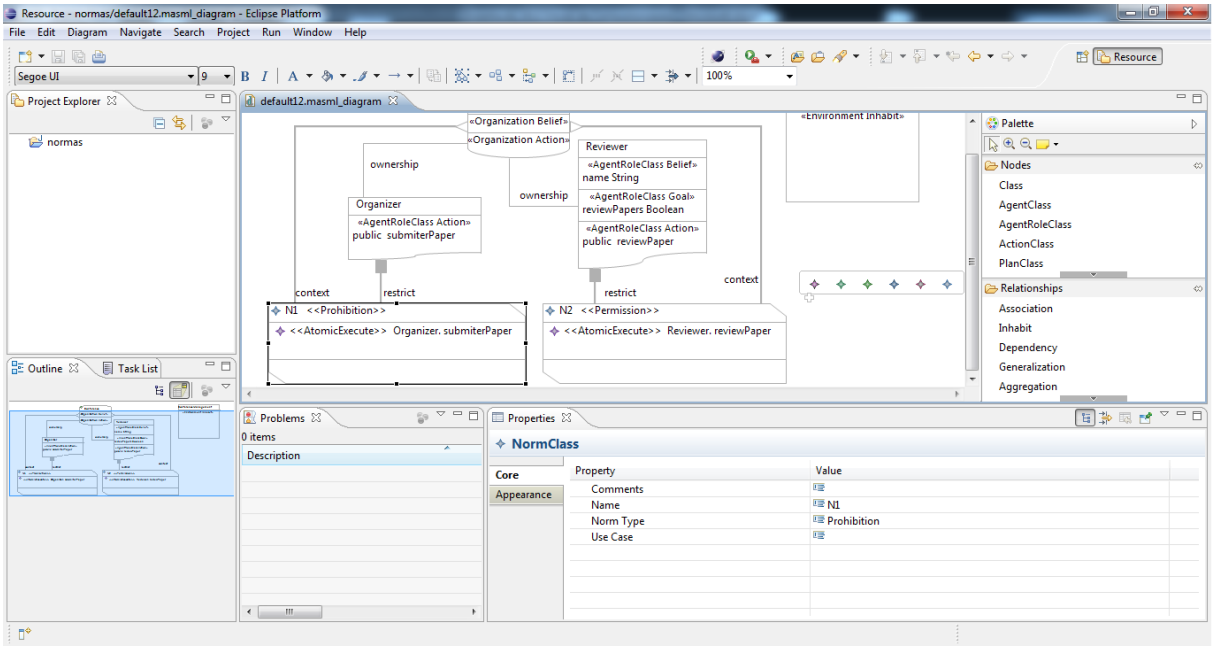


Figura 28 – Visão geral do editor do Diagrama de Normas implementado na MAS-ML *tool*

Além da criação do novo diagrama estático, a representação das entidades Organização (*OrganizationClass*) e Papéis (*AgentRoleClass*) tiveram que ser alteradas devido à extensão realizada nas sintaxes abstrata e concreta de MAS-ML para gerar NorMAS-ML.

Como justificado na Seção 4.4.2, o elemento gráfico que representa a entidade *OrganizationClass* teve seu compartimento intermediário alterado devido à remoção do estereótipo *<<axiom>>*. A Figura 29 apresenta a nova representação visual da entidade *OrganizationClass* na ferramenta MAS-ML *tool*.

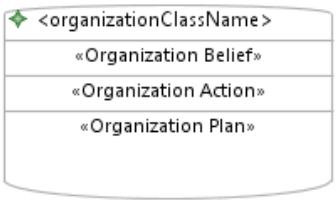


Figura 29 – Nova representação visual da entidade *OrganizationClass*

Similarmente, o elemento gráfico que representa a entidade *AgentRoleClass* teve seu compartimento inferior alterado devido à remoção dos estereótipos *<<duty>>* e *<<right>>*. A Figura 30 apresenta a nova representação visual da entidade *AgentRoleClass* na ferramenta MAS-ML *tool*.

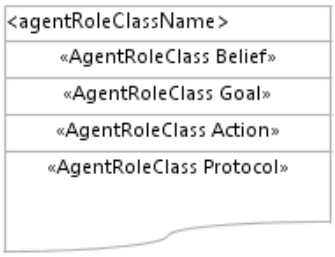


Figura 30 – Nova representação visual da entidade *AgentRoleClass*

As restrições OCL implementadas, as quais validam os modelos gerados a partir da ferramenta podem ser acessadas através do menu *edit* e da opção *validate*. Caso alguma regra seja violada, o elemento que apresenta problema é assinalado e os problemas são apresentados através do recurso *problems* do Eclipse. A Figura 31 apresenta o recurso de validação no Eclipse.

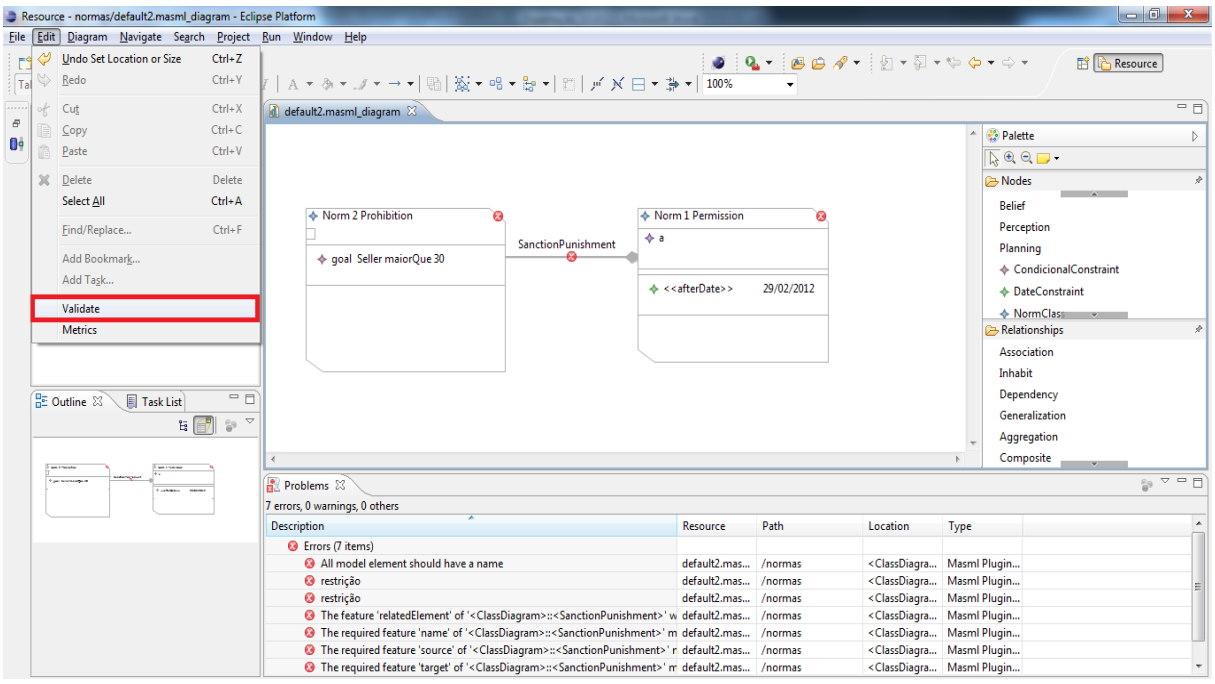


Figura 31 – Validação da ferramenta MAS-ML *tool*

Em consistência com as alterações descritas nos elementos gráficos, as versões da ferramenta que possibilitam a modelagem dos diagramas de organização (GONÇALVES, 2011) e de papéis (FEIJÓ, 2012) também foram alteradas. Para isso, foi utilizado o mesmo processo descrito nesta Seção para a atualização dessas versões da ferramenta. Consequentemente, os elementos gráficos das entidades *OrganizationClass* e *AgentRoleClass* foram alterados conforme citado anteriormente e todas as regras OCL associadas aos conceitos de *duty*, *right* e *axiom* foram retirados.

5.4 Conclusão

Este capítulo apresentou o processo de evolução da ferramenta MAS-ML *tool* em conformidade com NorMAS-ML descrita no Capítulo 4. Neste contexto, a extensão envolveu (i) a criação de nove novas metaclasses e quatro semânticas, (ii) a criação da representação visual dessas novas entidades, (iii) a inclusão das novas entidades e dos novos relacionamentos na paleta, (iv) a definição de novas restrições OCL e (v) a alteração nos elementos visuais das entidades *OrganizationClass* e *AgentRoleClass*. Adicionalmente, as versões da ferramenta que possibilitam a modelagem dos diagramas de Organização (GONÇALVES, 2011) e de Papéis (FEIJÓ, 2012) foram alteradas para refletir as alterações realizadas nos elementos visuais das entidades *OrganizationClass* e *AgentRoleClass*. Com isso, a ferramenta passa a dar suporte à modelagem do diagrama de normas proposto na extensão e dos diagramas de papéis e de organização em conformidade com NorMAS-ML. Com isso, MAS-ML *tool* permite a modelagem das propriedades e relacionamentos das entidades que fazem parte de um sistema multi-agente normativo.

6 ESTUDOS DE CASO

A seguir são apresentados dois estudos de casos: Mercado Virtual e Gestão de Submissão de Artigos. O primeiro estudo de caso foi modelado por SILVA (2004) utilizando a versão original de MAS-ML, enquanto o segundo, foi modelado por FIGUEIREDO (2011) utilizando a linguagem NormML. Os diagramas apresentados neste capítulo foram criados com a versão da ferramenta MAS-ML *tool* apresentada neste trabalho para os diagramas de papéis, organização e de normas. Adicionalmente, para o diagrama de classes, foi utilizada a versão disponibilizada por FARIAS et al. (2009).

6.1 Estudo de Caso 1: Mercado Virtual (*Virtual Marketplace*)

Este estudo de caso foi apresentado e modelado por (SILVA, 2004) utilizando a linguagem MAS-ML, e foi incluído no presente trabalho com o objetivo de avaliar a capacidade de NorMAS-ML na modelagem das entidades. Posteriormente, FIGUEIREDO e SILVA (2010) definiram normas para governar as entidades que fazem parte do Mercado Virtual, as quais serão utilizadas para demonstrar a utilização do diagrama de normas proposto em NorMAS-ML.

Os mercados virtuais são mercados localizados na *Web* em que os usuários compram e vendem itens. Cada um é composto por um mercado principal em que os usuários podem negociar qualquer tipo de item. Além disso, o mercado principal define dois tipos: (i) mercados de produtos especiais que negociam itens caros e de alta qualidade, e (ii) mercados de produtos usados que negociam itens de baixa qualidade e preço baixo.

Os usuários podem: (i) comprar itens no mercado principal, em mercados de produtos especiais e em mercados de produtos usados, e (ii) vender seus itens nos mercados de produtos usados. No mercado principal e nos mercados de produtos especiais, os usuários compram os itens disponíveis no mercado. O mercado principal avalia os lucros, portanto, os mercados de produtos especiais e usados devem enviar as informações relativas às vendas para o mercado principal.

No mercado principal e nos mercados de produtos especiais, os usuários procuram um vendedor e enviam uma descrição do item desejado. O vendedor, criado pelo mercado para negociar com o comprador, é responsável por verificar se há um item com as mesmas características no ambiente (o mercado virtual). O ambiente armazena todos os itens que serão vendidos nesses mercados. Se o item for encontrado, o vendedor negociará com o comprador. Nos mercados de produtos usados, os vendedores e os compradores são os usuários. Os

usuários que desejam vender itens devem anunciá-los. Os usuários que desejam comprar devem procurar anúncios no mercado. Se o comprador encontrar o item desejado, ele começará uma negociação com o vendedor.

6.1.1 Identificação das Entidades do Mercado Virtual

No ambiente *Virtual Marketplace* é possível identificar a organização principal *General Store* e suas duas suborganizações, *Imported Bookstore* e *Second-hand Bookstore* que exercem os papéis Ma*rket of Special Goods* e *Market of Used Goods*, respectivamente. Adicionalmente, foram modelados nesse sistema dois tipos de agente: *user agent*, que pode exercer o papel *buyer*, e *store agent*, que pode exercer os papéis *manager* e *seller*. Estes papéis foram definidos pela organização principal, juntamente com os papéis de objeto *desire* e *offer*. As instâncias desses papéis são exercidas pelas instâncias da classe *Book*, que herda da superclasse *Item* e possui duas subclasses *SecondHandBookstore* e *ImportedBookstore*. A Figura 32 apresenta o diagrama de classes, especificando os relacionamentos entre classes e ambientes.

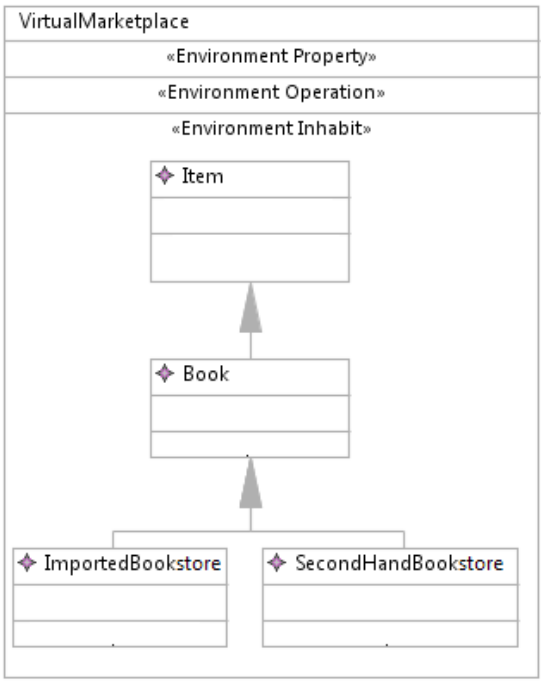


Figura 32 – Diagrama de Classes representado as associações entre classes e ambiente do MercadoVirtual (SILVA, 2004)

A Figura 33 ilustra o diagrama de organização referente à organização principal *General Store*. Para isso, foram definidas as classes da organização principal e das suborganizações, junto com as classes de agente, objeto, papel de agente e papel de objeto.

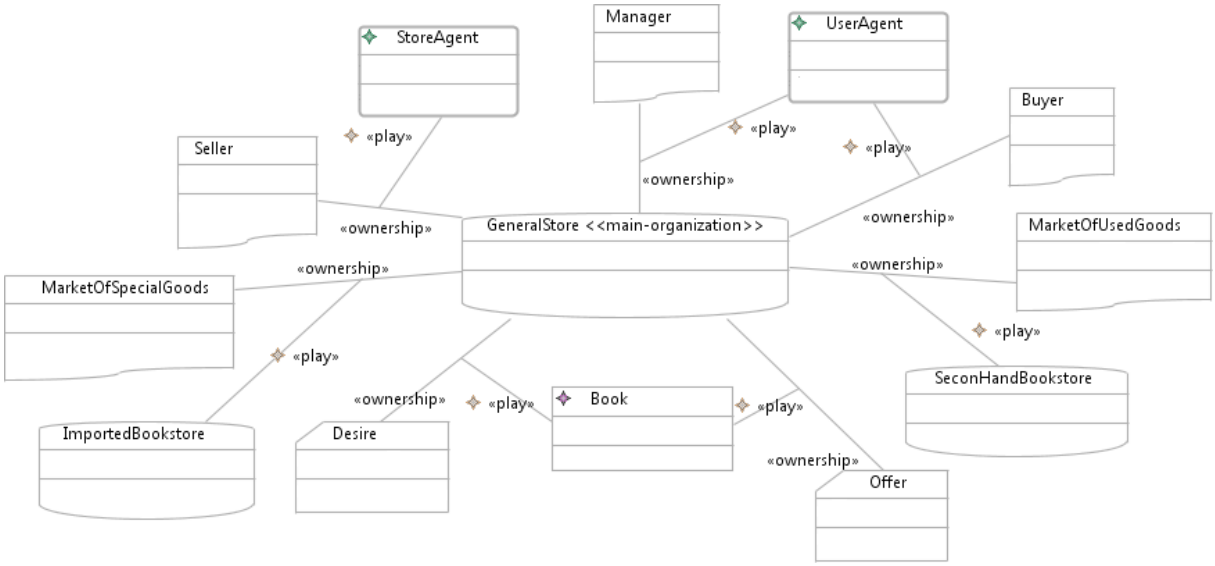


Figura 33 – O Diagrama de Organização representando a organização principal *General Store* (SILVA, 2004)

Para que as normas possam restringir o comportamento das entidades associadas a uma determinada organização, a lista de ações restringidas na entidade organização é explicitamente definida. A Figura 34 apresenta o detalhe da nova representação da classe da organização *General Store* modificada após a eliminação do estereótipo <<*axiom*>>.

Figura 34 – A classe da organização *General Store* modelado com NorMAS-ML

A Figura 35 ilustra o diagrama de papéis identificando os papéis exercidos por agentes e objetos na organização *General Store*.

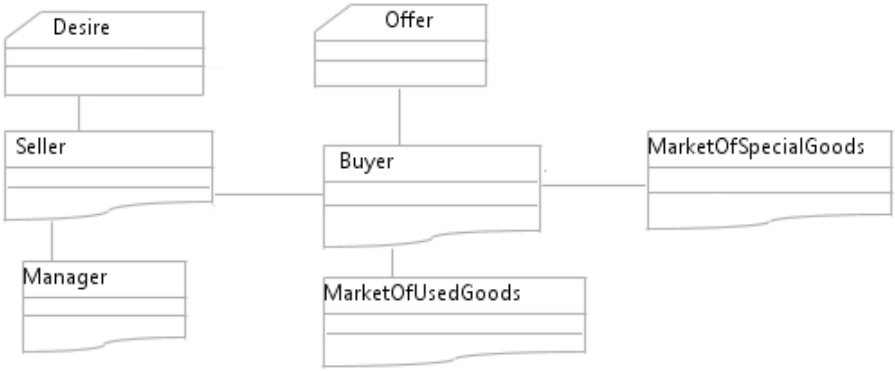


Figura 35 – O Diagrama de papéis referente à organização *General Store* (SILVA, 2004)

A Figura 36 apresenta o detalhe da representação da classe do papel *Buyer* modificada após a eliminação dos estereótipos <<*duty*>> e <<*right*>>. Adicionalmente, para

que as normas possam restringir o comportamento das entidades associadas a um determinado papel a lista de ações restringidas na entidade papel é definida.

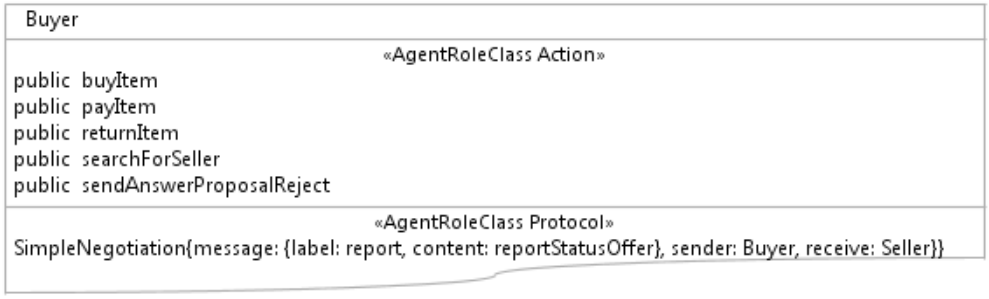


Figura 36 – A classe do papel *Buyer* modelado com NorMAS-ML

Na próxima seção é ilustrada a modelagem das normas do sistema utilizando NorMAS-ML.

6.1.2 Definição das Normas para o Mercado Virtual

Para o mercado virtual definido em FIGUEIREDO e SILVA (2010) foram estabelecidas as seguintes normas:

- N1: Os vendedores da organização *General Store* têm permissão para atualizar o preço de itens antes do recebimento do alerta de vendas feita pelo gerente.
- N2: Os vendedores da organização *General Store* são obrigados a retirar o anúncio de itens que estão em falta no estoque.
- N3: Os compradores da organização *General Store* são obrigados a pagar pelos itens que eles compraram.
- N4 (Punição para a violação da N3): Os compradores que violaram N3 são proibidos de comprar itens.

Adicionalmente, na definição e modelagem proposta em SILVA (2004), as entidades *Market of Special Goods*, *General Store*, *Imported Bookstore*, *Second-hand Bookstore*, *Buyer* e *Seller* definem restrições no comportamento de agentes e suborganizações. As restrições eram especificadas por meio dos estereótipos <<*axiom*>> (em organizações e suborganizações), <<*duty*>> e <<*right*>> (ambos no papel de agente).

Em SILVA (2004), a classe da organização *General Store* também foi modelada utilizando a versão original de MAS-ML, ou seja, com o conceito de axioma representado pelo estereótipo <<*axiom*>>. Nele é determinado que todos os vendedores têm a obrigação de

enviar informação sobre as vendas. Entretanto, em NorMAS-ML, o estereótipo <<*axiom*>> foi eliminado e a restrição das entidades que fazem parte de uma organização passou a ser feito por meio de normas (Figura 34). Logo, a obrigação definida na organização *General Store* passa a ser estabelecida a partir da definição da seguinte norma:

- N5: Os agentes vendedores (papel *Seller*) associados à organização *General Store* devem enviar informações sobre as vendas.

Em SILVA (2004), o papel de agente *Buyer* foi modelado utilizando os conceitos de *duty* e *right* presentes na versão original de MAS-ML. Entretanto, em NorMAS-ML, esses conceitos foram eliminados e a entidade papel de agente passou a restringir o comportamento de suas entidades por meio de normas (Figura 36). Consequentemente, para a completa modelagem do sistema, a obrigação e o direito definidos no papel *Buyer* passaram a ser estabelecidos a partir da formulação de duas novas normas, respectivamente:

- N6: Os agentes compradores vinculados ao papel *Buyer* devem procurar por agentes vendedores.
- N7: Os agentes compradores vinculados ao papel *Buyer* podem enviar mensagem sobre a situação da proposta para um agente vendedor.

6.1.3 Modelagem das Normas para o Mercado Virtual

A seguir é apresentada cada norma juntamente com a sua descrição e identificação das características que a compõe.

A norma N1 estabelece que os vendedores (entidade envolvida) da organização *General Store* (contexto) têm permissão (conceito deôntico) para atualizar (ação) o preço de itens (recurso) antes do recebimento do alerta de vendas feita pelo gerente (restrição de ativação).

A norma N2 determina que os vendedores (entidade envolvida) da organização *General Store* (contexto) são obrigados (conceito deôntico) a retirar o anúncio de itens (ação) que estão em falta no estoque (restrição de ativação).

A norma N3 define que os compradores (entidade envolvida) da organização *General Store* (contexto) são obrigados (conceito deôntico) a pagar pelos itens (ação) depois que eles comprarem esses itens (restrição de ativação). A norma N3 aplica uma punição (sanção) que é a norma N4. Se o comprador violar N3, N4 define que o dado comprador (entidade envolvida) é proibido (conceito deôntico) de comprar itens (ação).

A norma N5 define que todos os agentes vendedores (entidade envolvida) associados ao papel *Seller* da organização *General Store* (contexto) devem (conceito deôntico) enviar a situação sobre suas vendas (ação).

A norma N6 assegura que os agentes compradores vinculados (entidade envolvida) ao papel *Buyer* da organização *General Store* (contexto) devem (conceito deôntico) procurar por agentes vendedores (ação).

A norma N7 define que os agentes compradores vinculados (entidade envolvida) ao papel *Buyer* da organização *General Store* (contexto) podem (conceito deôntico) enviar mensagem sobre a situação da proposta para um agente vendedor (ação).

A Figura 37 apresenta a modelagem das normas N1, N2, N3, N4, N5, N6 e N7 utilizando o diagrama de normas descrito na Seção 4.4.3.

Por meio deste estudo de caso, podemos perceber que NorMAS-ML contempla a modelagem de todas as entidades em SMA definidas na versão anterior, como também possibilita a modelagem de todos os elementos que compõem as normas.

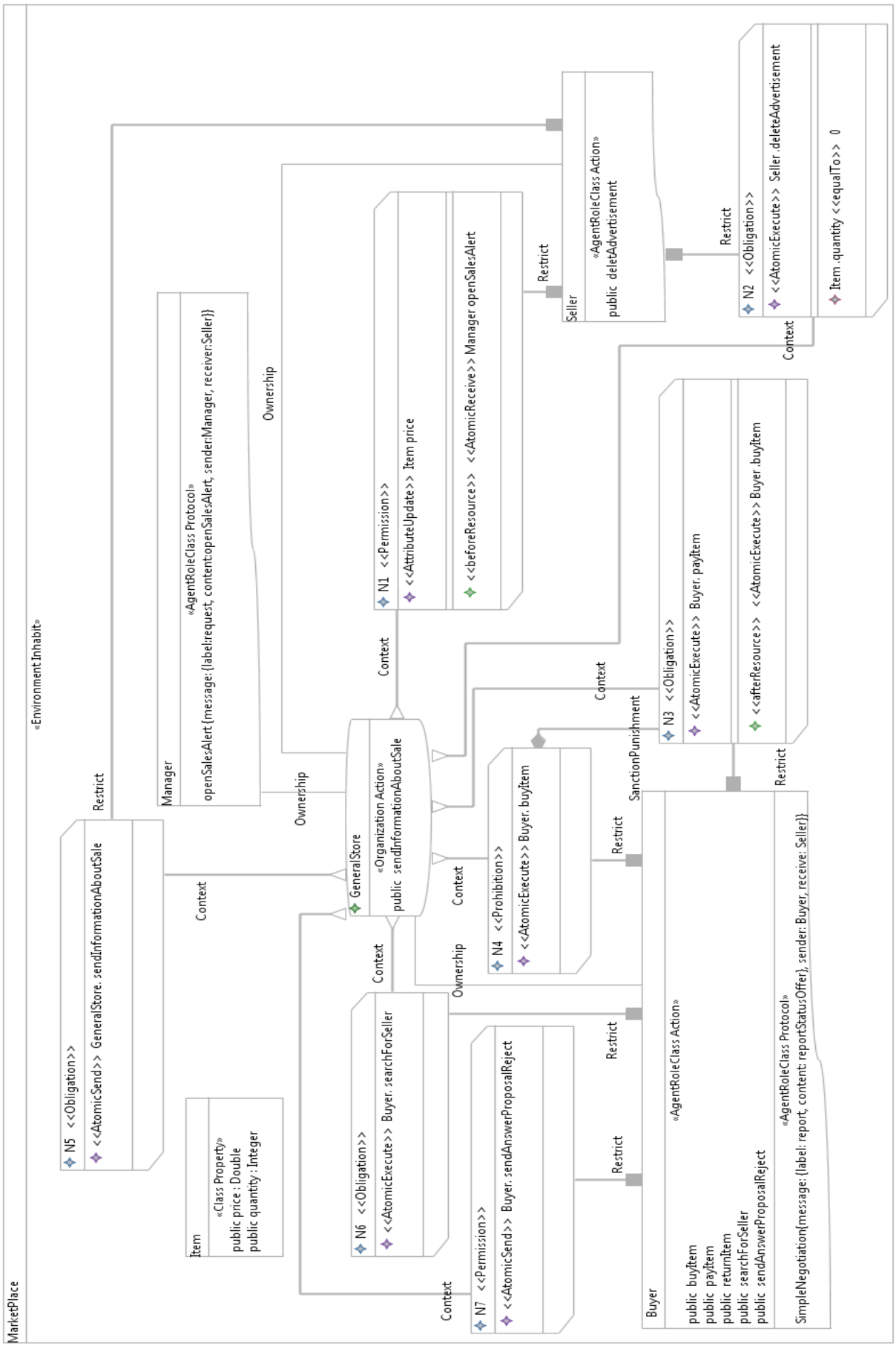


Figura 37 – Modelagem das normas N1, N2, N3, N4, N5, N6 e N7 apresentada por meio do diagrama de normas

6.2 Estudo de Caso 2: Gestão de Submissão de Artigos (*Conference Management*)

Este estudo de caso foi incluído com o objetivo de comparar a sintaxe concreta de NormML com a de NorMAS-ML apresentada neste trabalho. Isso é possível, pois este mesmo estudo de caso foi modelado por FIGUEIREDO (2011). Adicionalmente, este estudo de caso tem sido utilizado por vários autores, como ZAMBONELLI et al. (2001), DIGNUM (2004) e HARMON et al. (2008), para ilustrar suas abordagens.

Os sistemas de gestão de submissão de artigos são utilizados para a escolha dos artigos que serão publicados em um evento científico. Para isso, os autores devem submeter seus artigos até uma data determinada (*deadline*). Após essa data, os avaliadores devem iniciar o processo de revisão. Neste processo, cada artigo é avaliado por pelo menos três avaliadores que devem julgar os trabalhos considerando os seguintes aspectos: originalidade, estrutura do artigo, contribuição, entre outros. Após o término do período de revisão, os organizadores do congresso devem publicar os resultados para os autores e os que tiveram seu artigo aceito devem se registrar no congresso.

Os autores podem: (i) submeter seus artigos até a data final de submissão, (ii) visualizar a situação do seu artigo, (iii) visualizar os comentários feitos pelos avaliadores após o término do processo de revisão. Os revisores podem: (i) submeter artigos para o congresso, (ii) avaliar os artigos que foram indicados pelos organizadores do congresso. Os organizadores podem (i) estender o período de submissão de artigos, (ii) escolher os avaliadores que irão corrigir os artigos, (iii) divulgar o resultado das avaliações.

Os autores podem submeter dois tipos de artigos: (i) artigos completos, que representam trabalhos que já apresentam uma contribuição maior para a área de pesquisa e (ii) artigos curtos, que representam trabalhos que ainda não estão completamente terminados, mas que já possuem resultados.

6.2.1 Identificação das Entidades do Sistema de Gestão de Submissão de Artigos

No ambiente *Conference Management* é possível identificar a organização principal *Conference* e o tipo de agente: *user agent,* que pode exercer os papéis *author*, *speaker, organizer*, *conference chair*, *website manager* e *reviewer*. Estes papéis foram definidos pela organização principal, juntamente com o papel de objeto *submitted*. As instâncias desse papel são exercidas pelas instâncias da classe *Paper*, que possui duas subclasses *ShortPaper* e *FullPaper*. A Figura 38 apresenta o diagrama de classes, especificando os relacionamentos entre classes e ambientes.

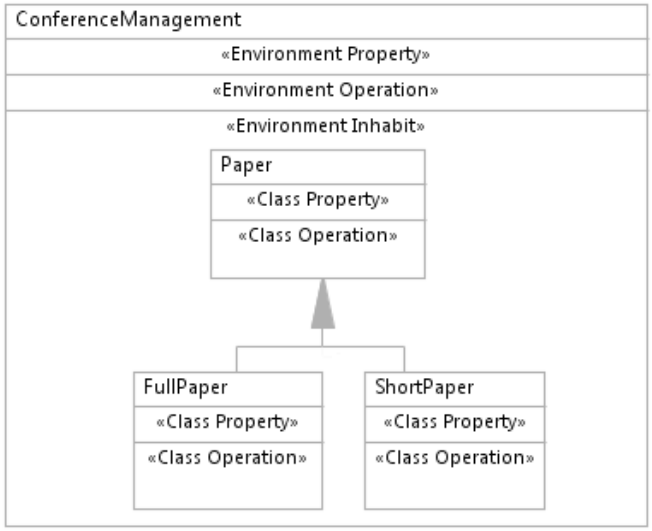


Figura 38 – Diagrama de Classes para o Sistema de Gestão de Submissão de Artigos

A Figura 39 ilustra o diagrama de organização referente à organização principal *Conference*. Para isso, foram definidas as classes da organização principal juntamente com as classes de agente, objeto, papel de agente e papel de objeto.

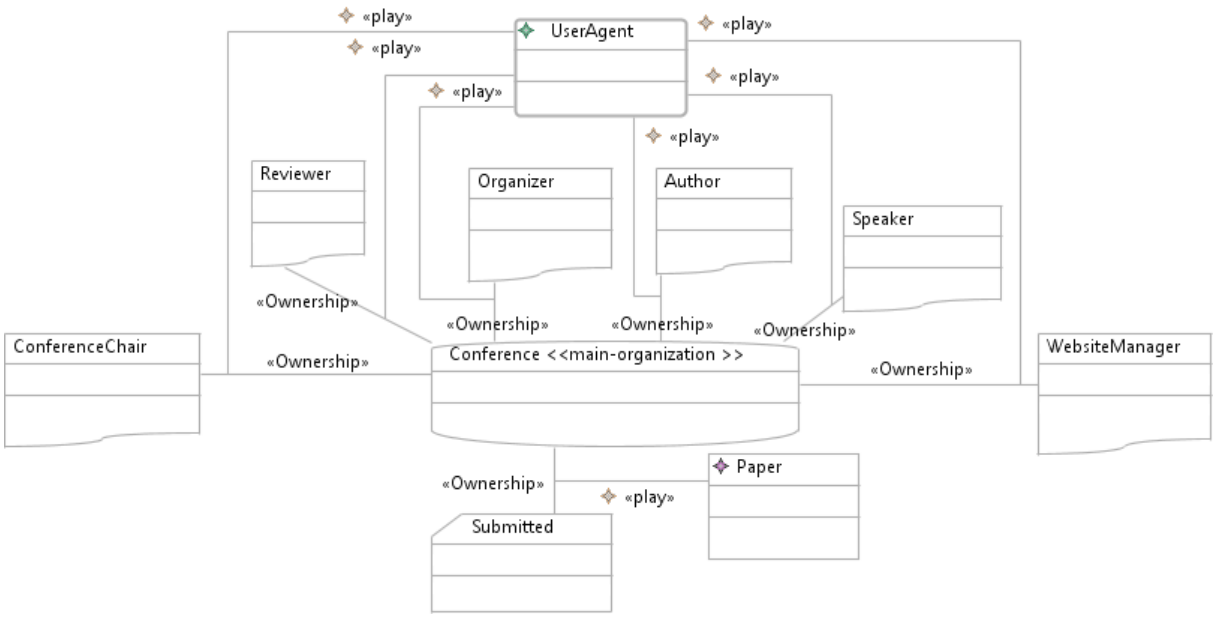


Figura 39 – O Diagrama de Organização representando a organização principal *Conference*

Para que as normas possam restringir o comportamento das entidades associadas a uma determinada organização, a lista de ações restringidas na entidade organização é explicitamente definida. A Figura 40 apresenta o detalhe da nova representação da classe da organização *Conference* modificada após a eliminação do estereótipo <<*axiom*>>.

Figura 40 – A classe da organização *Conference* modelado com NorMAS-ML

A Figura 41 ilustra o diagrama de papéis identificando os papéis exercidos por agentes e objetos na organização *Conference*. Neste diagrama, os papéis de agente *conference chair*, *website manager* e *reviewer* são apresentados como especializações do papel de agente *organizer*.

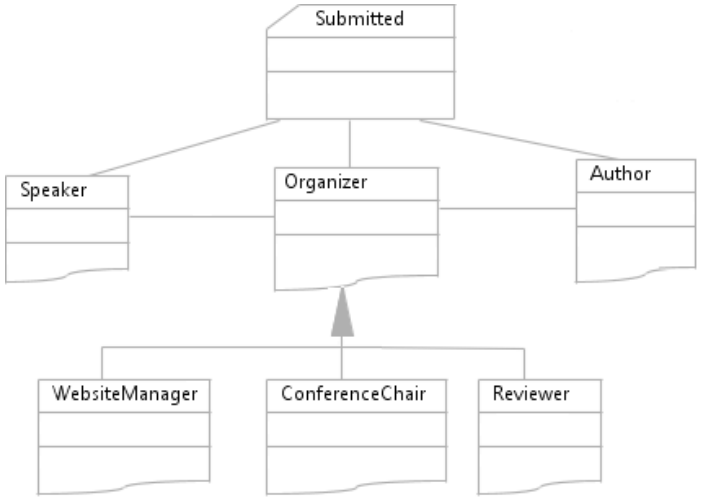


Figura 41 – O Diagrama de papéis referente à organização principal *Conference*

A Figura 42 apresenta o detalhe da representação da classe do papel *Reviewer* modificada após a eliminação dos estereótipos <<*duty*>> e <<*right*>>. Adicionalmente, para que as normas possam restringir o comportamento das entidades associadas a um determinado papel, a lista de ações restringidas na entidade papel é definida.

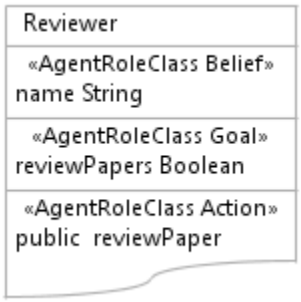


Figura 42 – A classe do papel *Reviewer* modelado com NorMAS-ML

Na próxima seção é ilustrada a modelagem das normas do sistema utilizando NorMAS-ML.

6.2.2 Definição das Normas para o Sistema de Gestão de Submissão de Artigos

Para o sistema de gestão de submissão de artigos definido em FIGUEIREDO (2011) foram estabelecidas 11 normas. Entretanto, somente um subconjunto das normas foi escolhido para este estudo de caso[4]:

- N1: Os organizadores estão proibidos de submeter artigos.
- N2: Os revisores estão permitidos de submeter artigos.
- N3: Os revisores estão proibidos de revisar seus próprios artigos.
- N4 (Punição para a violação da N3): Os revisores que violarem N4 devem ter seu papel cancelado.
- N5 (Punição para a violação da N3): O presidente da conferencia são obrigados a descartar o artigo.

6.2.3 Modelagem das Normas para o Sistema de Gestão de Submissão de Artigos

A seguir é apresentada cada norma juntamente com a sua descrição e identificação das características que a compõe.

A norma N1 define que os organizadores (entidade envolvida) da organização *Conference* (contexto) estão proibidos (conceito deôntico) de submeter artigos (ação).

A norma N2 assegura que os revisores (entidade envolvida) da organização *Conference* (contexto) estão permitidos (conceito deôntico) de submeter artigos (ação).

A Figura 43 apresenta a modelagem das normas N1 e N2 por meio do diagrama de normas.

A norma N3 determina que os revisores (entidades envolvidas) da organização *Conference* (contexto) estão proibidos (conceito deôntico) de revisar (ação) seus próprios artigos (restrição de ativação). A norma N3 aplica duas punições (sanção) que são as normas N4 e N5. Se algum revisor violar N3, N4 obriga (conceito deôntico) que o dado revisor (entidade envolvida) da organização *Conference* (contexto) tenha o seu papel cancelado (ação) e N5 estabelece que o presidente da conferência (entidade envolvido) da organização *Conference* (contexto) é obrigado (conceito deôntico) a descartar o artigo (ação).

[4] A modelagem de todas as normas definidas por FIGUEIREDO (2010) utilizando a extensão da MAS-ML *tool* está disponível em: https://sites.google.com/site/uecegessi/masmltool/estudodecasoutilizandoodiagramade normas.

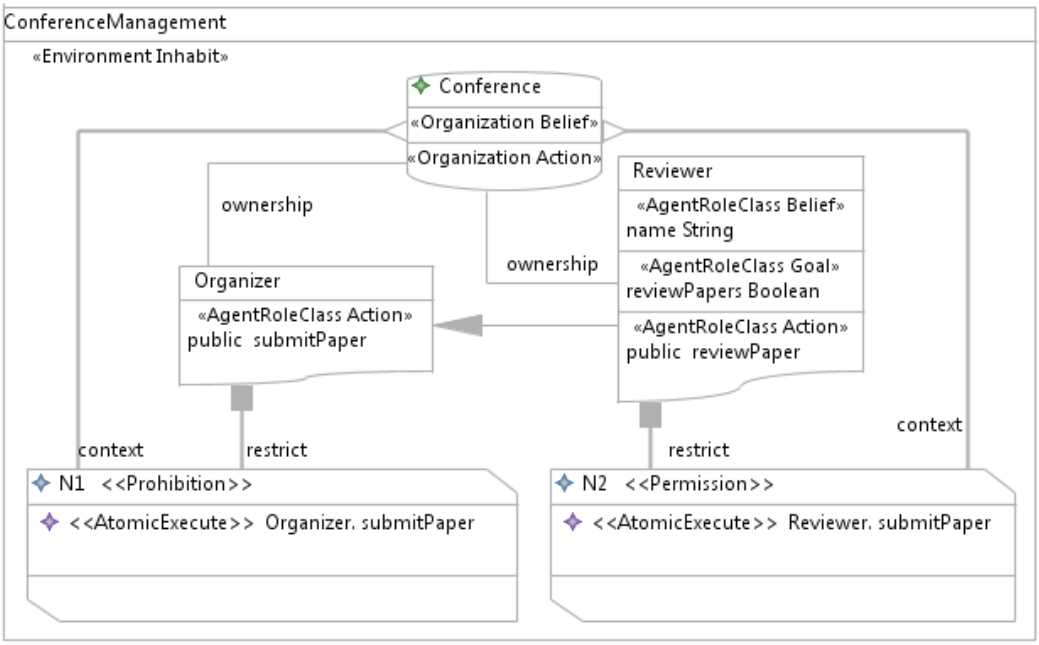


Figura 43 – Modelagem das normas N1e N2 apresentada por meio do diagrama de normas

A Figura 44 apresenta a modelagem das normas N3, N4 e N5 por meio do diagrama de normas.

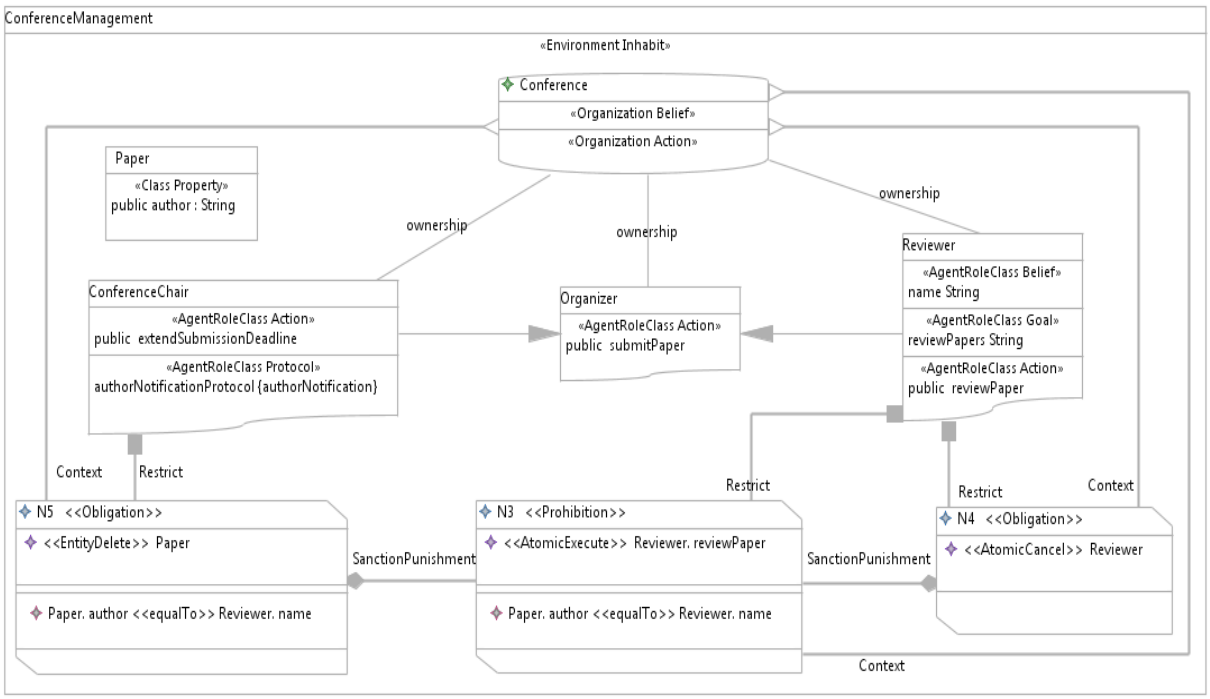


Figura 44 – Modelagem das normas N3, N4 e N5 apresentada por meio do diagrama de normas

Este estudo de caso tem como objetivo demonstrar a adequação de NorMAS-ML para a modelagem de SMA normativos. De acordo com o que foi ilustrado em ambos os estudos de caso, a capacidade de modelagem das entidades em SMA em relação a MAS-ML (SILVA, 2004) não foi alterada, mas estendida de forma a possibilitar a modelagem de

normas através de um novo diagrama que mostrou-se capaz de modelar todos os elementos normativos que já eram suportados em NormML (FIGUEIREDO, 2011) em conjunto com as entidades por elas governadas.

7 CONSIDERAÇÕES FINAIS

Neste capítulo serão feitas as considerações finais deste trabalho, apresentando as principais contribuições e limitações da pesquisa realizada, bem como delineando oportunidades para trabalhos futuros.

7.1 Contribuições e Limitações da Pesquisa

Nesta dissertação foram apresentados os conceitos relacionados com os elementos estáticos que compõem uma norma juntamente com a representação desses elementos na linguagem NormML com o intuito de analisar suas características, propriedades, atributos e comportamento, de forma a especificar os elementos de projeto necessários para estender a linguagem MAS-ML para permitir a modelagem de sistemas multi-agente normativos.

Após a análise de metodologias, linguagens de modelagens e modelos organizacionais que davam suporte parcial a modelagem dos elementos de uma norma juntamente com os elementos que compõem os SMAs, foi constatado que a linguagem MAS-ML era a mais adequada para receber as metaclasses e relacionamentos definidos em NormML por apresentar as seguintes características: (i) possuir uma ontologia adequada para a modelagem das entidades em SMA, (ii) identificar papéis, (iii) modelar adequadamente ambientes e a interação entre agente e ambiente, (iv) apresentar suporte parcial à modelagem dos elementos que compõem normas para SMAs, e (v) possuir ferramenta de suporte.

Com base nos elementos estáticos que compõem as normas, foi apresentada a extensão do *framework* TAO para possibilitar que as entidades definidas nesse *framework* pudessem ter seu comportamento restringido por normas. Para isso, foi necessária a criação do novo elemento *Norm* juntamente com a definição de suas propriedades e a criação de quatro relacionamentos: *Context, Restrict*, *SanctionReward e SanctionPunishment*. Por meio deles, é possível associar a entidade *Norm* às entidades previamente definidas no TAO.

Após a extensão do TAO, foi realizada a extensão da linguagem MAS-ML nas sintaxes abstrata e concreta juntamente com a criação de um novo diagrama estático para a modelagem das normas.

A extensão da sintaxe abstrata de MAS-ML envolveu a incorporação de 13 metaclasses originárias de NormML: (i) *Norm*, (ii) *Resource*, (iii) *NormConstraint*, (iv) *After*, (v) *Between*, (vi) *Before*, (vii) *If*, (viii) *Date*, (ix) *Operator*, (x) *NormAction*, (xi) *AtomicAction*, (xii) *CompositeAction* e (xiii) *Sanction*. Adicionalmente, foram definidas as

metaclasses *Context* e *Restrict* juntamente com os novos estereótipos para as metaclasses *Norm*, *Sanction*, *AtomicAction* e *CompositeAction*. Devido à extensão, os estereótipos <<*duty*>> e <<*right*>> da metaclasse *AgentAction* e <<*axiom*>> da metaclasse *Property* originalmente definidas em MAS-ML foram eliminados. Finalmente, a extensão foi realizada na sintaxe concreta por meio da definição dos elementos gráficos da entidade *Norm* e dos relacionamentos *Context*, *Restrict* e *Sanction*. Adicionalmente, os elementos gráficos das entidades *Organization* e *AgentRole* foram modificados em consistência com a sintaxe abstrata atualizada. A nova versão de MAS-ML passou a ser denominada NorMAS-ML.

A partir dos novos elementos propostos em NorMAS-ML, um novo diagrama estático para a representação de normas foi definido através do qual, todas as normas aplicadas a um SMA podem ser modeladas. Desta forma, a modelagem das entidades e das normas que as governam pode ser realizada através de uma única linguagem de modelagem, NorMAS-ML, reduzindo os esforços concentrados no aprendizado de outras linguagens e ambientes de suporte, assim como também garantindo a coerência e compatibilidade entre os modelos gerados.

A ferramenta MAS-ML *tool,* originalmente desenvolvida para dar suporte à geração de diagramas em MAS-ML, foi evoluída de forma a dar suporte à modelagem de SMA conforme as extensões e adequações propostas em NorMAS-ML. Para isso, foi necessária a criação de nove metaclasses e quatro semânticas para representar as metaclasses definidas na sintaxe abstrata de NorMAS-ML juntamente com a definição dos novos elementos gráficos para as novas entidades e relacionamentos, utilizados para compor o novo diagrama estático aqui proposto para a representação das normas do sistema. Com isso, novas restrições OCL para a checagem da corretude na construção do novo diagrama precisaram ser definidas. Vale ressaltar que as contribuições em relação ao suporte à modelagem de normas na ferramenta são valiosas no contexto de um projeto maior em andamento relativo à geração de código utilizando a abordagem orientada a modelos.

Por meio da utilização da ferramenta MAS-ML *tool* estendida com as novas funcionalidades foi possível a modelagem de dois estudos de casos considerando SMA com normas. O primeiro relacionado com mercados virtuais e o segundo com um sistema de gestão de artigos. Por meio da modelagem desses estudos de caso foi possível verificar que a partir da extensão proposta, NorMAS-ML é capaz de modelar todas as entidades típicas dos SMAs em associação com os elementos estáticos que compõem as normas de forma adequada. Consequentemente, NorMAS-ML, através das diversas visões fornecidas pelos

diversos diagramas estáticos e dinâmicos que possui, consegue dar um suporte maior para a modelagem de SMAs.

Como limitações do presente trabalho, podemos citar que NorMAS-ML não fornece suporte à verificação de conflitos entre normas.

7.2 Trabalhos Futuros

Existem alguns trabalhos futuros que poderão ser desenvolvidos com o intuito de dar continuidade ao trabalho apresentado nesta dissertação. Dentre eles podem ser citados:

(i) A inclusão da verificação de conflitos entre normas na ferramenta MAS-ML *tool*. Para essa tarefa, poderia ser utilizada a abordagem apresentada em FIGUEIREDO (2011) baseada na implementação de regras OCL;

(ii) Análise da adequação de NorMAS-ML para a modelagem das diversas arquiteturas de agente previstas em MAS-ML 2.0. Um trabalho preliminar neste contexto foi abordado por FREIRE, CAMPOS e CORTÉS (2012);

(iii) A formalização da extensão em NorMAS-ML. Para essa tarefa, uma abordagem baseada em ontologia semelhante à utilizada por BRANDÃO, SILVA e LUCENA (2007) poderia ser utilizada;

(iv) A extensão do TAO e de MAS-ML para contemplar os elementos dinâmicos das normas, tais como: criação, cancelamento e delegação (SILVA; BRAGA; FIGUEIREDO, 2010) e sua incorporação nos diagramas de sequência e de atividade para possibilitar a representação dos elementos dinâmicos das normas por meio da ferramenta MAS-ML *tool*;

(v) A geração de código a partir dos diagramas gerados pela ferramenta MAS-ML *tool*. Para esta tarefa, uma abordagem baseada em MDA (*Model Driven Architecture*) semelhante à proposta por LOPES (2012) poderia ser utilizada.

7.3 Conclusão

O benefício conseguido por meio da extensão representa uma contribuição tanto na área de Inteligência Artificial (IA) com de Engenharia de Software (ES). Do ponto de vista de IA, a possibilidade de modelar sistemas multi-agente normativos por meio da linguagem NorMAS-ML facilita o entendimento e a posterior implementação de tais sistemas.

Do ponto de vista de ES, o presente trabalho contribui especificamente para a modelagem de sistemas complexos, no contexto do paradigma orientado a agentes. Neste contexto, o presente trabalho apresenta a extensão tanto de um *framework* conceitual como de uma linguagem de modelagem, permitindo que o projetista possa gerar modelos mais adequados à arquitetura da solução, que serão aproveitados na fase de implementação de SMAs.

Ainda no contexto de ES, o presente trabalho apresenta a extensão de uma ferramenta de modelagem existente para possibilitar a modelagem dos novos conceitos propostos por meio do diagrama de normas. Com isso, estamos possibilitando que o projetista possa automatizar a atividade de modelagem e garantir que seus modelos sejam coerentes com todas as definições propostas na linguagem, diminuindo assim, a ocorrência de erros humanos que possam vir a se propagar na implementação, aumentando a produtividade e reduzindo o retrabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

BASIN, D.; DOSER, J.; LODDERSTEDT, T. (2006). Model driven security: from UML models to access control infrastructures, ACM TSEM, 15(1), pp.39-91.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. (2007). Developing Multi-Agent Systems with JADE. [S.l.]: Wiley (Wiley Series in Agent Technology).

BEZERRA, E. (2007). Princípios de Análise e projeto de sistemas com UML, 2ª Ed., Rio de Janeiro: Elsevier. p. 111.

BORDINI, R. H.; WOOLDRIDGE, M.; HÜBNER, J. F. (2007). Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology), John Wiley & Sons.

BRANDÃO, A. A. F.; SILVA, V. T.; LUCENA, C. J. P. (2007). Observed-MAS: an Ontology-based Method for Analyzing Multi-Agent Systems Design Models. In: Lin Padgham; Franco Zambonelli. (Org.). Agent-Oriented Software Engineering. Berlin: Springer, v. 4405, p. 122-139.

BRADSHAW, J. M. (1997). Software Agents. MIT Press Cambridge, MA, USA.

CASILLO, B. H. (2008). Agentes Auxiliando Ambientes de Engenharia de Software Centrado em Processos. Dissertação de Mestrado. São José dos Campos: INPE.

DANC, J. (2008). Formal Specification of AML. Department of Computer Science Faculty of Mathematics, Physics and Informatics Comenius University Formal Specification of AML Master's Thesis Ján Danc Advisor: Mgr. Bratislava.

DE MARIA, B. A.; SILVA, V. T.; LUCENA, C.J.P.; CHOREN, R. (2005). VisualAgent: A Software Development Environment for Multi-Agent Systems. Proceedings of the 19º Simpósio Brasileiro de Engenharia de Software (SBES 2005), Tool Track, Uberlândia, MG, Brazil, October 3-7, 2005.

DENNIS, L.; TINNEMEIER, N.A.M.; MEYER, J-J.CH. (2010). Model Checking Normative Agent Organisations. In J. DIx, M. Fischer & P. Novák (Eds.), Computational Logic in Multi-Agent Systems - 10th International Workshop, CLIMA X, Hamburg, Germany, 2009 Vol. 6214.

DIGNUM, V. (2004). A model for organizational interaction: based on agents, founded in logic. PhD dissertation, Universiteit Utrecht, SIKS dissertation series 2004-1.

ECLIPSE PLATFORM (2011), http://www.eclipse.org/, Acessado em 26 de Novembro de 2011.

EMF (2011), www.eclipse.org/modeling/emf/, Acessado em 26 de Novembro de 2011.

FARIAS, K.; OLIVEIRA, K.; NUNES, I.; SILVA, V. T.; LUCENA, C.J.P. (2009). MAS-ML *Tool*: Um Ambiente de Modelagem de Sistemas Multi-Agentes, V Workshop on Software Engineering for Agent-oriented Systems (SEAS 2009), Fortaleza, Brasil, pp. 1-12.

FEIJÓ, A. R. (2012). Evolução da Ferramenta MAS-ML *tool* para a Modelagem dos Diagramas de Papéis e Sequência. Monografia. Fortaleza: UECE, Departamento de Computação.

FIGUEIREDO, K.; SILVA, V. T. (2010). NormML: A Modeling Language to Model Norms. In: 1st Workshop on Autonomous Software Systems. Salvador, Brazil.

FIGUEIREDO, K. (2011). Modeling and Validation Norms in Multi-Agents Systems. Dissertação de Mestrado. Niterói: UFF, Instituto de Computação.

FREIRE, E. S. S. ; GONÇALVES, E. J. T. ; CORTÉS, M. I. ; SILVA, V. T. ; LOPES, Y. S. (2011). Extensão da Linguagem MAS-ML para a Modelagem de Sistemas Multi-Agente Normativos. In: II Workshop on Autonomous Sotware Systems, 2011, São Paulo.

FREIRE, E. S. S.; GONÇALVES, E. J. T.; CORTÉS, M. I.; LOPES, Y. S. (2012a). Modelando Sistemas Multi-Agente Normativos com a Linguagem MAS-ML. In: VIII Simpósio Brasileiro de Sistemas de Informações, 2012, São Paulo.

FREIRE, E. S. S.; GONÇALVES, E. J. T.; CORTÉS, M. I.; LOPES, Y. S. (2012b). Extensão da Sintaxe Abstrata da Linguagem MAS-ML para a Modelagem de Sistemas Multi-Agente Normativos. In: Computer on the Beach, 2012, Florianópolis.

FREIRE, E. S. S.; CAMPOS, G. A. L.; CORTÉS, M. I. (2012). Norm-based behaviour modification in Reflex Agents Architecture. In: Special Session on Intelligent Multi-Agent Systems at International Conference on Agents and Artificial Intelligence (IMAS@ICAART 2012). No prelo.

GARCÍA-CAMINO, A.; RODRÍGUEZ-AGUILAR, J.; SIERRA, C.; VASCONCELOS, W. (2006). Norm-Oriented Programming of Electronic Institutions. In Proc. 5th AAMAS, ACM Press, pp. 670-672.

GMF (2011). www.eclipse.org/modeling/gmf/, Acessado em 26 de Novembro de 2011.

GONÇALVES, E. J. T. (2009). Modelagem de Arquiteturas Internas de Agentes de Software Utilizando A Linguagem MAS-ML 2.0. Dissertação de Mestrado em Ciência da Computação da UECE, Centro de Ciência e Tecnologia. Fortaleza

GONÇALVES, E. J. T.; FARIAS, K.; CORTÉS, M. I.; FEIJÓ, A. R.; OLIVEIRA, F. R.; SILVA, V. T. (2011). MAS-ML TOOL - A Modeling Environment for Multi-agent Systems, In: The International Conference on Enterprise Information Systems (ICEIS 2011), Beijing, China.

HANNOUN, M. (2002). MOISE: Un Modèle Organisationnel Pour Les Systèmes Multi-Agents. Tese (Thèse(Doctorat)) – École Nacionale Supérieure des Mines de Saint-Etienne.

HARMON, S. J.; DELOACH, S. A. (2008). Trace-based Specification of Law in Guidance Policies for Multiagent Systems. Engineering Societies in the Agents World VIII, Springer-Verlag Berlin, Heidelberg.

HOWDEN, N.; RÖNNQUIST, R.; HODGSON, A.; LUCAS, A. (2001). JACK Intelligent Agents – Summary of an Agent Infrastructure. Em: Second International Workshop on Infrastructure for Agents, MAS, and scalable MAS, 5th International Conference on Autonomous Agents, Canadá.

HÜBNER, J. F.; SICHMAN, J. S.; OLIVIER, B. (2002). A model for the structural, functional and deontic specification of organizations in multiagent systems. In: SBIA '02 Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence, Springer-Verlag London, UK.

LOPES, Y. S. (2012). Desenvolvimento Orientado a Modelos em Sistemas Multi-Agentes com Diferentes Arquiteturas Internas de Agente. Dissertação de Mestrado. Fortaleza: UECE, Centro de Ciência e Tecnologia.

LÓPEZ Y LÓPEZ, F. (2003). Social Power and Norms: Impact on agent behavior. PhD. thesis, Univ. of Southampton, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science.

MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T. (2003). Introduction: Model-Driven Development, IEEE Software, vol. 20, no. 5, pp.14-18, Sep/Oct, 2003, Guest Editors'.

MEYER, J. J.; WIERINGA, R. J. (1993). Deontic logic in computer science: normative system specification. Deontic logic in computer science: normative system specification, John Wiley and Sons Ltd. Chichester, UK.

OCL (2011), disponível em:<http://www.eclipse.org/modeling/mdt/?project=ocl>, acessado em 20 de Junho de 2011.

ODELL, J.; PARUNAK, H. V. D.; BAUER, B. (2000). Extending UML for Agents. Proc. of the Agent-Oriented. Information Systems Workshop (AOIS‟00) at the 17th National conference on Artificial Intelligence (AIII‟00) (3-17).

OMICINI, A. (2001). SODA: Societies and infrastructures in the analysis and design of agent-based systems. In: First international workshop, AOSE 2000 on Agent-oriented software engineering, Springer-Verlag New York, Inc. Secaucus, NJ, USA.

PADGHAM, L.; WINIKOFF, M. (2004). Developing intelligent agent systems: a practical guide. John Wiley and Sons, 225 pages.

PADILHA, T. P. P.; JÁCOME, T. F. (2002). O Uso de Técnicas de Modelagem de Agentes em Ambientes Educacionais. In: VI Congresso Iberoamericano de Informática Educativa.

POKAHR, A.; BRAUBACH, L.; LAMERSDORF, W. (2003). Jadex: Implementing a BDI-Infrastructure for JADE Agents. EXP - In Search of Innovation (Special Issue on JADE), vol. 3, no. 3 , Telecom Italia Lab, Turin, Italy, S. 76-85.

PRESSMAN, R. S. (2009). Engenharia de Software. 6 Ed.São Paulo: Mcgraw-Hill Brasil Tecnicos.

RUSSELL, S.; NORVIG, P. (2003). Artificial Intelligence: A Modern Approach, 2nd Ed., Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-790395-2.

SILVA, V. T. (2004). Uma linguagem de modelagem para sistemas multi-agentes baseada em um framework conceitual para agentes e objetos, Tese de doutorado. Rio de Janeiro: PUC, Departamento de Informática.

SILVA, V.; GARCIA, A.; BRANDAO, A.; CHAVEZ, C.; LUCENA, C.; ALENCAR, P. (2003). Taming Agents and Objects in Software Engineering. In: Garcia, A.; Lucena, C.; Zamboneli, F.; Omicini, A; Castro, J. (Eds.), Software Engineering for Large-Scale Multi-Agent Systems, Springer-Verlag, LNCS 2603, pp. 1-26, 2003, ISBN 978-3-540-08772-4.

SILVA, V. T.; CHOREN, R.; LUCENA, C. J. P. de (2005). Using UML 2.0 Activity Diagram to Model Agent Plans and Actions. The International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2005), 4th. Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, Netherlands, Holanda, pp. 594-600, v. 2, n.1, ACM, ISBN: 1-59593-094-9.

SILVA, V. T.; CHOREN, R.; LUCENA, C. J. P. de (2007). MAS-ML: A Multi-Agent System Modeling Language. Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA); In: Companion of the 18th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications; Anaheim, CA, USA, ACM Press, pp. 304-305.

SILVA, V. T., CHOREN R.; LUCENA, C. (2008). MAS-ML: a multi-agent system modelling language, In IJAOSE, Modeling Lang. for Agent Systems,(2)4pp.382-421.

SILVA, V.; BRAGA, C. ; FIGUEIREDO, K. (2010). A Modeling Language to Model Norms. In: Workshop on Coordination, Organization, Institutions and Norms in agent systems at International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS10), Toronto, p. 25-32.

SOMMERVILLE, I. (2007). Engenharia de Software. 8 ed. São Paulo: Pearson Addison-Wesley.

UML (2011): Unified Modeling Language Specification, versão 2.2, OMG, (2011) Disponível em:<http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML>. Acessado em: 29 de novembro 2011.

VASCONCELOS, W.; KOLLINGBAUM, M.; NORMAN, T. (2007). Resolving Conflict and Inconsistency in Norm-Regulated Virtual Organizations. In Proc. AAMAS'07.

VECHT, B. VAN DER; DIGNUM, F.P.M.; MEYER, J-J.CH.; DIGNUM, M.V. (2009). Autonomous Agents Adopting Organizational Rules. In M.V. Dignum (Ed.), Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models (pp. 314-333). IGI Global.

VRIES, W. de; MEYER, J-J.CH.; BOER, F.S. de; HOEK, W. VAN DER (2009). A Coordination Language For Agents Interacting In Distributed Plan--Execute Cycles. In Int. J. Reasoning-Based Intelligent Systems Vol. 1. International Journal Reasoning-Based Intelligent Systems (pp. 4-17).

WAGNER, G. (2003). The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. Information Systems. 28(5), pp. 475–504.

WEISS, G. (1999). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Massachusetts.

ZAMBONELLI, F.; JENNINGS, N. R.; WOOLDRIDGE, M. J. (2001). Organisational Rules as an Abstraction for the Analysis and Design of Multi-Agent Systems. In: International Journal of Software Engineering and Knowledge Engineering, Volume 11, Number 3, p. 303-328.