International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:3, 2007

WAF: an Interface Web Agent Framework

Xizhi Li，Qinming He

***Abstract*—**A trend in agent community or enterprises is that they are shifting from closed to open architectures composed of a large number of autonomous agents. One of its implications could be that interface agent framework is getting more important in multi-agent system (MAS); so that systems constructed for different application domains could share a common understanding in human computer interface (HCI) methods, as well as human-agent and agent-agent interfaces. However, interface agent framework usually receives less attention than other aspects of MAS. In this paper, we will propose an interface web agent framework which is based on our former project called WAF and a Distributed HCI template. A group of new functionalities and implications will be discussed, such as web agent presentation, off-line agent reference, reconfigurable activation map of agents, etc. Their enabling techniques and current standards (e.g. existing ontological framework) are also suggested and shown by examples from our own implementation in WAF.

***Keywords*—**HCI, Interface agent, MAS.

I. Introduction

As the web becomes more meaningful, dynamic and intelligent, agent based programming paradigm will play an increasingly important role in constructing complex web applications. Especially multi-agent system (MAS) has been applied in a variety of application domains such as computational market place, remote education, information collecting and querying on the web, and many role-based simulation environments. A trend in agent community is that they are shifting from closed to open architectures composed of a large number of autonomous agents. In the near future, most software products might be delivered to users in the form of personal agents or agent services. It is likely to see the uprising of a new enterprise (maybe named *agent enterprise*) which builds and sells customized agents to users with specific needs.

One of its implications could be that interface agent framework is becoming more and more important in multi-agent system (MAS); so that systems constructed for different application domains could share a common understanding in human computer interface (HCI) methods, as well as human-agent and agent-agent interfaces. However, interface agent framework usually receives less attention than other aspects of MAS. This is mostly due to lack of common understandings among developers in agent based HCI (human computer interface). For example, many MAS applications implement only a very simple interface agent layer using the same technology as other task agents; and some systems do not have a formal interface agent layer at all.

In our opinion, interface agent design should be carried out under the vision of a ubiquitous computing environment [9] and a topology-reconfigurable web where live participants are both human and agent. Therefore, in addition to regarding an agent as independent software that answers user queries, we hope that agent can also be treated in the same way as we treat other human beings. For one example, in our imaginative mind a person can be recalled by many different real world situations in which it used to play a certain role; for another example, when we are working on a plan concerning several people, we actually manipulate them in our conscious mind to create a new situation or simulation about a real world problem solving.

All these mental activities must have their counterparts in the interface agent framework of any open MAS application. In our everyday life, we accept the existence of an object only through different perspectives and from many situations in which it used to act. Likewise, in order to let people accept the existence of an agent, we must allow the user to create multiple situations in which the same agent can be referenced. This leads to, but not limited to (1) a naming convention to be able to locate an agent whether it is static or mobile, (2) an off-line agent reference database to store and retrieve a short history or status about on-line agents in the local (user) environment, (3) an activation map of agents to allow automatic or manual reconfiguration of the topology of intelligent agent references.

In this paper, an interface web agent framework is proposed, which defines an easy and effective solution to aforementioned problems in open MAS domain. It is based on our former project called WAF and a Distributed HCI template. In [1], a Distributed HCI template is proposed, which describes a bunch of HCI concepts and their relationships. Although these concepts are abstracted for general purpose distributed applications, they can be used in the MAS context with almost a direct mapping.

In section II, some related works are presented; in section III, the proposed framework will be introduced by means of our WAF project. Its enabling techniques and current standards (e.g. existing ontological framework) are also examined by examples from our own implementation in WAF.

II. Related works

Related works fall into two categories. One is open MAS application that has put more effort on its interface agent design;

Manuscript received November 11, 2004.

Xizhi Li is with Zhejiang University, college of computer science, Hangzhou, China, 310027 (website: http://www.lixizhi.net/ e-mail: lixizhi@zju.edu.cn).

Qinming He is with Zhejiang University, college of computer science, Hangzhou, China, 310027 (e-mail: hqm@cs.zju.edu.cn).

the other is web navigation methodology.

In the first category, the Friend of a Friend (FOAF) project [4] has some overlapping visions with us. FOAF project is about creating a web of machine-readable homepages describing people, the links between them and the things they create and do. In doing so, it has proposed an RDF ontological framework for producing XML content that is comprehensible to software agents. In WAF, we use a similar approach, but it covers more aspects of interface agent framework (such as navigation, history and activation map reconfiguration, etc). Moreover, we are describing a general framework independent of computer technologies used.

In the second category, there is Nestor and iBrowsor. They are not directly related to interface agent design. However, their ideas are valuable to our proposed framework, because the greatest potential of an agent enabled Internet lies in its wise navigation ability and computing topology re-configurability.

Nestor [5, 6] is a free browser that draws interactive web maps while surfing the Web. People may use web maps as bookmark archives. It promotes “constructivist navigation".

iBrowser [7] is a convenient Internet application, one of the first in the world that delivers "high-quality multimedia digital content" over the Internet at extremely high speed using conventional low-end hardwares. Innotive's iBrowser technology allows real-time interactivity in both wired and wireless environments to display digital image content regardless of file size with easy, intuitive interface.

III. INTERFACE WEB AGENT FRAMEWORK

We will first introduce the example project WAF. Then we will describe the proposed framework in translated Distributed HCI [1] terms.

A. *The example WAF project*

Web agent framework or WAF is a web-alike topology [8] multi-agent system application. It aims to create a visible virtual human relationship on the Internet by means of using agent to represent human master and act on their behalf reactively or autonomously (such as providing / retrieving information to / from other human or agent). In WAF, user’s navigation path can be visualized in an off-line tree graph (See Figure 1). The client browser of WAF will remember each visited agent as well as any downloaded artifacts such as a piece of news or a group of other related agents (e.g. friend agents). It allows client side reconfiguration of the topology of all these intelligent agents as well as data resources and save them into local map files. Agents and information in these map files can be updated automatically when they are reactivated or re-opened from the history records kept in the local memory pool(database); they can later be used as the starting point of a new navigation or just provide a group of related agent services to its user. See Figure 1.

In WAF, although most computing occurs at the place where agents are actually situated, users (including other agents) can customize references of external agent in different activation maps on their local environment.

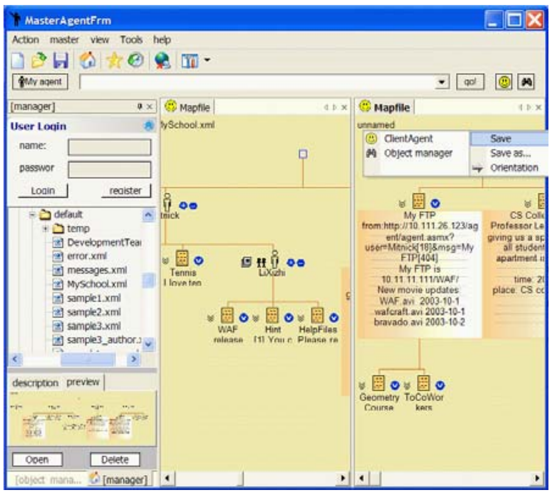


Figure 1. Several map files and user’s interaction with agents or data in these map files.

B. *Overview: the interface web agent framework*

In this section and next, we will propose the interface web agent framework. The framework is composed of seven interrelated interface objects as listed below. It is based on Distributed HCI template [1]. However, their names and explanations have been changed to suit the MAS context. Moreover, illustration of each interface object will be shown by examples in WAF project in section IV.

List of interface objects:

- DNode: Agent prototype which defines its communication language and schema.
- DNodeInstance: A unique agent situated in its runtime with a set of services and a set of actions it will perform whenever an internal state is reached. Both the agent and its actions are protected with locks.
- DNodeReference: A reference of DNodeInstance, which is used in off-line presentation in an activation map. The same agent can be referenced at many different places in both client and server runtimes
- ActivationMap: When navigating through the agent network, users will automatically produce a map file comprised of any visited agent. Agent references in a map file can be reorganized to form a new computing topology.
- History: A historical record of all the above objects so that any visited agent and navigation map files can be recollected in an offline mode by the Runtime.
- Owner: A master of agents, who owns privileges to their protected data and actions.
- Runtime: An environment where a set of activation maps are managed. This is usually an agent platform or runtime environment where computing and visualization of the interface agent framework occurs.

IV. WAF IMPLEMENTATION

The framework proposed here are trying to be general, although its current implementation is based on extant language and tools. Because the framework is an interface layer that might be adopted by heterogeneous MAS applications, we also try to minimize the amount of work to install it into an existing MAS framework. Besides MAS, other applications can also consider applying a similar approach such as in distributed computer games as in our previous work [2]. In the following text, details of the seven interface objects are given together with their implementations in WAF.

1) Agent prototype (DNode)

DNode := <∑, In, Out, URI>

∑:= alphabet in the agent's top-level communication language.

In:= All possible input, which can be further defined as a language($\subset \Sigma^*$) acceptable by this agent.

Out := All possible output, which can be further defined as a language($\subset \Sigma^*$) used in its outgoing message.

URI := A resource identifier where the agent prototype is defined

Several agents may be created from the same agent prototype, which must be made available in the form of a global Internet asset. Because most agent platforms choose to use one of the agent communication languages (ACL) [10] and RDF ontology [4] (or XML schema) in the form of XML encoding, an agent prototype can be defined in XML files on the Internet.

2) Agent Instance (DNodeInstance)

DNodeInstance := < DNode, address, state, actions, Δ, lock>

address := <namespace, local path>

state := <PublicState, ProtectedState>

PublicState: = public data or state that is accessible by other DNodeInstance

ProtectedState:= protected information which can be either static or dynamic.

actions := <visualization, lock>

visualization := an imagery that is reported to the user or fed to the (possibly virtual) environment

Δ := A transition function from $(state \times In)$ to $(state \times Out)$

Service:= {<name, IOPairs>}, where IOPairs $\subseteq In \times Out$ AND $\forall (i,o) \in IOPairs\{\exists s1, s2 \in state[\Delta(s1,i) = (s2,o)]\}$.

lock := require keys to open it

An agent instance is usually implemented as web services. However, we will introduce a more light-weighted approach to represent agent instances on the network. In this approach, we will insert a middle (file) layer between computing entities (i.e. between agents and users). Please see Figure 2.

We can regard the XML file layer as a collection of personal web pages owned by agents. Unlike human's web pages, they are managed by interface agents and written in machine readable format (Please refer to Semantic web). Common information is always included in each file in one or several predefined RDF schema. For example, the FOAF project [4] has proposed an ontological framework for specifying common agent information and their relationships (friendships).

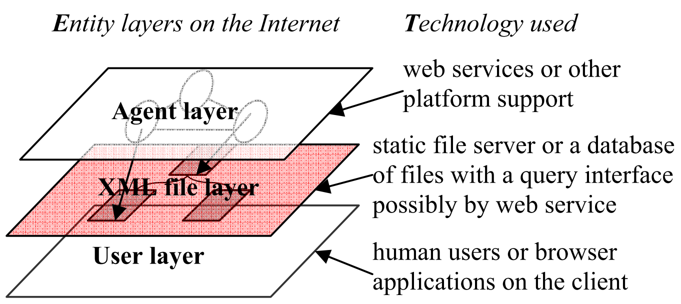
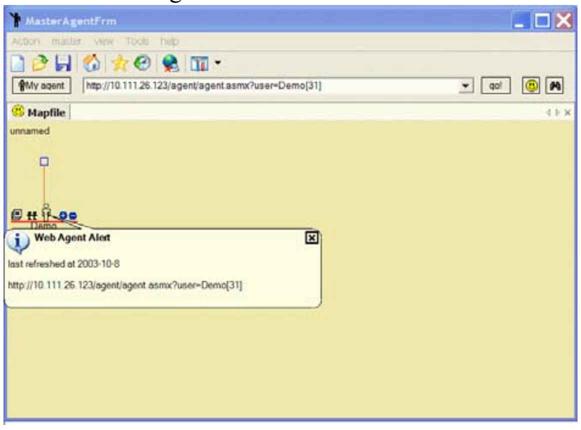

Entity layers on the Internet
Technology used
Agent layer
XML file layer
User layer
web services or other platform support
static file server or a database of files with a query interface possibly by web service
human users or browser applications on the client

Figure 2. Entity layers on the Internet

The advantage of using an additional file layer is that (1) file is the most light-weighted approach to interface agent, (2) file address is usually static, while agents could be mobile or static, (3) by allowing interface agents to publish information to their own unique web page, public (unprotected) information about the agent can be retrieved by users or other agents in a more efficient and convenient way, (4) file is more available on the network than an entity with computing capabilities. (e.g. either active agents or humans may be off-line, but file is always available.)

To increase the efficiency for software agents to access the XML file layer, web agent servers can use file databases to store files in the middle layer and provide a simple web service for their retrieval. Another advantage of using a file database with a query interface such as web service is that it enables a much secure and fine-grained retrieval and update to data sections in an XML file. For either approach, each file must be given a unique address on the Internet. In WAF, we use the database approach for the middle (file) layer. And the address of an agent instance is the web service's URL plus a local path name. Please see Figure 3.



http://10.111.26.123/agent/agent.asmx?user=Demo[31]

Web Agent Alert

last refreshed at 2003-10-8

http://10.111.26.123/agent/agent.asmx?user=Demo[31]

Figure 3. Download an agent reference by its address

In special cases, such as for personal information agents, we can get most public (unprotected) information solely from the middle layer. However, in most cases, we need to talk to the real software agent directly. Thus, the method to contact the agent layer should also be included in the XML file layer.

From 3) to 7), we propose the rest of the framework with explanations followed.

3) Agent reference (DNodeReference)

DNodeReference := <DNodeInstance, visibility>

visibility := A customized appearance of DNodeInstance that is used to display DNodeReference

4) Agent map file (ActivationMap)

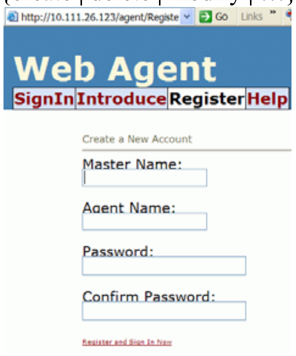
ActivationMap := <name, nodes, edges>
nodes := {DNodeReference}
edges := {< in , out>}
in := DNodeReference
out := DNodeReference

5) History

History := {< keywords, object, data>}
keywords := {time | name | address | …}
object := DNodeInstance | ActivationMap

6) Owner

Owner := <UserID, keys, privileges, {DNodeInstance}>
keys := {key}
privileges := {create | delete | modify | …}



Create a New Account

Master Name:

Agent Name:

Password:

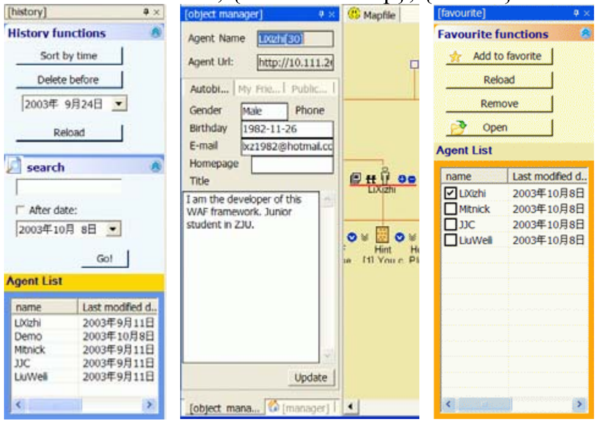
Confirm Password:

Figure 4. Create a master account

In WAF, after registering on an agent provider's website, users are given a master account and the address of its agent.

7) Client browser (Runtime)

Runtime := <address, {ActivationMap}, {Owner}>



2003年 9月24日

After date: 2003年10月 8日

Agent List

name	Last modified d..
LiXizhi	2003年9月11日
Demo	2003年10月8日
Mitnick	2003年9月11日
JJC	2003年9月11日
LiuWeli	2003年9月11日

Agent Name: LiXizhi[30]
Agent Url: http://10.111.2
Gender: Male
Phone:
Birthday: 1982-11-26
E-mail: lxz1982@hotmail.cc
Homepage:
Title:
I am the developer of this WAF framework. Junior student in ZJU.

Agent List

name	Last modified d..
✓ LiXizhi	2003年10月8日
Mitnick	2003年10月8日
JJC	2003年10月8日
LiuWeli	2003年10月8日

Figure 5. History in WAF browser

Once an *agent instance* has been downloaded to an *agent map file* of a *browser*, it becomes a reference of this agent on the map. Please see Figure 2, 3. An agent reference is a separate local copy of the web agent. (i.e. files in the middle layer are automatically stored in *history* of the *browser* after downloaded by it.) Disk files or a light-weight database system is usually the medium for storing *agent map files* and *agent references*. Figure 1, 5 shows some ways in WAF to retrieve off-line objects from the *history* such as by selecting disk files, entering query strings or date. Any agent reference in a map file can also be edited or annotated (the second one in Figure 1) and their topologies in map files can be reconfigured such as by drag and drop operations.

V. Conclusion

This paper examines the trends in open agent system especially in its HCI aspect, proposes an interface web agent framework and showed one possible implementation through our previous WAF project.

Applications of this interface framework include open multi-agent systems, 2D and 3D navigation systems with annotated links on its maps, web service browsers with reconfigurable service calls.

Conventions of existing user interfaces (like HTML, windows common controls, web forms) have a great impact on the design and functionality of current software applications. By making recommendations to a set of new user interface concepts, we hope it will help unleash the creativity of programmers, improve end-user experiences and promote distributed computing environment and agent technology.

Acknowledgment

Xizhi Li would like to thank his father for the insightful opinions about agent technology and also his friends WangTian *et al* for their involvement in the WAF project development and testing.

References

[1] Xizhi Li, "An HCI Template for Distributed Applications," IJCI. International Conference on Computational Intelligence 2004.
[2] Xizhi Li, "Using Neural Parallel Language in Distributed Game World Composing," in *Conf. Proc. IEEE Distributed Framework of Multimedia Applications.* 2005.
[3] Ricardo Choren *et al.* "Software Engineering for Large-Scale Multi-Agent Systems – SELMAS'04." Proceedings of the 26th International Conference on Software Engineering (ICSE'04).
[4] The Friend of a Friend (FOAF) project. http://www.foaf-project.org/
[5] Zeiliger, R., "Beyond Bookmarks: Enriching Web information," in proceedings of the Webnet 2000 Conference, San Antonio, USA, October 30- Nov 4, AACE (2000)
[6] Lilavati Pereira Okada, A., Zeiliger, R., "The Building of Knowledge through Virtual Maps in Collaborative Learning Environments," in proceedings of the ED-MEDIA 2003 conference, AACE, Hawaii, USA.
[7] iBrowsor. http://www.innotive.com/
[8] David Benyon, "The new HCI? Navigation of information space," Elsevier. Knowledge-Based System (2001).
[9] Irene. "The evoluation of objects into hyper-objects: will it be most harmless?." Personal and Ubiquitous Computing. Vol 7,no 3-4,2003.
[10] Haw Siang Hon. Agent Communication Language. ISE. 2001.