2
Fundamentos

Neste capítulo, apresenta-se alguns dos conceitos necessários para um melhor entendimento desta dissertação. Como é proposta desta dissertação uma abordagem para regular a interação de sistemas multi-agentes abertos utilizando as noções de leis, conceitos sobre sistemas abertos e sistemas multi-agentes são apresentados brevemente.

2.1
Sistemas Abertos

A noção de sistemas abertos foi primeiramente introduzida por Hewitz [19]. Neste trabalho seminal, já se previa que devido à competição, diferença de objetivos e responsabilidades, e distribuição geográfica, uma boa parte das aplicações seria baseada na comunicação entre subsistemas que seriam desenvolvidos separadamente e independentemente.

Mais precisamente, Hewitz identificou as seguintes características dos sistemas abertos.

(i) Não existem objetos globais. A única semelhança entre os subsistemas é a capacidade de comunicação de uns com os outros. Isto significa que em sistemas abertos não é possível presumir nada sobre aspectos internos dos subsistemas, como arquitetura, por exemplo.

(ii) É muito difícil determinar quais são os objetos que existem no sistema. Essa segunda característica reflete a autonomia que cada subsistema possui para realizar as suas ações. Subsistemas podem entrar e sair do sistema a qualquer momento, seja por uma falha de funcionamento ou por decisão dos seus desenvolvedores.

(iii) Subsistemas têm conhecimento apenas parcial do sistema como um todo. Essa característica reflete a incerteza inerente aos sistemas abertos. Cada subsistema tem somente conhecimento parcial sobre o sistema como um todo e novos conhecimentos podem ser adquiridos ou modificados em função da evolução da interação entre os subsistemas.

Em conseqüência dessas três noções básicas, pode-se dizer que sistemas abertos são (i) concorrentes, uma vez que são compostos por diferentes subsistemas operando geograficamente distribuídos e utilizam-se de diferentes fontes de informações; (ii) assíncronos; (iii) possuem controle descentralizado, no sentido de que o fluxo de execução do sistema é regido por decisões locais dos subsistemas; (iv) e podem conter informação inconsistente [20].

O desenvolvimento de sistemas abertos é frequentemente uma tarefa árdua, pois envolve vários atributos caracterizados por trazer complexidade ao desenvolvimento de software, tais como grande número de interação entre os subsistemas e concorrência [21].

Entretanto, com o advento de novas tecnologias como web-services [22] e o amadurecimento de paradigmas de software tal como a engenharia de software orientada a agentes (ESOA) [23, 24], o desenvolvimento de sistemas abertos tende a ser facilitado, pois estas tecnologias já foram concebidas considerando muitas das características de um sistema aberto.

2.2 Sistemas Multi-Agentes

Nesta seção, apresenta-se a abordagem multi-agentes como adequada para o tratamento de sistemas complexos. Considera-se que sistemas abertos podem se tornar complexos na medida em que a quantidade de subsistemas presentes aumenta. Além disso, como a abordagem de leis é baseada na interação entre os agentes, descreve-se em mais detalhes algumas das características da interação em sistemas multi-agentes.

A abordagem multi-agentes é apontada como promissora para a análise, projeto e implementação de sistemas de softwares complexos [23]. A complexidade dos sistemas aumenta na medida em que seus componentes representam hierarquias de sistemas, ou sub-sistemas, caracterizando o que a literatura freqüentemente denomina de Sistema Complexo [25]. Sistemas de larga escala, devido ao grande número de sub-sistemas envolvidos, são exemplos deste tipo de sistema. Além disso, sistemas abertos também podem se tornar complexos na medida em que a quantidade de subsistemas presentes aumenta.

Um software complexo é tipicamente caracterizado por um grande número de partes que interagem. Logo, o papel do engenheiro de software é prover estruturas e técnicas que tornem mais fácil o gerenciamento dessa complexidade. Segundo Jennings, apesar da grande variedade de tipos de complexidades existentes, elas possuem algumas regularidades [25]:

– A complexidade geralmente tem a forma de uma hierarquia. Isto é, um sistema é composto de subsistemas inter-relacionados, que por sua vez são compostos de outros subsistemas.
– Sistemas hierárquicos evoluem muito mais rapidamente do que sistemas não hierárquicos de tamanhos equivalentes.
– É possível distinguir as interações entre os sub-sistemas e dentro dos sub-sistemas, onde geralmente, as interações dentro dos sub-sistemas ocorrem com maior frequência.

Diante dessas observações os engenheiros de software fazem uso de várias técnicas para lidar com esta complexidade, dentre elas, as principais são [25]:

– Decomposição - a técnica mais básica para lidar com um grande problema é dividi-lo em partes menores, de forma que estas partes se tornem mais gerenciáveis, possibilitando seu tratamento de forma relativamente isolada.
– Abstração - esse processo define um modelo simplificado do sistema, enfatizando apenas algumas propriedades e ignorando outras. Desta maneira a atenção pode ser focada nos aspectos relevantes do problema.
– Organização - esse processo identifica e gerencia o relacionamento entre os vários elementos resolvedores de problemas. Este processo ajuda a lidar com a complexidade, principalmente, permitindo que alguns desses elementos possam ser agrupados e tratados como uma simples unidade em um nível mais alto de abstração.

Atualmente existem muitas definições para o termo agente [26, 27], muitas delas conflitantes. Wooldridge definiu um agente inteligente como uma entidade que possui as seguintes propriedades [28]:

– Autonomia: os agentes operam sem a intervenção direta de qualquer entidade, seja ela humana ou não. Além disso, eles possuem controle sobre suas ações e sobre o seu estado.
– Habilidade social: agentes interagem com outros agentes (possivelmente humanos).
– Reatividade: os agentes percebem o seu ambiente e respondem às mudanças que ocorrem.
– Pró-atividade: os agentes não agem simplesmente em resposta ao ambiente, mas eles são capazes de "tomar a iniciativa".
– Noções mentais: os agentes possuem conceitos que são geralmente aplicados a humanos, como por exemplo: conhecimento, crença, intenção e obrigação.

Porém, muitos problemas envolvem um grande número de agentes, seja para representar a natureza descentralizada do problema, para representar vários pontos de vista sobre um mesmo problema, ou até mesmo para representar interesses de competição. Portanto os agentes precisarão interagir para alcançar seus objetivos ou para gerenciar as dependências entre as tarefas que estarão sendo resolvidas.

Em geral os agentes interagem para atingir seus objetivos individuais ou para atingir os objetivos coletivos. Segundo Ferber o relacionamento entre agentes somente é possível dentro de uma organização [27]. Portanto ao interagir, os agentes estão sob algum contexto organizacional. Logo é possível definir um relacionamento organizacional baseado nestes contextos. Por exemplo, os agentes podem formar pares para trabalhar juntos em um time, ou um pode ser o chefe do outro e etc. Diante disso, as técnicas da ESOA são consideradas adequadas para lidar com esses tipos de problemas pois [25]:

- através dos agentes a decomposição do espaço de problemas pode ser feita de forma efetiva, e muitas vezes intuitiva.
- a abstração de agentes é adequada para modelar sistemas complexos (autonomia, pró-atividade, etc).
- a abordagem orientada a agentes para identificar e gerenciar relacionamentos organizacionais é apropriada para lidar com as dependências e interações em um sistema complexo.

2.2.1 Comunicação

A comunicação é resultante da habilidade social dos agentes. Frequentemente, a comunicação entre os agentes é baseada na teoria de atos da fala [29]. Desta forma, a comunicação é tratada como um tipo de ação que modifica o mundo da mesma forma que uma ação física modificaria. Mas ao contrário das ações físicas, uma ação de comunicação modifica as crenças das partes envolvidas na interação [30]. Em geral as ações de comunicações são representadas por um conjunto finito de ações que representam as intenções dos agentes. Exemplos dessas ações são: *agree*, representa a ação de concordância para realizar alguma ação, possivelmente no futuro; e *propose*, representa a ação de submissão de uma proposta para a realização de uma certa ação, dadas determinadas condições.

Para que a comunicação seja possível, é preciso que os agentes utilizem uma mesma linguagem. Linguagens com o propósito de tornar possível a comunicação entre os agentes são chamadas de ACL, abreviação para Agent Communication Language. Essas linguagens são compostas basicamente de duas partes:

envelope e conteúdo. O envelope contém meta-informações sobre o conteúdo que está sendo trocado entre os agentes. O conteúdo contém as informações trocadas entre os agentes. Essas informações incluem a ação de comunicação e o conhecimento que está sendo transmitido. Esse conhecimento, por sua vez pode ser expresso em várias linguagens usadas para representação de conhecimento, como Prolog [31] e KIF [32].

Porém, a interação entre os agentes nem sempre acontece através da simples troca de mensagens. Freqüentemente, os agentes interagem através de processos complexos, envolvendo várias trocas de mensagens sob um mesmo contexto. Em outras palavras, a interação entre os agentes para executar uma determinada tarefa pode envolver, por exemplo, um processo de negociação, competição ou cooperação.

Quando agentes estão envolvidos em uma interação sob um determinado contexto, diz-se que eles estão envolvidos em uma conversação. Conversações podem ser vistas sob dois pontos de vista: global e local.

Observar uma conversação de um ponto de vista global significa ter uma visão completa de todas as interações que estão ocorrendo no sistema. A visão local, por outro lado, é relacionada às interações a que cada agente tem acesso. A visão local é a visão particular de um agente em relação às interações do sistema. Dessa forma, a visão global pode ser construída através da união das visões locais.

Sendo os agentes entidades autônomas, as conversações podem evoluir e se tornar completamente imprevisíveis. Embora para alguns domínios, como por exemplo simulações biológicas [33, 34], essa imprevisibilidade é desejável, para outros domínios tamanha imprevisibilidade é sinal de dificuldade de manutenção e possibilidade de falhas no sistema. Neste contexto, as conversações podem ser estruturadas em protocolos de interação. Um protocolo define o que, como e quando se comunicar, possibilitando a compreensão do que o remetente da mensagem pode esperar como resposta, e como o destinatário deve reagir ao receber uma mensagem [35].

Muitas formas de representação para protocolos de interação foram propostas. Cada uma apresenta um conjunto de vantagens e desvantagens. Neste trabalho, optou-se por escolher uma abordagem que possuísse uma boa relação entre facilidade de especificação e expressividade.

Autômatos finitos [36] oferecem uma maneira intuitiva para especificar as conversações. Normalmente, passos na execução do protocolo são representados por estados e arcos direcionados representam a transição entre os estados. O principal problema com esta abordagem é a falta de expressividade para representar problemas relativos à concorrência. Entretanto, muitos pesquisadores

utilizam essa abordagem com pequenas modificações para este propósito, como pode ser visto em [9, 37, 38].

Redes de Petri foram introduzidas por Carl Adam Petri [39] no início da década de 60. Trata-se de uma linguagem formal e gráfica apropriada para modelar sistemas com concorrência e compartilhamento de recursos. A linguagem é uma generalização da teoria de autômatos, onde o conceito de eventos ocorrendo concorrentemente pode ser expressado. Apesar da expressividade das redes de Petri, a maior crítica ao seu uso é a explosão combinatorial do tamanho da rede quando a complexidade do protocolo aumenta [9]. Além disso, a especificação de protocolos utilizando essa abordagem não é tão fácil quando comparada com outras abordagens, como, por exemplo AUML [40]. Exemplos de trabalhos que utilizam redes de petri podem ser encontrados em [41], onde as redes são usadas para especificar e verificar as interações dos agentes e em [42] onde é proposta a especificação do protocolo através de AUML e então são traduzidos para redes de petri coloridas.

AUML estende os diagramas de seqüência UML para representar troca de mensagens assíncronas entre papéis de agentes. Uma das principais vantagens do AUML é o reuso das construções UML. Isto permite que os desenvolvedores de software já acostumados com UML tenham uma rápida curva de aprendizado com o AUML. AUML não dispunha de uma semântica formal que a descrevesse, no entanto em [43], propõe-se uma semântica formal para AUML baseado em redes de petri.