



OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

Title of Document: Final prototype applications

Editor(s): Giancarlo Cherchi, Francesca Mureddu

Affiliation(s): Arcadia Design

Contributor(s): All Open Partners

Affiliation(s): All Open Partners

Date of Document: March 2010

OPEN Document: D5.4

Distribution: EU

Keyword List: Prototypes, Applications

Version: 1.0

OPEN Partners:

CNR-ISTI (Italy)
Aalborg University (Denmark)
Arcadia Design (Italy)
NEC (Germany)
SAP AG (Germany)
Vodafone Omnitel NV (Italy)
Clausthal University (Germany)

"The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2010 by Arcadia Design, Aalborg University, Clausthal University, CNR, NEC, SAP, Vodafone."



OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

ABSTRACT

This document contains the description of the software prototypes that have been delivered for D5.4, in terms of hardware and software requirements, setup and run, and usage instructions needed for their correct operation. In particular, four prototypes covering the two selected domains (business and game) have been developed and integrated with the Migration Service Platform, namely: Emergency, Social Game, Twitter Wall and Pacman.



OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

TABLE OF CONTENTS

1. INTRODUCTION	4
2. EMERGENCY PROTOTYPE.....	5
2.1. DESCRIPTION	5
2.2. HARDWARE AND SOFTWARE REQUIREMENTS	6
2.3. SETUP AND RUN.....	6
2.4. USAGE	6
3. SOCIAL GAME.....	13
3.1. DESCRIPTION	13
3.2. HARDWARE AND SOFTWARE REQUIREMENTS	14
3.3. SETUP AND RUN.....	15
3.4. USAGE	17
4. TWITTER WALL	24
4.1. DESCRIPTION	24
4.2. HARDWARE AND SOFTWARE REQUIREMENTS	25
4.3. SETUP AND RUN.....	25
4.4. USAGE	25
5. PAC-MAN	27
5.1. DESCRIPTION	27
5.2. HARDWARE AND SOFTWARE REQUIREMENTS	28
5.2.1. <i>PacMan UI (mobile/desktop) / OPEN Client.....</i>	<i>28</i>
5.2.2. <i>Application logic / OPEN Client / Open Server</i>	<i>28</i>
5.3. SETUP AND RUN.....	28
5.3.1. <i>PacMan UI (mobile/desktop) / OPEN Client.....</i>	<i>28</i>
5.3.2. <i>Application logic / OPEN Client / Open Server</i>	<i>29</i>
5.3.3. <i>Usage</i>	<i>30</i>
6. CONCLUSIONS	32
7. REFERENCES	33

1. INTRODUCTION

The aim of this document is to provide instructions on how to install, run, and use the application prototypes that have been developed in WP5 and integrated with the Migration Service Platform. A detailed description of these prototypes in terms of functionality, architecture, and integration has already been provided in D5.3 [5]. This document describes, for each prototype, its hardware and software requirements, the instructions to install and run it, and some examples of usage that involve migration.

The document is organized into five sections: section 1 is a short general introduction; sections from 2 to 5, after a brief description for each prototype, illustrate the hardware and software requirements, the installation instructions and the usage.

2. EMERGENCY PROTOTYPE

This section presents the final application of the Emergency prototype. It discusses the hardware and software requirements for running it. Further, the section also gives a short overview of the setup and run procedures of the prototype, as well as an explanation of how to interact with it. The usage explanation is based on a single use case, but it still covers all the functions the prototype provides.

2.1. DESCRIPTION

The emergency scenario, as described in D 5.1, foresees tight cooperation between governmental agencies, organizations and companies, so that they can provide public security in emergency situations. In such cases it is vitally important to have all the information available gathered together, so that adequate response plans can be made. Examples of such type of information are simulating and forecasting of flood consequences, water level, traffic data, etc.

The Emergency prototype makes it possible to gather and integrate all that information on one screen. Thus it enables experts to better analyze and plan response activities in a flood situation. It gives them more flexibility for the visual representation of their data on a map because it supports the migration of different application components to one target device. By migrating the necessary components or even whole applications to one target, experts have all the needed information overlaid on one map at their disposal.

In order to support that functionality for emergency situations, the prototype uses the Migration Service Platform (MSP). The MSP is developed within OPEN and it provides the needed infrastructure to accomplish the described migration functionality, so the migration process is realized through this platform. The Emergency prototype uses the MSP to register devices on it (target and source devices), to gather context information about the target device and to accomplish the migration.

As already mentioned, the MSP provides the needed infrastructure for migrating from one device to another. The Emergency prototype exploits this infrastructure by enhancing its own functionality to better serve the expert needs in emergency situations. The functionality enhancements are described in detail in D5.3 [5]; here just a short overview is given. The main enhancement is the possibility to migrate only chosen components or the whole application and optionally to synchronize the source and target devices by selecting different migration modes. Further, each component can be easily chosen by a click on it; the correspondent component is then also highlighted so that it becomes clear that e.g. some map elements depend on the traffic control. Another enhancement of the Emergency prototype is the visualization of migrated components on the target device. A second, smaller map appears in the right upper corner of the application to show an overview of the simulation regions that have been migrated and merged on the target device.

The migration control panel that provides all migration relevant settings is not part of the application itself but should be seen as an integrated tool. The user can move it around the application UI so that it does not cover some important information shown on the screen.

How these enhancements can be used will be described in more detail in section 2.4.

2.2. HARDWARE AND SOFTWARE REQUIREMENTS

The business application requires a web browser supporting XHTML, JavaScript, AJAX, and Silverlight, and an Internet connection. The Internet connection is needed to communicate with the OPEN platform, to start the application (the prototype is implemented as a web application) as well as to access maps provided by the ESRI server.

Silverlight supports most platforms and browsers: Internet Explorer and Mozilla Firefox are supported on the Windows platform. Mozilla Firefox and Apple Safari are supported on the Apple platform.

The hardware requirements correspond to the requirements for using Silverlight [2]. For example, a personal computer running Windows with a x86 or x64 500-megahertz (MHz) CPU or higher processor with 128-megabytes (MB) of RAM is sufficient. For Mac OS a PowerPC G4 800-MHz or higher processor with 128-MB of RAM or Intel Core Duo 1.83-gigahertz (GHz) or higher processor with 128-MB of RAM is sufficient.

2.3. SETUP AND RUN

No application setup is required for the Emergency prototype. It runs in a browser and requires that the Silverlight Plugin has been downloaded and installed. The Silverlight Plugin can be downloaded free of charge from the Silverlight Developer Center [1]. However, if Silverlight has not been installed, it can be downloaded and installed by means of a One Click Install from Microsoft the first time the Silverlight Emergency prototype application is called.

No special setup of the Migration Service Platform is necessary. The IP address of the platform should be entered in the Migration Control of the application so that it can register at the platform.

2.4. USAGE

The final Emergency prototype demonstrates three different migration alternatives and it also suggests how to combine multiple application components on a target device. In order to explain its usage in a clear and easy way, a use case has been picked up along which the functionality of the prototype will be presented.

Dr. Smith is working at the Emergency Operating Center (EOC) as a flooding expert. He is in charge of creating flood simulations when a river in his region runs over. The information from these flood simulations is quite important for a colleague of his, Mr. Thomson. Mr. Thomson is an evacuation expert who observes the traffic situation in areas near the flood. The flood simulation of Dr. Smith, together with the traffic situation information that Mr. Thomson provides, is essential input for Mr. Brown. Mr. Brown is the managing director of the EOC and he has to plan the emergency response activities based on the insights gained from both simulations.

Yesterday it was announced that there is a flood danger in the region of Portland. Mr. Brown stirs his team into action. Dr. Smith and Mr. Thomson start doing their simulations.

In emergency situations Mr. Brown and his team usually meet at the EOC to discuss possible response activities after the simulation data is available. In the EOC meeting room there is a smart wall, which facilitates the presentation of different data and the interaction with it. At the beginning of the meeting Mr. Brown asks his colleague, Mr. Thomson, to present the traffic simulation he has been working on. So Mr. Thomson turns on his laptop and starts the emergency application from his browser. The first thing he does is to connect to the OPEN platform. To do that, he clicks on the migration control panel and goes to the “Connect” button and a settings menu shows up. The OPEN platform address is already specified by the application. He only has to choose whether his laptop should be registered at the server as a source device or as both, source and target. He opts for the first one and connects to the server by pressing the button “Connect”. Figure 1 shows Mr. Thomson’s screen just before connecting to the OPEN platform. The screenshot shows also a selected checkbox just next to the traffic control panel. The checkbox indicates which component is currently on focus (in that case there is only one component, so the checkbox is always selected).

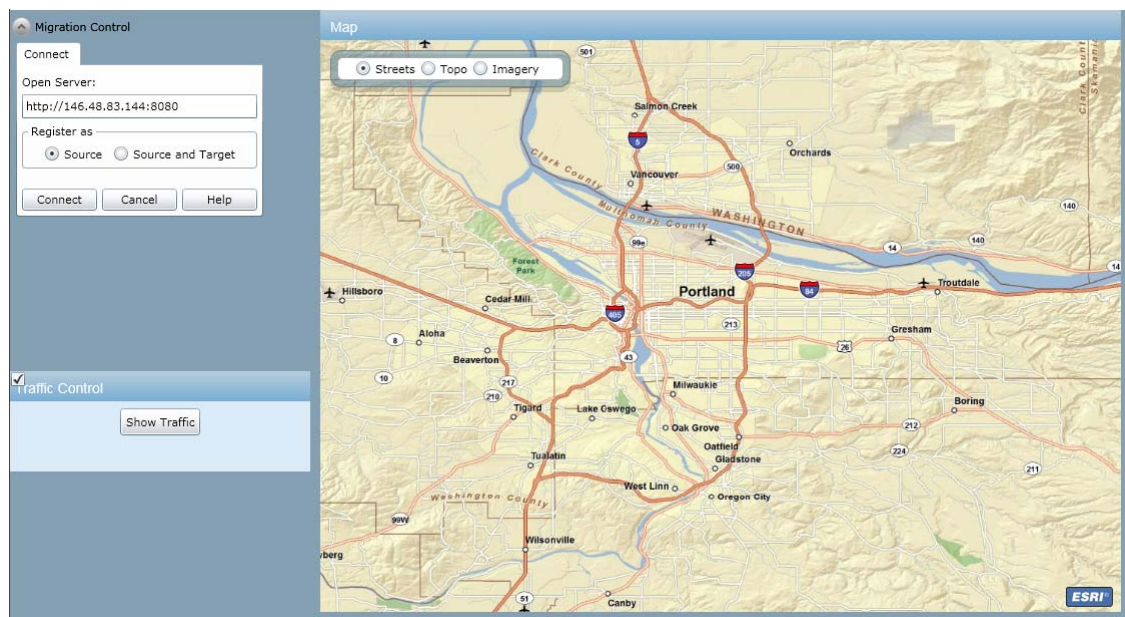


Figure 1: connecting to the OPEN platform

In the meantime Mr. Brown registers the smart wall to the OPEN platform. He does the registration within the browser, by accessing the OPEN platform site. Then he presses the “Connect” button in order to connect to the platform. Figure 2 shows the screen where the connection to the OPEN platform is being processed.

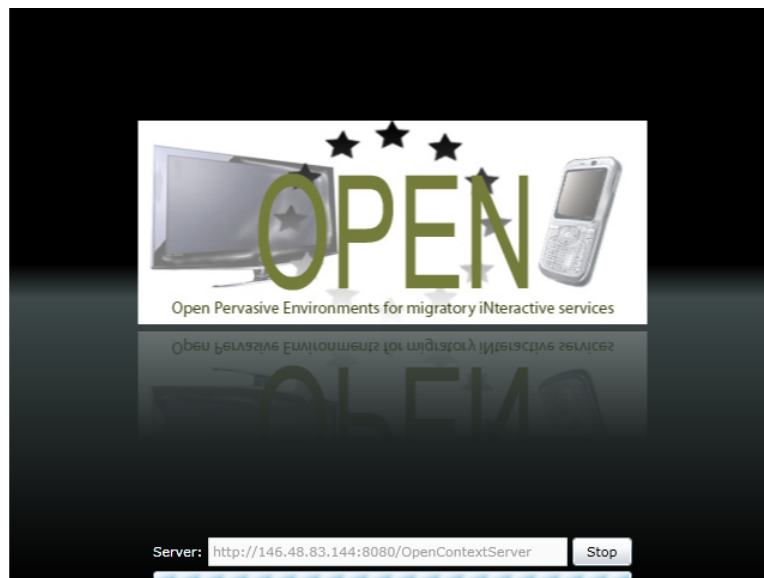


Figure 2: registration to the OPEN platform

After the connection between the smart wall and the OPEN platform has been established, Mr. Thomson can start the migration of his simulation. In his application he sees a second tab just next to the “Connect” tab. This second tab, “Migrate”, provides all necessary settings for a migration.

Mr. Thomson first loads the current list with possible target devices from the server by clicking the button “Load”. As target he chooses the smart wall. Then he can select between three kinds of actions, to perform total migration, to migrate single components or to keep the current and migrated applications synchronized. As shown in Figure 3 the choices Mr. Thomson has made can be tracked.

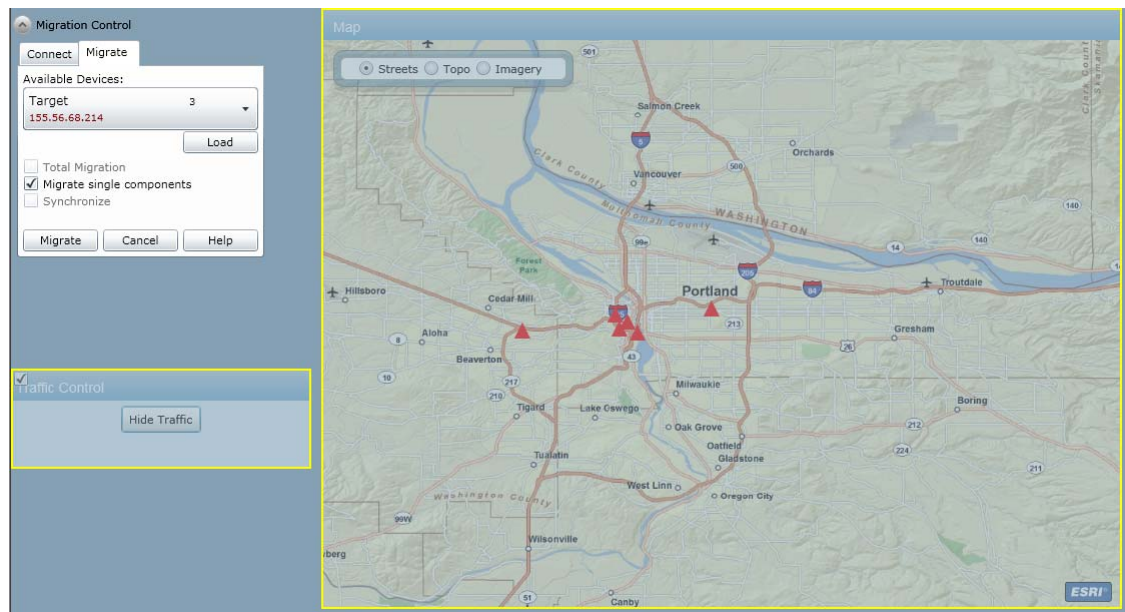


Figure 3: component selection for migration

Mr. Thomson prefers to migrate single components, more specifically, the traffic control together with the corresponding map element. The yellow frame surrounding both components makes clear the dependency between the traffic control and the map element. Mr. Thomson presses the button “Migrate” and the migration process starts. While the process is still running, the screen gets darker and Mr. Thomson cannot interact with the application for a couple of seconds.

The three migration types that can be chosen are characterized as follows:

Total migration – the whole application is migrated to the selected target device. After migration there are two independent applications – one on the source and one on the target device.

Migrate single components – only chosen components are migrated to the target device. After migration these components are also independent from the source application.

Synchronize – target and source device will be kept synchronized after completion of the migration process.

The traffic simulation has been migrated and can already be seen on the smart wall. However, in order to plan the possible response reactions Mr. Brown needs some more information. He asks Dr. Smith to also present the results of his flood simulation. Dr. Smith connects its application to the OPEN platform as shown in Figure 4.

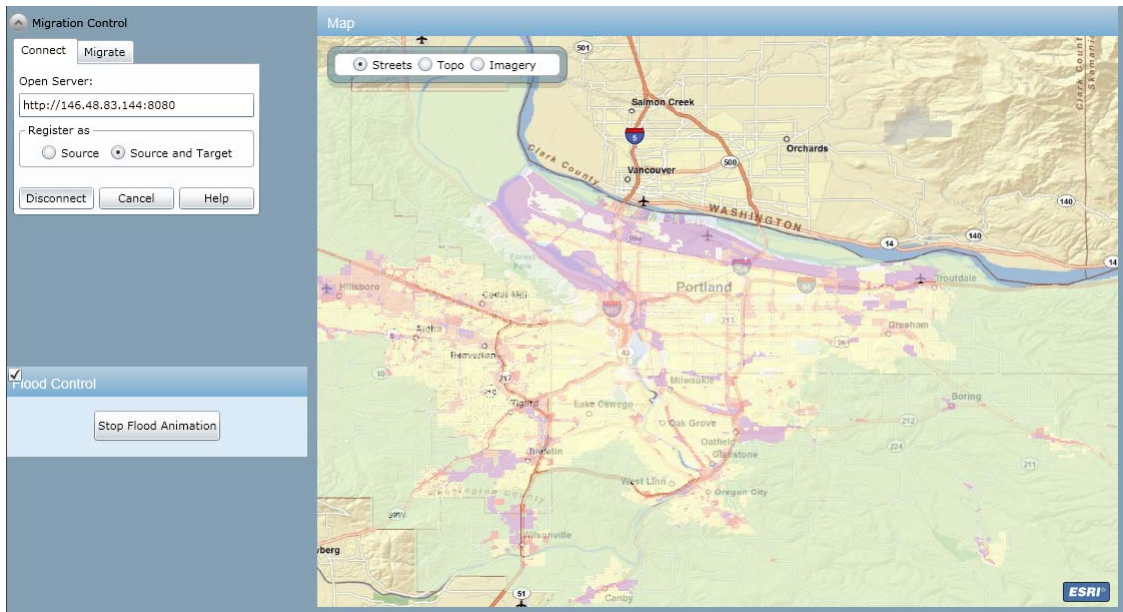


Figure 4: registration to the OPEN platform

Dr. Smith can directly choose the preferred migration settings and execute the migration since the smart wall has already been registered to the platform. He first changes the position of his migration control panel and puts it just on top of the map. Then he goes to the tab “Migrate”, selects the smart wall as target and picks up “Synchronization” as preferred migration type. Again Figure 5 shows that the components selected for migration (in that case the flood simulation) are highlighted by a yellow frame.

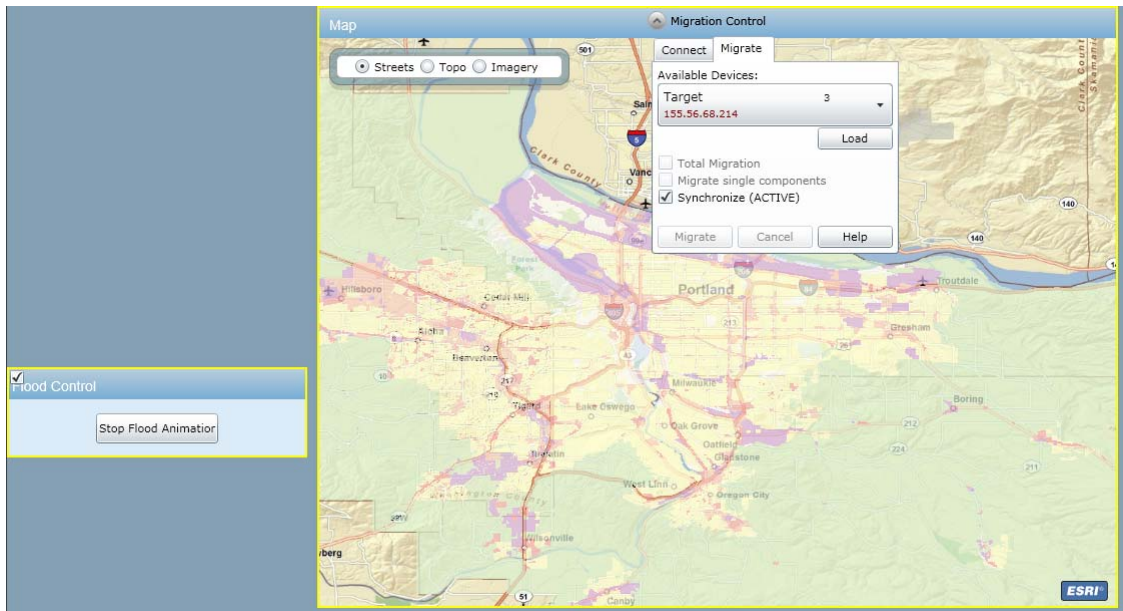


Figure 5: component selection for migration

After this second migration process has been accomplished, all experts see on the smart wall both the traffic and flood simulation laid over each other on the correspondent map, as shown in Figure 6.

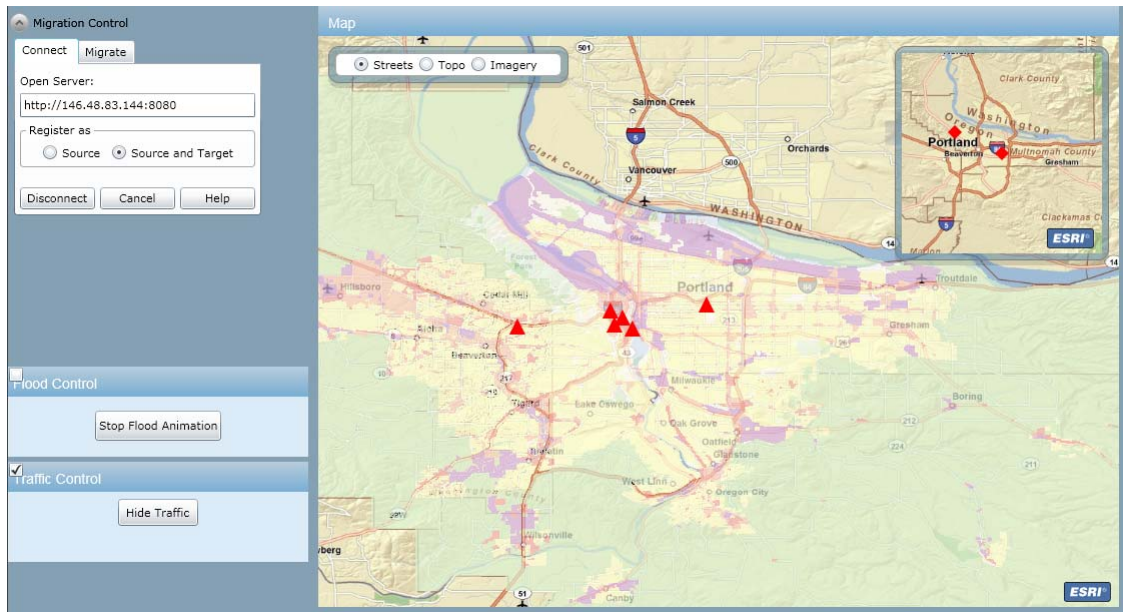


Figure 6: the overlaid traffic and flood simulations

Now the smart wall presents all needed information for the response planning. Mr. Brown can zoom in different regions and see them in detail on the bigger map, and still have the overview of the whole area in the smaller map depicted in the upper right corner. An example of that differentiated view is shown in Figure 7.

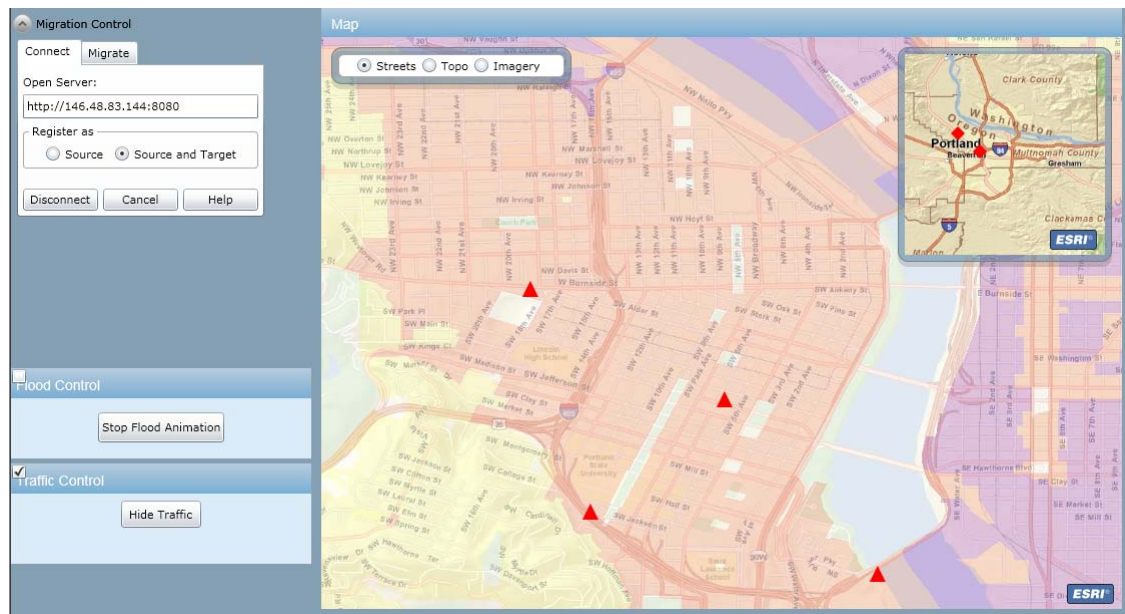


Figure 7: differentiated view of the Portland area

As described above, the usage of the Emergency prototype with the migration control panel includes altogether four steps:

1. Start the source application and connect to the OPEN platform either as source only or as source and target.
2. Register the target device to the OPEN platform.
3. Set migration type and, if necessary, select the components to be migrated.
4. Press the “Migration” button.

The here presented use case included all of these steps and described how to accomplish each of them.

3. SOCIAL GAME

This section presents the final integrated prototype of the Social Game. It discusses the hardware and software requirements for running it. Furthermore, the section also gives a short overview of the setup and run procedures of the prototype as well as a usage explanation of the migration case supported.

3.1. DESCRIPTION

The Social Game prototype is a rich web application that simulates a scenario in which the user can compete against real pilots while watching a live grand prix event, and interact with other members of a gaming community through a chat service. Some additional information about the track and the participants is also available, as well as pilots' performances and lap records. Moreover, the user can bet on the live race's results according to different parameters. The developed prototype is a simplified version of this scenario and allows the user to play a multiplayer racing game, watch a video, chat with other users and simulate betting on a set of different parameters.

As shown in Figure 8, the Social Game is organized into several elements, each taking up a different area of the screen, namely (from left to right): IPTV, Additional Content, Chat, Betting, and Racing Game. The access to some of these functionalities is handled by an authentication system.

The final Social Game prototype makes full use of the Migration Service Platform, and showcases the use of its functionalities, including: the Migration Orchestrator, the Context Management Framework and the Web UI adaptation.

A more detailed description of the Social Game prototype can be found in D5.3, section 4.2 [5].

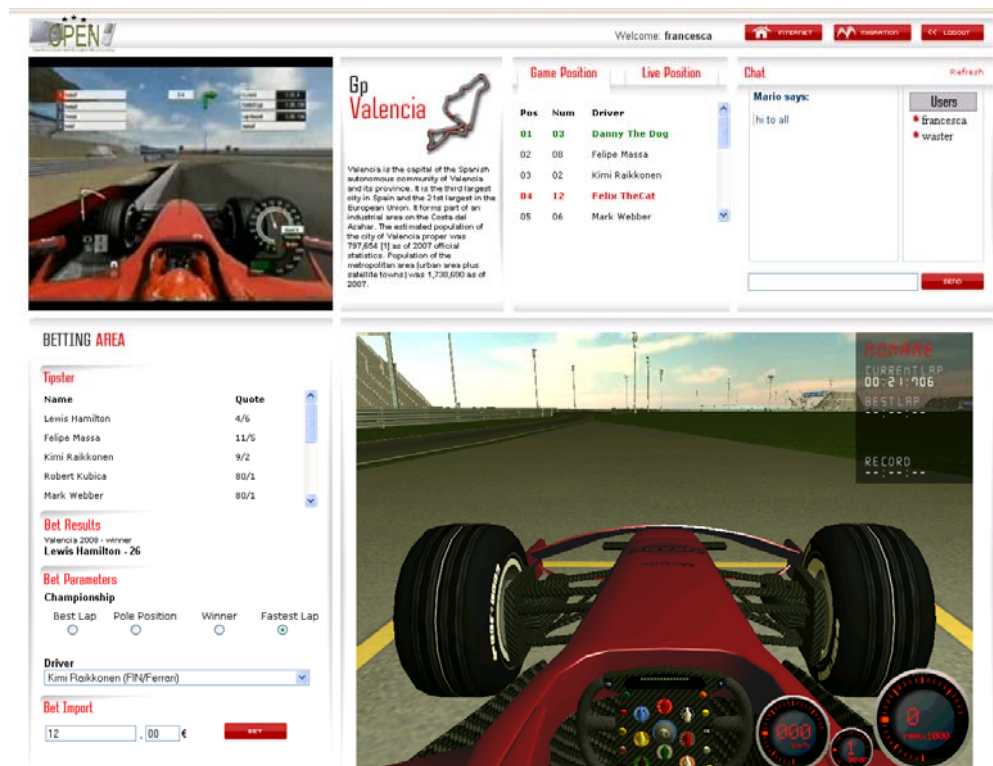


Figure 8: Social Game screenshot

3.2. HARDWARE AND SOFTWARE REQUIREMENTS

To properly run the Social Game prototype, a specific hardware and software configuration is needed.

The Racing Game is implemented as set of three applications: a client running inside the browser and two separate servers, one devoted to handle physics computations, the other one to manage the game logic. In principle, each of them can be run in a separate host, with the following minimum requirements:

- **Game Client:** Intel or AMD CPU running at 2Ghz with 1Gb RAM, ATI or NVIDIA GPU supporting Shader Model 2.0, and 100Mb of free disk space, Display with 1280x1024 resolution; Mozilla Firefox 3.x with JavaScript enabled running on Windows XP SP2 operating system.
- **Game Logic Server:** Intel or AMD CPU running at 1 Ghz, 256 Mb RAM, 100 Mb of free disk space; Windows XP SP2 or Linux operating system (Ubuntu 8.04 or above is recommended).
- **Physics Server:** Intel or AMD CPU running at 2 Ghz, 512 Mb RAM, 50 Mb of free disk space; Windows XP SP2 or Linux operating system (Ubuntu 8.04 or above is recommended).

Regarding the network configuration, each application needs a network connection with at least 2 Mbit bandwidth and a set of open TCP and UDP ports (see below). Moreover, an internet connection is needed.

Anyway, all the applications can be run on a unique host, provided that a more powerful hardware configuration is used. In this case, the minimum requirements are the following:

Intel or AMD Dual Core CPU running at 2.2 Ghz, 1 Gb RAM, ATI or NVIDIA GPU supporting Shader Model 2.0, network adapter with an internet connection, and 100Mb.

Finally, the Migration Service Platform needs a Java Runtime Environment installed.

3.3. SETUP AND RUN

To install all the applications under Windows operating system, a self-extracting setup package *setup_opengame.exe* has been provided, which allows selecting the applications to be installed. The typical configuration consists of two servers running on one PC and a number of clients running in different machines, connected to the servers through a LAN or an Internet connection. As a particular case, one client and the servers can be run on the same machine.

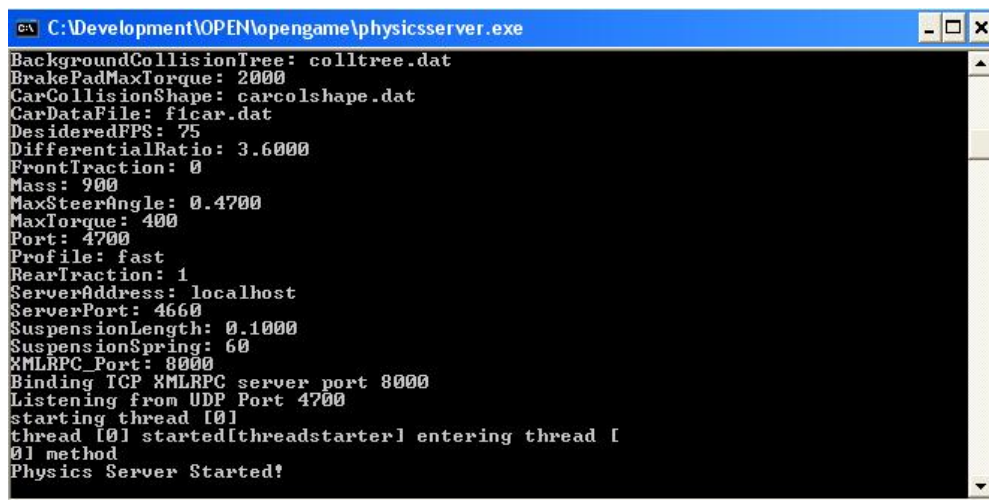
During the installation process, it is recommended to avoid changing the proposed destination folder, in order to allow the correct operation of the software.

In all the PCs that run the Social Game, the openGame plugin for Firefox must be installed. To this aim, the *open_plugin.xpi* package is provided. You can simply drag the file into the Firefox window and follow the instructions.

In order to support the gaming environment, both the Game and the Physics Servers must be run and configured for listening to clients connections.

First of all, launch the Physics Server through the corresponding shortcut in the start menu. By default, TCP port 4700 and UDP port 8000 are used for communicating with the Game Server. In case you need to change these ports, you can edit the "Port" and "XMLRPC_Port" settings in the configuration file *physicsserver.cfg*, which is located in the installation folder.

After launching the Physics Server, wait for the message "Physics Server started!" in the open console, as shown in Figure 9.



```
C:\Development\OPEN\opengame\physicsserver.exe
BackgroundCollisionTree: colltree.dat
BrakePadMaxTorque: 2000
CarCollisionShape: carcolshape.dat
CarDataFile: f1car.dat
DesideredFPS: 75
DifferentialRatio: 3.6000
FrontTraction: 0
Mass: 900
MaxSteerAngle: 0.4700
MaxTorque: 400
Port: 4700
Profile: fast
RearTraction: 1
ServerAddress: localhost
ServerPort: 4660
SuspensionLength: 0.1000
SuspensionSpring: 60
XMLRPC_Port: 8000
Binding TCP XMLRPC server port 8000
Listening from UDP Port 4700
starting thread [0]
thread [0] started[threadstarter] entering thread [
0] method
Physics Server Started!
```

Figure 9: Physics Server console window

Then, launch the Game Server through the corresponding shortcut in the start menu. By default, UDP port 4660 is used to communicate with the Physics Server and the clients. If you need to change this port, you can edit the “Port” setting in the configuration file *server.cfg*, located in the installation folder. Note that, in this case, you need to change also the “ServerPort” setting in the Physics Server configuration file *physicsserver.cfg*.

Once both servers are up and running, you can open the Social Game web page through the corresponding shortcut in the start menu. Before that, you should check if the client is properly configured to connect to the servers. To do this, open the client configuration file *client.cfg* and modify “ServerAddress” and “ServerPort” settings with the IP address and UDP port where the Game Server is hosted. If the client and the servers are running on the same machine, you should use “localhost” or “127.0.0.1” as “ServerAddress”.

Note that clients by default use UDP port 4662 for incoming connections. You can change this by modifying the “Port” setting in *client.cfg*.

Finally, each player should choose a different name by editing the “Name” setting in the client configuration file *client.cfg*.

The following instructions show how to setup and run a PC as source device and another PC as target device for performing a partial migration with state preservation. The source device must be compliant with the hardware and software requirements described in section 3.2. The Orchestration Server can be installed on the source device, on the target device or on another PC connected to the network. The Orchestrator Client must be installed on both source device and target device.

The MSP software related to the orchestration, is located in two zip files: *OrchestrationServer.zip* and *OrchestrationClient.zip*

To install the Orchestrator Server, the file *OrchestrationServer.zip* must be unzipped in a selected folder of the chosen device. To launch the OrchestratorServer, the file *runOrchestratorServer.bat* must be launched.

The *OrchestrationClient.zip* must be unzipped in a selected folder in both source and target devices. The variable PATH in the file *runOrchestratorClient.bat* must be changed in order to point to the installation path of the Java Virtual Machine. In order to enable the communication between the Orchestration clients and the Orchestration Server, the following conditions must be verified:

- TCP ports 8989 and 7098 must be open in both source and target devices (Check if this is true also for the device that hosts the Orchestration Server)
- the URL for connecting Orchestration Client and Orchestration Server must be set. The URL can be set in the file *resource\config.properties* in the Orchestration Client installation folder. The IP addresses values must be changed (bold values in the box below).

```
OrchestratorClientPort=7098
deviceName=YourDeviceName
deviceURL=http://IPAddressTarget:7098/xmlrpc
OrchestratorServerURL=http://IPAddressSource:8989/xmlrpc
```

Finally, the file *launchSocialGame.bat* must be copied in C:\

3.4. USAGE

The usage instructions for the Social Game prototype have been presented in D5.2, section 3.2.3 [4]. In this section only the usage of the migration part is described. The prototype supports different migration types, including multitarget migration that consists of migrating different components in different devices. Figure 10 shows an example of partial migration, where different components are migrated from a source PC to a target PDA and a target PC.



Figure 10: multitarget migration

The user interface for the migration is integrated within the prototype, as shown in Figure 11 and Figure 12. **Errore. L'origine riferimento non è stata trovata.** When the user press the migration button on the top right, the list of migrable components appears and s/he can select the components s/he wants to migrate.

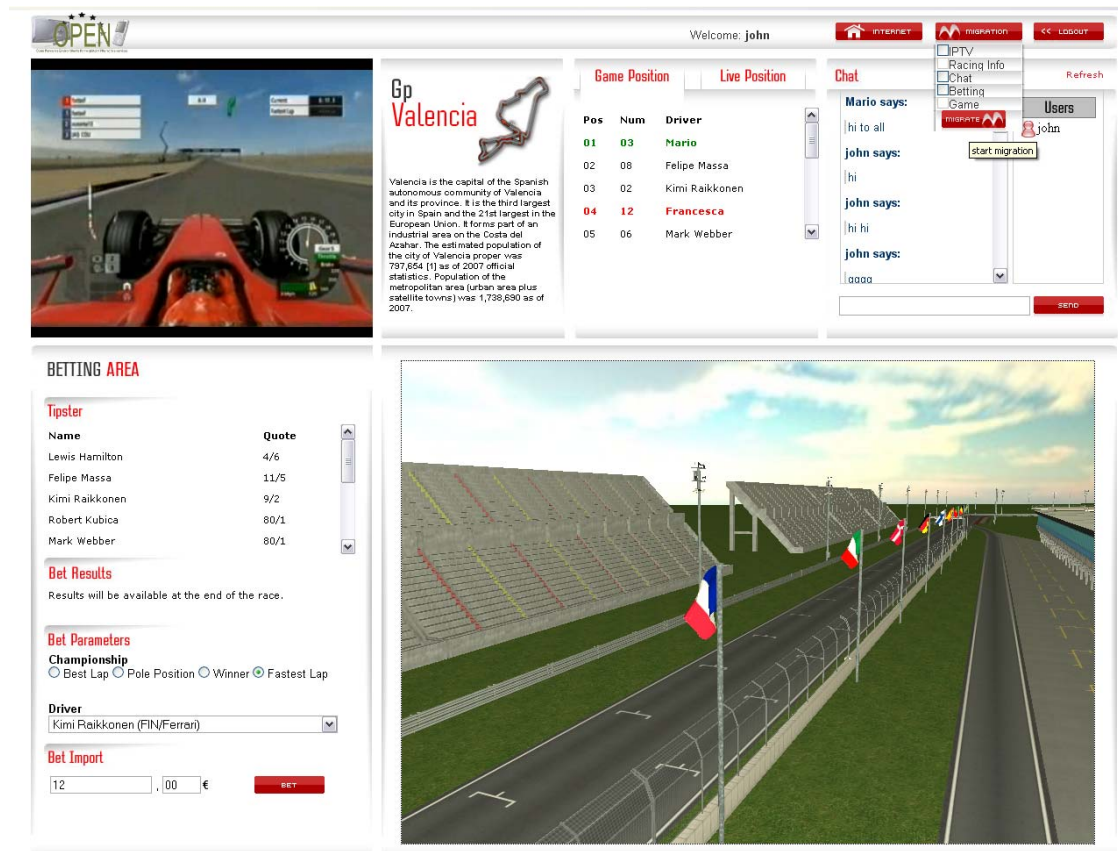


Figure 11: list of components

After choosing the components to be migrated, the list of available devices supporting the selected components is displayed in the menu (see Figure 12).

The screenshot displays a web application for a racing game. At the top, a navigation bar shows 'Welcome: john' and buttons for 'INTERNET', 'MIGRATION', and 'LOGOUT'. Below this, the interface is divided into several sections:

- Game View:** A first-person view of a race car on a track.
- Gp Valencia:** A panel with a map of the Valencia circuit and text describing the location: "Valencia is the capital of the Spanish autonomous community of Valencia and its province. It is the third largest city in Spain and the 21st largest in the European Union. It forms part of an industrial area on the Costa del Azahar. The estimated population of the city of Valencia proper was 797,664 [1] as of 2007 official statistics. Population of the metropolitan area (urban area plus satellite towns) was 1,738,690 as of 2007."
- Game Position Table:**

Pos	Num	Driver
01	03	Mario
02	08	Felipe Massa
03	02	Kimi Raikkonen
04	12	Francesca
05	06	Mark Webber
- Live Position Table:** (Empty table)
- Chat:** A window showing messages from 'Mario says: hi to all', 'john says: hi', 'john says: hi hi', and 'john says: gaaa'. A 'SEND' button is at the bottom.
- BETTING AREA:**
 - Tipster:** A table listing tipsters and their quotes:

Name	Quote
Lewis Hamilton	4/6
Felipe Massa	11/5
Kimi Raikkonen	9/2
Robert Kubica	80/1
Mark Webber	80/1
 - Championship:** Radio buttons for 'Best Lap', 'Pole Position', 'Winner', and 'Fastest Lap' (selected).
 - Driver:** A dropdown menu showing 'Kimi Raikkonen (FIN/Ferrari)'.
 - Bet Import:** A text input field with '12', a currency symbol '€', and a 'BET' button.

A 'Migrate to pc2?' pop-up dialog is overlaid on the center, featuring a PC icon, the text 'Components: CHAT', and 'MIGRATE' and 'CLOSE' buttons.

Figure 12: list of devices and confirmation pop-up

Once the user has selected the components to be migrated and the target device, s/he must confirm her/his choices, through a pop-up dialog.

After the migration has been performed, the selected components are deactivated in the source device and activated in the target device. In Figure 13 an example of a PC source device after the migration of Chat and Betting is shown.

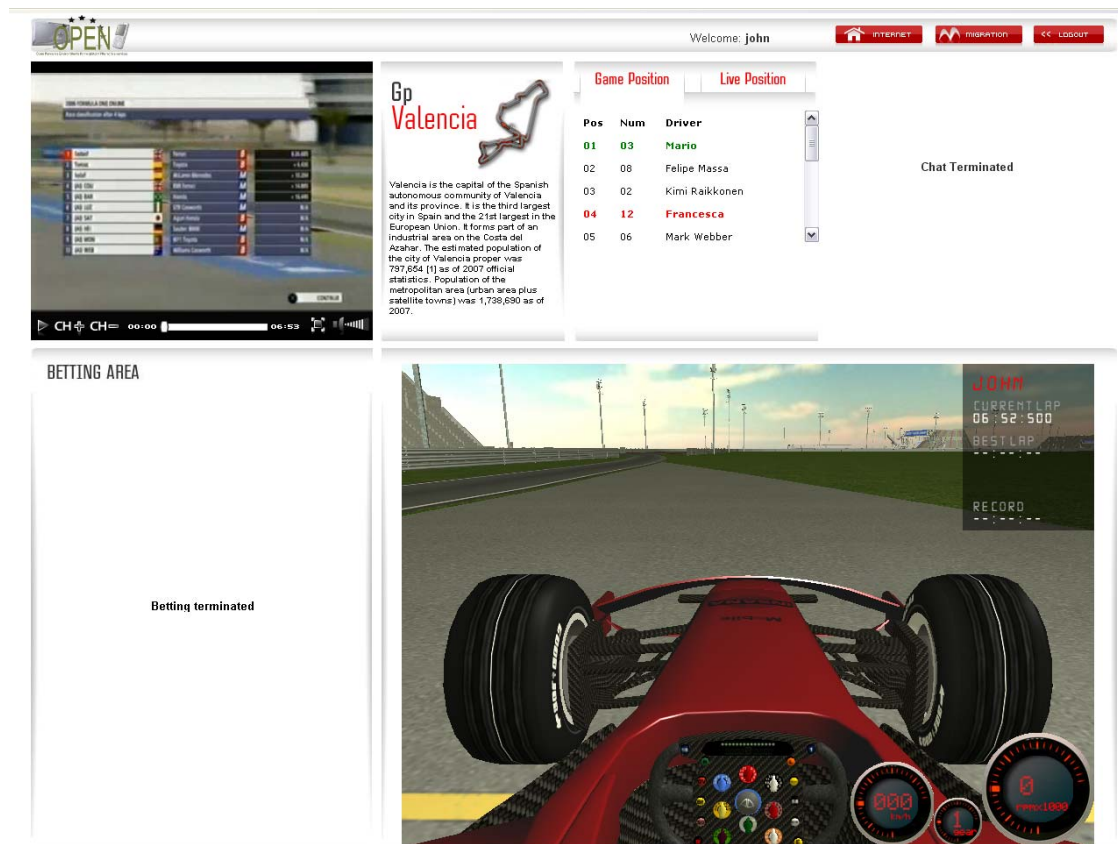


Figure 13: Social Game on source device after migration of Chat and Betting

Figure 14 shows the target device after the Chat has been migrated.

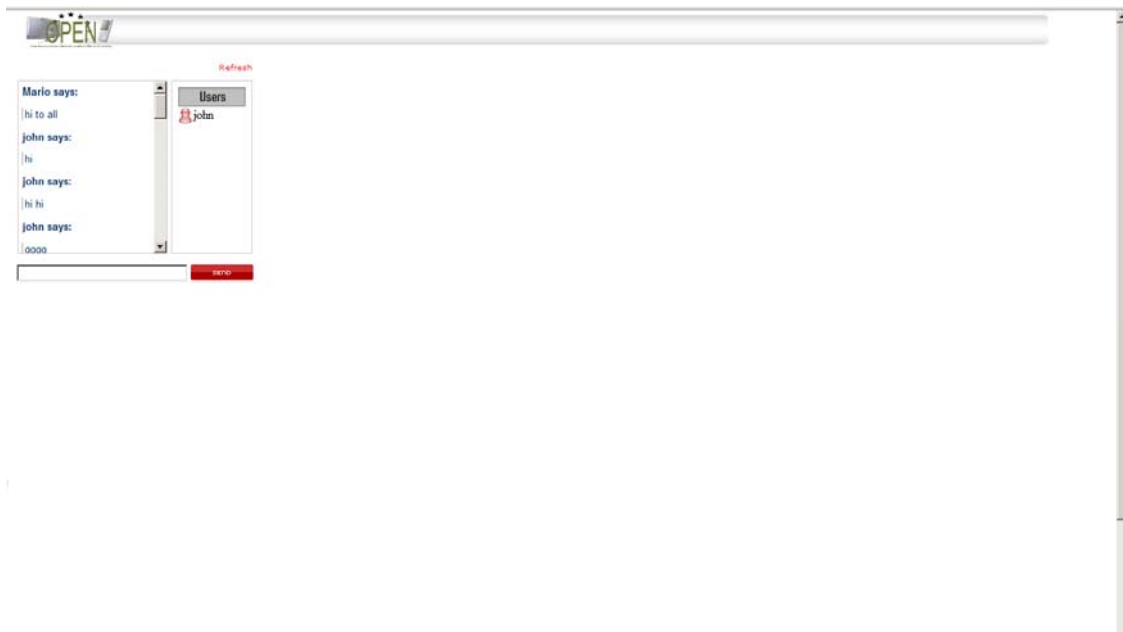


Figure 14: Chat migrated on a target device

Figure 15 shows another target device after the Betting has been migrated



Figure 15: Betting migrated on another target device

It is worth noting that the state of the components is preserved after the migration on the target device. For example, the user can continue in the target device the betting procedure s/he had started in the source device.

Instead of using the user interface provided by the Social Game, the user has the option to select the components and the target device from the user interface provided by the MSP, which has been described in details in D2.5 [4], sections 3 and 4.

4. TWITTER WALL

4.1. DESCRIPTION

The TwitterWall application is a social interaction tool where strangers can discover their common interests and start a conversation. While it has been described in detail in D5.3, this document provides an overview with a focus on the implementation aspects.

The application looks initially like a regular Twitter client for mobile devices. Whenever the user walks near an Open aware display, however, it can be partially migrated to show a public view that is adapted to the capabilities of the display, both in terms of available resolution and multicore capabilities.

This public output on a large display (seen in Figure 16) is actually a multiuser application component, where many users can input search terms on their mobile devices, while showing the combined query results on the public screen. More interestingly, matches between different users are highlighted, in order to identify common topics of interest and break the ice towards a conversation among people standing in front of the screen.

The application makes full use of the Migration Service Platform, and showcases the use of its functionalities, including: the Migration Orchestrator, the Trigger Management, the Application Logic Reconfiguration and the Context Management Framework.

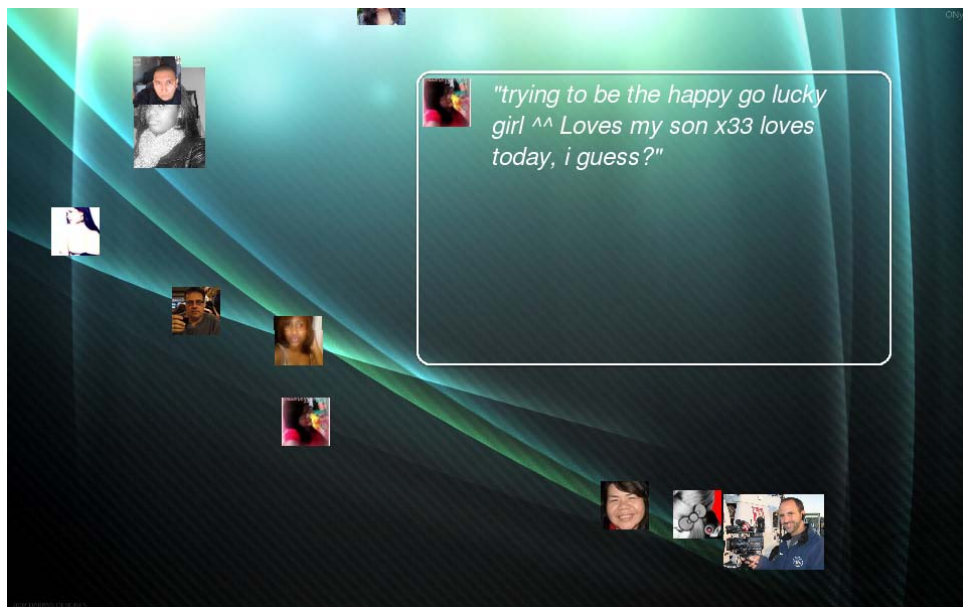


Figure 16: the TwitterWall in shared mode on a public display

4.2. HARDWARE AND SOFTWARE REQUIREMENTS

The application requirements vary from component to component. Two of the components run in a mobile device: the input and output component, which looks like a regular twitter client, and the input only component, which allows users to input keywords on the big display. These are implemented in Java using the SWT toolkit for UI design. Given the nature of Java, the components can be run on any device that has a native Java Virtual Machine and SWT support. This includes Windows Mobile phones (v5.0 and above), small embedded PCs with a graphics card and full-fledged PCs (Windows/Linux/MacOSX).

Because of the Open integration, these components also rely on the Open platform for registration and triggering.

The output component, shown in Figure 16, meant to run on a large display, also requires a Java Virtual Machine and SWT support. When optimizing for multicore processors, the Namuco library is used, which is natively implemented in C and requires a Linux operating system (Tested on Ubuntu 9.10, kernel version > 2.6). Additionally, an accelerated graphics card is recommended to handle the visual interface, especially on high resolution devices. Finally, an RFID reader and a Bluetooth dongle are used for user intention detection and device discovery.

4.3. SETUP AND RUN

These installation instructions assume that the Migration Service Platform has been installed both on the large public display, and on the mobile devices. Given that, only the application components need to be installed and executed.

On the large display, the TwitterWall component is fitted with an executable that will be run as appropriate by the orchestrator. No installation and configuration are needed. The TwitterWall will therefore need to register itself with the Open Platform, just like any other component.

The client software, which needs only be copied on the device, also includes mechanisms to execute it directly from Java code, which the Orchestrator client can handle. It is therefore enough to install the TwitterWall classes alongside the Open client software. Additionally, an RFID tag needs to be attached to the phone to make it detectable by the RFID reader.

4.4. USAGE

The TwitterWall application starts, in any number of mobile devices, as a standalone application that supports both input of search terms, and output of the search results. When approaching a supported display, the Trigger Management prioritizes the Application Logic Reconfiguration setups, and may choose to partially migrate the output component to the large display. This is shown in Figure 17: following the migration trigger, the Input + Output component is shut down in the mobile terminal, and its place, the input only is launched. Additionally, the output component comes into play in the large display.



Figure 17: partial migration on approaching a large display

5. PAC-MAN

5.1. DESCRIPTION

PacMan is a game where a character called PacMan, which is steered by the user, has to collect dots in a maze, as shown in Figure 18.

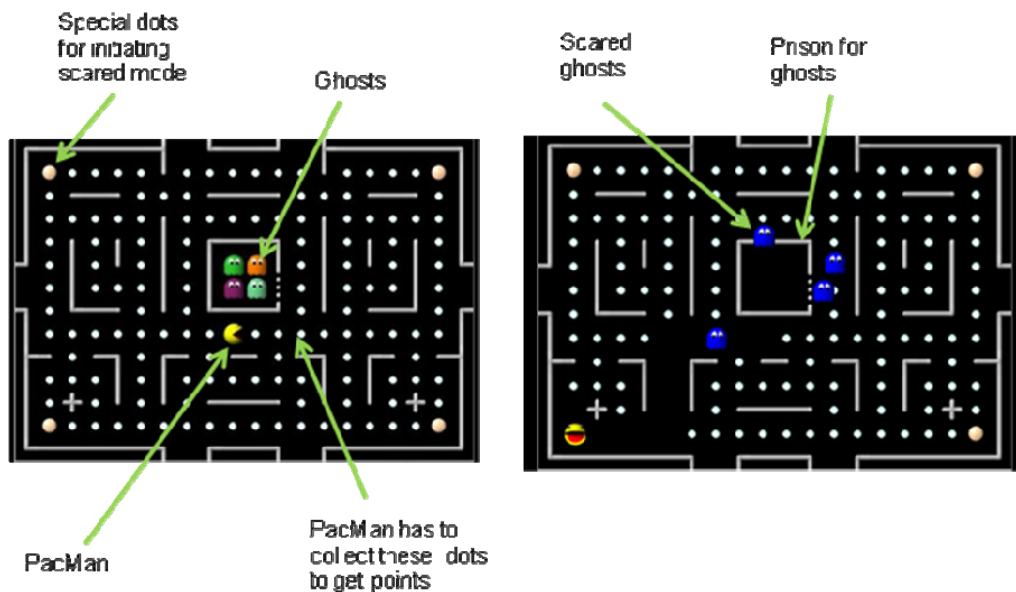


Figure 18: the two modes of a single PacMan game. On the left hand side the game is in normal mode where the ghosts try to catch the PacMan. On the right side the game is in scared mode where the PacMan can catch the ghosts.

Ghosts, who are controlled by the computer, are running around with the goal to catch the PacMan. If the PacMan collects special dots, ghosts and PacMan change roles for some seconds like depicted in Figure 18 on the right hand side. That means that the PacMan now can catch ghosts and that the ghosts try to run away. Caught ghosts will be imprisoned in the middle of the maze for some seconds. After some seconds the roles change back again. The goal for the player is to get as much scores as possible by collecting dots and catching ghosts.

The prototype is designed to work in two different device types: a desktop and a mobile device. The desktop device has a big display and a keyboard to control the game. In the mobile device the display is small and just a small control panel. The game can be migrated from the desktop to the mobile device.

On the desktop platform the Pacman game is implemented in a single XHTML-based page consisting of different components: the maze of the game, some UI controls for playing the game (e.g. for starting a new game, or pausing the currently running one, etc.) and some UI objects for specifying/visualizing the current configuration/status of the game (e.g. the number of Pacman lives still available, the currently selected animation speed of the game, etc.)

5.2. HARDWARE AND SOFTWARE REQUIREMENTS

The PacMan prototype consists of two parts:

1. PacMan UI (mobile/desktop) / OPEN Client
2. Application logic / OPEN Client / Open Server

These parts have different hardware and software requirements.

5.2.1. PACMAN UI (MOBILE/DESKTOP) / OPEN CLIENT

The Pacman UI prototype has been developed in XHTML and Javascript. Basically, the XHTML code specifies the presentation part of the prototype, while the JavaScript code handles the logic of the game. On the desktop platform an implementation of the game based on XHTML 1.0 has been used, while the version of the Pacman for the mobile device has been implemented in XHTML Mobile Profile language. On the desktop platform, a browser able to interpret both XHTML and JavaScript code is required (e.g. the most recent versions of Internet Explorer). Regarding the mobile devices, Internet Explorer Mobile v 6.5 has been verified to work well for the PacMan application example, in which also JavaScript excerpts are included.

In addition, in order to select the migration target and activate migration, on the source platform a Migration Client has to be available: this is a C# program, which needs .NET Framework on desktop platforms and .NET Compact Framework (version 3.5) on mobile devices.

5.2.2. APPLICATION LOGIC / OPEN CLIENT / OPEN SERVER

The device for the application logic / OPEN Client / Open Server has to provide a network card for the web service, the connection from the orchestrator client to the orchestrator server and from the context management framework (CMF) to the clients.

The application logic and the OPEN platform are implemented in Java. To execute the software at least Java Runtime Environment 1.5 has to be installed.

The CMF works with the operating systems Windows and Linux. Other operating systems are not yet supported.

5.3. SETUP AND RUN

5.3.1. PACMAN UI (MOBILE/DESKTOP) / OPEN CLIENT

In order to support migration, the Migration Server should be running and properly working. In addition, all the client platforms that want to subscribe to the migration service and will be possibly involved in a migration (either as a source platform or a target one), should be running the Migration Client software.

In order to start the Pacman game on the desktop platform, the user has to select the “New game” button in the XHTML page of the Pacman desktop prototype (see Figure 19). The URL for starting the Pacman is the following one: <http://146.48.83.144:8080/OpenDemo/pacman/pacmanUI/p.html>

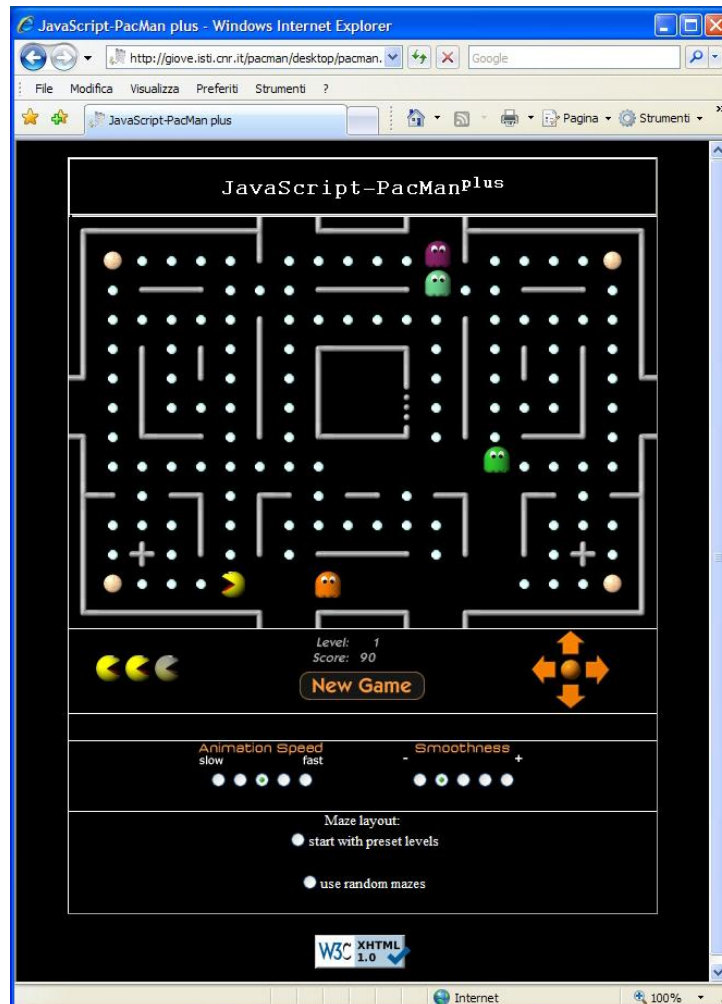


Figure 19: the desktop version of the Pacman prototype

5.3.2. APPLICATION LOGIC / OPEN CLIENT / OPEN SERVER

Open client for application logic

The application logic consists of the ghost logic component. The ghost logic component is delivered by an executable java archive called “GhostLogicComponent.jar” and a configuration file called “GhostLogic.ini”. To setup the ghost logic component these files have to be stored in one folder. In the configuration file the following entries have to be updated:

- LocalIp= [the ip of the target platform (server) out of the clients view e.g. 135.34.6.2]
- Port= [the port of the target platform view e.g. 8765]
- OpenServerAddress= [the address of the open server e.g. http://139.0.65.1:8989]
- CmfServerAddress= [the address of the cmf server e.g. http://139.0.65.1:8989]
- WebServiceAddress= [the address of the target platform (server) out of the clients view e.g. http:// 135.34.6.2:8989]

The ghost logic component can be started by the following command:

```
java -jar GhostLogicComponent.jar
```

If the operating system supports a direct execute of jar files, the ghost logic component can be started by a double click.

Open server

The Open server is delivered by an executable java archive called "OpenServer.jar" and a configuration file called "OpenServer.ini". To setup the Open server this files have to be stored in one folder. In the configuration file the following entries have to be updated:

- LocalIp= [the ip of the target platform (server) out of the clients view e.g. 135.34.6.2]
- Port= [the port of the target platform view e.g. 8765]
- OpenServerAddress= [the address of the open server e.g. http://139.0.65.1:8989]
- CmfServerAddress= [the address of the cmf server e.g. http://139.0.65.1:8989]
- WebServiceAddress= [the address of the target platform (server) out of the clients view e.g. http:// 135.34.6.2:8989]

The Open Server can be started by the following command:

```
java -jar OpenServer.jar
```

If the operating system supports a direct execute of jar files, the Open server can be started by a double click.

The context management framework (CMF) consists of the file ContextAgent.exe and can be started at the server.

5.3.3. USAGE

The considered migration scenario is a desktop-to-mobile where a user starts to interact with the desktop version of the Pacman and then moves to a mobile device.

After playing for a while the game on the desktop platform, the user decides to migrate towards a mobile device. Therefore, the user has to select, through the Migration Client, the platform towards which the migration will be activated, among the available devices (in our case a mobile device). After doing this, the application will be migrated on the new platform.

When migrating to the mobile device, the adaptation process will create two UI pages on the mobile device as a result of the UI adaptation splitting strategy. This is a consequence of the more limited capabilities of the target mobile device, which do not allow including all the UI elements in a single page. Regarding the state of the game, it has been preserved and then re-activated in the mobile device at the point where the migration was triggered on the desktop platform. Therefore, for instance, the current level and score achieved, as well as the positions currently held by both Pacman's and ghosts' characters are saved and restored on the mobile version, in order to allow the user to continue the interaction with the application from the point where s/he left off. In Figure 20 below you can see an example of adapted version of the Pacman prototype for a mobile platform is shown.



Figure 20: an example of adaptation of the Pacman prototype for a mobile device

6. CONCLUSIONS

In this document a description of the four application prototypes integrated with the MSP and developed for WP5 has been presented. These prototypes should be considered as demonstrators of the OPEN technology as well as validators for the services that the MSP provides. Each prototype exploits different aspects of the migration platform and shows a different use case. In particular, for each of them a brief description, installation and usage instructions have been presented.

7. REFERENCES

- [1] <http://msdn.microsoft.com/de-de/silverlight/default.aspx>
- [2] <http://www.microsoft.com/silverlight/get-started/install/default.aspx>
- [3] D2.5 – Engineered infrastructure for migratory interfaces
- [4] D5.2 – Initial prototype applications
- [5] D5.3 – Final application requirements and design