OPEN Project

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

Title of Document:	System support for application migration
Author(s):	Anders Nickelsen, Rasmus Olsen (AAU), Miquel Martin(NEC), Holger Klus (ClU), Giuseppe Ghiani (CNR)
Affiliation(s):	AAU, NEC, CNR, ClU, Vod
Date of Document:	Dec 2008
OPEN Document:	D3.2
Distribution:	Public
Keyword List:	
Version:	Draft

OPEN Partners:

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

Title: Detailed network architecture	**Id Number: D3.2**

Abstract

This document describes in brief the implemented prototypes and demos developed as a part of WP3 implementation work carried out in the first year of OPEN. This contains three demos; one related to context triggered service migration, one with specific use of context to select the appropriate target device to migrate the service session; and one to illustrate the OPEN platforms capability to support logic reconfiguration in a game scenario. The demos are for the first year working more or less independently, while second year will search to bring the demos into an integrated OPEN platform.

Title: Detailed network architecture	**Id Number: D3.2**

1 Introduction

WP3 has a prototype deliverable due in M12 and in M23. These deliverables will illustrate the system support for application migration researched and developed in WP3. The deliverable approach is iterative meaning that the M12 deliverable will illustrate a first version of the system support and the M23 deliverable is the M12 deliverable revised, containing additions developed during year 2 of OPEN.

This document specifies details regarding the M12 prototype deliverable. The purpose of the deliverable is to illustrate WP3 functions interaction in a scenario. The following text describes which functions are in focus in the first version and also describes a scenario performed by the prototype(s).

As the overall purpose of WP3 is to research context management solutions and mobility support these are the primary functions contained in the first version:

- Context management
- Migration orchestration
- Migration trigger
- Mobility support
- Application logic reconfiguration trigger

The **mobility support and context information management prototype** will show full migration of a *simple video-streaming application* from a mobile terminal to a large-screen computer. The application is served from a non-OPEN aware video server (e.g. using VLC) and is displayed to the user on the mobile terminal. All nodes, except the user's mobile device (which has a Windows Mobile system), are running *Linux*.

When the user uses a computer with a large screen, the receiving end of the application is migration onto that computer for enriched experience by the user. The migration triggers are issued based on gathering of relevant context information. The information used for the triggers is for instance *user location and orientation/direction.* The user location can be determined using the Bluetooth signal strength between the computers and a mobile phone on the user, and detecting whether the user is oriented toward the screen can be done by analysing if the user is active on the computer or using an electronic compass as done in the **DiscoveryMap context provider prototype** described below. Once the migration is triggered, the prototype illustrates context-aware migration by using *user-location and user-profile settings* and chooses the large screen computer as target device. Throughout the migration process, the connectivity of the streaming application is assured, despite the fact that the network context of the receiving device changes from the mobile terminal to the computer.

Upon successful migration, the application illustrates context-awareness by setting the volume depending on the number of people in the room. If the spectator is

alone, the volume is set high, and if there are more people, the volume is kept low. In the first version prototype, the number of people can be simulated by entering a specific number into the system. In future versions, the number can be detected by presence of device actually in the room.

The **context aware logic reconfiguration prototype** illustrates how context information can be used to reconfigure the internals of an application in a more complex way than just setting the volume.

2 Prototype description

2.1 Mobility support and context information management

Figure 1 illustrates the flow of functions and information involved in running the described scenario. Figure 2 shows the node and network setup for the prototype.

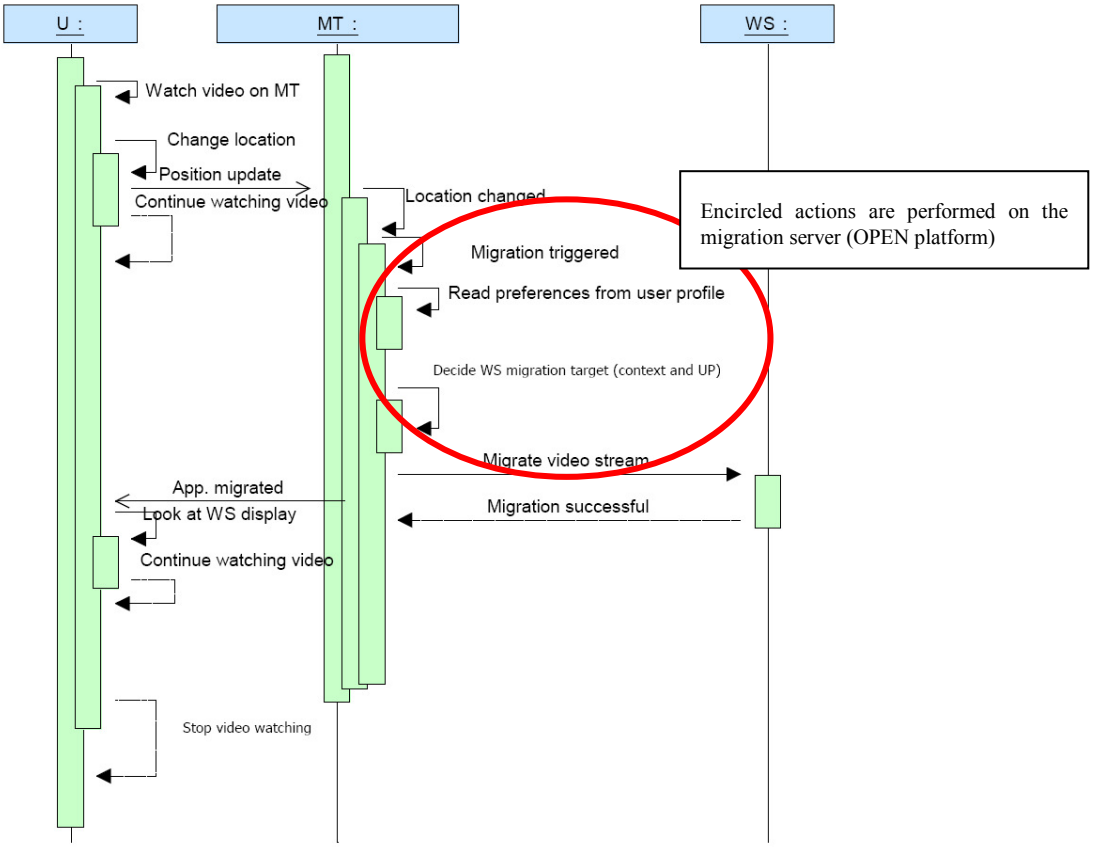

Figure 1: Sequence diagram of function flow during migration in the prototype (U=User, MT=mobile terminal, WS=workstation)

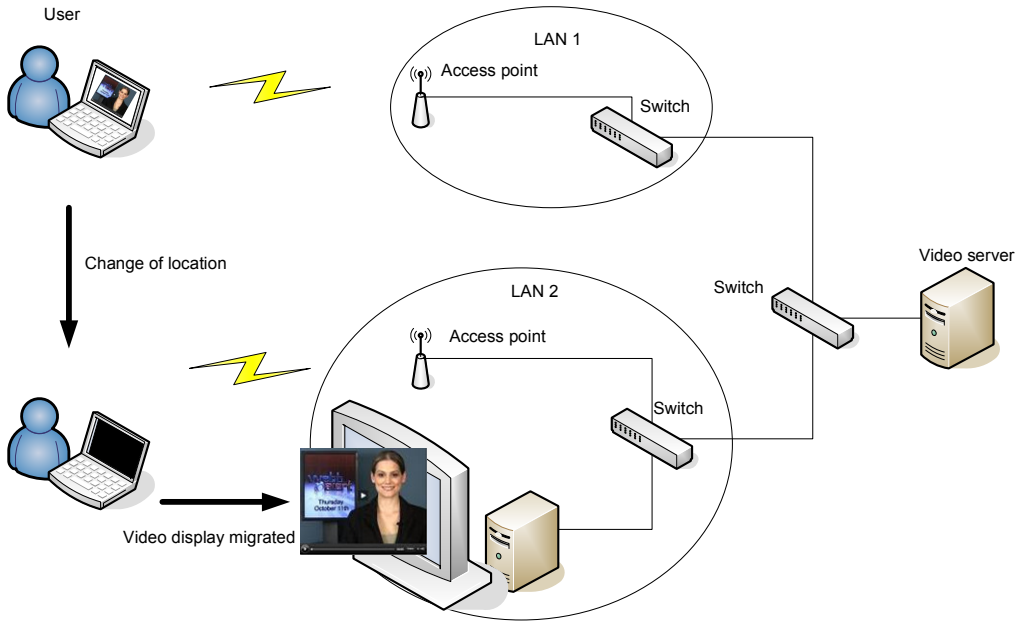

Figure 2: Architecture of prototype setup

The demonstrator demonstrates:

Context-triggered and context-aware migration of a video-streaming application using a migration server

Components active in demonstrator

- Streaming application (VLC server and client)
- Context management (SCMF, Siafu (context simulator), new providers (Bluetooth, DiscoveryMap))
- Trigger management (Simple user based, semi automatic via position change (notification from context), complex automatic algorithm)
- Migration orchestration (handles which components go where)
- Security (encrypted wireless channel, key exchange, security associations, login session, dialog pop-up on location change)
- Mobility support (application is first run from mobile device and then on workstation, hence, support for mobility is needed, forwarding, port-mapping, route reconfiguration, mobile IP, IPv6)
- State persistence logic (buffering video content…)

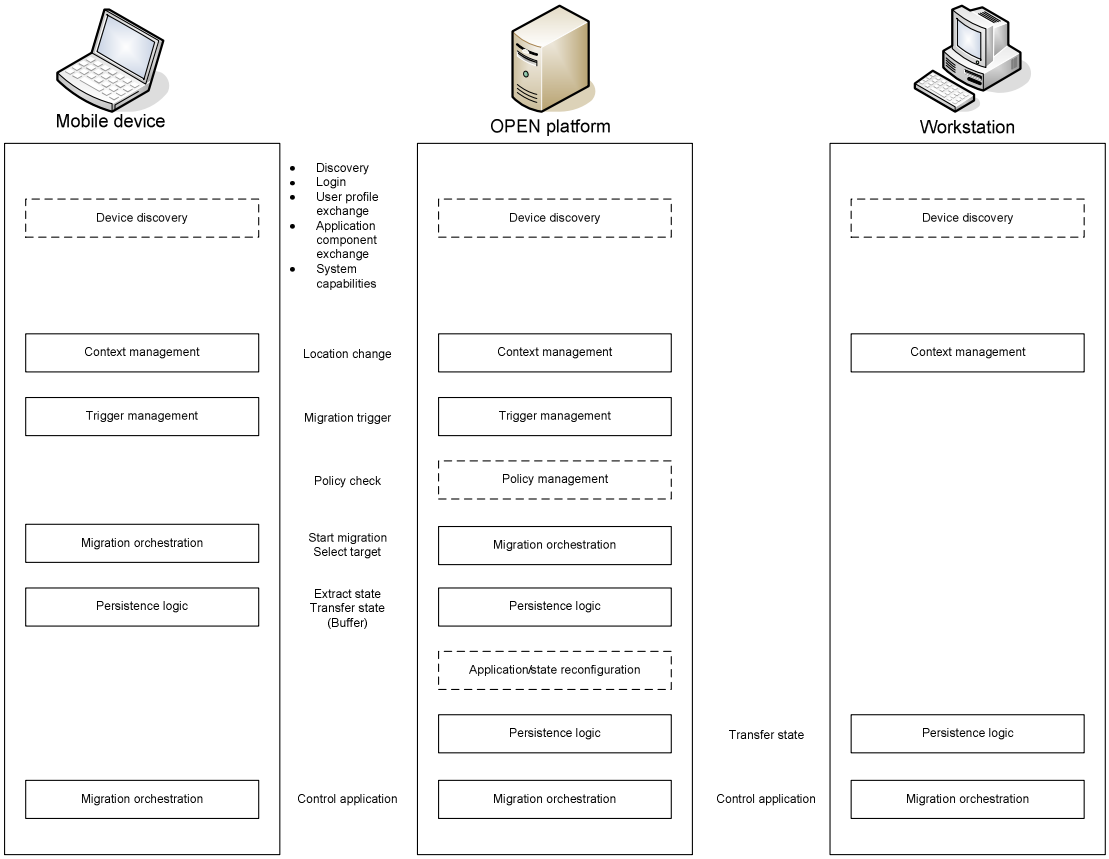

Figure 3: Illustration of which functions are used when during migration. Dashed functions are not of interest in D3.2 - but may be later on.

Milestones for development

- Bluetooth context provider (provides 'location' or 'signal strength' to SCMF)
- Mobility support (redirect stream to new target)
- Migration orchestration (control application on nodes)
- Trigger management (without persistence logic, handling manual triggers)
- Trigger management from context information (interface to SCMF, Siafu)
- Context-aware reconfiguration (resize screen, adjust volume, …)
- Framework for intelligent and automatic trigger management
- Persistence logic (hand-over of latest state)

2.2 DiscoveryMap as context provider

For the end user, the DiscoveryMap is mainly a graphical-interaction component embedded on the OPEN-client that may facilitate her in discovering the available target devices, their capabilities and their state.

From an internal point of view the DiscoveryMap is also a context-provider. It is aware of the layout of the environment and of current location/direction of the user and it raises relevant events related to user's location and/or heading: a "location changed" occurs whenever the user approaches a new tagged point; a "new device(s) pointed" event is raised only if the set of the available devices currently pointed by the user changes (i.e. when the user watches towards a new device).

What has already been done

So far, a DiscoveryMap version for Windows Mobile devices has been developed in C#. The component expects the following inputs:

- XML specification of the environment
    - Size of the area, expressed in centimeters, and absolute direction of the area, expressed in degrees (e.g., the heading of the room's edge with respect to the North).
    - Device information for all devices in the area (name, type, address, position within the area)
- ID of the tag currently detected by the localization support (we have used so far a RFID infrastructure)
- Absolute direction of the user detected by a wearable electronic compass
- State of the stationary devices, currently obtained by a device discovery protocol

The component provides the user with a basic map of the environment with respect to the deployed target devices. Each device is represented by an interactive icon which shows:

- the device type: different icons are used for different types
- the state of the device: a sign on the top-left corner of the icon indicates whether the device is active (V) or not (X).

The current area map is automatically scaled to let all the devices fit on the component widget. However, the user can zoom in or out the currently visited point in order to have a better view of the neighboring devices. For facilitating the spatial continuity when zooming, a grid of the area is also displayed.

The user's line of sight is displayed by a green line and the devices s/he is currently pointing are highlighted by green squares (see Figure 4).

Two presentation modalities are available on the DiscoveryMap:

- *Map-centered* (or *north up*): the map is fixed while user icon moves and rotates according to user's position and direction (see Figure 4 a and b);
- *User-centered* (or *heading up*): user icon is on the center of the widget while the map is translated and rotated when the user moves or changes direction (see Figure 4 c and d).

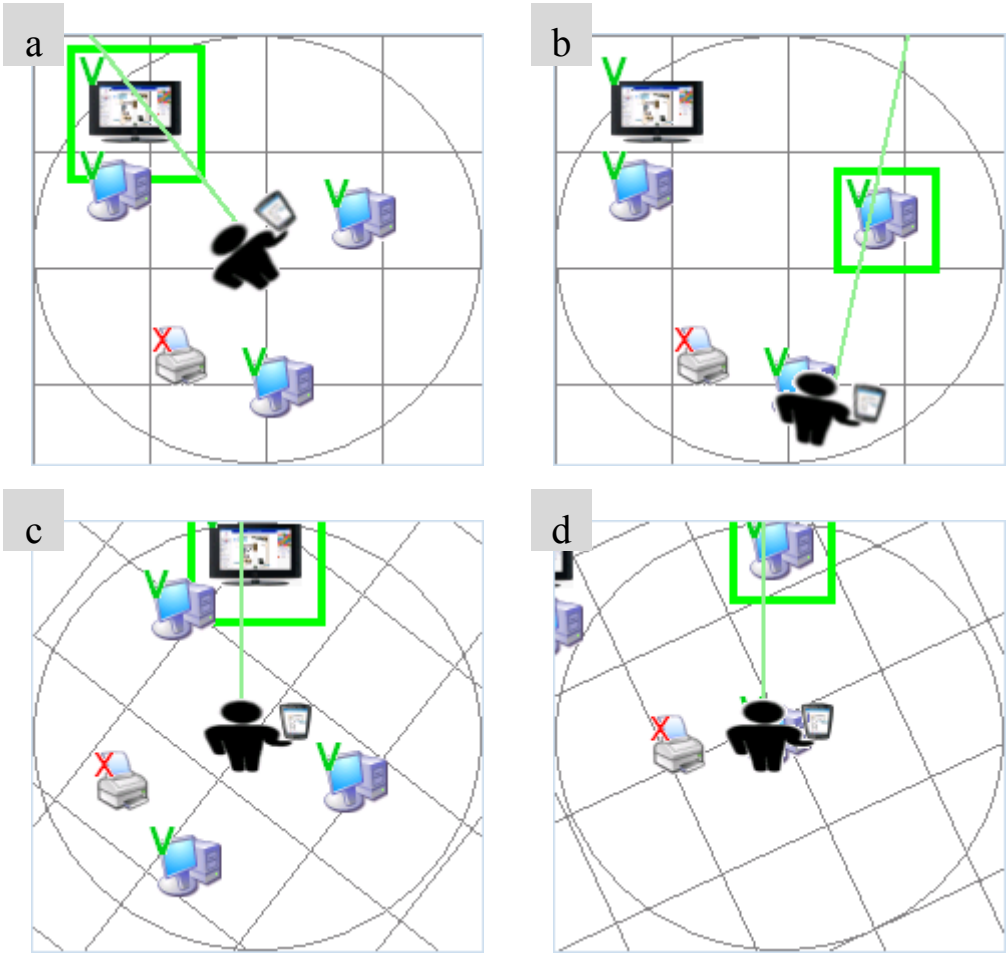

Figure 4: Map-centred (a, b) and user-centred (c, d) modes.

ISTI-CNR has performed a user test involving 11 people for evaluating the usability of the DiscoveryMap. Since just a slight preference towards the map-centered mode emerged, both modes are still available on the component (allowing the user switch to the preferred mode).

Planned work

DiscoveryMap will exploit context information provided by the SCMF such as the list of discovered devices (that, so far, comes from an embedded device discovery protocol) and other location related information (such as discovered Bluetooth beacons).

The Device Orientation parameter is also considered in the description of the target devices. Such a value represents the direction of the target device with respect to the visited area. It is fundamental to detect when the user is actually in

front of the main side of a device (e.g., at the display of a desktop PC or of a large screen).

An investigation on whether the orientation of target devices would improve the graphical presentation on the DiscoveryMap is being done (but no user test has been performed yet). Temporal features of the devices may be considered too. For example, information related to the estimated amount of time the user has to wait before accessing a busy target device. Time-related information should be available on the SCMF.

DiscoveryMap component may in the future be implemented in Java, in order to make it potentially suitable to any device.

2.3 Context aware logic reconfiguration

In this section we will describe how the work of WP3 is used for application logic reconfiguration which is considered in deliverable [3]. To illustrate that, we will introduce how the Pacman game as introduced in [2] can become context-aware.

PacMan is a game where a figure called PacMan, which is steered by the user, has to collect dots in a maze like depicted in Figure 5. 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. 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.

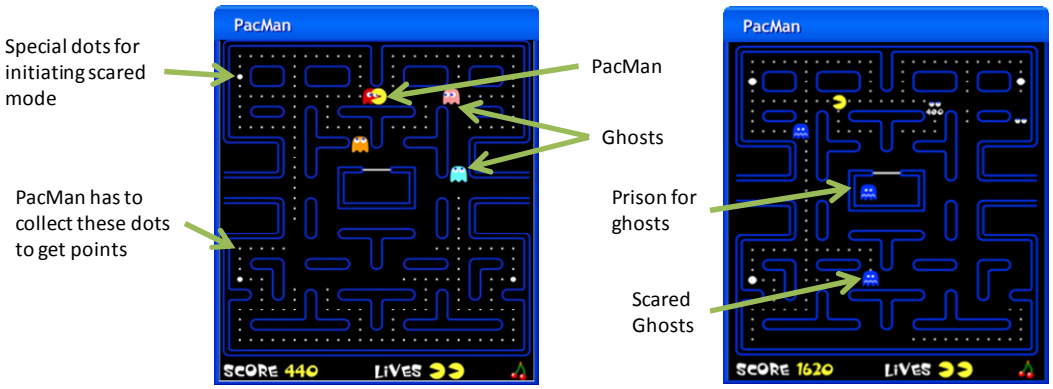

Figure 5: The two modes of a single PacMan game. On the left hand side the game is in normal mode where the ghosts try to catch the PacMan. On the right side the game is in scared mode where the PacMan can catch the ghosts.

Within OPEN we defined scenarios where migration, and therefore adaptation takes place [2]. Assume that the user starts playing Pacman on his PC at home using his keyboard to steer the PacMan. Then he remembers that he has a PDA with an accelerometer. Therefore, he wants to use the PDA to play the game. He migrates the game using the OPEN middleware from the PC to the PDA. Using

the accelerometer he now can steer the ghost by tilting the PDA to the according direction.

Already this small example shows why adaptation and the comprehension of context information are important. Because of the limited screen size and resources of his PDA, the game has to adapt to the target platform. Since there is no place to display all information at once on the PDA, some options are now no longer accessible directly, but moved into an extra dialog. Furthermore, the number of dots within the maze is less than on the PC because of limited screen size. Therefore, the game now has to adapt in a way that the player gets more points for collecting a single dot. As it is more difficult to steer the PacMan on such a small screen, the speed of the ghosts is also reduced. Depending on the current game level, the artificial intelligence of the ghost is also adapted. All this shows the importance of considering context information during adaptation. More adaptation scenarios are described in more detail in [3] (Solutions for Application Logic Reconfiguration) and [4] (Prototype for Application Logic Reconfiguration).

First prototypes of the Pacman game have already been developed with focus on different aspects of the game. We structured the game into several components which communicate with each other through services like depicted in Figure 6.

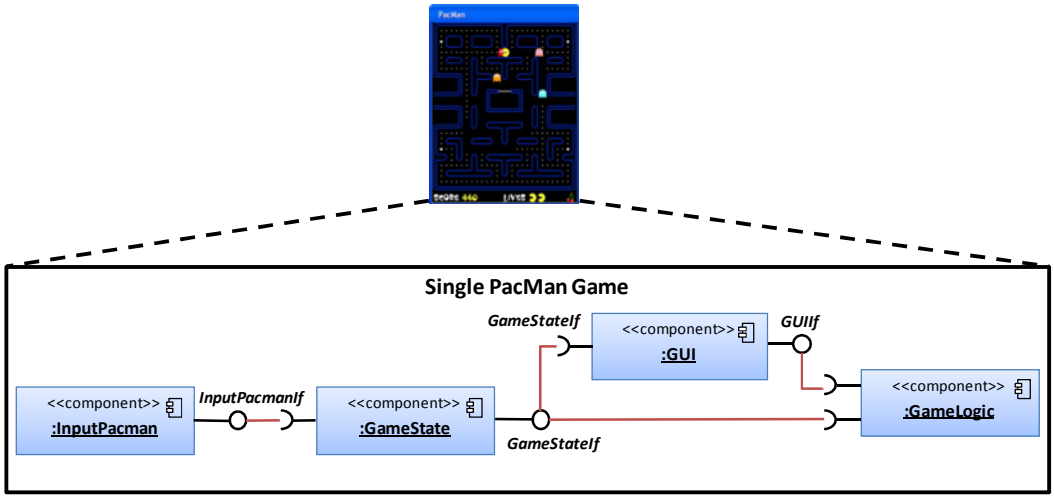

Figure 6: The internal structure of a single PacMan game.

We divided the game into four components, namely *InputPacman*, *GameState*, *GUI*, and *GameLogic*. This structure is derived from the MVC pattern (Model, View, Controller) [1] where the *GUI* and *InputPacman* represent the *View*, the *GameState* represents the *Model*, and the *GameLogic* represents the *Controller*.

However, most applications are built out of components and services which have to interact. Therefore, the Pacman example is only one example for such a system. The division into components enables us to migrate the whole application but also parts of it. So far we used two basic technologies to realize this application,

namely CORBA (Common Object Request Broker Architecture) [5] and Web Services.

Using CORBA, the application has been running satisfactorily with good performance. However, to write a CORBA component, the developer has to specify the interfaces in IDL (Interface Definition Language), generate stubs and skeletons, and he has to start the CORBA infrastructure, a quite heavyweight system. Therefore, it is difficult to integrate devices like PDAs or mobile phones. The advantage is that components can be written in different programming languages like Java or C++ and that the communication has good performance. Furthermore, there are already several infrastructure services available like a Naming Service and Event-based Communication.

On the other hand it has been found that the implementation with Web Services is not appropriate because of a lack of performance. The communication overhead is too large to realize a distributed application like this game with these components. However, there are other application domains where the usage of Web Service is more appropriate like for the orchestration of different independent services. Therefore, we developed a multiplayer Pacman scenario where different autonomic Pacman games are orchestrated to a multiplayer Pacman game.

For both types of realizations we also developed and evaluated concepts for application adaptation which are described in [3] and [4].

There are several touching points between WP3 and WP4 which we will now introduce roughly. First, context information like screen size, location of user and devices, network bandwidth and so on will be provided by WP3 and used in WP4 among other for application logic reconfiguration, but also for migration and the adaptation of the user interfaces which is considered in WP2. A second main point of collaboration is the realization of the communication between application services. As described before, both so far evaluated technologies have their advantages and drawbacks. Thus the network infrastructure is still a main issue where solutions are developed in WP3. Therefore, the goal will be to find a network infrastructure, which is easy to use for service developers, enables communication with high performance, and is able to integrate resource-constrained devices. Finally, it should enable the adaptation of services during migration or on change of context information.

3 Future work and next steps

The demos for the first year did not aim specifically on an integrated implementation, but on showing different core aspects of the migration concept, keeping the implementation focused and relative simple, e.g. we assume a migration server is present and do not implement all functionalities envisioned. The demos therefore support each other to show the benefits of using the OPEN platform. The next steps which will be the focus of year two of the project is first to remove the assumption of a migration server thereby enabling service migration in ad hoc scenarios, and second to provide the full integrated platform and show this is working and capable of supporting the migration of a variety of services (and in particular those developed in other work packages).

4 References

[1] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stahl. „A System of Patterns“, 1996.

[2] OPEN Deliverable D1.1, “Requirements for the OPEN Service Platform”, 2008.

[3] OPEN Deliverable D4.1 „Solutions for Application Logic Reconfiguration“, 2009.

[4] OPEN Deliverable D4.3 “Prototype for Application Logic Reconfiguration”, 2009.

[5] Object Management Group (OMG). “Common Object Request Broker Architecture: Core Specification”, 2004. Online available at: http://www.omg.org/docs/formal/04-03-12.pdf