2
Fundamentação Teórica

Neste capítulo são apresentados os fundamentos teóricos contemplados no desenvolvimento deste trabalho. Inicialmente é apresentada uma descrição de sistemas multiagentes. Em seguida, são descritos alguns conceitos sobre normas e seu uso para regular o comportamento dos agentes. Além disso, é apresentada a plataforma Jason para desenvolvimento de sistemas multiagentes. Por fim, será abordado o tema de simulações e o uso de sistemas multiagentes na sua implementação.

2.1.
Sistemas Multiagentes

As tecnologias tradicionais de desenvolvimento de software, tais como as de orientação a objetos, falham ao fornecerem técnicas de decomposição, abstração e organização adequadas para modelagem e desenvolvimento de sistemas complexos, tais como os apresentados em (LUCENA, 1987) e (CHOREN e LUCENA, 2005). Com o objetivo de melhor lidar com a complexidade inerente a tais sistemas, a tecnologia de agentes de software tem se mostrado mais adequada (LUCENA, 1987) (CHOREN e LUCENA, 2005) (WEISS, 1999) (SILVA e LUCENA, 2007).

Agentes são entidades situadas em um ambiente, capazes de realizar comportamentos autônomos neste ambiente para alcançar seus objetivos (JENNINGS e WOOLDRIDGE, 1996).

Wooldridge e Jennings (WOOLDRIDGE e JENNINGS, 1995) caracterizam agentes, como segue:

Autonomia: agentes devem ser capazes de solucionar os problemas delegados a eles sem a intervenção direta de humanos ou outros agentes, ou seja, agentes devem ter certo grau de controle sobre suas ações e seu estado interno.

Habilidade Social: agentes devem ser capazes de interagir, quando julgarem apropriado, com outros agentes de software e humanos, a fim de resolver seus problemas.

Reativo: agentes devem perceber seu ambiente e responder em tempo hábil as alterações que ocorrem no mesmo.

Pro atividade: agentes não devem simplesmente agir em resposta ao seu ambiente, eles devem ser capazes de visualizar oportunidades, manter um comportamento direcionado as suas metas e tomar a iniciativa quando apropriado.

Um agente de software, em geral não é encontrado sozinho em uma aplicação ou sistema, mas em conjunto com outros agentes, de tipos iguais ou diferentes, formando uma sociedade ou organização. A esta sociedade dá-se o nome de *Sistemas Multiagentes*. Segundo Wooldridge (WOOLDRIDGE, 2002), um sistema multiagente consiste em uma sociedade de agentes capazes de interagir entre si. Para interagirem com sucesso, serão necessárias as habilidades de cooperação, coordenação e negociação entre eles, assim como os humanos o fazem (WOOLDRIDGE, 2002).

2.2. Normas

Normas são mecanismos para regular o comportamento de agentes, e elas representam a maneira com que agentes entendem as responsabilidades dos outros agentes (LÓPEZ, 2003). Assim os agentes trabalham com a crença de que os outros irão se comportar de acordo com a prescrição das normas, mas, sobretudo, as normas são mecanismos para capacitar os agentes do direito de exigir que os outros agentes se comportem de uma determinada maneira. Desta forma, o uso de normas é uma necessidade em sistemas multiagentes em que os membros são autônomos, mas não autossuficiente, e que, portanto, a cooperação é assim necessária, mas não pode ser assegurada.

Sabendo que normas são mecanismos que a sociedade tem a fim de influenciar o comportamento dos agentes, normas servem para alcançar diferentes finalidades, bem como a construção de um simples acordo entre agentes, ou até mesmo o mais complexo sistema jurídico (LÓPEZ, LUCK e D'INVERNO, 2002). Estas podem persistir durante diferentes períodos de tempo, por exemplo,

enquanto o agente permanece na sociedade, ou apenas por um curto período de tempo até que o objetivo social tenha sido realizado (LÓPEZ, LUCK e D'INVERNO, 2002).

Com isso, diferentes aspectos podem ser usados para caracterizar normas. Primeiro normas são sempre criadas para serem cumpridas por um determinado conjunto de agentes a fim de alcançar objetivos sociais. Segundo, normas não são sempre aplicadas, e sua ativação depende de condições do contexto em que os agentes estão situados. Além disso, é preciso saber qual o tipo de elemento que a norma regula, se é uma ação do agente, ou um estado do ambiente. Finalmente, em alguns casos, normas podem estabelecer um conjunto de sanções para serem impostas quando os agentes cumprirem ou violarem com as normas (LÓPEZ, LUCK e D'INVERNO, 2002), (LÓPEZ, 2003), (SANTOS NETO, SILVA e LUCENA, 2011).

López *et al* (LÓPEZ, LUCK e D'INVERNO, 2002) descreve os agentes capazes de lidar com normas como sendo agentes autônomos cujo comportamento é regulado pelas obrigações ou proibições.

Para assegurar que os interesses pessoais dos agentes não sobreponham os objetivos sociais (LÓPEZ, LUCK e D'INVERNO, 2002) é interessante utilizar mecanismos capazes de associar punições e recompensas as normas. Punições e recompensas afetam a decisão dos agentes para cumprir com as normas se elas impedirem ou beneficiarem algum dos objetivos individuais dos agentes. Isto é, punições não podem ser levadas em conta se nenhuma delas influência na conquista de objetivos individuais pelos agentes. De forma similar para recompensas, elas procuram motivar os agentes a cumprirem com as normas, pois caso seja cumprida a norma, o agente receberá os benefícios associados a esta norma. Assim, pode-se dizer que recompensas e punições não tem nenhum efeito na decisão dos agentes se estas não estiverem associadas com algum dos objetivos individuais dos agentes.

2.3. Jason

A plataforma Jason possibilita o desenvolvimento e execução de agentes BDI. O interpretador disponibilizado pela plataforma Jason para a execução de

agentes BDI é apresentado em detalhe na Figura 2.2 (reproduzida de (MACHADO e BORDINI, 2002)). Na Figura, conjuntos de crenças, eventos, planos e intenções são representados por retângulos. Losangos representam a seleção de um elemento de um conjunto. Círculos representam alguns dos processos envolvidos pelo interpretador.

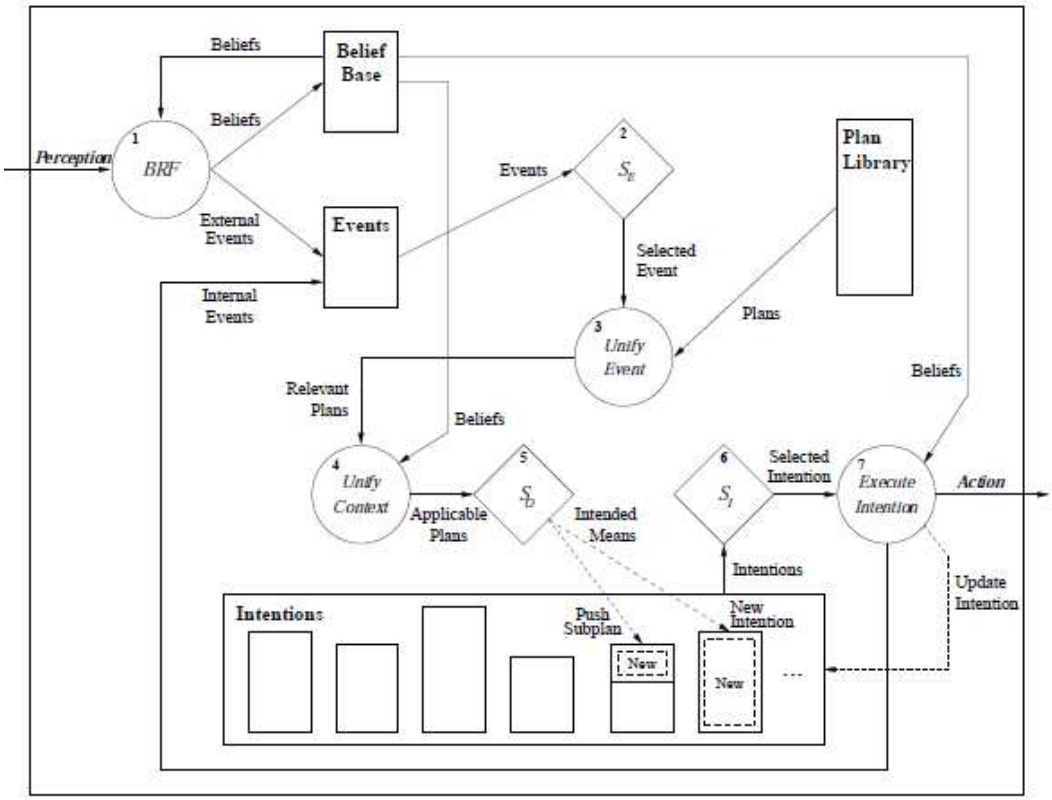

Figura 2.1 - Ciclo de interpretação do Jason (MACHADO e BORDINI, 2002)

A cada ciclo de interpretação de um programa, a lista de eventos é atualizada com o resultado do processo de revisão de crenças (realizado pela função *Belief Review Function* - *BRF*). Assume-se que as crenças são atualizadas pela percepção e que sempre que houver mudanças na base de crenças do agente, isso implica na inserção de um evento no conjunto de eventos.

Depois da função *Selecting Event* (S_E) selecionar um evento, o interpretador tem que unificar este evento com as condições de invocação dos planos armazenados na biblioteca de planos. Isso gera um conjunto de todos os *planos relevantes* para o evento escolhido. A partir da verificação dos contextos dos planos relevantes, o interpretador Jason determina o conjunto de *planos aplicáveis* (planos que podem ser usados, na situação presente, para tratar o evento

selecionado naquele ciclo). Depois, a função *Selecting Option* (S*O*) escolhe entre os planos aplicáveis aquele que será utilizado para lidar com o evento selecionado, e adiciona o plano para o conjunto de intenções do agente.

Por fim, uma intenção é selecionada, pela função *Selecting Intention* (S*I*), a partir do conjunto de intenções do agente para ser executada e, posteriormente, tal intenção é executada pela função *Execute Intention.*

Depois de apresentar o Jason em detalhes é interessante mostrar alguns recursos que ele disponibiliza. Ele possibilita a criação de ambientes onde os agentes podem perceber e atuar. Isto é feito a partir da extensão de uma classe *Enviroment* provida pelo Jason.

O objetivo desta seção foi apresentar informalmente alguns aspectos importantes do Jason, maiores detalhes do interpretador podem ser encontrados nos seguintes artigos (MOREIRA e BORDINI, 2002), (BORDINI e MOREIRA, 2004) e (MOREIRA, VIEIRA e BORDINI, 2003).

2.4. Simulação

Caccalano (CACCALANO, 2010) descreve Simulação como a criação e utilização de modelos de um sistema ou processo real, com o propósito de realizar experimentos e análises que conduzam a uma melhor compreensão do funcionamento do sistema ou processo. Vale ressaltar que o modelo não precisa necessariamente ser criado em um computador. Porém, a tecnologia e os softwares atuais de simulação estão em um estágio de desenvolvimento bastante avançado, fazendo com que as aplicações possam abranger desde problemas simples até sistemas bastante complexos. Modelos de simulação são empregados, por exemplo, no Suporte a Prevenção de Crimes (BOSSE e GERRITSEN, 2010), Evacuação de civis em Áreas de Risco (CERQUEIRA, SANTOS NETO, *et al.*, 2009), na programação de produção de uma empresa, na análise de problemas de uma célula de manufatura ou de um sistema de movimentação de materiais (CACCALANO, 2010).

A simulação permite a análise e o teste de diferentes alternativas de funcionamento de um sistema, com variações de parâmetros diversos, formas de

controle, priorização de eventos, diferentes equipamentos, velocidades, dentre outras características (CACCALANO, 2010).

O crescente uso de software para realizar simulações cada vez mais complexas, expõe desafios interessantes em relação a construção de sistemas mais próximos da realidade. Para tanto, diversos trabalhos (BOSSE e GERRITSEN, 2010), (LÓPEZ, LUCK e D'INVERNO, 2002), (TISUE e WILENSKY, 2004) e (BORDINI, HÜBNER e WOOLDRIDGE, 2007) utilizaram de sistemas multiagentes para reproduzir de forma mais realística suas simulações. Isto porque como visto na Seção 2.1, agentes de software possuem características que permite sem a intervenção direta de humanos ou de outros agentes realizarem tarefas, mas quando acha interessante, conseguem interagir com outros agentes e ajudá-los a resolver seus problemas, visualizam oportunidades e ainda possuem habilidade social. Apesar de o agente possuir estas características, isto não garante que ele irá decidir por um bem estar social e assim é interessante a inserção de normas para regular o comportamento dos agentes nas simulações. Neste caso, o foco deste trabalho, são os problemas de construção de simulações para sistemas multiagentes normativos e uma melhor compreensão dos impactos das normas sobre os agentes.