2
Conceitos Básicos

Este capítulo apresenta os conceitos básicos necessários para um melhor entendimento desta dissertação. É apresentado brevemente o conceito de governança de sistemas multi-agentes, juntamente com uma abordagem para governança baseada em leis, na qual o trabalho se baseia.

2.1.
Governança de SMA

Sistemas multi-agentes são compostos por vários agentes autônomos que interagem entre si para alcançar seus objetivos (Wooldridge et al., 2001). Apesar da autonomia, um agente deve seguir normas ou convenções do sistema em que está participando para atingir os objetivos globais do sistema. Algumas abordagens regulam o comportamento dos agentes no próprio código fonte do agente. No entanto, em sistemas multi-agentes abertos, onde os agentes podem entrar e sair de maneira independente e existe pouco controle sobre o comportamento dos agentes (Agha, 1997), outra abordagem faz-se necessária.

Agentes devem se comunicar para atingir seus objetivos, seja em cooperação, seja em competição (Jennings et al., 1998). Um controle sobre o comportamento dos agentes pode ser feito a partir da regulação de suas interações. De fato, as abordagens de leis podem ser usadas para minimizar esses riscos, que poderiam gerar falhas nos sistemas (Minski et al., 2000). Essas abordagens permitem separar a especificação das leis da especificação dos agentes, levando a um mecanismo externo de controle sobre o comportamento dos agentes, como se faz necessário em sistemas multi-agentes abertos.

Diversas abordagens de leis foram propostas visando minimizar os riscos, decorrentes da imprevisibilidade, que levariam a falhas nos sistemas. Destacam-se (Castelfranchi et al., 2000), (Dignum, V., et al., 2001a), (Dignum, F., 2001b), (Esteva et al., 2001), (Esteva, 2003), (Kollingbaum et al., 2003a), (Minsky et al.,

2000), (Paes, 2005a) e (Zambonelli, 2002). Esta dissertação se baseia em (Paes, 2005a), que será explicada na seção a seguir.

2.2. Abordagem de Leis XMLaw

Essa abordagem propõe um modelo conceitual para especificar a forma como as interações dos agentes são reguladas, uma linguagem declarativa para a instanciação do modelo conceitual – chamada XMLaw – e uma infra-estrutura de software mediadora – chamada M-Law – para o monitoramento das interações e aplicação da lei (Paes 2005a).

O diferencial dessa linguagem é tanto fornecer uma base conceitual abstrata, que coloca as leis como entidades de primeira ordem, levando em conta o tempo como entidade ativa nas interações, quanto prover uma infra-estrutura de software que permita a sua implementação.

O modelo conceitual relaciona conceitos necessários para especificar as leis de regulação de interações de agentes em sistemas multi-agentes abertos. Ele permite especificar os papéis de agentes que irão interagir dentro das cenas (abstrações que auxiliam a organizar e subdividir as interações), os protocolos de interação, as possíveis mensagens trocadas entre os agentes, as normas, relógios e restrições, reguladores da atuação dos agentes, e as ações, que permitem executar código de recuperação em casos de falha.

Esses elementos podem estar relacionados em termos de estrutura, como é o caso no qual uma norma pertence a uma cena, ou dinamicamente, como a ativação de uma ação, chamada por um relógio que teve o prazo expirado. Esse relacionamento dinâmico é feito por meio de eventos, ou seja, os elementos podem gerar eventos, perceber a ocorrências desses e serem ativados ou desativados como conseqüência da chegada de eventos.

O desenvolvimento de sistemas multi-agentes utilizando essa abordagem propõe que a regulação das interações seja feita de maneira externa à implementação do sistema em si. O desenvolvedor das leis pode não ser a mesma pessoa que desenvolve o sistema ou que desenvolve os agentes. Seu trabalho é especificar a lei do sistema em um arquivo XML, que será lido pela infra-estrutura de software provida por essa abordagem.

O arquivo com a especificação da lei segue uma linguagem declarativa baseada em XML, chamada XMLaw, que permite representar os elementos do modelo conceitual.

A infra-estrutura de software mediadora (M-Law) provê suporte e monitoramento da lei especificada no arquivo XMLaw. Ela intercepta todas as mensagens trocadas pelos agentes, verifica a conformidade com a lei e encaminha as mensagens aos destinatários, caso sejam permitidas, ou as bloqueia, caso contrário (Figura 2-1). Não é relevante para este trabalho detalhar como o sistema mediador realiza essa tarefa. Maiores informações podem ser obtidas em (Paes, 2005a).

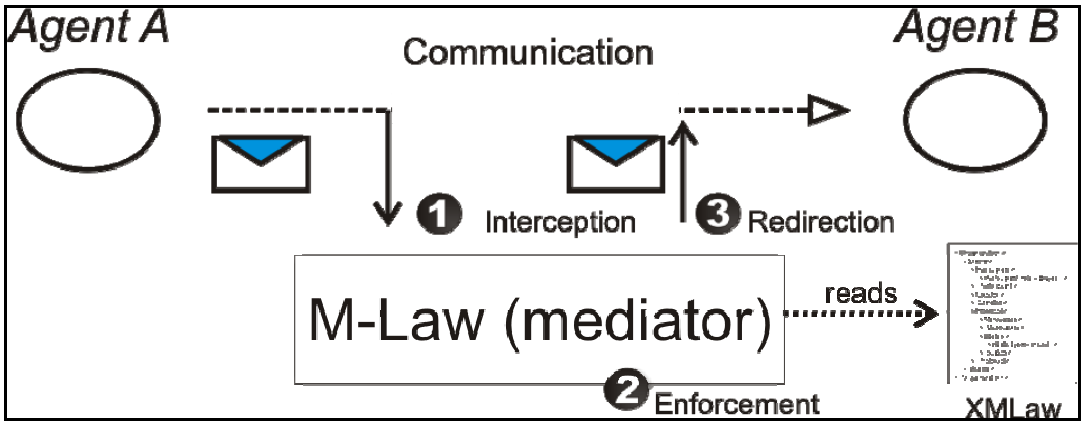

Figura 2-1 - Arquitetura M-Law (Paes, 2005c).

2.2.1. Modelo Conceitual da Abordagem de Leis XMLaw

A seguir, o modelo conceitual da abordagem de leis (Figura 2-2) será apresentado, juntamente com exemplos de representação de instâncias em código no formato XMLaw, conforme (Paes, 2005a). A gramática atualizada de XMLaw encontra-se no Anexo A, onde podem ser vistos maiores detalhes sobre a sintaxe dos modelos gerados.

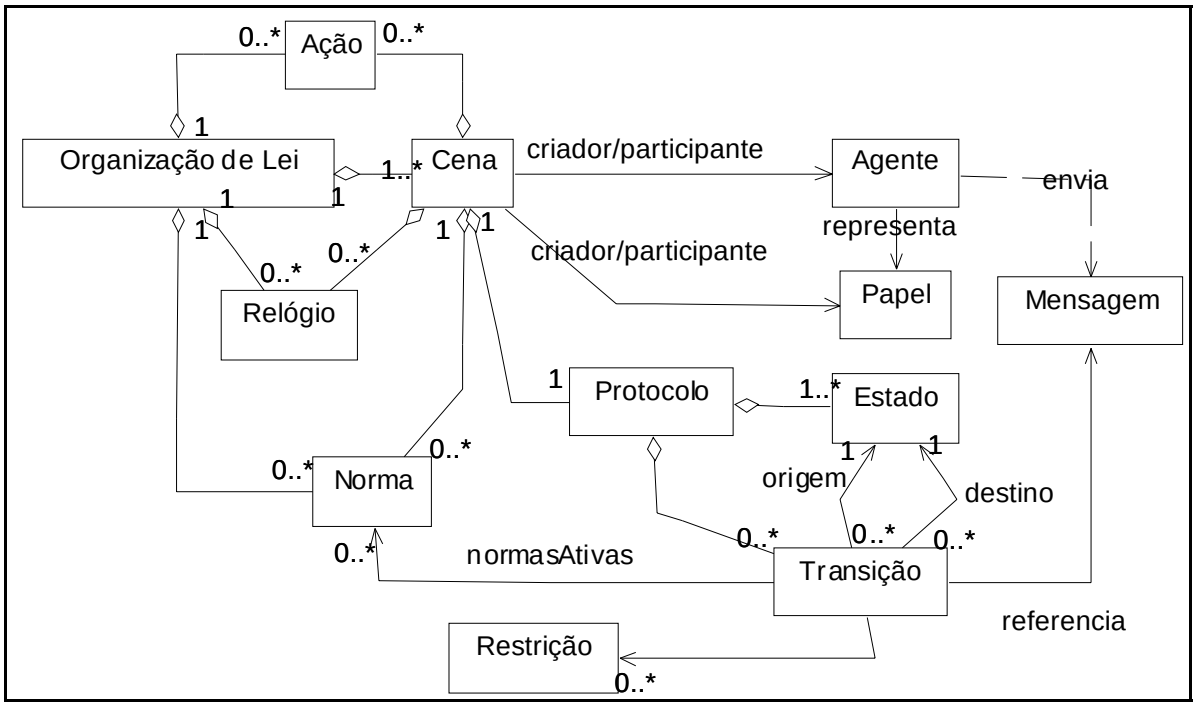

Figura 2-2 - Modelo Conceitual da Abordagem de Leis (Paes, 2005b)

Em primeiro lugar é importante destacar que todo elemento possui um atributo id:

- id – representa o identificador do elemento;

Os demais elementos são o seguintes:

Organização de Lei (*LawOrganization*)

O elemento *Organização de Lei* representa e especifica as leis de uma organização multi-agente. Esse elemento pode conter *Cenas*, *Papéis*, *Normas*, *Relógios* e *Ações*, e deve ser considerado como o ponto de entrada na especificação, tendo como função principal a conjunção dos demais elementos previstos no modelo.

Atributo da *Organização de Lei*:

- nome (*name*) – o nome da organização;

A Figura 2-3 mostra um exemplo de representação de uma *Organização de Lei* em XMLaw.

```
<Laws xmlns:xi="http://www.w3.org/2001/XInclude">
        <LawOrganization id="Id-da-organizacao" name="Nome-da-organizacao">
                <Role />
                ...
                <Scene/>
                ...
                <Action/>
                ...
                <Clock/>
                ...
                <Norm/>
                 ...
        </LawOrganization>
</Laws>
```

Figura 2-3 - Exemplo de XMLaw da *Organização de Lei*

Papel (*Role*)

O elemento *Papel* representa como um agente é visto no nível organizacional, ou seja, qual o seu papel na organização. Ao interagir, um agente precisa escolher um papel ao qual ele se associará, embora possa estar associado a vários papéis simultaneamente.

A Figura 2-4 mostra um exemplo de representação de um *Papel* em XMLaw.

```
<Role id="Id-do-Papel" />
```

Figura 2-4 Exemplo de XMLaw do *Papel*

Norma (*Norm*)

O elemento *Norma* descreve quais comportamentos dos agentes são permitidos, quais são obrigatórios e quais são proibidos. As *Normas* são geralmente adquiridas pelos agentes durante o decorrer das interações, representando compromissos adquiridos e cumpridos. Por exemplo, em um processo de negociação, um agente pode assumir o compromisso (obrigação) de pagar por uma mercadoria negociada e, enquanto essa obrigação não for cumprida, o agente fica impedido de participar de novas negociações.

Uma *Norma* pode ser do tipo *permissão*, *obrigação* e *proibição* e contém um conjunto de condições de ativação e desativação. Essas condições, baseadas em eventos, especificam quando uma *Norma* deve ser ativada e quando ela deve ser desativada.

Normas podem ser aplicadas dentro da *Organização de Lei* ou restritas a uma *Cena* específica. Uma *Norma* pode ser atribuída a um *Papel de* agente específico através do seu sub-elemento *Assinatura (Assignee)*, que indica a quem a Norma se restringe. Nesse caso, quando a *Norma* é ativada, ela é válida apenas para esse *Papel* de agente específico. Além disso, uma *Norma* também pode estar associada a uma ou mais *Restrições*.

Atributo da *Norma*:

- tipo (*type*) – permissão (*permition*), obrigação (*obligation*) ou proibição (*prohibition*);

A Figura 2-5 mostra um exemplo de representação de uma *Norma* em XMLaw.

```
<Norm type="permission" id="id-da-norma">
    <Assignee role-ref="id-do-papel-possuidor-da-norma" role-instance="?"/>
    <Activations>
        <Element ref="ref-elemento-ativador" event-type="tipo-de-evento-ativador"/>
                ...
    </Activations>
    <Deactivations>
        <Element ref=" elemento-desativador" event-type="tipo-de-evento-desativador"/>
                ...
    </Deactivations>
</Norm>
```

Figura 2-5 - Exemplo de XMLaw da *Norma*

Restrição (*Constraint*)

Restrições permitem o uso de código Java (Java, URL, 2007) para desempenhar validações complexas em *Mensagens* trocadas pelos agentes e em *Normas*. Uma validação poderia ser a indentificação de quais são os valores permitidos para determinadas partes da informação contida na *Mensagem*. O elemento *Restrição* é especificado utilizando-se um componente em Java e a sua definição.

Atributos da *Restrição*:

- classe (*class*) – identifica a classe do objeto que deve ser executado;
- tipo (*type*) – pode ser *abstract*, *completes* ou *extends*;

A Figura 2-6 mostra um exemplo de representação de uma *Restrição* em XMLaw.

```
<Norm >
    ...
        <Constraints>
                <Constraint id="id-da-restricao" class="sma.lei.Restricao">
                        <Semantic>Texto explicativo</Semantic>
                </Constraint>
                        ...
        </Constraints>
</Norm>
```

Figura 2-6 - Exemplo de XMLaw da *Restrição*

Relógio (*Clock*)

O elemento *Relógio* permite especificar os aspectos temporais da lei. Existem dois tipos de *Relógio*, o *regular* (*once*), que gera apenas um evento de "tempo esgotado" (*time-out)* após decorrido um intervalo de tempo, e o *periódico (periodic)*, que continuamente gera eventos "alarme de relógio" (*clock_tick*) a cada intervalo de tempo específico. Esses eventos podem ser associados à ativação ou desativação de *Normas*, à ativação de *Ações* ou até à ativação de outros *Relógios*. Assim como a *Norma*, o *Relógio* contém um conjunto de condições de ativação e desativação baseados em eventos, que especifica quando o *Relógio* deve ser ativado e quando ele deve ser desativado.

Relógios podem ser válidos dentro da *Organização de Lei* ou restritos a uma *Cena* específica.

Atributos do *Relógio*:

- tipo(*type*) – regular (*once*) ou periódico (*periodic*);
- período de alarme (*tick-period*);

A Figura 2-7 mostra um exemplo de representação de um *Relógio* em XMLaw.

```
            <! 1s = 1.000ms>
<Clock id="id-relogio" type="once" tick-period="1000">
      <Activations>
            <Element ref="ref-elemento-ativador" event-type="tipo-de-evento-ativador"/>
                  ...
      </Activations>
      <Deactivations>
            <Element ref=" elemento-desativador" event-type="tipo-de-evento-desativador"/>
                  ...
     </Deactivations>
</Clock>
```

Figura 2-7 - Exemplo de XMLaw do *Relógio*

Ação (*Action*)

O elemento *Ação* permite a ativação, através de eventos, de código Java (Java, URL, 2007), que pode ser usado para a recuperação automática no sistema em caso de falhas. Recuperação automática pode ser definida como sendo a habilidade de um sistema automaticamente detectar, diagnosticar e reparar problemas de software e hardware (Kephart et al., 2003).

Assim como *Normas* e *Relógios*, a *Ação* é ativada através de eventos, gerados por outros elementos de lei. A diferença é que as *Ações* não podem ser desativadas. As *Ações* podem ser válidas dentro da *Organização de Lei* ou restritas a uma *Cena* específica.

Atributos da *Ação*:

- classe (*class*) – identifica a classe do objeto que deve ser executado;
- tipo (*type*) – pode ser *abstract*, *completes* ou *extends*;

A Figura 2-8 mostra um exemplo de representação de uma *Ação* em XMLaw.

```
<Action id="id-acao" class="sma.lei.Acao">
        <Element ref="ref-elemento-ativador" event-type="tipo-de-evento-ativador"/>
</Action>
```

Figura 2-8 - Exemplo de XMLaw da *Ação*

Cena (*Scene*)

O elemento *Cena* permite a contextualização e organização das interações. Uma *Cena* deve especificar um conjunto de *Papéis* de agentes que estarão

interagindo em um contexto específico, um *Protocolo* de interação, organizando essa interação, e possivelmente um conjunto de *Normas*, *Ações* e *Relógios*. Esses elementos compartilham um contexto comum de interação definido pela *Cena*. Isso significa que uma *Norma* definida no contexto de uma *Cena* é somente visível naquela *Cena*.

Toda *Cena* deve ter um conjunto de *Criadores* e de *Participantes*. Um *Criador* representa o *Papel* do agente que pode criar a *Cena*. Um *Participante* representa o *Papel* do agente que participa da *Cena*, limitado à entrada em um conjunto específico de *Estados* do *Protocolo*.

Criadores e *Participantes* podem estar associados a um conjunto de *Normas Ativas*, que indicam quais *Normas* devem estar ativas para que esses *Papéis* possam criar ou entrar na *Cena*, respectivamente.

Atributo da *Cena*:

- tempo de vida (*time-to-live*) – prazo de duração máxima da *Cena*;

Atributo do *Criador* da *Cena*:

- referência de *Papel* (*role-ref*) – representa a referência ao *Papel* de agente *criador*;

Atributos do *Participante* da *Cena*:

- referência de *Papel* (*role-ref*) – representa a referência ao *Papel* de agente *participante*;
- limite (*limit*) – representa o número máximo de agentes no *Papel* em questão que podem participar da *Cena*;

Além desses atributos, o *Participante* da *Cena* deve possuir um conjunto de referências a *Estados* do *Protocolo* da sua *Cena*, que indica em quais *Estados* podem entrar.

A Figura 2-9 mostra um exemplo de representação de uma *Cena* em XMLaw.

```
<Scene id="id-da-cena" time-to-live="10000">
        <Creators>
                <Creator role_ref="id-do-papel"/>
                        ...
        </Creators>
        <Protocol/>
        <Entrance>
                <Participant role_ref="id-do-papel" limit="1">
                        <State ref="id-do-estado-de-entrada"/>
                </Participant>
                        ...
        </Entrance>
        <Action/>
        ...
        <Clock/>
        ...
        <Norm/>
         ...
</Scene>
```

Figura 2-9 - Exemplo de XMLaw da *Cena*

Protocolo (*Protocol*), Estado (*State*) e Transição (*Transition*)

O elemento *Protocolo* representa um padrão de interação entre os agentes; uma máquina de estados não determinística (Menezes, 1997) onde a *Transição* de um *Estado* é disparada pela ocorrência de evento. Cada *Estado* representa um ponto na execução do *Protocolo*, podendo ter diferentes significados, conforme o *Protocolo* de interação. O elemento *Transição* é a conexão entre os *Estados*. Protocolos de interação representam padrões de interação entre os agentes, definindo quais são as interações válidas e inválidas, isso é, quais *Mensagens* são válidas em determinado momento (*Estado* do *Protocolo*). O elemento *Protocolo* deve sempre estar associado a uma e somente uma *Cena.*

Atributos do *Estado*:

- tipo (*type*) – *Estado Inicial* (*initial*), *Estado de Execução* (*execution*), *Estado Final de Sucesso* (*success*) ou *Estado Final de Falha* (*failure*);
- rótulo (*label*) – representa a descrição da semântica do *Estado*.

Uma *Transição* pode ser ativada com a chegada de um evento de ativação, sendo que essa ativação pode estar condicionada a uma ou mais *Normas* estarem *ativas*, assim como ao resultado da execução de uma ou mais *Restrições*. Além disso, a ativação de uma transição envia um evento de “ativação de transição” (*transition-ativation*), que pode ser responsável pela ativação ou desativação de outro elemento de lei, como uma *Norma* ou *Relógio.*

Atributos da *Transição*:

- origem (*from*) – uma referência ao *Estado* de origem da *Transição*;

- destino (*to*) – uma referência ao *Estado* de destino da *Transição*;

- referência (*ref*) – uma referência ao elemento de lei que gerou o evento de ativação da *Transição*;

- tipo de evento (*event-type*) – o tipo do evento de ativação da *Transição*;

A Figura 2-10 mostra um exemplo de representação de um *Protocolo* em XMLaw.

```
<Protocol id="id-do--protocolo">
    <Messages/>
    <States>
        <State id="s1"  type="initial" label="semantica de s1"/>
        <State id="s2"  type="execution" label="semantica de s2"/>
                ...
    </States>
    <Transitions>
        <Transition id="t1" from="s1" to="s2" ref="id-elemento" event-type="eventol">
                <Constraints>
                        ...
                </Constraints>
                <ActiveNorms>
                    <Norm ref="id-da-norma"/>
                        ...
                </ActiveNorms>
        </Transition>
        ...
    </Transitions>
</Protocol>
```

Figura 2-10 - Exemplo de XMLaw de *Protocolo*

Mensagem (*Message*)

O elemento *Mensagem* permite especificar quais são as mensagens que os agentes podem trocar. Esse elemento possui atributos que especificam o padrão, ou formato das mensagens permitidas.

Atributos da *Mensagem*:

– performativa *(performative)* – especifica a intenção do agente em uma interação com outros agentes. A comunicação entre os agentes é geralmente baseada na teoria de atos da fala (Searle, 1969), permitindo que linguagens de comunicação entre agentes, como

(ACL, 2002) e (Finin et al., 1994), forneçam um vocabulário para os agentes expressarem suas intenções;

– conteúdo (*content*) – contém a informação que está sendo trocada entre os agentes;

– remetente (*sender*) – referencia o *Papel* do agente que está enviando a *Mensagem* e a instância desse *Papel*;

– destinatários (*receivers*) – referencia um ou mais *Papéis* de agente receptores da *Mensagem* e as instâncias desses *Papéis*.

A Figura 2-11 mostra um exemplo de representação de uma *Mensagem* em XMLaw.

```
<Messages>
    <Message id="id-da-mensagem" performative="inform">
        <Content>
            <Entry key="chave" value="=valor"/>
        </Content>
        <Sender role-ref="id-do-papel-remetente" role-instance="?" />
        <Receivers>
            <Receiver role-ref="id-do-papel-destinatario" role-instance="?" />
                ...
        </Receivers>
    </Message>
        ...
</Messages>
```

Figura 2-11 - Exemplo de XMLaw da *Mensagem*