Evolução da Ferramenta MAS-ML *tool* para a Modelagem do Diagrama de Papéis

Állan R. Feijó [1], Felipe J. A. Maia[1], Francisco R. O. de Lima[1], Igor B. Nogueira[1], Enyo J. T. Gonçalves[2], Emmanuel S. S. Freire[1], Mariela I. Cortés[1], Leandro L. C. de Souza[1]

Grupo de Engenharia de Software e Sistemas Inteligentes (GESSI)

[1]Universidade Estadual do Ceará, Campus Itapery, 60.740-903 – Fortaleza – CE – Brasil

[2]Universidade Federal do Ceará, Campus Quixadá, 63.900-000 – Quixadá – CE – Brasil

{allanfeijo1987, felipe.ja.maia, us.robson7, igor.bnog, savio.essf}@gmail.com, enyo@ufc.br, mariela@larces.uece.br

Abstract. *The modeling activity can be highly complex and tending to errors, particularly, in the modeling of Multi-Agent Systems (MASs). Thus, the existence of an adequated support tool can be crucial for choosing and adopting the modeling language to be used. In this context, the MAS-ML tool, based on the MAS-ML modeling language, includes mechanisms to model only two proposed diagrams by MAS-ML. The goal of this paper is present the evolution of the MAS-ML tool aiming to support the role diagram defined in the modeling language follows the model-based approach.*

Resumo. *A atividade de modelagem pode ser altamente complexa e propensa a erros, em particular no caso da modelagem de Sistemas Multi Agente (SMA). Assim sendo, a existência de uma ferramenta de suporte adequada pode ser crucial na escolha e adoção da linguagem de modelagem a ser utilizada. Neste contexto, a ferramenta MAS-ML tool, baseada na linguagem de modelagem MAS-ML, incorpora mecanismos para a modelagem de somente dois dos diagramas propostos por MAS-ML. O objetivo deste artigo é apresentar a evolução da ferramenta MAS-ML tool de forma a dar apoio à modelagem do diagrama de papéis prevista na linguagem de modelagem seguindo a abordagem por modelos.*

1. Introdução

A crescente busca por sistemas mais eficientes leva indefectivelmente, ao desenvolvimento de sistemas cada vez mais complexos. Neste cenário, o paradigma orientado a agentes vem sendo cada vez mais utilizado para lidar com essa complexidade, tanto na indústria quanto na academia.

Um agente pode ser definido como uma entidade autônoma capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores [Jennings, 1996]. O termo Sistema Multi Agente (SMA) refere-se à subárea de Inteligência Artificial que investiga o comportamento de um conjunto de agentes autônomos, objetivando a solução de um problema que está além da capacidade de um único agente [Russel e Norvig, 2004]. Neste cenário, a engenharia de software orientada a agentes (AOSE) [Lind, 2001] surge no intuito de fornecer métodos e ferramentas

adequados para o desenvolvimento de aplicações neste novo domínio, no qual podemos destacar as linguagens de modelagem apoiadas por ferramentas de suporte adequadas e que possuem papel central no desenvolvimento de SMA. No contexto das linguagens de modelagem para SMA destacamos a MAS-ML (*Multi-Agent System Modeling Language*) [Silva, 2004].

O objetivo da MAS-ML é modelar todos os aspectos dinâmicos e estruturais definidos no *framework* TAO (*Taming Agents and Objects*) [Silva, 2004]. O metamodelo de MAS-ML é uma extensão do metamodelo da UML (*Unified Modeling Language*) [OMG, 2011] de acordo com os conceitos definidos no TAO. A linguagem MAS-ML prevê um conjunto de diagramas estruturais, a saber: diagrama de classe, diagrama de papéis e diagrama de organização. Adicionalmente, define também os diagramas dinâmicos de sequência e atividades [Silva, 2007].

A ferramenta MAS-ML *tool* [Farias, 2009] é um ambiente de modelagem (editor gráfico) para SMAs desenvolvido como um *plug-in* da plataforma Eclipse [Eclipse, 2011]. Isto implica que os usuários podem, ao mesmo tempo, modelar SMAs e utilizar os recursos disponíveis dentro da plataforma Eclipse. Em [Gonçalves, 2010] a ferramenta foi estendida para adequar o diagrama de classes à evolução definida em MAS-ML 2.0. Adicionalmente, o diagrama de organização foi implementado. Entretanto, na sua versão atual, o diagrama de papéis, previsto na linguagem de modelagem MAS-ML, não é contemplado na ferramenta.

O diagrama de papéis é importante no contexto de modelagem de SMAs por ser capaz de representar os relacionamentos entre os papéis de agente, os quais não são visíveis nos diagrama de classe e organização. A entidade papel ocupa um lugar de destaque pelo fato dela só poder existir no contexto de uma organização, e suas instâncias devem ser exercidas por agentes, objetos ou mesmo suborganizações [Silva, 2004]. A função do papel é orientar ou restringir o comportamento da entidade que está a exercê-lo.

O objetivo desse artigo é apresentar a evolução da ferramenta MAS-ML *tool* de forma a dar suporte à modelagem do diagrama de papéis de acordo com a especificação apresentada em MAS-ML 2.0. A nova versão da ferramenta é ilustrada a partir de um estudo de caso no qual os papéis utilizados no ambiente de aprendizagem virtual *Moodle* (*Modular Object-Oriented Dynamic Learning Environment*) [MOODLE, 2011] foram modelados.

Este artigo está estruturado da seguinte maneira: Na Seção 2 são apresentados os principais conceitos da linguagem de modelagem MAS-ML 2.0 e da ferramenta MAS-ML *tool*. Na Seção 3, a implementação do diagrama de papéis na ferramenta MAS-ML *tool* é descrita. Em seguida, o estudo de caso utilizando a ferramenta é ilustrado na Seção 4. Alguns trabalhos relacionados são apresentados na Seção 5. Finalmente, as conclusões e trabalhos futuros são descritos na Seção 6.

2. Referencial Teórico

Nesta seção é apresentado o referencial teórico em relação à linguagem de modelagem MAS-ML em sua versão 2.0, e a ferramenta de suporte à modelagem, MAS-ML *tool*.

2.1. MAS-ML 2.0 e o Diagrama de Papéis

MAS-ML é uma linguagem de modelagem que estende a UML e que incorpora o conceito de agente definido no *framework* conceitual TAO para a modelagem SMAs. Originalmente, MAS-ML foi projetada para modelar apenas agentes pró-ativos orientados a objetivos e guiados por planos. Na sua versão mais atual, a linguagem MAS-ML contempla o suporte à modelagem das diversas arquiteturas internas, a saber: arquiteturas internas de (i) agentes reativos simples; (ii) agentes reativos baseados em conhecimento; (iii) agentes baseados em objetivo com planejamento e (iv) agentes baseados em utilidade. Em MAS-ML 2.0 foram incluídos os conceitos de arquiteturas internas de agente na linguagem de maneira conservativa em relação à MAS-ML, ou seja, mantendo-se a representação inicial prevista na linguagem.

A linguagem MAS-ML contempla um conjunto de diagramas estáticos, a saber: diagrama de classes, organização e papéis. O diagrama de papéis é responsável por expressar relacionamentos entre papéis de agente e papéis de objeto identificados no diagrama de organização. Esse diagrama também identifica as classes acessadas pelos papéis de agente e papéis de objeto. As interações entre agentes e organizações de um sistema são descritas com base nos papéis ilustrados pelo diagrama de papel [Silva, 2004].

Segundo [Silva, 2004], um diagrama de papel pode mostrar os relacionamentos entre a classe do papel do agente e a classe do papel de objeto e entre esses papéis e classes. O conjunto de relacionamentos usado nesse diagrama é: (i) *Control*, usado entre classes de papel de agente; (ii) *Dependency*, usado entre classes de papel de objeto, entre classes de papel de agente e classes de papel de objeto e entre classes de papel de agente; (iii) *Association*, usado entre classes de papel de objeto, entre classes de papel de agente e classes de papel de objeto, entre classes de papel de agente e entre classes e classes de papel; (iv) *Aggregation*, usado entre classes de papel de objeto e entre classes de papel de agente; (v) *Specialization/ Generalization*, usado entre classes de papel de objeto e entre classes de papel de agente.

2.2. MAS-ML *TOOL*

A ferramenta MAS-ML *tool* é um ambiente de modelagem (editor gráfico) que serve como um *plug-in* da plataforma Eclipse, possibilitando a utilização dos recursos disponíveis dentro da plataforma Eclipse durante a modelagem de SMAs. MAS-ML *tool* foi criada para dar suporte à modelagem dos diagramas contemplados na linguagem MAS-ML, e na sua versão atual, a ferramenta fornece apoio para a construção dos diagramas de classe e organização de acordo com MAS-ML 2.0.

MAS-ML *tool* foi gerada a partir do *plug-in* do GMF (*Graphical Modeling Framework*) [GMF, 2011] que é um *framework* para desenvolvimento de editores gráficos para modelos de domínio. Ele surgiu a partir da união de dois *frameworks* denominados GEF (*Graphical Editing Framework*) [GEF, 2011], utilizado para a criação de editores gráficos genéricos, e EMF (*Eclipse Modeling Framework*) [EMF, 2011], que permite ao desenvolvedor construir metamodelos e gerar código Java relativo ao mesmo.

A construção do GMF seguiu uma abordagem dirigida por modelos, onde o modelo central e de maior abstração é o próprio metamodelo da linguagem MAS-ML.

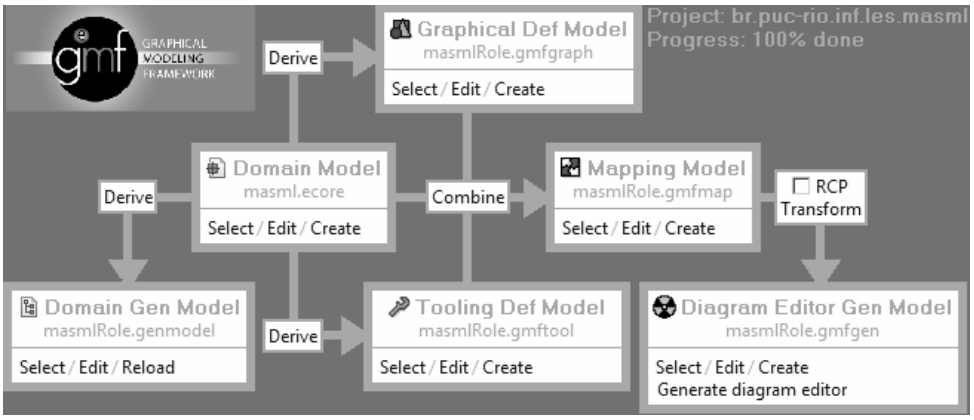

Fig. 1. GMF *Dashboard*.

As etapas (Figura 1) para a sua construção são as seguintes: (i) *domain model*, onde é definido o modelo de domínio; (ii) *domain gen model*, que estende o modelo de domínio através da geração de dados (Classes java); (iii) *graphical def model*, onde são definidos os controladores e as figuras; (iv) *tooling gef model*, onde é definido o menu, a paleta, os botões, etc.; (v) *mapping model*, onde é feito o mapeamento dos controladores e as figuras; (vi) *diagram editor gen model*, onde é feita a geração do projeto executável.

3. Implementação do Diagrama de Papéis

A estratégia adotada para implementar as extensões propostas segue a abordagem dirigida por modelos, utilizada originalmente para desenvolver a própria ferramenta. Neste caso é utilizado como modelo central o metamodelo da linguagem MAS-ML 2.0. Essa implementação seguiu as etapas descritas a seguir, de acordo com o GMF.

3.1. Extensão do *Domain Model*

O *Domain Model* contém o metamodelo "*masml.ecore*". Nesse metamodelo foram criados os relacionamentos entre a metaclasse "*MasmlClassDiagram*" e as metaclasses (i) *Class*, (ii) *ObjectRoleClass*, (iii) *AgentRoleClass, (iv) Association*, (v) *Dependency*, (vi) *Generalization*, (vii) *Aggregation*, (viii) *Control*. Note que as metaclasses citadas fazem parte do metamodelo do Diagrama de Papel.

Além disso, foi necessário adicionar dois novos atributos (*sourceMultiplicity* e *targetMultiplicity*) à metaclasse *Control* referentes à multiplicidade, para que a multiplicidade possa ser representada por este relacionamento.

A partir do "*masml.ecore*" estendido são gerados todos os outros metamodelos contidos no GMF.

3.2. Extensão do *Domain Gen Model*

O "*masml.ecore*" foi derivado (transformado) em um metamodelo mais específico, o "*masmlRole.genmodel*", a partir do qual as classes Java (códigos) são geradas. Os códigos gerados são: (i) *Model Code* (masml, masml.impl e masm.util); (ii) *Edit Code* (br.puc-rio.inf.les.masml.edit); (iii) *Editor Code* (br.puc-rio.inf.les.masml.editor); (iv)

Test Code (br.puc-rio.inf.les.masml.tests). Essas classes são usadas no projeto executável e são necessárias para a geração da ferramenta MAS-ML *tool*.

3.3. Extensão do *Graphical Def Model*

O "*masmlRole.gmfgraph*", derivado a partir do "*masml.ecore*", é responsável pela criação dos componentes gráficos. Durante o processo de derivação é selecionada a classe principal ou raiz (*MasmlClassDiagram*), em seguida são marcados os componentes requeridos no Diagrama de Papel, indicando o tipo de componente em termos de forma, relacionamento ou texto.

Após a derivação, foram criados os seguintes componentes: (i) *Nodes* (representam as entidades e os compartimentos pertencentes ao diagrama de papel), (ii) *Conections* (representam os relacionamentos pertencentes ao diagrama), (iii) *Figures* (representam as figuras de cada entidade, relacionamentos e compartimentos pertencentes ao diagrama) e (iv) os *Diagram Labels* (representam os rótulos para que o usuário receba um *feedback* de cada nó) no "*masmlRole.gmfgraph*". Adicionalmente, foram identificados e removidos os elementos que estão no metamodelo de MAS-ML do "*masml.ecore*" e que não fazem parte do diagrama de papel.

Nas entidades que fazem parte do diagrama de papel foram adicionados os C*ompartments*, que representam os compartimentos como eles devem ser vistos na ferramenta. Na entidade *AgentRoleClass* foram adicionados os compartimentos *Belief*, *Goal*, *Right*, *Duty* e *Protocol*. De forma semelhante, na entidade *Class* e na entidade *Object RoleClass*, foram adicionados os compartimentos *Operation* e *Property*. Na Figura 2 são apresentados os compartimentos da entidade *AgentRoleClass*.

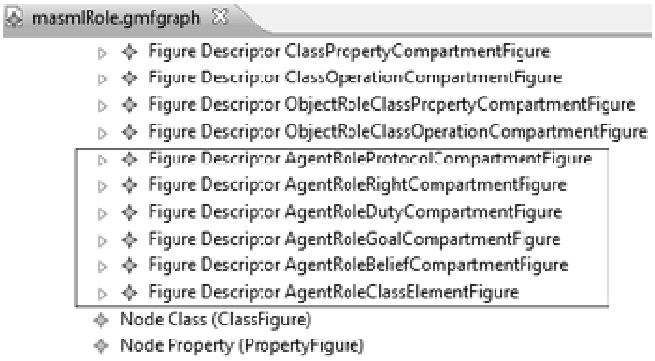

Fig.2. Compartimentos adicionados na entidade *agentRoleClass*.

No *graphical def model*, são definidas as formas visuais das entidades do diagrama. Dentre as formas disponíveis, nenhuma delas era adequada para representar o relacionamento *control* de acordo com a representação proposta em [Silva, 2004], isto é, através de uma reta com um círculo em uma das extremidades. Desta forma, foi necessário criar uma nova classe Java (*CircleDecoration.java*) no pacote "*masml.util*", a qual trata da representação do círculo. Assim, utilizando o círculo gerado por esta classe, é possível identificar quem é a entidade controladora (*controller*) do relacionamento.

Nos relacionamentos *Dependency*, *Generalization*, *Aggregation* e *Control* foram adicionadas as figuras correspondentes aos relacionamentos possibilitando a

diferenciação entre a entidade fonte e destino. Além disso, os relacionamentos *Association*, *Control* e *Aggregation* foram alterados para permitir a visualização das multiplicidades no diagrama de papel.

3.4. Extensão do *Tooling Def Model*

O "*masmlRole.gmftool*", derivado a partir do "*masml.ecore*", é responsável pela criação da paleta com os botões utilizados na ferramenta.

Durante o processo de derivação, é selecionada a classe principal ou raiz (*MasmlClassDiagram*), e em seguida, são marcados os componentes requeridos no Diagrama de Papel, indicando seu tipo (forma ou relacionamento).

Após a derivação, foram criados os botões no "*masmlRole.gmftool*", identificando e removendo os elementos que não fazem parte do diagrama de papel.

Após a remoção, a paleta passou a ser constituída por dois grupos: entidades e relacionamentos. O primeiro grupo é constituído por entidades (*AgentRoleClass*, *ObjectRoleClass*, *Class*) e compartimentos (*Belief*, *Goal*, *Right*, *Duty* e *Protocol*, *Operation*, *Property*). O segundo grupo é constituído por relacionamentos (*Association*, *Dependency*, *Generalization*, *Aggregation*, *Control*). Também foram definidas e adicionadas as figuras referentes a cada botão da paleta.

3.5. Extensão do *Mapping Model*

O "*masmlRole.gmfmap"* é resultado da combinação entre os metamodelos: "*masml.ecore*", "*masmlRole.gmfgraph*" e "*masmlRole.gmftool*". A combinação consiste na união de todas as informações contidas nesses três metamodelos. Com isso, é possível mapear cada elemento que compõe a paleta com o controlador e figura correspondentes. Durante o processo de combinação, é selecionada a classe principal ou raiz (*MasmlClassDiagram*). Em seguida, são identificados os elementos que são *Nodes* ou *Links* no diagrama de papel.

Após a combinação ser concluída, foram criados os *Nodes* e os *Links* correspondentes às entidades e relacionamentos, respectivamente. Em seguida, foi feita uma verificação no "*masmlRole.gmfmap*" e houve a necessidade de adicionar uma série de elementos, tanto nos *Nodes* como nos *Links,* para garantir que o diagrama de papel pudesse ser modelado corretamente.

Nos *Nodes*, foi feito o mapeamento dos elementos e seus compartimentos. No *Node* referente à entidade *Class,* foi adicionado em seu mapeamento os compartimentos *Property* e *Operation,* e suas devidas referências. Similarmente, no *Node* referente à entidade *ObjectRoleClass* foram inseridos os dois compartimentos citados. No *Node* referente à entidade *AgentRoleClass*, foi adicionado em seu mapeamento os compartimentos *Belief*, *Goal*, *Duty*, *Right*, *Protocol,* e suas devidas referências. Nas referências de cada compartimento foram feitas configurações, tais como o tipo dos elementos que seriam adicionados dentro do compartimento e a sua formatação.

Nos *Links* foi feito o mapeamento dos relacionamentos. No *Link* correspondente ao *Generalization* foi adicionada uma *Feature Label* que é responsável por mostrar o nome do relacionamento dentro do Diagrama de Papel. No *Link* correspondente ao *Control* foi determinado que a criação do relacionamento só fosse possível entre as entidades papel de agente (*AgentRoleClass*). Além disso, ainda no relacionamento

control, foram adicionadas três *Feature Label* responsáveis por mostrar o nome do relacionamento, multiplicidade da fonte e a multiplicidade do alvo do relacionamento dentro do Diagrama de Papel. No *Link* correspondente ao *Dependency* foi adicionada uma *Feature Label* é responsável por mostrar o nome do relacionamento *Dependency* dentro do Diagrama de Papel. No *Link* correspondente ao *Association* foram adicionadas cinco *Feature Label* as quais são responsáveis por mostrar a multiplicidade da fonte, a multiplicidade do alvo, o nome da fonte, o nome do alvo e o nome do relacionamento dentro do diagrama de papel. As mesmas cinco *Feature Label* foram adicionadas no *Link* correspondente ao *Aggregation*.

Além do mapeamento, foi preciso incorporar ao "*masmlRole.gmfmap*" a definição de algumas regras responsáveis pela validação da corretude dos modelos. Mais especificamente, estas regras se referem ao nome, de forma a garantir que toda entidade deve ter um nome, ao *Duty* e ao *Right*, estabelecendo que pelo menos um dos dois deva existir no papel de agente, e em relação ao *Protocol*, garantindo que esse compartimento sempre exista no papel de agente.

As regras adicionadas foram implementadas na linguagem OCL (*Object Constraint Language*) [OCL, 2011] que é uma linguagem declarativa para descrever as regras que se aplicam aos modelos. As novas regras criadas são descritas no Quadro 1, no qual pode ser vista a identificação da regra na ferramenta, propósito e sua definição em OCL.

Quadro I. Regras de Validação dos Modelos

Regra	Propósito e Definição em OCL
Regra 1	Todos os elementos do modelo devem ter um nome.
	name.size() > 0
Regra 2	Se não existe o Duty, então existe o Right.
	self.ownedDuty->isEmpty() = true implies self.ownedRight->isEmpty() = false
Regra 3	Se não existe o Right, então existe o Duty.
	self.ownedRight->isEmpty() = true implies self.ownedDuty->isEmpty() = false
Regra 4	O papel do agente deve ter protocolo.
	(self.ownedRight->isEmpty() = false or self.ownedDuty->isEmpty() = false) implies self.protocol->isEmpty() = false

3.6. Extensão do *Diagram Editor Gen Model*

Com a conclusão do mapeamento, é criado o "*masmlRole.gmfgen*". Em seguida é feita a transformação e a geração (*Generate diagram editor*) do código executável do projeto. Os modelos gerados a partir da ferramenta podem ser validados aplicando as restrições OCL implementadas, as quais podem ser acessadas através do menu *edit* e da opção *validate*. Caso alguma regra seja violada, o elemento que apresenta o problema é assinalado, e o detalhamento dos problemas é apresentado através do recurso *problems* do Eclipse.

A seguir é apresentado um estudo de caso onde é ilustrada a representação de entidades e relacionamentos modelados através da ferramenta MAS-ML *tool* de acordo com a especificação na linguagem.

4. Estudo de Caso

Como exemplo de modelagem para esse trabalho foi escolhido o ambiente de aprendizagem *Moodle* [MOODLE, 2011]. Esse ambiente é utilizado por instituições de ensino superior e médio como ambiente de aprendizagem colaborativa. O *Moodle* facilita a comunicação entre professores e alunos, para que eles possam trocar informações de maneira mais fácil, compartilhando os diversos recursos do ambiente via internet.

O *Moodle* assume que as pessoas aprendem melhor quando engajadas de forma colaborativa em um processo social de construção de conhecimento. Com base nas suas características e funções no sistema, uma variedade de agentes pode ser definida, onde cada um deles precisa desempenhar pelo menos um papel na organização. No ambiente *Moodle*, o papel Coordenador é responsável por administrar todo o ambiente e gerenciar a troca de informações entre as demais entidades. Adicionalmente, outros papéis de agente foram incorporados na modelagem do sistema: o CompanheiroAprendizagem, o Pedagogico, o BuscadorDeInformaçoes, o FormadordeGrupos e o AuxiliarDeUsabilidade.

De maneira geral, foi usado o relacionamento *Association* entre as classes de papel de agente, uma vez que esse relacionamento é o menos específico de todos e pode ser aplicado em qualquer caso. O uso do relacionamento *Control* é exemplificado entre as classes de papel de agente Coordenador e Pedagógico, uma vez que o agente que executa o papel Coordenador controla o agente que executa o papel Pedagógico.

Além desses papéis de agente, na modelagem foi adicionada a classe Disciplina, que mantém as informações que podem ser usadas pelo papel do agente Pedagógico.

Também foram adicionadas classes de papéis de objeto referentes às ferramentas de busca, criadas para fornecer ao papel de agente BuscadorDeInformaçoes as informações que ele precisa. Na Figura 3 pode ser vista a modelagem dos papéis exercidos no ambiente *Moodle* e seus relacionamentos.

5. Trabalhos Relacionados

Um ponto chave em relação às ferramentas de modelagem é que, normalmente, estas são projetadas com foco no suporte a uma linguagem de modelagem específica. Assim, as vantagens e desvantagens dessas linguagens são propagadas para as ferramentas que as implementam. Dentre as linguagens de modelagem que possuem ferramenta de suporte podemos destacar: (i) AUML [Odell, 2000], (ii) ANote [Choren e Lucena, 2004] e (iii) MAS-ML [Silva, 2004].

A AUML é uma linguagem que estendeu a UML para que pudesse modelar aspectos relacionados a agentes. No entanto, na AUML não são descritas as propriedades das organizações e dos papéis, além disso, os papéis de agentes e de objetos são definidos pela mesma metaclasse, o que segundo [Silva, 2004] não poderia acontecer. Com isso, a sua ferramenta de suporte não é capaz de modelar corretamente os papéis de um sistema. O ANote, apesar de possuir um *plugin* para a geração de código na plataforma eclipse chamado Albatroz, não identifica papéis. Portanto, não é possível modelá-los nessa ferramenta.

Fig.3. Diagrama dos papéis exercidos no ambiente *Moodle* gerado com MAS-ML *tool*.

Considerando as ferramentas de modelagem já existentes relacionadas a MAS-ML, *VisualAgent* [De Maria, 2005] é um ambiente de desenvolvimento que ajuda o desenvolvedor na produção de especificações, projeto, e implementação de SMAs.

O *VisualAgent* é baseado no metamodelo original da MAS-ML, e consequentemente, o suporte para à modelagem de agentes com diferentes arquiteturas internas é limitado. Além de também não fornecer suporte à modelagem de todos os diagramas descritos na linguagem de modelagem. Adicionalmente, a ferramenta *VisualAgent* não possui nenhum mecanismo que possibilite a checagem pela corretude dos modelos. A ausência desse recurso no *VisualAgent* pode comprometer a qualidade dos modelos elaborados e consequentemente, a do código gerado a partir de tais modelos. Além disso, a falta de documentação e o difícil acesso ao código fonte dificultam a continuidade do projeto.

Com a evolução proposta em MAS-ML *tool*, os três diagramas estáticos previstos para a modelagem de SMA são contemplados em consistência com a versão mais atual da linguagem possibilitando a modelagem das diversas arquiteturas de agentes. Adicionalmente, a ferramenta incorpora um mecanismo para a checagem de correção dos modelos gerados responsável pela verificação com base nas restrições estabelecidas em nível de linguagem.

6. Conclusão e Trabalhos Futuros

A função do papel no contexto de SMA é orientar ou restringir o comportamento da entidade que está incumbida a exercê-lo. O diagrama de papéis é importante no contexto de modelagem de SMAs por ser capaz de mostrar os relacionamentos existentes entre os papéis de agente no sistema. Neste trabalho foi apresentada uma extensão da ferramenta MAS-ML *tool* para possibilitar a modelagem do diagrama de papéis definido na linguagem MAS-ML de acordo com a versão 2.0. Com isso, os três diagramas estáticos previstos pela linguagem podem ser gerados através da ferramenta.

Com a extensão da ferramenta MAS-ML *tool*, é possível exibir todas as interações entre as entidades pertencentes ao diagrama de papéis que foram definidas na MAS-ML 2.0. Adicionalmente, a extensão proposta prevê a validação da corretude do diagrama gerado de forma a verificar a consistência da sua construção, reduzindo a ocorrência de erros e tornando a modelagem mais segura.

Existem alguns trabalhos futuros previstos no intuito de dar continuidade ao projeto de evolução da ferramenta apresentado nesse artigo. Dentre eles podem ser citados: (i) melhoramentos na representação gráfica dos construtores de acordo com a representação proposta pela linguagem MAS-ML; (ii) implementação dos diagramas dinâmicos na ferramenta MAS-ML *tool,* a saber: diagrama de sequência e atividades; (iii) geração de código a partir dos diagramas gerados pela ferramenta MAS-ML *tool* e (iv) unir todos os diagramas criados em um único projeto executável, pois os diagramas foram criados de forma separada na ferramenta (cada diagrama tem seu executável).

Agradecimentos

A. R. Feijó e F. R. O. Lima agradecem o apoio financeiro do CNPq/Brasil e da Funcap, respectivamente. A. R. Feijó, F. J. A. Maia e F. R. O. Lima agradecem aos professores orientadores M. I. Cortés e E. J. T. Gonçalves.

Referências

Choren, R. Lucena, C. "*Agent-Oriented Modeling Using ANote*", *3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS*

2004), 3rd; The Institution of Electrical Engineers, IEE, Stevenage, UK, 2004, pp. 74-80, ISBN: 0-86341-431-1, May 24-25 , 2004.

De Maria, B. A. V. T. da Silva, C. J. P. Lucena, R. Choren, "*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.

Eclipse, "*Eclipse Platform*," disponível em:<http:// www.eclipse.org>, acessado em 2 de Junho de 2011.

EMF, disponível em:<http://www.eclipse.org/modeling/emf/>, acessado em 7 de Junho de 2011.

Farias, K. I. Nunes, V. T. da Silva, C. J. P. de Lucena, "MAS-ML *Tool*: Um ambiente de modelagem de sistemas multi-agentes," *Fifth Workshop on Software Engineering for Agent-oriented Systems* (SEAS@SBES 09), *Brazil*, 2009.

GEF, disponível em:<http://www.eclipse.org/gef/>, acessado em 7 de Junho de 2011.

GMF, disponível em:<http://www.eclipse.org/modeling/gmf/>, acessado em 7 de Junho de 2011.

Gonçalves, E. J. T. K. Farias, M. I. Cortés, V. T. da Silva, R. G. F. Feitosa, "Modelagem de organizações de agentes inteligentes: uma extensão da MAS-ML *Tool*", *1st Workshop on Autonomous Software Systems September*, 27, 2010, Salvador – Bahia – Brasil.

Jennings, N. "*Coordination Techniques for Distributed Artificial Intelligence*," *In: Foundations of Distributed Artificial Intelligence*, pp. 187-210, *Wiley,* 1996.

Lind, J. "*Issues in Agent-Oriented Software Engineering*". *In: Ciancarini* P. e *Wooldride* M., LNCS 1957, *Germany*, *Springer*, p.45-58, 2001.

MOODLE, disponível em:<http://www.moodle.org.br>, acessado em 10 de Junho de 2011.

OCL, 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. "*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), 2000.

OMG, "UML: *unified modeling language specification*," versão 2.2, disponível em:<http://www.uml.org>, acessado em 2 de Junho de 2011.

Russell, S. P. Norvig, "Inteligência artificial: uma abordagem moderna," 2ª Ed. *Prentice-Hall*: São Paulo, 2004.

Silva, V. T. "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, 2004.

Silva, V. T. R. Choren, C. J. P. Lucena, "MAS-ML: *A Multi-Agent System Modeling Language*," *In: Conference on Object-oriented programming, systems, languages, and applications*, 18th *annual* ACM SIGPLAN; CA, USA, ACM *Press*, pp. 304-305, 2007.