



**Integrated Project on Pervasive Gaming  
FP6 - 004457**

WorkPackage WP6: Infrastructure

**Deliverable D6.8: Final release of the IPerG platforms**

Olov Ståhl, SICS

Jan Ohlenburg, FIT

Chris Greenhalgh, Nottingham University

Ville Nenonen, Nokia

Release date: 31 August, 2007

Status: public



## EXECUTIVE SUMMARY

This document describes the final releases of the Morgan, PART and MUPE platforms, as a result of work carried out in the infrastructure work package during year three. For each platform, the main modifications are described, along with instructions on how to access and install the corresponding software components. The document also describes extension to the Equip2 and the Web Application Framework, a reusable web application framework supporting the development and integration of Java-based web applications. Note that since each software release contains lots of documentation in the form of programming and API reference manuals, sample code, tutorials, etc, this type of information is not included in this document. Instead, the reader is encouraged to visit the corresponding platform web sites for more detailed information on each release. Pointers to these web sites can be found in the sections 3 – 6 of this document.

The software releases described in this document can be divided into public releases and releases requiring licensing. The public software releases consist of:

- Extension to PART as described in section 3, publicly available as PART version 1.2 on SourceForge . PART has been developed entirely within the IPerG project. The extensions include a tool that can be used to visually browse the state of a running PART session, better support for file logging and a port to the SNAP hardware platform .
- Extension to MUPE as described in section 4. These extensions are part of the publicly available MUPE version 3.0 release. MUPE is pre-existing know-how and was brought into the project by Nokia. Most extensions are focused on better support for handset GUI elements.
- Extensions to the Equip2 dataspace library and the Web Application Framework (WAF), described in section 6. Equip2 has been developed by Nottingham independently of IPerG. The Web Application Framework is a component framework aimed at supporting the development of Web applications using Equip2, and was initially developed within the City as Theatre showcase to support the Day of the Figurines game. Both Equip2 and the Web Application Framework are publicly available on SourceForge. The extensions made during the third year include a port to the Nokia N800 platform and Debian Linux and better support for reloading web applications.

The software releases requiring licensing consist of:

- Modifications to the Morgan system as described in section 5. Due to Morgan's current licensing model, this release is not freely available to the general public. However, interested parties may apply for a license from FIT. Morgan is pre-existing know-how and was brought into the project by FIT. Extensions include support for more IO devices, as part of the DEVAL device abstraction layer, as well as a light weight Morgan version targeted at handheld devices.

There has also been a number of integration activities, for instance between MUPE and Equip2/WAF and between PART and PIMP.

## Deliverable Identification Sheet

<b>IST Project No.</b>	<b>FP6 – 004457</b>
<b>Acronym</b>	<b>IPerG</b>
<b>Full title</b>	Integrated Project on Pervasive Gaming
<b>Project URL</b>	<a href="http://iperg.sics.se/">http://iperg.sics.se/</a>
<b>EU Project Officer</b>	Albert GAUTHIER

<b>Deliverable</b>	<b>D6.8 Final release of the IPerG platforms</b>
<b>Work package</b>	<b>WP6 Infrastructure</b>

<b>Date of delivery</b>	<b>Contractual</b>	M36	<b>Actual</b>	31-Aug-07
<b>Status</b>	version. 1.0		final <input checked="" type="checkbox"/>	
<b>Nature</b>	Prototype <input checked="" type="checkbox"/> Report <input type="checkbox"/> Dissemination <input type="checkbox"/>			
<b>Dissemination Level</b>	Public			

<b>Authors (Partner)</b>	Olov Ståhl, (SICS, ), Jan Ohlenburg (FIT), Chris Greenhalgh (Nottingham University) Ville Nenonen (Nokia)			
<b>Responsible Author</b>	Olov Ståhl		<b>Email</b>	olovs@sics.se
	<b>Partner</b>	SICS	<b>Phone</b>	+4686331560

<b>Abstract (for dissemination)</b>	This document describes the final IPerG releases of the Morgan, PART and MUPE platforms, as well as the Equip2 dataspace platform and its web application framework. These releases are the result of work carried out in the infrastructure work package during year three. For each release, the main modifications are described, along with instructions on how to access and install the corresponding software.	
<b>Keywords</b>	IPerG platform, PART, MUPE, Morgan, middleware for pervasive gaming, Equip2, dataspace, web application framework	

<b>Version Log</b>			
<b>Issue Date</b>	<b>Rev No.</b>	<b>Author</b>	<b>Change</b>
2007/09/15	0.9	Olov Ståhl	First full version



2007/10/01	0.92	Olov Ståhl	Fixed all references, sent for review
2007/10/09	0.95	Olov Ståhl	Modifications after input from reviewer Anneli Avatare
2007/10/12	1.0	Olov Ståhl	Modifications after input from reviewer Mårten Stenius

## TABLE OF CONTENTS

<a href="#"><u>EXECUTIVE SUMMARY.....</u></a>	<a href="#"><u>I</u></a>
<a href="#"><u>TABLE OF CONTENTS.....</u></a>	<a href="#"><u>IV</u></a>
<a href="#"><u>1 INTRODUCTION.....</u></a>	<a href="#"><u>1</u></a>
<a href="#"><u>2 THE BIGGER PICTURE.....</u></a>	<a href="#"><u>1</u></a>
<a href="#"><u>3 PART.....</u></a>	<a href="#"><u>3</u></a>
<a href="#"><u>3.1 The use of PART within IPerG.....</u></a>	<a href="#"><u>3</u></a>
<a href="#"><u>3.2 Modifications to PART during the third year.....</u></a>	<a href="#"><u>4</u></a>
<a href="#"><u>3.2.1 PART Object Browser.....</u></a>	<a href="#"><u>4</u></a>
<a href="#"><u>3.2.2 File logging.....</u></a>	<a href="#"><u>5</u></a>
<a href="#"><u>3.2.3 Better support for object subscriptions.....</u></a>	<a href="#"><u>6</u></a>
<a href="#"><u>3.2.4 Port to the SNAP microcontroller.....</u></a>	<a href="#"><u>7</u></a>
<a href="#"><u>3.3 Supported platforms.....</u></a>	<a href="#"><u>7</u></a>
<a href="#"><u>3.4 Download and installation.....</u></a>	<a href="#"><u>7</u></a>
<a href="#"><u>3.5 Contact.....</u></a>	<a href="#"><u>7</u></a>
<a href="#"><u>4 MUPE.....</u></a>	<a href="#"><u>8</u></a>
<a href="#"><u>4.1 The use of MUPE within IPerG.....</u></a>	<a href="#"><u>8</u></a>
<a href="#"><u>4.2 Modifications to MUPE during the third year.....</u></a>	<a href="#"><u>8</u></a>
<a href="#"><u>4.2.1 Better control of GUI elements in the MUPE client.....</u></a>	<a href="#"><u>8</u></a>
<a href="#"><u>4.2.2 Extended support for images.....</u></a>	<a href="#"><u>8</u></a>
<a href="#"><u>4.2.3 Handset game startup.....</u></a>	<a href="#"><u>9</u></a>
<a href="#"><u>4.3 Supported platforms.....</u></a>	<a href="#"><u>9</u></a>
<a href="#"><u>4.4 Download and installation.....</u></a>	<a href="#"><u>9</u></a>
<a href="#"><u>4.5 Contact.....</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>5 MORGAN.....</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>5.1 The use of Morgan within IPerG.....</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>5.2 Modifications to Morgan during the third year.....</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>5.2.1 DEVAL - A device abstraction hierarchy.....</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>5.2.2 Morgan Light.....</u></a>	<a href="#"><u>10</u></a>
<a href="#"><u>5.3 Download and installation.....</u></a>	<a href="#"><u>11</u></a>
<a href="#"><u>5.4 Contact.....</u></a>	<a href="#"><u>11</u></a>
<a href="#"><u>6 EQUIP2 AND THE WEB APPLICATION FRAMEWORK.....</u></a>	<a href="#"><u>11</u></a>
<a href="#"><u>6.1 The use of Equip2 and WAF within IPerG.....</u></a>	<a href="#"><u>11</u></a>



6.2 Modifications to Equip2 and WAF during the third year.....	12
6.2.1 Port of Equip2 to Debian Linux on Nokia N800.....	12
6.2.2 Support for ActionScript.....	12
6.2.3 Support for dynamic reloading of web applications.....	13
6.3 Supported platforms.....	13
6.4 Download and installation.....	13
6.5 Contact.....	14
<b>7 INTEGRATION ACTIVITIES.....</b>	<b>14</b>
7.1 Integration between PART and PIMP.....	14
7.2 Integration between PART and the log file analysis and evaluation tool.....	17
7.3 Integration between MUPE and the Web Application Framework.....	19
<b>REFERENCES .....</b>	<b>20</b>



## 1 INTRODUCTION

This document described the final release of a number of pervasive gaming platforms that have been (partly) developed within IPerG, and used to build various pervasive games in the showcases. These platforms are:

- **Morgan** – Focused on pervasive games based on AR/VR. Developed and maintained by FIT.
- **PART** – Focused on pervasive games based on ubiquitous computing principles (embedded processors, use of sensor and actuator hardware, etc.) Developed and maintained by SICS.
- **MUPE** – Focused on pervasive games run in mobile phones, connected to a server backend. Developed and maintained by Nokia.
- **Equip2 and the Web Application Framework** – Focused on pervasive games where players can participate using application embedded in ordinary web pages. Developed and maintained by Nottingham.

While Morgan, PART and MUPE are examples of platforms that have been developed within WP6 from the start of the project, the Web Application Framework is an example of a software that started its development in a showcase (City as Theatre) and was later moved to WP6.

Each platform release is briefly described in the following sections, along with instructions on how the software can be accessed and installed. It should be noted however that since all of the releases (except Morgan) are freely available as open source on various web sites, this document doesn't contain a great deal of detail concerning issues like programming manuals, example code, etc. This type of information is available on the web, and can be found via the various links that are given in the text below. Furthermore, since all of the platforms described in this document are used in other projects as well, it is not unlikely that new ("non-IPerG financed") releases of these platform will become available during the last phase of the IPerG project. If such releases do become available, they may be relevant to IPerG and for instance used in the last year showcase game activities.

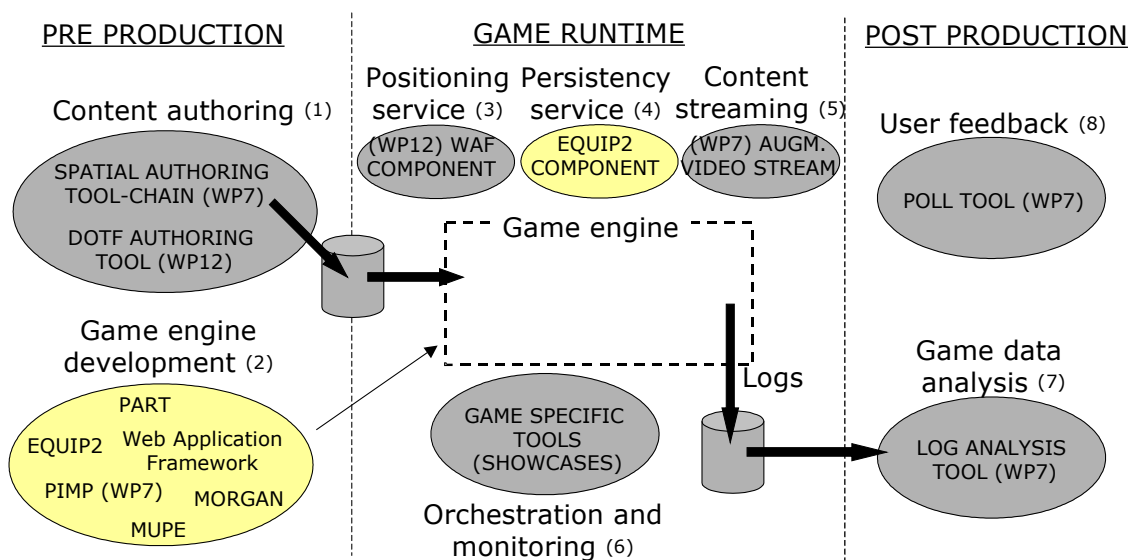
## 2 THE BIGGER PICTURE

The infrastructure (WP6) and tools (WP7) work packages are developing various platforms and tools that can be used in the development, test, deployment and pre-production of pervasive games. This development has been going on during the whole duration of the project, and has mostly been driven by the needs of the showcases. At the same time, the showcases have also been developing their own tools to support specific game instances. In some cases, such tools have later been moved from a particular showcase into either WP6 or WP7, where it has been developed further. These tools are typically the ones that have the potential of being used in other showcases as well, and since showcases typically don't have resources to develop such tools after the game activities in which they have been developed and used have ended, the development has been moved to WP6 or WP7 to ensure continuity.

The current state of the tools and platform work within IPerG is summarised by figure 1 below. This figure tries to illustrate into which functional areas (e.g., content authoring)

of pervasive games development and deployment the IPerG tools and platforms can be categorised, and how these areas relate. There is also a division into three temporal phases:

- Pre production – Tools and platforms for the production of the software system that is used to run the game, as well as various assets (images, text, videos, physical artefacts, etc) that the game will use.
- Runtime – Tools and platforms used to provide various services to the running game (e.g., positioning services) as well as providing support for data logging and game master monitoring and control.
- Post production – Tools and platforms used to analyse data gathered during game sessions, interview players, etc.



**Figure 1: The software components developed within IPerG, divided into a number of “functional areas” (the ellipses). The colored ellipses represent functional areas mostly focused by WP6.**

The functional areas in which IPerG is producing platforms and tools are listed below. As seen in figure 1, the platforms developed by WP6 are mostly used to build the games engine on top of which the games run, and to provide services to the running game (the coloured ellipses).

1. **Content authoring** – Tools for the production of various types of data loaded by the game engine during runtime. Examples of such tools include the spatial authoring tool-chain developed in WP7 (see ) and the authoring tools used for the production of the Day of the Figurines game in WP12 (see ).
2. **Game engine development** – Platforms and middleware used to build the software engine on top of which the games run. All platforms developed in WP6 belong to this category, as does the PIMP platform developed in WP7.
3. **Positioning service** – A run-time service that the game engine uses to get positioning information. An example of such a service is the Web Application

---

Framework positioning component that was developed for the Day of the Figurines game in WP12.

4. **Persistency service** – Services that allow game content and state to be persisted, so as to “survive” server restarts (e.g., due to maintenance, crashes, upgrades, etc). Examples include the Equip2 support for persistency using the Hibernate Object/Relational database system .
5. **Content streaming** – Services that are used to stream game content, typically from the game backend system to the devices carried by the players, or to spectators viewing the game without participating themselves. An example of such a service is the Augmented Video Streaming Tool developed in WP7 , which allows a live video feed to be augmented with an 3D graphics overlay, for instance used to indicate the positions of virtual objects.
6. **Orchestration and monitoring** – Tools that allow game masters to monitor the game progression, and to modify or add content while the game is running. So far in IPerG, such tools have mainly been developed within the showcases, for instance used in the Crossmedia Epidemic Menace game , and the Elarp Momentum game .
7. **Game data analysis** – Tools that can be used to analyse data collected during a game session. Examples include the Logfile Analysis and Evaluation tool developed in WP7.
8. **User feedback** – Tools that can be used to gather player feedback, e.g., concerning their game experience. Examples include the Polling Tool developed in WP7, which can be used to create online player interviews.

Concerning the work in the infrastructure and tools work packages in the final phase, there will be a large focus on packing and documentation. These work packages are currently in the process of creating *solution packages*, which are bundles of platforms and tools aimed at the production and deployment of certain categories of pervasive games. The packages will be delivered by the end of the project.

### 3 PART

PART is a light-weight distribution middleware that has been developed entirely within the IPerG project. The focus of PART is on mobile handsets, typically mobile phones, and a goal has therefore been to keep the size of the library small. PART provides services like setting up connections between game processes using various protocols (e.g., TCP, Bluetooth and IR), distribution of messages in-between such processes, and distribution and synchronisation of games state via a replicated object model. Detailed information about PART, its programming model and Java API can be found on the PART web site [part.sf.net](http://part.sf.net).

#### 3.1 The use of PART within IPerG

In the last phase of the project, PART has been used to realise the Momentum game in the Elarp showcase, and is also used to realise one of the games in the new Boxed Pervasive Games showcase.

In Momentum, PART was used to build both the Java midlet that ran in the mobile phones carried by all players, and also the backend server system where most of the

game logic was implemented. The game used PARTs synchronised object model to store and distribute game state, for instance the position of players, as well as game events, for instance the player discoveries of hidden RFID tags. PARTs persistency mechanism was used to store relevant game state in each mobile phone, allowing the midlet clients to “remember” certain types of events and states after restarts.

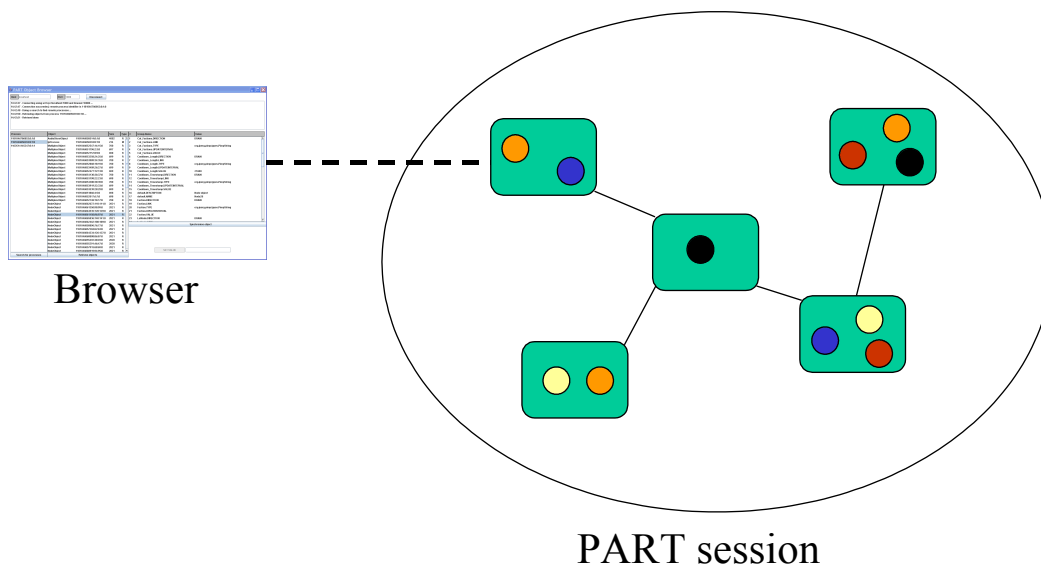
In the Box Pervasive Games showcase, PART is used as a building block in the development of a game based on tracking players as they move between different game zones, and triggering media (e.g., text messages, audio cues, etc.) on the players’ phone as they enter and leave zones.

### 3.2 Modifications to PART during the third year

During the third phase, the modifications to PART were mostly driven by the need of specific game development activities (e.g., the Elarp Momentum game). Therefore, there were no exact plans at the start of the phase concerning new functionality that needed to be implemented. In the following sub-sections, some of the modifications that were made on request by the showcases (primarily Elarp) are presented. The integration between PART and the PIMP tool developed in WP7 is described in section 7.1 at the end of the document. Furthermore, section 7.2 describes the PART support for the log file analysis and evaluation tool, developed within WP7.

#### 3.2.1 PART Object Browser

The PART Object Browser is a tool that was created as a result of the development, testing and deployment of the Elarp Momentum game. The browser is an object introspection tool that can be used to browse through the object state of a running PART (or PIMP) session. The browser connects to one of the processes participating in the session, and can then use this connection to extract information such as the identity of the session processes and the state of the objects held by these processes, see figure 2.



**Figure 2: The object browser connecting to a PART session**

The browser GUI allows a user to select a session process, and will then display all PART objects held by that process (see figure 3). Each such object can then be selected to view the state (name and value) of all of its properties. It's also possible to set the state of such a property via the browser. More detailed information about the browser can be found on the PART web site [part.sf.net](http://part.sf.net).

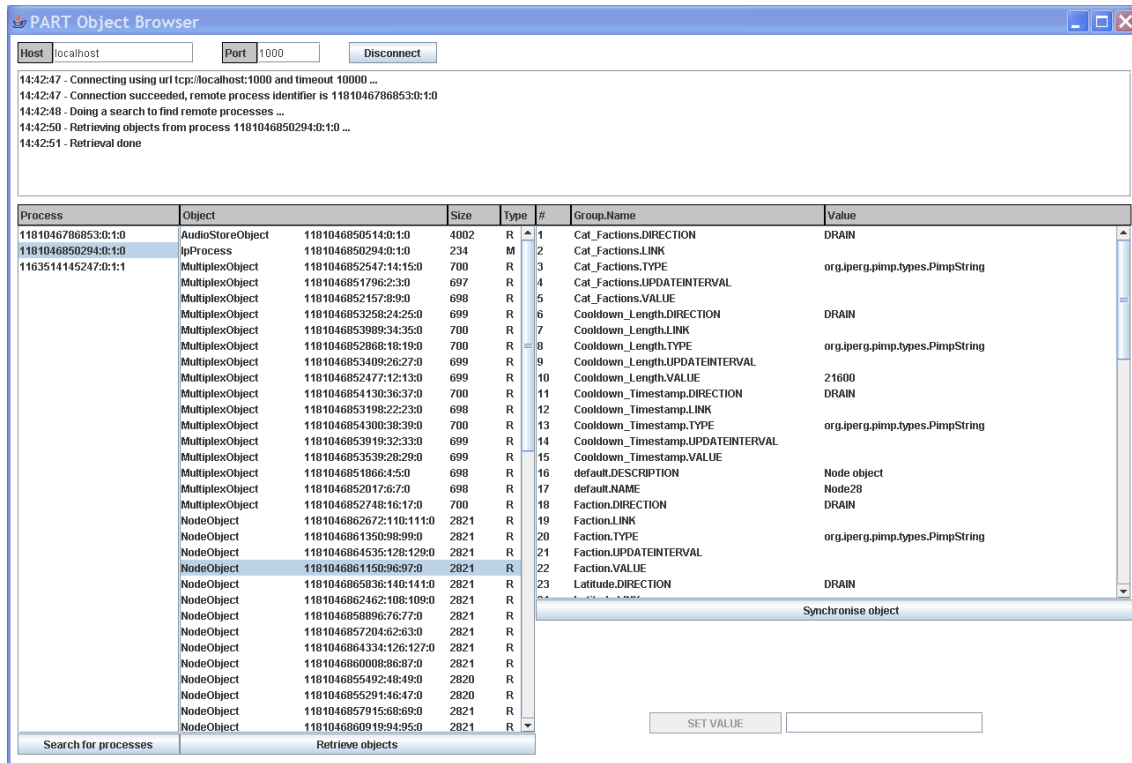


Figure 3: The browser GUI

During the Momentum activity in the Elarp showcase, the browser was used in several ways. During development of the game, the browser was used as a debug tool that could be connected to a running process to for instance verify that its object state was correct. The ability to change property values was also used to “simulate” various events. For instance, when a Momentum player discovers a RFID tag using a purpose built RFID reader, a property is set in an object representing that player. The update of such a property will then trigger an action by the game server. Using the browser tool, it was easy to test the handling of such events on the server side, since all that was required was to set the value of the correct property in a player object. Had the browser not existed, these kinds of test would have been much more complex to achieve. Once the game was running, the browser was used to monitor the game state and the players’ progression. By inspecting objects it was possible to determine where certain groups of players were located, if they had discovered any hidden RFID tags, if they had managed to occupy nodes, and so on. Details about the Momentum game can be found here .

### 3.2.2 File logging

PART provides an API for generating debug messages, which are by default printed on stdout. This is great when developing applications, since the programmer can turn on

debugging and then directly see the debug messages as they are generated. This even works for mobile phone development if the IDE running on the PC can display messages printed on stdout by the mobile phone, as for instance supported by the Device Explorer tool provided by Sony Ericsson. However, once the application has been deployed, printing messages on stdout may not be the ideal solution since they will probably not make much sense to the application users. The file logging feature was added to PART to allow processes to log messages to files instead of printing them on stdout.

The file logging module uses a file rotation scheme to prevent files from becoming too large, or to many. The application selects the file name, the maximum number of files it wants to use, and the maximum size of each file. If the current file to which PART writes messages becomes too large, PART will move on to the next file. Once the last file is full, PART will start to overwrite the first file, and so on. The reason for using several files instead of one is that the application will start logging to a new file each time it is started. If only one file is used, this means that the data on this file will be overwritten each time the application is restarted. If several files are used, and if the file size limit is selected carefully, a history of logs from different runs of the application may be achieved.

The file logging feature was used in the Elarp Momentum game. In Momentum, each player carried a mobile phone which connected to a backend server system whenever turned on. The Momentum phone client used file logging to log game events and system messages. The idea was that these log files could be used to trace the cause of errors should these occur on the phones. A drawback was of course that the players needed to hand in their phones to the IPerG crew in order for them to extract the log files. A future extension may be to allow log files to be inspected remotely, e.g., using the browser tool mentioned in section 3.2.1, or downloaded to some backend storage for further inspection.

### **3.2.3 Better support for object subscriptions**

PART uses a subscription mechanism to allow copies of the same object to be synchronised between processes. Similar to many systems providing a data subscription service, each PART subscription has a lease time that determines how long the subscription is valid. If the subscribing process doesn't renew the subscription within this time, the data publisher (i.e., the process holding the subscribed object) assumes that the subscription is no longer valid and removes it. In the previous release, 1.1, PART used one lease per subscription. This means that processes had to renew subscriptions on an object, or even property level.

When the Momentum game was being developed, it was soon realised that a large portion of the network traffic (almost 30 percent) was caused by subscription renew messages. Since this came as a surprise to the development team, an investigation was made to determine the reason why this traffic was so large in relation to the more game oriented traffic (game events, property updates, etc.) After a while it became evident that the reason was PIMPs use of PARTs subscription mechanism to implement its link concept (see section 7.1 for a brief description). Each link uses a PART subscription to achieve the data flow, and since links connects properties, not objects, it means that a lot of subscriptions will be set up if a lot of links are used. This was exactly the case in Momentum. For instance, if all players were active, the game server had close to 500

incoming and outgoing links, each one corresponding to one PART subscription which had to be renewed at a regular interval.

Since the Momentum link setup was not considered an extreme case, PARTs subscription mechanism was rewritten to better handle lots of subscriptions to and from a single process. The solution was to renew subscriptions on a process level, not on an object property level. This means that a process can renew all of its subscriptions to objects held by a remote process with a single message, not one message per object or even property as was the case before. For instance, in the Momentum case, this would mean that the server would only need to send around 30 messages if all players were active instead of around 500. Even though the server didn't actually send 500 messages to the network, due to message aggregation on the sending side, the new solution would still have reduced the network traffic caused by subscription renew information considerably.

### **3.2.4 Port to the SNAP microcontroller**

During the last phase, PART has also been ported to the SNAP hardware platform from Imsys Technologies AB . SNAP is a network-ready, Java-powered plug & play reference platform, ideal for remote control, data processing and managing of everything from small sensors to advanced factory equipment. This work has partly been done in another project, but should be beneficial to IPerG as well.

## **3.3 Supported platforms**

PART can be used on devices for which there exists a J2SE or J2ME capable virtual machine. For J2SE, Java version 1.4 or 1.5 is required. For J2ME, MIDP2 is required. PART has been tested on Windows 2000/XP PCs, SonyEricsson K700/K750i/K800i mobile phones and the TINI and SNAP microcontroller platforms.

## **3.4 Download and installation**

PART is publicly available as open source from Sourceforge under a BSD license . The project URL is [www.sourceforge.net/projects/part](http://www.sourceforge.net/projects/part). From the PART web page it is possible to download a release of PART version 1.2 in the form of pre-compiled jar files, documentation and some sample applications.

The PART source code is accessible via a CVS repository maintained by SourceForge. The source files can be browsed online by visiting <http://part.cvs.sourceforge.net/part/>. It is also possible to check out the sources to a remote computer. How this can be done is explained on the SourceForge project page for the PART cvs, [http://sourceforge.net/cvs/?group\\_id=167709](http://sourceforge.net/cvs/?group_id=167709).

## **3.5 Contact**

The PART software is maintained by Olov Ståhl, SICS. Questions concerning PART should be sent via email to the address [olovs@sics.se](mailto:olovs@sics.se).

---

## 4 MUPE

MUPE is an application platform for creating mobile multi-user context-aware applications. MUPE is a client-server system where clients are run on mobile phones and the server is running on the Internet. MUPE is developed by Nokia and is available under the Nokia Open Source License (NOKOS License) Version 1.0a . MUPE existed before IPerG started and was brought into the project by Nokia.

### 4.1 The use of MUPE within IPerG

In previous phases of the project, MUPE has been used to build several game in the Socially Adaptable Games showcase prototypes (e.g., Coup and Insectopia). These games are described in D9.8 . In the third phase, MUPE is used to build the Mythical: The Mobile Awakening game in the Massively Multiplayer Mobile (MMM) Games showcase.

### 4.2 Modifications to MUPE during the third year

In line with the global work plan concerning all IPerG platforms, most of the modifications to MUPE has been very showcase driven, instead of planned beforehand by WP6. As the main IPerG activity involving MUPE in the third year is the game developed by the MMM showcase, most changes originate from this work. The following sub-sections give examples of such changes. As part of the MMM work, there has also been an integration between MUPE and the Equip2 and Web Application Framework software. This is described in section 7.3 at the end of this document.

#### 4.2.1 Better control of GUI elements in the MUPE client

The development of the GUI for the Mythical: The Mobile Awakening game has required a number of changes to the MUPE client GUI API. Examples include the possibility to create text input fields inside the game area instead of having to use pop-up windows as before, the possibility to express the positioning and extent of GUI elements as permillages of the screen width and height (instead of percentages), the possibility to use bitmaps fonts, as well as support for drawing triangles on the screen.

#### 4.2.2 Extended support for images

Since the Mythical: The Mobile Awakening game uses quite a lot of images, there has been some work on the support for using images in MUPE GUIs, both on client as well as on the server side.

MUPE uses URL-like expressions to refer to images. A MUPE client that needs to display an image on the screen, use the URL to fetch the image over the network. Previously, URL could only refer to images on disk, for instance, held by a web server or the MUPE server. As part of the Mythical: The Mobile Awakening game, images can also reside inside an Equip2 dataspace, persisted using the Hibernate Object/Relational database system . MUPE has been extended so that the URLs can now refer to images stored in this database. The advantage is that all the assets used by the game can now be stored in one place, which can be easily accessed and modified using authoring tools built using the Web Application Framework.

---

There has also been some work on the MUPE client's ability to cache images in the mobile handset, so as to avoid downloading the same image several times, for instance when the game is restarted.

#### 4.2.3 Handset game startup

The "standard" way for a user to start a game in the mobile phone is to first start the generic MUPE client, and then select a game from a menu of available games presented by the client. In a way, the MUPE client can be seen as a "game player" application, where the actual games are "data files" that can be loaded and executed by the player.

For the Mythical: The Mobile Awakening game, a new version of the MUPE client was developed, one that can be configured to always load and run a particular game when started. Since the user no longer needs to use the menu to select the game, the user can start to play the game more quickly than was possible using the standard MUPE client.

The new MUPE client is installed alongside the standard client in the handset. Since the client only supports one game (e.g., Mythical: The Mobile Awakening), more clients need to be installed if other games should be supported (one client per game).

### 4.3 Supported platforms

The MUPE server side requires J2SE 1.5.0, and has been tested on Linux and Windows XP. The client requires MIDP 2.0, which is available on a majority of the new medium/high end phones. However, since MIDP 2.0 implementations often have bugs or are not following the standard, problems may still occur. The MUPE site hosts a list of problems encountered using various phone models, which can be found here <http://mupe.nrln.net/wiki/index.php/Phonematrix> (cited 070812).

### 4.4 Download and installation

Instructions on how to download, install and run the MUPE system can be found on the MUPE web site using this link <http://www.mupe.net/download/>. The MUPE version that includes most of the modifications made during the second phase of the project is 3.0.

The MUPE downloads available on the MUPE site are divided into client downloads and server downloads.

1. The client downloads allows the MUPE client to be downloaded and installed on a phone. There is also a file containing the source code of the MUPE client. Since the MUPE client doesn't contain any application logic, it needs to connect to a MUPE server in order to download a script to run the application. There is usually some test MUPE servers running at [mupe.no-ip.org](http://mupe.no-ip.org), (ports 8082, 8084, 8085, 8086, 8088, 8091, or 8092), but it's also possible to download a server distribution and set up a local server (see 2 below).
2. The server downloads allows for downloads of a number of demo game servers that can be installed and started on a local machine. Once downloaded and started, a MUPE client running on a phone can be directed to connect to the server in order to download the application script.

## 4.5 Contact

Up to date contact information can be found on the MUPE web site, [www.mupe.net](http://www.mupe.net). The FAQ on the MUPE web site gives answers to a number of questions, including various problems. There are also a number of MUPE forums that can be used to discuss issues and get in contact with other MUPE users and developers. The link to these forums are <http://www.mupe.net/forum/>. Emails concerning MUPE can also be sent to [webmaster@mupe.net](mailto:webmaster@mupe.net).

## 5 MORGAN

The Morgan system from FIT is designed to support the development of AR/VR applications. Morgan is a component-based framework that relies on the CORBA middleware for network communication. It currently supports many devices, including mouse and keyboard as well as haptic input devices, object tracking systems and speech recognition libraries. Morgan existed before IPerG started and was brought into the project by FIT.

### 5.1 The use of Morgan within IPerG

In the first two years of the IPerG project, Morgan was used in the Crossmedia showcase to build the Epidemic Menace game . In the third phase of the project, Morgan is used within the Boxed Pervasive Games showcase in the realisation of the AR box game. In this game, the Morgan AR/VR viewer Marvin is run on ultra-mobile PCs, where it is used to display 3D models on top of a video image. More information about the use of Morgan in the Boxed Pervasive Games showcase will be available in future deliverables from work package 14.

### 5.2 Modifications to Morgan during the third year

During the third year, the work on the Morgan platform has mainly been focused on stability issues, as well as packing and documentation. The Morgan documentation on the FIT web site has been extended with a Wiki containing tutorials, howtos, high level descriptions, and a frequently asked questions section. Morgan developers also get access to an online bug tracker from bugzilla . Some of the changes concerning the code base of Morgan are described in the sections below.

#### 5.2.1 DEVAL - A device abstraction hierarchy

The Device Independency Concept has been extended and renamed DEVAL, Device Abstraction Layer. Using DEVAL allows applications to be configured before or during runtime according to changing environments. Hence applications are not dependent on specific devices, but can be adjusted afterwards in case of changing environments or hardware setups, e.g. exchanging tracking systems or substituting speech commands with keyboard input. Compared to the previous Morgan release, DEVAL has been extended to support more input and output devices.

#### 5.2.2 Morgan Light

Morgan Light is the smaller companion of the Morgan system and is especially developed for the requirements of handheld devices such as PDAs and Smartphones.

The work on Morgan Light has continued, and has for instance been focused on supporting the augmentation of video streams with 3D objects, as used in the AR box game developed in the Boxed Pervasive Games showcase. However, since Morgan Light is being developed as a tool within WP7, it won't be part of the final Morgan release described in this document. A final version of the Morgan Light tool will be available at the end of the project.

### 5.3 Download and installation

Due to the Morgan system's licensing model, the 1.4 release of Morgan which includes the modifications presented in this document is currently not freely available to the general public. Researchers and developers of AR/VR projects may license the Morgan AR/VR Framework for evaluation, development and distribution. Academic institutes and universities may apply for free licenses. For more information on the licensing model and access to the Morgan platform for non-project partners, please contact Jan Ohlenburg (see next section for details). For project partners, contact Jan for more information regarding the download of the latest Morgan release.

### 5.4 Contact

Questions regarding Morgan should be sent to Jan Ohlenburg, FIT, [jan.ohlenburg@fit.fraunhofer.de](mailto:jan.ohlenburg@fit.fraunhofer.de).

## 6 EQUIP2 AND THE WEB APPLICATION FRAMEWORK

EQUIP2 is a dataspace platform that aims to support cross-platform and cross-language data sharing between networked clients. Equip2 provides a shared dataspace that can be accessed by distributed processes, providing synchronous, database-like operations such as adding, matching, updating and removing objects, and also asynchronous change notifications supported by optional template based filtering and matching. Equip2 is developed by Nottingham, but has been used in various activities within the IPerG project. The Web Application Framework (WAF) is a component framework aimed at supporting the development of Web applications using Equip2. The framework contains data-driven interface components, which can support authoring, configuration and monitoring views on the core Equip2 dataspace.

### 6.1 The use of Equip2 and WAF within IPerG

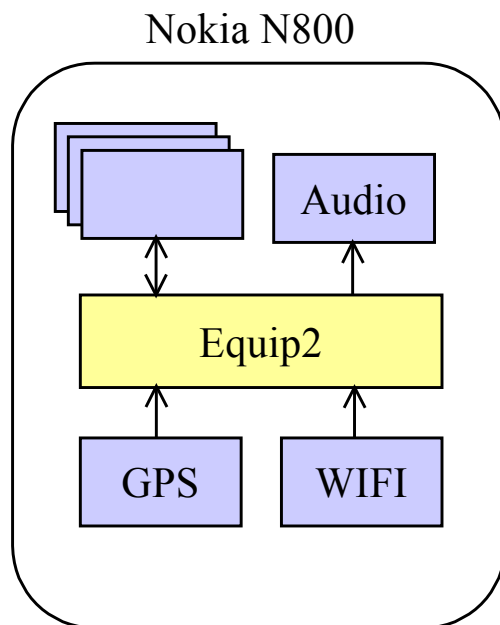
In the last phase of the project, Equip2 and WAF has been used in the City as Theatre showcase to support the performance of the Day of the Figurines II game in Berlin in October 2006 and Singapore in December 2006. Equip2 and WAF are also used in the ongoing Massively Multiplayer Mobile (MMM) Games showcase, mainly to support persistence and orchestration in the game's backend system. A third use of these software is in the Cultural Console Game (CCG) showcase, where Equip2 is used in the game engine, supported by a database for persistency, and WAF is used to build a web based authoring tool with direct access to the Equip2 dataspace.

## 6.2 Modifications to Equip2 and WAF during the third year

During the third year, the Equip2 platform and the Web Application Framework have been made open source software, publicly available from SourceForge under a BSD license. As part of this publication, lots of effort has been devoted to documentation, resulting in a number of guides on how to use various features of the software, sample applications, etc. also being available on the SourceForge project page (see section 6.4 below). Apart from the work on documentation and packaging, some changes have also been made to the code itself, a few examples are given in the sub-sections below. There has also been an integration of Equip2 and WAF with the MUPE platform, this is described in section 7.3 at the end of this document.

### 6.2.1 Port of Equip2 to Debian Linux on Nokia N800

The N800 port of Equip2 has been done as part of the game being developed in the Cultural Console Game (CCG) showcase. In CCG, the port allows Equip2 to be used as a datastore and event publisher running in a handheld game client, allowing indirect interactions between a set of components focused on tracking GPS locations, available WIFI access points, recording and playback of audio, etc (see figure 4). Each component can publish data in the datastore and/or subscribe to events concerning modifications to data relevant for that component.



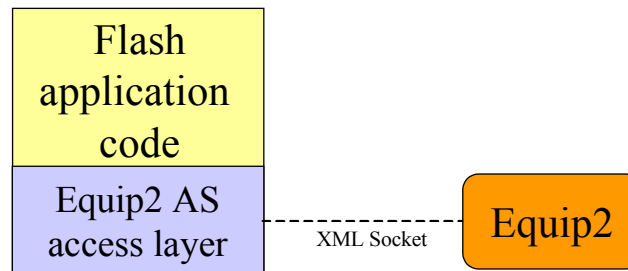
**Figure 4: The (tentative) use of Equip2 to build the CCG client**

Note that since the CCG game is currently under construction, the exact use of Equip2 in this work may change in comparison to what's being illustrated by figure 4. More information about the technical realisation of the CCG game will be presented in future deliverables from work package 17.

### 6.2.2 Support for ActionScript

As part of the game developed in the Cultural Console Game (CCG) showcase, Flash will be used to build some of the game client GUIs (currently the client running on a

PC, not the mobile version). To support the use of Flash, the Equip2 remote dataspace protocol has been extended to allow ActionScript-compatible text encoding to be used when communicating with a remote Equip2 dataspace. An ActionScript client code access layer has also been developed, which can be included in a Flash application (e.g., the game GUI to be developed in CCG) and which provides access to an Equip2 dataspace via a set of ActionScript methods. The access layer uses an XML-socket internally to communicate with the Equip2 dataspace, sending and receiving data as specified by remote dataspace protocol (see figure 5).



**Figure 5: Connection between a Flash application and Equip2**

More information about the Equip2 support for ActionScript can be found via this link [http://equip.sourceforge.net/equip2/docs/EQUIP2\\_ActionScript\\_Client.html](http://equip.sourceforge.net/equip2/docs/EQUIP2_ActionScript_Client.html) (cited 070825).

### 6.2.3 Support for dynamic reloading of web applications

During Java web application development on a local machine it can be frustrating and time-consuming to have to restart the web application after each file update for each code/test iteration. The Web Application Framework has therefore been extended with a number of new classes (e.g., dispatcher servlets and class loaders) that can be used to cause a running Web application to be “refreshed” instead of restarted when configuration files or game classes have been modified. More information about the support for reloading of web applications can be found using this link [http://equip.sourceforge.net/equip2webapp/samples/tutorial/docs/EQUIP2\\_WebApp\\_Dynamic\\_Reloadiing.html](http://equip.sourceforge.net/equip2webapp/samples/tutorial/docs/EQUIP2_WebApp_Dynamic_Reloadiing.html) (cited 070825).

## 6.3 Supported platforms

Equip2 currently targets J2SE, J2EE, J2ME and C++ (Windows, Unix and partially Symbian OS). The Web Application Framework requires Java 1.5, and has been tested on Linux and Windows XP. It is designed for web application use, and is typically deployed in a production setting on a Linux web server machine, behind an Apache web server.

## 6.4 Download and installation

Equip2 and the Web Application Framework is publicly available as open source from SourceForge, the link is <http://equip.sourceforge.net/equip2/index.html>. This page is in

---

turn linked to other pages containing installation guides, documentation and application development tutorials.

The source code is accessible via a CVS repository maintained by SourceForge. The source files can be browsed online by visiting <http://equip.cvs.sourceforge.net/equip/>. It is also possible to check out the sources to a remote computer. How this can be done is explained here <http://equip.sourceforge.net/equip2webapp/docs/index.html>.

## 6.5 Contact

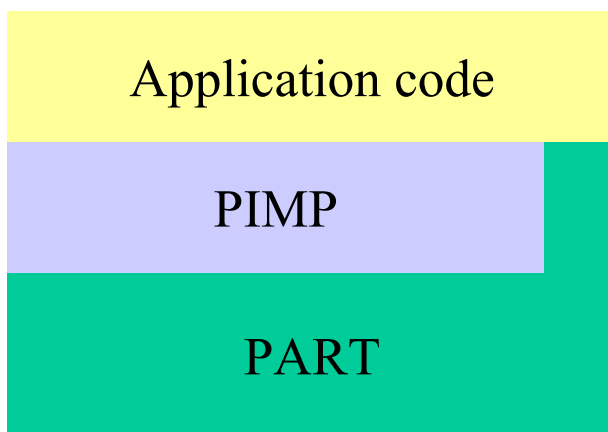
The core framework and data-driven interface are being maintained by Chris Greenhalgh, Nottingham University (email: [cmg@cs.nott.ac.uk](mailto:cmg@cs.nott.ac.uk)).

## 7 INTEGRATION ACTIVITIES

This section describes some of the integrations that have been made between various IPerG platforms and tools, typically as a result of some showcase game development activity. Furthermore, as WP6 and WP7 is currently producing a number of solution packages, that is, bundles of IPerG platforms and tools focused on different pervasive gaming domains (e.g., AR based games), some integration has been necessary in order to achieve coherency within these packages. The final versions of the solution packages will be delivered at the end of the IPerG project.

### 7.1 Integration between PART and PIMP

PART has been almost entirely integrated with the PIMP system developed in WP7. This integration, which is illustrated by figure 6 below, is done in such a way that PIMP use most of the services provides by PART (synchronised objects, messaging, persistent data storage, etc.) internally, and then adds its own functionality on top of this. Only a small portion of the PART API is actually exposed to the PIMP programmer.

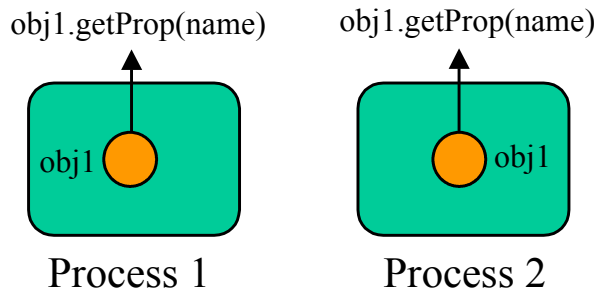


**Figure 6: The relation between PART and PIMP**

The PART/PIMP integration has been so successful that in most of the IPerG activities where PART has and is used, it's in this integrated form. This means that the developers in these activities are actually using PIMP to develop games. PART is of course used indirectly, but the developers are using PIMPs programming model and API, not

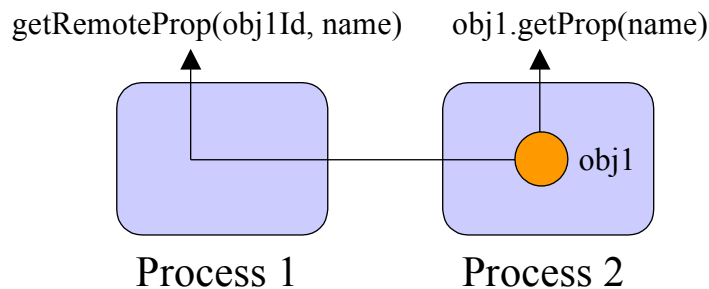
PARTs. One reason for not using PART directly is that PIMP has a better support for various sensor and actuator hardware, e.g., GPS receivers, RFID readers, Mogile sensor and actuator platforms, etc. Such hardware devices have been used frequently in the games produced within the Elarp showcase (e.g., Momentum and the Node Game), and also in the location box game produced in the Boxed Pervasive Games showcase.

In a way, PART and PIMP have a similar programming model. Both systems provide the concept of distributed objects that can be discovered and accessed by processes participating in an application (e.g., game) session. The objects may have properties, a form of attributes, which can be added and removed dynamically. Both systems also provide mechanisms for processes to be notified about updates to “remote” object properties. However, there is one big difference. While the PART model is based around replicated objects, i.e., an object created by one process can be copied and thus exist in several places, PIMP uses a non-replicated model where objects are always located in one process only. For instance, if a PART process needs to retrieve the value of an object property, it simply calls the *getProp* method on its own copy of the object, see figure 7.



**Figure 7: Accessing a property in PART**

In PIMP on the other hand, the process holding the object can retrieve the property value directly via the *getProp* method, while other processes need to fetch the value over the network as illustrated by figure 8.



**Figure 8: Accessing a property in PIMP**

The difference in the object model can be explained by the application focus of each system. Even though PART has a rather generic focus, the replicated object model is intended to support applications that need to represent some game state to be shared

among a set of processes, and potentially accessed rather frequently. In PIMP, objects are more solitary, primarily intended to represent some kind of sensor and actuator hardware, and since such objects will typically contain code that talks to the corresponding hardware device, the objects need to be stationary (since the hardware is).

Since it's fairly easy to implement a stationary, single copy, object model using a replicated object model (you basically hide the replication API from the application), we had no real problems in building PIMP's object model on top of PART. This means that PIMP objects are actually PART objects, which allow PIMP to use many of PART's services for object management, i.e., searching for objects on remote processes, setting and getting properties, subscribing to property update events, etc. A somewhat more complicated issue was providing PIMP's support for data flows, via the *link* concept. In PIMP, it's possible to set up a link between a property held by an object in one process and a property held by another object in a remote process. The effect of creating such a link is that when one of the properties is updated, PIMP makes sure that the property in the other object is updated in the same way. Links have directions, so the data will flow in only one direction, see figure 9.

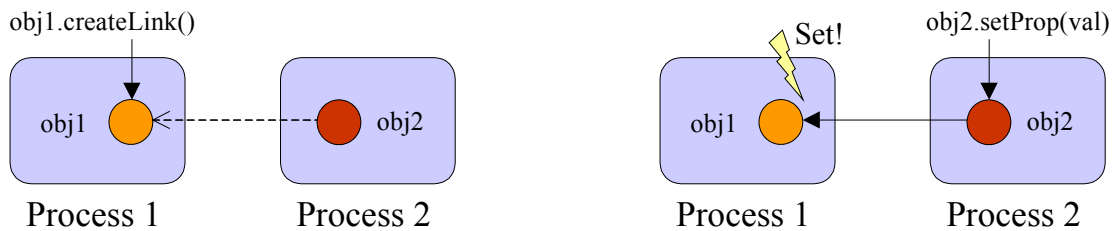


Figure 9a-b: Creating a link in PIMP

Since PART already provided a mechanism for distributing property values in-between processes, the challenge was to use this mechanism to implement the PIMP link concept, and not implementing something new from scratch. In PART, a process holding a copy of a replicated object can decide how this copy should be synchronised with other copies of the same object (held by remote processes), should they be updated. This is achieved using a subscription model, allowing a particular object copy to subscribe to updates on a remote copy of the same object. If the remote object copy is later updated, PART will make sure that the subscribing copy is updated in the same way, see figure 10.

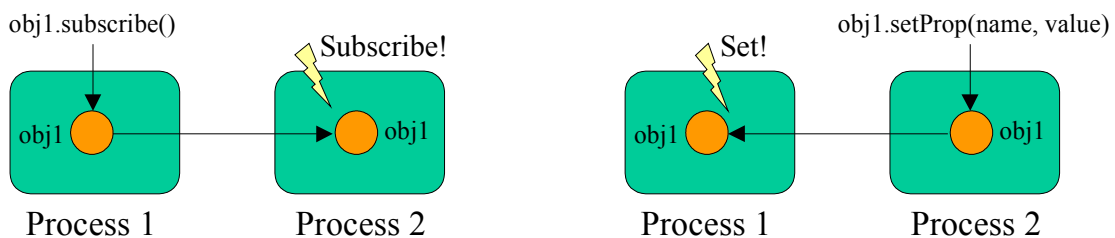
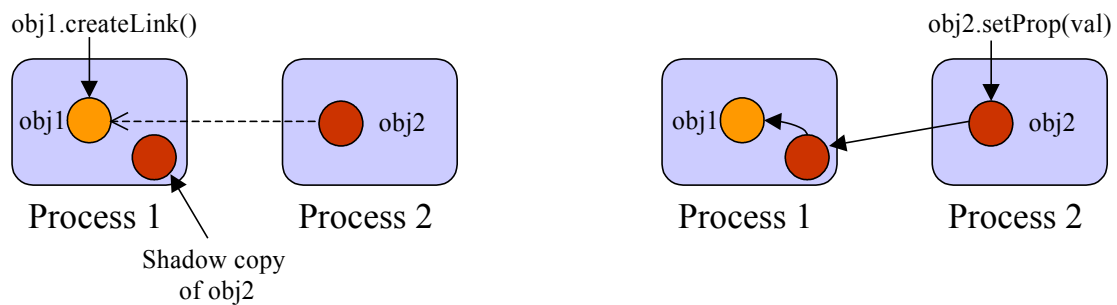


Figure 10a-b: Subscribing to PART objects

The main problem when using PART's subscription mechanism to implement PIMP's link concept was that the subscription mechanism is based on replication, which doesn't exist in the PIMP model. However, as mentioned above, since PIMP objects are in fact PART objects, they can be replicated. The solution was then to "secretly" replicate one of the PIMP objects (the one from where the link started) to the other process, and then set up a PART subscription so that this "shadow" copy would be updated if a property in the main object was updated. The update of the shadow copy would then be caught by PIMP, causing the PIMP object at the end of the link to be updated. This is illustrated by figure 11.



**Figure 11a-b: Supporting PIMP links using PART**

Although this solution works quite well, it has one major drawback. Since the PIMP objects are actually (secretly) replicated when links are created, they can't contain code which is specific to a particular device or even platform. For instance, a PIMP object existing on a mobile phone, can't use any J2ME specific code if there's any chance that the object will be replicated to a J2SE based platform. During the PIMP based game development in the third year, this was an annoying feature which caused a lot of confusion and workarounds. Part of the problem is that the PIMP model doesn't include replication, so the programmer only reading the documentation will not know that replication can in fact occur as a result of creating links. If the programmer then adds code which can't run on all platforms, which is fine according to the PIMP model, it's very confusing when error messages saying that object so and so can't be replicated. To solve this problem, parts of PIMP is currently being rewritten in WP7, so that in the future, PIMP objects will no longer be based on PART objects and need therefore not be replicated.

## 7.2 Integration between PART and the log file analysis and evaluation tool

As part of producing an IPerG solution package focused on pervasive gaming based on ubiquitous computing principles, PART has been extended to support the format used by the log file analysis and evaluation tool developed in WP7.

Logging is supported in PART via the *IpLogManager* interface. This interface defines a single method, *log*, which takes a message to add to the log.

```
log(String message);
```

This will produce logs where lines have the following syntax:

```
TT:MM:SS - message
```

For instance

```
11.10.33 - This is a log message
```

How the actual logging is done depends on the implementation of the log manager used. For instance, as described in section 3.2.2, PART supports logging to files, but other possibilities include logging to a database, logging to the record store on a mobile file, and so on.

To support the log file analysis tool, the logging interface of PART had to be modified to support the tool's more elaborate log syntax. The solution was to develop a specific log manager that was capable of generating logs in the format understood by the log file analysis and evaluation tool. A decision was also made to focus the log information on PART object property changes (additions, modifications and removal of properties). Property changes tend to be the most common event that occurs in PART applications, and since most application logic is typically somehow connected to properties, the state and changes of properties tells us much about what goes on in the application.

The log method of the new log manager was extended to include information about the update type (ADD, SET or REMOVE), property name, the object name, the source process (i.e., the process changing the value), etc. and was given the following profile:

```
log(String type, String procName, String objName,
     String propName,
     String newValue, String oldValue);
```

The first and second lines of each log file produced by the manager have the following format:

```
Time_Stamp Proc_Name Obj_Name Type Prop_Name new_Value old_Value
yyyy:mm:dd:hh:mm stringstring string string string string
```

Which means that an event logged using the log manager can look something like this in the log file:

```
2007:08:23:11:23 GameServer Player1 SET Score 23 12
```

Using the log file analysis and evaluation tool on files produced by this particular PART log manager, makes it possible to retrieve lot of statistics from an application (e.g., game) session, e.g., which properties are updated the most, which process does the most updates, at which days of the week do most updates happens, etc.

The log manager supporting the log file analysis tool format is part of the new PART release downloadable from SourceForge (see section 3.4).

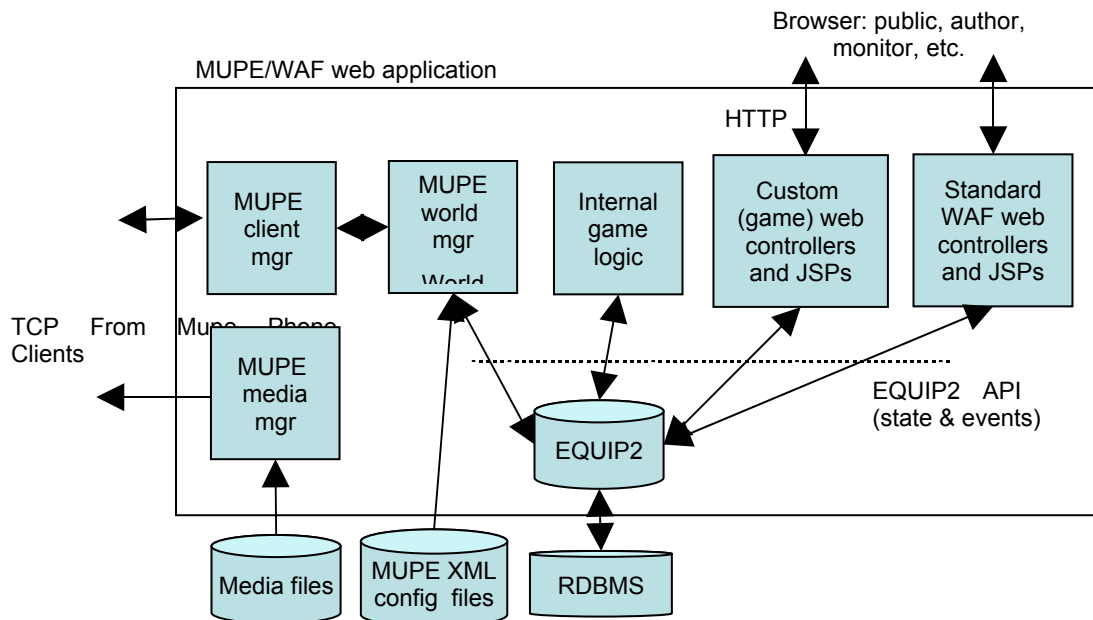
### 7.3 Integration between MUPE and the Web Application Framework

Taking the MMM (Massively Multiplayer Mobile Games) showcase as the driving application, we have integrated the MUPE mobile phone-based game platform with the Web Application Framework (WAF). This integration allows us to combine MUPE's support for rich game clients on mobile phones with the Web Application Framework's support for persistence, orchestration, analysis and HTML/browser based interfaces.

The Web Application Framework integrates the open source Java Spring Framework for application building with the Hibernate Object/Relational mapping system and the EQUIP2 dataspace for persistence and pattern-based notification. The MUPE/WAF integration incorporates a slightly modified version of MUPE into the Spring application framework and links it directly to the EQUIP2 dataspace for persistence and coordination. The three core sub-applications of the MUPE server are:

- The MUPE TcpIp manager, which handles communication with phone clients;
- The MUPE media manager, which handles serving of media files and archives to phone clients; and
- The MUPE world manager, which hosts the shared game world.

This is illustrated in figure 12 and specified in the top-level Spring configuration `applicationContext.xml`. As is typical for a WAF web application underlying persistence is provided by a relational database (typically MySQL), and the whole web application is hosted by a suitable J2EE container (typically Apache Tomcat).



**Figure 12: An overview of the MUPE/WAF integrated system**

The MUPE world manager hosts the MUPE game world and its contents, in particular Users, "Rooms", "Services" and "Items". Messages received from clients are passed to a Parser which in turn invokes corresponding methods on the game world or its

---

elements. The essence of the integration is that it allows the MUPE game server objects (World, Rooms, etc.) to access and update persistent state in the EQUIP2 dataspace. This persistent state can also be accessed concurrently by other game logic objects within the server (e.g. to interaction-independent handle timer-based game elements) and by Spring controllers and JSP-based web pages to provide dynamic browser-based interfaces to the game for authoring, testing, monitoring, non-phone game play, and so on.

A key element of the integration is that MUPE User objects (the representation of particular users or phone clients within MUPE) are always linked to corresponding UserInfo records in the WAF (EQUIP2) persistent dataspace. This allows phone-based and web-based interaction to be coordinated.

This core integration and linking of User/UserInfo objects is applicable to any game/experience constructed using MUPE, allowing use of the full Web Application Framework (and other J2EE technologies) to provide a rich web-based element to the game. The native persistence of MUPE is also quite limited, and the integration allows the WAF/EQUIP2 persistence over a commercial RDBMS to be used for all significant user and game state.

As documented elsewhere, the WAF includes a number of useful standard facilities including generic web-based form interfaces for viewing and authoring persistent state, XML and Hessian binary bulk upload and download links for database backup, transfer and analysis, and a JSP taglib for simple access to the dataspace from dynamic web pages.

The MUPE/WAF integration is available as a template combined application. As with normal MUPE application development this template application is then extended and specialised through a combination of XML configuration, XML MUPE scripts and Java extensions to create the required game logic and interface. As with normal WAF application development, any additional persistent data classes (to be stored in the dataspace) must be specified using a simple XML schema during the application development process. Additional server classes and elements of the web-site can be configured using the Spring Framework's normal mechanisms.

## REFERENCES

- [1] D7.3 Toolbox with Prototypes of the first Basic Tools, <http://iperg.sics.se/Deliverables/D7.3-Toolbox-with-Prototypes-first-Basic-Tools-Public-version.pdf> [cited 070912]
- [2] D12.5 Delivery of the Second City as Theatre prototype, <http://iperg.sics.se/Deliverables/D12.5-Delivery-of-City-as-Theatre-prototype-II.pdf> [cited 070912]
- [3] <http://www.hibernate.org> [cited 070912]
- [4] D8.7 Delivery of the second Crossmedia prototype, <http://iperg.sics.se/Deliverables/D8.7-Delivery-of-the-second-Crossmedia-prototype.pdf> [cited 070912]
- [5] D11.7 Game Design, Momentum, <http://iperg.sics.se/Deliverables/D11.7-Game-Design-Momentum.pdf> [cited 070912]
- [6] <http://www.imsystech.com/products/modules/snapmodule.htm> [cited 070824]
- [7] <http://www.opensource.org/licenses/bsd-license.php> [cited 070824]
- [8] <http://www.opensource.org/licenses/nokia.php> [cited 070824]

- [9] <http://www.corba.org> [cited 070912]
- [10] <http://www.bugzilla.org> [cited 070912]
- [11] D9.8 Game Prototypes, <http://iperg.sics.se/Deliverables/D9.8-Game-Prototypes.pdf> [cited 070912]
- [12] <http://www.sics.se/ice/projects/pimp> [cited 070905]
- [13] <http://www.sics.se/ice/projects/mogile> [cited 070812]
- [14] <http://www.sourceforge.net> [cited 070812]
- [15] <http://hessian.caucho.com/> [cited 070815]