Developing Multi-User Online Games with Agents

Agostino Poggi
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parco Area delle Scienze 181A, Parma, 43100
ITALY
poggi@ce.unipr.it http://www.ce.unipr.it/people/poggi

Abstract: - This paper presents a software framework able to extend the coverage of virtual communities across the boundaries of the everyday Internet by providing ubiquitous and integrated access to applications regardless of the users connecting though conventional, fixed-network computers, mobile, wireless-network devices or even interactive digital television set-top boxes. This software framework is an extension of the JADE multi-agent development environment and has been experimented for the realization of a special kind of virtual community applications, i.e., multi-user online games.

Key-Words: - Multi-agent systems, Mobile devices, DTV, Multi-games, Java.

1 Introduction

Interactive multi-user internet games have their origins from programs sharing a common heritage known variously as MUGs (Multi-User Games), MUDs (Multi- User Dungeons or Multi-User Dimensions) and MUAs (Multi-User Adventures) [31]. Along with the improvements to computer graphics, audio and real-time processing, multi-user games have also improved in terms of accessibility moving from computers connected through wired networks to small devices (e.g., mobile phones) connected through wireless networks and to digital televisions where user are usually connected integrating television broadcasting with internet connections (see, for example, [9] and [29]).

Moreover, multi-user games took and may take advantage of multi-agent systems technologies (see, for example, [6], [16], [17] and [22]). In fact, their intelligent and autonomous behavior make agents suitable to act as addition virtual users, manage the virtual environment where the game is situated, and act in place of user on the basis of her/his preferences and her/his previous behavior. Moreover, multi-agent systems may help in the development and execution of a multi-user game on a distributed and heterogeneous networked environment, because agents acting on low power devices can delegate actions to other agents on other devices, agents can move on the network from a device to another devices, remote agents can work for their users when they are disconnected from the network.

This paper presents how JADE, a software framework to aid the development of multi-agent applications, has been extended for the realization of multi-user games for users that can also act on mobile devices and digital televisions . The paper is organized as follows. Section 2 introduces interactive digital television technology. Section 3 describes the JADE agent architecture and its main features. Section 4 shows how agents are specialized for working in mobile environment on devices with limited memory and processing power. Section 5 shows how agents are specialized for working in a interactive digital television application. Section 6 shows an entertainment application that can involve users acting through computers, mobile phones and digital televisions. Finally, section 7 concludes the paper and outline some future work.

2 Interactive Digital Television

Interactive digital television (DTV) is a technology which combines broadcast video, broadcast radio, computing power and the Internet. This combination of different mediums and services provides the viewer with a new experience. This is possible because of an ongoing transition from analogue TV to digital TV.

We can clearly say that the digital technology is driving television towards a new world of amazing possibilities, where spectator is no longer limited to observe contents selected by the operator. More and more, new dynamic and interactive services are being introduced in everyday digital TV: complementary information to audio-visual contents, electronic program guides, selection of properties in configurable contents (language, camera angle or particularized advertisement), pay-per-view, etc. So we can consider the term "interactivity" as the possibility for the consumer to actively influence the behavior of broadcasted television, services and

applications. This can be accomplished, for example, by means of a remote control for channel hopping, by fetching information via teletext or by sending data via an interaction channel. This all creates a context, which allows to have a mutual influence between the viewer, the broadcaster and the application provider.

The interactive TV technology, as we will see in next section, is based on the broadcasting of a digital transport stream which permits operators to mix traditional audio-visual contents with binary data, so making possible to deliver multimedia applications to be executed in a digital TV or in a set-top box. These applications, synchronized with audio-visual contents, adapt themselves to the spectator's characteristics, implement interaction with users and provide return channels for communication with content providers.

The Multimedia Home Platform [28] is a standard published by the DVB (Digital Video Broadcasting) consortium in 2001, which consists of a combination of broadcast and Internet, offering a common Application Programming Interface (API) accessible for everyone who wants to develop applications, set-top boxes, television devices or the combination of all.

Fundamentally the MHP standard defines a generic interface between interactive digital applications and the terminals on which those applications execute. This interface decouples different provider's applications from the specific hardware and software details of different MHP terminal implementations.

The MHP extends the existing, successful DVB open standards for broadcast and interactive services in all transmission networks including satellite, cable, terrestrial, and microwave systems.

The applications downloaded to the MHP terminals, typically set-top boxes, are Java applications called Xlet, built on a suite of APIs tailored specifically for the interactive TV environment: Java TV APIs [25], HAVi (user interface) [20], DAVIC APIs [14] and DVB APIs [15].

The 1.1 version of the MHP standard defines three profiles:

1. Enhanced Broadcast: the basic profile which only allows the enrichment of the audio-video contents with information and images that can be viewed and navigated by users on the TV screen;
2. Interactive Broadcast: the intermediate profile in which uses the set-top box return channel to supply services with a higher level of interactivity: in fact this profile supports the loading of MHP applications not only through the broadcast channel but also through the return channel;
3. Internet Access: this profile, using the return channel, allows the user to access to the Internet contents.

As we can understand from the previous description of the MHP levels, the interactive TV paradigm is based on two different channels: a broadcast channel from the application/contents provider to the set-top box and a return channel (dial-up, GPRS, ADSL, Ethernet, etc.) from the set-top box to the provider.

Figure 1 shows the use of a carousel to continue play-out a Java application: the application and the corresponding audio-visual material are the multiplexed to form a single MPEG-2 transport stream.

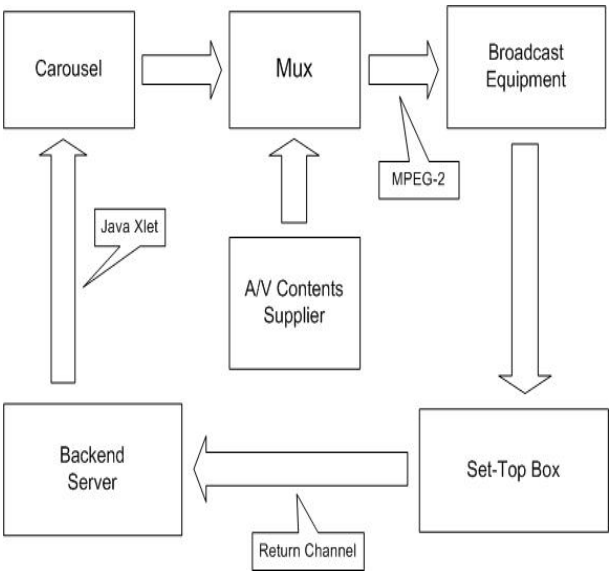

Fig. 1. The interactive broadcasting chain.

The resulting broadcast is received and decoded by the set-top box, the audio-visual content played and the Java application run.

Subsequent user interactions with the application lead to information being sent via the return channel to a back-end server. Depending on the application, this information may result in modifications to the current application content (i.e. voting information) or stored for later processing in a database present on the server (i.e. for an online shopping application).

About the transport, in digital TV MPEG-2 is not only a standard for encoding audio and video, but it is also used as the means by which raw data and applications are transported in the broadcast stream. In particular, DVB has extended the traditional scheme and way to use MPEG-2 for MHP by specifying how to embed a Java application within the stream, this includes information on how to

specify the main class, class search path and the application argument list etc.

Although MPEG-2 provides a means of transporting the Java applications along the audio-visual content, to support the possibility that the user may change channel and select the Java program at any point of the transmission, the same application has to be broadcasted in loop. This is exactly what a broadcast carousel does: it keeps playing the same application around and around: the application is continuously multiplexed with the audio-visual content for the transmission, to allow the viewer to access to the interactive TV application whenever he wants.

About the applications, as we said previously, we have Java applications, but they are not complete Java applications in the normal sense. These applications are much more like applets in that they are loaded and run by a life cycle manager residing on the set-top box.

3 JADE Software Architecture

JADE (Java Agent DEvelopment framework) is a software framework designed to aid the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems (see, for example [1] and [2]). JADE is one of the most known and used software framework for realizing multi-agent systems and several companies used and are using it for very different application sectors including network management, supply chain management, rescue management, fleet management, health care, auctions, tourism, etc. (see, for example, [5], [8], [11], [21], [26] and [30]). JADE is an active open source project, and the framework together with documentation and examples can be downloaded from JADE Home Page [23].

JADE is based on a peer-to-peer communication architecture. The intelligence, the initiative, the information, the resources and the control can be fully distributed across mobile terminals as well as computers connected to the fixed network. The environment evolves dynamically together with peers – that in JADE are called agents – that appear and disappear in the system according to the needs and the requirements of the application domain. Communication between the peers, regardless of whether they are running in the wireless or wired network is completely symmetric with each peer being able to play both initiator and responder roles.
JADE is fully developed in Java and is based of the following driving principles:

- Interoperability – JADE is compliant with the FIPA specifications [18]. As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard.
- Uniformity and portability – JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version (edition, configuration and profile). More in details, the JADE run-time provides the same APIs both for the J2EE, J2SE and J2ME environment. In theory, application developers could decide the Java run-time environment at deploy-time.
- Ease of use – The complexity of the middleware is hidden behind a simple and intuitive set of APIs.
- Pay-as-you-go philosophy – Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them, neither they add a computational overhead.

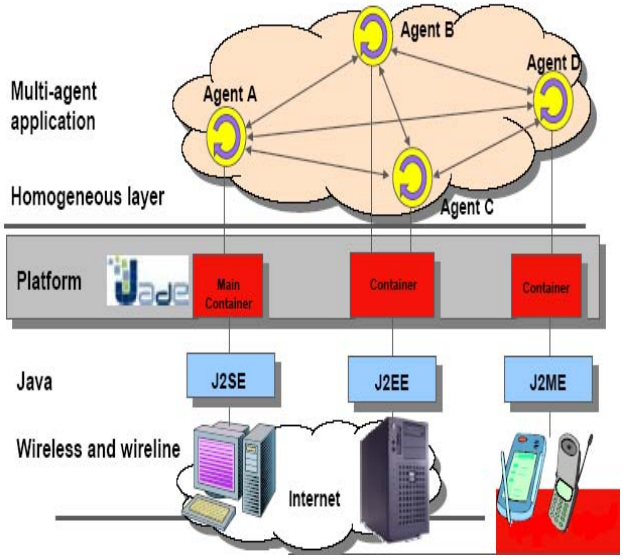

Fig. 2. The architecture of a JADE agent system.

JADE includes both the libraries (i.e. the Java classes) required to develop application agents and the run-time environment that provides the basic services and that must be active on the device before agents can be executed. Each instance of the JADE run-time is called container (since it "contains" agents). The set of all containers is called platform and it provides a homogeneous layer that hides to agents (and to application developers) the complexity and the diversity of the underlying tires (hardware, operating systems, types of network, JVM). Figure 2 draws the architecture of a JADE agent system deployed on a set of heterogeneous computing nodes.

The JADE run-time memory footprint, in a MIDP1.0 environment, is around 100 KB, but can be further reduced until 50 KB using the ROMizing technique [4], i.e. compiling JADE together with the JVM. JADE is extremely versatile and therefore, not only it fits the constraints of environments with limited resources, but it has already been integrated into complex architectures such as.NET or J2EE [7] where JADE becomes a service to execute multi-party proactive applications. The limited memory footprint allows installing JADE on all mobile phones provided that they are Java-enabled and support them also in complex applications [30]. Analyses and a benchmarks of Scalability and Performance of the JADE Message Transport System are reported by different works (see, for example, [13] and [32]).

From the functional point of view, JADE provides the basic services necessary to distributed peer-to-peer applications in the fixed and mobile environment. JADE allows agents to dynamically discover other agents and to communicate with them according to the peer-to-peer paradigm. From the application point of view, each agent is identified by a unique name and it provides a set of services; it can register and modify its services and/or search for agents providing given services, it can control its life cycle and, finally, it communicates with all other peers.

Agents communicate through the exchange of asynchronous messages, i.e., agents just sends messages to other agents without waiting for an answer, which is a communication model almost universally accepted for distributed and loosely-coupled communications, i.e. between heterogeneous entities that do not know anything about each other. Agents are identified by a name that does not statically include the actual distributed object reference of the agent and therefore there is no temporal dependency between communicating agents. The sender and the receiver could not be available at the same time. The receiver may not even exist (or not yet exist) or it could be not directly known by the sender that can specify a property (e.g. “all agents interested in football”) as an intentional description of destination agents. Because agents identifies each other by their name, hot changes of their object reference are transparent to applications.

Despite this type of communication, security is preserved, because, for applications that require it, JADE provides proper mechanisms to authenticate and verify rights and capabilities assigned to agents (see, for example, [10] and [27]). When needed, an application can verify the identity of the sender of a message and prevent any not allowed action. For instance, an agent may be allowed to receive messages from the agent representing its boss, but not to send messages to it. All messages exchanged between agents are transported within an envelope that includes only the information required by the transport layer. This allows, among other things, to encrypt the content of a message separately from the envelope and to guarantee reachability with no loss of security.

The structure of a message complies with the ACL language defined by FIPA [18] and includes fields, such as variables indicating the context a message refers to and the amount of time that is waited before an answer is received, aimed at supporting complex interactions and multiple parallel conversations.

To further support the implementation of complex conversations, JADE provides a set of skeletons of typical interaction patterns to perform specific tasks, such as negotiations, auctions and task delegation. By using these skeletons (implemented as Java abstract classes), programmers can get rid of the burden of dealing with synchronization issues, timeouts, error conditions and, in general, all those aspects that are not strictly related to the application domain. To facilitate the creation and handling of message contents, JADE provides support for automatically converting back and forth between formats suitable for content exchange, including XML and RDF, and the formats suitable for content manipulation (i.e. Java objects). This facility is integrated with some well known ontology-creation tools, e.g. Protégé, allowing programmers to graphically create the ontology agents should use to validate and provide semantics to messages.

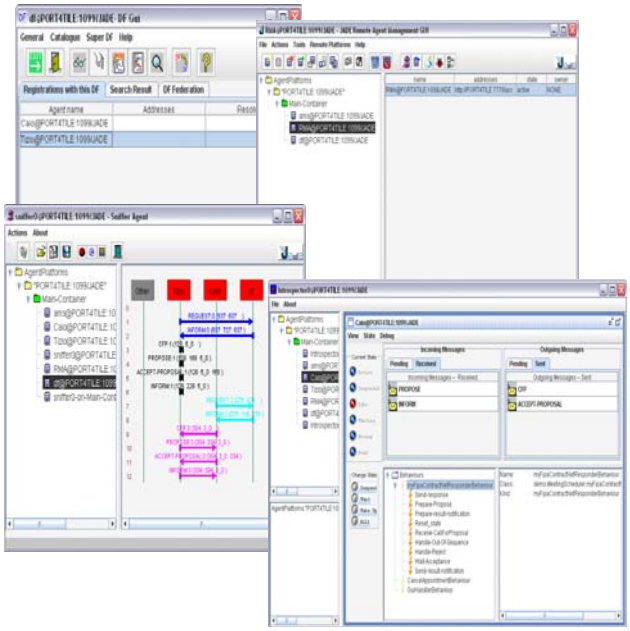

Fig. 3. JADE software development tools.

To increase scalability and to meet the constraints of environments with limited resources, JADE provides the opportunity of executing multiple parallel tasks within the same Java thread. Several elementary tasks, such as communication, can then be combined to form more complex tasks structured as concurrent finite states machines.

JADE supports mobility of code and of execution state. That is, an agent can stop running on a host, migrate on a different remote host (without the need to have the agent code already installed on that host), and restart its execution from the point it was interrupted (actually, JADE implements a form of not-so-weak mobility because the stack and the program counter cannot be saved in Java). This functionality allows, for example, distributing computational load at runtime by moving agents to less loaded machines without any impact on the application.

The platform also includes a naming service (ensuring each agent has a unique name) and a yellow pages service that can be distributed across multiple hosts. Federation graphs can be created in order to define structured domains of agent services. Another very important feature consists in the availability of a rich suite of graphical tools supporting both the debugging and management/monitoring phases of application life cycle. By means of these tools, it is possible to remotely control agents, even if already deployed and running: agent conversations can be emulated, exchanged messages can be sniffed, tasks can be monitored, agent life-cycle can be controlled. Figure 3 presents the GUIs of some JADE software development tools.

Moreover, given that the success of any software middleware and framework mainly depends on the possibility of easily integrating its components with components provided by other technologies. JADE has been designed to satisfy this requirements and several tools and software libraries allow the integration of JADE with the most known and used technologies (e.g., Web Services [19] and application servers [12]) and for encapsulating "intelligence" into JADE agents (e.g., rule engines [3] and reasoning tools [24]).

4 JADE for the Mobile Environments

As already mentioned, the JADE run-time can be executed on a wide class of devices ranging from servers to cell phones, for the latter the only requirement being Java MIDP1.0 (or higher versions). In order to properly address the memory and processing power limitations of mobile devices and the characteristics of wireless networks (e.g., GPRS and UMTS) in terms of bandwidth, latency, intermittent connectivity and IP addresses variability, and at the same time in order to be efficient when executed on fixed network hosts, JADE can be configured to adapt to the characteristics of the deployment environment. JADE architecture, in facts, is completely modular and, by activating certain modules instead of others, it is possible to meet different requirements in terms of connectivity, memory and processing power.

More in details, a module called LEAP allows optimizing all communication mechanisms when dealing with devices with limited resources and connected through wireless networks. By activating this module, a JADE container is "split", as depicted in figure 4, into a front-end, actually running on the mobile terminal, and a back-end running in the fixed network. A proper architectural element, called mediator, must be already active and is in charge of instantiating and holding the back-ends (that basically are entries in the mediator itself).

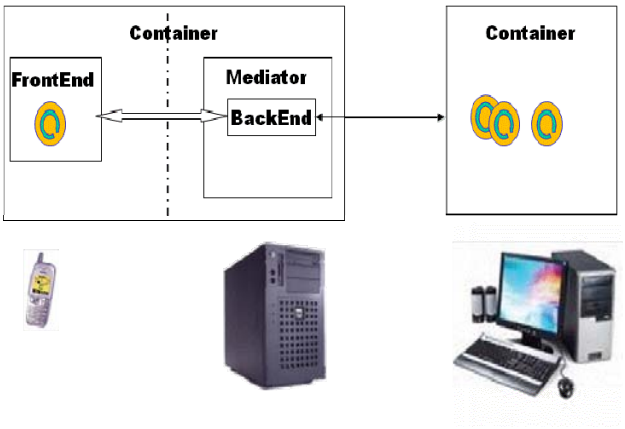

Fig. 4. JADE architectures for mobile environments.

To face work-load problems it is possible to deploy several mediators each one holding several back-ends. Each front-end is linked to its corresponding back-end by means of a permanent bi-directional connection. It is important to note that there is no difference at all for application developers depending on whether an agent is deployed on a normal container or on the front-end of a split container, since both the available functionality and the APIs to access them are exactly the same. The approach has a number of advantages:

- Part of the functionality of a container is delegated to the back-end, thus making the front-end extremely lightweight in terms of required memory and processing power.
- The back-end masks, to other containers, the actual IP address assigned to the wireless device and, among the others, allows hiding to the rest of

the multi-agent system a possible change of IP address.
- The front-end is able to detect a loss of connection with the back-end (for instance due to an out of coverage condition) and re-establish it as soon as possible.
- Both the front-end and the back-end implement a store-and-forward mechanism: messages that cannot be transmitted due to a temporary disconnection are buffered and delivered as soon as the connection is re-established.
- Several information that containers exchange (for instance to retrieve the container where an agent is currently running) are handled only by the back-end. This approach, together with a bit-efficient encoding of communications between the front-end and the backend, allows optimizing the usage of the wireless link.

5 JADE for DTV Environments

To enable agents for DTV technology based on MHP (Multimedia Home Platform) standard, we developed a framework called MHP-VC, realized starting from the idea to integrate the technology of interactive DTV with the concept of "virtual community", which we can define as a technology-supported cyberspace, centered upon communication and interaction of participants, resulting in a relationship being built up. With this type of integration, our aim is to offer to the interactive DTV users, a new range of services (such as multiplayer games, on-line auctions, etc.) which are very common if we think to the idea of virtual community related to the Web. In this way, the potentialities of interactive DVT can enormously grow allowing its users to take advantage of a new number of useful applications and moving the concept of interactivity from the simple interaction user-application to a new type based on the cooperation among a wide number of users.

MHP-VC is developed as a multi-agent system which we can split in two main sides: a server and a client side. The server side is set on Web servers in the fixed network and it is deployed using the standard FIPA specifications, while the client side is the more innovative one because, since it is deployed on set-top boxes, it requires to enable FIPA Agents on these types of devices. Figures 5 and 6 draw the software architectures of the Backend Server and of the Set-Top Box for realizing a JADE multi-agent DTV application.

5.1 Client Side

The agent container set on the client side must be flexible enough to allow the integration of new services for the virtual community users. For this reason, we think that the best choice is to conceive the client-side of our framework as a MHP interactive application running into the set-top box of the user digital television. Figure 5 draws the software architecture for a JADE MHP client.

In the DVB MHP approach, applications are executed in the context of concrete services or events in a service, and, usually, they do not survive after finishing that context. In order to support services for virtual communities, we need to take into account that MHP-VC systems need to store all the users' preferences about a particular topic, e.g. the user profile in a community game. So our approach integrates special agents, named User Agents, which has the basic roles to work as an interface between the user (i.e., the viewers of the DVB MHP jargon) and the rest of the system and to store their dynamic preferences.

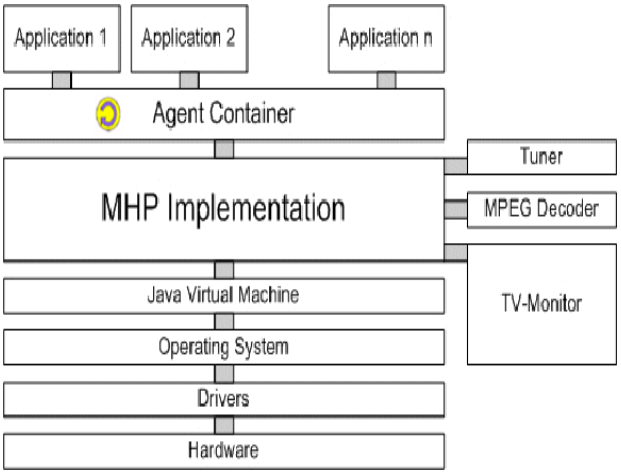

Fig. 5. Client side architecture.

The User Agent is always active and is responsible of building the user profile, maintaining it when its user is on-line, notify to the system when his related user is active, and, if enabled, to act in behalf of its user, The communication UA-user is performed by a standard GUI by which the user can manage his profile and the different services. Clearly, on the other side, the communication between the UA and other agents is based on FIPA specifications.

5.2 Server Side

The server side of our framework consists in an agent container set on a standard Web server (acting as backend server of a MHP application) connected

with the clients through the return channel of the set-top boxes. Figure 6 draws the architecture of the JADE multi-agent system acting as MHP backend server.

In order to support services for virtual communities, the server side of the system has to include at least five different types of agents: a SP Agent (Set-top box Proxy Agent), a MP Agent (Mux Proxy Agent), a User Profile Manager, one or more Service Agent and a Directory Facilitator.

The SP Agent represents the interface between the server side and client-side devices: this agent receives requests from the User Agent running on users' set-top box and manages them interacting with other agents.

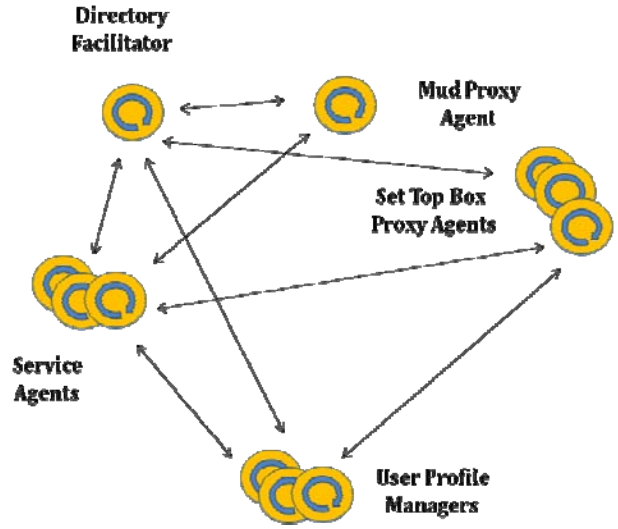

Fig. 6. Server side architecture.

On the other side, we also provide another proxy agent, called MP Agent, which is responsible to update the state of the application and to notify it to the Multiplexer, in order to update the raw data related to the Xlet embedded in the MPEG-2 stream and, consequently, the state of the interactive application displayed on the user's TV screen.

Between these two proxy agents we designed a third specific kind of agent, named Service Agent, which is responsible of particular types of service offered by the framework to the virtual community. For example, if we think to a multi-player game, the Service Manager related to this type of service is responsible to manage the state of the game, to find one or more appropriate partners to play, etc.

The User Profile Manager agent is responsible of maintaining the profile of users and information/preferences of the users in relation to the particular types of services offered by the system (i.e. game preferences, skill level, etc.).

Finally, the Directory Facilitator is responsible to inform an agent about the address of the other agents of the system.

6 Ubiquitous Applications

The idea to play a game in a virtual way with other people connected by a network or, in general, by a technology supporting the real-time interaction between the game participants is very common and diffuse on Internet. With our system we match this idea with the DTV interactive television, allowing the DTV users to play a community game without using any type of computer, but through their DTV device and interacting with other DTV users or with users acting on both traditional computers and mobile devices.

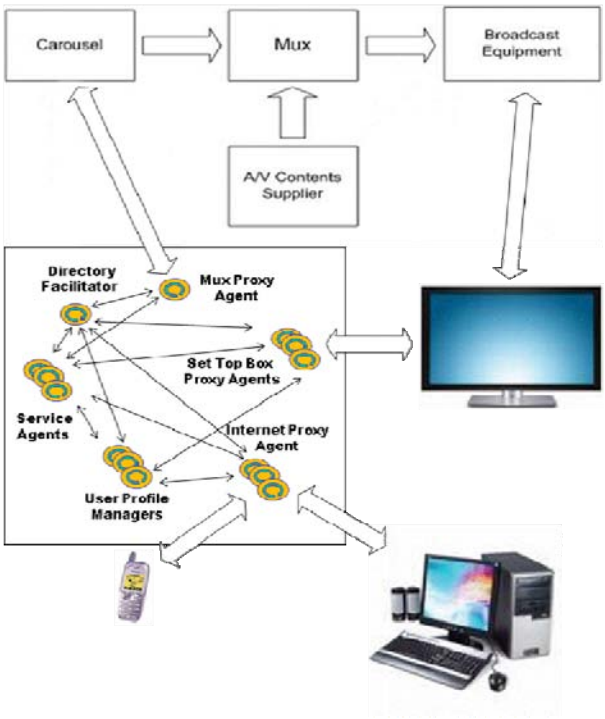

Fig. 7. The architecture of an application connecting computers, mobile devices and digital televisions.

A multi-user game is based on a central Web server on which is set the server-side of the MHP-VC and a set of clients based on computers, mobile devices and DTV. The communication between server and DTV clients is performed by using the return channel of the devices, while the communication between server and computers or mobile devices is a standard IP network communication. To enable the communication of computers and mobile devices, we added another type of agent to the server-side of MHP-VC; this

agent is called IPA (Internet Proxy Agents) and represents the interface between the server and the computer or mobile clients. Figure 7 draws the architecture of a multi-user application involving users connected through computers, mobile devices and digital televisions.

To describe quickly the system behavior, we can consider a simple type of game like “Othello”, which requires two players.

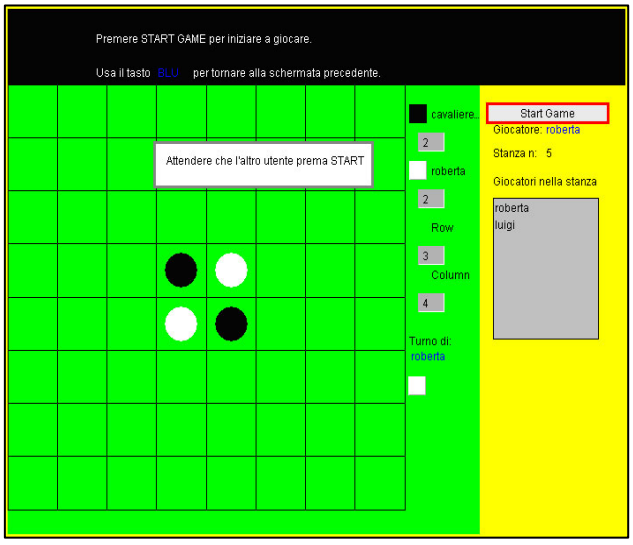

Fig. 8. Othello DTV GUI.

When a user wants to play an Othello match versus another user he has fundamentally to complete two steps before starting the match: the service configuration and the choice of the opponent. The service configuration is a task that the user has to perform only the first time she/he uses the application: the user has to insert some information like the game preferences, the skill level, etc. While both computer and mobile device based users directly configure the service on the server (the mobile user through the front-end back-end pair), a DTV user, once the game has been locally configured, takes advantage of her/his User Agent that communicates the service configuration to the server side of the system to update the user profile managed by the User Profile Manager agent. Now the user is able to play: when she/he run the game by his set-top box, the User Agent notifies the server-side that his associated user wants to play. At this point the Service Agent related to that game creates a new game instance and the User Profile Manager agent find a possible opponent (the other user has to be “on-line” and has to be a compatible skill level), the user chooses an opponent and then the match can start.

The system, e.g., the Service Agent, continuously updates the state of the application in relation to the moves made, one after the other, by the participants until the end of the match. Obviously, in relation to the result, the system updates users’ profiles. Figure 8 shows the Othello digital television GUI.

7 Conclusions

In this paper, we presented how JADE, a software framework to aid the development of multi-agent applications, has been extended to support the realization of multi-user applications whose users can also act on mobile devices and digital televisions.

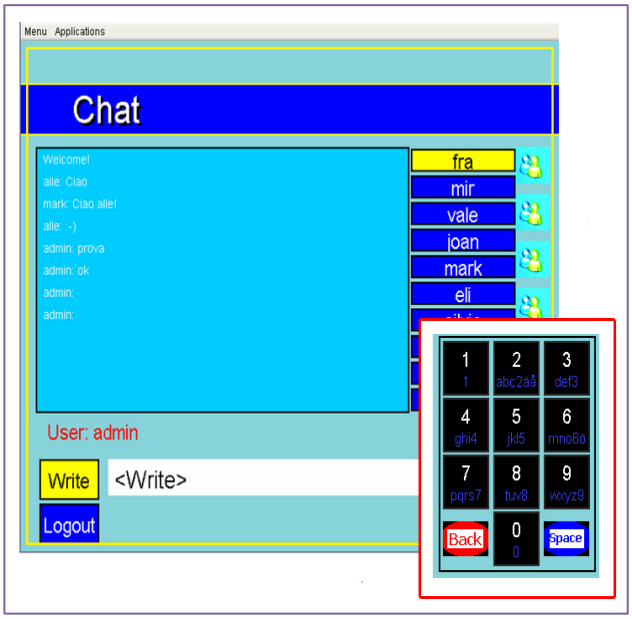

Fig. 9. Chat DTV GUI.

The first experimentation of the software showed that is quite easy to extends applications to the three kinds of clients (i.e., traditional computers, device phones and digital televisions). It is because the ability of the JADE software to distribute agents of the same application on such kinds of device and the flexibility of the Java and Web technologies to adapt the user interface to a specific devices. In particular, we experimented this software for the realization of some multi-user online games and for the realization of some tools supporting the interaction of the users while they are playing together a game. Figure 9 shows the digital television GUI of a chat system that users can use to exchange messages while they are playing a game.

Our future work is related to the experimentation of the software by realizing both more complex multi-user games and other kinds of virtual community services dedicated, for example, to support the collaborative work and e-business applications.

References:

[1] Bellifemine, F. Poggi, A., Rimassa, G. Developing multi agent systems with a FIPA-compliant agent framework. Software - Practice & Experience, 31:103-128, 2001.

[2] Bellifemine, F., Caire, G., Poggi, A., Rimassa, G. JADE: a Software Framework for Developing Multi-Agent Applications. Lessons Learned. Information and Software Technology Journal, 50:10-21, 2008.

[3] Beneventi, A., Poggi, A., Tomaiuolo, M., Turci, P. Integrating Rule and Agent-Based Programming to Realize Complex Systems. WSEAS Trans. on Information Science and Applications, 1(1):422-427, 2004.

[4] Bergenti, F., Poggi, A., Burg, B., Caire, G. Deploying FIPA-Compliant Systems on Handheld Devices. IEEE Internet Computing 5(4):20-25, 2001.

[5] Berger, M., Bouzid, M., Buckland, M., Lee, H., Lhuillier, N., Olpp, D., Picault, J., Shepherdson, J.W. An Approach to Agent-Based Service Composition and Its Application to Mobile Business Processes. IEEE Transaction on Mobile Computing 2(3):197-206, 2003.

[6] Best, B.J., Lebiere, C. Cognitive Agents Interacting in real and Virtual Worlds. In Sun, R. (Ed.), Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation, pp. 186-218. Cambridge, UK, University Press. 2006.

[7] BlueJade Web site, 2003. Available from http://sourceforge.net/projects/bluejade.

[8] Calisti, M., Funk, P., Biellman, S., Bugnon, T., A multi-agent system for organ transplant management. In Moreno, A., Nealon, J. (Eds.), Applications of Software Agent Technology in the HealthCare Domain, Birkhäuser, Basel. pp. 199-212,. Birkhäuser Verlag, 2004.

[9] Cheok, A.D., Fong, S.W., Goh, K.H.,Yang, X, Liu, W., Farzbiz, F. Human Pacman: A Sensing-based Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction. In Proceedings of the Second Workshop on Network and System Support for Games, pp 106-117, Redwood City, 2003.

[10] Conti, V., Vitabile, S., Pilato, G., Sorbello, F. An Enhanced Authentication System for the JADE-S Platform. WSEAS Transactions on Information Science and Application, 1(1):178-183, July 2004.

[11] Covino, G., Caire, G. Innovative NGN Management in Telecom Italia leveraging distributed agent technology, Tele Management Forum, IT & Operations Track. Nice, 2007.

[12] Cowan, D., Griss, M., Burg, B. BlueJADE - a service for managing software agents, HP Technical Report 2001. Available from http://www.hpl.hp.com/techreports/2001/HPL-2001-296R1.pdf.

[13] Chmiel, K., Gawinecki, M., Kaczmarek, P., Szymczak, M., Paprzycki, M. Efficiency of JADE agent platform. Scientific Programming, 2:159-172, 2005.

[14] DAVIC Web Site, 2008. Available from http://www.davic.org.

[15] DVB Web Site, 2008. Available from http://www.dvb.org.

[16] Fielding, D., Fraser, M., Logan, B., and Benford, S. Extending game participation with embodied reporting agents. In Proceedings of the 2004 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology, pp. 100-108, Singapore, 2005.

[17] Fielding, D., Logan, B., Benford, S., Balancing the needs of players and spectators in agent-based commentary systems, In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 996-998, Hakodate, Japan, 2006.

[18] FIPA Specifications, 2000. Available from http://www.fipa.org.

[19] Greenwood, D., Nagy, J., Calisti, M. Semantic Enhancement of a Web Service Integration Gateway, In Proceedings of the AAMAS 2005 workshop on Service Oriented Computing and Agent Based Engineering (SOCABE), Utrecht, Netherlands. 2005.

[20] HAVi Web Site, 2008. Avaiulable from http://www.havi.org.

[21] Hodík, J., Bečvář, P., Pchouček, M., Vokřínek, J. and Pospíšil, J. ExPlanTech and ExtraPlanT: multi-agent technology for production planning, simulation and extra-enterprise collaboration. International Journal of Computer Systems Science and Engineering. 20(5):357-367, 2005.

[22] Huang, Z., Eliëns, A., and Visser, C. 3D agent-based virtual communities. In Proceedings of the Seventh international Conference on 3D Web Technology, pp. 137-143. Tempe, AZ, 2002.

[23] JADE Web Site, 2008, Available from http://jade.tilab.com/.

[24] Jadex Web Site. 2008, Availablr from http://sourceforge.net/projects/jadex.

[25] Java TV API Web Site, 2008. Available from http://java.sun.com/products/javatv.

[26] Lee, H., Mihailescu, P., Shepherdson, J. Realising team-working in the field: an agent-

based approach, IEEE Pervasive Computing, 6(2):85-92, 2007.
[27] Lhuillier, N., Tomaiuolo, M., Vitaglione, G.Security in Multi-Agent Systems: JADE-S goes Distributed, Exp in Search of Innovation 3(3):42-51, 2003.
[28] MHP Web Site, 2008. Available from http://www.mhp.org.
[29] Nolan, S. iDTV Gamers: The Emergence of a New Community? In Proceedings of Computer Games and Digital Cultures Conference. Tampere, Finland, 2002. Available from http://www.digra.org:8080/Plone/dl/db/05164.17549.pdf.
[30] Shepherdson, J., Lee, H., Mihailescu, P. J. Evaluation of Multi-agent Platforms as an Enabler of Enterprise Mobilisation, WSEAS Transaction Journal on Systems, 11(4):1899 - 1905, 2005.
[31] Smed, J., Kaukoranta, T., Hakonen, H. A Review on Networking and Multiplayer Computer Games. Technical Report 454, TUCS, 2002. Available from http://www.tucs.fi/publications/techreports/TR454.pdf.
[32] Vrba, P., Cortese, E., Quarta, F., Vitaglione, G. Scalability and performance of the JADE message transport system: analysis of suitability for Holonic manufacturing systems. Exp in Search of Innovation 3(3):52-65, 2003.
[33] Zimmermann, R., Winkler, S., Bodendorf, F. Supply Chain Event Management with Software Agents. In Kirn, S., Herzog, O., Lockemann, P., Spaniol, O. (Eds.), Multiagent Engineering - Theory and Applications in Enterprises, pp. 157-175, Springer, Berlin-Heidelberg 2006.