Engenharia de Software Orientada a Agentes: Um Estudo Comparativo entre UML e Metodologias que Suportam o Processo de Desenvolvimento de Sistemas Multiagente

Rafhael R. Cunha [1], Diana F. Adamatti[1], Cléo Z. Billa[1]

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Av. Itália km 8 – Bairro Carreiros – Rio Grande - RS - Brasil

{rcrafhaelrc,dianaada,cleo.billa}@gmail.com

Resumo. *Este artigo pertence ao domínio da Agent Oriented Software Engineering (AOSE) e Sistemas MultiAgente (SMA). O trabalho apresenta brevemente as metodologias Prometheus, Tropos, MaSE e Ingenias, elencando suas etapas e características de desenvolvimento. O objetivo deste trabalho é analisar as metodologias exploradas, comparando-as com os artefatos presentes na Unified Modelling Language (UML). Ao final deste trabalho, conclui-se que cada uma das metodologias analisadas possui enfoques distintos, atribuindo ao projetista a tarefa de escolher a que melhor se adeque as suas necessidades.*

1. Introdução

No decorrer dos anos, a área de engenharia de software (ES) tem procurado definir processos de desenvolvimento de software e linguagens de modelagem que visem estabelecer etapas bem definidas para a construção de um software. Este fato tem como propósito tornar a produção de um software mais robusta, rápida, organizada, confiável e de fácil manutenção e reutilização [Guedes 2012].

Na área de inteligência artificial, o paradigma orientado a agentes tem sido pesquisado e utilizado para minimizar a complexidade e aumentar a eficiência de softwares distribuídos [Jayatilleke et al. 2007] [Rodriguez et al. 2011]. Esta prática tem se mostrado eficiente para a construção de softwares com essas características, viabilizando um aumento no desenvolvimento de sistemas multiagente (SMA) [Guedes 2012].

Entretanto, devido à complexidade e distribuição desse tipo de sistema, novos desafios para a área de ES tradicional foram encontrados. Esses desafios ocorreram em virtude de metodologias conceituadas para se fazer modelagem na área, como *Unified Modelling Language* (UML), não suprirem as características presentes neste grupo de sistemas. Essa ausência levou ao surgimento de uma subdivisão da área que mescla conceitos de engenharia de software e inteligência artificial, chamada de *Agent Oriented Software Engineering* (AOSE) ou Engenharia de Software Orientada a Agentes. Seus objetivos principais são propor métodos e linguagens/notações para projetar e modelar softwares orientados a agentes [Guedes 2012].

Dentro desse contexto, diversas metodologias foram propostas buscando suprir a demanda de softwares orientados a agentes [Bergenti et al. 2004]. Essas metodologias foram criadas pelos mais variados motivos. Algumas, basearam-se em melhorias nos diagramas presentes na UML, outras, criaram seus próprios meta-modelos e notações para seu uso.

Este trabalho tem como motivação apresentar uma revisão de algumas metodologias para construção de sistemas multiagente, mostrando as etapas presentes nessas metodologias e os artefatos utilizados para apoiar essas etapas. Por fim, como objetivo principal na realização deste trabalho, pretende-se fazer uma análise comparativa, especificando a aproximação dos artefatos gerados por estas metodologias em relação aos presentes na linguagem UML.

2. Metodologias AOSE

Segundo [Brandão 2014], as metodologias existentes até o presente momento para modelar SMA sobre a perspectiva da engenharia de software são as ilustradas pela figura 1.

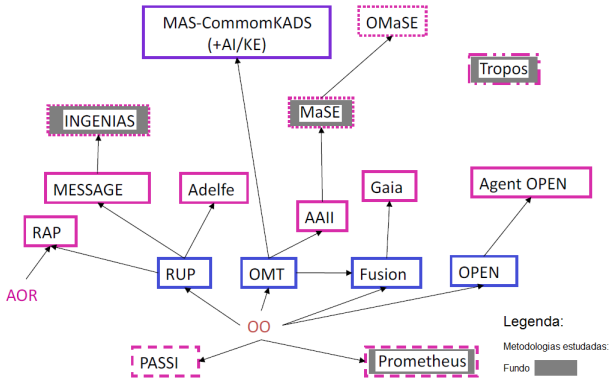

Figura 1. Metodologias AOSE [Brandão 2014]

A escolha das metodologias exploradas neste trabalho baseou-se na seleção de algumas metodologias distintas que compõem ramos heterogêneos da figura 1.

2.1. Prometheus

Segundo [Padgham and Winikoff 2005] e [Khallouf and Winikoff 2009] Prometheus é uma metodologia que consiste em três fases: A fase de especificação do sistema; A fase de projeto arquitetural e a fase de projeto detalhado.

A fase de especificação do sistema é a primeira fase da metodologia Prometheus. Essa fase é composta por duas etapas: Determinar o ambiente do sistema (percepções e ações) e determinar os objetivos e funcionalidades do sistema (objetivos e cenários de casos de uso). De acordo com [Padgham and Winikoff 2002], simultaneamente ao descobrimento ou especificação das percepções e ações, o usuário da metodologia deve começar a descrever quais são as funções do agente em um contexto mais amplo, ou seja, quais são as funcionalidades do sistema.

Na fase de projeto arquitetural, toma-se a principal decisão acerca do sistema [Padgham and Winikoff 2002]. De acordo com [Padgham and Winikoff 2002], é nesta fase que se define quais agentes devem existir no sistema a ser modelado. Todavia, o diagrama crucial desta etapa de desenvolvimento é o diagrama de visão geral do sistema. Neste diagrama são reunidos agentes, eventos e objetos de dados compartilhados.

O objetivo final dessa etapa da metodologia é especificar completamente a interação entre os agentes. Para tanto, são utilizados os diagramas de interação, como

uma ferramenta inicial para modelar essa interação. Para a evolução dessas interações entre os agentes, [Padgham and Winikoff 2005] avançaram essas interações para protocolos de interação, tendo como finalidade definir precisamente quais sequências de interações são válidas dentro do sistema.

De acordo com [Padgham and Winikoff 2002] a fase do projeto detalhado foca no desenvolvimento da estrutura interna de cada um dos agentes e como os mesmos irão realizar suas tarefas dentro do sistema modelado. É nesta fase do projeto que a metodologia se torna específica para agentes baseados na arquitetura BDI. Outro ponto de destaque dessa fase é a definição de capacidades (módulos dentro dos agentes), eventos internos, planos e estruturas de dados detalhados.

[Padgham and Winikoff 2005] citam que a metodologia Prometheus não se baseou na notação presente na UML para a criação de seus diagramas, gerando dois pontos distintos. O ponto positivo consiste na descentralização de agentes em objetos, favorecendo a modelagem desse tipo de sistemas. Sobre o ponto negativo, este é identificado em razão da não utilização de uma notação padrão, o que contrariamente, favoreceria a metodologia visto que uma padronização contribuiria para a sua aceitação.

2.2. Tropos

Segundo [Bresciani et al. 2004], a metodologia Tropos tem como propósito apoiar todas as atividades de análise e projeto do desenvolvimento de software. Tropos consiste de cinco fases distintas no seu processo de desenvolvimento, sendo: Requisitos Iniciais; Requisitos Finais; Projeto Arquitetural; Projeto Detalhado e Implementação.

O objetivo da análise de requisitos em Tropos é fornecer um conjunto de requisitos funcionais e não funcionais para o sistema [Bresciani et al. 2004]. Essa análise é dividida em duas fases: Análise dos Requisitos Iniciais e Análise dos Requisitos Finais.

Na primeira fase de análise dos requisitos, o usuário da metodologia identifica os agentes de domínio e modela-os como atores sociais, que dependem uns dos outros para atingirem os objetivos, planos e compartilharem dos mesmos recursos. Na segunda fase, o modelo conceitual é estendido de modo a incluir um novo ator, o qual representa o sistema, além de uma série de dependências com outros atores do ambiente.

[Bresciani et al. 2004] afirmam que o projeto arquitetural e as fases de projeto detalhado se concentram na especificação do sistema, em maneiras de garantir as exigências resultantes das fases anteriores. O projeto arquitetural define a arquitetura global do sistema em termos de subsistemas, interligados através de dados e fluxos de controle. O projeto arquitetural também prevê um mapeamento dos atores do sistema para um conjunto de agentes de software, cada um caracterizado pelas capacidades específicas.

A fase do projeto detalhado visa especificar as capacidades dos agentes e suas interações. Neste ponto, geralmente a plataforma de aplicação já foi escolhida, possibilitando a construção de um projeto detalhado que irá mapear diretamente o código.

A etapa de implementação segue passo a passo, de uma forma natural, a especificação do projeto detalhado com base no mapeamento estabelecido entre as construções da plataforma de execução e as noções do projeto detalhado [Bresciani et al. 2004].

2.3. MaSE

A metodologia *Multiagent Systems Engineering* (MaSE) foi proposta por [DeLoach et al. 2001]. É uma metodologia desenvolvida para análise e projeto de sistemas multiagente. MaSE utiliza da abstração fornecida por SMA para ajudar projetistas a desenvolver sistemas de software distribuídos inteligentes [Henderson-Sellers and Giorgini 2005].

A metodologia MaSE consiste de duas fases principais que resultam na criação de um conjunto de modelos complementares [Henderson-Sellers and Giorgini 2005]. A primeira fase é análise e a segunda é projeto.

A primeira fase da metodologia inclui três etapas: captura de objetivos, aplicação de casos de uso e refinamento de papéis. A segunda fase é composta por quatro etapas: criação de classes de agentes, construção das conversações, montagem das classes dos agentes e o projeto do sistema.

A finalidade da primeira etapa da fase de análise é capturar os objetivos do sistema extraindo-os da especificação inicial do sistema. Após a extração, os objetivos são estruturados em um diagrama chamado de hierarquia de objetivos. Posterior a elaboração deste diagrama, o analista deve associar cada objetivo com um papel e um conjunto de classes de agentes responsáveis para satisfazer esses objetivos.

Na etapa de aplicação de casos de uso, o analista transforma os objetivos e sub-objetivos em casos de uso. Esses casos de uso são descritores narrativos de uma sequência de eventos que define um comportamento desejado para o sistema. Além disso, a partir dos casos de uso é possível desenvolver um conjunto de diagramas de sequência que são semelhantes aos utilizados na UML.

A terceira etapa presente na fase de análise da metodologia MaSE é o refinamento de papéis. Esta etapa tem como propósito garantir que foram identificados todos os papéis necessários no sistema e desenvolver as tarefas que definem o comportamento dos papéis e os padrões de comunicação.

A primeira etapa da fase de projeto do sistema é chamada de criação de classes de agentes. Nesta etapa, as classes de agentes são identificadas a partir dos papéis refinados e são transcritas em um diagrama de classes de agentes. Além disso, são identificadas as conversas em que diferentes classes de agentes devem participar. Tais conversas são geralmente derivadas de comunicações externas dos papéis atribuídos ao agente. Nesse contexto, o projetista pode combinar papéis múltiplos em uma única classe de agentes ou mapear uma única função para ser exercida por classes de agentes múltiplos, objetivando eficiência e simplificação.

A próxima fase presente na metodologia é chamada de construção de conversações. Até o início dessa etapa, o projetista apenas identificou as conversas. O objetivo desta etapa é definir os detalhes dessas conversas, baseados nos detalhes internos de tarefas simultâneas.

Os elementos internos dos agentes são projetados durante a etapa de montagem das classes dos agentes, que inclui duas sub-etapas: a definição da arquitetura de agentes e a definição dos componentes da arquitetura. Analistas/projetistas tem a escolha de projetar sua própria arquitetura ou usar arquiteturas pré-definidas, como BDI. Da mesma

forma, um projetista pode usar componentes pré-definidos ou desenvolvê-los a partir do zero. Segundo [Bergenti et al. 2004], componentes consistem de um conjunto de atributos, métodos, e possivelmente uma sub-arquitetura.

Projeto do sistema é o último passo da metodologia MaSE. Nessa etapa é utilizado o diagrama de implantação para demonstrar os números, tipos e locais das instâncias dos agentes no sistema.

2.4. Ingenias

Segundo [Henderson-Sellers and Giorgini 2005], a metodologia Ingenias fornece uma notação para modelagem de SMA através de uma coleção bem definida de atividades, para orientar o processo de desenvolvimento. Especificamente, nas tarefas de análise, projeto, verificação e geração de código. A notação Ingenias, baseia-se em cinco metamodelos que definem os diferentes pontos de vista e conceitos a partir dos quais um SMA pode ser descrito.

2.4.1. Processo Ingenias

Segundo [Henderson-Sellers and Giorgini 2005], o processo Ingenias ajuda os desenvolvedores a produzir uma especificação SMA e sua implementação.

Sobre os pontos de vista, cada um deles é construído através de conjuntos de atividades estruturados em diagramas de atividades.

Um conjunto visa elaborar uma visão no nível de análise, enquanto o outro se concentra no projeto. No total, um desenvolvedor tem dez diagramas de atividades que propõem uma centena de atividades. Essas atividades foram numeradas para ajudar os desenvolvedores na sua aplicação. Sua enumeração é única dentro de um ponto de vista.

Durante a análise, o foco é sobre a definição de um modelo de organização que esboce a estrutura do SMA e identifique seus principais componentes. Este resultado, o equivalente a uma arquitetura SMA inicial, mais tarde é refinado, identificando os objetivos da organização e tarefas relevantes em relação a esses objetivos. Essas tarefas posteriormente são executadas por cada agente na organização.

Na fase de elaboração do projeto, mais detalhes são adicionados ao definir fluxos de trabalho entre os diferentes agentes, completando a definição de fluxo de trabalho com as interações do agente, e refinando o estado mental do agente, como consequência.

Ingenias baseia a sua proposta de implementação nas facilidades do *Ingenias Development Kit* (IDK) em mapear elementos de especificação em entidades computacionais. No domínio de agente, entidades computacionais relevantes poderiam ser arquiteturas de agentes, arquiteturas SMA, ou plataformas de agentes. Este mapeamento é realizado na IDK por módulos.

Segundo [Henderson-Sellers and Giorgini 2005], atualmente Ingenias apoia a verificação e validação utilizando a *activities theory* (AT). No IDK, é apenas apoiado a etapa de verificação. Ainda segundo os autores, qualquer módulo pode fornecer capacidades de verificação se um desenvolvedor segue as instruções básicas para a construção do módulo.

3. Comparativos entre Metodologias AOSE x UML

O objetivo dessa comparação é verificar se os artefatos criados pelas metodologias não poderiam fazer uso dos diagramas presentes na UML. Para tal finalidade, realizou-se um mapeamento dos diagramas através de tabelas. [Guedes 2004], [Bezerra 2007] e [Sommerville 2007] não definem com exatidão quais diagramas devem ser usados em quais fases das metodologias. Todos os autores deixam implícito que a escolha e o uso dos diagramas varia conforme a necessidade do projeto. Todavia, as tabelas ilustradas nas próximas seções tiveram a sua coluna UML organizada de acordo com a sugestão de uso presente no livro [Guedes 2004]. A seguir, é explicado cada uma das comparações.

3.1. Comparação com o Prometheus

Tabela 1. Comparação entre os artefatos gerados pela Metodologia Prometheus X Linguagem UML

UML	Prometheus
Diag. de Casos de Uso/Detalhado	Diag de Cenários; Formulário de Descrição do agente; Diag. de Ligação dos papéis dos agentes; Formulário Descritor de Planos
Diag. de Classes	
Diag. de Objetos	
Diag. de Estrutura Composta	
Diag. de Sequência	Diag. de Interação
Diag. de Comunicação	Diag. de Familiaridade de Agentes
Diag. de Máquina de Estados	
Diag. de Atividades	
Diag. de Componentes	Diag. de Acoplamento de Dados
Diag. de Implantação	
Diag. de Pacotes	Diag. de Visão Geral de Objetivos
Diag. de Interação	
Diag. de Tempo	
Diags. extras das Metodologias	Diag. de Papéis do Sistema
	Diag. de Visão Geral do Sistema
	Diag. de Protocolos de Interação
	Diag. de Visão Geral do Agente
	Diag. de Capacidades Modelando Planos

A metodologia Prometheus é composta de 13 artefatos para auxiliar o projetista na sua utilização, confirme visto na tabela 1. Alguns desses artefatos, podem ser modelados utilizando os diagramas presentes na UML sem perder a sua expressividade. É o caso dos seguintes artefatos:

- **Diagrama de Cenários:** Este artefato tem como finalidade mostrar uma visão generalizada do sistema [Padgham and Thangarajah 2004]. Para [Guedes 2004], o diagrama de casos de uso da UML tem o poder de expressar uma ideia geral sobre como o sistema irá se comportar. Se os dois diagramas tem praticamente a mesma finalidade, é possível a sua utilização sem perda nenhuma de sentido.

- **Diagrama de Visão Geral de Objetivos:** De acordo com [Padgham and Thangarajah 2004], este artefato tem como propósito representar os objetivos do sistema. [Guedes 2004] explica que os diagramas de pacotes representam os subsistemas englobados por um sistema de forma a determinar as partes que o compõem. Se pensarmos que cada objetivo principal é um sistema, e seus subobjetivos são subsistemas, é possível fazer uso do diagrama de pacotes para modelar os objetivos do SMA na metodologia.
- **Diagrama de Familiaridade de Agentes:** Conforme [Guedes 2004], o diagrama de comunicação é um complemento para o diagrama de sequência. Ademais, o diagrama de comunicação também desempenha a modelagem do vínculo entre os objetos do sistema e suas trocas de mensagens durante o processo. Se pensarmos que cada agente é um objeto, esse diagrama pode ser utilizado para modelar a familiaridade de agentes. Os objetivos são semelhantes, visto que o diagrama de familiaridades de agentes, consiste em ligar um agente com demais agentes que ele possui alguma interação [Padgham and Winikoff 2002].
- **Diagrama de Acoplamento de Dados:** Para [Padgham and Winikoff 2002], o diagrama de acoplamento de dados tem como propósito identificar os tipos de dados que necessitam ser armazenados pelo sistema e as relações que os papéis identificados tem com esses dados. [Guedes 2004] explica que o diagrama de componentes serve para representar os componentes do sistema quando o mesmo for ser implementado em termos de módulos de código fonte, bibliotecas, formulários, arquivos de ajuda, etc. Se pensarmos que cada papel do sistema e cada tipo de dado é um componente do sistema, esse diagrama consegue replicar fielmente a representação do diagrama de acoplamento de dados.
- **Diagrama de ligação de papel agente:** Segundo [Padgham and Winikoff 2002], o diagrama de ligação de papel de agente auxilia no agrupamento dos papéis identificados anteriormente, além da ligação aos agentes que irão executá-los. Fazendo uma analogia com o diagrama de casos de uso presente na UML, os papéis podem ser representados por "funcionalidades", já que os agentes devem executá-los, e os atores podem representar os agentes, conseguindo expressar similarmente a transição de um diagrama para outro sem perder a sua essência.
- **Diagrama de Interação:** Conforme [Padgham and Winikoff 2002], este diagrama contribui para especificar completamente a interação com os agentes. [Guedes 2004] afirma que o diagrama de sequência presente na UML é utilizado para modelar a troca de mensagens entre os objetos de um sistema. Se pensarmos que os agentes podem ser equiparados com objetos para a utilização do diagrama de sequências, podemos utilizá-lo sem perder o significado do diagrama de interação.
- **Formulário Descritor de Agente e Formulário Descritor de Planos:** Ambos os formulários sugeridos pela metodologia Prometheus permitem a escrita de texto em linguagem natural. Essa característica também é encontrada no diagrama de casos de uso detalhado, que compõe a UML.

Os demais diagramas presentes na metodologia Prometheus não oportunizam ser expressados utilizando os diagramas da linguagem UML, pois apresentam características que vão além das encontradas nesses diagramas.

Tabela 2. Comparação entre os artefatos gerados pela Metodologia Tropos X Linguagem UML

UML	Tropos
Diag. de Casos de Uso/Detalhado	
Diag. de Classes	
Diag. de Objetos	
Diag. de Estrutura Composta	
Diag. de Sequência	Diag. de Interação
Diag. de Comunicação	
Diag. de Máquina de Estados	
Diag. de Atividades	Diag. de Capacidade
Diag. de Componentes	
Diag. de Implantação	
Diag. de Pacotes	
Diag. de Interação	
Diag. de Tempo	
Diags. extras das Metodologias	Diag. de Autor
	Diag. de Objetivos
	Diag. de Objetivos Modelando Planos

3.2. Comparação com o Tropos

A metodologia Tropos é composta de 5 artefatos para auxiliar o projetista na sua utilização, confirme visto na tabela 2. Dois desses artefatos, podem ser transcritos utilizando diagramas presentes na ferramente UML. É o caso dos seguintes artefatos:

- **Diagrama de Capacidade:** Conforme [Bresciani et al. 2004], este diagrama é representado por meio de diagramas de atividades presente na linguagem UML. A metodologia utiliza esse diagrama por padrão, ou seja, ele satisfaz as necessidades fundamentais para modelar o conceito de capacidades que os agentes necessitam.
- **Diagrama de Interação:** Segundo [Bresciani et al. 2004], o diagrama de interação da metodologia Tropos é modelado utilizando o diagrama de interação presente na AUML. Entretanto, a diferenciação deste diagrama em relação ao de sequência da UML ocorre em virtude do primeiro modelar as interações dos atores com os agentes, ao invés de atores com objetos igual ao segundo caso. Portanto, se representar os objetos como sendo agentes, é possível transcrever as interações por meio do diagrama de sequências sem perda de informações.

Os outros três artefatos presentes na metodologia não podem ser modelados utilizando os artefatos da UML, pois englobam características que vão além do domínio suportado pelos diagramas da UML.

3.3. Comparação com o MaSE

A metodologia MaSE é composta de 8 artefatos para auxiliar o projetista na sua utilização, confirme visto na tabela 3. Segundo [Henderson-Sellers and Giorgini 2005], a metodologia MaSE foi construída sobre as técnicas orientadas a objeto existentes, porém, foi especializada para o domínio de SMA. Por essa razão, é a metodologia que mais apresenta

Tabela 3. Comparação entre os artefatos gerados pela Metodologia MaSE X Linguagem UML

UML	MaSE
Diag. de Casos de Uso/Detalhado	
Diag. de Classes	Diagrama de Classe de Agentes; Diagrama de Arquitetura de Agentes;
Diag. de Objetos	
Diag. de Estrutura Composta	
Diag. de Sequência	Diag. de Sequência
Diag. de Comunicação	
Diag. de Máquina de Estados	
Diag. de Atividades	Diag. de Tarefas Concorrentes; Diag. de Classe de Comunicação
Diag. de Componentes	
Diag. de Implantação	Diag. de Implantação
Diag. de Pacotes	Diag. de Hierarquia de Objetivos
Diag. de Interação	
Diag. de Tempo	
Diags. extras das Metodologias	Diag. de Modelo de Papéis

afinidades com os diagramas da linguagem UML. Dos 8 artefatos presentes na metodologia, 7 podem ser expressados utilizando a UML. Eles são:

- **Diagrama de Hierarquia de Objetivos:** Segundo [Henderson-Sellers and Giorgini 2005], este diagrama serve para estruturar os objetivos em ordem de importância. Fazendo a mesma analogia feita com o diagrama de Diagrama de Visão Geral de Objetivos da metodologia Prometheus, o diagrama de Hierarquia de Objetivos presente na metodologia MaSE também pode ser ilustrado utilizando o diagrama de pacotes presente na UML.
- **Diagrama de Sequência:** Conforme [Henderson-Sellers and Giorgini 2005], este diagrama é semelhante ao utilizado na UML. Sua diferenciação ocorre, devido ao da metodologia MaSE representar sequências de eventos entre os papéis, ao invés de objetos igual ao da UML. Entretanto, ambos tem a mesma notação gráfica e elementos, por isso podem ser equiparados.
- **Diagrama de Tarefas Concorrentes:** Para [Henderson-Sellers and Giorgini 2005], este diagrama serve para modelar as tarefas simultâneas realizadas pelos agentes. Um diagrama da UML que modelaria sem perda de expressividade essa situação, é o diagrama de atividades, visto que o mesmo também possui a notação para representar tarefas paralelas, além deste diagrama também possuir as condições de guarda.
- **Diagrama de Classe de Agente:** Este diagrama é semelhante ao de classes presente na UML. Sua diferenciação ocorre em virtude de um representar as classes de agentes e outro as classes de entidades do mundo real. É possível transcrever o diagrama de classes da agentes utilizando o de classes da UML sem nenhuma perda.
- **Diagrama de Classes de Comunicação:** De acordo com [Henderson-Sellers and Giorgini 2005], uma conversa define um protocolo

de coordenação entre dois agentes, e isto pode ser documentado utilizando o diagrama de classes de comunicação. Este diagrama tem uma notação semelhante ao de atividades da UML. Portanto, é possível transpassar os dados de um diagrama ao outro sem extravios de informação.

- **Diagrama de Arquitetura dos Agentes:** Este diagrama serve para especificar o conjunto de atributos, métodos, e sub-arquiteturas que possam conter os agentes [Henderson-Sellers and Giorgini 2005]. Sua notação gráfica é semelhante a presente no diagrama de Classes da UML. Por esta razão, é plausível reproduzir um diagrama no outro sem redução de expressividade.
- **Diagrama de Implantação:** Este diagrama serve para demonstrar números, tipos e locais das instâncias dos agentes no sistema [Henderson-Sellers and Giorgini 2005]. Sua notação gráfica é similar a encontrada no diagrama de implantação da UML. A diferença entre eles, ocorre devido ao diagrama de implantação da UML ter como objetivo a determinação das necessidades de hardware do sistema, as características físicas como servidores, estações, protocolos de comunicação, etc [Guedes 2004]. Devido a sua similaridade de notação gráfica, é cabível transcrever um diagrama no outro sem perda de sentido.

O número elevado de similaridades entre os diagramas da metodologia MaSE e os da UML decorre de MaSE ser baseado no paradigma de orientação a objetos. O único diagrama que não pode ser expressado pela UML é o diagrama de papéis, pois apresenta informações distintas não suportadas por diagramas UML.

3.4. Comparação com Ingenias

A metodologia Ingenias se difere das demais por ter sido construída utilizando pontos de vista distintos. Segundo [Henderson-Sellers and Giorgini 2005], a instanciação desses pontos de vista com entidades que representem problemas concretos, são o ponto base para o processo de desenvolvimento dessa metodologia. A fim de padronização, neste trabalho chamou-se os diagramas desta metodologia de diagramas de ponto de vista.

A tabela 4 exibe o comparativo entre os artefatos da metodologia Ingenias com os da linguagem UML. Conforme é ilustrado, a metodologia Ingenias praticamente não possui nenhuma similaridade em relação a UML, somente em um caso:

- **Diagrama do Ponto de Vista de Interação:** [Henderson-Sellers and Giorgini 2005] definem as notações utilizadas para modelar as interações entre os agentes e entre os agentes e os seres humanos. Eles explicam que essas interações também podem ser modeladas utilizando o diagrama de comunicação presente na UML, entretanto, este diagrama contém algumas restrições que fazem o modelo perder um pouco de sua expressividade conforme aumenta a complexidade das interações. Para tanto, os autores da metodologia propuseram uma versão de diagramas de Comunicação da UML, chamada de Diagrama de Interação GRASIA.

Nos estudos realizados acerca dessa metodologia, notou-se que os autores propuseram em sua ferramenta intitulada IDK um conjunto de diagramas para apoiar a modelagem do SMA utilizando da sua metodologia. Nas bases científicas abertas, não encontrou-se nenhum trabalho que explicasse integralmente as possibilidades que a ferramenta IDK

Tabela 4. Comparação entre os artefatos gerados pela Metodologia Ingenias X Linguagem UML

UML	Ingenias
Diag. de Casos de Uso/Detalhado	
Diag. de Classes	
Diag. de Objetos	
Diag. de Estrutura Composta	
Diag. de Sequência	
Diag. de Comunicação	Diag. do ponto de vista de Interação
Diag. de Máquina de Estados	
Diag. de Atividades	
Diag. de Componentes	
Diag. de Implantação	
Diag. de Pacotes	
Diag. de Interação	
Diag. de Tempo	
Diags. extras das Metodologias	Diag. do ponto de vista da Organização
	Diag. para especificar Fluxo de Trabalho
	Diag. do ponto de vista do Agente
	Diag. do ponto de vista de tarefas/objetivos
	Diag. do ponto de vista do Ambiente

podem proporcionar a seu usuário. Portanto, acredita-se, com base nas definições dessa metodologia, que os diagramas presentes no IDK, nos quais os autores intitularam GRASIA, servem para modelar o SMA utilizando das notações presentes nos diagramas de ponto de vista.

4. Conclusões e trabalhos futuros

Este trabalho teve como objetivo apresentar uma revisão de algumas metodologias para modelagem de sistemas multiagente. Também foi realizado um estudo comparativo entre as metodologias estudadas (Prometheus, Tropos, MaSE e Ingenias) e a linguagem UML, de forma a apresentar similaridades e diferenças.

Com base na exploração das metodologias, percebe-se que cada uma possui enfoque em características distintas. Enquanto Prometheus e Tropos focam sua modelagem no âmbito de agentes, MaSE e Ingenias possuem conceitos que propiciam também a modelagem de recursos e arquiteturas alternativas de agentes. Percebe-se também que a linguagem UML não é capaz de suprir os conceitos envolvidos no paradigma multiagente e que as metodologias AOSE ainda apresentam disparidade de objetivos, enfraquecendo sua normalização.

Em suma, a área de AOSE deve progredir para atingir uma padronização. Sugere-se que seja feito um mapeamento dos requisitos necessários para desenvolver qualquer tipo de SMA. Com isso, as entidades responsáveis por tal finalidade, deverão chegar há um consenso em relação a melhor metodologia para modelar SMA atualmente. Posterior a isso, deve-se trabalhar na metodologia selecionada para que esta atenda todos os conceitos envolvidos nas demais metodologias.

5. Agradecimentos

Os autores agradecem à Universidade Federal do Rio Grande - FURG e a Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - FAPERGS pelo suporte financeiro na realização do presente trabalho.

Referências

Bergenti, F., Gleizes, M.-P., and Zambonelli, F. (2004). *Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook*, volume 11. Springer.

Bezerra, E. (2007). *Princípios de Análise e Projeto de Sistemas com UML*. Elsevier.

Brandão, A. A. F. (2014). Apresentação de oficina no wesaac 2014 - engenharia de software orientada a agente. Enviado por e-mail pela autora (anarosabrandao@gmail.com), em 17 julho 2014.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

DeLoach, S. A., Wood, M. F., and Sparkman, C. H. (2001). Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(03):231–258.

Guedes, G. (2004). Uml: uma abordagem prática. *Novatec Editora*.

Guedes, G. T. A. (2012). *Um metamodelo UML para a modelagem de requisitos em projetos de sistemas multiagentes*. PhD thesis, Universidade Federal do Rio Grande do Sul.

Henderson-Sellers, B. and Giorgini, P. (2005). *Agent-oriented methodologies*. IGI Global.

Jayatilleke, G. B., Padgham, L., and Winikoff, M. (2007). Evaluating a model driven development toolkit for domain experts to modify agent based systems. In *Agent-Oriented Software Engineering VII*, pages 190–207. Springer.

Khallouf, J. and Winikoff, M. (2009). The goal-oriented design of agent systems: a refinement of prometheus and its evaluation. *International Journal of Agent-Oriented Software Engineering*, 3(1):88–112.

Padgham, L. and Thangarajah, J. (2004). Tutorial prometheus. http://www.cs.rmit.edu.au/agents/pdt/docs/Tutorial.pdf.

Padgham, L. and Winikoff, M. (2002). Prometheus: A methodology for developing intelligent agents. *John Wiley & Sons*.

Padgham, L. and Winikoff, M. (2005). *Developing intelligent agent systems: A practical guide*, volume 13. John Wiley & Sons.

Rodriguez, L., Insfran, E., and Cernuzzi, L. (2011). *Requirements Modeling for Multi-Agent Systems*. INTECH Open Access Publisher.

Sommerville, I. (2007). *Engenharia de Software*, volume 8. Person Education.