3
Trabalhos Relacionados

Os trabalhos relacionados apresentados nesta Seção estão divididos em duas categorias: metodologias para SMA e utilização da abordagem MDA no desenvolvimento de SMA. As metodologias (MaSE, Tropos, Prometheus, PASSI, Message e Gaia) estão relacionadas neste trabalho pois elas também propõem, de forma indireta, um processo de desenvolvimento para SMA.

Os demais trabalhos estão associados ao trabalho apresentado por utilizarem também a abordagem MDA em alguma etapa do processo de desenvolvimento de SMA.

3.1.
Metodologias existentes para desenvolvimento de SMA

Os trabalhos apresentados a seguir correspondem a metodologias propostas para o desenvolvimento de sistemas multi-agentes, que utilizam linguagens de modelagem específicas para este tipo de sistema e algumas que propõem ferramentas para darem suporte a mesma.

3.1.1.
MaSE

A metodologia MaSE (Multi-Agent System Engineering) [30] é semelhante às metodologias tradicionais, porém é voltada para o paradigma de agentes. Seu principal objetivo é abordar todo o ciclo de vida do desenvolvimento do sistema. Nesta metodologia agentes são vistos como simples processos de software que interagem para chegarem a um objetivo em comum. Esta metodologia é composta por sete passos, divididos em duas fases (análise e design).

A fase de análise contém três passos principais: captura dos objetivos, utilização de casos de uso e refinamento dos papéis. No primeiro passo, o analista deve identificar os objetivos e sua estrutura e representá-los em uma hierarquia. Com base neste levantamento, tem-se o passo seguinte que é responsável por extrair os casos de uso, traduzindo os objetivos em papéis e tarefas. A última etapa desta fase utiliza os artefatos produzidos no passo

anterior para refinar o conjunto de papéis e tarefas que serão exercidas para atingir cada objetivo definido no primeiro passo.

Na fase seguinte (design) existem quatro passos: criação das classes de agentes, construção das conversações, organização dos agentes e modelagem do sistema, que devem ser seguidos nesta ordem. No passo de criação das classes de agentes, os agentes são identificados através dos papéis. O próximo passo detalha melhor a conversa entre os agentes, definida através de um protocolo coordenado entre eles. O passo denominado organização de agentes pode ser realizado paralelamente ao passo anterior, onde é definida a arquitetura dos agentes e os componentes que serão construídos nesta arquitetura [31]. O último passo da segunda fase corresponde à configuração do sistema a ser implementado.

Apesar de a literatura [30, 31] expor que esta metodologia envolve todo o ciclo de desenvolvimento, não é apresentado o modo como é feita a transformação dos modelos em código. A única forma apresentada da geração do código a partir dos modelos é através da ferramenta criada para dar apoio a este processo denominada AgentTool. Esta ferramenta implementa todos os sete passos definidos na metodologia MaSE, e também dá suporte a transformação da fase de análise para a de design. AgentTool é baseado no framework agentMOM e tem como resultado final o código Java referente à modelagem realizada na ferramenta.

3.1.2. Tropos

Tropos [32] é uma metodologia para SMA que visa fornecer suporte a todas as atividades de análise e projeto no processo de desenvolvimento, desde a análise do domínio até a implementação do sistema. Tropos está dividido em cinco fases: fase inicial de requisitos, fase final de requisitos, projeto arquitetural, projeto detalhado e implementação.

Nesta metodologia, a análise de requisitos envolve duas fases: a fase inicial e a fase final. Na fase inicial (*early requirements*) são definidos os *stakeholders* do domínio, modelados como atores sociais, com dependências baseadas em objetivos, planos e fornecimento de recursos. Na fase final (*late requirements*), o modelo conceitual é estendido, incluindo-se um novo ator que representa o sistema, e as dependências com outros atores do ambiente [33].

As fases de projeto arquitetural e detalhado estão mais relacionadas à especificação do sistema. O projeto arquitetural define a arquitetura global do sistema em termos de sub-sistemas (atores), troca de dados e fluxo de controles (dependências). Nesta fase também é feito um mapeamento dos atores do sistema em um conjunto de agentes de software, cada um caracterizado por suas capacidades [32]. O projeto detalhado especifica as capacidades (habilidades) dos agentes e interações. Cada agente é definido, mais especificamente em termos de entradas, saídas, controles e outras informações relevantes. Nesta fase, a plataforma de desenvolvimento já é escolhida.

A ultima fase desta metodologia corresponde a fase de implementação, que se baseia na definição do projeto detalhado. Para esta fase, Tropos optou por escolher uma plataforma BDI [34], especificamente a JACK [35], para a implementação SMA. Os agentes em JACK são componentes de software autônomos que apresentam objetivos a serem alcançados ou eventos a serem tratados. São programados com um conjunto de planos para torná-los capazes de alcançar seus objetivos. Os modelos desenhados para esta plataforma são da seguinte forma [32]:

- agente: em JACK, o agente é usado para definir o comportamento de um agente de software, incluindo as suas capacidades, tipos de mensagens, eventos aos quais ele responde e os planos para atingir seus objetivos;

- capacidade: uma capacidade em JACK pode incluir planos, eventos, crenças e outras capacidades;

- evento: eventos internos e externos são mapeados para os eventos definidos em JACK;

- plano: são mapeados para os planos definidos em JACK.

A modelagem realizada em Tropos é bastante confusa e rebuscada, dificultando esta fase do processo de desenvolvimento. A fase de projeto detalhado é orientada especificamente à plataforma JACK.

3.1.3. Prometheus

Prometheus [36] é uma metodologia para desenvolvimento de sistemas multi-agentes que abrange desde a modelagem até a implementação. Esta metodologia é composta por três fases, onde os artefatos produzidos são utilizados tanto na geração do esqueleto do código, como também para depuração e teste.

A primeira fase, correspondente à especificação do sistema, compreende duas atividades: determinar o ambiente do sistema e determinar os objetivos e funcionalidades do sistema. O ambiente do sistema é definido em termos de percepções (informações provenientes do ambiente) e ações. Além disso, são definidos dados externos. As funcionalidades do sistema são definidas através da identificação de objetivos, da definição das funcionalidades necessárias para se alcançar esses objetivos e dos cenários de casos de uso [37].

A fase seguinte (projeto arquitetural) utiliza as saídas da fase anterior para determinar quais agentes existirão no sistema e como os mesmos irão interagir. Esta fase envolve três atividades: definição dos tipos de agentes, definição da estrutura do sistema e definição das interações entre os agentes [36].

A última fase desta metodologia (projeto detalhado) é responsável por definir capacidades dos agentes, eventos internos, planos e uma estrutura de dados detalhada de cada tipo de agente identificado na fase anterior.

Atualmente existem duas ferramentas que utilizam o Prometheus. O ambiente de desenvolvimento do JACK (JDE) [35], que inclui uma ferramenta de modelagem para a construção dos diagramas, resultando na geração do código na linguagem de programação JACK. O JDE dá suporte à metodologia Prometheus pelo fato dos conceitos utilizados por JACK corresponderem aos artefatos gerados na fase de projeto detalhado da metodologia.

A outra ferramenta é o Prometheus Design Tool (PDT). Ela permite que o usuário entre e edite o projeto utilizando os seguintes conceitos: verificar o projeto para um conjunto de possíveis inconsistências, gerar automaticamente um conjunto de diagramas de acordo com a metodologia e gerar automaticamente a descrição do projeto, o que inclui descritores para cada entidade, um dicionário para o projeto e os diagramas gerados anteriormente.

Nenhuma dessas duas ferramentas dão suporte à fase de especificação. Por exemplo, os processos de decidir os tipos de agentes através da combinação das facilidades e determinar as funcionalidades do sistema através dos cenários de caso de uso não são oferecidos por essas ferramentas.

3.1.4. PASSI

PASSI (Process for Agent Societies Specification and Implementation) [38] é uma metodologia para desenvolvimento de sistemas multi-agentes que integra a definição da modelagem e filosofia de SMA como também de orientação a

objetos, utilizando UML. Esta metodologia é composta de cinco modelos que endereçam diferentes visões e doze passos durante seu processo de desenvolvimento.

Os modelos existentes nesta metodologia estão dispostos da seguinte forma: requisitos do sistema, sociedade dos agentes, implementação dos agentes, código e distribuição. O modelo relacionado aos requisitos do sistema compreende quatro passos: descrição do domínio, identificação dos agentes, identificação dos papéis e especificação das tarefas. Já o modelo de sociedade dos agentes é responsável por modelar as interações e dependências entre os agentes presentes na solução. A elaboração deste modelo envolve os três passos seguintes: descrição das ontologias, descrição dos papéis e descrição dos protocolos.

O modelo seguinte, implementação dos agentes, tem por objetivo modelar a arquitetura utilizada como solução em termos de classes e métodos. Este modelo compreende os seguintes passos: definição da estrutura do agente e descrição do comportamento do agente. O modelo de código visa modelar a solução a nível de código e compreende mais dois passos presentes nesta metodologia. E como último modelo tem-se o modelo de distribuição que descreve a distribuição das partes do sistema.

Existem duas ferramentas que oferecem suporte a esta metodologia: PTK Toolkit e AgentFactory. A PTK é um add-in para a ferramenta Rational Rose [39], que dá suporte a fase de geração de código Java. AgentFactory é uma aplicação que permite a geração de protótipos de partes de sistemas que utilizam plataformas que estejam de acordo com a FIPA [40].

3.1.5. MESSAGE

A MESSAGE (Methodology for Engineering Systems of Software AGEnts) [41] é uma metodologia voltada para SMA que estende o meta-modelo de UML através da adição de novos conceitos. Esta metodologia está divida em duas fases: análise e design. A fase de análise tem por objetivo produzir a especificação (modelo de análise) do sistema a ser desenvolvido. Enquanto a fase de design pretende definir a solução do problema, através da identificação dos componentes computacionais do sistema, especificação sobre o que eles devem realizar e estabelecer as interfaces entre esses componentes.

Além de utilizar as linguagens de modelagem UML e AUML, MESSAGE oferece cinco diferentes visões com diferentes perspectivas que dão ênfase a diferentes aspectos do sistema. Estas visões estão presentes na fase de análise, são elas:

- visão organizacional - define a estrutura e o comportamento de um grupo de agentes que trabalham juntos para alcançar um determinado objetivo;

- visão de tarefas/metas – apresenta as metas, tarefas, estados e dependências entre elas;

- visão de agentes/papéis - descreve os agentes e os seus papéis. Serve para modelar o propósito, os relacionamentos, o comportamento e as demais características dos agentes e dos papéis do sistema;

- visão de interações – descreve as interações entre os agentes e os protocolos que as implementam;

- visão do domínio – apresenta os conceitos específicos do domínio e as relações que são relevantes para o sistema. Descreve o ambiente e seus recursos.

É importante ressaltar que esta metodologia não demonstra como é feita a transformação de modelos em código. No entanto, foi desenvolvida uma ferramenta denominada MetaEdit+ [41] baseada na metodologia MESSAGE que permite a geração do código a partir da modelagem desenhada na própria ferramenta. Esta ferramenta foi estendida para atender a metodologia MESSAGE, através da criação de novos conceitos que não estavam disponíveis em UML, como por exemplo novos diagramas. MetaEdit+ não dá suporte a todas funcionalidades presentes na metodologia, como por exemplo, os diagramas de interação e seqüência.

3.1.6. Gaia

A metodologia Gaia [1] oferece uma linguagem própria para a modelagem de SMA. O processo de desenvolvimento Gaia contém duas fases: análise e design. A metodologia tem início na fase de análise, visando coletar e organizar a especificação que será base para a fase de design.

A fase de análise tem por objetivo entender o sistema e decompô-lo em papéis que serão desempenhados na organização, através do modelo de papéis, e definir como os mesmos interagem, através do modelo de interação. Portanto, o modelo de papéis identifica os papéis existentes no sistema e o

modelo de interação um conjunto de definições de protocolos, um para cada tipo de interação entre os papéis. Esta fase de análise inclui identificar [1]: as metas das organizações presentes no sistema e o comportamento esperado dos mesmos, o ambiente, os papéis iniciais, as interações iniciais e as regras que a organização deve seguir.

Os artefatos gerados na fase de análise, mencionados no parágrafo anterior, são utilizados como entrada para a fase seguinte, denominada aqui de fase de design. A fase de design pode ser logicamente decomposta em duas novas fases: elaboração da arquitetura e detalhamento da implementação. A fase da elaboração da arquitetura inclui: a definição da estrutura organizacional do sistema em termos de sua topologia e regime de controle e a identificação completa dos papéis e interações. Uma vez que a fase da elaboração da arquitetura é finalizada, a fase de detalhamento pode ser iniciada. Esta fase compreende a definição do modelo de agentes e a definição do modelo dos serviços que os agentes devem oferecer para desempenhar seus papéis. O modelo de agentes identifica as classes de agentes existentes no sistema. Enquanto o modelo de serviços identifica os serviços (funções do agente) associados a cada classe de agentes e define as principais propriedades desses serviços.

3.1.7. Considerações sobre as metodologias apresentadas

As metodologias apresentadas nesta Seção propõem um processo de desenvolvimento que se inicia na análise do SMA em um nível mais abstrato até seu nível mais detalhado, através do refinamento consecutivo dos modelos. No entanto, não há uma divisão explicita de uma modelagem independente de plataforma para uma modelagem dependente de plataforma. Isto é, durante a modelagem da aplicação já são inseridos conceitos relacionados à plataforma que será utilizada para o desenvolvimento da aplicação.

Um dos principais problemas das metodologias Gaia, MaSE e MESSAGE é o fato delas, embora cubram as fases de análise e design do processo de desenvolvimento de SMA, não endereçam de forma satisfatória a fase de implementação. Isto é, não resolvem por completo como alcançar a derivação de um modelo, a partir da modelagem até uma implementação concreta. Contudo, tanto a metodologia MaSE quanto a MESSAGE possuem ferramentas de apoio

que permitem a criação dos modelos e a partir dele a geração do código, porém a forma como é realizada está transformação não é encontrada na literatura.

As demais metodologias propõem um processo de desenvolvimento de SMA englobando desde a análise até a implementação. No entanto, nenhuma delas consegue apresentar com nitidez como é feita a transformação dos modelos propostos em código. Além disso, em sua grande maioria os modelos sugeridos são dependentes de plataforma, uma vez que em sua modelagem é necessário especificar estruturas que variam de acordo com a plataforma a ser utilizada. Como exemplo temos a metodologia Tropos que na sua fase de modelagem do projeto detalhado já deve se inserir características da plataforma que será utilizada para a implementação do sistema.

As metodologias Tropos e Prometheus apresentam uma associação entre seus modelos mais detalhados e a plataforma JACK, devido ao fato de utilizarem conceitos em comum. No entanto, não é apresentado de forma clara como é feita a transformação do modelo em código.

A metodologia proposta neste trabalho propõe um processo de desenvolvimento de SMA desde a sua modelagem até a sua implementação utilizando a abordagem MDA. Neste processo são criados modelos independentes de plataforma e modelos dependentes de plataforma, permitindo assim que os modelos de mais alto nível de abstração não contenham detalhes de implementação. A utilização da abordagem MDA, faz com que se tenha uma melhor qualidade, permitindo o reuso dos modelos e o mapeamento entre os mesmos, facilitando a manutenção através da consistência entre os modelos e o código. Desta forma, o processo proposto apresenta etapas com características bem definidas e o mapeamento entre essas etapas através da aplicação de regras de transformação claras, fazendo com que se tenha uma nítida representação de como os modelos gerados na primeira etapa geraram o código resultante do processo.

3.2. Trabalhos que utilizam a abordagem de MDA

3.2.1. Abordagem de MDA e a metodologia Tropos

Nuvikau [42] analisa como a utilização do GR (Graph Rewriting) [43] para a construção de modelos Tropos está em concordância com MDA. O autor julga então razoável a representação da sintaxe de Tropos através da utilização de

um conjunto de regras que permita a geração de todos os modelos existentes em Tropos.

A abordagem de MDA é interpretada pelo autor como uma atividade de modelagem visual onde modelos mais abstratos são refinados em modelos mais detalhados utilizando técnicas de transformação [42]. Neste trabalho as técnicas de GR junto com as suas transformações são utilizadas para garantir que as diferentes visões existentes em Tropos estejam em conformidade umas com as outras e também asseguram a consistência dos modelos nas diferentes fases do processo de modelagem.

A utilização das técnicas de GR em conjunto com as transformações derivadas torna possível [42]:

- a conformidade de diferentes visões do modelo Tropos: diagrama Tropos, diagrama de seqüência e especificações formais;

- a consistência dos modelos nas diferentes fases do processo de modelagem, oferecendo rastreamento de requisitos e propagação automática de mudanças de um modelos de uma fase para outro modelo de outra fase.

3.2.2. MetaDIMA

O projeto MetaDIMA oferece uma arquitetura baseada em modelos para o desenvolvimento de SMA, inspirada na abordagem MDA. Este projeto propõe realizar a ligação entre as arquiteturas existentes para agentes com suas ferramentas de desenvolvimento e metodologias também baseadas em agentes [44]. Prega que o conhecimento do negócio deve ser permanente enquanto conceitos técnicos têm geralmente uma sobrevida curta e estão limitados a uma certa tecnologia.

O objetivo do MetaDIMA é oferecer um conjunto de meta-modelos e um conjunto de sistemas baseados no conhecimento. Estes meta-modelos serão utilizados para descrever os modelos. O último pode ser considerado como uma especificação do sistema, constituindo os alicerces para a construção de SMA. O sistema multi-agente pode ser considerado como uma sucessão de automáticas ou semi-automáticas transformações de modelos, sendo que algumas transformações requerem uma entrada preliminar do desenvolvedor.

Para a elaboração do MetaDIMA são necessários os seguintes passos [44]:

- identificação de diferentes níveis de abstração: análise das teorias de SMA e ferramentas para identificar os PIMs e PSMs;

- definição de uma biblioteca de meta-modelos: identificação de conceitos de cada nível de abstração e determinação do meta-modelo apropriado;

- definição de regras de transformação: análise do conhecimento envolvido no desenvolvimento de SMA para definir as regras de transformação.

Este projeto foi experimentado utilizando a plataforma DIMA (framework para construção de SMA) e a ferramenta de desenvolvimento MetaGen (framework para geração de sistemas de informação a partir dos requisitos do usuário). O MetaGen foi utilizado para definir os meta-modelos e o DIMA responsável por gerar o código a partir do modelo.

3.2.3. Projeto e implementação orientada a agentes com MDA

Em [45] é apresentado como MDA pode ser utilizado para derivar implementações de SMA a partir de modelos orientados a agentes, independente da metodologia usada e da plataforma selecionada. Este trabalho propõe a utilização de MDA para transformar modelos de acordo com a metodologia Tropos em modelo Malaca [45].

O modelo Malaca [46] é um modelo para agentes que não é dependente da plataforma onde será executado, e portanto não há a necessidade de desenvolver diferentes implementações para cada tipo de plataforma. Sua arquitetura é baseada na definição e reuso de componentes de software. Em Malaca os agentes podem ser “programados” simplesmente através da edição de um documento XML.

Este trabalho tem como propósito endereçar o problema relacionado com a transformação dos diagramas modelados na fase de design em um conjunto de classes implementadas por uma plataforma orientada a agentes (por exemplo, Zeus e Jade). Este problema é solucionado através da visualização dos diagramas e das plataformas como *modelos*. Desta forma, a transformação dos diagramas em código será interpretada como uma transformação entre modelos, de acordo com MDA [45].

Na fase de análise, Tropos é utilizado para detalhar as interações, capacidades e planos dos agentes. Esta modelagem é então traduzida para a descrição de agentes definido em Malaca. Esta transformação não depende da plataforma na qual o sistema será executado e portanto não é necessário

desenvolver diferentes implementações para cada plataforma baseada na FIPA. Portanto, esta transformação corresponderá a transformação existente em MDA de PIMs para PSMs.

3.2.4. Considerações sobre os trabalhos apresentados

O trabalho apresentado na Seção 3.2.1 apresenta apenas a etapa de requisitos existente em Tropos. Não cobre a etapa referente à transformação dos modelos em código, diferente do que está sendo proposto neste trabalho, onde todas as etapas necessárias para o desenvolvimento de um sistema são contempladas, desde a modelagem até a implementação, a partir de sucessivas transformações realizadas automaticamente de uma etapa para outra.

O MetaDIMA tem uma dependência muito grande da plataforma DIMA. No processo proposto neste trabalho, o desenvolvedor está apto a escolher diferentes plataformas existentes para agentes.

No trabalho apresentado utilizando Tropos e a plataforma Malaca, a transformação entre os modelos definidos em Tropos para o modelo Malaca, não é totalmente automatizado, necessitando assim a intervenção manual em algumas regras de transformação. Além disso, o trabalho não trata a transformação do modelo Malaca em código.

O processo proposto neste trabalho visa cobrir as deficiências encontradas nos trabalhos apresentados acima, visto que contempla desde a fase de design até a implementação, através de transformações automáticas e sucessivas entre os modelos.