7
Conclusão e Trabalhos Futuros

Como um novo e poderoso paradigma para o design e a implementação de sistemas de software (Lind, 2001;Wooldridge et al., 2001), o SMA requer metodologias, linguagens de modelagem, plataformas de desenvolvimento e linguagens de programação que explorem seus benefícios e suas características próprias (Zambonelli et al. 2002). No entanto, diferentes metodologias, linguagens e plataformas para SMAs propõem conjuntos muito variados de abstrações. Normalmente é muito difícil entender a definição de cada abstração e os relacionamentos entre elas.

Nesse contexto, foi desenvolvido o *framework* conceitual TAO (Silva et al., 2003). A principal função do *framework* TAO é oferecer um *framework* conceitual para entender as abstrações e seus relacionamentos a fim de oferecer suporte ao desenvolvimento de SMAs de larga escala. O *framework* proposto traz à tona uma ontologia que conecta abstrações consolidadas, como objetos e classes, e outras abstrações como agentes, papéis e organizações, que são as bases da engenharia de software orientada a objetos e agentes. O principal conjunto de abstrações foi desenvolvido com base em nosso extensivo trabalho de investigação das metodologias com base em agentes e orientadas a objetos existentes, linguagens e teorias e em nosso trabalho experimental no desenvolvimento de muitos SMAs de larga escala. TAO define as entidades, relacionamentos e o comportamento independente do domínio freqüentemente descritos em SMAs.

Depois de analisar muitas das linguagens de modelagem para SMAs publicadas na literatura (Wagner et al., 2003; Odell et al., 2001; Depke et al., 2002), foi possível perceber que nenhuma delas contemplava todos os conceitos definidos em TAO, ou seja, elas não contemplavam muitos dos conceitos normalmente encontrados em SMAs. As linguagens de modelagem não modelam os aspectos estruturais (entidades e relacionamentos) tampouco os aspectos

dinâmicos (comportamento independente do domínio) normalmente descritos em SMAs e definidos em TAO. Algumas propostas descrevem um subconjunto dessas abstrações, e outras não modelam a interação entre as entidades definidas.

A fim de definir uma linguagem de modelagem para SMAs que contemple todos os conceitos descritos em TAO, propomos a linguagem de modelagem MAS-ML (Silva et al., 2004a,b,c). MAS-ML é uma linguagem de modelagem que estende UML com base na definição dos aspectos dinâmicos e estruturais apresentados em TAO. A extensão preserva todos os conceitos relacionados a objetos que constituem o metamodelo de UML enquanto inclui os conceitos relacionados a agentes descritos em TAO.

Os pontos fortes de MAS-ML são:

1. É possível representar os aspectos dinâmicos e estruturais de SMAs normalmente usados, conforme definidos em TAO. Usando os três diagramas estruturais e o diagrama de seqüência estendido, todas as entidades, propriedades, relacionamentos e interações descritos em TAO podem ser modelados.

2. Duas importantes características de SMAs podem ser modeladas usando os diagramas de seqüência de MAS-ML: concorrência e distribuição. Usando diagramas de seqüência, é possível modelar uma entidade que está exercendo diferentes papéis concorrentemente, diferentes entidades que estão executando em paralelo e diferentes entidades que estão executando em diferentes ambientes.

3. A nova linguagem de modelagem baseia-se em – e estende – uma linguagem conhecida como um padrão *de facto* para a modelagem orientada a objetos. Basear a criação de uma nova linguagem em uma linguagem padrão e conhecida diminui a curva de aprendizado para designers de software e reduz o risco da adoção da nova tecnologia (Odell et al., 2000; Bauer, 2002).

4. O design de seus modelos é orientado por uma abordagem de modelagem, ainda que MAS-ML seja um processo independente, como UML. Como a linguagem introduz muitos conceitos diferentes que devem ser modelados usando vários diagramas, é importante usar uma abordagem de modelagem para ajudar o designer.

5. A implementação das aplicações modeladas usando MAS-ML tem o suporte de um transformador que refina os modelos estruturais em código, conectando o nível de modelagem e o nível de implementação.

Os pontos fracos de MAS-ML são:

1. Como MAS-ML baseia-se em TAO para criar os diagramas de MAS-ML é necessário entender todos os conceitos definidos em TAO. Como há muitas novas abstrações no *framework* conceitual, entender todas as definições, relacionamentos e diferentes tipos de interações é uma tarefa muito complexa.

2. Os diagramas de MAS-ML podem ficar muito grandes quando sistemas grandes são modelados e todos os elementos do diagrama são expressos usando a forma completa (sem omitir qualquer compartimento).

3. A identificação das entidades em um diagrama de seqüência pode se tornar uma tarefa muito complicada, caso sejam usados *pathnames* completos. Os *pathnames* completos podem ficar longos, o que dificulta sua leitura.

4. MAS-ML não oferece um diagrama de alto nível para modelar planos e ações. O único diagrama usado para modelar planos e ações é o diagrama de seqüência. Normalmente, diagramas de seqüência são usados para modelar os detalhes das interações entre muitas entidades. Um diagrama de alto nível, como o diagrama de atividade, pode ser usado para modelar planos, ações e eventos que disparam as ações sem detalhar a execução das mesmas. As execuções são modeladas pelos diagramas de seqüência.

Nas próximas Seções, apresentamos as principais contribuições desta tese e futuros trabalhos propostos.

7.1. Análise de Nossas Principais Contribuições

As principais contribuições desta tese são estas:

1. O *framework* conceitual TAO
2. As extensões ao *framework* conceitual TAO
3. A linguagem de modelagem MAS-ML
4. A abordagem de modelagem
5. A arquitetura abstrata orientada a objetos para SMAs
6. A gramática de MAS-ML
7. O transformador MAS-ML2Java

7.1.1. O *Framework* Conceitual TAO

O *framework* conceitual TAO define os aspectos estruturais dos SMAs, ou seja, o *framework* conceitual descreve as entidades que são normalmente

apresentadas nos SMAs, suas propriedades e um conjunto de relacionamentos que podem ser usados para unir essas entidades. Neste documento, apresentamos uma versão revisada e estendida dos aspectos estruturais de TAO. Foi publicada uma versão detalhada das características estruturais contidas em TAO em (Silva et al., 2003).

O *framework* conceitual TAO foi criado depois de uma revisão exaustiva das teorias, metodologias e métodos para SMAs. Sentimos a necessidade de um *framework* conceitual que definisse completamente as abstrações normalmente usadas para modelar SMAs e seus relacionamentos.

O benefício desse *framework* é oferecer suporte ao desenvolvimento de novas tecnologias, métodos, linguagens e arquiteturas com base nos conceitos essenciais definidos e relacionados no *framework*. Cada conceito é visto como uma abstração candidata para linguagens de modelagem, metodologias e ambientes com suporte que deve ser aplicada em diferentes fases do desenvolvimento de SMAs.

7.1.2. As Extensões ao *Framework* Conceitual TAO

Apesar de TAO definir diferentes abstrações para SMAs e seus relacionamentos, ele não descreve as interações entre entidades, ou seja, ele não descreve os aspectos dinâmicos entre entidades. O *framework* conceitual TAO foi estendido para definir as interações independentes do domínio que podem ocorrer entre as entidades apresentadas nos SMAs (Silva et al., 2004b). As interações independentes do domínio foram definidas com base nos relacionamentos independentes do domínio definidos nos aspectos estruturais de TAO. A análise dos aspectos dinâmicos oferece feedback para estender o diagrama de seqüência de UML.

7.1.3. A Linguagem de Modelagem MAS-ML

MAS-ML é uma linguagem de modelagem que estende UML com base no *framework* conceitual TAO estendido. O principal objetivo da linguagem de modelagem MAS-ML é tratar das características particulares dos SMAs que não são tratadas em UML e não são tratadas de forma satisfatória nas propostas disponíveis na literatura (Capítulo 2).

Foi feita uma extensão conservativa a UML incorporando a definição das entidades, propriedades, relacionamentos e interações identificados em TAO estendido. Ao incorporar as abstrações de TAO, o metamodelo de UML foi estendido pela criação de novas metaclasse e estereótipos. Além disso, novos diagramas foram definidos, diagramas existentes foram estendidos e novos elementos do diagrama foram criados.

MAS-ML define três diagramas estruturais a fim de modelar os aspectos estruturais e dinâmicos de SMAs criando dois novos diagramas e estendendo o diagrama de classes definido em UML. Ela também estende o diagrama de seqüência a fim de modelar os aspectos dinâmicos de SMAs. Usando o conjunto de diagramas definido por MAS-ML, é possível modelar todos os aspectos estruturais e dinâmicos do metamodelo de TAO.

7.1.4. A Abordagem de Modelagem

O objetivo da abordagem de modelagem é ajudar o designer a modelar seu SMA usando MAS-ML. Essa abordagem descreve o que é necessário ser especificado a fim de modelar completamente os diagramas de MAS-ML. Diferentes diagramas modelam diferentes aspectos da mesma entidade. Ademais, a abordagem de modelagem orienta o designer descrevendo a seqüência de identificação de entidades e definindo as informações que devem ser fornecidas ao identificar cada uma delas.

É importante ressaltar que, da mesma forma que UML, a linguagem de modelagem MAS-ML não possui um modelo de processo associado. A abordagem de modelagem proposta é apenas uma indicação de como nossa linguagem de modelagem proposta pode ser mais facilmente aplicada em aplicações complexas.

7.1.5. A Arquitetura Abstrata OO para SMAs

Uma arquitetura abstrata OO para SMAs foi desenvolvida para conduzir a implementação de SMAs. A arquitetura abstrata é gerada a partir do metamodelo de MAS-ML de forma que possa ser usada para implementar os modelos desenvolvidos usando a linguagem de modelagem MAS-ML. Ela também pode

ser usada por qualquer SMA baseado no metamodelo de TAO porque os conceitos baseados em agentes do metamodelo de MAS-ML são derivados de TAO.

A arquitetura abstrata OO contempla todas as abstrações definidas no metamodelo de MAS-ML. A arquitetura abstrata representa as entidades de MAS-ML, suas propriedades e relacionamentos independentes do domínio pelo uso de classes OO. Para cada entidade de MAS-ML, foi definido um conjunto de classes relacionadas a fim de representar entidades abstratas e suas propriedades. Além do mais, os relacionamentos independentes do domínio entre as entidades de MAS-ML (relacionamentos *play*, *inhabit* e *ownership*) são mapeados em relacionamentos OO e, como conseqüência, a atributos das classes abstratas.

A arquitetura abstrata é usada pelo transformador MAS-ML2Java como a base da geração de código. As regras para entidades independentes do domínio definidas pelo transformador geram um conjunto de classes, atributos e métodos que compõem a arquitetura abstrata orientada a objetos para SMAs.

7.1.6. A Gramática de MAS-ML

A fim de formalizar a sintaxe dos aspectos estruturais de MAS-ML, foi descrita uma gramática de MAS-ML. A gramática é descrita usando a linguagem de programação TXL cuja notação é semelhante a Backus-Nauer Form (BNF).

A gramática de MAS-ML é definida com base no metamodelo de MAS-ML. Usando a gramática de MAS-ML, é possível descrever textualmente todas as informações apresentadas nos diagramas estruturais de uma aplicação. É possível descrever todas as entidades, suas propriedades e relacionamentos.

7.1.7. O Transformador MAS-ML2Java

O transformador MAS-ML2Java foi criado a fim de gerar código a partir de modelos de MAS-ML. A entrada do transformador é uma descrição textual de uma aplicação usando a gramática de MAS-ML. Ele se aplica a um conjunto de regras de entrada e gera uma saída que também seja compatível com a gramática.

O processo de transformação baseia-se na arquitetura abstrata orientada a objetos para SMAs. O transformador gera os módulos de componente representados na arquitetura abstrata e os componentes originados a partir dos modelos de MAS-ML relacionados ao domínio da aplicação. Os componentes

originados a partir dos modelos de MAS-ML especializam os componentes definidos na arquitetura abstrata.

Ao desenvolver o transformador MAS-ML2Java, não pretendemos criar uma ferramenta que gere automaticamente código a partir de todos os modelos de MAS-ML. Nosso objetivo era gerar um design OO em que o designer de MAS-ML pudesse facilmente identificar as entidades, suas propriedades e relacionamentos, conforme ilustrado nos modelos MAS-ML. Esse design OO deve ser tão simples quanto possível, ou seja, não ter mais classes do que o conjunto mínimo de classes usado para representar as abstrações.

7.2. Trabalhos Futuros

Propomos os trabalhos futuros a seguir:

1. Desenvolver uma ferramenta de modelagem MAS-ML
    1. Estender a gramática de MAS-ML para incorporar aspectos dinâmicos
    2. Estender a arquitetura abstrata orientada a objetos
    3. Criar MAS-ML DTD com base na gramática de MAS-ML estendida
    4. Transformar XMIs com base em MAS-ML DTD em XMIs com base em UML DTD
2. Formalizar a semântica de MAS-ML
3. Analisar outros diagramas de UML

7.2.1. Desenvolvimento de uma Ferramenta de Modelagem MAS-ML

Uma ferramenta de modelagem MAS-ML deve ajudar o desenvolvedor de sistemas multiagentes durante a modelagem e a implementação de suas aplicações. Os objetivos da ferramenta de modelagem são simplificar e acelerar os designs de diagramas de MAS-ML.

Ela deve ser composta de um ambiente para a modelagem de diagramas de MAS-ML e uma máquina geradora a partir de diagramas de MAS-ML em código. Como MAS-ML estende UML, deve ser possível criar qualquer diagrama de UML usando o ambiente fornecido pela ferramenta MAS-ML. Uma ferramenta de modelagem MAS-ML poderia ser gerada com base em uma ferramenta de UML.

Em nosso trabalho, definimos uma gramática de MAS-ML a fim de descrever modelos de MAS-ML usando uma notação textual. Ademais, também desenvolvemos um transformador que gerava código Java a partir da descrição textual de uma aplicação. Contudo, a gramática de MAS-ML não descreve as informações presentes em diagramas de seqüência e, portanto, o código gerado não pode usar essas informações.

Ferramentas UML, como Together, também geram código a partir de uma descrição textual. A linguagem XMI é usada para descrever modelos de aplicação com base em UML DTD. Usando UML DTD, é possível descrever todas as informações representadas em todos os diagramas de UML. Os transformadores são aplicados a XMI de uma aplicação a fim de gerar código a partir de diferentes linguagens de programação.

Para criar uma ferramenta de MAS-ML, deve ser usada a abordagem a seguir:

1. A gramática de MAS-ML deve ser estendida a fim de descrever os aspectos dinâmicos.
2. A arquitetura abstrata orientada a objetos deve ser estendida.
3. A descrição textual dos modelos de MAS-ML deve ser descrita usando XMI. Deve ser criada MAS-ML DTD.
4. Deve ser criado um transformador de XMIs baseado em MAS-ML DTD em XMIs baseados em UML DTD.

Se as tarefas descritas anteriormente forem bem-sucedidas, a ferramenta de MAS-ML será capaz de descrever as informações representadas em todos os modelos de MAS-ML usando XMI e gerar código a partir das informações descritas em XMI para diferentes linguagens de programação. A Figura 118 ilustra a ferramenta da MAS-ML no contexto de nosso trabalho atual e das ferramentas de UML disponíveis.

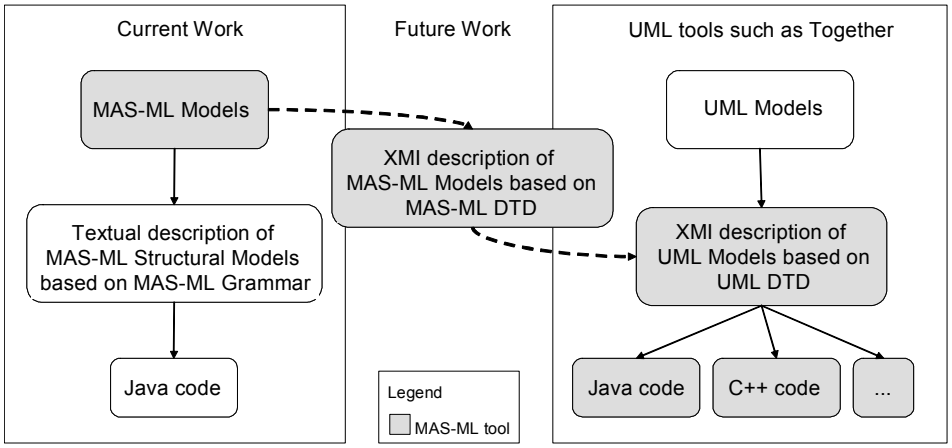

Figura 118 – A ferramenta de MAS-ML.

7.2.1.1. Estender a Gramática de MAS-ML para Incorporar Aspectos Dinâmicos de SMAs

A gramática de MAS-ML usada pelo transformador MAS-ML2Java apenas define as abstrações representadas nos modelos estruturais de MAS-ML. A fim de gerar código a partir de todos os diagramas de MAS-ML, é necessário estender a gramática para incorporar os aspectos dinâmicos associados ao diagrama de seqüência de MAS-ML. A gramática de MAS-ML estendida deve descrever, por exemplo, as instâncias de MAS-ML, os atributos de mensagens e métodos, os planos e as ações sendo executadas e os estereótipos associados ao papel.

7.2.1.2. Estender a Arquitetura Abstrata

O código gerado a partir do transformador de MAS-ML2Java baseia-se na arquitetura abstrata descrita na Seção 5.4.1. A arquitetura abstrata representa as entidades de MAS-ML, suas propriedades e relacionamentos por meio do uso de classes OO mas não descreve nenhum componente relacionado às interações das entidades. Ela deve ser estendida a fim de incorporar mecanismos relacionados a aspectos dinâmicos de SMAs.

A fim de criar uma arquitetura abstrata que contemple todos os aspectos dinâmicos e estruturais de SMAs, a arquitetura abstrata da FIPA (FIPA Architecture, 2004) poderia ser estendida. Essa arquitetura já define alguns mecanismos abstratos para gerenciar diferentes meios de transporte de mensagens, diferentes linguagens de comunicação de agente, diferentes linguagens de conteúdo e para localizar agentes e serviços por meio de serviços de diretório.

7.2.1.3. Criar MAS-ML DTD com Base na Gramática de MAS-ML Estendida

Enquanto estiver usando a ferramenta, o desenvolvedor deve ser capaz de trocar seus modelos e implementações. A fim de oferecer modelos intercambiáveis, XMI pode ser usado para descrevê-los. O formato XMI especifica um modelo intercambiável de informações abertas que é estendido para oferecer aos desenvolvedores que estão trabalhando com a tecnologia de objetos a capacidade de trocar os dados de programação de forma padronizada. O uso do formato traz consistência e compatibilidade às aplicações.

Para usar XMI para descrever os modelos de MAS-ML, é necessário definir MAS-ML DTD. MAS-ML DTD deve ser definido com base na gramática de MAS-ML estendida. Como MAS-ML estende UML, UML DTD poderia ser estendido para criar MAS-ML DTD.

Com base em MAS-ML DTD, as informações representadas em todos os modelos de MAS-ML poderiam ser descritas em XMI. A ferramenta de MAS-ML poderia exportar os modelos de MAS-ML de uma aplicação criando uma instância de XMI e poderia importar uma instância de XMI gerando modelos de MAS-ML.

7.2.1.4. Transformar XMIs com Base em MAS-ML DTD em XMIs com Base em UML DTD

Conforme mencionado anteriormente, ferramentas de UML usam diferentes transformadores para gerar código a partir de XMIs para diferentes linguagens de programação. XMI usado pelas ferramentas de UML baseiam-se em UML DTD. Um dos benefícios de ter um transformador de XMIs baseado em MAS-ML DTD (ou MAS-ML XMIs) em XMIs com base em UML DTD (ou UML XMIs) é a reutilização dos transformadores disponíveis para UML XMIs a fim de gerar código. Para gerar código a partir de MAS-ML, XMIs teriam de ser desenvolvidos. Além disso, UML XMIs poderiam ser usados a fim de gerar modelos de UML independentes da linguagem de programação orientada a objetos usada para gerar o código. Para isso, MAS-ML deve ser descrito usando MAS-ML XMI e MAS-ML XMI deve ser transformado em UML XMI. Usando UML XMI, podem ser gerados modelos de UML. O desenvolvedor poderia analisar e modificar os modelos de UML antes de gerar código.

O transformador de MAS-ML XMIs em UML XMIs pode ser criado com base na arquitetura abstrata estendida e nas regras para entidades dependentes do domínio definidas pelo transformador MAS-ML2Java (Capítulo 5). Ademais, outras regras devem ser definidas a fim de transformar as informações relacionadas ao diagrama de seqüência de MAS-ML em informações orientadas a objetos.

7.2.2. Formalizar a Semântica de MAS-ML

Alguns dos principais benefícios de uma semântica precisa para a linguagem de modelagem são clareza, equivalência e consistência, estendibilidade, refinamento e prova (Evans et al., 1999). Como no caso de UML, a semântica da linguagem de modelagem MAS-ML não foi ainda formalizada. A fim de formalizar completamente a semântica de MAS-ML, será necessário formalizar a semântica de UML, porque MAS-ML estende UML. Algumas iniciativas como (France et al., 1998; Evans et al., 1998; Barbier et al., 2003) tentaram formalizar UML. Há ainda um grupo chamado *Precise* UML (Precise UML, 2004) cujo objetivo é esclarecer e tornar a semântica de UML precisa. Entretanto, nenhuma dessas iniciativas até agora conseguiu formalizá-la completamente.

7.2.3. Analisar Outros Diagramas de UML

Ao criar a linguagem MAS-ML, apenas a classe de UML e os diagramas de seqüência foram analisados e estendidos a fim de incorporar as características de SMAs. A fim de modelar diferentes visões de uma aplicação de SMAs e para representar melhor algumas de suas características, devem ser analisados outros diagramas de UML, que provavelmente também devem ser estendidos.

O diagrama de objeto de UML modela as instâncias de classes. Esse tipo de diagrama é usado para descrever o sistema em um determinado momento. O diagrama de objeto de UML poderia ser estendido para representar qualquer instância de SMA e seus relacionamentos.

Outro exemplo é o diagrama de atividade de UML. Os diagramas de atividade de UML documentam a lógica de uma única operação ou método, um único caso de uso ou o fluxo de um processo de negócios. O diagrama de

atividade de UML poderia ser estendido para também documentar a lógica de um plano e a lógica de uma ação.