1
Introdução

A abordagem multi-agentes vem sendo apontada como apropriada para o desenvolvimento de sistemas de software complexos [36]. Sistemas compostos por um conjunto de agentes autônomos que interagem uns com os outros para atingirem objetivos, oferecem uma forma promissora de desenvolvimento de softwares complexos [10][36]. Além de serem compostos por vários agentes, os sistemas atuais vêm sendo desenvolvidos por equipes dispersas geograficamente, e são freqüentemente executados em ambientes distribuídos e abertos [4]. Muitos sistemas abertos oferecem serviços que podem ser utilizados por um outro conjunto de sistemas que possuem comportamento desconhecido. Isso faz com que a heterogeneidade e a imprevisibilidade sejam as principais características de sistemas abertos. Muitas aplicações de diversos domínios tais como comércio eletrônico, cadeia de suprimentos e aplicações médicas têm sido desenvolvidas como experimentos com a utilização da tecnologia multi-agentes para serem executadas de forma distribuída e em ambientes abertos. Exemplos desses experimentos estão em [41][42][43]. Com o objetivo de lidar com a heterogeneidade e a imprevisibilidade destas aplicações e tentando evitar comportamentos indesejáveis e falhas de execução, mecanismos de gerenciamento do comportamento emergente dos agentes de software vêm sendo propostos na literatura.

1.1
Motivação

Sistemas multi-agentes abertos podem ser considerados altamente complexos devido à quantidade enorme e variável da população de agentes que podem participar desses sistemas. Por não se ter um controle centralizado sobre o desenvolvimento dos agentes participantes de um sistema multi-agente aberto, estes são desenvolvidos de forma heterogênea, sendo escritos (codificados) por pessoas diferentes, em diferentes linguagens de programação. Esses agentes possuem normalmente interesses individuais e apresentam comportamentos diferentes entre si e não esperados. Portanto, não se pode esperar que esses agentes se comportem de forma previsível.

Sistemas multi-agentes abertos e sem controle podem acabar se tornando caóticos. Portanto, sistemas dessa natureza requerem um mecanismo de aplicação de leis que deva estipular o que os agentes podem, não podem e devem fazer [14]. Tratar a imprevisibilidade é fundamental para o desenvolvimento de softwares confiáveis, o que significa basicamente que os softwares devem funcionar de acordo com as suas especificações. Tratar esse problema através do monitoramento de todas as ações executadas pelos agentes, além de dispendioso em termos de performance, não seria completo, pois agentes podem executar ações, tais como a execução de algoritmos e acesso a recursos do sistema (arquivos ou banco de dados), que não são observáveis sob o ponto de vista do sistema e de outros agentes. Além disso, em sistemas multi-agentes abertos, é necessário considerar que aspectos internos dos agentes, tais como objetivos, planos, intenções, entre outros não são observáveis nem controláveis [15]. Apenas alguns aspectos, como a troca de mensagens, por exemplo, são observáveis [5]. Dessa forma, não se pode impedir que ações proibidas estejam na lista de intenções dos agentes, assim como não se pode fazer com que obrigações a serem cumpridas façam parte de seus objetivos.

Sendo assim, é necessário um mecanismo que possa tratar não somente o comportamento observável dos agentes (as ações que são observáveis), como também o comportamento não observável (as ações que não são observáveis) do ponto de vista deste mecanismo e de outros agentes do sistema. Na seção 1.5 será apresentado um mecanismo de governança cujo objetivo é tratar o comportamento não observável dos agentes através da verificação de fatos e eventos que são conseqüência desse comportamento.

1.2 Sistemas Multi-Agentes

Construir aplicações complexas e de qualidade é considerado muito difícil [36]. Por esse motivo, diversos paradigmas de software foram propostos. Programação estruturada, programação orientada a objetos e *frameworks* são alguns exemplos. Os sistemas desenvolvidos atualmente tendem a ser abertos, distribuídos e compostos por diversos componentes que interagem uns com os outros e que sofrem alterações dinamicamente com a entrada de novos componentes e a saída de componentes já existentes [37]. Esses componentes podem possuir objetivos diferentes e, portanto conflitantes. Esses fatos aumentam ainda mais a complexidade dos sistemas atuais.

Algumas das técnicas utilizadas para tratar a complexidade de sistemas distribuídos são:

- **Decomposição:** Sistemas complexos devem ser compostos por um conjunto de sub-sistemas organizados de forma hierárquica. Os sub-sistemas devem trabalhar em conjunto para atingir os objetivos do sistema.
- **Abstração:** Definição de modelos simplificados do sistema que enfatize alguns detalhes e propriedades, limitando a área de interesse (escopo) do desenvolvedor em um determinado momento.
- **Organização:** Sistemas complexos exigem relações organizacionais. As organizações permitem agrupar componentes de acordo com algum critério (i.e. família de problemas a serem resolvidos) e tratá-los como se fossem uma única entidade, além de estabelecer relações de alto nível entre essas entidades (i.e. relações entre os problemas a serem resolvidos).

Tendo apontado alguns problemas e características de sistemas complexos, é preciso dirigir o desenvolvimento de software para um paradigma que resolva ou minimize essas questões. Muitos pesquisadores argumentam que a abordagem multi-agentes é adequada ao desenvolvimento de sistemas complexos [36][37]. Essa abordagem permite que sistemas sejam implementados como uma coleção de agentes autônomos que interagem entre si com o objetivo de resolverem problemas. Seu principal conceito é o Agente: Uma unidade de software autônoma situada em um ambiente e capaz de executar ações para atingir seus objetivos [36]. Os agentes são considerados entidades solucionadoras de problemas e para os autores de [38] um agente deve ser:

- **Autônomo:** Agentes devem agir sem a necessidade de intervenção humana direta e devem ter controle sobre suas ações e seus estados internos.
- **Reativo:** Agentes devem perceber o ambiente onde estão situados e responder a mudanças ocorridas neste ambiente.
- **Pró-ativo:** Agentes devem possuir um comportamento oportunista, orientado a objetivos, executando suas ações sempre que oportuno.
- **Social:** Agentes devem interagir com outros agentes, quando apropriado, com o intuito de atingir seus objetivos ou de ajudar outros agentes.

- **Intencional:** Agentes devem ter a intenção de realizar uma tarefa. Para agentes agirem de forma intencional, eles devem possuir crenças (conhecimento sobre o ambiente) e objetivos a serem atingidos.

Essas características são apropriadas para que as técnicas que tratam complexidade, citadas anteriormente (decomposição, abstração e organização), sejam utilizadas. Sendo assim, agentes autônomos podem ser considerados componentes de software que possuem as características apresentadas acima. Conseqüentemente, agentes podem ser utilizados como uma técnica de decomposição e modularização, com controle descentralizado, além de representarem abstrações para problemas existentes. Reatividade e pró-atividade tratam aspectos dinâmicos e imprevisíveis de um sistema complexo e aberto. Por fim, habilidades sociais estão fortemente relacionadas com questões organizacionais, já que é através delas que será possível tratar as dependências e interações entre os componentes (organizações) de um sistema complexo.

1.3 Sistemas Abertos

O conceito de sistemas abertos é definido em [33] como sendo sistemas computacionais, compostos por vários sub-sistemas distribuídos, que são desenvolvidos separadamente e independentemente e que possuem a capacidade de se comunicarem. A distribuição geográfica, divisão de responsabilidades e características econômicas, podem ser considerados razões pelas quais os sub-sistemas são desenvolvidos separadamente e independentemente. Essa natureza aberta permite que a evolução desse tipo de sistema seja contínua e incremental.

Em um sistema aberto não existem objetos ou variáveis globais e a comunicação entre os sub-sistemas é a sua característica básica. Por serem projetados independentemente, a única característica comum que, necessariamente, os sub-sistemas terão é a habilidade de se comunicarem uns com os outros. Para isso, eles deverão obedecer uma linguagem comum, caso contrário, não haverá comunicação [33].

A natureza aberta de um sistema faz com que seja muito difícil determinar quais objetos (sub-sistemas) compõem um sistema em um determinado momento no tempo [33]. Cada sub-sistema deve ter conhecimento próprio de suas habilidades e funcionalidades e conseqüentemente, cada sub-sistema possui um conhecimento parcial do sistema como um todo. Para aumentar a base de conhecimento dos sub-sistemas, é preciso fazer com que o conceito de

disseminação de informação esteja na base da comunicação entre esses sub-sistemas. Uma vez que parte do conhecimento seja disseminada, todos os sub-sistemas deverão ser afetados, adicionando-o e relacionado-o ao conhecimento já existente.

É possível concluir que sistemas abertos lidam com uma grande quantidade de informações e por isso devem possuir as seguintes características [34]:

- **Concorrência:** Sistemas abertos são compostos por vários componentes capazes de trocar informações. Esses componentes devem ser capazes de controlar o fluxo de entrada (de diferentes fontes) e saída de informações de forma simultânea.
- **Assincronia:** Como o comportamento do ambiente onde os componentes do sistema aberto não é previsível, novas informações podem surgir, a qualquer momento, fazendo como que o sistema tenha que operar de forma assíncrona com o mundo externo.
- **Controle descentralizado:** Um processo de tomada de decisão centralizado em sistemas abertos pode se tornar um gargalo na aplicação. O controle do sistema deve se dar de forma distribuída e decisões locais devem acontecer perto de onde elas são necessárias.
- **Tratar inconsistência de informações:** Informações do mundo externo e até mesmo de fontes internas ao sistema podem ser inconsistentes. Ainda assim, decisões devem ser tomadas pelos componentes do sistema aberto de acordo com as informações disponíveis no momento.
- **Rede de relacionamentos:** Os componentes de um sistema aberto devem pertencer a uma rede de relacionamentos onde o estado interno de cada um desses componentes pode ser desconhecido para outros componentes. Dessa forma, as informações de um sistema aberto devem ser transferidas entre os componentes através de comunicação explicita, tal como a troca de mensagens, por exemplo.
- **Execução contínua:** Os sistemas abertos devem ser confiáveis (tolerantes a falhas). Esses sistemas devem ser projetados de forma que a falha de um componente dever ser compensada por outros componentes em funcionamento.

1.4 Descrição do Problema

Sistemas multi-agentes abertos são sociedades constituídas por agentes autônomos, heterogêneos, independentemente projetados e de comportamento imprevisível. Esses agentes podem trabalhar para atingirem objetivos comuns ou diferentes [3]. Neste cenário, a habilidade de coordenar e controlar as operações que esses agentes executam se torna importante, pois essas características citadas, se não tratadas, podem causar falhas no sistema projetado. Com o objetivo de lidar com a autonomia, a imprevisibilidade, a heterogeneidade e a diversidade de interesses entre os diferentes membros da sociedade de agentes, sistemas de governança têm sido propostos. Esses sistemas de governança têm o objetivo de controlar o comportamento dos agentes, estabelecendo um conjunto de normas que descrevem ações (destes agentes) cujas execuções são permitidas, proibidas ou obrigatórias [1][7]. Alguns desses sistemas assumem que as normas podem, eventualmente, ser violadas pelos agentes, com o intuito de manter a autonomia dos mesmos [15] e que o estado interno dos agentes não é observável, nem controlável [15][5].

Alguns dos sistemas de governança propostos na literatura têm o objetivo de regular a interação entre os agentes, através de um mecanismo que media essa interação, verificando se as mensagens trocadas violam ou não as normas estabelecidas [4][5][13][14]. Cada mensagem enviada por qualquer agente é analisada por este mecanismo. Caso esta mensagem viole alguma norma da aplicação, ela não é encaminhada para o seu destinatário. As principais desvantagens desse tipo de abordagem são: (i) ela viola a privacidade dos agentes. Analisar as mensagens trocadas entre os agentes representa não só verificar sua estrutura e conformidade com o protocolo em que ela está inserida, mas também verificar seu conteúdo e sua conformidade em relação ao contexto da aplicação no momento em que ela foi enviada. A verificação do conteúdo da mensagem viola a sua confidencialidade. E (ii) ela regula apenas a interação entre os agentes. Ou seja, as ações executadas pelos agentes não relacionadas à troca de mensagens, acesso a recursos por exemplo, não são reguladas [15].

Outras abordagens provêem suporte para a aplicação de normas que regulam não apenas interações, mas também acesso a recursos [2]. TuCSoN [2] provê um mecanismo de coordenação que gerencia a interação entre os agentes e um mecanismo de controle de acesso a recursos. Em TuCSoN os agentes interagem através de um meio de coordenação independente chamado *Tuple*

Centre. O mecanismo de controle de acesso executa a sua funcionalidade tornando os *Tuple Centres* visíveis ou invisíveis. Embora em TuCSoN seja possível descrever normas para controlar o acesso a recursos, este controle está restrito apenas a recursos inseridos no ambiente onde existam os *Tuple Centres*.

Em [15] os autores propõem um sistema de governança para regular o comportamento observável dos agentes através da verificação de mensagens públicas e ações visíveis. Neste trabalho é proposta uma classificação das normas, além de guias de implementação para a verificação do cumprimento e aplicação das normas, de acordo com essa classificação. A principal desvantagem da abordagem proposta é que ela não oferece suporte a governança de mensagens e ações que não são diretamente observáveis pelo mecanismo de governança. Em sistema multi-agentes abertos existirão mensagens que não serão públicas e que serão percebidas apenas por remetentes e destinatários, além de execução de ações que serão notadas apenas pelos agentes que executam as ações e por agentes que sofrem as conseqüências dessas ações [44].

Percebe-se que a maioria das abordagens propostas na literatura trata a governança de sistemas multi-agentes através da verificação de mensagens trocadas entre os agentes ou através da verificação de ações observáveis executadas pelos agentes. Observa-se então que existe um conjunto de ações que ainda não é tratado por tais abordagens. Tais ações podem não estar relacionadas à troca de mensagens ou não serem observáveis sob o ponto de vista do mecanismo de governança ou dos agentes, entre elas estão a execução de algoritmos, o acesso a recursos como arquivos ou banco de dados, a troca de papéis desempenhados e outras. Portanto, ainda existe a necessidade de mecanismos de governança que, além de não violar a privacidade dos agentes, regule tal conjunto de ações.

1.5 Solução Proposta

O objetivo deste trabalho é apresentar um mecanismo de governança para sistemas multi-agentes baseado em testemunhos e uma infra-estrutura de software (*framework*) que dê suporte a esse mecanismo.

Testemunhos podem ser enviados pelos agentes da aplicação, relatando fatos ou eventos que violam as normas estabelecidas para a aplicação. Agentes são capazes de observar fatos ou eventos no ambiente em que estão inseridos. Já que eles devem conhecer as normas estabelecidas pela sociedade onde

estão, eles são capazes de saber quais fatos ou eventos observados estão relacionados à violação de normas. A partir dessa observação, agentes podem enviar testemunhos relatando a ocorrência desses fatos. Entretanto, em sistemas multi-agentes abertos, agentes podem possuir interesses próprios, serem mal intencionados ou serem mal projetados e, portanto tais testemunhos podem não ser verdadeiros. Dessa forma, o mecanismo de governança, além de receber os testemunhos enviados pelos agentes, deve verificar a veracidade de tais testemunhos. Esse processo de verificação pode fazer uso da reputação dos agentes envolvidos. Uma vez verificada a veracidade de um testemunho, o mecanismo deve aplicar uma punição ao agente infrator, caso tenha sido constatada a violação de uma norma, ou aplicar uma punição ao agente que enviou o testemunho, caso tenha sido verificando o envio de um falso testemunho.

Portanto o mecanismo de governança proposto assume que normas podem ser violadas pelos agentes e trata a violação das normas através de punição aos agentes infratores. Tal mecanismo é composto por abstrações que são responsáveis por receber e julgar os testemunhos enviados pelos agentes da aplicação, manter a reputação desses agentes e aplicar punições aos agentes infratores.

Com a abordagem proposta é possível regular não apenas violações relacionadas à troca de mensagens entre os agentes, mas também violações relacionadas à execução de outros tipos de ações que podem ou não ser observadas pelo sistema e por outros agentes. Além disso, a privacidade dos agentes é mantida, já que as mensagens não precisam ser verificadas por um mecanismo de mediação.

1.6 Principais Contribuições

Nesta dissertação é apresentado um mecanismo de governança para sistemas multi-agentes abertos cujo objetivo é controlar o comportamento dos agentes participantes do sistema. As principais contribuições desta dissertação são:

1. Um mecanismo de governança baseado em testemunhos, capaz de regular ações não relacionadas à troca de mensagens.
2. Um processo de julgamento de testemunhos e de depoimentos fornecidos pelos agentes da aplicação. O processo de julgamento é uma das partes

constituintes do mecanismo de governança e foco principal desta dissertação.

3. Um *framework* que fornecerá infra-estrutura de software necessária ao processo de julgamento.

1.7 Organização do Documento

O restante do trabalho está organizado da seguinte forma:

O capítulo 2 apresenta alguns trabalhos relacionados, mostrando uma visão geral do estado da arte em relação à questão da governança de sistemas multi-agentes abertos.

O capítulo 3 apresenta o mecanismo de governança baseado em testemunhos, que tem como objetivo regular o comportamento de agentes. Nesse capítulo são apresentados também a arquitetura desse mecanismo e um processo de julgamento necessário para julgar a veracidade dos testemunhos.

O capítulo 4 apresenta o *framework* para o sub-sistema de julgamento, parte integrante da arquitetura do mecanismo de governança. O *framework* é uma infra-estrutura de software que fornece suporte necessário ao processo de julgamento apresentado no capítulo 3.

O capítulo 5 apresenta dois estudos de casos utilizados para ilustrar e validar a utilização do mecanismo de governança, mais especificamente, o sub-sistema de julgamento.

A conclusão, considerações finais e trabalhos futuros são apresentados no capítulo 6.