Checagem de Consistência de Modelos de Sistemas Multiagentes Normativos Utilizando MAS-ML *Tool*

Igor B. Nogueira[1], Mariela I. Cortés[1], Enyo J. T. Gonçalves[2]

[1]Universidade Estadual do Ceará, Fortaleza– Brasil

[2]Universidade Federal do Ceará

igor.bnog@gmail.com, mariela@larces.uece.br, enyo@ufc.br

Abstract. *Modeling tools are important allies in the development of complex software systems. The accuracy and detail may vary, but the models need to be correct and precise. This paper presents the evolution of the modeling tool for multi-agent systems MAS- ML tool in order to provide support for checking consistency of the internal properties and dependencies between diagrams.*

Resumo. *Ferramentas de modelagem são aliadas importantes no desenvolvimento de sistemas de software complexos. O detalhe e o rigor podem variar, mas em qualquer caso, os modelos precisam ser corretos e precisos. Este artigo apresenta a evolução da ferramenta de modelagem para sistemas multiagentes MAS-ML tool de forma que forneça apoio à checagem de consistência das propriedades internas e dependências entre diagramas.*

1. Introdução

A modelagem de sistemas é o processo de desenvolvimento de modelos abstratos em que cada modelo representa uma visão diferenciada do sistema [Sommerville 2011]. Os modelos precisam ser corretos no uso da notação e a consistência entre as diferentes visões precisa ser garantida. No entanto, analisar e estabelecer a boa formação de diagramas é uma tarefa difícil, principalmente no caso de sistemas complexos.

Uma linguagem de modelagem normalmente fornece um conjunto de diagramas através dos quais diferentes visões do sistema podem ser capturadas. Inconsistências inter-diagramas são definidas como resultantes da violação às propriedades de interdependência entre modelos. Por outro lado, uma inconsistência intra-diagrama está relacionada a violações a propriedades internas de um diagrama analisado isoladamente.

O método Observed-MAS [Brandão 2005] propõe a análise sistemática de modelos de SMAs descritos na versão original de MAS-ML através do uso de ontologias. Consequentemente, o método não suporta a versão mais atual da linguagem. O presente artigo aborda a implementação de um mecanismo que possibilite a checagem de propriedades estruturais e detecção de inconsistências intra e inter-diagramas utilizando a ferramenta MAS-ML Tool contemplando os diagramas definidos em MAS-ML 3.0.

2. Consistência de Modelos

O gerenciamento da consistência entre modelos envolve um conjunto de métodos e ferramentas que possibilitem estabelecer e manter a consistência entre diversos artefatos criados e utilizados por múltiplos *stakeholders* [Spanoudakis e Zisman 2001], [Kuster 2004].

3. Evolução de MAS-ML *tool*

3.1. Consolidação da Ferramenta

MAS-ML *tool* [Farias et al. 2009], [Gonçalves et al 2015] [Feijó 2012] [Freire et al. 2012] [Lima et al. 2013] é uma ferramenta que funciona como um *plug-in* da plataforma Eclipse[1] e serve como ambiente de modelagem para SMAs. O desenvolvimento da ferramenta ocorreu de forma gradativa onde, para cada diagrama foi desenvolvido um componente ou plugin independente, o que dificulta a utilização da ferramenta para a modelagem e análise global do sistema.

A ferramenta foi gerada a partir dos *plug-ins* GMF[2] (*Graphical Modeling Framework*) e EuGENia[3], responsável por automatizar e facilitar os procedimentos necessários para o desenvolvimento de diagramas utilizando o GMF. A estratégia adotada para a implementação do plug-in da ferramenta segue a abordagem orientada por modelos a partir do metamodelo consolidado em MAS-ML 3.0.

3.2. Checagem de Consistência de Modelo

A checagem de consistência em MAS-ML *tool* integrada dá-se por meio de verificações a nível interno de cada modelo através da checagem intra-diagrama, e por meio de verificação que leva em consideração a interdependência entre os modelos analisados, através da checagem inter-diagramas.

A checagem intra-diagrama é baseada nas regras de validação considerando as propriedades internas do modelo especificadas em MAS-ML 3.0 (Tabela 1).

Tabela 1 – Regras de validação intra-diagrama em MAS-ML 3.0.

Regra	Descrição	Implementação em OCL
Regra 1	Se um agente possui percepção, uma norma não pode restringi-la.	(self.resourceOfNorm->notEmpty() = true and self.normRestrict->notEmpty() = true and self.normRestrict.agentClass->notEmpty() = true and self.resourceOfNorm.feature->notEmpty() = true and self.resourceOfNorm.feature.oclIsTypeOf(Perception) = true and self.normRestrict.agentClass.ownedPerception->exists(perc | perc.name = self.resourceOfNorm.feature.name) = true) implies false
Regra 2	Se um agente possui planejamento, uma norma não pode restringi-la.	(self.resourceOfNorm->notEmpty() = true and self.normRestrict->notEmpty() = true and self.normRestrict.agentClass->notEmpty() = true and self.resourceOfNorm.feature->notEmpty() = true and self.resourceOfNorm.feature.oclIsTypeOf(Planning) = true and self.normRestrict.agentClass.ownedPlanning->exists(plann | plann.name = self.resourceOfNorm.feature.name) = true) implies false
Regra 3	Se um agente possui função-próximo, uma norma não pode restringi-la.	(self.resourceOfNorm->notEmpty() = true and self.normRestrict->notEmpty() = true and self.normRestrict.agentClass->notEmpty() = true and self.normRestrict.agentClass.owendAction->notEmpty() = true and self.normRestrict.agentClass.owendAction->exists(a | a.actionSemantics <> ActionSemantics::DefaultSemantics) = true) implies false
Regra 4	Se um agente possui função de formulação de objetivo, uma norma não pode restringi-la.	
Regra 5	Se um agente possui função de formulação de problema, uma norma não pode restringi-la.	
Regra 6	Se um agente possui função utilidade, uma norma não pode restringi-la.	

[1] http:// www.eclipse.org
[2] http://eclipse.org/gmf-tooling/
[3] http://www.eclipse.org/epsilon/doc/articles/eugenia-patching/

Por outro lado, a checagem de consistência inter-diagramas em MAS-ML *tool* proposta se baseia na definição de propriedades inter-diagramas, isto é, propriedades de interdependência entre diagramas. A Tabela 2 apresenta as propriedades de interdependência entre diagramas estáticos de MAS-ML, incluindo o diagrama de normas e de acordo com o método Observed-MAS [Brandão 2005].

Tabela 2 – Propriedades de Interdependência entre Diagramas Estáticos

Diagrama 1	Diagrama 2	Regra
Classes	Organização	Toda classe de agente ou de organização modelada num diagrama de classes precisa, necessariamente, estar modelada em algum diagrama de organização.
Papel	Organização	Toda classe de papel de agente ou de papel de objeto definida num diagrama de papel precisa estar definida em algum diagrama de organização.
	Classes	Toda classe de objeto modelada num diagrama de papel precisa estar modelada em algum diagrama de classes.
Classes	Normas	Toda classe de objeto, agente ou de organização modelada num diagrama de classes precisa, necessariamente, estar modelada em algum diagrama de normas.
Organização		Toda classe de agente, classe de organização, classe de papel de agente ou de papel de objeto definida num diagrama de organização precisa estar definida em algum diagrama de normas.
Papel		Toda classe de papel de agente ou de papel de objeto definida num diagrama de papel precisa estar definida em algum diagrama de normas.

3.3. Verificador Inter-Diagramas

A fim de fazer a checagem de consistência entre modelos, um algoritmo foi proposto a fim de categorizar os diagramas e verificar suas propriedades internas. Tomando as diferenças entre os diagramas estáticos de MAS-ML 3.0, percebe-se que o diagrama de normas é o mais abrangente, em relação a entidades e relacionamentos contemplados, seguido do diagrama de organização, papel e classes, respectivamente.

Seguindo a ordem de precedência dos diagramas e suas propriedades, o mecanismo de categorização considera a seguinte lógica: (i) modelo que possui, no mínimo, uma entidade norma definida em sua estrutura é categorizado como diagrama de normas; (ii) modelo que possui, no mínimo, uma entidade organização é categorizado como diagrama de organização; (iii) modelo que possui, no mínimo, um papel de agente ou papel de objeto sem organização e ambiente definidos em sua estrutura é categorizado como diagramas de papel e (iv) modelo que não atende às condições anteriores é categorizado como diagrama de classes. Com os modelos categorizados e, a partir das informações contidas nos arquivos dos diagramas projetados em MAS-ML *tool* (.masml), as propriedades de interdependência entre modelos podem ser verificadas (Algoritmo 1).

O mecanismo de categorização e a definição das propriedades inter-diagramas foram implementados em Java e a manipulação dos arquivos .masml foi facilitada através do JDOM [JDOM, 2015].

Algoritmo 1 – Interdependência entre Diagramas de Classes e Normas

```
Boolean containElement = true;
String inconsistencyWarning = "Rule 1: Inconsistency Class-Norm Diagrams - ";
String elementValue = "";
String valueElement = "";
switch(type){
   case ClassDiagram:
      if(type2 == Diagram.Type.NormDiagram){
         for(Element element2:agentClassChildren2){
             valueElement = element2.getAttributeValue("name");
             element2 = element; element2.setAttribute("name", valueElement);
```

```
            }
            if(!agentClassChildren2.contains(element)){ containElement = false;}
        }
        if(!containElement){ inconsistencyWarning +=  "Agent Not Defined"; }
        containElement = true;
        for(Element element: classChildren){
            for(Element element2: classChildren2){
              valueElement = element2.getAttributeValue("name");
              element2 = element; element2.setAttribute("name", valueElement);
             }
             if(!classChildren2.contains(element)){ containElement = false; }
         }
         if(!containElement){ inconsistencyWarning += "Class Not Defined"; }
         containsElement = true;
         for(Element element: organizationClassChildren){
            for(Element element2: organizationClassChildren2){
                valueElement = element2.getAttributeValue("name");
                element2 = element;
                element2.setAttribute("name", valueElement);
            }
            if(!organizationClassChildren2.contains(element)){ containElement = false;}
          }
          if(!containElement){ inconsistencyWarning += "Organization Not Defined"; }
          containElement = true;
}
```

4. Estudo de Caso

Para ilustrar a checagem de consistência de modelos utilizando a ferramenta MAS-ML *tool* integrada, foi usado TAC-SCM [Collins et al., 2006].

5.1 Identificação das Entidades do Sistema

O ambiente *TACEnvironment* possui uma organização principal *TacOrganizacao* e os agentes Comprador (*BuyerAgent)*, que pode desempenhar o papel de Comprador, Vendedor (*SellerAgent*), que pode desempenhar o papel de Vendedor (Seller), Entregador (*DeliveryAgent*), que pode desempenhar o papel de Entregador (Delivery), o agente fornecedor (*SupplierAgent*), que pode desempenhar o papel de fornecedor, e o agente gerente, que pode desempenhar o papel de gerente.

A Figura 1 (a) ilustra o diagrama de classes, especificando as relações dos agentes entre si e entre os agentes e o ambiente. Os relacionamentos que os papéis possuem entre si podem vistos na Figura 2 (b).

As seguintes normas foram definidas no contexto do TAC-SCM:

- N1: O agente vendedor é proibido de ter acesso aos pagamentos (proibição);
- N2: Todos os vendedores da organização TAC-Organization têm permissão para atualizar o estoque (permissão);
- N3: Os compradores da organização TAC-Organization são obrigados a pagar pelos itens que eles compraram (obrigação);
- N4 (Punição para a violação da N3): Os compradores que violaram N3 são proibidos de comprar itens (proibição).
- N5: O agente comprador é proibido de verificar as atualizações de lançamento de novos lances (proibição).

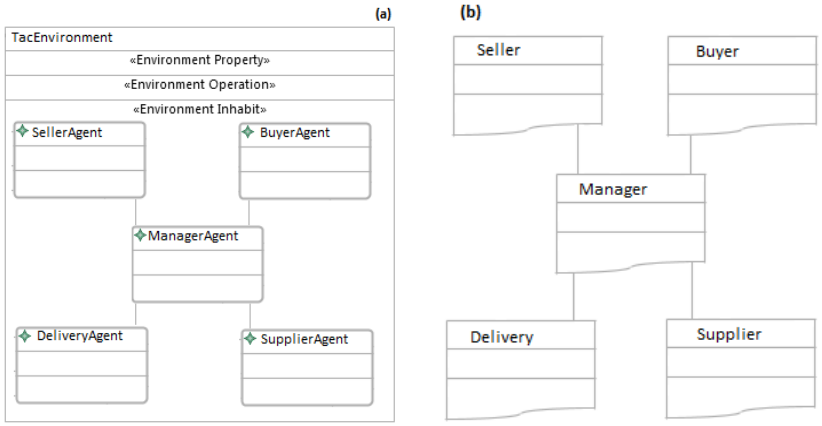

Figura 1. Diagrama de Classes (a) e Diagrama de Papel (b) para o TAC-SCM

A Figura 2 representa o diagrama de normas relacionando as normas ao ambiente juntamente com as entidades e a organização que está no contexto de N1 e N2.

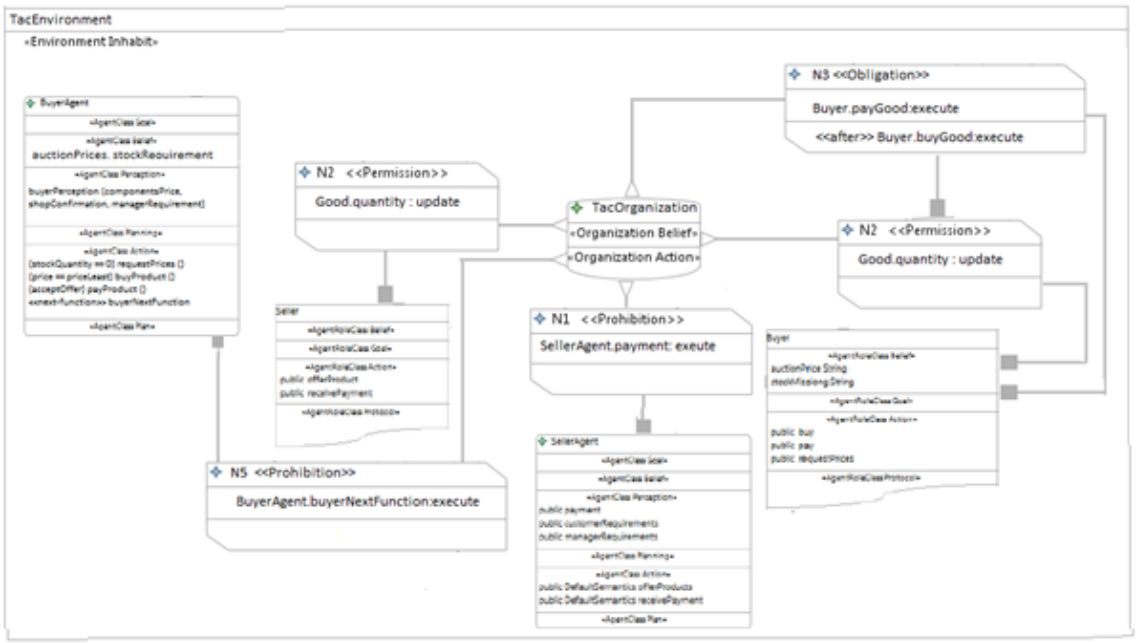

Figura 2. Modelagem do Diagrama de Normas

O processo de verificação de consistência entre os diagramas inicia executando o mecanismo de categorização a partir dos arquivos .masml. No estudo de caso são avaliados o diagrama de classes e o diagrama de normas. A partir da checagem das condições de categorização o tipo de diagrama é estabelecido em cada caso. Com o diagrama-origem (do tipo Diagrama de Classes) e o destino (do tipo Diagrama de Normas) categorizados, a checagem de consistência inter-diagramas pode ser feita.

Considerando que toda classe de objeto, de agente ou de organização modelada no diagrama de classes precisa, necessariamente, estar contemplada no diagrama de normas, uma inconsistência é detectada uma vez que os agentes *SupplierAgent, DeliveryAgent* não foram modelados. Consequentemente a mensagem *"Rule 1: Inconsistency Class-Norm Diagrams - Class Not Defined"* é exibida ao projetista.

6. Conclusão e Trabalhos Futuros

Este artigo apresenta a integração e evolução da ferramenta MAS-ML *tool* propiciando a checagem de consistência entre os modelos de sistemas multiagentes normativos modelados na ferramenta. A integração dos vários plug-ins da ferramenta MAS-ML *tool* seguiu a abordagem orientada por modelos, e na versão consolidada contempla os diagramas definidos em MAS-ML 3.0, a saber: classes, organização, papéis e normas.

A checagem de consistência de modelos em MAS-ML *tool* integrada deu-se por meio de verificações a nível interno de cada modelo através da checagem intra-diagramas baseada em regras OCL, e inter-diagramas com base na definição de propriedades de interdependência. Como trabalhos futuros podemos citar a extensão da ferramenta MAS-ML *tool* para abranger aspectos dinâmicos e estáticos de forma integrada, e incluir a checagem de consistência desses diagramas.

Referências

Brandão, A. A. F. (2005) “Um método para estruturação e análise de modelos de sistemas multiagentes baseado em ontologias”, Tese de doutorado. Rio de Janeiro: PUC, Departamento de Informática.

Collins, J.; Arunachalam, R.; Sadeh, N.; Eriksson, J.; Finne, N.; Janson, S. The Supply Chain Management Game for the 2007 Trading Agent Competition. 2006.

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

Feijó, A. R. (2012). “Evolução da Ferramenta MAS-ML tool para a Modelagem dos Diagramas de Papéis e Sequência”. Monografia. Departamento de Computação, UECE.

Freire, E. S. S.; Cortés, M. I.; Gonçalves, E. J. T.; Lopes, Y. S. (2012) “A Modeling Language for Normative Multi-Agent Systems”. In: 13th AOSE, Valencia (Spain)

Gonçalves, E., Cortés, M., Campos, G., Lopes, Y., Freire, E., Silva, V., Farias, K., Oliveira, M. (2015) "Mas-ml 2.0: Supporting the modelling of multi-agent systems with different agent architectures." Journal of Systems and Software, V. 108, 77-109.

JDOM, disponível em:<http://www.jdom.org/>, acessado em 20/02/2016.

Kuster, J. (2004) “Consistency management of object-oriented behavioral models,” Ph.D. dissertation, Universit¨at Paderborn.

Lima, F. R. O.; Feijó, A. R; Rocha Jr., R. M.; Nogueira, I. B.; Goncalves, E. J. T. ; Freire, E. S. S. ; Cortes, M. I . (2013) “Dynamic Modeling of Multi-Agent Systems Using MAS-ML Tool”. In: VII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC), São Paulo.

Sommerville, I. (2011) “Software Engineering”. 9ª ed. Boston: Pearson.

Spanoudakis, G. and Zisman, A. (2001) “Inconsistency management in software engineering: Survey and open research issues,” in in Handbook of Software Engineering and Knowledge Engineering. World Scientific, pp. 329–380.