Sistemas Multiagente Plenamente Distribuídos

Tiago Mazzutti[1], **Ricardo A. Silveira**[2]

[1]Departamento de Informática - Instituto Federal
de Educação Ciência e Tecnologia Catarinense (IFC) - Câmpus Concórdia
Caixa Postal 56 - 89700-000 - Concórdia - SC - Brasil

[2]Departamento de Informática e Estatística - Universidade Federal
de Santa Catarina (UFSC)
Caixa Postal 476 - 88.040-900 - Florianópolis - SC - Brasil

tiago.mazzutti@ifc-concordia.edu.br, silveira@inf.ufsc.br

Resumo. *A fim de termos sistemas multiagente plenamente distribuídos na Web é necessário que os agentes possam ser executados na máquina do cliente satisfazendo os requisitos da aplicação e ignorando os problemas habituais causados por questões como a segurança e execução concorrente. As soluções atuais disponíveis fazem uso de applets, que dependem da extensão, instalação ou configuração de plug-ins pelo usuário, ou através de aplicações completamente dependentes de servidores, que têm apenas uma fina camada no lado do cliente. Neste trabalho é apresentado um novo modelo de implementação de um ambiente de execução para agentes plenamente distribuídos. A solução proposta permite a execução de uma plataforma multiagente através do navegador do usuário e faz uso do Google Web Toolkit para para compilar Java para código JS que é compatível com a maioria dos navegadores web modernos, deixando o desenvolvedor em um nível mais abstrato de desenvolvimento, programando na linguagem Java.*

Abstract. *In order to have fully distributed multiagent systems in the Web it is necessary that agents can be executed on the client machine satisfying the requirements of the application and bypassing the usual problems caused by issues such as safety and concurrency in currently available systems. Current solutions available use applets, which depend on the extension, installation or configuration of plug-ins by the user, or through applications completely reliant on servers, which only have a thin layer on the client side. This paper presents a new model of implementation of an execution environment for fully distributed agents. The proposed solution will allow the execution of a multiagent platform through the user?s browser and makes use of Google Web Toolkit to compile Java to JS code that is compatible with the majority of modern browsers, leading the developer to a higher abstract level of development, programming in Java.*

1. Modelo para o Desenvolvimento de SMAs no lado do Cliente

No desenvolvimento de aplicações web no lado do cliente existem algumas características que devem ser levadas em consideração, entre elas a portabilidade, execução sem a instalação/atualização de plug-ins e facilidade de aprendizado das tecnologias envolvidas. Esses fatores têm influência positiva em todo o processo de desenvolvimento.

Ao se planejar e desenvolver um sistema multiagente destinado a execução no lado do cliente essas boas características devem ser bem exploradas. Sendo assim, o sistema deve ser compatível com o maior número de tecnologias possíveis, proporcionando a mesma experiência, independentemente de onde e como o aplicativo esteja sendo acessado. Do ponto de vista do desenvolvedor, é extremamente importante que a portabilidade seja obtida de maneira fácil e natural, evitando-se a necessidade de retrabalho ao acrescentar suporte a uma nova plataforma.

Forcar o usuário a instalar, estender, atualizar/configurar qualquer software no seu dispositivo para poder ter acesso a um aplicativo e um fator preocupante para qualquer desenvolvedor

web. Portanto, um SMA no lado do cliente deve explorar todos os recursos disponíveis sem a necessidade de novos plug-ins ou extensões.

2. Recursos e Restrições no lado do Cliente

Procurando atender as características e requisitos apresentados anteriormente, e buscando respostas para questões sobre como e onde executar?, quais os recursos de software/hardware disponíveis? e quais as principais restrições? realizou-se um estudo no sentido de elucidar como e sob que condições um SMA no lado do cliente poderia atender as necessidades de seus usuários.

Inicialmente, verificou-se através de um estudo das tecnologias web atualmente disponíveis, a maneira mais indicada para executar um SMA web no lado do cliente é através de um motor de execução JavasScript de um navegador web. No entanto, existem duas alternativas dentro desta solução: aplicações híbridas (web/desktop) - quando o motor e incorporado ao aplicativo - e aplicações *standalone* - onde o motor utilizado e parte do navegador web do usuário.

2.1. Aplicações Web/Desktop Híbridas

Um motor de navegador web permite que sejam criados aplicativos híbridos. Essas aplicações híbridas podem usar recursos de desktop e também pode acessar o conteúdo web. Como exemplos de motores podemos citar o Webkit [WebKit 2011, Nokia 2011, Network 2011]. Alguns motores de navegador web contêm uma biblioteca com recursos implementados que facilitam sua utilização. Por exemplo, QTWebKit [Nokia 2011] possui bibliotecas que incluem recursos NPAPI [Wikipedia 2011] disponíveis em uma interface JavaScript.

Embora com os motores dos navegadores Web seja possível a integração e utilização de bibliotecas destinadas a aplicativos desktop, execução nativa e a disponibilidade de implementações de código aberto, existem alguns problemas relacionados a portabilidade e a necessidade de instalação/configuração. Também existe a necessidade de incorporar o motor na própria aplicação, levando a um tamanho final do aplicativo muito maior. Esta abordagem possui outro ponto negativo onde o desenvolvedor e forçado a usar mais de uma linguagem de programação para implementar sua aplicação, levando a uma curva de aprendizagem e probabilidade de erros maior.

2.2. Aplicações Web *Standalone*

A ideia de uma aplicação web *standalone* é que os aplicativos sejam executados diretamente no navegador web do cliente, eliminando a necessidade de qualquer instalação e/ou configuração. Neste caso, as aplicações são desenvolvidas com base nos recursos já disponíveis nos navegadores (HTML e JavaScript), sendo que o usuário sempre tem acesso a versão mais atualizada do aplicativo automaticamente. A portabilidade é mais um fator favorável de aplicações *standalone*.

Por outro lado, existem problemas relacionados com a curva de aprendizado da linguagem JavaScript e algumas limitações em relação a característica *single-threaded* desta linguagem, e também, restrições relacionadas a segurança como por exemplo o acesso ao sistema de arquivos locais. Lidar com essas questões pode exigir um maior conhecimento do desenvolvedor, embora existam algumas soluções práticas para estes problemas, como por exemplo, o uso de *frameworks* como o Google Web Toolkit para o desenvolvimento.

Pelo exposto acima podemos observar que com o uso de recursos de uma aplicação web *standalone*, é possível obter muitas vantagens tais como atualizações automáticas, portabilidade e a não necessidade de instalação e configuração por parte dos usuários, portanto, e a melhor opção. Assim, apresentamos neste trabalho um modelo para o desenvolvimento de sistemas multiagente plenamente distribuídos, que utiliza apenas recursos disponíveis no navegador web do usuário.

3. Modelo Proposto

Para apoiar o desenvolvimento de sistemas multiagentes web utilizando os recursos nativos no lado do cliente e observando as restrições anteriormente relatadas, o modelo apresentado na Figura 1 foi proposto.

No modelo, há uma distinção intencional do lado do cliente e do lado do servidor. De acordo com o estudo realizado, o paradigma de agentes pode proporcionar maior autonomia e proatividade aos aplicativos rodando e utilizando recursos nativos no lado do cliente, permitindo que os agentes possam atender as necessidades de seus usuários, e, portanto, ter um sistema multiagente plenamente distribuído e autônomo.

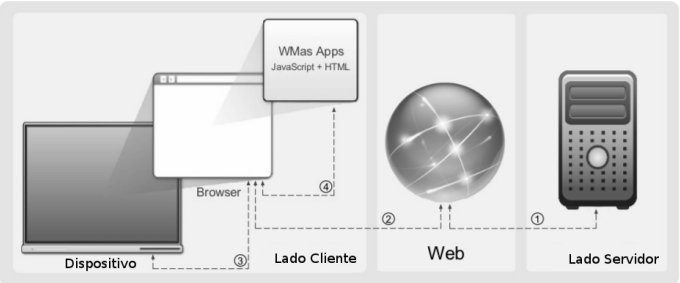

Figura 1. Modelo Proposto

No lado esquerdo da Figura 1, identificada como do lado do cliente, é possível observar que as aplicações executam em navegadores web usando apenas os recursos fornecidos por eles (JavaScript + HTML). As linhas bidirecionais tracejadas e enumeradas de 1 a 4 mostram os fluxo possível de informações.

O modelo não restringe a comunicação entre dois aplicativos que estejam sendo executados no mesmo navegador, em diferentes navegadores no mesmo cliente, ou entre diferentes navegadores em diferentes clientes. Em outras palavras, é possível que as aplicações web seguindo o modelo possam efetuar troca de dados entre duas aplicações rodando em diferentes clientes sem a necessidade de um servidor.

Outro aspecto importante é que o dispositivo representado no modelo pode ser qualquer dispositivo que permita a execução de um navegador Web, como desktops, tablets e smartphones.

Neste modelo, o servidor tem o papel de armazenar o código fonte da aplicação e prestar serviços globais, tais como autenticação, tabela de endereços, suporte a migração e banco de dados. Isso faz com que o servidor atue como uma espécie de coordenador das sub-plataformas em execução no lado dos clientes.

4. WebMAS

A solução em desenvolvimento para sistemas multiagente plenamente distribuídos apresentada nesta seção usa a ferramenta de Web Google Toolkit (GWT) [Murugesan 2007, Dewsbury 2007] para criar o front-end usando linguagem de programação Java e o GWT é responsável pela compilação cruzada para JavaScript otimizado que executa nativamente nos navegadores web mais utilizados atualmente. Como resultado desta proposta, a Figura 2 apresenta a arquitetura abstrata de um aplicativo desenvolvido usando WebMAS.

Na porção superior da imagem está representada a execução no lado do cliente do sistema multiagente. Com o uso de GWT e possível compilar um sistema multiagente desenvolvido em Java, gerando um arquivo executável (JavaScript + HTML) compatível e otimizado para cada navegador selecionado.

Do ponto de vista do desenvolvedor e virtualmente possível utilizar qualquer biblioteca Java com WebMAS. Portanto, com o WebMAS, é possível que qualquer *framework* de agentes desenvolvimento em Java, seja utilizado, levando em consideração que isso pode exigir que algumas adaptações ao *framework* em utilização sejam necessárias para integrá-lo as bibliotecas disponíveis ao desenvolvimento.

Na porção inferior da Figura 2 está representado o lado do servidor que executa parte do sistema multiagente desenvolvido com WebMAS. Além de hospedar a plataforma WebMAS

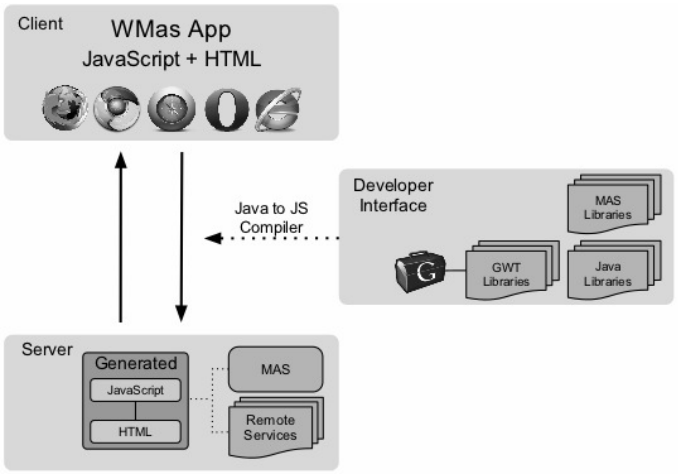

Figura 2. *Framework* WebMAS

global, o servidor e responsável por manter e servir os arquivos executáveis, e fornecer serviços remotos como comunicação, localização, páginas amarelas e páginas brancas.

5. Considerações Finais

Uma Implementação experimental para o desenvolvimento de sistemas multiagente plenamente distribuídos está em desenvolvimento. Os resultados obtidos são muito promissores e mostram que a abordagem pode ser uma solução para os problemas aqui apresentados.

A solução satisfaz os requisitos encontrados e tenta resolver as limitações de desenvolvimento de aplicações que executem no lado do cliente de uma forma elegante, com o uso de estruturas robustas e bem documentadas, como o uso do GWT. Alguns problemas como as limitações de JavaScript estão sendo contornadas com o uso dos novos recursos do HTML5.

Confrontado-se a solução aqui apresentada com as já existentes, que fazem uso de applets, obteu-se um grande avanço, uma vez que o WebMAS não depende da extensão, instalação e/ou configuração de plug-ins pelo usuário, e tem grande potencial para a utilização dos recursos de computação disponíveis no lado do cliente. Sendo assim, o WebMAS pode ser utilizado para o desenvolvimento de sistemas multiagente web, destinados a solução dos problemas computacionais apresentados pelo usuário.

Referências

Dewsbury, R. (2007). Google web toolkit applications.

Murugesan, S. (2007). Understanding web 2.0. *IT professional*, 9(4):34–41.

Network, M. D. (2011). Gecko. `https://developer.mozilla.org/en/Gecko`. Online: acessado em Julho-2011.

Nokia, C. (2011). Webkit in qt. `http://doc.qt.nokia.com/4.7-snapshot/qtwebkit.html\#details`.

WebKit (2011). The webkit open source project. `http://www.webkit.org/`. Online: acessado em Julho-2011.

Wikipedia (2011). Npapi — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=NPAPI&oldid=434583869`. Online: acessado em Julho-2011.