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

Állan Ribeiro Feijó
Faculdade 7 de Setembro (FA7), Eng. Luciano Cavalcante, 60811-020 – Fortaleza/CE – Brasil
allanfeijo1987@gmail.com

Abstract. *The modeling activity can be highly complex and tending to errors, particularly, in a Multi-Agent System (MAS) modeling case. Thus, the existence of some adapted support tool can be crucial for choosing and adopting the modeling language to be used. In this respect, the MAS support tool, MAS-ML Tool, developed to give a support for MAS-ML modeling language includes mechanisms to model only three proposed diagrams by MAS-ML. The goal of this paper is present the MAS-ML tool evolution to provide a support to the sequence diagram defined on this language. The tool evolution follows the model-based approach using Graphical Modeling Framework, based on the MAS-ML 2.0 metamodel as a starting point.*

Resumo. *A atividade de modelagem pode ser altamente complexa e propensa a erros, em particular no caso da modelagem de Sistemas Multiagente (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 de suporte a modelagem de SMA, MAS-ML tool, desenvolvida para dar suporte à linguagem de modelagem MAS-ML incorpora mecanismos para a modelagem de somente três dos diagramas propostos por MAS-ML. O objetivo deste artigo é apresentar a evolução da ferramenta MAS-ML tool de forma de dar apoio à modelagem do diagrama de sequência previsto na linguagem. A evolução da ferramenta segue a abordagem dirigida por modelos utilizando o Graphical Modeling Framework (GMF), com base no metamodelo de MAS-ML 2.0 como ponto de partida.*

I. INTRODUÇÃO

Com a crescente busca por sistemas mais eficientes leva indefetivelmente, 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]0. O termo Sistema Multiagente (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]0.

Neste cenário, a engenharia de software orientada a agentes (AOSE) [Lind, 2001] 0 surge no intuito de fornecer métodos e ferramentas adequados para o desenvolvimento de aplicações neste novo domínio. Mais especificamente, linguagens de modelagem cuja utilização, é apoiada por ferramentas de suporte adequadas, têm um papel central no desenvolvimento de SMA. A sua importância advém do fato de que ferramentas projetadas

tomando como base as premissas estabelecidas na linguagem propiciam a boa formação dos modelos gerados, reduzindo erros e aumentando a produtividade na execução das atividades de análise e projeto.

No contexto das linguagens de modelagem para SMA destacamos a MAS-ML (*Multi-Agent System Modeling Language*) [Silva, 2004]0. O objetivo da MAS-ML é modelar todos os aspectos dinâmicos e estruturais definidos no *framework* TAO (*Taming Agents and Objects*) [Silva, 2004]0. O metamodelo de MAS-ML é uma extensão do metamodelo da UML (*Unified Modeling Language*) 0[OMG] 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]0.

Originalmente, MAS-ML foi projetada para modelar apenas agentes pró-ativos orientados a objetivos e guiados por planos. Com a necessidade de modelar as outras arquiteturas internas definidas em [Russel e Norvig, 2004]0, a versão 2.0 de MAS-ML [Gonçalvez, 2010]0, foi gerada. Com ela é possível projetar agentes com diversas arquiteturas internas, assim como papéis adequados às arquiteturas, além das características originais de MAS-ML como organizações e ambientes e os diversos aspectos estruturais e dinâmicos dos SMAs.

A ferramenta MAS-ML *tool* 0[Farias, 2009] é um ambiente de modelagem (editor gráfico) para SMAs desenvolvido como um *plug-in* da plataforma Eclipse [Eclipse]0. Isto implica que os usuários poderão ao mesmo tempo modelar SMAs e utilizar os recursos disponíveis dentro da plataforma Eclipse. Em [Feijó, 2012]0 a ferramenta foi estendida com a implementação do diagrama de papéis definido em MAS-ML 2.0. Embora a ferramenta MAS-ML *tool* foi desenvolvida para dar suporte à modelagem dos diagramas previstos em MAS-ML, na sua versão atual, o diagrama de sequência contemplado na ferramenta.

A importância do diagrama de sequência de MAS-ML 2.0 está no fato de conseguir representar os aspectos dinâmicos de SMAs, ou seja, de representar as interações entre as instâncias do SMA e as intra-ações (i.e., ações executadas internamente pelas entidades).

O objetivo desse artigo é apresentar a evolução da ferramenta MAS-ML *tool* de forma a dar suporte à modelagem do diagrama de sequência de acordo com a especificação apresentada na versão 2.0 de MAS-ML é fundamental à modelagem de SMAs através desta linguagem. A nova versão da ferramenta é ilustrada a partir de um estudo de caso utilizados no ambiente de aprendizagem virtual *Moodle* (*Modular Object-Oriented Dynamic Learning Environment*) [MOODLE]0 foram modelados.

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

II. REFERENCIAL TEÓRICO

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

A. MAS-ML 2.0 e o Diagrama de Sequência

MAS-ML é uma linguagem de modelagem que estende a UML e que incorpora o conceito de agente do *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: (i) arquiteturas internas de agentes reativos simples; (ii) agentes reativos baseados em conhecimento; e (iii) agentes baseados em objetivo com planejamento e (iv) agentes baseados em utilidade. Na 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 Figura 1 mostra o metamodelo MAS-ML estendido.

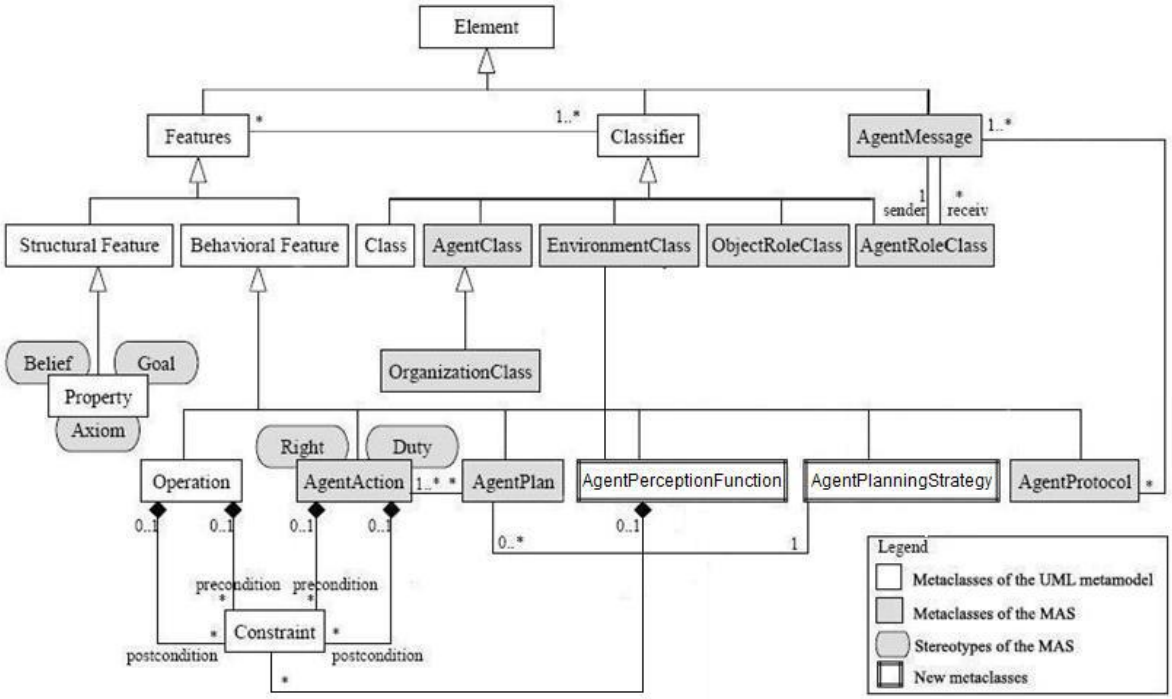

Fig. 1. Metamodelo MAS-ML estendido0.

A linguagem MAS-ML contempla um conjunto de diagramas estáticos e dinâmicos, a saber: diagrama de classes, organização e papéis como estáticos, e diagrama de sequência e atividade como dinâmicos.

Segundo [Gonçalvez, 2009], o diagrama de sequência modela o comportamento de um sistema orientado a objetos, compreendendo a modelagem de atores, objetos, criação e destruição dos mesmos e troca de mensagens. A extensão do diagrama de sequência de UML para representar os aspectos dinâmicos de SMAs, ou seja, para representar as interações entre as instâncias do SMA e as intra-ações definidas por cada instância envolve dois aspectos:

A extensão do diagrama de sequência se deu pela definição de novos *pathnames* e ícones para as instâncias dos SMAs (agentes, organizações e ambiente). Segundo, para representar as interações entre agentes, organizações, ambientes e objetos, a definição do conceito mensagem usado em UML precisou ser estendida para representar entidades que estão enviando e recebendo mensagens, mas não através de chamadas a métodos de outras entidades. Agentes interagem enviando mensagens e não chamando métodos. Além disso, foram criados estereótipos para representar a criação e 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.

O diagrama de sequência ainda foi estendido para representar a execução de planos e ações enquanto modela as intra-ações relacionadas a agentes, organização e ambientes. A seguir são descrita a representação das características de cada arquitetura interna no diagrama de sequência.

De acordo com a extensão da MAS-ML, o conceito de percepção é representado nos agentes reativos. Portanto, se um agente reativo simples for modelado, inicialmente teremos sua percepção e em seguida suas ações guiadas pelas regras condição-ação. No caso de um agente reativo baseado em conhecimento, inicialmente teremos a percepção, em seguida a função próximo e, por último, suas ações baseadas em regras condição-ação. Já os agentes pró-ativos, agente baseado em objetivo guiado por plano, mantém à representação proposta por Silva (2004), assim como também o plano definido durante a fase de projeto. No caso do agente baseado em objetivo guiado por planejamento, inicialmente é executada a percepção, função próximo, função de formulação de objetivo, função de formulação do problema e em seguida a execução de seu planejamento. Finalmente, para o agente baseado em utilidade, necessitamos da percepção, função próximo, função de formulação de objetivo, função de formulação do problema, planejamento, função utilidade e resulta nas ações que são executadas, nesta ordem.

B. MAS-ML TOOL

A ferramenta MAS-ML *tool* é um ambiente de modelagem (editor gráfico) que serve como um *plug-in* da plataforma Eclipse. Isto implica que os usuários poderão ao mesmo tempo modelar SMAs e utilizar os recursos disponíveis dentro da plataforma Eclipse. MAS-ML *tool* foi criada para dar suporte a 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*) 0[GMF] que é um *framework* para desenvolvimento de editores gráficos para modelos de domínio. Ele surgiu de uma união de dois *frameworks* denominados GEF (*Graphical Editing Framework*) [GEF]0, utilizado para a criação de editores gráficos genéricos e EMF (*Eclipse Modeling Framework*) [EMF]0, 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.

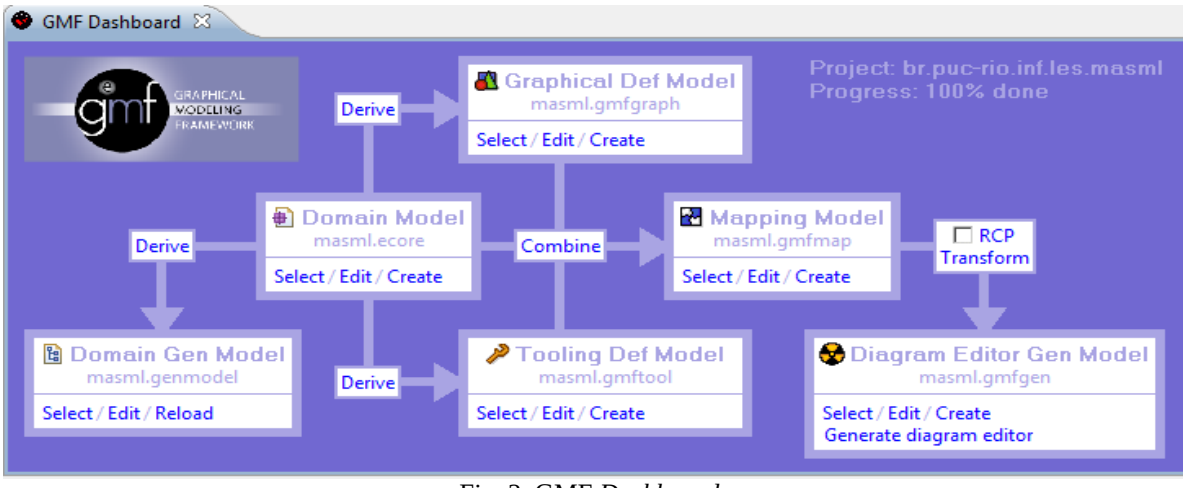

Fig. 2. GMF *Dashboard*.

As etapas (Figura 2) para a sua contrução são as seguintes: (i) *domain model*, onde é definido o modelo de domínio; (ii) *domain gen model*, que estende modelo de domínio através da geração de dados (Classes java); (iii) *graphical def model*, onde é definido as figuras e controladores; (iv) *tooling gef model*, onde é definido a paleta, o menu, 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.

III. IMPLEMENTAÇÃO DO DIAGRAMA DE SEQUÊNCIA

A estratégia adotada para implementar as extensões propostas segue a abordagem dirigida por modelos, que foi utilizada originalmente para desenvolver a 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.

A. 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.* Sendo que todas essas já estavam presentes no metamodelo do Diagrama de Papel.

O *Domain Model* contém o metamodelo "*masml.ecore*". Nesse metamodelo foram criados os relacionamentos entre a metaclasse "*MasmlClassDiagram*" e as metaclasses (i) *Class*, (ii) *AgentClass*, (iii) *OrganizationClass*, (iv) *EnvironmentClass*, (v) *ControlStructures*, (vi) *AgentRoleClass*, (vii) *PlanClass*, (viii) *ActionClass*, (ix) *Planning*, (x) *Perception*, (xi) *AgentMessageClassDefault*, (xii) *ObjectMessageClass*. Sendo que todas essas já estavam presentes no metamodelo do diagrama de dequência, exceto as metaclasses *ControlStructures* (*for*, *if* e *else*) e *ObjectMessageClass*, que foi preciso adicionar ao metamodelo.

Adicionalmente, foi necessário adicionar outros elementos ao metamodelo para dar suporte à construção do diagrama de sequência. Os elementos são: *FocusControl* (foco de ativação) e *Line* (linha da vida). A Figura 3 apresenta tanto as metaclasses como os relacionamentos adicionados.

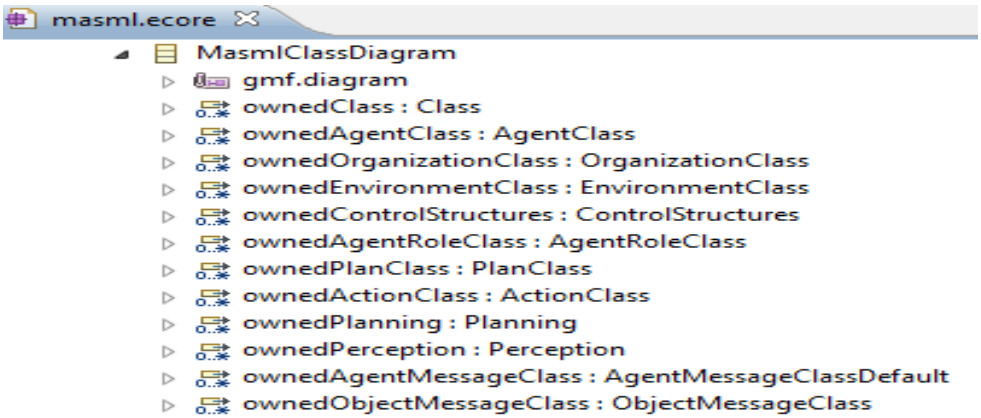

Fig. 1. As metaclasses como os relacionamentos adicionados.

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

B. Extensão do Domain Gen Model

O "*masml.ecore*" foi derivado (transformado) em um metamodelo mais específico, o "*masml.genmodel*". Com esse metamodelo, são geradas as classes Java (códigos) de acordo com o metamodelo proposto (*masml.ecore*): (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*. A Figura 4 apresenta tanto os pacotes como os projetos gerados.

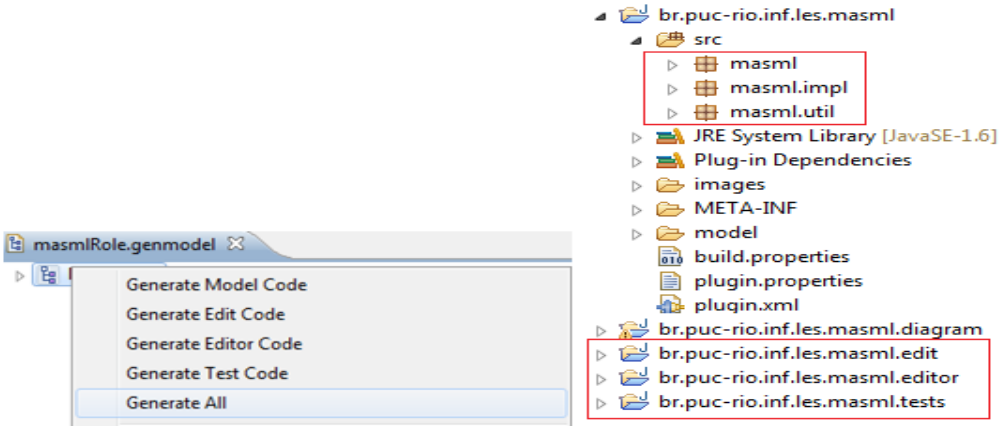

Fig. 2. Os pacotes e projetos gerados a partir do *genmodel*.

C. Extensão do Graphical Def Model

O "masml.gmfgraph" foi derivado a partir do "masml.ecore" e é responsável pela criação dos componentes gráficos.

Durante o processo de derivação foi selecionada a classe principal ou raiz (*MasmlClassDiagram*), em seguida foram marcados os componentes que estão presentes no diagrama de sequência indicando o tipo de componente em termos de forma, relacionamento ou texto.

Após a derivação, foram criados os *Nodes* (representam as entidades), *Conections* (representam os relacionamentos pertencentes ao diagrama), *Figures* (representam as figuras de cada entidade e os relacionamentos pertencentes ao diagrama) e os *Diagram Labels* (representam os rótulos para que o usuário tenham um *feedback* de cada nó) no "*masml.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 sequência.

No *graphical def model*, são definidas as formas visuais das entidades do diagrama. Dentre as formas disponíveis, nenhuma delas era adequada para representar as entidades *OrganizationClass* e *AgentRoleClass* de acordo com as suas definições em [Silva, 2004]. Por isso, foi necessário criar duas novas classes Java (*OrganizationClassShape.java* e *AgentRoleClassShape.java*) no pacote "*masml.util*", a qual trata da representação da forma gráfica.

Nos relacionamentos *Action*, *AgentMessageActivate*, *AgentMessageCancel*, *AgentMessageCommitment*, *AgentMessageCreate*, *AgentMessageDeactivate*, *AgentMessageDefault*, *AgentMessageDestroy*, *ChangeDeactivateReactivate*, *ChangeDeactivateCreate*, *ChangeDestroyCreate*, *ChangeDestroyReactivate*, *For*, *If*,

ObjectMessage, *Perception*, *Plan* e *Planning* foram adicionadas as figuras correspondentes dos relacionamentos para haver uma diferenciação entre a entidade fonte e destino. A Figura 5 apresenta a definição da forma visual do relacionamento *ActionClass*.

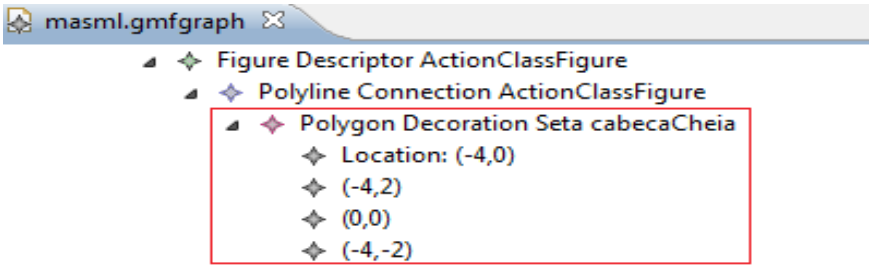

Fig. 5. A definição forma visual dos relacionamentos.

D. Extensão do Tooling Def Model

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

Durante o processo de derivação, foi selecionada a classe principal ou raiz (*MasmlClassDiagram*), e em seguida, foram marcados os componentes requeridos no diagrama de sequência, indicando seu tipo (forma ou relacionamento).

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

Após a remoção, a paleta passou a ser constituída por dois grupos: entidades e relacionamentos. O primeiro grupo é constituído por entidades (*AgentClass*, *AgentRoleClass*, *Class*, *EnvironmentClass*, *OrganizationClass*) e alguns recursos (*LifeLine*, *FocusControl*, *SubFocusControl*), o segundo grupo é constituído por relacionamentos (*Action*, *AgentMessageActivate*, *AgentMessageCancel*, *AgentMessageCommitment*, *AgentMessageCreate*, *AgentMessageDeactivate*, *AgentMessageDefault*, *AgentMessageDestroy*, *ChangeDeactivateReactivate*, *ChangeDeactivateCreate*, *ChangeDestroyCreate*, *ChangeDestroyReactivate*, *For*, *If*, *Else*, *ObjectMessage*, *Perception*, *Plan*, *Planning*). A Figura 6 apresenta as entidades e os relacionamentos adicionados à paleta.

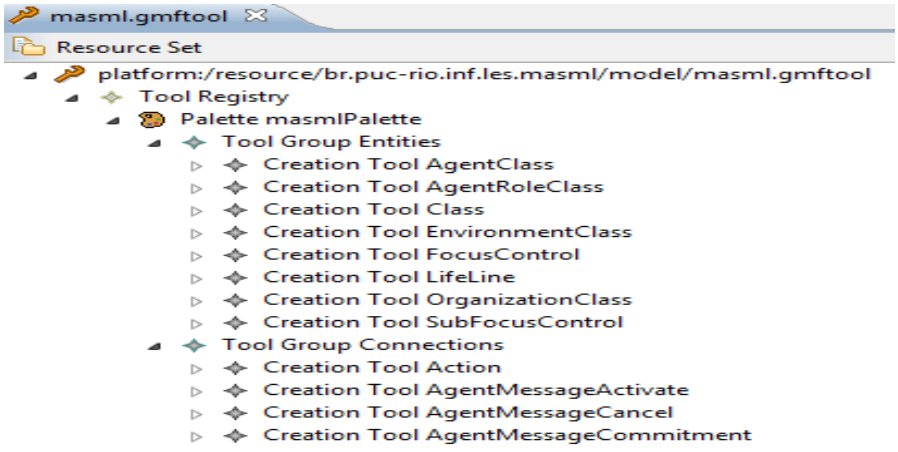

Fig. 6. As entidades e os relacionamentos adicionados à paleta.

Também foram definidas e adicionadas as figuras referentes a cada botão.

E. Extensão do Mapping Model

O "*masml.gmfmap*" é resultado da combinação entre os metamodelos: "*masml.ecore*", "*masml.gmfgraph*" e "*masml.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 seu controlador e sua figura correspondentes. Durante o processo de combinação, foi selecionada a classe principal ou raiz (*MasmlClassDiagram*). Em seguida, foram identificados os elementos que são *Nodes* ou *Links* no diagrama de sequência.

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 "*masml.gmfmap*" e houve a necessidade de adicionar uma série de elementos tanto nos *Nodes* como nos *Links* para garantir que o diagrama de sequência pudesse ser modelado corretamente.

Nos *Nodes*, foi feito o mapeamento dos elementos com seus recursos. No *Node* referente à entidade *Class*, por exemplo, foi adicionado em seu mapeamento os recursos (*LifeLine*, *FocusControl*, *SubFocusControl*), suas devidas referências e formatações. Similarmente nos demais *Nodes*.

Nos *Links* foi feito o mapeamento dos relacionamentos. No *Link* correspondente ao *For* e *If* foi adicionada uma *Feature Label* que é responsável por mostrar o nome do relacionamento dentro do diagrama de sequência. Similarmente nos demais *Links*. As Figuras 7 e 8 apresentam os mapeamentos feitos no *Node* e no *Link*, respectivamente.

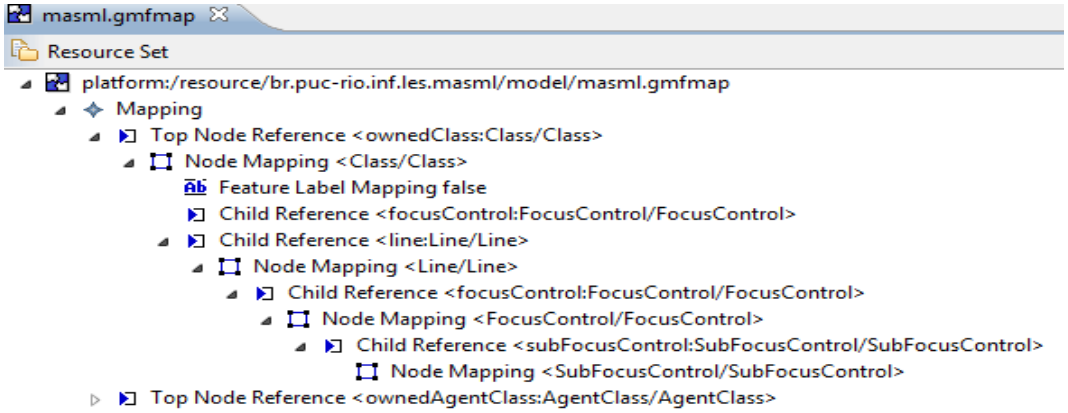

Fig. 3. Os mapeamentos feitos nos *Nodes*.

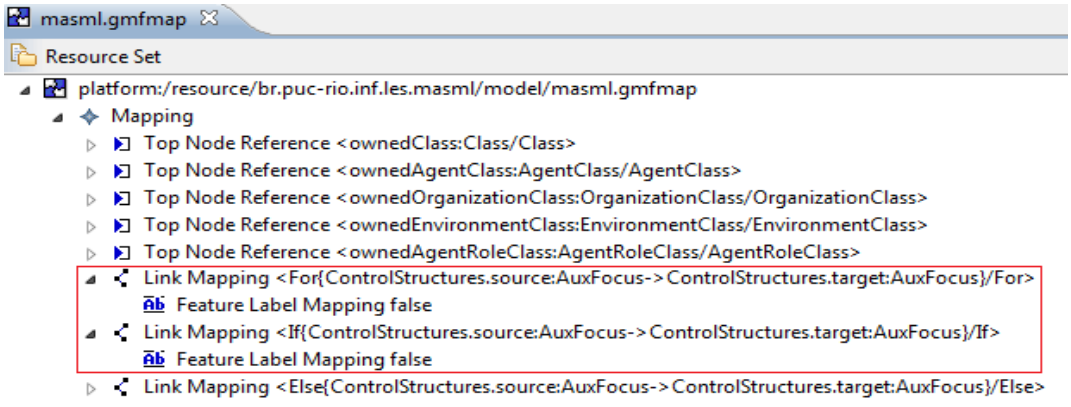

Fig. 4. Os mapeamentos feitos nos *Links*.

Além do mapeamento, foi preciso incorporar ao "*masml.gmfmap*" a definição de algumas regras para a validação do modelos, são: (i) todos os elementos do modelo devem ter um nome, (ii) se o agente possui plano, então ele possui ação, (iii) se o agente possui plano, então não possui percepção, (iv) se o agente possui planejamento, então ele possui percepção e ação, (v) se o agente possui plano, então ele não possui planejamento, (vi) caso o agente possua planejamento, então ele não terá plano.

As regras adicionadas foram implementadas na linguagem OCL (*Object Constraint Language*) [OCL, 2011]. As regras criadas são descritas no Quadro 1, na qual pode ser vista a identificação da regra na ferramenta, propósito e sua definição em OCL.

QUADRO 1. REGRAS DE VALIDAÇÃO DOS MODELOS (DIAGRAMA DE SEQUÊNCIA).

Regra	Propósito e Definição em OCL
Regra 1	Todos os elementos do modelo devem ter um nome.
	name.size() > 0
Regra 2	Se o agente possui plano, então ele possui ação.
	self.ownedPlan->isEmpty() = false implies self.owendAction->isEmpty() = false
Regra 3	Se o agente possui plano, então não possui percepção.
	self.ownedPlan->isEmpty() = false implies self.ownedPerception->isEmpty() = true
Regra 4	Se o agente possui planejamento, então ele possui percepção e ação.
	self.ownedPlanning->isEmpty() = false implies (self.ownedPerception->isEmpty() = false and self.owendAction->isEmpty() = false)
Regra 5	Se o agente possui plano, então ele não possui planejamento.
	self.ownedPlan->isEmpty() = false implies self.ownedPlanning->isEmpty() = true
Regra 6	Caso o agente possua planejamento, então ele não terá plano.
	self.ownedPlanning->isEmpty() = false implies self.ownedPlan->isEmpty() = true

F. Extensão do Diagram Editor Gen Model

Com a conclusão do mapeamento é criado o "*masml.gmfgen*". Em seguida é feita a transformação e a geração (*Generate diagram editor*) do código executável do projeto. A Figura 9 apresenta o resultado da geração do "*masml.gmfgen*", modelo responsável pela geração do projeto executável.

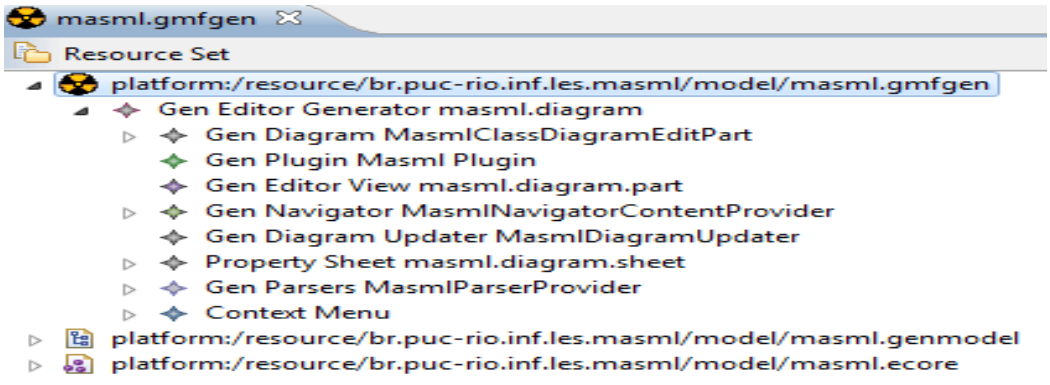

Fig. 5. O modelo responsável pela geração do projeto executável.

As Figuras 10 e 11 apresentam, respectivamente, a representação de cada uma das entidades e os relacionamentos na ferramenta MAS-ML tool de acordo com a especificação na linguagem. 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*.

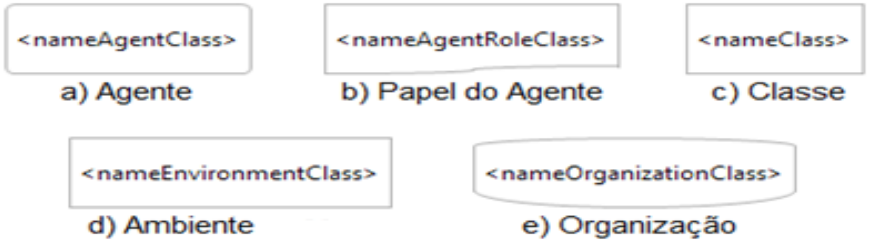

Fig. 6. A representação das entidades na ferramenta MAS-ML *tool*.

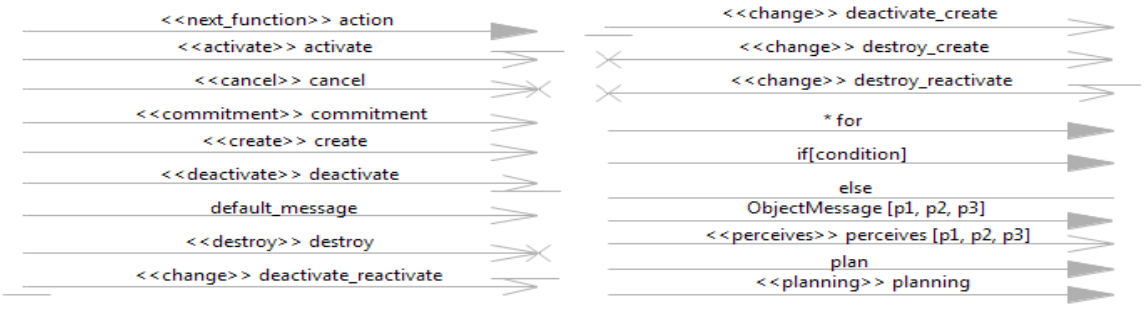

Fig. 7. A representação dos relacionamentos na ferramenta MAS-ML *tool*.

IV. 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 via internet compartilhando os diversos recursos do ambiente.

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 de pelo menos um papel para poder desempenhar na organização. No ambiente *Moodle* existem basicamente seis tipos de agente, são eles: (i) AgenteCoordenador, (ii) AgenteFormadorDeGrupos, (iii) agenteAuxiliarDeUsabilidade, (iv) AgenteBuscadorDeInformacoes, (v) AgentePedagogico, (vi) AgenteCompanheiroAprendizagem. Cada um desses agentes realiza tarefas diferentes no ambiente *Moodle*. A seguir o diagrama de sequência desenvolvido em MAS-ML *tool* é utilizado para ilustrar as várias interações de dois desses agentes em tempo de execução, o agente AgenteBuscadorDeInformacoes e AgenteCompanheiroAprendizagem respectivamente.

O AgenteBuscadorDeInformacoes, dependendo do tipo de informação a ser pesquisada, busca em sua base de dados e envia as informações que pode encontrar. Caso ele procure por uma pessoa, ele tenta localizá-la, após encontrar essa informação, ele busca as pessoas que estão relacionadas à pessoa que foi encontrada. O mesmo acontece quando o agente busca por documentos. A Figura 12 apresenta as interações do agente buscador com o ambiente.

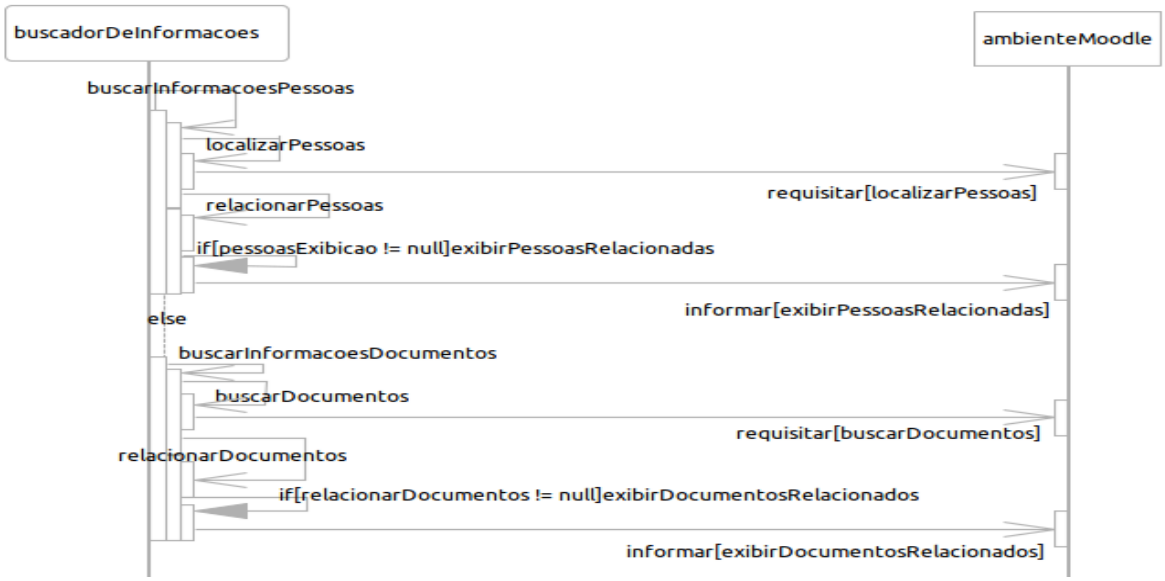

Fig. 12. Agente buscador.

O AgenteCompanheiroAprendizagem percebe, através do ambiente, se o aluno está com dificuldades em uma certa disciplina, dependendo de sua situação o AgenteCompanheiroAprendizagem pode informar ao AgenteCoordenador a situação do aluno para que este requisite algum apoio para o mesmo. A Figura 13 apresenta as interações do agente companheiro aprendizagem.

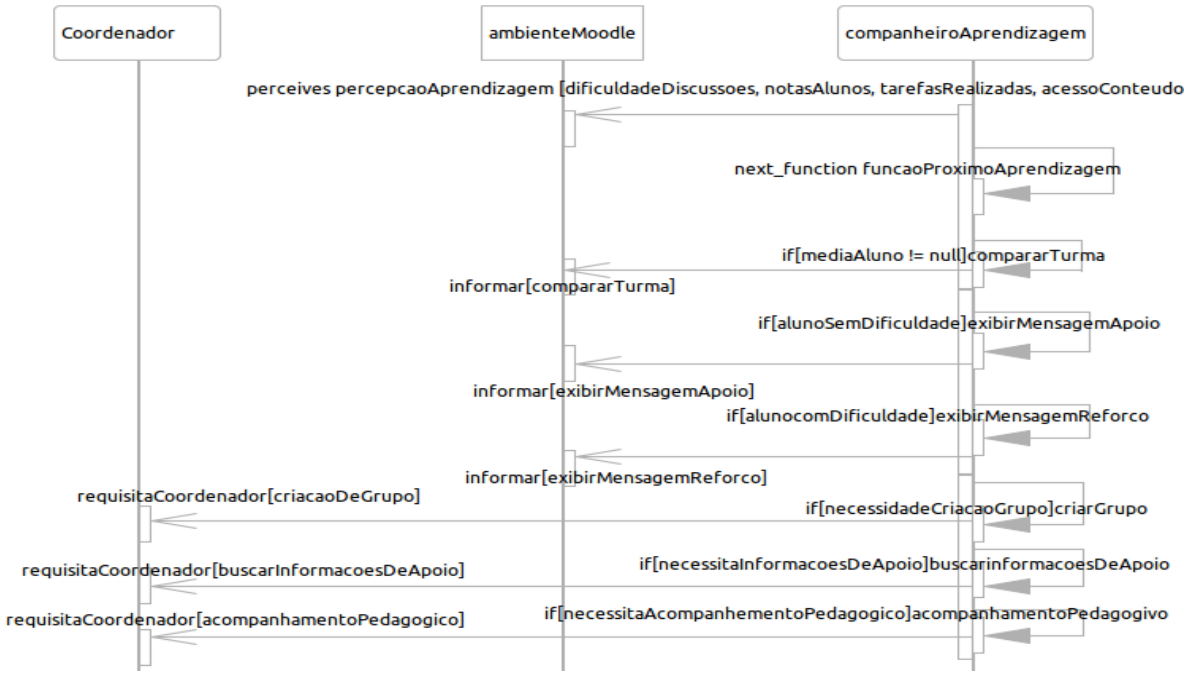

Fig. 13. Agente companheiro aprendizagem.

V. TRABALHOS RELACIONADOS

Um ponto chave em relação às ferramentas de modelagem de SMAs é que, normalmente, estas são projetadas com foco no suporte de uma linguagem de modelagem específica. Assim, as vantagens e desvantagens dessas linguagens são propagadas para as ferramentas que as implementam.

Considerando as ferramentas de modelagem já existentes relacionadas a MAS-ML, *VisualAgent* 0[De Maria] 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 pode ser limitado. Além de também não fornecer suporte à modelagem de todos os diagramas descritos pela MAS-ML 2.0. 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*, dos dois diagramas dinâmicos previstos na linguagem o diagrama de sequência está contemplado, junto com os três diagramas estáticos previstos pela linguagem para a modelagem de SMA, em consistência com a versão mais atual da linguagem para a modelagem das diversas arquiteturas de agentes. Nesse sentido, o mecanismo para a checagem pela corretude dos modelos gerados contempla a verificação com base nas restrições estabelecidas a nível de linguagem.

VI. CONCLUSÃO E TRABALHOS FUTUROS

O diagrama de sequência modela características dinâmicas das entidades de um SMA como a criação e a destruição dos mesmos e a troca de mensagens. Neste trabalho foi apresentada uma evolução da ferramenta MAS-ML *tool* para possibilitar a modelagem do diagrama dinâmico de sequência definidos na linguagem MAS-ML 2.0. Com isso, os três diagramas estáticos: classes, organização e papéis, e o diagrama dinâmico de sequência 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 sequência que foram definidas na MAS-ML 2.0.

Adicionalmente, a extensão proposta prevê a validação da boa formação do diagrama gerado de forma a verificar se existem erros em 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 neste trabalho. Dentre eles podem ser citados: (i) implementação do diagrama de atividades; (ii) geração de código a partir dos diagramas construídos a partir da ferramenta MAS-ML *tool*; (iii) unir todos os diagramas criados em um único projeto executável, pois os diagramas foram criados de forma separada na ferramenta (cada diagrama têm seu executável).

AGRADECIMENTOS

A. R. Feijó agradece a Deus, por ter me amparado em todos os momentos da minha vida, me ajudando, orientando e incentivando a seguir. Aos meus pais, irmão, e avós, que mesmo distantes me deram muita força, incentivando os meus estudos e que sempre foram exemplos de vida. Aos professores da pós-graduação da FA7 que contribuíram ao longo do curso para aprimorar os meus conhecimentos. A meu orientador Marum Simão Filho, pela paciência, atenção e dedicação oferecidas durante a construção deste trabalho. Aos colegas da pós-graduação pelas experiências trocadas. E a todas as pessoas de uma forma geral que direta ou indiretamente contribuíram para a realização deste trabalho. A todos meus eternos agradecimentos!

REFERÊNCIAS

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.

Feijó, A. R., Evolução da Ferramenta MAS-ML tool para a Modelagem do Diagrama de Papéis, Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCações - WESAAC, Florianópolis, Santa Catarina, Brasil, 2012.

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. "Modelagem de Arquiteturas Internas de Agentes de Software Utilizando a Linguagem MAS-ML 2.0". Dissertação de Mestrado. UECE, Centro de Ciência e Tecnologia, 2009, Fortaleza – Brasil.

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.

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.