1
Introdução

O uso de sistemas multiagentes governados pela abordagem de leis nos permite, certamente, obter uma clara separação entre as regras que regem os relacionamentos inter-agentes e o código fonte que implementa funcionalidades estruturais.

Entretanto, é sabido que, por definição, sistemas multiagentes não possuem um comportamento previsível. Este fato é ainda mais agravado quando consideramos sistemas abertos, onde as fronteiras do sistema não estão bem definidas. Dessa forma, um sistema multiagente aberto deve estar preparado para qualquer tipo de agente que venha a interagir com o sistema.

Os avanços feitos para esse tipo de sistema têm, até então, se preocupado com questões ligadas ao comportamento do sistema como um todo. Com isso, muito enfoque foi dado ao modo utilizado para promover uma clara separação de regras de interação, a expressividade das linguagens que promovem tais separações e a desempenho global dos sistemas [22][11][7].

Esta abordagem garante uma melhor integridade do sistema como um todo, pois exige que os agentes estejam de acordo com as especificações das leis. Assim, para que um agente seja implementado, seu desenvolvedor deve conhecer com detalhes as operações permitidas no sistema. Dessa maneira, os riscos de comportamento inesperado em sua execução serão reduzidos.

Entretanto, o desenvolvimento dos agentes pertencentes a esses sistemas ainda continua desprovido de elementos que facilitem a interação com o sistema aberto e outros agentes participantes. Em outras palavras, o que foi feito até então resume-se a uma forma modular de limitar as ações de agentes num sistema aberto. O objetivo é sem dúvida garantir uma integridade global. No entanto, os agentes são forçados a se adaptarem a tais regras, ficando desprovidos de infra-estrutura para seu desenvolvimento.

Deve ser considerado que, durante o desenvolvimento de um agente, o desenvolvedor pode perder bastante tempo com o entendimento da lei de um sistema. Em alguns casos, quando não possui a correta disciplina pode ser levado a um processo de tentativa e erro, elevando ainda mais o tempo de desenvolvimento. Assim sendo, a repetição do comportamento do agente não é trivial, já que pode envolver diversos

outros agentes de comportamento imprevisíveis.

1.1 Definição do Problema

As implementações existentes de sistemas abertos são destinadas ao controle de interações dos agentes, ou seja, muitas vezes as mensagens são verificadas e validadas contra a lei definida no sistema em questão.

Algumas abordagens propõem o monitoramento e a prática de *logging* como solução para o desenvolvimento de agentes em um sistema aberto. Um sistema de log pode ser bastante útil para verificação de assertivas ou erros inesperados no sistemas. Um sistema de monitoramento pode ajudar ao administrador do sistema a tomar decisões referentes a performance.

No entanto, essas técnicas quando utilizadas para depurar o desenvolvimento de agentes podem voltar-se contra os desenvolvedores. Tornando a tarefa de descobrir possíveis falhas de implementação trabalhosa, o que pode levar ao desenvolvedor a uma lenta, desgastante e repetível tarefa manual de inspeção de log ou análise de monitoramento.

O uso de técnicas manuais orientadas à execução para verificação de software são conhecidas como ineficientes, pois além de lentas são mais suscetíveis ao erro humano. O desenvolvedor, ao verificar manualmente a execução de um código, está sujeito ao desgaste e, portanto, pode demorar mais tempo até encontrar o erro. Nesses casos, a repetição da verificação de execução deve ser feita toda vez que uma alteração de código ocorrer e o desenvolvedor notar que existe uma falha mas não sabe o porquê dela ter acontecido.

Existe uma solução proposta para resolver os problemas dessas técnicas através do uso de teste automatizado para verificação de software. A idéia é automatizar toda verificação feita manualmente por um software de teste. Assim, o processo não fica sujeito a erros de fadiga do desenvolvedor, podendo ser executado em tempo menor e repetido sem grandes custos sempre que uma alteração no código for feita. Se bem feito, o software de teste, ao comparar um resultado obtido com um esperado, poderá indicar exatamente onde é o problema, poupando bastante tempo do desenvolvedor.

Raramente uma aplicação vai permitir que todas as possibilidades de entrada sejam mapeadas para casos de teste [29]. Portanto, é quase impossível provar que uma aplicação está correta usando apenas teste automatizado, somente em alguns casos isso é viável, quando todas as combinações de entrada podem ser combinadas. Assim, a elaboração dos testes é uma tarefa bastante importante e deve-se considerar casos extremos, onde as possibilidade de falhas são maiores [27].

Quando falamos de desenvolvimento orientado a objetos, podemos ver diver-

sas iniciativas voltadas para o teste automatizado, como é o caso de frameworks como Junit [33] que é destinado ao teste unitário de objetos java e é bastante popular entre os desenvolvedores. Outras iniciativas se preocupam com outros níveis de teste, como a interface [1] ou a simulação de ambientes para teste [32]. Todas essas iniciativas indicam que o uso automatizado de testes pode ser uma boa abordagem para o desenvolvimento de software.

1.2 Solução Proposta

O objetivo desse trabalho é facilitar o desenvolvimento de agentes para sistemas abertos através do uso de testes automatizados. Especificamente, a implementação será destinada à implementação de agentes de software para o middleware M-Law, melhor descrita em [7].

Uma análise quantitativa é usada como motivação para medir o número de agentes necessários para automatização do desenvolvimento quando usamos o middleware M-Law. Um framework é proposto para implementação de agentes de teste, possibilitando diversas aplicações que facilitam os testes de agentes em diversos aspectos.

1.3 Contribuições

Nesse trabalho é apresentado um framework para implementação de agentes de teste em sistemas multiagentes abertos. O framework apenas considera agentes implementados para o middleware M-Law. As contribuições são listadas abaixo:

- Análise sobre como funcionam testes para sistemas multiagentes desenvolvidos para o middleware M-Law;
- Implementação de um framework que permite a criação de aplicações de teste automatizado;
- Algumas aplicações desenvolvidas pelo framework;
- Aplicações em potencial que podem ser desenvolvidas.

1.4 Organização da Proposta

Essa dissertação esta estruturada da seguinte forma:

– No Capítulo 2, apresentam-se os conceitos de sistemas multiagentes abertos e o que é XMLaw;

– No Capítulo 3, listagem e breve descrição de abordagens existentes para teste de sistemas multiagentes, possibilitando uma comparação da dissertação com essas soluções;

– No Capítulo 4, uma análise quanto aos testes em sistemas multiagentes abertos implementados com o middleware M-Law;

– No Capítulo 5, apresentação do framework para desenvolvimento de agentes stubs proposto. Algumas aplicações derivadas do frameworks são apresentadas;

– No Capítulo 6, alguns estudos de caso apresentam como o framework pode ser estendido ou suas aplicações utilizadas;

– No Capítulo 7, conclusões juntamente com a possibilidade de trabalhos futuros considerando-se o framework desenvolvido.