Uma Metodologia MDA Para Apoio Ao Desenvolvimento Semi-Automático de Sistemas Multi-Agentes

Carlos Eduardo Pantoja[1], Ricardo Choren[2]

[1]CEFET/RJ – UnED Nova Friburgo – Av. Gov. Roberto da Silveira, 1900 – Nova Friburgo – RJ – Brasil

[2]Instituto Militar de Engenharia – IME/RJ – Pça Gen Tibúrcio 80 – Rio de Janeiro – 22270-290 – RJ – Brasil

pantoja@cefet-rj.br, choren@ime.eb.br

Abstract. *This paper presents a MDA methodology for MAS development that uses models in different abstraction levels, going from system specification, and after a transformation set between models, get to the specific platform JASON/Moise+ codification. The methodology uses the FAML as platform independent model, which is a metamodel that englobes the concepts of several agent-oriented methodologies for MAS development into a single model. As platform specific model, the JaCaMo model was chosen. The transformations between platform independent model and platform specific model are programmed in QVT language and transformations between platform specific model and the execution platform are programmed in M2T language. This paper also presents a MAS development environment, which is a set of Eclipse plug-ins and the Gold Miners system that was modeled using Prometheus.*

Resumo *Este artigo apresenta uma metodologia MDA de desenvolvimento de sistemas multi-agentes que utiliza de modelos em diferentes níveis de abstração, partindo da especificação do sistema, e após um conjunto de regras de transformações entre modelos, chegar até a codificação da plataforma de execução JASON/Moise+. A metodologia utiliza o FAML como modelo independente de plataforma, que é um meta-modelo que reúne os conceitos de diversas metodologias de desenvolvimento de sistemas multi-agentes em um único modelo. Como modelo específico de plataforma foi escolhido o modelo JaCaMo. As transformações entre o modelo independente de plataforma e o modelo específico de plataforma são programadas na linguagem QVT e as transformações entre o modelo específico de plataforma e a codificação da plataforma de execução são programadas na linguagem M2T. Este trabalho apresenta, também, um ambiente de desenvolvimento de sistemas multi-agentes, que consiste de um conjunto de plug-ins para a plataforma de desenvolvimento Eclipse e o sistema Gold Miners que foi modelado usando Prometheus.*

1. Introdução

Agentes são componentes autônomos e cognitivos, originados da inteligência artificial, situados em um ambiente e não são receptores passivos de ações executadas por outras entidades, pois possuem uma biblioteca de planos com possíveis ações em resposta aos estímulos percebidos com a finalidade de atingir seus objetivos de projeto e

modificar o ambiente em que estão inseridos [Wooldridge, 2000].

As metodologias de desenvolvimento de Sistemas Multi-Agentes (SMA) consistem em uma coleção de modelos para formalizar e entender o sistema modelado antes de sua implementação. Esses modelos começam com uma tentativa de serem os mais abstratos possíveis até atingirem um nível mais próximo da implementação. Existem diversas metodologias orientadas a agentes como Tropos [Cossentino, 2005] e Prometheus [Padgham; Winikoff, 2004].

No desenvolvimento de SMA, ao se utilizar de uma metodologia existente para modelar um sistema orientado a agentes e implementá-lo usando uma linguagem de programação específica, de forma não automática, uma série de problemas podem ser gerados relacionados à engenharia de software, como a lentidão no processo de desenvolvimento dos produtos de software; a não confiabilidade na transformação da modelagem para o código-fonte; e o desenvolvimento passíveis de erros humanos.

Portanto, para auxiliar na análise e desenvolvimento de SMA, existem diversas plataformas de desenvolvimentos que geram codificação automatizada a partir de uma linguagem de modelagem específica para determinada linguagem de programação orientada a agente, como o *Prometheus Development Toolkit* (PDT) [Sun; Padgham, 2010], *INGENIAS Development Kit* (IDK) [Gomez-Sanz et al., 2008] e *PASSI Toolkit* (PTK) [Cossentino; Potts, 2002]. Contudo, tais plataformas geram um atrelamento entre a linguagem de modelagem e a linguagem de programação, limitando o desenvolvedor na escolha da metodologia e linguagem para o desenvolvimento do SMA. E ainda, algumas tecnologias não permitem a agregação de novas funcionalidades, por não utilizarem arquiteturas reutilizáveis.

Para superar os problemas de geradores de código para linguagens de modelagens específicas é importante a utilização de meta-modelos. A Arquitetura Orientada a Modelos (*Model Driven Architecture* - MDA) é uma arquitetura que permite a criação de soluções de software utilizando meta-modelos que separam a especificação independente de modelo computacional da implementação em plataformas específicas [Mellor, 2004].

O objetivo desse trabalho é a criação de uma metodologia para desenvolvimento de SMA que utiliza a abordagem de desenvolvimento orientado a modelos para a geração de código semi-automático para as linguagens de programação orientada a agentes Jason [Bordini et al., 2007] e o modelo organizacional Moise+ [Hubner et al., 2002]. A abordagem utiliza o *Fame Agent-oriented Modeling Language* (FAML), como o modelo independente de plataforma e o modelo baseado no JaCaMo [Boissier et al., 2011], formado pelas linguagens Jason, Moise+ e Cartago como modelo específico de plataforma. O FAML é um meta-modelo genérico que reúne os conceitos das linguagens de modelagens mais utilizadas na especificação de SMA,

A abordagem proposta permitirá a utilização de uma linguagem de modelagem que seja aderente ao FAML, e através de um conjunto de transformações na especificação *Query-View-Transformation* (QVT), realizar a transição da especificação para a plataforma específica de desenvolvimento SMA. Após o modelo da plataforma específica estiver instanciado, um conjunto de transformações realizará a transição da plataforma específica de desenvolvimento SMA para a codificação semi-automática da plataforma JASON/Moise+ usando a especificação *Model-To-Text* (M2T). A abordagem ainda utilizará a linguagem *Object Constraint Language* (OCL) da OMG, para restrição

e validação dos modelos FAML e JaCaMo. Este artigo está estruturado da seguinte forma: a seção 2 apresentará a metodologia proposta; a seção 3 será apresentado um exemplo completo; na seção 4 estarão os trabalhos relacionados; e por último a seção 5 apresenta a conclusão.

2. Metodologia Proposta

Esta seção apresenta a extensão da metodologia de desenvolvimento de SMA proposta por Pantoja e Choren (2012) que utiliza a Arquitetura Orientada a Modelos e divide o desenvolvimento de um SMA em diferentes níveis de abstração partindo de um modelo de especificação para uma plataforma específica de desenvolvimento através de um conjunto de transformações. A figura 1 ilustra os níveis de abstração propostos na metodologia.

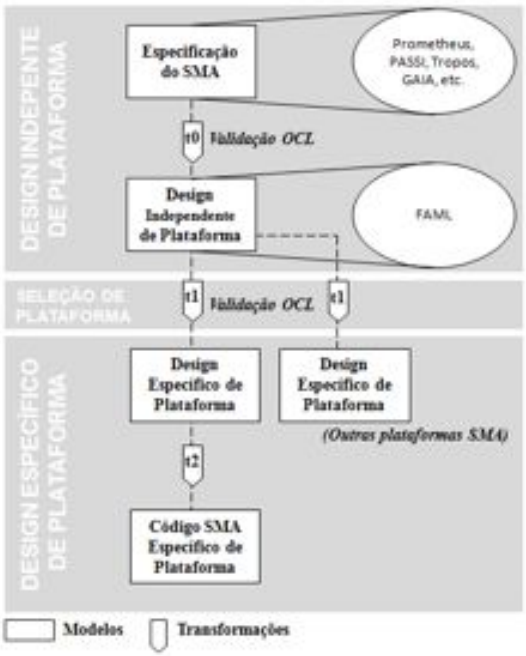

Figura 1. A metodologia proposta.

O nível de *design* independente de plataforma descreve uma aplicação de SMA das perspectivas tanto do nível interno (agente), quanto do nível externo (organização). O modelo de destino consiste da implementação de conceitos do FAML [Beydoun et al., 2009], derivados do modelo de especificação, onde, um meta-modelo, composto dos níveis de ambiente, agente, sistema e definição do agente, foi construído. O FAML é um meta-modelo genérico para desenvolvimento e especificação de SMA, que combina diferentes linguagens de modelagem orientadas a agentes dentro do mesmo domínio da engenharia de software. A intenção é que o FAML forneça um conjunto de conceitos genéricos úteis para qualquer linguagem de modelagem orientada a agentes e metodologias, e.g., TAO, Islander, Adelfe, PASSI, Gaia, INGENIAS e Tropos.

O FAML permite a utilização de diferentes linguagens de modelagem orientadas a agentes, adicionando flexibilidade à especificação e, consequentemente, ao desenvolvimento do SMA. No entanto, para que a modelagem adequada seja feita, um conjunto

de conceitos devem ser instanciados para que a metodologia funcione. Os conceitos de modelo, sistema, ambiente e agentes, devem estar presentes na modelagem. O meta-modelo, baseado no FAML, pode ser visto na figura 2.

As transformações QVT, representadas por *t1* na figura 1 recebem como entrada o modelo FAML instanciado e retornam o modelo JaCaMo instanciado de acordo com o modelo de especificação do SMA. O meta-modelo JaCaMo adaptado, utilizado como modelo específico de plataforma na metodologia, reúne os conceitos tanto da dimensão da programação do agente quanto da programação organizacional, utilizando o Jason e Moise+. Após a geração do modelo JaCaMo, este pode ser editado antes da transformação *t2*.

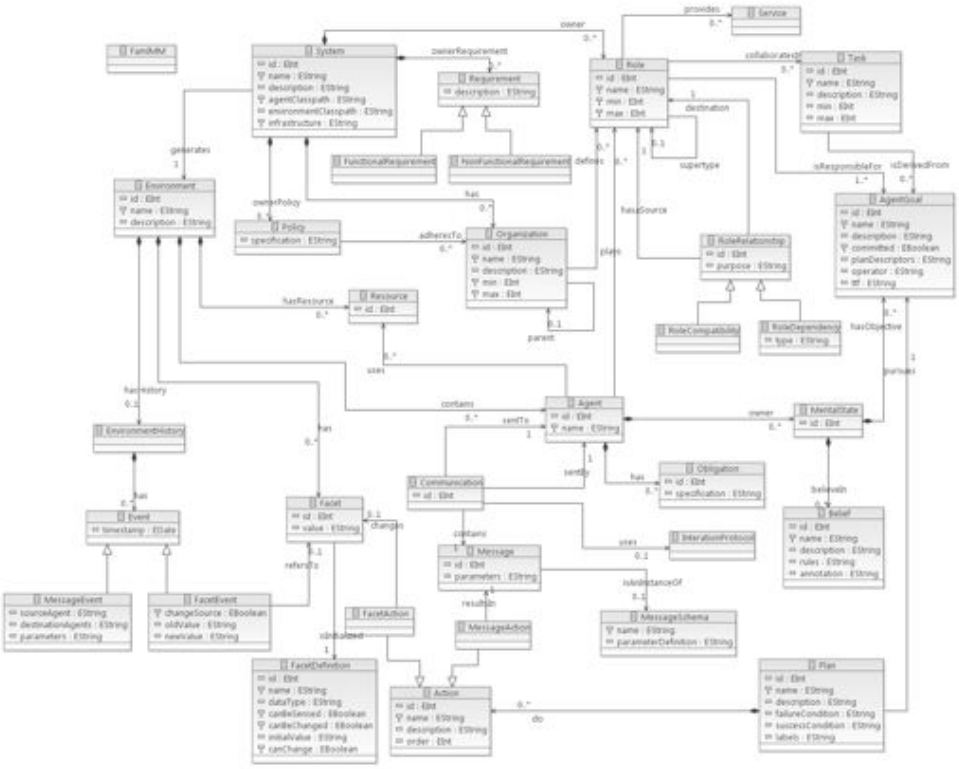

Figura 2. O meta-modelo FAML.

A metodologia proposta neste artigo utiliza a dimensão organizacional, através do modelo organizacional Moise+, e a dimensão do agente, através da linguagem de programação orientada a agentes Jason. A dimensão ambiental utiliza a programação em Java padrão do Jason para a codificação dos ambientes onde os agentes estarão inseridos. Portanto, algumas modificações foram realizadas no meta-modelo para adaptá-lo a metodologia.

As modificações realizadas no meta-modelo adaptado deste trabalho consistem na simplificação da dimensão ambiental e da extensão da dimensão organizacional do meta-modelo inicial do JaCaMo. Na dimensão ambiental do JaCaMo foi necessário simplificar o modelo, já que a metodologia proposta utiliza apenas as classes Java para

implementação do ambiente do agente, não fazendo o uso de artefatos. A simplificação consistiu em manter a classe *Environment*, para representar o ambiente, e a classe *Percept* pra representação das percepções. Desta forma, não se faz necessário a utilização do CArtAgO

Na dimensão organizacional uma extensão de relacionamentos entre as classes são inseridas de forma que a codificação do arquivo organizacional do Moise+ seja estruturada levando em consideração as especificações funcionais, estruturais e normativas. A classe *Link* foi adicionada ao meta-modelo para representar as relações entre grupos e papéis da especificação de grupos do modelo estrutural. Um auto-relacionamento nomeado *supertype* foi adicionado a classe *Role* de forma que a hierarquia de papéis possa ser representada. Um auto-relacionamento, nomeado de *monitoringScheme* foi adicionado na classe *Scheme* de forma que possa ser identificado o esquema de monitoração de cada esquema. Outro auto-relacionamento nomeado *subGoal* foi adicionado a classe *Goal* com o objetivo de identificar a hierarquia de metas da especificação funcional. O meta-modelo proposto da metodologia pode ser visto na figura 3.

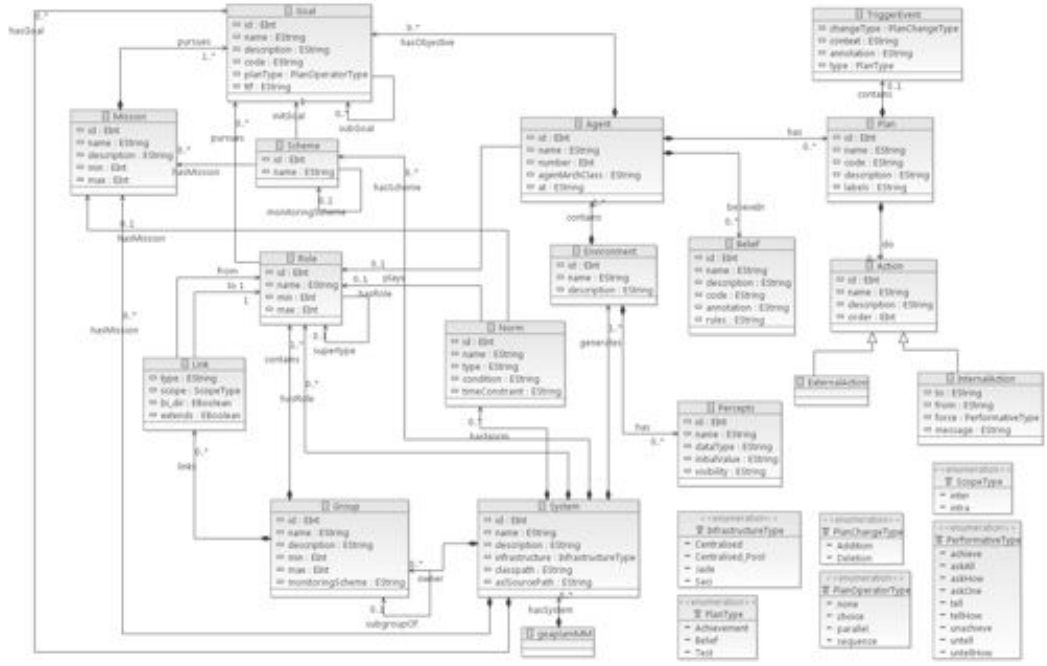

Figura 3. A meta-modelo JaCaMo adaptado.

A transformação do modelo de projeto para a plataforma de execução consiste de um conjunto de transformações entre o modelo JaCaMo e a linguagem de programação Jason e Moise+. As transformações são implementadas na linguagem M2T para geração de artefatos de textos, que utiliza *templates* e modelos. As transformações, representadas por *t2* (figura 1), são um conjunto de *templates* que geram as especificações funcional, estrutural e normativa do Moise+ em conjunto com a codificação Jason dos agentes, ambiente e sistema. A transformação *t2* tem como entrada o modelo instanciado do JaCaMo resultante da transformações *t1*.

A transformação *t2* ainda apresenta uma extensão para o modelo *Unmanned Aerial Vehicle using AgentSpeak* (UAVAS) [Hama et al., 2011], que é uma extensão do Jason para Veículos Aéreos Não Tripulados (VANT) utilizados em simulações e *hardware*. A plataforma UAVAS utiliza o Jason para a programação dos agentes que serão utilizados nos VANT. Porém os conceitos de comunicação são diferentes na plataforma. O envio e o recebimento de mensagens são feitos através de formatos diferentes. A comunicação do Jason normal utiliza para envio de mensagens as ações internas *.send* e *.broadcast*, enquanto que o Jason da plataforma UAVAS utiliza as ações externas *request*, *inform*, *ask* e *ack*.

Dessa forma, se faz necessário, para atender os conceitos de comunicação para o modelo UAVAS, estender as transformações *t2*. Quando o sistema for identificado como Jason, a comunicação levará em conta os conceitos *.send* e *.broadcast*, caso contrário, e este for identificado como UAVAS, as informações de envio de mensagens serão realizadas utilizando através do *request*, *inform*, *ask* e *ack*.

2.1 Discussão

A metodologia proposta permite ao desenvolvedor algumas facilidades no processo de desenvolvimento de *software*. Ao utilizar a abordagem MDA, a rastreabilidade entre o código gerado e o sistema modelado pode ser garantido já que a transformação do modelo independente de plataforma e o modelo de plataforma de execução garante que os conceitos presentes na modelagem serão formalmente transformados para os conceitos da linguagem de programação.

Da mesma forma, a codificação gerada estará em conformidade com o sistema modelado, pois a automatização do desenvolvimento de software utilizando o processo MDA, não permite que os conceitos sejam confundidos ou interpretados de forma errada pelo desenvolvedor que adotar um processo não-automatizado direto da especificação para a plataforma de execução, que em contrapartida apresentam um ciclo de desenvolvimento mais rápido por não precisarem de uma série de transformações entre seus modelos.

A transformação entre o modelo de especificação e o modelo de projeto é realizado por um conjunto de transformações QVT, que é uma linguagem para transformações entre modelos padronizada da OMG com base na OCL. A transformação entre o modelo de projeto e a plataforma de execução utiliza o M2T, que é uma especificação para geração de artefatos de texto e também é padronizada pela OMG. A metodologia ainda prevê validações dos modelos independentes e específicos de plataforma através de regras para restringir os modelos utilizando a OCL, de forma que as instâncias dos modelos sejam o mais consistente possível durante o processo de transição entre os modelos.

3. Prova de Conceito

Esta seção apresenta um exemplo da metodologia proposta, utilizando as linguagens de modelagem orientada a agentes Prometheus. Um ambiente de desenvolvimento, que consiste de *plugins* para o ambiente de desenvolvimento *Eclipse* foi implementada utilizando a ferramenta *Ecore* do *Eclipse Modeling Framework* (EMF) para implementação dos meta-modelos FAML e JaCaMo; as transformações QVT foram implementadas

utilizando o M2M do EMF; e os *templates* em M2T foram implementados utilizando o gerador de artefatos de textos *Acceleo*.

O sistema *Gold Miners* é um SMA no qual quatro agentes procuram ouro dentro de um ambiente com obstáculos e eles precisam se comunicar para coletar o maior número possível de peças de ouro e levá-las até um depósito central. O SMA possui dois tipos de agentes: *Leader* e *Miner*. O agente *Leader* é responsável por liderar, orientar e designar aos outros agentes as tarefas de coleta das peças de ouro através das metas *allocate quadrant* e *allocate gold*. A mensagem com a informação do quadrante *quadrant(X1,Y1,X2,Y2)* é enviada ao agente *Miner* após este enviar sua posição inicial *init_pos(X,Y)*. Da mesma forma que a mensagem *allocated(Gold, Ag)* é enviada ao agente *Miner*, quando este avisa através da mensagem *gold(X,Y)* que encontrou uma peça de ouro e negocia o direito de recolher uma determinada peça enviando a mensagem *bid(VI)*.

O agente *Miner* informa as posições das peças de ouros encontradas para os demais agentes no sistema e negocia com o agente *Leader* a possibilidade de coletar determinada pela de ouro dependendo de fatores como distância e disponibilidade. Além disso, o agente *Miner* ainda possui a meta *handle gold*, no qual as ações *pick*, *drop* e *move*, que são responsáveis pela movimentação do agente, são executadas dependendo das crenças *carrying gold*, *cell(X,Y,gold)*, *free* e *gold_location*; a meta *scan gold* onde através da crença de posicionamento *pos(X,Y)* e *last checked* executa a ação *move*; a meta *inform location* onde a localização é enviada ao agente *Leader*. As mensagens *committed_to(Gold)* e *picked(Gold)* são enviadas aos outros agentes *Miner*.

A figura 4 representa o diagrama do agente *Leader*, onde três mensagens são recebidas e duas enviadas ao agente *Miner*; uma crença e duas metas para serem atingidas podem ser observadas.

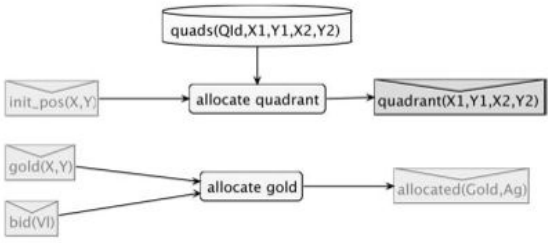

Figura 4. A modelagem do agente *Leader*.

A figura 5 representa o diagrama do agente *Miner*, onde duas mensagens são recebidas do agente *Leader* e três recebidas do agente *Miner*; cinco mensagens enviadas; crenças, metas e um conjunto de ações para atingir as metas de projeto.
A comunicação entre os agentes do sistema é representada através do diagrama de atividades, que mostra a troca de mensagens entre os agentes *Miner* e *Leader*. O agente *Miner* envia a mensagem *committed_to(Gold)* em resposta a mensagem recebida *quadrant(X1,Y1,X2,Y2)*. O agente *Leader* envia a mensagem *quadrant(X1,Y1,X2,Y2)* em resposta a mensagem *init_pos(X,Y)*, para informar ao agente *Miner* a posição em que ele deve procurar peças de ouro; e envia a mensagem *allocated(Gold, Ag)* para alocar as peças de ouro a determinado agente, em resposta as mensagens *gold(X,Y)* e *bid(VI)*, que informam ao agente *Leader* o posicionamento das peças de ouro encontradas e os pedi-

dos de poder carregá-las até o depósito. A figura 6 exibe os diagramas de atividades dos agentes.

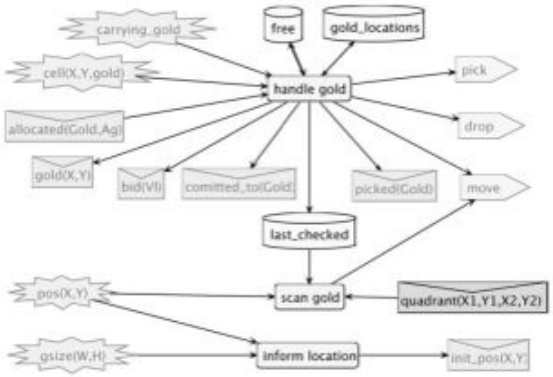

Figura 5. A modelagem do agente *Miner*.

Uma instância do meta-modelo FAML, baseado na modelagem do sistema *Gold Miners*, utilizando a ferramenta *Ecore* pode ser vista na figura 7.

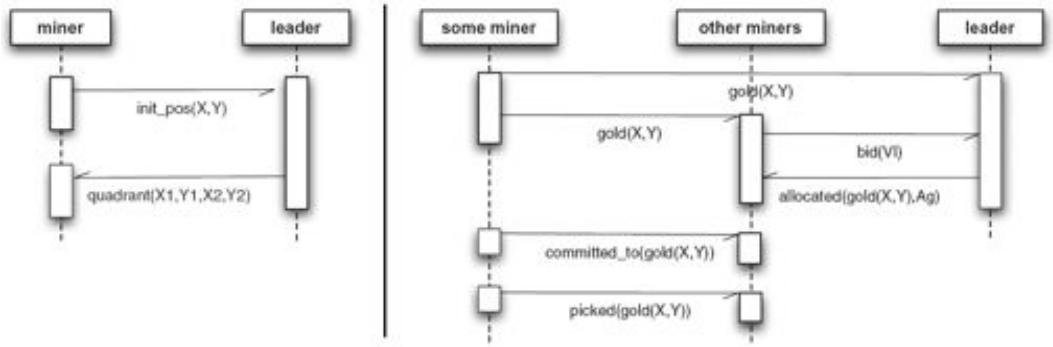

Figura 6. Os diagramas de atividades dos agentes.

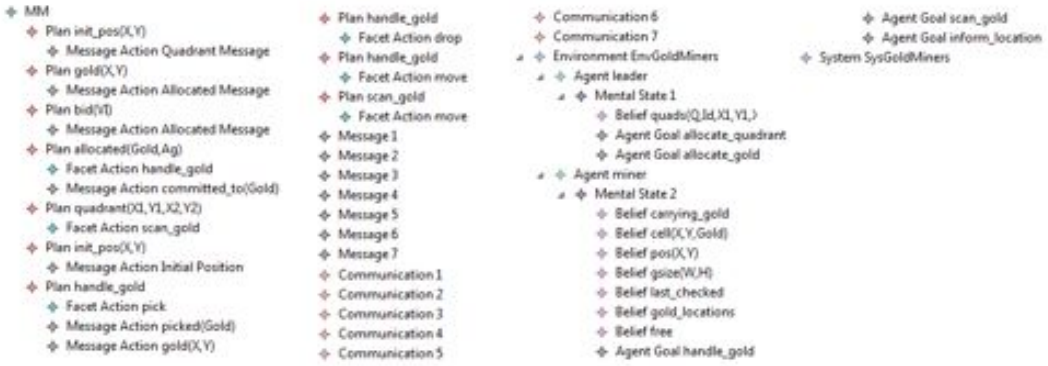

Figura 7. A instância do meta-modelo FAML no Ecore.

Após as transformações QVT serem executadas, a instância do modelo JaCaMo é gerada e as transformações M2T são executadas para que a codificação semi- automática JASON/Moise+ seja gerado. A figura 8 apresenta o código JASON gerado pela ferramenta.

Figura 8. O código Jason gerado pela ferramenta.

4. Trabalhos Relacionados

Nesta seção serão apresentados alguns trabalhos relacionados que abordam a geração de código automático utilizando a orientação a agentes. O INGENIAS *Development Kit* (IDK) é uma ferramenta para análise, design e codificação de SMA que gera codificação automática para a linguagem de programação orientada a agentes Jade, usando a metodologia INGENIAS, partindo de utilização da UML e de uma notação particular. A geração de código para o Jade é realizado através de um mecanismo que utiliza *templates*. O IDK utiliza um editor visual com opções de teste e debug para facilitar o design e a implementação do SMA.

A ferramenta permite ao desenvolvedor modelar o SMA optando pela UML ou pela metodologia INGENIAS. A modelagem é feita a partir da criação de um conjunto de diagramas disponibilizados pela ferramenta como: ambiente, componente, organização, tarefa, atividades e agentes. A geração da codificação em JADE é realizada através do menu Modules onde os *templates* podem ser executados. Apesar do IDK utilizar o MDA para geração de codificação, ele não utiliza um meta-modelo como plataforma e é atrelado ao modelo INGENIAS e gera a codificação apenas a linguagem de programação JADE.

O Prometheus Development Toolkit (PDT) é uma ferramenta que suporta o desenvolvimento de SMA, utilizando a metodologia Prometheus, que oferece

facilidades para geração de codificação automática para a linguagem de programação orientada a agentes Jack. A ferramenta permite a integração com outras linguagens de modelagem a fim de facilitar o desenvolvimento do SMA. O PDT utiliza o EMF e o Ecore para o desenvolvimento de um meta-modelo chamado Prometheus Ecore Meta-Model (PEMM) e o GMF para a criação de uma ferramenta gráfica baseada na metodologia Prometheus. A tecnologia Java Emitter Templates é utilizada para a geração intermediária de código.

Contudo o PDT somente permite a geração de código para a plataforma Jack, atrelando a ferramenta a uma plataforma de execução. Se for necessária a integração de outra linguagem de modelagem ao PDT, é obrigatório a implementação de meta-modelos e regras de transformações entre eles. A ferramenta PDT não utiliza a abordagem MDA, tendo apenas um meta-modelo específico para sua metodologia, o PEMM.

O PASSI Toolkit é uma ferramenta que propõe um refinamento passo a passo desde os requisitos até a codificação, compilando os diagramas da metodologia PASSI para codificação Jack. Por não usar a abordagem MDA, a responsabilidade da geração do código é da ferramenta AgentFactory. A ferramenta PASSI é um plug-in para o programa de modelagem Rational Rose e não utiliza um meta-modelo orientado a agentes fazendo com que o código seja gerado direto do modelo para o código, além de não estar preparada para aceitar outras linguagens de modelagens.

A metodologia para desenvolvimento de SMA proposta neste trabalho utiliza um meta-modelo genérico, o FAML, que possibilita ao desenvolvedor escolher a linguagem de modelagem de SMA que for mais apropriada as suas necessidades, visto que o meta-modelo reúne os conceitos das principais linguagens de modelagem e metodologias orientada a agentes em um único modelo. Além disso, permite a integração das linguagens de modelagens e metodologias que sejam aderentes ao FAML sem necessidade de implementação de novas regras de transformações, já que o núcleo da transformação PIM para PSM já está definido.

Da mesma forma, a metodologia permite também a geração de codificação semi-automática para outras linguagens de programação, além do Jason e do Moise+ utilizadas atualmente, pois utiliza a abordagem da MDA, dividindo os diversos níveis de abstração do desenvolvimento do sistema em modelos independentes entre si. Essa codificação é feita partindo do FAML, passando por um conjunto de transformações até chegar à plataforma de execução do SMA. A metodologia utiliza a linguagem de programação Jason e o modelo organizacional Moise+, que utilizam a arquitetura BDI, para lidar com conceitos de organizacionais e ambientais.

A metodologia não possui uma ferramenta gráfica específica para as metodologias aderentes ao FAML, porém a metodologia proposta permite a integração das ferramentas já existentes ao meta-modelo adaptado do FAML através de transformações entre modelos e a criação de ambientes próprios utilizando o GMF. A tabela 1 apresenta um comparativo entre a metodologia proposta e as ferramentas existentes na atualidade.

Tabela 1. Comparativo das ferramentas.

Conceito	IDK	PDT	PASSI	Metodologia Proposta
Permite codificações em outras linguagens	NÃO	NÃO	NÃO	SIM
Utilização de templates	SIM	NÃO	NÃO	SIM
Linguagens de Modelagens diferentes	NÃO	SIM	NÃO	SIM
Ambiente Gráfico	SIM	SIM	SIM	NÃO
Abordagem MDA	NÃO	SIM	NÃO	SIM

5. Conclusão

Este trabalho desenvolveu uma metodologia MDA para geração de SMA a partir de linguagens e metodologias aderentes ao meta-modelo FAML. O FAML foi utilizado como PIM pois ele reúne os conceitos de diversas metodologias orientadas a agentes em um único meta-modelo. Como PSM foi utilizada a plataforma JaCaMo, que integra as linguagens JASON, Moise+ e CArtAgO. Para permitir as transformações entre os modelos, foi necessário desenvolver um meta-modelo baseado nos conceitos do JaCaMo, onde a dimensão do agente e da organização tem uma importância maior, visto que o modelo da plataforma de execução não precisa de artefatos para o seu funcionamento.

As regras de transformações entre o modelo de especificação e o modelo de projeto foram desenvolvidas utilizando a linguagem QVT padronizada pela OMG. Em seguida, um conjunto de regras de transformações entre o modelo de projeto e a plataforma de execução foi desenvolvido utilizando a linguagem M2T da OMG. A M2T foi utilizada por ser uma linguagem que utiliza *templates* para geração de artefatos de textos, que faz a geração parcial do código do SMA modelado a partir da instância do meta-modelo gerado pelas transformações M2T.

Foi criado um ambiente de desenvolvimento a agentes para o Eclipse, onde os meta-modelos da metodologia foram construídos utilizando-se a ferramenta Ecore integrante do *Eclipse Modeling Framework* (EMF). As transformações em QVT foram implementadas utilizando-se o M2M e as transformações em M2T foram implementadas utilizando o *Acceleo*, que é um gerador de artefatos de textos para o Eclipse. Um exemplo utilizando a metodologia e o ambiente desenvolvido foi apresentado: o sistema *Gold Miners* que foi modelado na linguagem de modelagem Prometheus.

Apesar da metodologia não possuir uma parte de modelagem gráfica, a sua utilização facilita o processo de desenvolvimento do sistema já que permite ao desenvolvedor do sistema optar pela metodologia orientada a agentes que mais lhe for familiar, desde que esta seja aderente ao meta-modelo FAML. A metodologia garante a rastreabilidade dos conceitos gerados e dos conceitos modelados, por utilizar um conjunto de transformações entre modelos conforme a arquitetura MDA, garantindo assim, que a codificação gerada está em conformidade com o sistema modelado. Da mesma maneira, o processo de gestão de mudanças no software é facilitado, de forma que o desenvolvedor invista na modelagem do sistema para atingir a modificação necessária no código.

Como trabalhos futuros, o desenvolvimento de um ambiente gráfico para modelagem de SMA com as metodologias aderentes ao FAML permitirá uma versatilidade e aplicabilidade maior à ferramenta do ponto de vista do desenvolvedor. Além disso, um ambiente gráfico permitirá ao desenvolvedor utilizar diagramas de modelagens diferentes, pois essas estarão unificadas graficamente em uma mesma ferramenta. Também será

desenvolvida a geração de codificação automática para outras plataformas específicas como o Jade/Jadex.

6. Referências Bibliográficas

Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomezsanz, J. J., Pavon, J. e Gonzalez-Perez, C. (2009) "FAML: a generic metamodel for MAS development." IEEE Trans. Softw. Eng.

Boissier, O., Bordini, R. H., Hubner, J. F., Ricci, A. e Santi, A. (2011) "Multi-agent oriented programming with jacamo" Science of Computer Programming.

Bordini, R. H., Hubner, J. F. e Wooldridge, W. (2007) "Programming Multi-Agent Systems in AgentSpeak using Jason" Jonh Wiley and Sons, London.

Cossentino, M. (2005) "From Requirements to Code with the PASSI Methodology", Em SELLERS, H. B. e GIORGINI, P., editores, Agent-Oriented Methodologies, volume 3690 of LNCS, p_ags. 79{106. Idea Group Pub.

Cossentino, M. e Potts, C. (2002) "A CASE tool supported methodology for the design of multi-agent systems".

Gomez-Sanz, J. J., Fuentes, R., Pavon, J. e Garcia-Magarino, I. (2008) "Ingenias development kit: a visual multi-agent system development environment", In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: demo papers, AAMAS '08, Richland, SC.

Hama, M. T. ; Allgayer R. S. ; Pereira, C. E. ; Bordini, R. H. (2011) "UAVAS: An Agent Oriented Infrastructure for Unmanned Aerial Vehicles Development" In: AutoSoft@CBSoft, 2011, São Paulo. II Workshop sobre Sistemas de Software Autônomos. São Paulo : CBSoft, v. 10. p. 15-21.

Hubner, J. F., Sichman, J. S. A. e Boissier, O. (2002) "A model for the structural, functional, and deontic specification of organizations in multiagent systems", In Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence, SBIA '02, London, UK. Springer-Verlag.

Mellor, S. (2004) "Mda Distilled: Principles of Model-Driven Architecture", Addison Wesley Object Technology Series, Addison-Wesley.

Padgham, L. e Winikoff, M. (2004) "Developing Intelligent Agent Systems: A Practical Guide", Wiley Series in Agent Technology, John Wiley.

Pantoja, C. E. e Choren, R. (2012) "A mda approach for agent-oriented development using faml." In ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems, Volume 2, Wroclaw, Poland, 28 June - 1 July, 2012, SciTePress.

Sun, H., Thangarajah, J. e Padgham, L. (2010) "Eclipse-based prometheus design tool" In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, AAMAS '10, Richland, SC.

Wooldridge, M. (2000) "Reasoning about rational agents", Intelligent robotics and autonomous agents, MIT Press.