Embedding the Remote Application Control: Visual API for PDA Programming

Salvatore Sorce, Paolo Raccuglia, Alessandro Genco
Department of Computer Science and Engineering (DINFO) - University of Palermo
Viale delle Scienze, edificio 6 - 90128 Palermo (Italy)
{sorce|genco}@unipa.it, p.raccuglia@tin.it

Abstract

PDAs are more and more used as advanced adaptive HEI (human-environment interaction) interfaces, thus enabling their users to easily operate applications remotely running in pervasive computing scenarios.

Based on a previous implementation, in this paper we discuss the development of a new set of .NET-compliant Application Programming Interfaces to be used within the Visual Studio IDE. Our main goal is to provide programmers with a set of components to be used with the common drag-and-drop operation, in order to embed a remote application control within an application running on the PDA and vice-versa.

The interaction relies on ad-hoc communication protocols over a framework devoted to pair PDAs and remote devices according to their relative position, with no need to use any connection software.

1. Introduction

Pervasive systems are more and more used in real-life environments in order to provide people with some kind of service. They are composed of a large variety of networked smart devices, thus enriching the environment they are deployed in. Devices devoted to the processing logic should be mostly hidden to avoid the perception of their presence and to prevent the environment flooding (*pervasive* systems have not to be *invasive*). On the other hand, devices of a pervasive system that are devoted to implement the user interface cannot be hidden because of their own purpose.

Personal mobile devices (such as PDAs and SmartPhones) are successfully exploited for human-environment interaction purposes within pervasive systems. In fact, it has to be considered that such interaction should not be the same for all, since differences in needs and skills of people have to be taken into account to avoid heavy compromises which could not satisfy anyone. Due to their programmability and wide popularity, they can be made suitable to operate as remote controllers, or personal adaptive I/O interfaces, for applications remotely running. This way needed services can be accessed by means of a well-known device, with no need to learn how to use new kind of interface.

There are several existing pervasive frameworks for service provision within public accessible areas exploiting personal mobile devices as human-environment interface with the goal to improve the users' experience[1]. This common interest is justified by the wide diffusion of such devices, that are almost in everyone's pocket and that can be used almost any when and anywhere.

There is a large variety of application fields where services provided by pervasive systems can be accessed by mobile devices, such as interactive guides in cultural heritage sites based on the user profile [2], augmented reality objects assembly in mobility[3], context-aware information provision within university campuses[4], healthcare systems[5] [6].

One of the problems research groups are still dealing with, is the composition of pervasive systems and, in particular, the design of tools for user-friendly interface definition[7]. In this field, a common basis to build a suitable graphical user interface on a personal mobile device for remote applications control, is to exploit the remote desktop feature.

In 2004, de Paiva Guimarães et al.[8] proposed a tool to generate graphical interfaces for PDAs in a straightforward manner. The tool relies on an existing library for distributed computing, allowing PDAs to interact with immersive multiprojection environments. The generated interface is light and simple, since it is composed of an image on which a set of hot spots are built. The interaction takes place by handling events on the hot spots.

Authors of [9] describe a Java-Swing application

978-0-7695-3575-3/09 $25.00 © 2009 IEEE
DOI 10.1109/CISIS.2009.174

running on personal devices to launch, communicate with, and control a shared set of interactive applications running on different networked machines. Applications to be controlled must agree to follow the proposed conventions, since application-specific program control protocols have to be encapsulated in an XML document. This XML document is downloaded and parsed by the remote application controller on the PDA.

The Redar system[10] is a remote desktop architecture for distributed virtual personal computing. It integrates various application interfaces from different service nodes into one virtual desktop environment, and presents them on a virtual desktop to the thin-client. The core of the proposed framework is a framebuffer-based GUI merger. It composes the remote desktop which is presented on a ultra-thin client running on a PC equipped with just as a set of network attached I/O devices.

One of the most recent works is presented in [11], where the server-based application to be remotely controlled is first analyzed and classified using image processing. This generates an interface description that is used on the client side to build the graphics element corresponding to the application interface on the server. This solution lowers the amount of data to be exchanged over a wireless connection and allows users to view only the useful GUI elements of a given application on a PDA. In turn it adds a significant overhead for the automatic recognition of components within a GUI, and does not handle situations where the interaction with an element of the GUI produces dynamic changes on the GUI itself.

Most of discussed works propose solutions to optimize the bandwidth usage or to adapt the size of a remote desktop to a PDA display, by means of an ad-hoc hardware/software framework. This means that some overhead is added, applications to be controlled have to be appropriately designed to fit the solution purposes, controller interfaces are specific for each application.

In this paper we present a framework aimed at the easy building of applications to be remotely operated by PDAs. Based on a previous implementation[12], the IRIN (Intuitive Remote INterfaces) project allows programmers also to visually design interfaces for PDAs that are easy to be used and effective to control remote applications.

2. IRIN Overall Architecture

The design of an infrastructure for remote applications control involves different elements that interact one with each other, and in particular (fig. 1):

- the controlled element;
- the remote control device;
- the user / human controller.

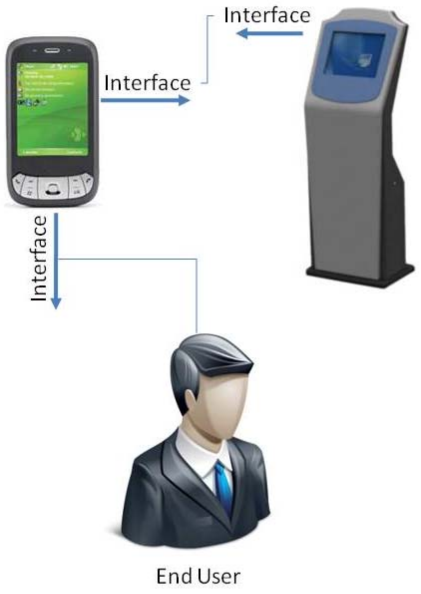


Figure 1. Actors of a remote control infrastructure

Besides the effectiveness of interaction, the interface between the remote control device and the controlled element should not be perceived by the end user. In more detail, designers of such infrastructure should pursue the following goals:

- the controlled element (e.g. a remote display) must provide an interface for the remote control, such as a given protocol;
- the control device must correctly use such interface, adapting controls and commands to its own capabilities;
- the control device must provide human users with a friendly interface, requesting them minimum technological skills or, better, not at all.

As a consequence, the design of a framework with such goals involves different devices, different technologies, and different tools.

Due to the features and capabilities of personal mobile devices, the integration of such factors is not so easy, and some proposed solution lacks in flexibility and usefulness of user interfaces[13].

The main goal of the IRIN project is to provide designers with an easy-to-use tool to implement applications that are remotely controllable by PDAs. In other words, IRIN is an integrated framework to easily implement controllable applications and controller interfaces with the support of visual development tools. The "intuitiveness" feature of IRIN refers both to the easiness in building interfaces and to the friendliness of built interfaces.

To this end, IRIN is composed of three logical elements (fig. 2):

- *the IRIN server*, which role is to control the IRIN framework;
- *the IRIN Mobile Platform*, which allows programmers to implement and deploy applications for PDAs;
- *the IRIN Remote Control Protocol (IRCP)*, by which PDAs and remote devices communicate with each other.

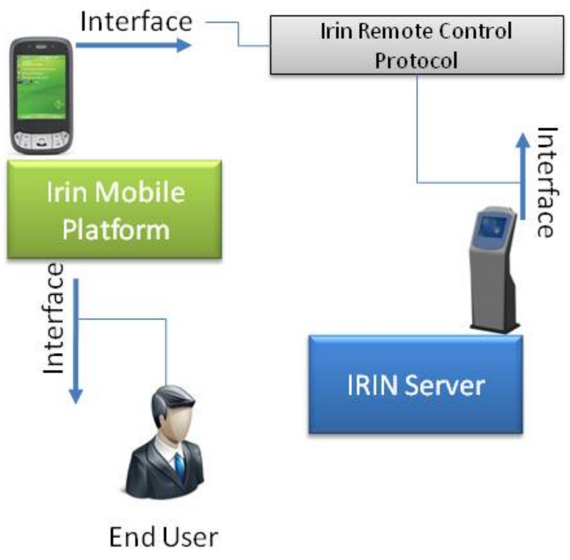


Figure 2. The IRIN framework

3. The IRIN Server

The IRIN server improves any device by making it remotely controllable. It is not strictly bound to a given architecture, since its core is written in Java and so it can be ported in every compatible device. Furthermore, the server can be easily adapted to almost every computer system with no need to rewrite its core. This goal is achieved by means of a set of drivers that can be created and updated by means of a simple API that will be described in the following.

The IRIN server exploits the simplicity of the IRCP protocol, which has been developed as more light as possible, in order to not waste too much resources.

3.1. The IRIN server architecture

The IRIN server is composed of (fig. 3):

- a *core*;
- a library of *standard drivers*. In the current implementation, they allow users to interact with a PC running the Windows OS as a remote display by means of a PDA;
- a set of *visual APIs*, which allow designer to develop ad-hoc drivers and controls for the remote control of different devices, such as robots, networked computers, multimedia devices, etc.

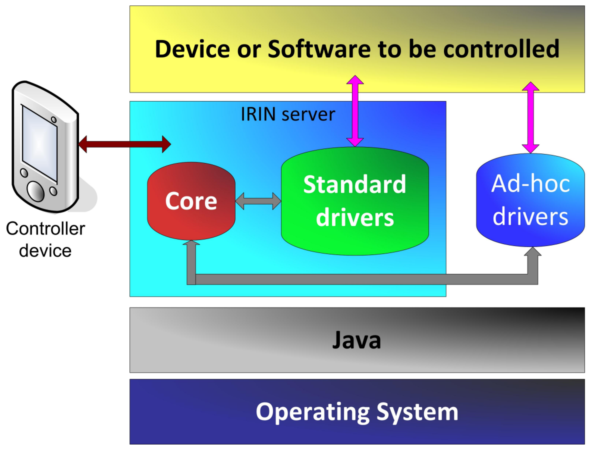


Figure 3. The IRIN server architecture

The IRIN server's core manages IRCP transactions, loads the current configuration which is coded in an XML file, and loads the selected controls.

The implemented control library allows developers to set up remote displays control, with dedicated interfaces for multimedia stream control, remote mouse control by drag&drop and buttons, remote content management, PDA screen capture forwarding.

Besides the above described features, the control library can be used to write new ad-hoc controls by means of a set of re-usable APIs based on the Java Native Interface.

The use of native interfaces allowed us to keep separate the server core and the control system, thus obtaining a solution that is interoperable, simple, intuitive, portable. Therefore the remote display control is only one of the features of IRIN, that claims to be a set of control interfaces.

4. The IRIN Mobile Platform

The IRIN Mobile Platform (IMP) is a framework for the rapid development of applications with graphical user interface to be deployed on a PDA for the remote control of computer devices by means of the IRCP protocol. The IMP has been developed by using the .NET Compact Framework 3.5, so it is compatible with any device running Windows Mobile 2003 or above. The API makes use of all standards of the .NET platform, so applications can be developed exploiting the visual IDE of Microsoft Visual Studio.

We decided to use the .NET Compact Framework in order to validate our design choices using a large variety of mobile devices with different input interfaces (touch screen, qwerty mini-keyboards, touch pen, even used in combination).

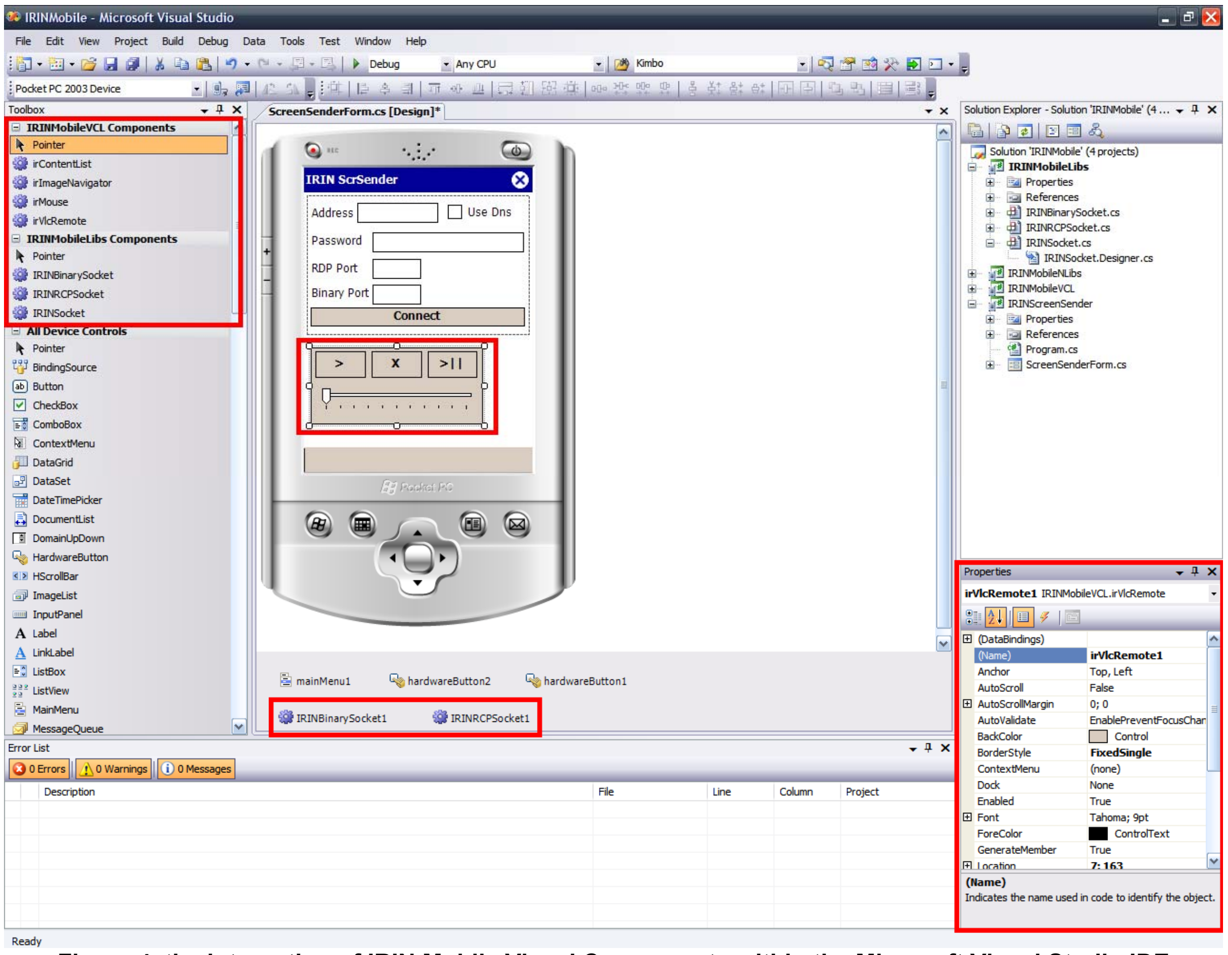
4.1. IMP implementation details

The IMP is composed of two libraries: the *IRIN Mobile Libs*, and the *IRIN Mobile Visual Components*.

The IRIN Mobile Libs are the foundation for the development of any mobile application. They represent the mobile counterpart of the IRIN server. In particular they provide simplified socket-based interfaces for communications between the PDA and the IRIN server. The IRIN Mobile Libs are composed of three sockets:

- the *irSocket* that is a virtual class. It provides basic functions to connect the PDA with the remote device;
- the *irIRCPSocket* that provides all needed methods to manage the IRCP Protocol;
- the *irBinarySocket* that provides the irIRCPSocket with additional functions for binary data exchange.

The IRIN Mobile Visual Components is a set of libraries for the intuitive development of user-friendly GUIs on PDA. Programmers are allowed to concentrate on the GUI development, since the IRIN Mobile Visual Components are integrated with the sockets and with the Visual Studio IDE as above mentioned (fig. 4).



irVlcRemote1 IRINMobileVCL.irVlcRemote

(DataBindings)	
(Name)	irVlcRemote1
Anchor	Top, Left
AutoScroll	False
AutoScrollMargin	0; 0
AutoValidate	EnablePreventFocusChan
BackColor	Control
BorderStyle	FixedSingle
ContextMenu	(none)
Dock	None
Enabled	True
Font	Tahoma; 9pt
ForeColor	ControlText
GenerateMember	True
Location	7; 163

(Name)
Indicates the name used in code to identify the object.

Figure 4. the integration of IRIN Mobile Visual Components within the Microsoft Visual Studio IDE

The current implementation of visual controls allows a PDA to control remote displays according to the standard library set that is included within the IRIN server. In more detail, the available visual controls are:

- *irContentList* for handy visualization of the list of contents available on the remote display side;

- *irImageNavigator* for navigation of hi-res images in small-sized areas by dragging;
- *irMouse* for the mouse pointer remote control;
- *irVlcRemote* for the remote control of the VLC media player by means of the standard set of buttons.

We implemented these four controls for our tests because they allow users to control a wide range of applications, considering that the implementation of new controls is very straightforward. Actually, the *irMouse* control could be enough to control almost any kind of application. We decided to add some more visual control with ad-hoc widgets (such as the "play" button for the media player) in order to provide users with a more intuitive interface.

Besides the re-usability, the strength of these APIs is that they provide a full support to the multithreading, they uses non-blocking sockets, and they support event-based programming for a fast development and deployment of applications. Fig. 5 shows an example of source code for event-based interface programming.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using IRINMobileLibs;
using IRINMobileVCL;

namespace IRINConsoleSample
{
  class Program
  {
    public IRINRCPSocket remoteController;

    static void Main(string[] args)
    {
      Program thisProgram = new Program();
      thisProgram.remoteController = new _
        IRINRCPSocket();
      thisProgram.remoteController.initialize _
        ("password", "192.168.2.1", 1984, true);
  //ADDING EVENTS:
      thisProgram.remoteController.rcpSocketReady _
        += new rcpSocketReadyHandle(thisProgram. _
        remoteController_rcpSocketReady);
      thisProgram.remoteController. _
        connectAndGetInfo();
      Console.Out.WriteLine("Connecting...");
    }

    private void remoteController_rcpSocketReady _
     (object Sender)
    {
      Console.Out.WriteLine("Connection Done. _
        Ready...");
    }
  }
}
```

Figure 5. Source code for event-based interface programming

All IRIN components are designed to be easily modified in order to upgrade the IRIN framework. This will allow IRIN users to easily develop specific interfaces to control a wider range of devices and applications. Fig. 6 gives an overview of the IRIN project components.

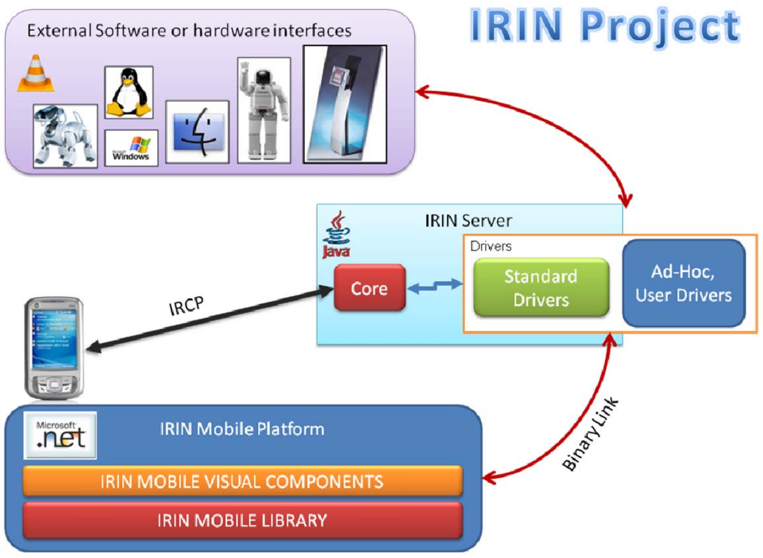


Figure 6. A complete view of the IRIN project

5. Implementation Details

The current available and tested version of the IRIN framework provides programmers with all functions and components for the fast deployment of a remote display network to be controlled by PDAs.

The remote display control feature relies on the IRCP protocol, that is an evolution of a previous trial implementation[12]. The IRCP protocol has been set up aiming at the tuning of control interfaces for PDAs based on their computing and visualization capabilities. It is based on a set of input text-strings. The protocol does not need great computing resources for the parsing of input strings, and it can be extended with new commands for specific interfaces.

For example, the "GETCOMMANDS" and "GETCONTENTS" strings allows the client to request all needed information to control the remote display from the IRIN Server, thus avoiding complex protocol-based interaction. If the controlled application uses the IRIN Mobile Platform, complex control functions are already available, such as the mouse control by drag&drop operation, or the visualization of remote display on the PDA and vice-versa. The IRIN Mobile Platform does not force the developers to implement the IRCP protocol within their mobile applications.

The *IRIN Easy Client* is an example of a mobile application that exploit the first implementation of the

IRIN Mobile Platform. This allowed us to integrate all interfaces to use a remote display within one mobile application. At the same time, due to the re-usability of available components, the implementation work has been extremely agile.

The *IRIN Screen Sender* is an example of integration with PDA's native APIs. This application exploits the IRIN Mobile Native Libs that bind the native libraries of the Windows Mobile OS with the IRIN Mobile Platform. These libraries are currently under further development, and they only implement the PDA screen capture feature. The captured screen can be sent to the remote display for visualization. The transaction between the PDA and the server involves a direct communications among controls, and does not involve the IRCP text-based protocol. In this example the transaction relies on a binary link (data are exchanged in a binary form).

6. Conclusion

In this paper we presented the IRIN project, an enabling framework for the rapid development of applications to be remotely controlled, and of control interfaces to be deployed on PDAs. The remote control takes place without any external connection software, such as the ActiveSync one.

The IRIN project is an evolution of our previous works[12], where an indoor positioning system was used to locate a remote display to be controlled by a PDA in the neighborhood. The relative position was managed by a central server.

We are carrying out experiment about the implementation of an intelligent pairing system. In particular, we are aiming at the integration of the positioning system within the IRIN server, by means of the triangulation method used over the WIFI link. Experiments carried out gave us good results in integration, but they are not so satisfying concerning the position detection accuracy and precision.

As a further enhancement, we plan to redesign the protocol in order to manage a IRIN-based framework as a unique distributed server.

7. References

[1] Genco A., S. Sorce, G. Reina, G. Santoro, "An Agent-Based Service Network for Personal Mobile Devices", *IEEE Pervasive Computing*, vol. 5, no. 2, pp. 54-61, Apr-Jun, 2006

[2] Raptis D., Tselios N., Avouris N., "Context-based design of mobile applications for museums: a survey of existing practices", Proc. of the 7th ACM International Conference on Human-Computer Interaction with Mobile Devices & Services, Salzburg, Austria 2005, pp: 153-160

[3] Henrysson A., M. Ollila, M. Billinghurst, "Mobile phone based AR scene assembly", Proc. of the 4th Int'l Conference on Mobile and Ubiquitous Multimedia, Christchurch, New Zealand, 2005, pp. 95-102

[4] Genco A., S. Sorce, G. Reina, G. Santoro, R. Messineo, R. Raccuglia, L. Lo vecchio, G. Di Stefano, "An Augmented Campus Design for Context-aware Service Provision", proc. of the 33rd annual ACM SIGUCCS Conference on User Services, Monterey, CA, Nov. 6-11 2005, pp. 92-97

[5] Price, S.; Summers, R., "Mobile Healthcare in the Home Environment", Proc. of 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society EMBS '06, New York, NY, Aug. 2006, pp. 6446-6448

[6] Ardizzone E., O. Gambino, A. Genco, R. Pirrone, S. Sorce, "Pervasive Access to MRI Bias Artifact Suppression Service on a Grid", *IEEE Transaction on Information Technology in Biomedicine*, DOI: 10.1109/TITB.2008.2007108.

[7] Oh, M., Lee, J., Chang, B., Ahn, J., and Doh, K., "A programming environment for ubiquitous computing environment", SIGPLAN Not. 42, 4 (Apr. 2007), pp. 14-22

[8] de Paiva Guimarães, M., Gnecco, B. B., and Zuffo, M. K., "Graphical interaction devices for distributed virtual reality systems", proc. of the 2004 ACM SIGGRAPH international Conference on Virtual Reality Continuum and Its Applications in industry (Singapore, June 16 - 18, 2004). VRCAI '04. ACM, New York, NY, 363-367

[9] Miller, J. R., Yengulalp, S., and Sterner, P. L., "A framework for collaborative control of applications", proc. of the 2005 ACM Symposium on Applied Computing (Santa Fe, New Mexico, March 13 - 17, 2005). L. M. Liebrock, Ed. SAC '05. ACM, New York, NY, pp. 1244-1249

[10] Yuedong Zhang; Zhenhua Song; Dingju Zhu; Zhuan Chen; Yuzhong Sun, "Redar: A Remote Desktop Architecture for the Distributed Virtual Personal Computing", proc. of *Fifth International Conference Grid and Cooperative Computing*, GCC 2006, pp.1-8, Oct. 2006

[11] Lamberti, F.; Sanna, A., "Extensible GUIs for Remote Application Control on Mobile Devices," Computer Graphics and Applications, IEEE, vol.28, no.4, pp.50-57, July-Aug. 2008

[12] Genco A., S. Sorce, R. Messineo, P. Raccuglia, "PDA – Remote Display Interaction Framework", proc. of First International Workshop on Intelligent Interfaces for Human-Computer Interaction (IIHCI-2008), march 4-7, 2008, Barcelona, Spain, pp. 763-768

[13] Maekawa, T.; Uemukai, T.; Hara, T.; Nishio, S., "A Java-based information browsing system in a remote display environment," *Proceedings IEEE International Conference on e-Commerce Technology, CEC 2004,* pp. 342-346, 6-9 July 2004