A New Internet Agent Scripting Language Using XML

Danny B. Lange
Tom Hill
Mitsuru Oshima

General Magic, Inc.
Sunnyvale, California, U.S.A.
danny@acm.org

Abstract

Java and other system programming languages are not ideal for software agent development on the Internet. We have found it very challenging to produce reliable yet lightweight agent systems. Even basic agents often require colossal amounts of highly complex code. We are addressing this issue by new agent scripting language and an associated execution environment. Taken together, these two developments provide a number of benefits to agent developers and users. Once the user learns the scripting language, he or she will be able to produce personal and enhanced agents. The scripting language supports rapid development since it allows programming at a much higher level than Java. It makes it easy to manipulate information in the XML format. Since the language is open-ended, it can also be easily extended with new tags written in the Java programming language.

Introduction

Experience gained from agent development projects (Lange and Oshima 1998) has led us to conclude that Java (Arnold and Gosling 1998) and other system programming languages are not ideal for software agent development on the Internet. We have found it very challenging to produce reliable yet lightweight agent systems. Even very basic agents often require colossal amounts of highly complex code. Perhaps most importantly, programming in these languages requires a skill set that most agent users do not posses, thereby preventing them from developing their own Internet agents.

We have taken steps to address this issue by developing a new agent scripting language and an associated execution environment. This new scripting language is based on the XML Internet standard (Harold 1998). It is optimized for manipulating and producing XML data and it is itself represented in the XML format. Drawing ideas from the popular Tcl (Ousterhout 1994) and LISP (Graham 1996), it is an extensible, persistent, and adaptive scripting language for creating agent-based applications. The execution environment consists of a network of communicating agent servers based on the HTTP and TCP/IP protocols. Agents can safely be hosted in this environment, which has been built to support long-term hosting of agents. The environment also provides a graphical agent workbench for developing and testing agents. The execution environment is entirely written in Java for optimal portability.

Taken together, these two developments provide a number of benefits to agent developers and users. Once the user learns the scripting language, he or she will be able to produce personal and enhanced agents. The scripting language supports rapid development since it allows programming at a much higher level than Java. Like HTML allows people to create personal Web pages, we anticipate that agent users with a technical flair can script personal and productive agents. It makes it easy to manipulate information in the XML format. Since the language is open-ended, it can also be easily extended with new tags written in the Java programming language.

Software Agents

Software agents are different from conventional software in a number of important ways. Most notably, they are personalized software objects that are continuously running, semiautonomous, and able to communicate with their surroundings.

Personalized. An agent is a personal software object acting on behalf of a specific user (we will use the term *principal* to denote the owner of an agent). Not only are data unique and local to each agent, but its behavior may also be personalized directly or indirectly by its principal. E.g., the principal may change the script of the agent to better fit her particular needs.

Continuously Running. Agents are long-lived software objects running 7 by 24 hours. Hosted on one or more computers in a network, agents are always ready to respond to events. For example an agent may try to find its principal whenever someone leaves her a voicemail.

Semiautonomous. Agents may hold enough information about their principals to act as proxies in e-commerce transactions. E.g., a principal may have provided an agent her user id and password for it to place proxy bids in an Internet auction.

Communicating. Agents are communicating with agents, network services, and their principals. Agents may retrieve product information from Web sites, exchange price information, and via a phone call, confirm an e-commerce transaction with the principal.

Useful agents are usually well-connected agents. We believe that agents should reside as an integral part of computer networks. Agents residing in the network are able to monitor local and remote information services ranging from the principals' private calendars to public Web sites, receive and process information, and deliver it to their principals whether they are on the road, watching television, or working in the office.

Agent Definition Format

The Agent Definition Format (ADF) is an agent scripting language developed by General Magic and based on the XML Internet standard. Taken together they provide a programming system for developing and using Internet agents. ADF has been designed that agent users with a technical flair can use ADF to script personal and productive agents.

Overview

ADF is a tag language similar to HTML (Musciano and Kennedy 1998). It provides generic programming tags for variables (cells), procedures (handlers), and flow of control.

Cell. Cells are the variables of ADF. Cells can be placed anywhere in the agent and can store XML data structures. A cell is defined as follows:

```
<cell name="meeting">
  <month>March</month>
  <day>4</day>
</cell>
```

Other tags include <set>, <unset>, and <recall>.

Handlers. Handlers are the procedures of ADF. Handlers are activated by via agent references (URLs). Handlers can also use the call tag to activate handlers in other agents.

```
<handler name="next-meeting">
  Next meeting is in
  <recall name="meeting.month"
</handler>
```

Here is an example of the call tag:

```
<call aref="#next-meeting"/>
```

Other handler tags include the asynchronous <send> with <callback> and <schedule>.

Flow of Control. ADF supports a number of tags to control the execution of handlers. Among those tags are the conditional tag (if) and the loop tag (foreach):

```
<if cond="$next-meeting.month == May">
  Other meetings in May are ...
</if>
```

ADF is an extensible scripting language very much like Tcl. The generic programming tags in ADF are realized as a set of *tag handlers*. New tag handlers can be written in Java and easily added to the Agent Server. A set of tag handlers can be grouped into a *wrapper*.

Hello World with ADF

This sample agent consists of a single handler named index, see Example 1. The index handler is set to return data of the text/html mime type. The body of the handler defines an html name space with html tags (body and center), and the Hello World greeting. The handler is activated from a Web browser with a url like this: http://server.net/hello.adf#index. The greeting Hello World will show up in the browser's window.

```
<adf version="0.1">
 <handler name="index" mimeType="text/html">
  <html xmlns="http://www.w3.org/TR/REC-html40">
   <body>
    <center>
     Hello World!
    </center>
   </body>
  </html>
 </handler>
</adf>
```

Example 1. The "Hello World" Example.

Web Site Access with ADF

In the second example we use an agent to retrieve the ranking of Danny and Mitsuru's book on Aglets from the Amazon Web site (www.amazon.com).
This agent uses two handlers. The first one (index) is the one that returns the html and the second handler (update) is the handler that retrieves the ranking number from the Aglets book's Web page at Amazon.

The index handler demonstrates the use of the aref attribute in the call tag (agent references similar to href attribute in the anchor tag in HTML). The agent reference allows agent to activate local handlers as well as handlers in other possible remotely located agents. The update handler demonstrates the use of an extension tag, post, from the Web wrapper. The Web wrapper is given a logic name in the XML name space expression in the first line of the agent. The actual ranking number is

retrieved by a regular expression and is the result of activating the `update` handler.

```
<adf version="0.1" xmlns:url="adf:svc:web">
 <handler name="index" mimeType="text/html">
  <html xmlns="http://www.w3.org/TR/REC-html40">
   <body>
    Aglet book is ranked <call aref="#update"/>.
   </body>
  </html>
 </handler>
 <handler name="update">
  <url:post action="http://www.amazon.com/query"
            value="keyword-query=Aglets"
            result="page"/>
  <regexp pattern="Rank:.*?>(.*?)&lt;"
          result="rank">
     $page
  </regexp>
  $rank
 </handler>
</adf>
```

Example 2. The "Aglet Book Ranking" Example.

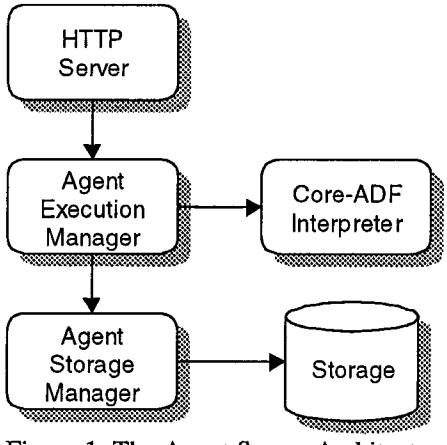

Figure 1. The Agent Server Architecture

Agent Server

The *Agent Server* is our implementation of an execution environment for ADF agents. The implementation consists of a *core-ADF* interpreter, an *agent execution manager*, an *agent storage manager*, and an *HTTP server*, see Figure 1.

The architecture is open to ADF language extensions. New *wrappers* (collections of tag handlers) can be added to the Agent Server. Each wrapper defines an additional set of tag handlers that extend the ADF language. We use the term wrapper because it often provides access to services and applications external to the Agent Server. We have developed wrappers that support extra ADF tags for *email*, *calendar*, *Web access*, *telephony*, and many more services and applications, see Figure 2.

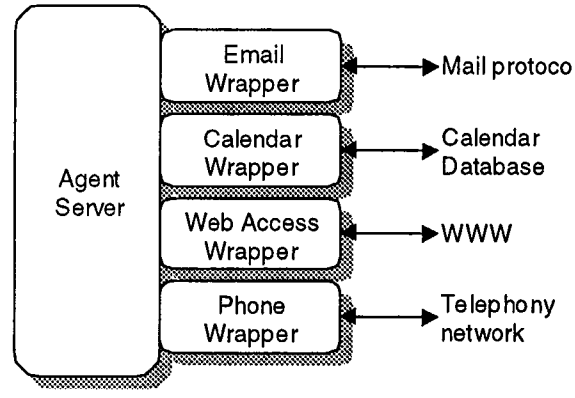

Figure 2. The Agent Server Wrappers.

Applications

As an example of a real-world agent application, we will briefly describe the *Auction Agent*. This wholly scripted agent is capable of monitoring its principal's auction portfolio at the eBay's Internet auction site (`www.eBay.com`).
The Auction Agent is what term a *voice agent*. That is, the agent is using a combination of text-to-speech, pre-recorded prompts, and advanced speech recognition in order to communicate with its principal via a telephone. The entire dialog scheme is scripted in a Dialog Definition Language based on XML.

The Auction Agent is a powerful tool for bidders and sellers. Internet auctions are very time sensitive and the Auction Agent offers online auction users ubiquitous and instantaneous voice access to their auction portfolios from regular as well as cell phones.

Here follows a brief sample dialog from the Auction Agent. The principal has been outbid on an eBay auction. She is in her car on her way home and is being called on her cell phone:

Agent: *Hello this is your Auction Agent. I have a notification for you. Please say your pass code.*
Principal: *123456*
Agent: *Hi there. You have been outbid on an item in your eBay portfolio. Here it is: Antique radio from 1923. Highest bid is 125 Dollars. Do you want to bid 130 Dollars?*
Principal: *No, bid 150 Dollars.*
Agent: *Did you say 150 Dollars?*

Principal: *Yes.*
Agent: *Just a moment while I place your bid at the eBay Web site. [on-hold music...] I successfully placed your bid. Is there anything else I can do for you?*
Principal: *No thanks.*
Agent: *Looking forward to hearing from you soon, bye.*

Conclusion

We have presented a quick overview of General Magic's ongoing effort to create a viable agent technology platform for the Internet. This platform is designed to function as "glue" between many services and media. The Auction Agent demonstrates an entirely new way of interacting with Internet services. Accessible via voice from any phone, the Auction Agent brings an entirely meaning to the term ubiquitous Internet access. We believe that is an example of agent technology that will change the landscape of Internet services and e-commerce.

References

1. Lange, D.B. and Oshima, M. 1998. *Programming and Deploying Java™ Mobile Agents with Aglets™,* Addison-Wesley.

2. Ousterhout, J.K. 1994. *Tcl and the Tk Toolkit,* Addison-Wesley.

3. Graham, P. 1996. *The ANSI Common Lisp,* Prentice Hall.

4. Harold, E.R. 1998. *Xml: Extensible Markup Language,* IDG Book Worldwide.

5. Arnold, K. and Gosling, J. 1998. *The Java™ Programming Language,* Addison-Wesley.

6. Musciano, C. and Kennedy, B. 1998. *HTML The Definitive Guide,* O'Reilly.