7
Trabalhos Futuros e Discussões

O uso de teste automatizado é uma prática que vem se tornando bastante popular entre os desenvolvedores, como podemos notar com as diversas aplicações de teste que surgem a cada dia. Espera-se que tais práticas façam efeito também no domínio de sistemas multiagentes.

Neste trabalho foi apresentada uma análise para o teste automatizado de sistemas multiagentes abertos desenvolvidos com o middleware M-Law. Esta análise fornece conceitos necessários para o desenvolvimento de testes automatizados nesse tipo de sistema.

A partir desta análise foi apresentado um framework que pretende suprir as dificuldades encontradas ao se desenvolver testes automatizados para esse tipo de sistema, ou seja, a necessidade de maior facilidade para criação de agentes *Stubs*. Com isso, o teste automatizado é beneficiado já que boa parte do esforço de desenvolvimento pode ser concentrado na criação desses agentes *Stubs*.

Algumas aplicações de teste foram criadas a partir do framework como forma de se exemplificar seu funcionamento. Outras instâncias demonstraram como o framework pode ser utilizado para resolver problemas mais complexos. A intenção, ao apresentar o uso dessas instâncias, é demonstrar que o trabalho aqui apresentado é útil para o desenvolvimento de aplicações de testes.

Este capítulo apresenta algumas sugestões de aplicações ou novos componentes para o framework. O objetivo é ajudar ainda mais no uso de teste automatizado para sistemas multiagentes abertos em XMLaw.

7.1
Teste de Integração entre os Agentes

Embora um estudo quantitativo do número de agentes necessários para o teste completo de um sistema aberto tenha sido apresentado, este trabalho apenas apresentou uma solução para o teste unitário.

Durante o desenvolvimento deste trabalho, especulou-se uma arquitetura para uma aplicação de testes que combine todos os agentes implementados na execução de scripts de teste. Para isso o framework aqui proposto já dispõe de ferramentas

que possibilitam a execução de diversos agentes *Stubs* com um único script e filtros que possibilitam a manipulação do conteúdo dos scripts em tempo de execução.

Seguindo uma abordagem semelhante ao teste unitário, o teste de integração poderia estender de um caso de teste JUnit [33]. Os métodos de template forneceriam as mesmas informações com exceção do agente em teste, que seria substituído por todos os agentes do sistema, e do arquivo de saída, que pode ser gerado de acordo com os agentes reais executados no teste. O modelo de classes é apresentado na figura 7.1.

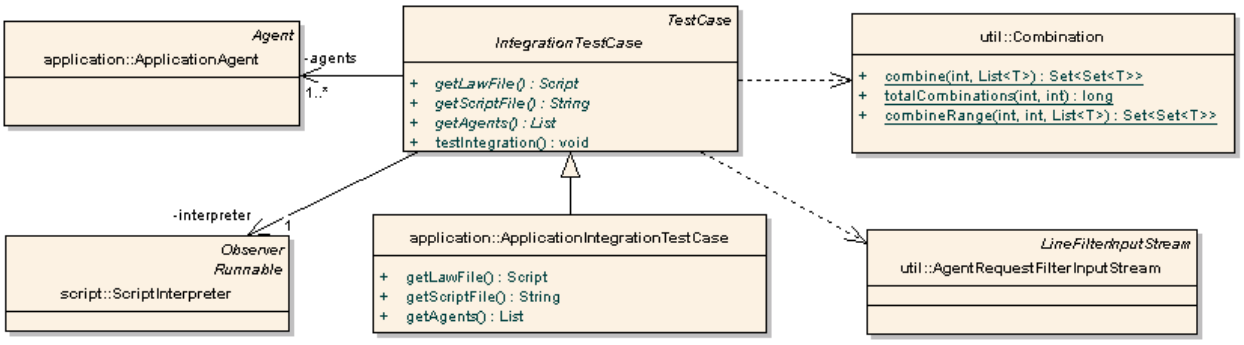

Figura 7.1: Modelo de Classes para Teste de Integração

Um componente de combinação poderia realizar todas as combinações necessárias aos agentes fornecidos pelo método de template. O caso de teste pode iterar pelas combinações e aplicar os filtros necessários para a remoção das requisições dos agentes no script de teste. Assim, para cada iteração um conjunto de agentes *Stubs* seria substituído pelos agentes reais equivalentes e o script seria executado juntamente com esse conjunto de agentes reais, realizando, portanto, todas as combinações necessárias para integração.

7.2 Outra Aplicações Interessantes

Outra aplicação em potencial seria a adaptação do mediador M-Law para fornecer scripts de teste. Desta forma, ao tentar entrar em um sistema aberto o agente poderia pedir o script de teste ao mediador e verificar se está apto para interagir com os outros agentes da organização em questão.

Uma variação dessa aplicação seria o próprio agente mediador simular o comportamento dos agentes no script. Nesse caso, o ele poderia autenticar que o agente está apto a entrar na organização ou negar sua participação caso encontre comportamentos indesejáveis.

Ainda no contexto de testes, componentes decoradores poderiam ser criados para enriquecer os relatórios de saída. Um componente decorador seria o inverso de um filtro. Enquanto o filtro se preocupa em trabalhar dados enviados por um

canal de entrada, o decorador está preocupado com um canal de saída. Na prática, a implementação java [30] considera ambos como filtro. O primeiro implementado por *FilterInputStream* e o outro por *FilterOutputStream*.

Um uso interessante para os decoradores poderia ser a geração de relatórios em formato html visando uma melhor compreensão. Outro tipo de decorador poderia gerar um relatório em formato xml de forma que fosse possível aplicar geradores de relatório automáticos disponíveis por ferramentas de automação de construção de projetos, tais como ant [14] e maven [15].

Eventualmente, as aplicações derivadas do framework podem servir também para a criação de aplicativos que não sejam de teste. A primeira instância apresentada fornecia uma interface de linha de comando. Nela, o usuário é obrigado a digitar no console as operações que deseja executar. Variações dessa aplicação podem implementar interfaces gráficas que forneçam a listagem de comandos de forma mais visual. Com essas aplicações, variações dos comandos poderiam ser criadas para domínios específicos e o esforço de implementação seria apenas a composição e criação de comandos.