International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:6, 2007

An HCI Template for Distributed Applications

Xizhi Li

***Abstract*—**Both software applications and their development environment are becoming more and more distributed. This trend impacts not only the way software computes, but also how it looks. This article proposes a Human Computer Interface (HCI) template from three representative applications we have developed. These applications include a Multi-Agent System based software, a 3D Internet computer game with distributed game world logic, and a programming language environment used in constructing distributed neural network and its visualizations. HCI concepts that are common to these applications are described in abstract terms in the template. These include off-line presentation of global entities, entities inside a hierarchical namespace, communication and languages, reconfiguration of entity references in a graph, impersonation and access right, etc. We believe the metaphor that underlies an HCI concept as well as the relationships between a bunch of HCI concepts are crucial to the design of software systems and vice versa.

***Keywords*—**HCI, MAS, computer game, programming language

I. Introduction

COMPUTING, visualization and manipulation are closely related to each other in the design of a software system. A paradigm shift in one of them will raise new requirements in the rest two. Historically, computing has been the driving force of the other two HCI related techniques. For example, when computing was single threaded and relied on I/O polling, we had only text, buttons and bitmaps, etc to interact with; when it became multi-threaded and event-driven, we had windows, dialog boxes, menus, etc to access to a large group of functionalities in a small flat screen; when it employed a client/server architecture, we had HTML pages and server side computing, etc to exchange information on the Internet.

In recent years, computing has become more and more distributed, ubiquitous and intensive. As a result, distributed computing paradigms and data presentation techniques have been standardized, such as in web services, multi-agent system, semantic web/grid, etc; and more interaction techniques and 3D graphic rendering devices[6] are available to generate immersion and networked virtual environment on the Internet. However, HCI concepts used in their applications remain mostly unchanged. The lack of distributed HCI concept has, to some extent, limited the creativity of software developers and increased the difficulty of distributed framework design.

This paper proposes an HCI template for distributed applications. Three representative applications that we recently developed have been used in exemplifying these HCI concepts as well as explaining the metaphors that underlie them. It is insufficient to describe HCI concepts without explaining the computing paradigms that they represent. Therefore, we will show how visualization (HCI) is elicited by the computing framework and how computing will be stimulated or reconfigured by HCI manipulations.

In section II, short introductions to the three representative applications as well as related works will be given. In section III, the HCI template is presented in abstract terms, and some clues of applying them in our sample applications will be provided. Section II is most important, since it covers not only new HCI requirement, but also the relationships between computing and visualization.

Manuscript received October 22, 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).

II. Three Representative Applications

A. Web Agent Framework

Web agent framework [2] or WAF is a web-alike topology [12] multi-agent system application. It aims to create a visible virtual human relationship network on the Internet. Using agent to represent human beings and provide information to other visitors (including agent) is not a new idea. However, current implementations lack a flexible user interface to convince users that agents are active humanoid entities that exist on the network with them. This is due to static user interfaces and conventional object manipulation techniques adopted by these agent applications.

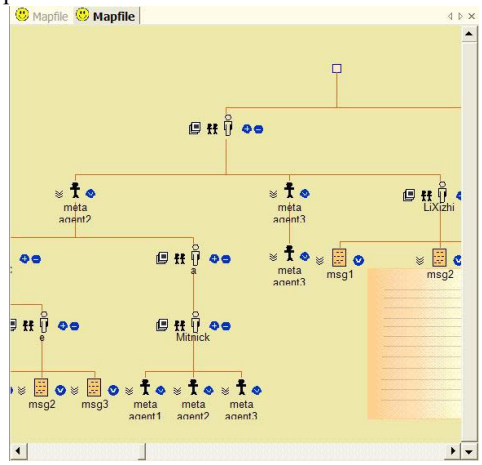

Figure 1. Users leave an off-line tree graph of visited agents and artifacts as they navigate through the agent network.

In WAF, user's navigation path can be visualized in an off-line tree graph (See Figure 1). The client explorer 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 reconfiguration of the topology of all these intelligent agents as well as data resources on the client side 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 web services to its user. See Figure 2.

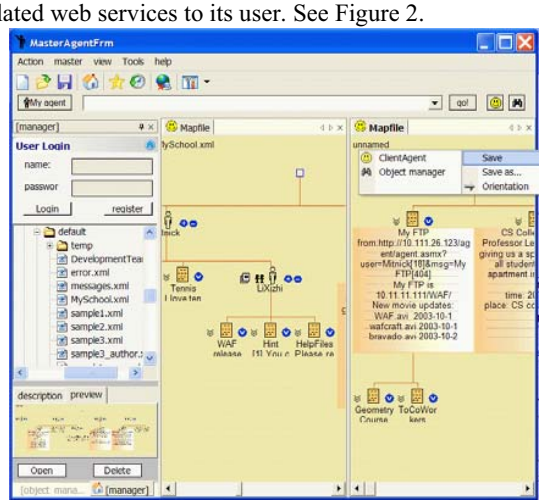

Figure 2. Several map files and user's interaction with agents or data in these graph files.

In WAF, although most computing occurs at the place where agents are actually situated, users (including other agents) can customize foreign agent references in different activation maps on their local environment. 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. There is a set of client side HCI concepts defined in our template to give software systems this ability. Please see section III.

B. Parallel World: a 3D Internet game

While some current effort on Semantic Web/Grid tells a computer program exactly what to compute and visualize on the Internet, there still lacks formal approaches on telling it how to compute or visualize. Web3D technology[3] such as X3D language is exploiting a new possibility of expressing networked virtual environment[4] that is as distributed as web pages and more interactive than just hyperlinks. X3D code generally describes a tree hierarchy of nodes with routes or stimuli specified for their input fields. Nodes can be associated with script files or other Internet assets. Script files contain logics and hence logics can be distributed on the Internet (the latter needs special runtime environment support where scripts are situated). Although most X3D applications involve only a static assembly of dynamic scene data from one or several file servers on the Internet, the existence of scripts and dedicated runtime environment on both client and server makes it possible to construct active virtual environment spanning the network.

Thus, networked 3D content or virtual world is, in fact, a composition of the computing result of many distributed active entities on the Internet. Like current HTML browsers, the next generation web browsers must be able to (1) locally simulate and present 2D and 3D visualizations to its user (2) reconfigure or interact with any downloaded entities in its local environment (3) simultaneously communicate with other computing entities on the network to keep them up-to-date and alive. As the relationship between computing and visualization becomes more complex in future applications, human computer interface design will become a hot issue. One way to think of its complexity is through designing 3D computer game engines based on this framework.

In [1], we have proposed and implemented a game engine framework called ParaEngine for developing games based on distributed game world data and logic. I.e. game data and logic are contained in neuron files which are distributed on the Internet. Neuron files are program codes written in Neural Parallel Language (NPL) and deployed to its language runtimes. One can compare a neuron file to a light-weight web service modeled as an abstract neuron and simulated in a special runtime environment.

A screen shot of the game is shown in Figure 3. All text, buttons, graphics and sounds in it are mental elicitations of a neural network constructed by NPL. Players might walk around, talking with other NPCs (None-player characters), complete complex tasks all in a continuous infinitely-large distributed game world. It is like browsing 3D web pages [5], however, it is more interactive, purposeful and fun. We will see later how the HCI template can be used in this framework.

Figure 3. Screen shots from *Parallel World* game

C. Neural Parallel Language or NPL

NPL is the programming language used in constructing the game in Section II.B.

In our viewpoint, the compiling of code (that targets distributed environment) may also be carried out in a distributed manner (from command-line compiler to rich HCI enabled ones with network capabilities); the next generation

high-level language may be able to express adaptive and distributed behaviors with its own language primitives; its compiler may be able to generate low-level code that runs on any part of the network; and its development environment may allow visualized design of any parallel code and deployment scheme. In other words, the coding and compiling process may both be carried out in a distributed manner and environment. This calls for a new language dedicated to this task and a new human-computer interface (HCI) adopted by its compiler and runtime environment.

With this vision, we had proposed a neural network based programming paradigm called Neural Parallel Language (NPL). An informal description of the language can be found in section 3 of [1].

III. The HCI Template

In previous sections, we have described three representative domains of computing and their visualizations. Although their HCI differs greatly in both functions and appearance, there is a set of HCI concepts that is common to all of them. These concepts are described in abstract terms in this section. We hope they will help designers to define their software systems at an early stage.

A. Related works of HCI template

Some related works are dedicated to the requirement, analysis and design phase of a distributed application, such as some agent development platform [7, 8] and UML extensions (like AUML [9]). They are focusing on the internal hierarchy of an agent framework. There are also tools and interface models to provide assistance to interface developers [10, 11]. Many methodologies have been developed to support the design phase of Multi-Agent Systems (MAS). Among them are AUML, Gaia and MAS-CommonKADS. These methodologies offer a set of diagrams to help conceptualize and represent the system under development. And there are also tools and network computer games that automatically generate the presence of agent networks over the Internet.

Our purposed HCI template is not a concrete implementation to the above helper software, nor is it a substitute for these proposed visual tools; instead, we aim to suggest to application developers that much work can be refined and new important functionalities may be needed in future software applications.

B. HCI Template Overview

Some common requirements of distributed HCI are (1) a naming convention for distributed object references, (2) using multiple references of the same object in different places, (3) a network transparent reconfiguration of the topology of objects, (4) a mechanism to remember visited objects and their activation topology, (5) a certification mechanism for runtime environments as well as for each individual action that is performable by an object.

To visualize the aforementioned requirements, the following top-level interface objects are defined in the HCI template:

- DNode: an entity prototype defined on the network
- DNodeInstance: an instance of DNode(s) with a unique address on the network.
- DNodeReference: A reference of DNodeInstance, which is used in off-line presentation in an activation map.
- ActivationMap: a collection of DNodeReference organized in a directed graph.
- History: A historical record of all the above HCI objects.
- Owner: An authorized entity which owns a collection of DNodeInstance.
- Runtime: An environment where a set of activation maps are managed. Both client and server are regarded as runtimes.

C. The HCI Template

1) DNode object

DNode := <∑, In, Out, URI>

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

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

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

URI := A resource identifier where the DNode is defined

2) 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

3) DNodeReference

DNodeReference := <DNodeInstance, visibility>

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

4) 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 | …}

7) Runtime

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

D. Explanation and application

We will now map these HCI concepts to the first two representative applications in Section II. The metaphors that underlie this mapping as well as their relationships will provide useful insights to both user interface design and software system's function specifications. Please see TABLE I.

TABLE I.
MAPPING FROM HCI TEMPLATE TO WAF AND PARALLEL WORLD GAME

HCI concepts	Web Agent Framework	Parallel World game
DNode	Agent prototype which defines its communication language and schema.	Neuron file prototype and its message transfer protocol.
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.	A unique neuron file that is situated in its runtime with certain transfer functions and a set of multimedia elicitations that will be fed to the environment (game engine) whenever an internal state is reached.
DNodeReference	The same agent can be referenced at many different places in both client and server runtimes	A neuron file can be referenced by any game engine runtime, without its logic (code) being downloaded.
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.	Game world logics are scripted in many neural networks (the maps) which are distributed on the Internet. Client and server neural networks exchange messages between runtimes to synchronize their visualizations.
History	Any visited agent and navigation map files can be recollected in an offline mode by the Runtime.	Even though the game client is disconnected from servers, users could still play in the networked virtual environment in an off-line manner.
Owner	A master of agents who owns privileges to their protected data and actions.	A host player or a visitor.
Runtime	Agent platform or runtime environment where computing and visualization of each agent occurs.	NPL runtime (embedded in a game engine) where activation and execution of local neurons occurs and messages to external neurons are routed via network.

IV. CONCLUSIONS

We proposed an HCI template for economic use of distributed resources. New computing frameworks such as multi-agent system, distributed neural network based computing require new HCI concepts designed for their efficient manipulation in both development and application stage. The proposed HCI concepts are summarized from three representative applications we have previously implemented. We hope it could provide some useful insights and read-to-use HCI patterns in the design of future distributed applications.

REFERENCES

[1] Xizhi Li, "Using Neural Parallel Language in Distributed Game World Composing," in *Conf. Proc. IEEE Distributed Framework of Multimedia Applications.* 2005.
[2] Xizhi Li, Qinming He. "WAF: an Interface Web Agent Framework." IJIT. International Conference on Information Technology 2004.
[3] Web3D Consortium. http://www.web3d.org
[4] Singhal, S., and Zyda, M. (1999). Networked Virtual Environments: Design and Implementation, ACM Press.
[5] Jed Hartman and Josie. The VRML 2.0 Handbook: Building Moving Worlds on the Web Wernecke (1996) Addison-Wesley. ISBN 0-201-47944-3.
[6] Bowman, D. A., and Hodges, L. F. (1999). "Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments." Journal of Visual Languages and Computing, 10, 37-53.
[7] Bauer., B. "UML Class Diagrams Revisited in the Context of Agent-Based Systems." In the econd International Workshop on Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada, May 28- June 01. 2001. pp 1-8.
[8] Bernon., C., Gleizes., G., Peyruqueou., S., Picard., G. ADELF, "a Methodology for Adaptive Multi-Agent Systems Engineering." Workshop Notes of the Third International Workshop Engineering Societies in the agents world, 16-17 septembre 2002, madrid, spain, pp. 21-34.
[9] Odell., J., Van Dyke Parunak., H., and Bauer., Bernhard. "Extending UML for Agents." Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, Gerd Wagner, Yves Lesperance and Eric Yu eds., Austin, Tx, pp 3-17, AOIS Workshop at AAAI 2000.
[10] Puerta, A.R. "State-of-the-Art in Intelligent User Interfaces" Knowledge-Based Systems, 10(5), 1998, pp. 263-264.
[11] Puerta, A.R. "A Model-Based Interface Development Environment." IEEE Software, 14(4), July/August 1997, pp. 41-47.
[12] David Benyon, "The new HCI? Navigation of information space," Elsevier. Knowledge-Based System (2001).