2
Fundamentos Teóricos e Tecnologias Básicas

2.1.
Sistemas Multi-Agentes

Para se ter uma melhor compreensão sobre o que são sistemas multi-agentes, é necessário primeiramente entender o conceito de *agentes*. De acordo com [21], a definição de agente ainda não é um consenso em toda a comunidade. No entanto, um crescente número de pesquisadores utilizam a seguinte definição: “um agente é um sistema computacional situado em um ambiente, e que é capaz de ações flexíveis e autônomas neste ambiente a fim de alcançar seus objetivos”.

O comportamento de um agente dentro de uma organização é definido por suas propriedades. Diferentes agentes podem possuir diferentes propriedades, fazendo com que tenham comportamentos distintos. Segundo [22], um agente possui algumas das seguintes propriedades:

- Autonomia: capacidade de tomar ações conduzindo para o término de algumas tarefas e objetivos, sem interferência do usuário final;
- Interação: comunicação com o ambiente e outros agentes;
- Adaptação: capacidade do agente de modificar seu comportamento em função de experiências anteriores;
- Mobilidade: capacidade de se transportar de um ambiente para outro;
- Cooperação: capacidade de agentes trabalharem juntos alcançando um objetivo em comum através da ajuda mútua.

De acordo com as suas propriedades, um agente pode ser classificado em diversos tipos, a seguir são apresentados alguns dos possíveis tipos de um agente [23]: mediadores, facilitadores, de mercado, entre outros.

Um sistema baseado em agentes pode ser de dois tipos [24]: sistemas distribuídos ou sistemas centralizados em um único agente. Um sistema centralizado possui um único agente que controla todas as decisões, enquanto outros agentes agem como escravos remotos. Um sistema multi-agentes distribuído apresenta um grupo de agentes interagindo em um ambiente comum, onde um agente é uma entidade vivendo nesse ambiente e podendo modificar o

ambiente (comunicação, decisão e ação) e a ele mesmo (percepção, aprendizado e raciocínio).

Os sistemas multi-agentes oferecem vários benefícios no que diz respeito a solucionar problemas monolíticos, como por exemplo [25]: (i) maior rapidez na resolução de problemas através do aproveitamento do paralelismo; (ii) melhor aproveitamento da comunicação, por transmitir somente soluções parciais em alto nível para outros agentes, ao invés de dados brutos para algum lugar central; (iii) mais flexibilidade, por ter agentes de diferentes habilidades que são dinamicamente agrupados para resolver problemas; e (iv) aumento da segurança, pela possibilidade de agentes assumirem responsabilidades daqueles que falham.

Uma das formas de se identificar SMA é observando se o problema possui as seguintes características [25]:

- O domínio envolve distribuição intrínseca de dados, capacidade de resolução de problemas e responsabilidades;
- Necessidade de manter a autonomia de subpartes, sem a perda da estrutura organizacional;
- Complexidade nas interações, incluindo negociação, compartilhamento de informação e coordenação;
- Impossibilidade de descrição da solução do problema a priori, devido à possibilidade de perturbações em tempo real no ambiente (p.ex: falhas no equipamento) e processos de negócio de natureza dinâmica.

São muito variadas as áreas de aplicações de SMA, como por exemplo: monitoramento de veículos distribuídos (conjunto de agentes monitoram veículos que passam através de suas respectivas áreas), controle de tráfego aéreo (agentes são utilizados para representar aeronaves e os vários sistemas de controle de tráfego aéreo), comércio eletrônico, entre outras.

2.2. MAS-ML

A linguagem MAS-ML [21] é uma linguagem de modelagem específica para sistemas multi-agentes. Esta linguagem é uma extensão de UML utilizando os conceitos apresentados no framework conceitual TAO [26]. O framework conceitual TAO define as abstrações comumente encontradas em SMA assim como suas propriedades, seus relacionamentos e a maneira como as entidades executam e interagem. Sendo assim, utilizando-se MAS-ML é possível modelar

não somente objetos (já modelados em UML) mas também agentes, ambientes, organizações e papéis: entidades definidas no TAO e comumente encontradas em SMA. A Figura 1 ilustra todas as entidades que podem ser modeladas em MAS-ML e todas os relacionamentos definidos entre estas entidades.

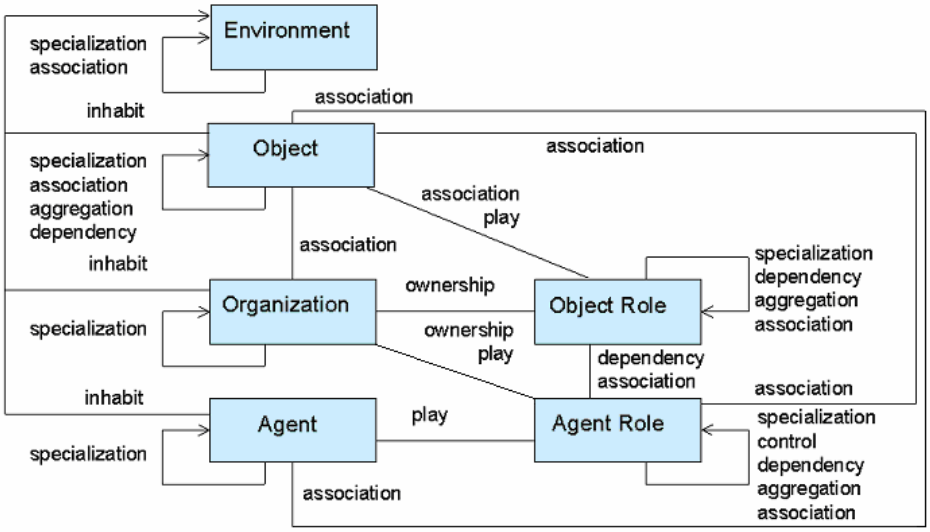

Figura 1 – Entidades e relacionamentos propostos por TAO

A extensão realizada em UML para a criação das novas entidades e relacionamentos consistiu na criação de novas meta-classes e estereótipos incluídos no meta-modelo de UML para atender aos conceitos apresentados no TAO. Na Tabela 1 apresentada abaixo são listadas algumas das meta-classes que foram adicionadas ao meta-modelo de UML:

TAO		**MAS-ML**
Entidade/ Relacionamento	**Descrição**	**Meta-Classe/ Estereótipo**
Agent	Elemento autônomo, adaptativo e interativo que tem um estado mental. Seu estado é expresso através de componentes mentais como crenças, metas, planos e ações.	*AgentClass*
Environment	Elemento habitado por agentes, objetos e organizações. Pode ser heterogêneo, dinâmico, aberto e distribuído. Apresenta estado, comportamento e relacionamentos. O estado de um ambiente armazena a relação de recursos, serviços e permissões de acesso.	*EnvironmentClass*
Organization	Elemento que agrupa agentes, que desempenham papéis e têm metas em comum. É uma extensão de agente, pois suas propriedades e seus relacionamentos são descritos do mesmo modo que em agentes	*OrganizationClass*
Object Role *Agent Role*	Elemento definido no contexto de uma organização. É um elemento que guia e restringe o comportamento de um agente, objeto ou sub-organização em uma organização, pode estar relacionado a um objeto (object role) ou agente (agent role)	*ObjectRoleClass* *AgentRoleClass*

Inhabit	Relacionamento que especifica que um elemento que habita em um ambiente é criado e destruído no habitat e deve sair e entrar no habitat, respeitando as permissões do mesmo	*Inhabit*
Ownership	Relacionamento onde um elemento (membro) pertence a outro elemento (proprietário) e um membro deve obedecer um conjunto de regras definidas pelo proprietário	*Ownership*
Play	Relacionamento que especifica que objetos, agentes ou sub-organizações que desempenham um papel têm propriedades e relacionamentos definidos pelo papel	*Play*
Control	Relacionamento que define que um elemento que é controlado deve obedecer seu controlador	*Control*
Action	Ação desempenhada por um agente ou organização	*AgentAction*
Plan	Plano desempenhado por um agente ou organização, está associado a uma meta e é representado por um conjunto de ações	*AgentPlan*
Protocol	Descreve os protocolos que um agente deve obedecer quando está desempenhando um papel e interagindo com outras entidades do sistema	*AgentProtocol*
Message	Para complementar a definição de protocolo, é enviada ou recebida por um agente	*AgentMessage*
Goal	Estado futuro ou desejo que um agente deve alcançar ou satisfazer	*Goal*
Belief	Conhecimento de um agente sobre o ambiente, metas, ele próprio e outras entidades	*Belief*
Axiom	Regra de uma organização que o agente deve obedecer	*Axiom*

Tabela 1 – Algumas meta-classes e estereótipos criadas por MAS-ML

Com o objetivo de modelar os aspectos estruturais e os aspectos dinâmicos das entidades relacionadas a agentes definidas no TAO, MAS-ML define três diagramas estruturais e um diagrama dinâmico. Os diagramas estruturais possibilitam a modelagem de todas as classes das entidades, suas propriedades e seus relacionamentos. Utilizando-se diagramas dinâmicos é possível modelar o comportamento das instâncias das entidades enfocando a execução interna das mesmas e as interações entre elas. Os diagramas estruturais propostos por MAS-ML são o diagrama de classes de UML estendido, o diagrama de organizações e o diagrama de papéis. O diagrama dinâmico é o diagrama de seqüência de UML estendido.

O Diagrama de Classes presente em MAS-ML difere em alguns pontos do Diagrama de Classes presente em UML, visto que novas meta-classes foram incorporadas ao meta-modelo para representar agentes, organizações e ambientes e os relacionamentos entre os agentes e classes, organizações e classes e entre ambientes e classes definidos em TAO.

O Diagrama de Organizações tem por objetivo modelar as organizações existentes no sistema identificando seus ambientes, os papéis que as

organizações definem e os elementos (objetos, agentes e sub-organizações) que desempenham esses papéis. Já o Diagrama de Papéis tem como finalidade esclarecer as relações presentes entre os papéis dos agentes e os papéis dos objetos.

O Diagrama de Seqüência é utilizado para representar o comportamento de agentes, organizações e ambientes. Este diagrama foi estendido de modo a comportar as interações existentes nos sistemas multi-agentes e as execuções internas dos agentes, organizações e ambientes.

2.3. Framework ASF

O framework ASF [17] possibilita a implementação de SMA usando-se a tecnologia orientada a objetos (OO). Assim como MAS-ML, o framework ASF também é baseado no framework conceitual TAO, sendo capaz, portanto, de representar todos os elementos e relacionamentos presentes em TAO e representados em MAS-ML. Com o uso do framework é possível implementar os agentes, papéis, organizações e ambientes modelados em MAS-ML utilizando-se orientação a objetos.

Cada módulo que compõem o framework representa um tipo de entidade existente em SMA. O conjunto de classes e relacionamentos definidos em um módulo possibilita a implementação dos aspectos estruturais de uma entidade (suas propriedades e relacionamentos com outras entidades) e dos aspectos dinâmicos da mesma (seu comportamento). A Figura 2 apresenta todas as classes dos cinco módulos que estão presentes no framework. Os módulos que representam agentes (delimitado através de um retângulo contínuo), organizações (definido pelo enlace pontilhado) e papéis de agentes (marcado pelo retângulo tracejado) são os módulos mais complexos. A complexidade se dá pela necessidade da criação de classes para representar as propriedades destes tipos de entidades. Os módulos para representação de ambientes (círculo tracejado) e objetos (círculo contínuo) são simplesmente representados por uma classe devido às propriedades destas entidades poderem ser representadas diretamente como atributos e métodos. Para maiores detalhes sobre a definição dos módulos se referir a [17].

Para implementar um tipo de ambiente ou um tipo de papel desempenhado por um objeto utilizando-se o framework, é bastante simples. Basta estender as classes *Environment* e *ObjectRole*, respectivamente. Já a implementação de um

tipo de agente se dá através da criação de uma classe que estenda a classe abstrata *Agent* definida no framework e a implementação do construtor da classe para criar os objetivos, crenças, planos e ações de acordo com as propriedades do agente. Os objetivos e crenças do agente são criados instanciando-se objetos das classes *Goal* and *Belief*. A representação dos planos e das ações dos agentes ocorre através da criação de classes concretas que estendam as classes abstratas *Plan* and *Action* e que implementem o comportamento definido planos e ações.

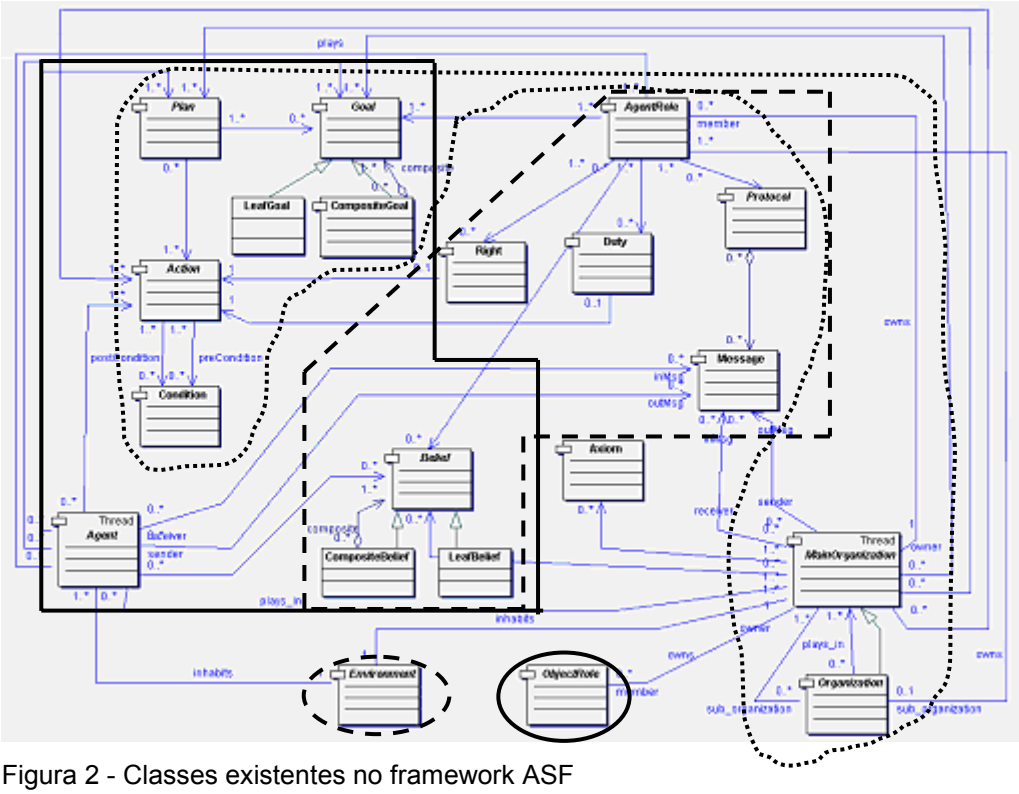

Figura 2 - Classes existentes no framework ASF

A implementação de um tipo de organização é bastante parecida com a implementação de um tipo de agente a menos de duas diferenças. A classe que representará o tipo de organização poderá estender da classe *MainOrganization* ou da classe *Organization*, de acordo com sua definição. Além disso, será necessário criar instâncias da classe *Axiom* no construtor da organização para representar as regras (ou leis) da organização.

Um tipo de papel de agente é implementado através da criação de uma classe que estenda a classe *AgentRole*. A definição dos direitos e deveres do papel se dará pela instanciação das classes *Right* and *Duty*. Os protocolos são definidos através da criação de classes concretas que implementem o comportamento definido nos protocolos e que estendem a classe *Protocol*.

2.4. MDA

Model Driven Architecture [15] é uma arquitetura conceitual para desenvolvimento de software. Esta arquitetura visa separar decisões orientadas ao negócio de decisões de plataforma permitindo assim maior flexibilidade durante as fases de especificação e de desenvolvimento de sistemas. A idéia central de MDA é a criação de diferentes modelos com diferentes níveis de abstração e a ligação destes modelos através de refinamentos e transformações. Alguns desses modelos existirão independente da plataforma, outros serão específicos para uma plataforma.

Os três modelos a seguir compõem os principais elementos de MDA:

- CIM (*Computational Independent Model*): descreve o domínio da aplicação, sem ater aos detalhes da estrutura do sistema. É importante para a ajuda no entendimento do problema, como também é uma fonte de vocabulário a ser usado nos demais modelos.

- PIM (*Plataform Independent Model*): descreve o sistema, porém não apresenta os detalhes da tecnologia que será usada na implementação. O PIM oferece especificações formais da estrutura e funcionalidade do sistema, abstraindo-se de detalhes técnicos.

- PSM (*Plataform Specific Model*): combina a especificação do modelo PIM com detalhes específicos de uma determinada plataforma.

A transformação de modelos é a chave principal desta abordagem e consiste no processo de converter um modelo em outro modelo do mesmo sistema [15]. A idéia principal é construir modelos no seu mais alto nível de abstração e transformá-los em modelos com baixa abstração, de forma automática ou semi-automática, facilitando e tornando mais rápido o processo de desenvolvimento.

O MDA pode ser dividido em quatro grandes etapas:

1. Desenvolvimento do modelo CIM, modelo de mais alto nível de abstração.

2. Transformação do modelo CIM no modelo PIM, que é independente de qualquer tecnologia.

3. Transformação do modelo PIM em um ou mais modelos voltados para especificação do sistema em uma determinada tecnologia, ou seja, transformar o modelo PIM em modelo(s) PSM.

4. Transformação dos modelos PSM em código. Esta etapa é um processo direto, ou quase direto, pois o modelo PSM já possui as informações necessárias da tecnologia específica na qual o software será implementado.

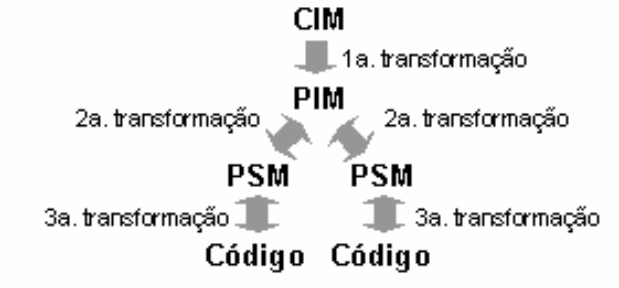

Figura 3 – O relacionamento entre CIM, PIM, PSM e código

Devido as suas características, pode-se dizer que a MDA oferece os seguintes benefícios [27]:

- produtividade: a transformação do PIM para o PSM precisa ser definida uma única vez e pode ser aplicada no desenvolvimento de diversos sistemas. Devido a este fato, tem-se uma redução no tempo de desenvolvimento.

- portabilidade: dentro de MDA a portabilidade é alcançada através do foco dado no desenvolvimento do PIM, que é independente de plataforma. Um mesmo PIM pode ser automaticamente transformado em vários PSMs de diferentes plataformas.

- interoperabilidade: diferentes PSMs gerados a partir de um mesmo PIM podem conter ligações entre eles, denominadas em MDA de pontes. Quando PSMs são gerados em diferentes plataformas, eles não podem se relacionar entre si. É necessário então transformar os conceitos de uma plataforma para conceitos da outra plataforma. MDA endereça este problema gerando não somente PSMs, como também as pontes necessárias entre eles. Uma ponte pode ser construída através das especificações técnicas das plataformas referentes aos modelos PSMs e de como os elementos existentes no modelo PIM foram transformados nos modelo PSMs.

2.5. XMI

O XML Metadata Interchange (XMI) [18] é um padrão utilizado pela OMG que permite expressar objetos utilizando a linguagem XML (Extensible Markup Language) [28], visando solucionar o problema de interoperabilidade através da definição de um padrão de codificação genérico e da definição de um padrão para esquemas conceituais.

O XMI não é uma extensão de XML e sim se baseia em XML. É uma especificação de como gerar linguagens XML adequadas para modelos de dados e como codificar esses meta-dados em documentos XML, assegurando que o compartilhamento de objetos entre ferramentas seja feito de forma consistente.

XMI é importante, pois o XML não é orientado a objetos e, portanto, é preciso ao utilizar XML para descrever modelos orientados a objetos, realizar um mapeamento de objetos para XML. Mais ainda, não existe uma maneira única de realizar este mapeamento e com isso diversas aplicações utilizam-se de diferentes formas de mapeamento para especificar o mesmo objeto. O XMI oferece modelos para assegurar que objetos sejam compartilhados de forma consistente. Uma aplicação que utiliza XMI pode trocar objetos com outras aplicações que também utilizam XMI. O XMI é uma ponte entre os objetos e XML, pois prevê um padrão para o mapeamento de objetos definidos em UML, por exemplo através de XML Document Type Definition (DTD).

A especificação XMI é composta de dois conjuntos de regras: um conjunto de regras para a produção de DTD e um conjunto de regras para construção de documentos XML. O primeiro conjunto de regras ensina como derivar a gramática da linguagem XML correspondente ao meta-modelo e são denominados DTD. Os DTDs são documentos que descrevem a gramática de uma linguagem baseada em XML. Portanto, descrevem regras para a construção do documento XML correspondente ao modelo.

Um DTD deve conter os seguintes elementos: declarações XML obrigatórias, cabeçalho XMI, declarações dos meta-dados de um modelo específico, declarações incrementais dos meta-dados de um modelo específico e declarações de extensões a meta-modelos.

O DTD pode ser usado com XMI para permitir que as informações presentes no meta-modelo sejam verificadas através da validação do XML, provendo meios pelos quais um XML possa ser validado sintaticamente, e em alguns casos, semanticamente. Embora algumas verificações possam ser feitas, é impossível utilizar somente esta validação para verificar se as informações que estão sendo transferidas satisfazem todas as regras presentes na semântica do meta-modelo.

Com base nas características e funções do XMI, pode-se dizer que o XMI oferece diversos benefícios. São eles [29]:

- Oferece um padrão de representação de objetos em XML, permitindo uma troca efetiva de objetos utilizando XML;

- Especifica como criar XML schemas através do modelo;

- Permite criar um documento XML simples e fazê-lo evoluir de acordo com a aplicação;

- Permite trabalhar com XML sem que se tenha um pleno conhecimento de XML;

- Permite modelagem com XML;

- Permite trabalhar com dados e meta-dados;

- Softwares que suportam XMI oferecem um alto nível de abstração que os elementos e atributos XML, visto que definem o mapeamento ente XML e objetos, sem que haja a necessidade de criar uma representação específica em XML dos objetos utilizados.

Conforme mencionado anteriormente, o XMI é uma ponte entre a modelagem orientada a objetos e XML. Atualmente, a especificação de XMI apresenta um mapeamento de objeto definido em UML para XML, através de uma estrutura de representação de modelos UML, conforme seu meta-modelo. Esta representação do meta-modelo de UML é apresentada através de um extenso DTD XML (UML DTD) que permite que os modelos sejam representados em um formato ASCII, facilmente trocados entre diferentes aplicações e legíveis por indivíduos técnicos.

Através destes DTDs padrões definidos em XMI, como é o caso do UML DTD, o XMI pode ser utilizado como um meio universal de troca de informação entre ferramentas de desenvolvimento orientadas a objeto. Com a sua utilização, é necessário apenas que cada ferramenta importe e exporte meta-dados em XMI, para que a troca de meta-dados entre as ferramentas seja possível. Deste modo, o XMI permite a troca de modelos entre ferramentas.

Devido a todas as suas características, o XMI pode ser aplicado na abordagem MDA, pelo fato de prover um padrão de representação para dados (e meta-dados), permitir a troca e o compartilhamento dos dados. Usando XMI, é possível gerar uma representação em XML dos modelos.