IST Amigo Project
Deliverable D7.4

Implementation of the Extended Home Environment Prototype

IST-2004-004182
Public

Information Society Technologies

Project Number	:	IST-004182
Project Title	:	Amigo
Deliverable Type	:	Software

Deliverable Number	:	D7.4
Title of Deliverable	:	Implementation of the Extended Home Environment prototype
Nature of Deliverable	:	Public
Internal Document Number	:	amigo_d7.4_final
Contractual Delivery Date	:	31-12-2007
Actual Delivery Date	:	06-01-2008
Contributing WPs	:	WP7
Author(s)	:	Stan Borkowski (FT), *editor*
		Anne Gerodolle, Thibaud Flury, Gilles Privat (FT)
		Eugenio Rizzi, Katia Bianciotto (ITAL)
		Michael de Regt (Philips Design)
		Christian Quaedvlieg, Leo Rozendaal (Philips CE)
		Remco Poortinga, Dirk-Jan van Dijk (TELIN)
		Jörg Schmalenströer, Maik Bevermeier, Volker Leutnant (PAE)
		Max Larsson, Thomas Dexheimer (SIT)

Abstract

This document describes the results of WP7. It presents applications developed for the "Extended Home" scenario as well as components, building blocks and basic services that compose the applications. It provides a description of the architecture of these applications, together with installation and deployment instructions, and user manuals for the presented software.

Keyword list

Ambient Communication, Distributed Interface Devices, Shared Activities, Shared Picture Browsing, Shared Gaming, Tangible Interface Devices, Audiovisual Communication, Presence Management, Presence detection

Table of Contents

Table of figures

1 Introduction

This document describes the results of WP7. It presents applications developed for the "Extended Home" environment as well as components, building blocks and basic services that compose these applications. It provides a description of the architecture of these applications, together with installation and deployment instructions, and user manuals for end-users.

The objectives of WP7 were twofold: on the one hand we aimed at validating the Amigo middleware and services from WP3 and WP4 by integrating them into application prototypes, on the other hand, we explored scenarios and novel user interaction concepts for the "extended home" domain, complementing other application domains addressed by WP5 and WP6.

The three application work packages (WP5, 6, 7) provided WP3 and WP4 with both requirements for new services or middleware features, and with feedback on problems encountered while using the Amigo software. This was a very important role, as it led to the improvement of the Amigo software quality. The design of WP7 applications was done partly in parallel with the development of the middleware and Amigo services. In consequence, we did not exploit all middleware features that are available in the end. Nevertheless, we extensively use the WP3 development and deployment frameworks and the WP4 context management system. We have also reported to our colleagues from WP3 and WP4 a number of implementation issues that hence had been corrected in the middleware. With the Activity Sharing application, we also illustrated how applications developed using conventional technologies can be integrated in an Amigo enabled environment.

The design of applications targeted by WP7 was focused on providing support for a shared feeling of presence rooted in the environment and activities of persons, between one's home and other distant environments such as homes of friends, relatives, or the workplace. The aim was thus to extend the home environment for both *interpersonal communication* and *shared activities*, using for this the generic Amigo platform (Middleware and Intelligent User Services). All WP7 applications focus on one of the two sub-domains; the Ambience Sharing, Social Radio and parts of the Feeling@ applications are oriented towards new forms of interpersonal communication, whereas the Activity Sharing, Board Game and Feeling@ sketch sharing applications provide means for sharing activities with remote people. We also present special interaction devices such as the Personal Amigo Device allowing the user to transparently and securely access home services while visiting other Amigo domains, and the Awareness Globe that can show the status of user's buddies and allows launching Amigo applications.

The WP7 demonstrator applications are context-aware, in particular they take into account user presence and user location in the environment. This form of context-awareness allows for dynamic adaptation to context changes and is a very good illustration of the power of the Amigo middleware. We have designed a service oriented architecture in which resource managers, arbitrating access to shared interaction resources, and an application scheduler play a pivotal role, offering a flexible solution leveraging the Amigo middleware and services for dynamic context adaptation in distributed communication applications.

The WP7 applications have been installed and deployed at three locations; Philips' HomeLab, Ital Design's IDGLab, and France Telecom's VisionLab. The most mature prototype elements will be evaluated by end-users. This work will be done in WP8 and is out of the scope of this deliverable.

This document presents the WP7 software in a bottom-up manner. In chapter 2, basic building blocks are presented. Each component is briefly described, and development status is reported. In chapter 3, we provide installation and deployment instructions. Chapter 4 focuses on interactions between the presented building blocks, and chapter 5 describes in more detail how the applications can be used in a home environment.

2 Components Overview

In this chapter, we describe components, building blocks, basic services and application parts developed for the "Extended Home" demonstrator. We provide a brief description of the software's role, its development status and location were it can be downloaded. Installation instructions and more detailed description of the components' interfaces are presented in following chapters.

2.1 Scheduler

Provider

FT

Introduction

The Scheduler is an Amigo service that coordinates the execution of applications in a centralized manner. It is needed because in WP7 applications share an environment but are agnostic about each other. Also not all applications are compositions of web-services, so the WP3 service composition framework cannot be applied. In consequence, a scheduler is needed for coordination. All WP7 applications must negotiate with the Scheduler service the permission to start and must notify it when execution starts and ends. The Scheduler arbitrates execution requests and is authorized to stop running applications if needed.

Development status

Development is completed, but it implements only a simple authorization policy allowing a single application to execute at a time.

Intended audience

Developers

License

This software is proprietary (Copyright France Telecom). Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: Any Java 5 enabled hardware with IP network connectivity.

Software: Java Runtime Environment 5 or higher, Amigo middleware and bundles.

Platform

Java virtual machine, OSGi

Tools

Java development kit 1.5 or higher, Maven2

Files

All files, software, and bundles can be found in the gforge repository in:

[amigo] / WP7 / Scheduler

Documents

-

Tasks

-

Bugs

No known bugs

Patches

N/A

2.2 Resource Manager

Provider

FT

Introduction

WP7 demonstrator is composed of a number of applications coexisting in the same environment. Some of these applications are composed of services that run on the same PC and might share interaction resources and access them concurrently (e.g. a display screen). In order to arbitrate concurrent access to resources, we developed a service called ResourceManager.

The ResourceManager represents the resource in a form of a web-service to which services must register and with which the services must negotiate prior to accessing the resource. The Resource manager implements a context source that exposes all services registered to the resource manager, the location of the resource manager (and implicitly that of the services), and the status (availability) of the services. The ResourceManager facilitates thus the setup of the Amigo-based system, as services that register to the manager do not need to implement a context source for publishing their status and location.

The ResourceManager also subscribes to WP4::LMS for user location and position data. Based on the location of the user and the resource, the resource manager calculates the distance from the user and maps it into one of five distance-dependent interaction zones:

- `outofRange` corresponds to a distance from which the interaction is impossible
- `ambientRange` is a zone within which the user can probably notice the interaction resource, but is to far to interact with it
- `notificationRange` is a zone where the user can interact in a subtle way, but is to far to perform full range of explicit interactions
- `interactionRange` is a distance at which the user can fully interact with the resource.
- `unknownRange` used when the resource is unable to determine the distance

The mapping of relative distance to the interaction zones is interaction resource dependent. In other words a resource manager for a wall-size touch display should have different zone settings than one for a 15 inch PC display. Note that concept is not limited to visual interaction resources and can be applied to other modalities (e.g. audio devices). The resource manager is equivalent to self conscience of the represented interaction device.

Services such as the Palantir can subscribe to the ResourceManager's CS for changes in user relative distance. The resource manager's CS provides information such as: user1 is in ambientRange of serviceA, serviceB and serviceC that are available.

Development status

Development is completed.

Intended audience

Developers

License

This software is proprietary (Copyright France Telecom). Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: Any Java 5 enabled hardware with IP network connectivity.

Software: Java Runtime Environment 5 or higher, Amigo middleware and bundles.

Platform

Java virtual machine, OSGi

Tools

Java development kit 1.5 or higher, Maven2

Files

All files, software, and bundles can be found in the gforge repository in:

[amigo] / WP7 / ResourceManager

Documents

-

Tasks

-

Bugs

No known bugs

Patches

N/A

2.3 Palantir Presence Management System

2.3.1 Palantir service

Provider

France Telecom (FT)

Introduction

The Palantir is an Amigo service automatically estimating the level of user presence and availability. The aim of the Palantir service is to indicate user status for ambient communication and activity sharing applications on the basis of contextual information relative to persons who are potentially involved. In order to subscribe for user status change events, an application must define a Presence Model in which it specifies requirements of interaction resources that are necessary to perform its tasks. When user context satisfies the defined requirements, the application is notified. The Palantir can also transfer events explicitly issued by users willing to enter in communication using a particular application (see Figure 1).

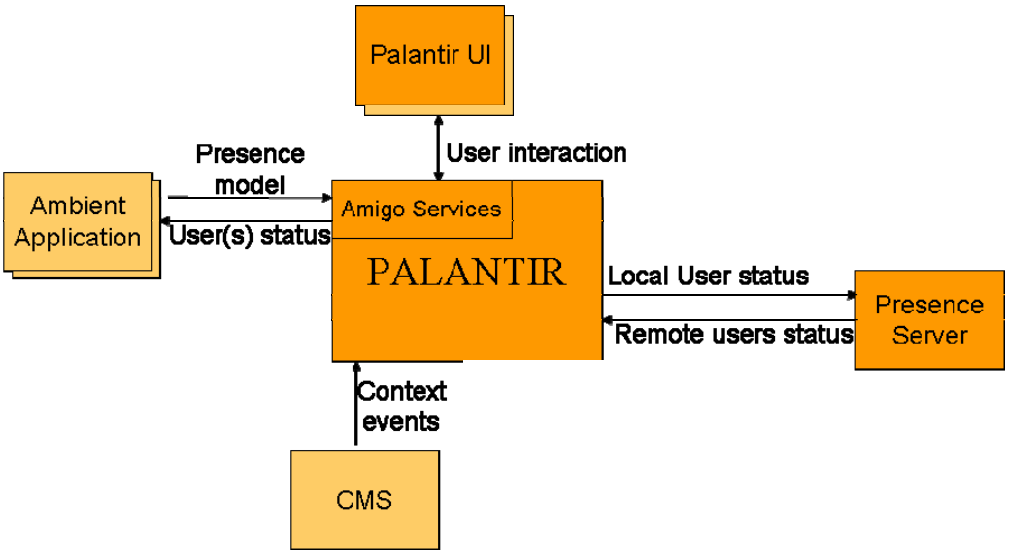

Figure 1: Palantir service in an Amigo environment

Presence for ambient communication can be considered as a multidimensional piece of context information characterizing to what extent the user is ready to communicate. Willingness to communicate, user attention, availability, intentness with respect to the communication, availability of communication and interaction resources, can all be considered as components of this multidimensional notion of user presence. The Palantir restricts the concept of presence to a two-dimensional presence model that including user ability to communicate and user "awareness".

The ability to communicate depends on the existence of interaction resources needed for communication. It is defined over four states corresponding to the following situations:

1. The user is unable to communicate if there are no interaction resources that could support the communication.
2. The user can receive messages or notifications, but there are no resources allowing her/him to respond immediately.
3. The user can send messages or notifications, but no resources can support the interlocutor's response.
4. The user is in proximity of sufficient resources to both send and receive communications.

User ability to communicate is evaluated each time the user moves to a different area, e.g. to a room. On such event, the Palantir performs a location-aware service discovery to obtain a list of services available in the area. This list is then compared with the list of services required by each application and ability level is set accordingly.

User awareness is defined on a linear scale from 0 to 100. It is meant to capture the user's willingness to communicate, i.e. awareness of 0 means that the user does not want to communicate, 1 corresponds to an urgent need to communicate. As a "state of mind" of the user, this willingness to communicate is of course impossible to reflect accurately on the basis of a purely automatic detection. It may however be partially inferred or modulated on the basis of such objective clues as the user being engaged in other activities than communication, or getting closer to devices that afford communication resources. The Palantir modulate the awareness level when the user approaches or moves away from a device hosting services required by an application. However the awareness is meant to be set primarily by the user, and any estimated variation must be validated by the user.

Development status

The Palantir software development is completed. In its current version the Palanir is able to detect user presence based on context information and transfer status levels set by applications or by users.

Intended audience

Developers of ambient intelligence applications.

License

This software is proprietary (Copyright France Telecom). Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: Any Java 5 enabled hardware with IP network connectivity.

Software: Java Runtime Environment 1.5 or higher, Amigo middleware and bundles. XMPP/Jabber Server (we recommend eJabberd, a simple open source xmpp server)

Platform

Java virtual machine, OSGi

Tools

Java development kit 1.5 or higher.

Files

All files, software, and bundles can be found in the amigo Palantir repository at: http://amigo.gforge.inria.fr/obr/palantir/

http://amigo.gforge.inria.fr/obr/palantir/repository/

http://amigo.gforge.inria.fr/obr/palantir/oscar_Palantir.zip

The XML schemas defining the extended presence protocol, the presence model, and presence digest are available at:

http://amigo.gforge.inria.fr/obr/palantir/resources/schemas/presence_status.xsd

http://amigo.gforge.inria.fr/obr/palantir/resources/schemas/presence_model.xsd

http://amigo.gforge.inria.fr/obr/palantir/resources/schemas/user_digest.xsd

Documents

D7.1, D7.2, D7.3

Palantir online tutorial: http://amigo.gforge.inria.fr/obr/palantir/HowTo-Palantir.htm

Tasks

-

Bugs

A few warn or error log traces may appear but this does not affect the runtime comportment of the service.

Beware of network configuration (DNS name resolution, multicast packet, and such). Bad network configuration can cause fatal or blocking errors for the Palantir core bundles.

Patches

N/A

2.3.2 Palantir GUI

Provider

France Telecom

Introduction

The PalantirGUI service allows a user to visualize information computed by the Palantir service, i.e. the awareness and communication abilities of his/her buddies. It also offers means to control the information sent to buddies, i.e. the presence level described by user's ability to communicate and the awareness.

Awareness and communication abilities are computed by the Palantir service according to context information and are sent to all registered PalantirGUIs. The PalantirGUI implements a user interface that shows presence of buddies as a patchwork of pictures with varying size and color intensity. The more "aware" a buddy, the bigger his/her picture. Buddy's communication abilities are reflected by the color intensity of their pictures (see Figure 2). The interaction with the PalantirGUI is described in more detail in section 5.1.

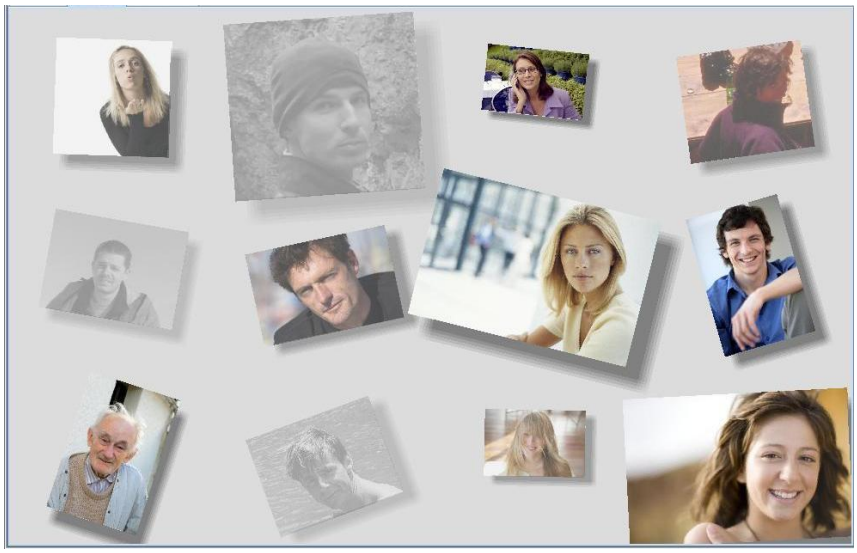

Figure 2 Picture patchwork showing status of remote buddies

Development status

The development of the PalantirGUI service is completed. The service implements:

- service declaration using WP3::OSGi framework
- interaction with the Palantir service
- displaying the buddies, controlling user's state, notifying buddies

Intended audience

End-users, administrators, developers of user presence aware applications.

License

This software is proprietary (Copyright France Telecom). It will not be made open source since it is not middleware software. The Palantir GUI bundle (binary code) can be freely downloaded, installed, and used for test and demonstration purposes. Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: PC with a display, mouse or other pointing device

Software: Java J2SE run time, OSGi R3 platform, Amigo middleware and bundles.

Files

The Palantir GUI bundle is downloadable / installable from the Amigo Palantir bundle repository:

http://amigo.gforge.inria.fr/obr/Palantir/repository/repository.xml

Documents

D7.3

Palantir online tutorial: http://amigo.gforge.inria.fr/obr/palantir/HowTo-Palantir.htm

Tasks

-

Bugs

No bugs detected so far.

Patches

N/A

2.3.3 Palantir composers

Provider

France Telecom (FT)

Introduction

While the Palantir is a web-service that can be used by other applications to obtain user status, it can be also combined with visualization and control services (e.g. PalantirGUI). The composition of the Palantir and a visualization service allows the user to monitor presence status of remote buddies and to control her/his own status.

We developed several services implementing different methods for composing the Palantir with visualization services. The palantir_gui_composer allows the user (administrator) to connect a Palantir service and a PalantirGUI by selecting them from a list of discovered services. The palantir_standalone_composer automatically connects the first found matching services. Finally, the Palantir CMS composer dynamically connects a Palantir service with a PalantirGUI service that is most suitable for the user (end-user) based on user and services location.

The CMS composer subscribes to WP4::CMS for user location events, service location and service availability data. These data can be provided by the ResourceManager's context source. When a user approaches a display screen running the PalantirGUI bundle, the composer binds the GUI service with the Palantir service. The user is free to move in the environment and the PalantirGUI will appear on the closest display and constantly show the status of user's buddies.

The standalone and the gui composer can be used for testing purposes or for simple setups. The CMS composer is used in the WP7 demonstrator to show dynamic context-aware service composition.

Development status

The development of Palantir composers is completed. All composers supports:

- discovery of Palantir and PalantirGUI services through WP3::ws-discovery
- binding of a Palantir service to a PalantirGUI service

The Palantir CMS composer also supports:

- subscription to WP4::CMS for user relative location, service location and service availability data
- dynamic binding and unbinding of Palantir services

Intended audience

Administrators, developers of user presence aware applications.

License

This software is proprietary (Copyright France Telecom). Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: Any Java 5 enabled hardware with IP network connectivity.

Software: Java Runtime Environment 1.5 or higher, Amigo middleware and bundles.

Platform

Java virtual machine, OSGi

Tools

Java development kit 1.5 or higher.

Files

All files, software, and bundles can be found in the amigo Palantir repository at: http://amigo.gforge.inria.fr/obr/palantir/

http://amigo.gforge.inria.fr/obr/palantir/repository/

Documents

Palantir composers are described in the online tutorial:

http://amigo.gforge.inria.fr/obr/palantir/HowTo-Palantir.htm

Tasks

-

Bugs

-

Patches

N/A

2.3.4 Palantir UI test

Provider

France Telecom

Introduction

The Palantir UI test allows a developer to test a Palantir UI without having to set up a complete Palantir system (xmpp server etc).

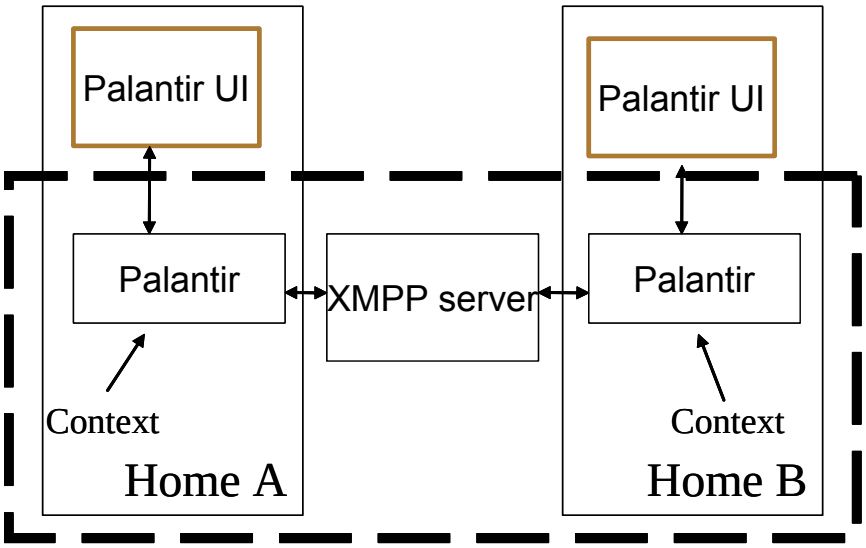

Figure 3: Standard configuration of the Palantir. Two (or more) Palantirs communicate through an XMPP server, obtain context information and communicate with the user through one or more Palantir UIs. The Palantir UI test "replaces" the outlined set of compounds

In a standard configuration, awareness and communication abilities are computed by the Palantir system according to context information and given all subscribed services (e.g. PalantirGUI service). Interaction between a Palantir UI and a Palantir are done through Web services (palantir_interaction_ws bundle).

A graphical UI for the Palatir service is presented in section 2.3.2. This test bundle allows the developer of alternate (not necessarily graphical) Palantir UIs to do tests without setting up a complete Palantir system.

Development status

The development of the Palantir UI test is completed. In this version it supports:

- discovery through WP3::ws-discovery
- connection to a Palantir
- initialization of the digest
- communication between two users, each using a different Palantir UI

Intended audience

Developers of user presence aware applications and testers of Palantir UIs.

License

This software is proprietary (Copyright France Telecom). The Palantir UI test bundle (binary code) can be freely downloaded, installed, and used for test and demonstration purposes.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: PC with screen, mouse or pointing system

Software: Java J2SE run time, OSGi R3 platform, Amigo middleware and bundles.

Files

The Palantir UI test bundle is downloadable / installable from the Amigo Palantir bundle repository: http://amigo.gforge.inria.fr/obr/Palantir/repository/repository.xml

Two bundles must be installed: interaction_ws.jar (containing Java interfaces common to the GUI and the interaction module) and Palantir_gui_test.jar.

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.4 Ambience Sharing Application Components

2.4.1 Ambiance Sharing application composer

Provider

France Telecom (FT)

Introduction

The Ambience Sharing application is a context-adaptive extension of traditional person to person visual communication services such as videoconference. It offers a quasi-permanent communication channel supporting smooth switching between different communication modes, covering a continuous spectrum between non-communication and full communication. Ambience sharing adapts to activities and recedes into the background as much as possible. By using unobtrusive communication modes and implicit context-adaptive interaction it avoids the saturation of the users' attention and helps them to manage the range of services offered by ambient intelligence environments.

The Ambiance Sharing Application composer module (AmSharingApp) is an Amigo service implementing dynamic service composition for audiovisual communication. It is responsible for user location-driven video redirection and for interacting with the WP7::SAInt audio system and the WP7::Scheduler. The AmSharingApp is implemented using the WP3::OSGi development framework and uses WP4::CMS to observe user location.

Development status

Development is complete. The AmSharingApp service implements:

- discovery through WP3::ws-discovery of the Palantir service and of the AVT video streaming/decoding services
- subscription to WP7::Palantir for user status change events
- interaction with WP7::SAInt audio system
- subscription to WP4::CMS for user relative location, service location and service availability data
- SIP-based dynamic redirection of video streams to the device closest to the user

Intended audience

Developers of ambient intelligence applications, end-users.

License

This software is proprietary (Copyright France Telecom). Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: Any Java 6 enabled hardware with IP network connectivity.

Software: Java Runtime Environment 1.6 or higher, Amigo middleware and bundles, and a SIP proxy server (tested with Brekeke and openser)

Platform

Java virtual machine, OSGi

Tools

Java development kit 1.5 or higher.

Files

The software is available in the gforge repository at:

[amigo] / WP7 / AmbienceSharing / AmSharingApp

Documents

D7.1

D7.2

Tasks

-

Bugs

ITAL reported that video redirection does not work properly with Asterisk SIP server.

Patches

N/A

Adaptive Video Transmission module

Provider

France Telecom (FT)

Introduction

The Adaptive Video Transmission (AVT) module is the core element of the Ambience Sharing application. It is video-conference-like software implementing various image processing techniques to offer a set of filters allowing the adaptation of the visual communication channel to user context. In order to ensure high performance all filters are implemented to run on the Graphics Processing Unit (GPU) of a PC. The software integrates face detection and face distance estimation to detect user presence and proximity to the interaction device. The proximity estimates influence the transparency of users in the streamed video.

Audio transmission is handled by the SAINT system developed by PAE.

Development status

Development is completed. Adaptive Video Transmission module supports:

- Real time video encoding and streaming
- Information hiding by making foreground objects transparent in the video

Intended audience

Developers of ambient intelligence applications.

License

Adaptive Video Transmission module is developed on top of eConf framework, which is proprietary software of France Telecom. Licenses will be granted to Amigo partners for the duration of the project.

Language

C++

Environment (set-up) info needed if you want to run this sw (service)

For a basic setup to test the module on a single PC, the following hardware is needed:

- PC equipped with a graphics card supporting OpenGL 2.0
- Webcam (tested with Logitech Quickcam 4000 Pro)

Software:

- Windows XP, SP2
- OpenlGL 2.0 or higher
- Glut library
- GLEW 1.3 or higher

Platform

No particular platform is needed

Tools

eConf Framework – AV encoding and streaming software

Windows: VisualStudio 6.0 or higher

Files

The binaries are available in the gforge repository at:

[amigo] / WP7 / AmbienceSharing / AVT

Documents

D7.1, D7.2, D7.3

Tasks

-

Bugs

No known bugs at the time being.

Patches

N/A

2.4.2 AmigoVisioService

Provider

France Telecom (FT)

Introduction

The AmigoVisioService is an Amigo proxy for the AVT software (described above). Its role is to inform the ResourceManger (see section 2.2) about the availability of the video streaming/decoding software (AVT). It is also used by the AmSharignApp service to discover the instances of AVT deployed in the environment.

Development status

Development is complete. The AmigoVisioService service implements:

- discovery through WP3::ws-discovery of the ResourceManager service

Intended audience

Developers of ambient intelligence applications.

License

This software is proprietary (Copyright France Telecom). Special arrangements can be made with Amigo partners.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: Any Java 5 enabled hardware with IP network connectivity.

Software: Java Runtime Environment 1.5 or higher, Amigo middleware and bundles, WP7::ResourceManager service deployed on the same PC

Platform

Java virtual machine, OSGi

Tools

Java development kit 1.5 or higher.

Files

The software is available in the gforge repository at:

[amigo] / WP7 / AmbienceSharing / AmigoVisioService

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.5 Picture sharing and game sharing components

Provider

PHILIPS CE INNOVATION LAB

Introduction

The Amigo "Activity Sharing" module is a TV specific GUI, providing the user the ability to communicate with friends and family. These applications can be considered as a means to enforce social relations between remote people. Once a connection has been established, live audio and video is part of the interface. The "Activity Sharing" is part of the normal TV menu, where online communities are displayed, and where other community members are displayed with additional status information (online / offline / busy). After selecting a person to communicate with, the user is enabled to select different activities to start; e.g. Picture Apart Together (shared photo browsing) or a Gaming Apart Together (a shared lean-back game) (4-in-a-row). Both of them are designed for interaction with a standard remote control on a TV screen.

Activity Sharing components/functionality:

- **Launch Poker**: From within the EasyLogic[1] menu the Poker game (see section 5.4) can be launched directly. Entering the menu item "home content" automatically triggers CMS to search for Amigo devices that are registered, and lists them in the EasyLogic interface. Each Amigo service will be found automatically and summed as menu items within Activity Sharing. Hitting the Poker Game will launch this application in an "iframe"[2], displaying it full screen.
- **Launch Palantir**: The same way as described in the previous bullet-point can be applied to start the Palantir application.
- **Presence status**: Changes in presence status is shared amongst all Amigo applications, by making use of the same shared XMPP server. Activity Sharing uses a shared SQL server for both SIP and XMPP, instead of standard (internal) SIP/XMPP user/roster management. Nevertheless, presence status changes that are triggered by e.g. Palantir will affect availability indications within the EasyLogic interface, showing users either online/offline/busy.

[1] EasyLogic is currently the name for Philips' standard TV menu UI

[2] IFrame (from *inline frame*) is an HTML element which makes it possible to embed another HTML document inside the main document.

- **CHESS**: Community CE-HTML based Experience Sharing Service is used to communicate messages between clients. CHESS is used to coordinate co-browsing (see next bullets), play a game together (e.g. notify when a user switch turn), as well as Picture Sharing. Also 'invite' messages are sent through CHESS, using Jabber functionality to communicate.
- **User management**: We implemented a framework that makes it possible to have a more realistic demo in which users can be dynamically added/deleted, and where friends are actually able to invite each other to their (own) community. The reason for this is that the previous version of the demo had "hard-coded" users, where user management was not yet possible. This resulted in the development of a (RESTful) webservice (build with Ruby on Rails) that manages the user accounts dynamically. The webservice acts as an interface to the shared Jabber and SIP (MySQL) database. Demo clients can now dynamically manage user accounts via AJAX calls to the web service making it a truly interactive demo; from the CE-HTML scripting language we are now able to talk to underlying databases.
- **Friend codes**: In order to be able to add friends to you own community/friends list, the Activity Sharing environment introduces the concept of *friend codes*. In this already wide spread manner to enable/create uniqueness to user accounts, and the possibility to distribute the user account over a selected group of people, each user is provided a unique ID# during account creation. To add a friend to your community, this number is then used as simplified input in the TV, leaving the system to do the actual insertion into the database etc.
- **Picture management**: The same framework as mentioned under User Management has been reused in the Picture Apart Together section of the application. It enables the system with a user-centric picture sharing environment. Pictures can be added (from e.g. a USB card reader), removed, and distributed over different users and folders.
- **On-screen keyboard**: An interactive CE-HTML keyboard has been developed, controllable with the regular RC, using either arrow keys, or multi-tab functionality.
- **Image uploader**: After insertion of a USB storage device (USB memory stick / USB card reader / etc) the Activity Sharing application is automatically notified, providing the user the opportunity to upload the images to a location (folder) specified by the user. These images in turn can be shared with your friends/community.
- **Co-browsing**: At the point where two clients are connected, either through the Shared Photo Browsing, or the Sharing Game activity, navigation on both screens is synchronized. This master-master mechanism (instead of master-slave) enables both interfaces to be similar on both screen, no matter which user navigates where. Through A/V communication users can agree on who has "the lead", and who guides the other party.

Development status

The interface for the "Activity Sharing" application is completed, as well as the services within the application (shared photo browsing and shared gaming).

Intended audience

End users.

License

This software is proprietary software by PHILIPS. It will not be made available as open source. Note that the use of this software is restricted to the scope of the Amigo project.

Languages

- GUI: CE-HTML + JavaScript
- AJAX calls to mongrel services, talking to Ruby on rails
- Shared SQL database for SIP/Jabber
- Video splitter: C++

Environment (set-up) info needed if you want to run this sw (service)

Minimal hardware requirements:

- One server, holding SIP and XMPP server software.
- Two client desktop computers, holding client application.
- Two webcams (incl. microphone), connected to the client desktop computers
- Two RC receivers, connected to the desktop computers (e.g. RedRat, see http://www.redrat.co.uk)
- Two widescreen TV's, connected through VGA cable to the client desktop computers. The application generates 1280 x 768 pixels video output. The TVs and the PC graphics boards must support this resolution.
- Local network connecting all machines, and to the Internet (access to photos for photo sharing functionality).
- Two remote controls.

Software:

- Server (1x; Linux)
    - **Openfire Jabber** (XMPP) server 3.4: This is an open source jabber server that is written in JAVA. The jabber protocol is used for communication between amigo clients.
    - **Asterisk SIP** server 1.2.6 (with TCP patch): Asterisk is a popular open source SIP server with many options and plug-ins. The sip protocol is used for setting up voice and video communication between amigo clients.
    - **Ruby on rails**: The Ruby on rails web framework is used to create an HTTP interface for the amigo backend.
    - **MySQL**: MySQL is the popular open source database server. Openfire, Asterisk and Ruby on Rails use a shared database. This way we only need one data storage point for user accounts which ensures data integrity.
    - **Mongrel server**: Ruby on rails uses mongrel as application server. Several mongrel processes are run on different ports. A proxy/load balancer in the Apache webserver routes http requests to the mongrel processes.

- **Apache webserver**: Apache acts as a proxy and loadbalancer for the ruby on rails HTTP interface. When a http requests is made apache sends the request to a availible mongrel process.

- Client (2x; Windows)
  - **Video splitter** (dedicated PHI SW), or the AVT module (see section 0): splits webcam into two separate streams (local / remote)
  - **VLC 0.8.5**
  - **NotifSocket** (Firefox plugin) (dedicated PHI gateway SW, mediating between JavaScript and TCP)
  - **RedRat** hardware and software (receives and translates received RC patterns to key presses)
  - **Firefox 1.0.5.9**
  - **Browser addons**: autohide; cursor hider, keyconfig
  - Jabber client: Gaim
  - Philips screen font

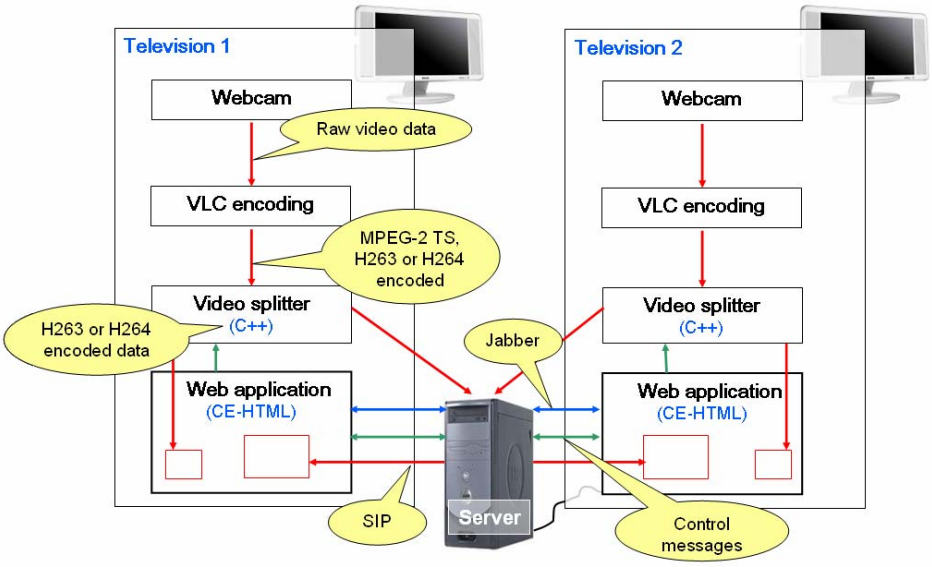

Figure 4: Activity Sharing application architecture

Platform

As described above, two Windows clients and one Linux server is used, with respectively CE-HTML (supported by JavaScript) as dedicated platform for the TV. SIP is used for video call connection setup. For community management, XMPP (a.k.a. Jabber) is used. The Linux server hosts the SIP server (Asterisk) and the XMPP / Jabber server (Wildfire).

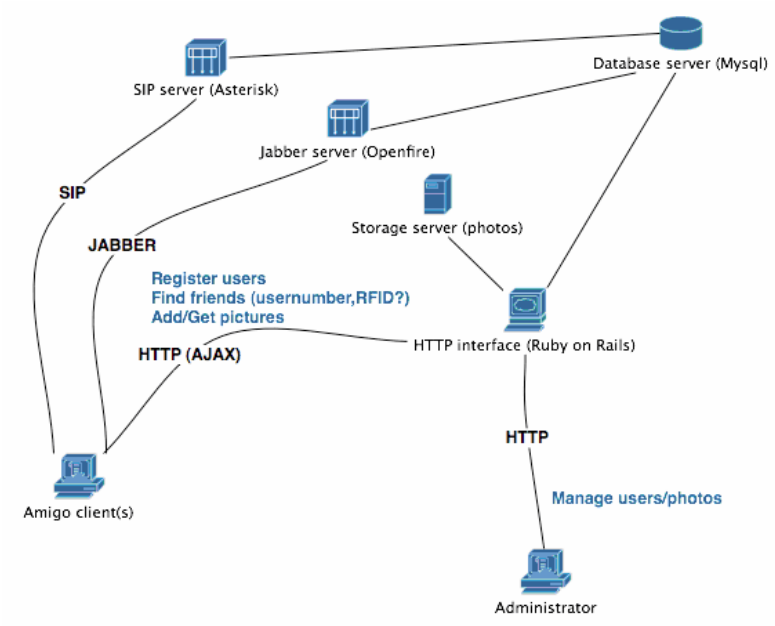

Figure 5: Activity Sharing backbone architecture

Tools

- **Gaim** was renamed to Pidgin by its development team. It is used for testing the Jabber communication messages if they were sent correctly between the applications in this project. (http://pidgin.im/pidgin/home/)
- **X-Lite** is used for testing default SIP server configurations, and soft phone to VoIP devices via standard SIP servers. X-Lite is a proprietary freeware VoIP soft phone that uses the Session Initiation Protocol. (http://www.counterpath.com/)
- **Wireshark** (formerly known as Ethereal) is a free software protocol analyzer, or "packet sniffer" application, used for network troubleshooting, analysis, software and protocol development, and education. It has all of the standard features of a protocol analyzer. Wireshark is software that "understands" the structure of different network protocols. Thus it's able to display encapsulation and single fields. Wireshark uses pcap to capture packets, so it can only capture on networks supported by pcap. (http://www.wireshark.org/)

Files

The binary release of the software is available on request from leo.rozendaal@philips.com or christian.quaedvlieg@philips.com

Documents

None

Tasks

None

Bugs

N/A

Patches

N/A

2.6 Awareness Globe

Provider

Philips Design

Introduction

The overall goal of the Awareness Globe concept is to provide persons with a tangible interface to stay aware of activities and presence of their contacts, and to initiate ambient or explicit communication between the user and their contacts. The initial Awareness Globe was developed as a proof of concept in task 4.7 of WP4. The created mock-up consisted of a smart artifact (Figure 6) and a Macromedia Flash demo (Figure 7).

The initial Awareness Globe from WP4 is a stationary, social interaction device in the house. It is located at a place where people leave their Amigo keys when at home. By leaving their key on the Awareness Globe people give themselves access to in-house services and applications. By interacting with the Awareness Globe people can control their availability to interact with the outside world.

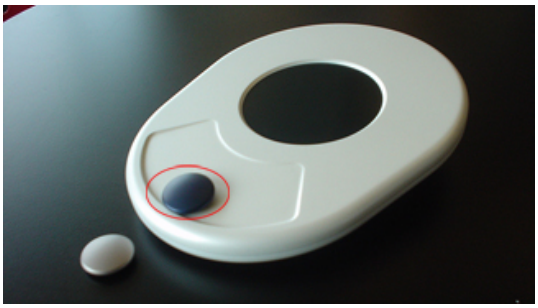

Figure 6: Awareness Globe 1 product mock-up

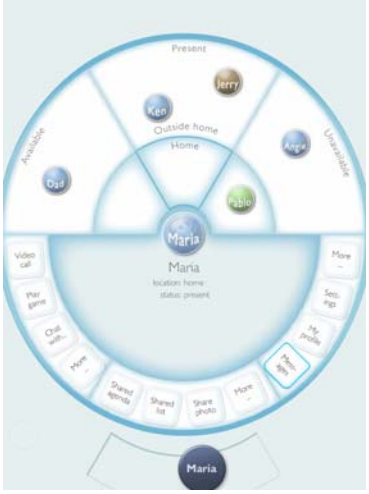

Figure 7: Awareness Globe 1 graphical User Interface

The Awareness Globe concept was forther developed towards the Awareness Globe 2 which is part of the 'shared experiences' demonstrator part of WP7. Philips Design designed and developed its Amigo prototype in conjunction with other Philips partners.

The Awareness Globe 2 GUI is deployed on a ‘portable’ photo frame, on which the user can perform the following tasks:

- Pick-up and open log-in screen, by pressing on the current picture depicted on the Photo frame.
- Choose himself as the current user of this device on the log-in screen, by this action the user makes him/herself available in the shared activities environment of the AG2
- See user status, activities and location of his/her contacts in his ‘Circle of friends’ (main UI screen of the Awareness Globe 2)
- Browse his/her ‘circle of friends’ via touch interaction and select contact per contact, while seeing which shared activities are available per contact (filter per contact)
- Invite a contact to this shared activity (main interaction path 1)
- Select him/herself (current user) and browse through his/her shared activities. Per activity the available users are shown (filter per activity)
- Invite an available contact to this shared activity (main interaction path 2)
- Launch a shared activity
- Receive invites from other contacts in his circle of friends
- Control his status for shared activities, by logging out of the Awareness Globe 2 service.

For more details and related screenshots please refer to the extensive user guide in chapter **Error! Reference source not found. Error! Reference source not found.**.

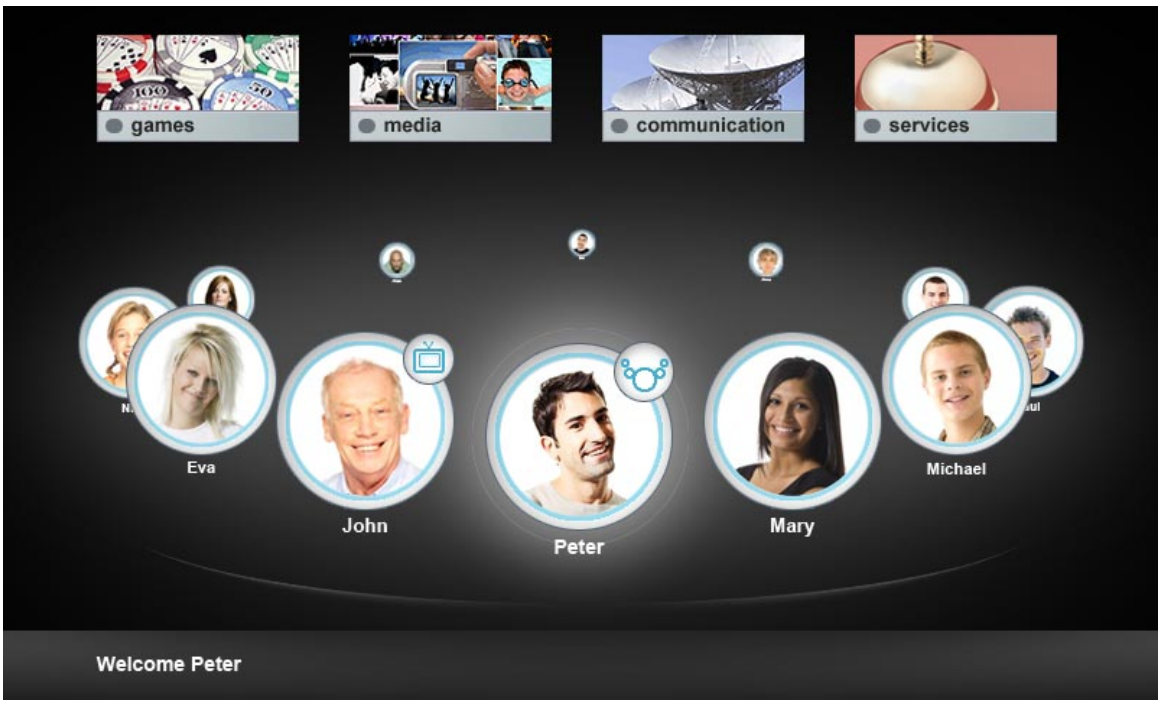

Figure 8: Awareness Globe 2 main screen

The user is able to launch shared activities by either selecting an activity he/she wishes to engage in and see who is able and available to. Or select a contact and invite this person to a shared activity.

The Awareness Globe 2 can act as a:

- Photo frame
- Log in screen (for present users)
- Status controller
- Shared activity launcher
- Secondary screen towards an activity
- Link to users social network during a shared activity

In order to organize all sorts of shared activities, these are organized in 4 main categories, as shown on top of the screen:

1. Games (play games together)
2. Media (share media, e.g. picture sharing)
3. Communication (e.g. chat)
4. Services (all other services that are related to activity sharing)

Development status

The interface for the “Awareness Globe” application is completed. The Awareness Globe has been integrated with two amigo services, CMS and UMPS. It can be used as a home amigo

device to start and share activities with friends. The first shared activity that is integrated is a multi-user Poker Game application.

Intended audience

Amigo internal

License

This software is proprietary software by PHILIPS. It will not be made available as open source. Note that the use of this software is restricted to the scope of the Amigo project.

Language

Awareness Globe GUI: CE-HTML / JavaScript

Television menu GUI: CE-HTML / JavaScript

AvailableDevicesPublisher: C#, .NET

GameService (Poker): C#, .NET

MiniPalantir: C#, .NET

Environment (set-up) info needed if you want to run this sw (service)

- One tablet PC
- One desktop PC (server)
- Two client desktop PC's.
- Two RC receivers, connected to the desktop computers (e.g. RedRat, see http://www.redrat.co.uk)
- Two widescreen TV's, connected through VGA cable to the client desktop computers. The application generates 1280 x 768 pixels video output. The TVs and the PC graphics boards must support this resolution.
- Local network connecting all machines.
- Two remote controls.
- Sensite Solutions HBL100 Wireless Network controller (RFID reader)
- Sensite Solution BN208 Intelligent tag (2x)

Software:

- Server (Windows XP SP2)
    - OSCAR OSGI Framework
    - Amigo .NET Framework
    - WAMP webserver
    - Wildfire
- Client (2x; Windows XP SP2)
    - Mozilla Firefox browser

- NotifSocket (Firefox plugin) (dedicated gateway SW, mediating between JavaScript and TCP)
- Full-Fullscreen Firefox extension
- RedRat hardware and software (receives and translates received RC patterns to key presses)

Platform

The server and two client PC's runs Windows XP SP2 with respectively CE-HTML (supported by JavaScript) as dedicated platform for the TV/Tablet PC. Wildfire is used as communication management, and runs on the server.

Tools

Pidgin (XMPP/Jabber client)

Oscar

Files

All related files can be found in the gforge repository at:

[amigo] / WP7 / ActivitySharing / AwarenessGlobe

Documents

Deliverable D4.4 - Mock-ups of user interfaces/artifacts for user testing
D4.7.2 (Amigo internal deliverable) - User-based Multiple-lab Evaluation
Previous WP7 documentation (D7.1, D7.2, D7.3)

Tasks

-

Bugs

-

Patches

N/A

2.7 Social Radio

Provider

Fraunhofer SIT

Introduction

Social Radio (SR) is a novel approach for mediating awareness in small intimate groups. Instead of traditional communication media, music is used to inform users about the presence and mood of multiple remote peers. The system consists of several smart artifacts and an underlying multi-user communication infrastructure.

Figure 9: Social Radio artifacts

Each user has several artifacts at home that represent their personal circle of friends. Each artifact represents one remote individual and displays awareness information about that person. The presence of a remote person is indicated using ambient light. In addition, an artifact communicates the mood of a remote user by re-playing the music the person is currently listening to.

Figure 10: Interaction with Social Radio artifacts

In order to provide users with lightweight interaction mechanisms the artifacts are controlled via a tangible user interface. Depending on the position, an artifact is switched off or in different operating modes.

Components

The Social Radio Demonstrator application is composed of several Components:

- *Artifacts* the physical devices used to control the application state.
- The *ArtifactController* (Social Radio driver) is a low level application, that is used to turn the light on and off, and to determine and propagate the current lie position of each artifact
- *Social Radio Demostrator* encapsulates the whole logic of the application and provides a simple GUI to observe the current state.
- *Social Radio Amigo service.* The service warps and forward the state messages from Palatir to the Social Radio Demonstrator and vice versa.
- *Music Player and its binding.* As a music player we took the jlGui 3.0[3]. We extend the player for translation of the music streams over the network. The receiving of the

[3] MP3 Player for the Java Java ™ platform: http://www.javazoom.net/jlgui/jlgui.html

stream on the other side is done by the JMF[4] receiver. The receiver itself is controlled from the Social Radio Demonstrator.

Development status

A fully working version is available.

Intended audience

WP7 partners – currently only 4 Social Radio artefacts exist, which limits the intended audience.

License

Social Radio software (the SR driver and the demonstrator program, the Amigo service) is licensed under GPLv2.

Language

The Social Radio driver and configuration program is developed in C.

The Amigo service integrating the Social Radio, is developed using the Java and the WP3::OSGi development and deployment framework.

Environment (set-up) info needed if you want to run this sw (service)

Hardware:

- Computer with at least 1 sound card and 3 USB ports. A USB hub can be used. Up to 3 sound cards are supported. One sound card for each working artifact.
- The 3 working, 1 dummy (just case) artifacts from SIT
- 3 transceivers for controlling the artifacts.

Software:

- Windows XP SP2
- The SR driver
- The SR configuration program

Platform

An Oscar OSGi runtime environment with Amigo middleware is needed.

Tools

For development Visual Studio is needed with a QT SDK.

For the Social Radio Amigo service, a Java IDE, like Eclipse or similar is needed.

[4] Java Media Framework API (JMF): http://java.sun.com/products/java-media/jmf/

Files

The sourcecode for the Social Radio driver, amigo service and configuration program is hosted in the gforge repository:

[Amigo] / WP7 / Socialradio

Documents

The artifacts have been described in the Amigo deliverable D4.4. This document will be extended to cover deployment of the social radio in section 3.7.

The artifacts have furthermore been presented at the International Conference on Tangible and Embedded Interaction and at the Conference on Intelligent User Interfaces:

- R. Etter, C. Röcker: "A Tangible User Interface for Multi-User Awareness Systems", Proceedings of the International Conference on Tangible and Embedded Interaction '07, Baton Rouge, USA, February 15-17, 2007.
- C. Röcker, R. Etter: "Social Radio: A Music-Based Approach to Emotional Awareness Mediation", Proceedings of the 10th International Conference on Intelligent User Interfaces, Honolulu, USA, January 28-31, 2007.

Tasks

-

Bugs

In the Social Radio Demonstration program, in some cases the icon representing the status of the artifact is shown incorrect.

Patches

N/A

2.8 Board Game

Provider

Fraunhofer SIT,

VTT (3D gesture device integration)

Introduction

There is a growing trend in the computer gaming research community to augment traditional video games with aspects from the real world. These pervasive games combine the virtual nature of traditional video games with physical and social context, thus creating immersive gaming experiences that pervade the boundaries of virtual, physical and social domains. The Board Game is a pervasive tabletop game that provides tangible interfaces borrowing interaction techniques from traditional Board Games (see Figure 11).

The Board Game is part of both, WP6 and WP7, but the use case scenarios differ from each other. In the home information and entertainment scenario of WP6, the Board Game is deployed within a single household, where players can play the "Caves & Creatures" game

while being in the same physical space. Since most of the development of the current version has been done within WP6, a detailed description of the Board Game will be put in the respective final deliverable (D6.4) of WP6.

Within the WP7 scenario, the Board Game experience is shared by people at distant locations. In this setup, at one location users can play the game by manipulating figures on a physical board, while at the second location people use a conventional GUI. Further development of the Board Game was necessary to make it fit into the scenario of the extended Amigo home environment. This documentation of the Board Game focuses on the WP7-related issues only.

Figure 11: The Boardgame

Overview

As the complete description of the Board Game (BG) is in deliverable D6.4, this section contains only a short overview.

The Board Game is a hybrid board game that uses ambient intelligence methods for enhancing the experience of a traditional role playing game. A tangible interface can be controlled via physical figures and gestures, PDAs and RFID tokens. The demonstrator features a working setup of a physical board, a 3D visualization, sound output, the VTT gesture device, RFID and PDA support. The game features multi-device and multi-modality interfaces and is build upon a framework called PEGASUS, to synchronize the various input and output devices with the game play.

The game play is oriented towards the original role-playing game “Dungeons and Dragons” (D&D) that is played with pawns, dices and “magic” cards. Each pawn on the board represents one player. Tangible artifacts (RFID tokens) represent items, e.g. sword or axe can be applied to the active player using an RFID reader. The players can attack each other using weapons, whereby rolling the virtual dices determines the winner. The virtual dices are rolled by using the VTT gesture device as if one shakes a dice cup. In the same manner, the gesture device can be used as a virtual wand allowing to apply magic spells.

Along the physical board that was built upon an electronic chess board, the look and feel of the Board Game is mainly achieved by the use of public and private displays. One “public display” reflects the setting of the pawn on the physical board by displaying avatars in a 3D visualization. Each avatar can carry items, e.g. a weapon. 3D animations were inspired by the famous chess game “Battle Chess” known as the first chess game that shows explicit combat action between the pawns. PDAs are used as “private displays” to show the properties of each player, e.g. his health status, movement points, collected items and available magic spells.

Components

The Board Game application is composed of several Components:

- *The electronic chess board including the pawns. The board is connected via USB to a PC running the Board Game application and showing the public 3D display*
- *A PDA running a java midlet to serve as "private display"*
- The VTT gesture device "soap box" and the gesture server. The controller is connected via RS232 to the PC's serial port.
- *A RFID reader that is also connected via USB to the PC.*
- *The Board Game Amigo service.* The service wraps and forwards the state messages from Palantir to the Board Game Demonstrator and vice versa.

Development status

A fully working version is available. Development has finished at a level, were end-users can experience the Board Game with its specific interfaces from two distant locations. At one location the physical board is set-up and at the other location only the 3D visualization is used to interact with other players. The set of game rules allows to move pawns on the board or from the 3D visualization, use weapons and dices, apply magic spells etc. But the game play doesn't define a goal that would lead to an end or makes someone win.

Intended audience

Amigo consortium, end-users.

License

Board Game software is licensed under GPLv2. Drivers and applications required to run the hardware components, such as PDA, electronic chess board and RFID drivers as well as the JRE are subject to the respective manufacturer of these components.

Language

The Board Game application is developed in C++. The Amigo service integrating the Board Game is developed using the Java and the Amigo OSGI runtime.

Environment (set-up) info needed if you want to run this sw (service)

Hardware:

- PC with at least Pentium 4, 3GHz, 1GB ram with a DirectX9-capable graphics card, sound card, 4 USB ports (USB hub can be used) and 1 serial port
- The hardware components listed in the above section "components"

Software:

- Windows XP SP2
- Java 1.5
- The Board Game application

Platform

The Amigo OSGi based runtime environment is needed.

Tools

For development of the BG, Visual Studio 2003 is recommended.

For the Board Game service, a Java IDE, like Eclipse or something similar is recommended.

Files

The source code for the Board Game and Amigo service is hosted in the gforge repository:

[Amigo] / WP7 / Boardgame

Documents

A complete documentation of the Board Game is part of deliverable D6.4

The configuration and setup procedure for the Board Game is described in the document *TechnicalDescription.doc*, available together with the development project on the gforge source code repository at:

[Amigo]/WP7/Boardgame/TechnicalDescription.doc

A document containing the full description of the developed PEGASUS framework is available at the same location:

[Amigo]/WP7/Boardgame/PEGASUS_manual.doc

Tasks

-

Bugs

The public display may crash due to the version of the 3D engine used. But as the main focus was to develop a prototype for hybrid gaming build upon AMIGO and the PEGASUS framework the very first running version of the 3D visualization used a very early version of the “Irrlicht engine” that was assumed to suffice our needs. The migration to the latest stable version of Irrlicht would require a total redevelopment of the 3D visualization as the interfaces have changed significantly.

Patches

N/A

2.9 Feeling@ application building blocks

2.9.1 The Shared Organizer

Provider

ITAL

Introduction

The Shared Organizer component handles user activities shared among different locations. The component has both a backend logic and a frontend GUI. The Shared Organizer backend interacts between multiple instances of itself distributed over the network. In such a way the user has the feeling of having all his activities at hand in any place he goes. The Shared Organizer frontend offers a unified view of the user's activities and allows him to manage them. The information is transparently persisted on the node where the activity belongs.

The Shared Organizer component interacts with the WP4::IUS::UMPS for user profile information retrieval and publishes events (WP3::WS-Eventing) on activity changes for other services to consume. It also carries a WP4::IUS:CMS hook to retrieve and publish user location information from RFID sensors. At a higher level, the Shared Organizer registers to the Palantir service for presence and awareness notifications and interacts with the WP7::Scheduler and the Browser Service.

Development status

The development of the Shared Organizer module is completed. The current version offers:

- Activity data persistence and retrieval
- Activity information discovery and aggregation
- Activity changes propagation through WS-Eventing
- User profile and location retrieval
- Activity data manipulation through web GUI

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Uses the *Spring Framework*, *Castor XML* and *Jakarta Commons Logging* libraries all released under Apache 2.0 License. *MySQL Server* is offered with GPL license for open source projects. *OpenLaszlo* comes with the Common Public License (CPL 1.0).

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3, MySQL Server 5.0 (or above), OpenLaszlo Server 4.0.0 (or above).

Libraries: Spring Framework 2.0.2, Castor XML 1.1, Jakarta Commons Logging 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13.

Amigo Services: WP3::AmigoCore, WP4::IUS::UMPS, WP4::IUS::CMS Context Helper, WP7::Feeling@Core, WP7::Feeling@DBPool, WP7::Feeling@XmlRpc-3.0, WP7::Palantir, WP7::Scheduler, WP7::BrowserService.

Platform

JVM 5.0 (or above), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code, MySQL tools for database administration.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/SharedOrganizer/trunk (backend)
[amigo] /WP7/Feeling@/SharedOrganizer.GUI/trunk (frontend)

The Feeling@ support libraries are located at:
[amigo] /WP7/Feeling@/Core/trunk
[amigo] /WP7/Feeling@/DBPool/trunk
[amigo] /WP7/Feeling@/XmlRpc-3.0/trunk

Documents

D7.1 WP7 Initial Specification and Test Plans
D7.2 Extended Home Application Specification

Tasks

None scheduled. Developers can improve GUI functionality and add new activity features.

Bugs

-

Patches

N/A

2.9.2 The Sketch Presentation

Provider

ITAL

Introduction

The Sketch Presentation component allows users to share and compare sketches (bitmaps). Sketches are organized in two folders: local and remote. Both folders hold thumbnails of the bitmaps retrieved from URLs. On user request, sketches are displayed individually in center screen or in couples side by side. Bitmap choice is determined by the user selecting freely from both folders. Sketch Presentation aims at demonstrating advanced ambient interaction with voice and gesture control. All features supported by the application can be issued via voice and gesture commands.

The component holds both a backend logic and a frontend GUI. The Sketch Presentation backend registers to the Palantir application for presence and awareness notifications and supports WP7 integration guidelines by interacting with the application Scheduler (see section 2.1) and the Browser Service. The Shared Organizer frontend hooks to the WP4 Voice Service and Gesture Service subsystems to offer either way of interaction.

The Sketch Presentation component also provides a simple chat client based on the Jabber/XMPP standard.

Development status

The development of the Sketch Presentation module is completed. The current version offers:

- Sketch retrieval and display both in thumbnail and full size
- Command issuing using both gesture and voice recognition
- Simple chat client

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Uses the *Smack API*, *Castor XML* and *Jakarta Commons Logging* libraries all released under Apache 2.0 License. *OpenLaszlo* comes with the Common Public License (CPL 1.0).

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3, OpenLaszlo Server 4.0.0 (or above).

Libraries: Smack 3.0.0, Castor XML 1.1, Jakarta Commons Logging 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13.

Amigo Services: WP3::AmigoCore, WP7::Feeling@Core, WP7::Feeling@XmlRpc-3.0, WP7::Palantir, WP7::ApplicationScheduler, WP7::BrowserService

Platform

JVM 5.0 (or above), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/SketchPresentation/trunk (backend)
[amigo] /WP7/Feeling@/SketchPresentation.GUI/trunk (frontend)

The Feeling@ support libraries are located at:
[amigo] /WP7/Feeling@/Core/trunk
[amigo] /WP7/Feeling@/XmlRpc-3.0/trunk

Documents

D7.1 WP7 Initial Specification and Test Plans
D7.2 Extended Home Application Specification

Tasks

None scheduled. Developers can improve GUI functionality and blend voice and gesture interaction (multimodality).

Bugs

-

Patches

N/A

2.9.3 The User Notification Messenger

Provider

ITAL

Introduction

The User Notification Messenger component manages user notifications by sending messages possibly in multiple formats. The component reasons over context changes incoming from the WP4:IUS:CMS (user location) and the WP7::Feeling@SharedOrganizer (activity changes). It invokes the WP4:IUS:UMPS and the WP6::HomeAgenda services to obtain user information and schedule. If instructed to do so, the component sends a

notification message to a user according to his presence and availability values obtained from the WP7::Palantir service.

Development status

The development of the User Notification Messenger module is completed. The current version offers:

- User schedule accordance by matching user location to planned location
- Schedule responsible notification if user location is unexpected
- Activity changes notification to involved users

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Uses the *Castor XML* library released under Apache 2.0 License.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3.

Libraries: Castor XML 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13.

Amigo Services: WP3::AmigoCore, WP4::IUS::UMPS, WP4::IUS::CMS Context Helper, WP6::HomeAgenda, WP7::Palantir, WP7::Feeling@SharedOrganizer, WP7:Feeling@Core, WP7::Feeling@MailService.

Platform

JVM 5.0 (or above), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/UserNotificationMessenger/trunk

The Feeling@ support libraries are located at:
[amigo] /WP7/Feeling@/Core/trunk
[amigo] /WP7/Feeling@/MailService/trunk

Documents

D7.1 WP7 Initial Specification and Test Plans
D7.2 Extended Home Application Specification

Tasks

None scheduled. Developers can add further messaging gateways (SMS for ex.) and more processors on Amigo events.

Bugs

-

Patches

N/A

2.9.4 The Conference Manager

Provider

ITAL

Introduction

The Conference Manager component handles audio and video communication between local and distant parties. Currently the module supports both hardware and software solutions for conferencing.

Hardware support is given using the Tandberg video stations. These hardware solutions offer remote HTTP control which is then exposed by the Conference Manager as a web and Xml-Rpc services. Tandberg video stations offer advanced audio and video streaming with SIP support.

Software support is provided by France Telecom eConf-based AVT (see section 0) player. The Conference Manager indirectly triggers eConf by handling SIP session initialization and packet switching.

The Conference Manager is based on WP7 Ambience Sharing application and similarly registers to the Palantir application for presence and awareness notifications and supports WP7 integration guidelines by interacting with the Application Scheduler.

Development status

The development of the Conference Manager module is being finalized. The current version offers:

- Remote control of the Tandberg video station hardware
- SIP management of outgoing and incoming calls with audio and video support

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

This service only works with specific hardware and software:

- Tandberg MXP video stations
- France Telecom eConf player 5.0.8 or above
- OpenSer or Brekeke SIP servers

PC, JVM 6.0, OSGi-Based Programming & Deployment Framework from WP3.

Libraries: France Telecom SipJavaStack 1.0 SNAPSHOT, Castor XML 1.1, Jakarta Commons Logging 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13.

Amigo Services: WP3::AmigoCore, WP7::Feeling@Core, WP7::Feeling@XmlRpc-3.0, WP7::Palantir, WP7::ApplicationScheduler

Platform

JVM 6.0, OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/ConferenceManager/trunk

The Feeling@ support libraries are located at:
[amigo] /WP7/Feeling@/Core/trunk
[amigo] /WP7/Feeling@/XmlRpc-3.0/trunk

Documents

D7.1 WP7 Initial Specification and Test Plans
D7.2 Extended Home Application Specification

Tasks

-

Bugs

Only specific SIP servers support Sdp packet switching. Currently Asterisk is not supported.

Some webcam hardware evidences problems with eConf and the Sdp packet switching.

Patches

N/A

2.9.5 The RFID Reader

Provider

ITAL

Introduction

The RFID Reader component provides RFID sensor data as context information to the Amigo system. Implemented as a "Context Source" it registers itself to the "Context Broker" of the WP4::IUS::CMS. The RFID Reader is usable in any context aware environment. In WP7 the RFID Reader is be leveraged by the WP7::Palantir component in order to determine "User Presence" in a local environment. The RFID Reader comes in two flavors: Java based for passive RFID readers (and tags) and .NET based for active beacons.

Development status

The development of the RFID Reader is completed. The current version offers:

- User location context information via direct queries and subscriptions
- Support for both passive and active RFID sensors

Intended audience

End-users, Developers

License

GNU Lesser General Public.

The C# implementation uses *Semweb (Joshua Tauberer)*, which is released under Creative Commons Attribution License and *Sparql engine for Java (Ryan Levering)* converted from Java to C# with *IVKM (Jeroen Frijters),* which is released under GNU classpath license. Also used is the *Log4net* library released under Apache 2.0 License and the *Iesi.Collections* library (Enhanced Collections for .NET) by *Jason Smith* released as open source. For hardware support the *IDENTEC SOLUTIONS SDK For The Microsoft .NET Framework* has been used. The library is not open source and comes with a specific EULA.

The Java implementation uses *Jena Semantic Web Framework* by Hewlett-Packard Development Company with LGPL style license. For hardware support *FEIG OBID Java SDK 2.0.4* has been used. The library is not open source and comes with a specific EULA.

Language

Java & C# (.NET)

Environment (set-up) info needed if you want to run this sw (service)

This service only works with specific hardware:

- FEIG Electronic ISC.PR100 passive RFID reader (optional SENA Technologies LS100 Serial to Ethernet converter)
- IDENTEC SOLUTIONS i-PORT R2 active RFID reader + antenna (optional SENA Technologies PS110 Serial to Ethernet converter)

The software platform varies upon RFID reader choice:

- PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3 (for passive readers).
- PC, Microsoft .NET Framework v2.0, Amigo .NET Programming Framework from WP3 (for active readers).

Support libraries (included in the bundles) depend on the software platform:

- WP7::Feeling@Core library, FEIG OBID Java SDK (for the Java platform).
- Semweb, Log4net, IDENTEC SOLUTIONS SDK For The Microsoft .NET Framework (for the .NET platform).

The OSGi necessary bundles are Service Binder, Log4J 1.2.13 and Jena Semantic Web Framework 2.4.

The Amigo required components are the WP4::IUS::CMS Context Broker (both platforms), WP3::AmigoCore and WP4::IUS::CMS Context Source Manager (Java platform only).

Platform

JVM 5.0 (or above), OSGi-Based Programming & Deployment Framework from WP3

Microsoft .NET Framework v2.0, Amigo .NET Programming Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code (passive readers).

Visual C# .NET 2005 for compiling the source code (active readers).

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/RFIDReader/trunk (Java platform)
[amigo] /WP7/Feeling@/RFIDReader.Net/trunk (.NET platform)

The Feeling@Core library is located at:
[amigo] /WP7/Feeling@/Core/trunk

Documents

D7.1 WP7 Initial Specification and Test Plans
D7.2 Extended Home Application Specification

Tasks (TBD)

-

Bugs

-

Patches

N/A

2.9.6 The Gesture Service Manager

Provider

ITAL

Introduction

The Gesture Service Manager component is a support bundle for the OSGi platform that bridges VTT's SoapBox hardware to the Amigo service-aware environment. The module opens a socket server to receive SoapBox real time commands in plain, MMIL and SOAP headers formats. The issued commands are queued, ready to be consumed by Amigo applications. A web service and Xml-Rpc endpoint are exposed and to which applications can poll and consequently process the queued commands.

Development status

The development of the Gesture Service Manager is completed.

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

This service only works with VTT's SoapBox specific hardware and software.

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3.

OSGi bundles: Service Binder, Log4J 1.2.13.

Amigo Services: WP3::AmigoCore, WP7:Feeling@Core, WP7::Feeling@XmlRpc-3.0.

Platform

JVM 5.0 (or above), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/GestureServiceManager/trunk

The Feeling@ support libraries are located at:
[amigo] /WP7/Feeling@/Core/trunk
[amigo] /WP7/Feeling@/XmlRpc-3.0/trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.9.7 The Voice Service Manager

Provider

ITAL

Introduction

The Voice Service Manager component is a support module for the Sketch Presentation component. The module works in conjunction with the WP4::UIS Voice Service to provide simpler voice interaction methods. Acting as a proxy, the Voice Service Manager relays voice synthesis and recognition requests handling the low level operations needed as wave file conversion, microphone capture and speaker output. On the other side offers a simplified web service interface.

Development status

The development of the Voice Service Manager is completed.

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Language

C# (.NET)

Environment (set-up) info needed if you want to run this sw (service)

The component should be installed on the same PC where the WP4::UIS Voice Service is.

PC, Microsoft .NET Framework v2.0, Amigo .NET Programming Framework from WP3.

Amigo Services: WP4::UIS Voice Service.

Platform

Microsoft .NET Framework v2.0, Amigo .NET Programming Framework from WP3

Tools

Visual C# .NET 2005 for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/VoiceServiceManager.Net/trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.9.8 The DB Pool

Provider

ITAL

Introduction

The DB Pool component is a support bundle for the OSGi platform that provides JDBC connection pooling to any application that requires fast, consistent and resource independent RDBMS access. The bundle offers JDBC Connection resources through the standard *DataSource* interface. It embeds the fast and lightweight DBCP connection pooling library from the Apache Jakarta project. Currently the bundle carries the MySQL JDBC drivers only, but it can include other JDBC drivers as well without requiring any modification to the code.

Development status

The development of the DBPool is completed.

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Uses the *Jakarta Commons DBCP*, *Pool* and *Collections* libraries which are released under Apache 2.0 License. *MySQL Server* JDBC drivers (5.0.4) included come with GPL license for open source projects.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 1.4, OSGi-Based Programming & Deployment Framework from WP3.

Libraries: Jakarta Commons DBCP 1.2.1, Pool 1.3, Collections 3.2 and MySQL Connector/J 5.0.4 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13, WP3::AmigoCore.

Platform

JVM 1.4 (or upper), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/DBPool/trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.9.9 The Mail Service

Provider

ITAL

Introduction

The Mail Service component (aka JavaMail Service) is a support bundle for the OSGi platform that provides e-mail commodities to any application that needs the SMTP, POP and IMAP protocols. The bundle offers JavaMail resources through the standard J2EE Session class.

Development status

The development of the Mail Service is completed.

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Uses Sun's *JavaMail* and *JavaBeans Activation Framework* libraries which are released as open source.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 1.4, OSGi-Based Programming & Deployment Framework from WP3.

Libraries: J2EE JavaMail 1.4 and JavaBeans Activation Framework 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13, WP3::AmigoCore.

Platform

JVM 1.4 (or upper), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/MailService/trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.9.10 The XmlRpc-3.0

Provider

ITAL

Introduction

The XmlRpc component is a support bundle for the OSGi platform that provides standard XML-RPC remoting protocol to any application that requires fast and lightweight XML based RPC style distributed computing. The bundle is intended to be used by the rendering platform of choice of the Feeling@ demonstrator: OpenLaszlo.

Development status

The development of the XmlRpc component is completed.

Intended audience

End-users, Developers

License

GNU Lesser General Public.

Uses *Apache XML-RPC* and *Jakarta Commons Logging* libraries which are released under Apache 2.0 License.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3.

Libraries: Apache XML-RPC 3.0, Jakarta Commons Logging 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13, HTTP Service, WP3::AmigoCore.

Platform

JVM 5.0 (or upper), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] / WP7 / Feeling@ / XmlRpc-3.0 / trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.9.11 The Core Library

Provider

ITAL

Introduction

The Core library is an effort to encapsulate OSGi and Service Binder services with an event based approach. The idea is to free the developer from coding helper classes for each

deployable OSGi bundle. With the Core library developers can register multiple objects for OSGi and Service Binder events (as component initialization, service availability, etc.), avoiding the need to share service references among several code units of the same bundle. Most important events published are: bundle lifecycle, component lifecycle, HTTP Service availability, service discovery & service exporting availability. Developers can extend the base classes to include further services support.

Development status

Development completed.

Intended audience

Developers

License

GNU Lesser General Public.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3.

Libraries: Castor XML 1.1 (included in the bundle).

OSGi bundles: Service Binder, Log4J 1.2.13, HTTP Service, WP3::AmigoCore.

Platform

JVM 5.0 (or upper), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/Feeling@/Core/trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.9.12 The Browser Service

Provider

ITAL

Introduction

The Browser Service component is a simple web browser launcher for WP7 applications that provide a thin client GUI. According to the WP7 integration guidelines, GUIs that share a common display, should be able to appear and hide upon application and Scheduler requests. As such the Browser Service exposes a simple web service interface to open a specified URL starting a web browser or close a previously opened window. Internally the Browser Service registers to the display resource manager service hosted on the same PC. As soon as the browser is opened or closed, the component will notify the resource manager accordingly.

Development status

Development is completed.

Intended audience

Developers

License

GNU Lesser General Public.

Language

Java

Environment (set-up) info needed if you want to run this sw (service)

PC, JVM 5.0, OSGi-Based Programming & Deployment Framework from WP3.

OSGi bundles: Service Binder, Log4J 1.2.13, WP3::AmigoCore.

Platform

JVM 5.0 (or upper), OSGi-Based Programming & Deployment Framework from WP3

Tools

Oscar, Java IDE such as Eclipse for compiling the source code.

Files

The following locations in the Gforge repository contain the component source code and deployable units:

[amigo] /WP7/BrowserService/trunk

Documents

-

Tasks

-

Bugs

-

Patches

N/A

2.10 Personal Amigo Device

Provider

TELIN

Introduction

In ubiquitous environments such as provided by Amigo, access to services is limited to known clients and devices which have undergone some sort of imprinting technique to make them known to the security service. This limits the use of services to clients that are well-known, e.g. the ones belonging to the users of the home itself. Allowing access to guests is cumbersome since they have to undergo the same imprinting technique, the same requirements for getting guest access also limits the ability to securely access services from your own home while visiting another Amigo home.

The idea behind the Personal Amigo Device (PAD) is to create a dynamic and trusted association between the home environment of the guests and the visited environment, using the Personal Access Device (PAD) as a trust-bridge. This association supports service discovery and service usage between the homes, allowing the guests to use their own services as if at home, using the devices available in the visited environment.

The Personal Amigo Device (PAD) offers users a simple and transparent way to access home services in a visited environment. It facilitates seamless creation of trust between a user's own home and a visited intelligent environment (i.e. home).

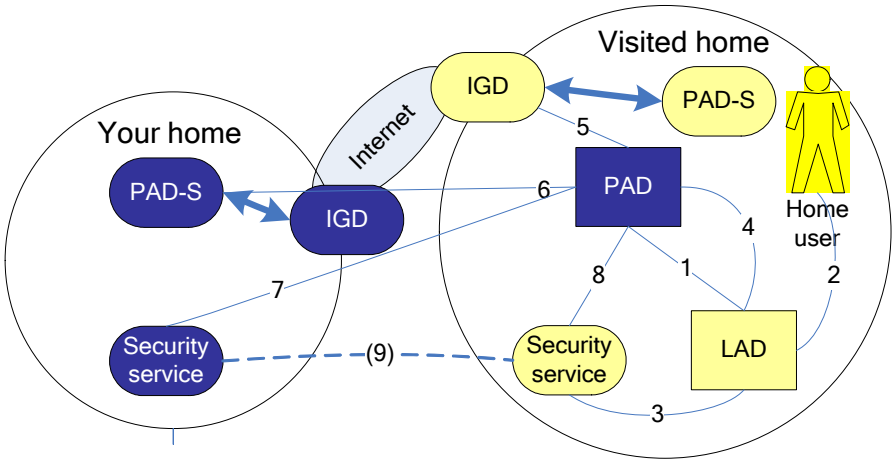

Figure 12: Steps in trust establishment between two homes.

To dynamically establish the trust between the two homes, the PAD acts as a mediator between the (WP3) security services of the two different homes. **Error! Reference source not found.** Figure 12 shows the different steps and interactions between the different components that play a role in the PAD scenario, these are: Personal Amigo Device (PAD), PAD-Service (PAD-S), Internet Gateway Device (IDG, a home router), Local Authentication Device (LAD, an RFID reader like the Awareness Globe v1). These components work together to enable the sharing of services between homes. The interactions between these components are described in more detail in section 4.9.

The practical setup is shown in Figure 13, below. For the experiments and realization the Qtek9000 (Windows Mobile) and QBIC belt (Linux) were used, although any Windows Mobile device with Bluetooth and WLAN connectivity can potentially be used to run the PAD software.

Determining which home services can be exported to a visited home is done by looking for services which have a property named *exportable* set to *true*. When publishing these services in the visited home, these properties are modified to a different property named *exported*, again set to *true*. Modifying the property name when publishing the reference in a visited home prevents these services from being exported again to yet another home if a user of the visited home has a PAD as well and is visiting another home himself.

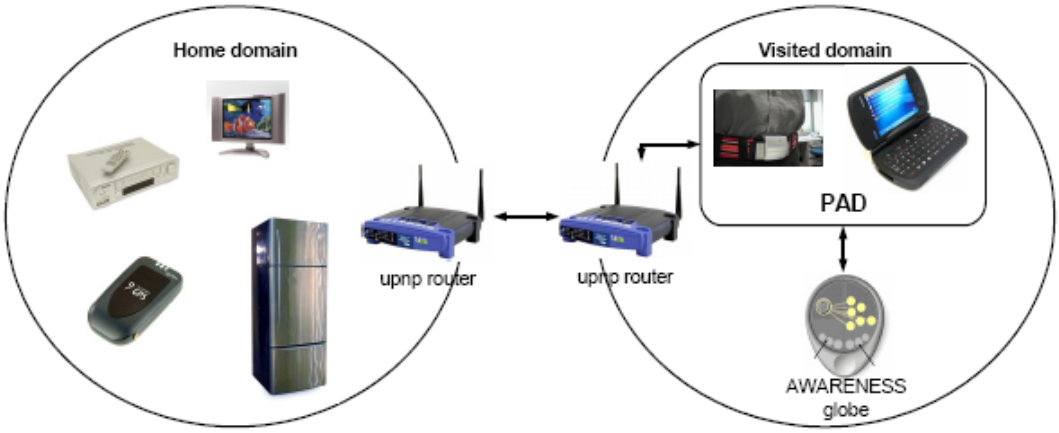

Figure 13: Practical setup of the PAD scenario.

A more detailed discussion of this PAD scenario is available from the proceedings of the Amigo workshop that was held in conjunction with the AmI-07 conference.

Development status

The PAD and corresponding software are completed and include:

- Transparent advertisement of home services in the visited home.
- User-consent using an RFID identification service (or simulator for demo purposes).
- Basic security using UPnP and NAT traversal.
- Integration with the Amigo security service.
- Graceful disconnection of the user and visited home.
- Integration with the Media Manager Core from WP6; allowing content to be exported to visited homes, next to the export of Amigo services.

Intended audience

End-users, Developers

License

LGPL

Language

Java (main functionality and logic), C++/C# (in supporting libraries and native code for the Qtek 9000)

Environment (set-up) info needed if you want to run this sw (service)

Hardware

Qtek 9000 (Windows Mobile 5 OS) or QBIC Belt computer (Linux OS), PC with a Bluetooth dongle for the visited domain, PC for the home domain.

Other devices with WLAN and Bluetooth connectivity running Windows Mobile 5 or Linux may work, but are untested.

Two UPnP enabled routers/home gateways (Linksys WRT54Gv7 are currently used), to create two different home environments.

Hardware for initiating the PAD scenario: Awareness Globe or another type of RFID reader, RFID tag. (This can also be done using a simulated RFID reader, which is also provided on gforge)

Software

J9 VM (IBM WEME), BlueCove (for Qtek), BlueZ (for QBIC), Oscar for mobile devices, Oscar for J2SE, Amigo middleware and bundles, WP3 Security Service.

Platform

A java virtual machine for back-end systems, and in case of mobile devices the IBM J9 virtual machine. A windows machine to run the Security Service from WP3.

Tools

Ant, C-compiler, Java compiler, Visual Studio for compiling native code for windows mobile version of the PAD.

Files

All files, software, and bundles specific to the PAD can be found in the gforge repository at the following locations:

[amigo] / WP7 / PersonalAmigoDevice

[amigo] / ius / context_mgmt / Bluetooth / trunk

The subdirectories of this location contain the different functionality and test services needed to run the PAD scenario.

Additional components that are needed for full functionality are the Media Manager related bundles and components (see deliverable D6.4 for a description and installation instructions), as well as the Security Service from WP3 (see deliverable D3.5 for more details).

Documents

D7.1, D7.2, D7.3

D3.5

D6.4

Poortiga et al. “Sharing Intelligent Services between Homes”, In proceedings of the AmI-07 Amigo workshop, Darmstadt 2007.

Tasks

-

Bugs

-

Patches

N/A

2.11 SAInt – Seamless Audio Interface

Provider

University of Paderborn (PAE)

Introduction

The SAInt (Seamless Audio Interface) is a software building block for real-time audio communication and ambiance sharing. It is divided in a C++-based communication software

and a JAVA-based controlling interface, which are linked with an inter-process communication (IPC) interface. The JAVA based interface is an OSCAR bundle, implementing web-services for establishing, terminating and controlling communications. The SAInt implements software modules for audio processing and real-time streaming.

The Seamless Audio Interface (SAInt) software components provide the following functionality:

- SPARK (Speech Processing and Recognition Toolkit)
    - Adaptive echo cancellation
    - Noise reduction
    - Real-time audio streaming and routing
    - Low-latency audio processing (jackd)
    - User registration and deregistration at ACS
    - Handling of follow-me elements and handovers
- SAInt (Seamless Audio Interface Amigo web service)
    - Subscription at Location Management Service (LMS)
    - Publishing context information about SAInt
    - IPC with SPARK to notify about location changes
- ACS (ambient communication server)
    - Keeps list of registered users and their IP-Addresses
    - Allows NAT and firewall transversal
    - Simple session initialization
- SAInt GUI
    - Graphical representation of context information about
        - Registered users at ACS
        - Rooms connected to local SAInt
        - Current connections (persons, privacy level, Gain factor, rooms, IP-Addresses)
    - Controls connections between
        - Local Rooms (not bound to user, always public)
        - Users with local rooms (baby phone functionality)
        - Local users with other local users
        - Local users with external users (ACS)
    - Modification of connections
        - Privacy level
        - Gain factor in dB
        - Timeout limit in seconds
- Location Management System
    - Combines the location information from various context sources (currently only RFID and APE context is gathered)

Development status

System architecture is defined and implemented.

- Modules implemented: Filter-Sum-Beamformer, Adaptive Interference Canceller, Microphone interface, Loudspeaker interface, Router, SAInt, Oscar SAInt Service, Post-filter, Real-time communication, SAInt GUI

Intended audience

Amigo consortium partners involved in demonstrators

License

This software is proprietary software by PAE. It will not be made open source since it is not middleware software. Special arrangements can be made for Amigo partners.

Language

C++ / C on Linux operating systems; Java

Environment (set-up) info needed if you want to run this sw (service)

Hardware: single- or multi-channel soundcard; microphones; PC

Software: SuSe Linux 9.3 or higher; Jack Audio Connection Kit

Platform

Java 1.5, Oscar platform

Tools

-

Files

One executable (asreng – main engine), modular based shared libraries and special configuration files.

Oscar bundle for inter-process communication and SAInt webservices.

Software packages and installation routines are available on our webpage (http://nt.uni-paderborn.de) for project partners.

Documents

Installation guide and usage guidelines are integrated in the software package.

Bugs

No known bugs

Patches

None available, but patches can be made available on demand.

3 Components Deployment

In this chapter, we describe deployment procedures for the software components described in chapter 2.

3.1 Scheduler

3.1.1 System requirements

Hardware: Any Java 5 enabled hardware with IP network connectivity

Software: Java Runtime Environment 1.5 or higher, Amigo middleware and bundles.

3.1.2 Download

All files, software, and bundles can be found in the amigo repository at:

[Amigo] / WP7 / Scheduler

3.1.3 Install

Deploy the bundle scheduler.jar from Amigo repository:

http://amigo.gforge.inria.fr/obr/WP7/

3.1.4 Configure

The following properties can be defined on the Oscar platform for the scheduler:

```
scheduler.envId = home1
scheduler.showGUI = 1
```

The first one can be specified if you want to setup two Amigo domains on the same local network. In each domain you must have only one Scheduler service.

The `showGUI` property is a Boolean toggle for enabling a simple GUI of the Scheduler service showing the status of registered applications.

3.2 Resource Manager

3.2.1 System requirements

Hardware: Any Java 5 enabled hardware with IP network connectivity

Software: Java Runtime Environment 1.5 or higher, Amigo middleware and bundles, including the CMS and Jena2.4

3.2.2 Download

All files, software, and bundles can be found in the amigo Palantir repository at:

[Amigo] / WP7 / ResourceManager

3.2.3 Install

Deploy the bundle resourcemanager.jar from Amigo repository:

http://amigo.gforge.inria.fr/obr/WP7/

3.2.4 Configure

The resourceManager service requires the following properties to be defined on the Oscar platform:

```
resourcemanager.roomId=roomA
resourcemanager.hostname=g-deimos
resourcemanager.position.x=0
resourcemanager.position.y=0
resourcemanager.threshold.interactionrange=70
resourcemanager.threshold.notificationrange=85
resourcemanager.threshold.ambiantrange=100
```

The `roomId` property specifies the location of the interaction resource that should be managed.

The `hostname` property defines the name of the PC to which the interaction resource is bound and on which services accessing the resource will run. The ResourceManager will add the value of this property to its scope: "urn:amigo" and "urn:hostname". The scope can be then used to limit the discovery of resource managers to only the local machine.

The remaining properties define the position of the interaction resource within the given location and thresholds used when mapping user distance to interaction zones (as described in section 2.2).

3.3 Palantir Presence Management System

3.3.1 System requirements

All Palanir components require:

- Java J2SE run time (version 5 or higher), OSGi R3 platform, Amigo middleware bundles.

Palantir service also requires:

- XMPP/Jabber server (we recommend eJabberd, a simple open source xmpp server).

PalanirGUI also needs:

- PC with a display, mouse or other pointing device.

3.3.2 Download

All files, software, and bundles can be found in the amigo Palantir repository at: http://amigo.gforge.inria.fr/obr/palantir/

http://amigo.gforge.inria.fr/obr/palantir/oscar_Palantir.zip

3.3.3 Install

Install an OSGi platform with Amigo middleware: install the obr bundle from Oscar, then install amigo_ksoap_export and amigo_wsdiscovery with dependencies using the obr (see the Amigo OSGi framework user's guide for details).

See the online tutorial explaining how to install the Palantir components and how to run the application:

http://amigo.gforge.inria.fr/obr/palantir/HowTo-Palantir.htm

3.3.4 Configure

Configuration needed to run the Palantir Core bundles:

- Install an XMPP/Jabber compliant server accessible via IP unicast
- In each house deploy the following services on a single OSGi platform:
  - *api / interaction_ws / aggregation_ws* (interfaces and tools for databinding)
  - *interaction_module / aggregation_module / gateway / admin*
  - The *palantir_admin* will start a **palantir instance** for each user of the home
- On each interaction device (e.g. display screen) start the Palantir GUI bundles:
  - *interaction_ws*
  - *Palantir_gui_impl*
- For each ambient application that uses Palantir deploy:
  - *aggregation_ws* and *interaction_ws*
- To "manually" compose the PalantirGUIs with the instances of Palantir you can use the following composer services:
  - *Palantir_standalone_composer* (bundle for composing a Palantir instance with a Palantir GUI based on properties published by the two WS)
  - *Palantir_gui_composer* (a GUI for composing any Palantir instance with a Palantir GUI)
- To compose the PalantirGUIs with the Palantir service based on user context you can use the following composer service:
  - *Palantir_CMS_composer*

More details on Palantir configuration procedure are available in the online tutorial:

http://amigo.gforge.inria.fr/obr/palantir/HowTo-Palantir.htm

3.3.4.1 PalantirGUI service

Create a Palantir-gui.properties file in the folder where you run the OSGi platform. When the Plantir GUI is started, the GUI reads this file, and publishes all properties read from it as service metadata. This allows a composition system to choose among several GUIs possibly present on the network and compose them with a right Palantir service.

For more details on Palantir and Palantir GUI deployment please see the online tutorial:

http://amigo.gforge.inria.fr/obr/palantir/HowTo-Palantir.htm

3.3.4.2 Palantir UI test

By default, the Palantir UI test will search for two Palantir UIs published on the network with ServiceType=PalantirGui and a property “user” set to Tom and Jerry. It initializes those two GUIs with the contents of files Tom-digest.xml and Jerry-digest.xml, then ensures that the actions initiated by “Tom” are delivered (if needed) to “Jerry”, and vice versa.

You must provide one digest file per user, named by default Tom-digest.xml and Jerry-digest.xml. Consistent digest files come with the palantir tutorial.

You may change this default behavior by setting the bundle property palantirtest.user to any comma-separated list of names. Provide also a digest for each of these users, in a file named <user>-digest.xml.

3.4 Ambience Sharing application

3.4.1 System requirements

Recommended hardware:

- Any Pentium IV class PC with a graphics processor supporting OpenGL 2.0 and with network connectivity
- Logitech QuickCam 4000 Pro USB webcam or equivalent

Software:

- OpenGL, GLEW and glut libraries
- Java Runtime Environment 1.6 or higher, Amigo middleware and bundles
- SIP proxy server (tested with Brekeke and openser)

3.4.2 Download

The packaged software is available from gforge:

[amigo] / WP7 / AmbienceSharing / AmSharingApp

[amigo] / WP7 / AmbienceSharing / AVT

[amigo] / WP7 / AmbienceSharing / AmigoVisioService

3.4.3 Install

AmSharingApp and AmigoVisioService installation:

- Install an OSGi platform with Amigo middleware
- The Palantir, Scheduler and ResourceManager services should be running in the environment
- Install and deploy AmigoVisioService bundle on each PC running the AVT software
- Install and deploy SipJavaStack and AmSharingApp bundles

AVT installation:

- Unzip the AVT.zip file.
- Run the ftlicenseclient.exe. This program will show a popup window with the HostID for your computer. Send this HostID to stan.borkowski@ornage-ftgroup.com or gilles.privat@orange-ftgroup.com France Telecom will provide a license file for use only on your PC. The license file should be saved in the same directory as avt.exe. Note that the program will fail to start without a valid license file.

3.4.4 Configure

3.4.4.1 Ambiance Sharing application composer

The AmSharingApp service requires the following properties defined on the Oscar platform:

```
username          = john_phi
presencemodel-url = http://myServer/presencemodel.xml
default-device    = roomA1_john_phi
default-room      = roomA
default-contact   = alain_ft
outboundProxy     = 192.168.1.100
proxyPort         = 5060
listeningPort     = 5061
```

The `username` defines the owner of the Ambiance Sharing application. There should be one deployed AmSharingApp instance per user.

The `presencemodel-url` property specifies the address of the presence model that is submitted by AmSharingApp to the Palantir service.

The `default-device/room/contact` properties define the default endpoints for a communication. Typically, this information is not needed as it is provided by WP4::CMS.

The `outboundProxy` and `proxyPort` parameters specify the IP and the listening port of the SIP proxy server to which the Ambiance Sharing application should register.

The `listeningPort` defines on which port the AmSharingApp service listens for SIP messages.

3.4.4.2 Adaptive Video Transmission module

Each AVT module reads a configuration file at startup. This file should contain the following entries:

```
[General]
[SIPConfiguration]
username1=roomA1_visiodevice
proxyURI=sip:192.168.1.100
SIPport=5060
```

`Username1` defines the id used by the AVT to register to the SIP proxy server. This id should be typically related to the location of the device running the AVT, because in Ambience Sharing the interaction devices and the video streaming/decoding software can be used by various users.

The `proxyURI` and `SIPport` properties define the SIP server to use (must be the same as for the AmSharingApp service).

3.4.4.3 AmigoVisioService

The AmigoVisioService module requires only one property defined on the Oscar platform; `sip-id` that must be the same as the `username1` property of the AVT for which the AmigoVisioService acts as a proxy.

3.5 Activity Sharing components

3.5.1 System requirements

The client and server applications run on conventional PC hardware.

The client systems run on a P4 2 GHz (or AMD equivalent) having 1024 MB memory. CPU load during the demo varies between 30% and 80% (including video encoding); OS: Windows XP with SP2. The client PCs need a video card that can generate video in the format 1280 x 768; we used nVidia GeForce FX 5200 (128MB) and ATi Radeon 9250 (128MB).

The server PC has a P4 processor (2 GHz) and 512 MB memory; the CPU load is around 5 % so a slower CPU could also be used. OS: Linux Fedora Core 5, with all latest updates (as of 24/04/2007), kernel 2.6.15.

3.5.2 Download

The binary release of the software is available on request from leo.rozendaal@philips.com or christian.quaedvlieg@philips.com

3.5.3 Install

The following software (and hardware) needs to be installed before starting the application:

Server

- Linux, version: Fedora Core 5, with all latest updates (as of 24/04/2007), kernel 2.6.15.
- XMPP/Jabber server: Wildfire 3.2.4; this is an open source jabber server that is written in JAVA. The jabber protocol is used for communication between amigo clients.
- SIP server: Asterisk 1.2.6 (with TCP Patch); Asterisk is a popular open source SIP server with many options and plug-ins. The sip protocol is used for voice and video communication between amigo clients.
- Ruby on rails; The Ruby on rails web framework is used to create an HTTP interface for the amigo backend.
- MySQL is the popular open source database server. Openfire, Asterisk and Ruby on Rails use a shared database. This way we only need one data storage point for user accounts which ensures data integrity.
- Mongrel server; Ruby on rails uses mongrel as application server. Several mongrel processes are run on different ports. A proxy/load balancer in the Apache webserver routes http requests to the mongrel processes.
- Apache webserver; Apache acts as a proxy and load balancer for the ruby on rails HTTP interface. When an http requests is made apache sends the request to a available mongrel process.

Clients

- Windows XP, with SP2
- **Video splitter** providing video streams for the Activity Sharing applications. In current demo setup we can use either Philips' dedicated software or a version of the AVT described in sections 2.4 and 3.4 of this document.

- **FireFox browser.** It is recommended to use Firefox 1.5.8, Firefox 2.0.0 or Firefox 1.5.0.9 because not every extension is working in the latest version of Firefox. *http://www.mozilla-europe.org/nl/products/*
- **NotifSocket**. To install the notifsocket-plugin you need to; (1) copy the xpt-files to the subdirectory components that you find in the folder Mozilla Firefox, (2) copy the .dll-files to the subdirectory plugins that you also find in the folder Mozilla Firefox, and (3) restart Firefox
- **Keyconfig**. Download from *http://extensionroom.mozdev.org/more-info/keyconfig*. It is a Firefox-extension, which can be used to disable keyboard-shortcuts in Firefox. Note: keyconfig is not (yet) working in Firefox 2.0.0.2. You can install the extension by dragging "keyconfig.xpi" to a Firefox window. After the installation you need to restart Firefox. In the menu-entry tools you will now find an entry keyconfig. In this window you need to disable the following entries:
    - Help contents (F1)
    - Find next (F3)
- **Cursor hider**. It is a shareware program that is capable of hiding your cursor when it is inactive. You can download a 30 days trial version (for continued operation, you have to pay a small amount). After installation cursor hider will start automatically and will hide your cursor after an inactivity of 3 seconds. *http://www.softexe.com/cursorhider.html*
- **Philips Screen font**. To get the best appearance of the application you need to install the Philips font. ***NOTE – these fonts may be used only in the scope of the Amigo project, not elsewhere***. To install the screen font one should:
    - Open fonts in Control Panel
    - Drag the four files to the fonts-directory
- **Remote control receiver and software**. Buy a remote control receiver and associated Windows reception application. We use a RedRat, from *http://www.redrat.co.uk*, which connects to a USB interface and can handle a wide range of infrared remote controls. The RedRat comes with an application that can map key presses on a remote control to key-events. You have to "learn" the infrared patterns for the remote control that you use. The supplied file corresponds to the RC6-patterns generated by the remote control for a Philips Streamium product.
- **VLC**. In spite of the name VLC = VideoLAN client, it also contains an encoding and server part, all downloaded in a single package from *http://www.videolan.org*. Current version on the website is 0.8.6; both versions 0.8.5 and 0.8.6 have been tested and you can use either version. For configuration details, see next section. For the **sending** side, VLC can be used to encode and stream the video from a webcam; alternatively, another video streaming (like the eConf based streamer presented in section 2.4.1) is also possible. For the **receiving** side, VLC must be used, independent of the choice for the sending side.
- **Webcam**. If needed a webcam driver has to be installed. The included driver is for the Philips SPC 600NC. This step can be skipped if another streaming source is used.

3.5.4 Configuration

3.5.4.1 MySQL Database

Since we use 1 database for both servers we need to create and setup the database first. Open a console on the server where you want to run your mysqlDatabase and run the following commands:

```
mysqladmin create amigo_production;
```

The database is now created. We need to populate the database with the tables and data used by Asterisk and Openfire. On request we can provide you with a database dump (amigo_production.sql). This file contains all the data you need. SCP or paste the file to the server and run the following command:

```
cat openfire_mysql.sql | mysql amigo_production;
```

This command dumps the database schema and data in the newly created database. There are many tabels created but only the table "jiveUser" is used by both servers. In the table jiveUser there are a few columns that are interesting:

Name	Properties
nickname	Nickname of user, used by Asterisk
name	Userlogin, used by Asterisk (same as username)
md5secret	Used to store md5 hashed Asteriks password. Not used due incompatibilities with Openfire
username	Username, used by Openfire (same as name)
secret	Password, used by Asterisk (same as password)
password	Password, used by Openfire (same as secret)
encrypted_password	Encrypted password used by openfire. Not used due incompatibilities with Asterisk

The table extensions are used to store "extensions" for Asterisk.

Extensions are used by Asterisk to bind a username to a "telephone" number. Each user has one (or more) extension(s).

3.5.4.1.1 Setting up Openfire to use MySQL

JDBC Drivers

The JDBC driver for MySQL is bundled with Openfire, so you do not need to download and install a separate driver. In the Openfire setup tool, use the following values:

- driver: `com.mysql.jdbc.Driver`
- server: `jdbc:mysql://localhost/amigo_production`

Setup Instructions

1. Make sure that you are using MySQL 4.1.18 or later (5.x recommended)
2. Start the Openfire setup tool, and use the appropriate JDBC connection settings.

3.5.4.1.2 Setting up Asteriks to use MySQL

Asterisk does not come by default with MySQL bindings. We need to install them. Make sure you have a working Asteriks installation before proceeding. First we need to get the asterisk-addons source code from the asterisk project subversion repository.

```
cd /usr/src
svn checkout http://svn.digium.com/svn/asterisk-addons/branches/1.2
asterisk-addons-1.2
```

Now we have the source code we can begin compiling:

```
cd asterisk-addons-1.2
make clean
make
make install
```

Now the MySQL bindings are installed we can configure the database.

We included the configuration files in the Amigo project files. You need to copy/overwrite the following files to the Asterisk directory (/etc/asterisk).

- res_mysql.conf (mysql database configuration)
- extconfig.conf (table to database binding)
- extensions.conf (database extension binding)
- sip.conf (database user binding)

Now restart Asterisk and Openfire.

3.5.4.2 Ruby on Rails production server installation

Installing and configuring a ruby on rails production server can be quite hard. Therefore is included this chapter that will explain how to install and configure the amgiobackend application. Before we start:

Make sure you have a working Fedora core installation with the following services installed:

- Asteriks
- Openfire
- MySQL
- Ruby (the programming language that RoR uses)
- Ruby gems (Ruby packaging system (like Debian apt-get))
- Apache 2
- SVN server with the Amigo backendcode on it

Now we need to install Mongrel and its dependencies. Mongrel is an application server used to run Ruby on Rails applications. Mongrel cluster is an extension on Mongrel that allows you to run multiple mongrel processes which is necessary when running a production server.

Open a SSH session on the production server and run the following commands:

```
sudo gem install daemons gem_plugin mongrel mongrel_cluster rails --
include-dependencies
```

Now mongrel is installed we need to configure Apache. First we disable the default apache config: `sudo mv welcome.conf welcome.conf.disabled`

Now scp/copy the following configuration files (see 3.5.2 how to download) to /etc/httpd/

- amigobackend.common
- amigobackend.conf
- amigobackend.proxy_cluster.conf
- amigobackend.proxy_frontend.conf

Apache should now be configured but we still need to create a new system user that runs the mongrel server: `sudo useradd -g wheel app`

The new user should have sudo rights so make sure it is configured in the /etc/sudoers file. You can now start apache and deploy the amigobackend application.

3.5.4.3 Deploying the amigobackend application

Now the server is configured we can deploy the application. The following steps have to be run on a developer machine that runs ruby on rails. We first need to get the source code from the svn server (see 3.5.2 how to download these files). Please contact Philips Innovation Lab to get the source code.

In our situation the SVN server runs on the production server but it can run on another server. Now we have source code we have to check some configuration files: Open database.yml and make sure the database configuration for the production server is valid. Open deploy.rb and check if all settings are correct. Read the comments for more explanation. If you don't have Capistrano and palmtree already installed on your machine you need to install them first:

```
gem install capistrano palmtree
```

We can now begin with deploying the application:

```
cap deploy:cold
```

This command will create the necessary file structure on the production server and start the mongrel servers. If everything went well you can now open your browser and surf to http://192.168.2.101/. If you make changes to the amgiobackend and you want re-deploy the application you have to commit the changes to the svn server and run the following command:

```
cap deploy
```

This command will check out the latest source code from the svn server on the production server and restart the mongrel servers.

3.5.4.4 Networking (NAT) setup

For connections between various labs via the internet, the machines are typically connected through a router with a NAT (Network Address Translation) function. The NAT function for both end-point machines should have port-forwarding enabled (1234). This means that the stream on port 1234 coming in on the NAT router is forwarded to the client in the local network that is listening to that specific port (VLC; 1234). This probably is very difficult for the NAT router in a typical company network, since tight rules are applied to these networks. A direct connection to the outside world is preferred, with a “normal” home router – in such a home router, the port forwarding can be controlled easily.

For testing the streaming and NAT configuration over the Internet, if you want to check if the stream is received on the client side, try to play (originate) the stream from a totally different PC on the internet (e.g. your PC at home) rather than another PC in the same sub-network, since this is the only way that you can test the NAT setting. We used Remote Desktop on a private PC to receive the stream.

3.6 Awareness Globe

3.6.1 System requirements

- Server PC with Windows XP SP2
- Two client PC’s with Windows XP SP2
- Tablet PC with Windows XP Tablet Edition
- RFID Hardware (Sensite solutions HBL100)
- Wireless router configured DHCP (like. 192.168.1.*)
- RF Positioning (WP4::CMS)

3.6.2 Download

All software and configuration files are available on gforge at:

[amigo] / WP7 / ActivitySharing / AwarenessGlobe

3.6.3 Install

To run the Awareness Globe the following software has to be installed. How to install is described per package. Please download the AwarenessGlobe archive from the Amigo gforge, extract it and follow the instructions below.

<u>Oscar OSGI Framework</u>

Documentation about installing the Oscar framework can be found in ‘/documentation/Amigo_user_guide_V2.doc’. Oscar has to be installed at the server PC.

<u>Microsoft EMIC Framework</u>

To install the Microsoft EMIC Framework a simple setup is located in ‘/dependencies/ WSDiscoveryFramework-Build15674.msi’. Run this setup at the server PC.

<u>UMPS</u>

User modeling and profiling services are used to save preferences and statistics from users. To use these functionalities a clean install of the UMPS services will be installed. Run

'UMPS.msi' from the '/dependencies' directory. The UMPS services will run from the server PC.

Openfire

Communication between the services and interfaces is done by XMPP. XMPP is a chat protocol based on XML. To install such a XMPP/Jabber server, run the 'openfire_3_4_2.exe' executable from '/dependencies'. Install Openfire on the server PC. (Default login, username 'admin' password empty!)

XAMPP

To centralize the interfaces a webserver have to be installed at the server PC. XAMPP is a light-weight easy-to-install webserver. The installation can be found in '/dependencies'. Run the setup 'xampp-win32-1.6.4-installer.exe' at the server PC.

3.6.4 Configure

Some of the installed components have to be configured for our needs.

3.6.4.1 Openfire

Click on 'Launch admin' to go to the administration panel of the Openfire server. Navigate to the tab 'Plugins' and upload the two *.jar files from '/dependencies/plugins/xmpp'.

The Broadcast plugin is used to send messages to all users. The User import export plugin will be used to insert all users at once.

Now click the tab 'Users/Groups' and choose 'Users import & export' from the submenu. Import the file called 'AmigoUsersXMPP.xml', it can be found in '/dependencies'. 18 users will be imported, see 'User summary' to list the users.

All users have to be in the same group. Create a group called 'amigo' and make sure all users are members of this group. See 'Group Summary' to list groups, click the 'amigo' group and enable 'contact list group sharing' followed by 'all users'. Members of the group 'amigo' are listed at the bottom of the page.

Finally set the servername to 'amigo'.

3.6.4.2 UMPS

The UMPS services use a Microsoft access database to save the profiles. Overwrite these database files with those in the directory '/software/UMPS database'. Default UMPS installation directory is 'C:\UMPS', put the files in the directory 'DB'.

3.6.4.3 Mozilla Firefox

Mozilla Firefox is used as default CE-HTML browser, because it has many configuration possibilities. If not yet installed surf to http://www.mozilla-europe.org/nl/products/firefox/ The browser have to be installed at the client PC's.

Security

In order to use the CE-HTML standard, some of the security options of a standard browser should be disabled. To change these settings go to 'about:config' and filter on 'signed'.

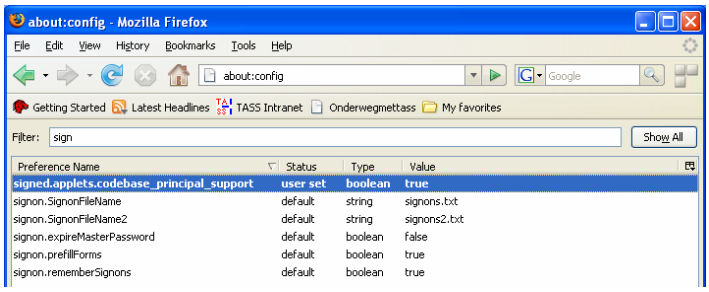

Figure 14: Set signed.applets.codebase_principal_support

Set the variable 'signed.applets.codebase_principal_support' to 'true'. This will add extended privileges to the user.

Firefox uses profiles to provide a different look and feel for every user on the system. For this demo only one profile will be available, so we use the default profile.

The profiles are located in 'C:\Documents and Settings*********\Application Data\Mozilla\Firefox\Profiles**********.default'.

We have to add privileges to load data from another PC in the network. This can be done by manually adding these privileges to the preferences file.

```
user_pref("capability.policy.XMLHttpRequestToAnySite.XMLHttpRequest.open", "allAccess");

user_pref("capability.policy.XMLHttpRequestToAnySite.sites",      "file://      http://localhost
http://**SERVER-IP**");

user_pref("capability.policy.policynames", "XMLHttpRequestToAnySite");

user_pref("capability.principal.codebase.p0.granted",                     "UniversalBrowserWrite
UniversalBrowserRead");

user_pref("capability.principal.codebase.p0.id", "http://**SERVER-IP**");

user_pref("capability.principal.codebase.p0.subjectName", "");


user_pref("capability.principal.codebase.p1.granted",                    "UniversalBrowserAccess
UniversalBrowserWrite UniversalBrowserRead CapabilityPreferencesAccess");

user_pref("capability.principal.codebase.p1.id", "http://**SERVER-IP**");

user_pref("capability.principal.codebase.p1.subjectName", "");


user_pref("capability.principal.codebase.p2.granted",                     "UniversalBrowserWrite
UniversalBrowserRead");

user_pref("capability.principal.codebase.p2.id", "http://localhost");

user_pref("capability.principal.codebase.p2.subjectName", "");


user_pref("capability.principal.codebase.p3.granted",                    "UniversalBrowserAccess
UniversalBrowserWrite UniversalBrowserRead CapabilityPreferencesAccess");

user_pref("capability.principal.codebase.p3.id", "http://localhost");

user_pref("capability.principal.codebase.p3.subjectName", "");


user_pref("capability.principal.codebase.p4.granted",                     "UniversalBrowserWrite
UniversalBrowserRead");

user_pref("capability.principal.codebase.p4.id", "file://");
```

```
user_pref("capability.principal.codebase.p4.subjectName", "");


user_pref("capability.principal.codebase.p5.granted",                    "UniversalBrowserAccess
UniversalBrowserWrite UniversalBrowserRead CapabilityPreferencesAccess");

user_pref("capability.principal.codebase.p5.id", "file://");

user_pref("capability.principal.codebase.p5.subjectName", "");
```

Before editing this file, make sure Firefox is not running. Add the user preferences to the file, and replace **SERVER-IP** with your servers PC ip-address.

Plugin

Normal browsers are based on user requests, and can't receive information from the outside without user interaction. Our browsers have to be extended with the plugin 'notifsocket' to act like an event driven browser. Copy the two files from '/dependencies/plugins/notifsocket' to 'C:/Program Files/Mozilla Firefox/plugins'. After restarting the browser, enter 'about:plugins' and make sure the notifsocket plugin is listed.

NotifSocket Plugin for Mozilla/Firefox version 1.4

File name: npNotifSocket.dll
npNotifSocket

MIME Type	Description	Suffixes	Enabled
application/notifsocket	npNotifSocket		Yes

Figure 15: NotifSocket plugin

Add-ons

Add-ons can be found in '/dependencies/plugins/other firefox'. The *.xpi file will be installed when dragged into a browser. Install the FullFullscreen add-on from the Mozilla website.

Go to Tools → keyconfig to disable the F1, F2, F3, and F4 shortcuts in Firefox. These keys will be used by the remote controls.

3.6.4.4 Services

The following services are necessary, copy these from '/software' to any place on the server PC.

AvailableDevicesPublisher

Simple service to set and get status/locations of services and devices in the network.

MiniPalantir

Simple service to set and get status/locations of users in the network.

Configuration: edit variables in config.xml (webserver, xmppserverip address)

PokerGame

Service to handle user action and control poker logic.

Configuration: edit variables in config.xml (webserver, xmppserverip address)

3.6.4.5 Awareness globe

When all other software is installed and configured properly, all CE-HTML files can be copied from the '/source/root_www' directory to the local root of your webserver. (E.g. c:/wamp/www, c:/xamp/htdocs)

3.6.5 Run

The sequence of starting the services is very important, start all services or applications in the following sequence.

1. Oscar (oscar.bat)
2. AvailableDevicesPublisher.exe
3. PokerGame.exe
4. Minipalantier.exe
5. XAMP or other webserver
6. Openfire

Now open a Firefox browser and enter *'http://localhost'.* The window will resize to 800*480 which is the resolution of the tablet PC. The size can be set in the file *'index.html'* located in the root directory of your webserver.

3.7 Social Radio

3.7.1 System requirements

- Two PCs with at least 2 USB Ports on each of them, connected over TCP network
- To play the music a sound card is necessary
- Installed JRE version at least 1.5
- Correctly installed Java Media Framework, with mp3 plug-in[5]
- A Logitech Gamepad Driver[6]

3.7.2 Download

The Social Radio Demonstrator application is composed of four depended applications: the ArtifactController, the Social Radio Demonstrator, the Social Radio Amigo services and the modified music player jlGui. The first one is used to establish the communication with artifacts witch is used for controlling and monitoring of the artifact states. The Social Radio demonstrator application is a GUI to demonstrate the behavior of the project in the real life. The Social Radio service uses Palantir to propagate the user presence information between the two Amigo installations. The modified music player is used to generate a music stream and to send the stream to the Social Radio Demonstrator.

Both projects can be found in the sourcecode repository of the gforge server under:

[amigo] \ WP7 \ Socialradio

[5] JMF MP3 Plugin: http://java.sun.com/products/java-media/jmf/mp3/download.html

[6] Logitech: http://www.logitech.com/index.cfm/441/287&cl=gb,en

3.7.3 Install

Connect all transceivers with a computer.

Install the Logitech Gamepad Drivers witch can be found here:

http://www.logitech.com/index.cfm/downloads/categories/DE/DE,CRID=1784

To run the program start the ArtifactController and press the "Detect all force feedback devices" on the top of the Window, then start the UI by running the *SocialRadioDemonstrator.exe.* Alternatively, you may start only the Social Radio UI without the artifact communication and change the states of those manually.

3.7.4 Configure

The GUI for controlling the Social Radio application is shown in Figure 16. It can be run either with or without the SR artifacts. Each buddy is assigned to one of three "rooms". The person assigned to the room and its state can be changed by using the Person and State Combo boxes in the corresponding room boxes. The color of the artifact is bound to the person and defined by the configuration profile.

It is required that the names of configured users are identical with the names, in Palantir system. The state of the Artefact is presented as an icon in the *Artefact State* group box.

Four different Artefact lie states are possible:

- OFF – the artefact is lying on the colour stripe. In this position the artefact is turned off and will neither play music nor light.
- Stripe on top – the artefact is lying with the colour stripe on the top. It will be lighting if the associated person is at home OR listening to music. No music is played in this state.
- Bridge – the artefact is forming an arc on the underlying surface. The music the associated person is listening will be played if the artefact is lying in this position. If the *Laptop Mode* (for PC with 1 sound card, chosen by default) is chosen, then the music will be played from a single sound card. If more then one artefact is lying in this position then the played music will be scheduled and artefact, which music is played, is lighting. If the associated person does not hearing any music (*do not listening to music* state) then the artefact will show the presence of the person via turning the light on.
- Moon – the artefact is lying on the side (looks like moon). If the associated person is hearing music, then the artefacts light is turned on. No other actions.

To hear the music from remote users you need to register to the remote Music Player (for streaming and event propagation). Pressing the *Connect to Friends* button register the remote music player with user.

Pressing the *ACP Emulator* button emulates messages from Palantir or the Artefact Controller.

For help using the jlGui you may refer to the official documentation[7].

[7] jlGui user Guide: http://www.javazoom.net/jlgui/documents.html

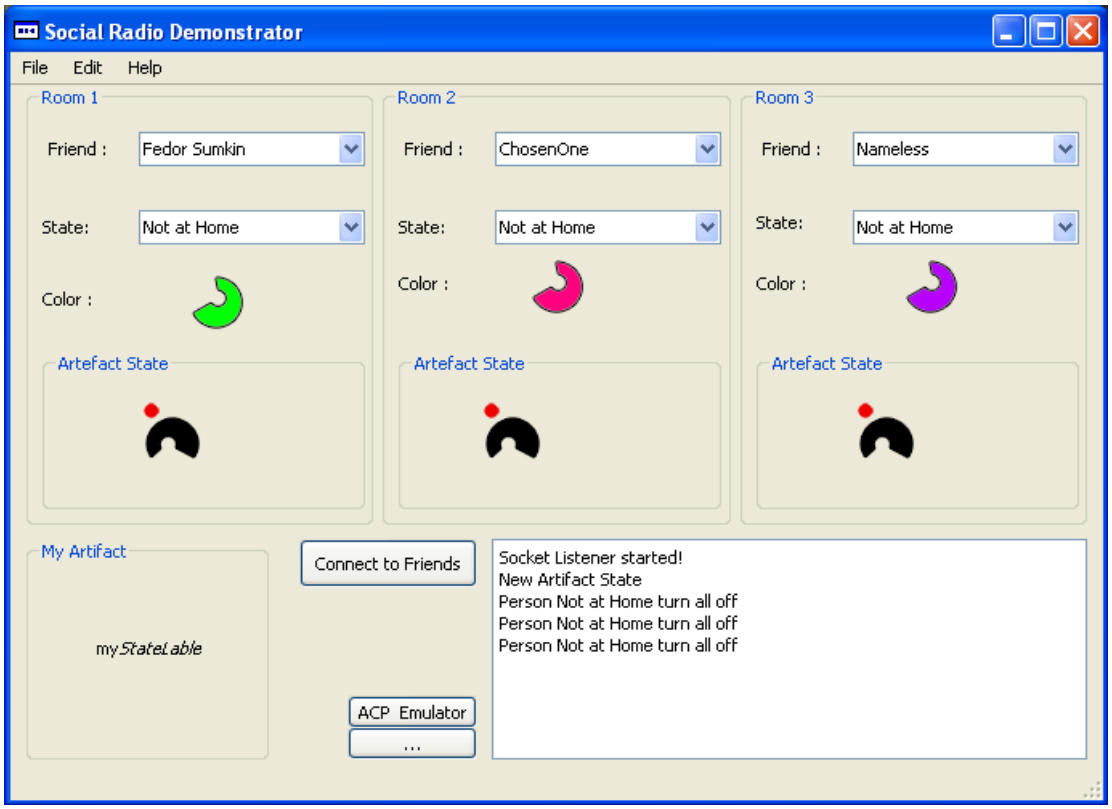

Figure 16: The Main Screen of the Social Radio Demonstrator

To configure the artifacts start the configuration dialog from the Edit menu. The dialog has two tabs. The *Persons* tab is used to configure the data about the friends, wherewith the artifacts can be associated. Each person contains 3 fields: name, host and music stream port. Name should be the same, as used in Palantir. Host could be the host name or the IP address of the remote user's machine. The host and the port information are used to locate the remote machine, for listening to user streams.

The *Artifacts* tab allows configuring the artifact-user associations as well as the hardware configuration of each artifact. Each artifact has sound and an IO numbers. These numbers are used to identify the soundcard and the gamepad used to control the artifact. Unfortunately The JMF does not allow choosing the sound card for an output, so the streamed output is done over a main sound card. To choose the correct light number you may use the "*Turn Light On/Off*" button. One artifact is marked by a radio button as *My artifact*. This artifact is used to control the own shared state.

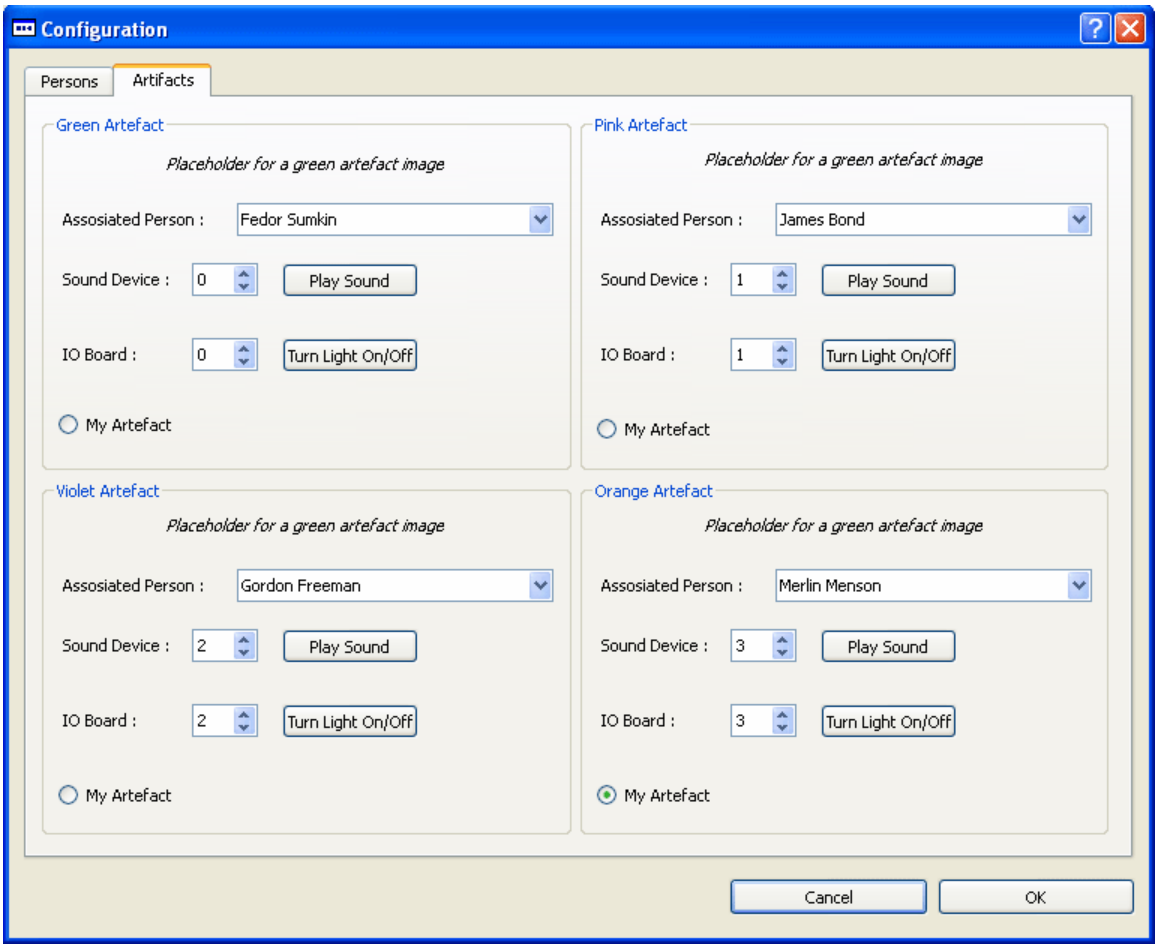

Figure 17: The configuration dialog

3.8 The Board Game

For installation and deployment instructions please refer to D6.4.

3.9 Feeling@ application components

3.9.1 System requirements

- Conventional PC with JVM 5.0 or higher (JVM 6.0 for the Conference Manager)
- Amigo Programming & Deployment Framework from WP3

System requirements vary between components. For detailed requirements specification please see Environment and Platform paragraphs in Components Overview section.

3.9.2 Download

All software is available from Gforge:

[amigo] /WP7/Feeling@/Core/trunk

[amigo] /WP7/Feeling@/DBPool/trunk

[amigo] /WP7/Feeling@/MailService/trunk

[amigo] /WP7/Feeling@/XmlRpc-3.0/trunk

[amigo] /WP7/Feeling@/RFIDReader/trunk
[amigo] /WP7/Feeling@/RFIDReader.Net/trunk

[amigo] /WP7/Feeling@/GestureServiceManager/trunk
[amigo] /WP7/Feeling@/VoiceServiceManager.Net/trunk

[amigo] /WP7/Feeling@/SharedOrganizer/trunk
[amigo] /WP7/Feeling@/SharedOrganizer.GUI/trunk

[amigo] /WP7/Feeling@/SketchPresentation/trunk
[amigo] /WP7/Feeling@/SketchPresentation.GUI/trunk

[amigo] /WP7/Feeling@/UserNotificationMessenger/trunk
[amigo] /WP7/Feeling@/ConferenceManager/trunk

[amigo] /WP7/BrowserService/trunk

Software that is not included in the Gforge repository is freely available from each publisher's web site:

Sun Java SE JDK: http://java.sun.com/javase/downloads/index.jsp

Microsoft .NET Framework 2.0: http://www.microsoft.com/downloads/

Oscar 1.0.5 (OSGi platform): http://forge.objectweb.org/projects/oscar/

MySQL Server 5.0: http://dev.mysql.com/downloads/

OpenLaszlo 4.0.0: http://www.openlaszlo.org/download

OpenSER SIP server: http://www.openser.org/mos/view/Download/ (open source)

or

Brekeke SIP server: http://www.brekeke.com/products/products_sip_2.php (commercial)

Openfire Jabber server: http://www.igniterealtime.org/downloads/index.jsp#openfire or any other XMPP compliant server.

For the Oscar platform, an already preconfigured instance is available at:

http://amigo.gforge.inria.fr/obr/tools/index.html

All WP3 middleware components (including specifically the OSGi Programming & Deployment Framework and the .NET Programming & Deployment Framework) are available at:

http://amigo.gforge.inria.fr/home/index.html

All required OSGi bundles can be downloaded and installed in the Oscar GUI from the following URLs:

Oscar Bundle Repository: http://oscar-osgi.sf.net/repository.xml

Amigo Bundle Repository: http://amigo.gforge.inria.fr/obr/v2/repository.xml

Amigo CMS/CASD Bundle Repository: http://core.lab.telin.nl/~amigo/obr/repository.xml

ANS Bundle Repository: http://amigo.gforge.inria.fr/obr/ans/repository.xml

Palantir Bundle Repository:
http://amigo.gforge.inria.fr/obr/palantir/repository.new/repository.xml

The WP6::HomeAgenda application is available from Philips QuickPlace website:

https://share.philips.com/quickplace/amigo/pagelibrary85256fea00434808.nsf/h_Index/2774A9D75973AC90C12572FA004D5B5F/?OpenDocument&Form=h_PageUI

The WP4::IUS:UMPS services (.NET based) are currently available from Gforge:

[amigo] / ius / user_modeling

or as a Windows installable package (UMPS.msi).

The WP4 Voice Services (.NET based) are also available from Gforge:

[amigo] / ius / user_interface / voice_service

but they also require specific speech and recognition engines which are provided directly by the partner in charge of the module.

VTT's SoapBox Gesture Recognizer software is available directly from the vendor along with the hardware device.

France Telecom eConf player is available upon specific license agreement with the vendor.

3.9.3 Install

First install all the software not available from the Gforge repository. This includes the Sun Java SE JDK, the Microsoft .NET Framework, MySQL Server, OpenLaszlo, a SIP server and a Jabber server. Also install the Amigo preconfigured Oscar instance and the .NET Programming & Deployment Framework. Also, be sure to have installed and ready the WP4 UMPS executables, the WP4 Voice Service packages (including the speech and recognition engines), VTT's Gesture Recognizer software and France Telecom's eConf player.

Start the Oscar OSGi server and make sure the following bundles are installed (from the OBR tab) in the following order:

- Service Binder
- Servlet
- HTTP Service (+ Amigo mods)
- log4j
- commons-logging-1.1
- jena-2.4
- amigo_core
- amigo_stubgen
- amigo_ksoap_binding
- amigo_ksoap_export
- amigo_wsdiscovery

In order to do so, you might need to configure the "bundle.properties" file under the Oscar /lib folder with the following entry:

oscar.repository.url=http://oscar-osgi.sf.net/repository.xml \
http://amigo.gforge.inria.fr/obr/v2/repository.xml \
http://core.lab.telin.nl/~amigo/obr/repository.xml \
http://amigo.gforge.inria.fr/obr/ans/repository.xml

From the Gforge checkout folders install the base bundles:

- db-pool-dbcp ([amigo] /WP7/Feeling@/DBPool/trunk/db-pool-dbcp.jar)
- javamail-service ([amigo] /WP7/Feeling@/MailService/trunk/javamail-service.jar)
- xmlrpc-3.0 ([amigo] /WP7/Feeling@/XmlRpc-3.0/trunk/xmlrpc-3.0.jar)

From the OBR tab deploy the WP4::IUS:CMS bundles:

- context-broker-service
- Context Source Manager
- Context Helper

From your download folders deploy the WP6::HomeAgenda bundles:

- home_agenda_api
- home_agenda_context_source

From the Gforge checkout folders install the Feeling@ bundles:

- wp7-feeling-rfid-reader ([amigo] /WP7/Feeling@/RFIDReader/trunk/f@rfid-reader.jar)
- wp7-feeling-gesture-service ([amigo] /WP7/Feeling@/GestureServiceManager/trunk/f@gesture.jar)
- wp7-feeling-conference-manager ([amigo] /WP7/Feeling@/ConferenceManager/trunk/f@conference.jar)
- wp7-feeling-shared-organizer ([amigo] /WP7/Feeling@/SharedOrganizer/trunk/f@organizer.jar)
- wp7-feeling-sketch-presentation ([amigo] /WP7/Feeling@/SketchPresentation/trunk/f@sketches.jar
- wp7-feeling-notification-messenger ([amigo] /WP7/Feeling@/UserNotificationMessenger/trunk/f@messenger.jar)

And finally from the Gforge checkout folders install the WP7 Browser Service:

- browser-service ([amigo] /WP7/BrowserService/trunk/browser-service.jar)

Before starting the bundles, please configure them (see section below). When done, bounce the Oscar instance and start all the deployed bundles in their respective order.

For the .NET platform, install or checkout the WP4 UMPS module following the User's or Developer's Guide. Start the Static Modeler (StaticModeler.exe) and the Reasoning Module (ReasoningModule.exe) and configure the desired number of user profiles (see the configuration section below).

Be sure to install also the WP4 Voice Service components following the User Tutorial document. Start the WaveIO (WaveIOServer.exe) and VoiceIO (VoiceIO.exe) executables.

The .NET platform version of the RFID Reader component, after configuration (see section below), only needs to be started from:

[amigo] /WP7/Feeling@/RFIDReader.Net/trunk/RFIDReader.Net/bin/Release/F@RFIDReader.Net.exe

Finally the .NET based VoiceServiceManager, configured accordingly (see section below), can be started from:

[amigo] /WP7/Feeling@/ VoiceServiceManager.Net /trunk/ VoiceServiceManager.Net /bin/Release/ VoiceServiceManager.exe

As for the hardware, make sure the required devices are correctly wired and configured to be reached over an IP network.

3.9.4 Configure

3.9.4.1 Pre-requisites

Follow your chosen SIP vendor documentation to configure the server. As a minimum, the server should be configured to have one user definition (call endpoint) per effective Amigo user plus one for each video terminal station (both eConf or other SIP compliant video recorder/player). For better results disable user authentication since having it enabled might need the application code to be altered, although video terminals will behave correctly in both situations. Be sure to check that the SIP server is ready for RTP relay. For further details on the SIP server setup please refer to France Telecom's Ambient Sharing documentation.

In a similar manner, configure your Jabber server of choice. Create all the profile definitions necessary to match the desired Amigo users. For further details on the Jabber server setup please refer to France Telecom's Palantir documentation.

Please follow the installation and configuration instructions for France Telecom's Palantir application.

3.9.4.2 DB Pool

The DBPool bundle requires RDBMS pool configuration to be defined in the "bundle.properties" file under the Oscar /lib folder. Open the specified file and add the following fragment:

```
#DB Pool service pool list
it.italdesign.amigo.osgi.dbpool.pools=AmigoWP7_SO

#AmigoWP7 pool settings
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.driverClass=com.mysql.jdbc.Driver
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.driverURL=jdbc:mysql://localhost:3306/am_so
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.user=amigo
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.password=amigo
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.poolPreparedStatements=true
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.statementsTimeBetweenEvictionRuns=30000
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.statementsMinEvictableIdleTimeMillis=120000
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.connectionsMaxActive=10
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.connectionsMaxIdle=2
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.connectionsMaxWait=120000
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.connectionsTimeBetweenEvictionRuns=30000
it.italdesign.amigo.osgi.dbpool.AmigoWP7_SO.connectionsMinEvictableIdleTimeMillis=12000
0
```

The entries define a single RDBMS pool called "AmigoWP7_SO" that points to a MySQL Server database instance named "am_so", located on the same machine as the Oscar server, listening at port 3306 and with both username/password credentials set to "amigo". If necessary, change the "*driverURL*", "*user*" and "*password*" lines accordingly. All other settings should not require any modification. For more information on the MySQL instance see the "SharedOrganizer" chapter below.

3.9.4.3 Mail Service

The MailService bundle does not strictly require any further configuration. However to give other services the ability to send emails, at least the basic JavaMail properties should be added to the “bundle.properties” file under the Oscar /lib folder. Consider for example the following fragment:

```
#JavaMail properties
mail.transport.protocol=smtp
mail.smtp.host=smtp.mycompany.com
```

The above entries define SMTP as the default outgoing mail protocol and “smtp.mycompany.com” as the default SMTP server host. Change or add any necessary JavaMail property in order to enable email delivery.

3.9.4.4 RFID Reader

Both the Java and the .NET platform RFID Reader components configure themselves by reading a common XML file. Consider the following XML fragment:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
	<readers>
		<reader id="iPortR2-270969" host="172.16.1.118" port="7001" location="CED" pollingDelay="1000" skipUnresolvedTags="true"/>
	</readers>
	<users>
		<user id="300003733" name="katia.bianciotto@italdesign.it"/>
		<user id="300003751" name="luca.jozzo@italdesign.it"/>
		<user id="300003736" name="patrizio.lionello@italdesign.it"/>
		<user id="300003735" name="valerio.vigna@italdesign.it"/>
		<user id="300003732" name="denis.agnelli@italdesign.it"/>
		<user id="300003737" name="roberto.bolognesi@italdesign.it"/>
	</users>
</configuration>
```

The XML file is divided into two segments. The first, enclosed by the *<readers>* tags, specifies one or more RFID readers configuration. The attributes of the *<reader>* element are:

`id`: an unique reader identifier.

`host`: the reader’s hostname or IP address.

`port`: the reader’s TCP listening port.

`location`: the reader’s physical location. This information is passed over as context information.

`pollingDelay`: number of milliseconds between each subsequent poll of the RFID reader.

`skipUnresolvedTags`: if true ignores any RFID tag that does not match to a specific user.

The second section, enclosed by the *<users>* tags, identifies all users owner of an RFID tag. The attributes of the *<user>* element are:

`id`: the RFID identifier as detected by the reader.

`name`: the Amigo username tied to specified tag id.

The configuration XML can be named freely (one is provided with each RFID Reader component implementation). It’s location depends on the running platform.

For the OSGi based RFID Reader the XML file name is specified in the bundle manifest file under the “Configuration-file” key. Default name is *feig-readers-configuration.xml*. The actual file is searched in the Java system property “user-dir” or “user-home” folders and, if not found, in the Java Classpath.

The .NET implementation, stores the XML file name in the *F@RFIDReader.Net.exe.config* file located in the same folder as the executable under the *ConfigurationFile* setting entry (default is *iport-readers-configuration.xml*). If no path is specified, the XML file is intended to be in the executable folder.

3.9.4.5 Gesture Service Manager

Be sure to have the hardware connected and setup correctly. Switch the SoapBox devices on and start the Gesture Trainer software by VTT. Specify a desired username (any will do, only one at a time will be used at runtime) and select the COM port to which the SoapBox receiver is attached. Proceed with the gesture training by inserting a command name, a command key and a repetition count for each desired command. When done save your work and exit.

The Sketch Presentation application will listen and accept the following commands (note that the list specifies only the command keys, names can be different though it is discouraged):

- **Next**
- **Right** (same as Next)
- **Previous**
- **Left** (same as Previous)
- **Show Left**
- **Show Right**
- **Fullscreen**
- **Select Local**
- **Down** (same as Select Local)
- **Select Remote**
- **Up** (same as Select Remote)
- **Open**
- **Close**

Note that it is not mandatory to train all the specified commands.

The Gesture Service Manager should not need any configuration, though it is possible to change the listening socket port and the receiving buffer size. This is done by editing the *MANIFEST.MF* file found in the jar bundle archive under the *META-INF* folder. The *Server-Port* entry specifies the TCP port which the Gesure Service Manager binds to (default is 7776), while the Buffer-Size entry indicates the data buffer size in bytes (default is 2048).

When done, start the Gesture Service Manager bundle first (as it listens remotely for gesture commands) and then proceed by running Gesture Recognizer software. Select the username used to train the commands previously from the drop down list and specify the COM port to which the SoapBox receiver is attached. In the *Socket server address* field type the hostname or the IP address of the PC which is running the Gesture Service Manager, specifying 7776 as the port number (or whatever has been chosen if customized). As for the protocol checkboxes, choose any desired combination, though leaving them all cleared is suggested.

3.9.4.6 Voice Service Manager

The Voice Service Manager requires the WP4 Voice Service software to be installed and configured according to the relative User Guide. Once both Speech Synthesis and Recognition are installed and configured, proceed by installing the *WaveIO* and *VoiceIO* services. When these are configured as well, the Voice Service Manager can be taken care of.

The software only needs little or no configuration in order to run, and it is already preconfigured to handle voice recognition for the Sketch Presentation application. Locate the *ApplicationResources* folder in the Voice Service Manager project, which contains grammar and speech files. Choose a location for it, though preferably it should be the *C:\Amigo\ApplicationResources* path and copy all the contents there. In the executable directory locate the VoiceServiceManager.exe.config file. Locate the settings entries, these are respectively for:

`RecognizerGrammarPath`: the full path to the *ApplicationResources* folder.

`VoiceServicePublishedPort`: the TCP port used to publish the application's web service.

`RecognitionRepeatCount`: the number of times the recognizer software should try to re-understand user's speech when not understood or not heard.

`RecognitionConfidenceLimit`: the minimum confidence limit above which the recognizer software will accept the speech as recognized correctly.

`DefaultGrammarFile`: the default grammar XML file if no specific one is resolved based on the requestor application.

`DefaultSemanticValue`: the default semantic value if no specific one is resolved based on the requestor application.

`DefaultPromptMessage`: the default text file which will be synthesized as the recognition prompt if no specific one is resolved based on the requestor application.

`DefaultNotUnderstoodPrompt`: the default text file which will be synthesized when the recognition engine cannot understand the user's utterance.

`DefaultNotHeardPrompt`: the default text file which will be synthesized when the recognition engine cannot hear any user speech.

The Sketch Presentation application has specific entries:

`SketchPresentationGrammar`: for the grammar XML file.

`SketchPresentationSemantic`: for the semantic value.

`SketchPresentationPrompt`: for the default prompt text file.

Any new application can be supported by adding three new settings suffixed as *AppGrammar*, *AppSemantic* and *AppPrompt*. These will be picked up automatically.

3.9.4.7 Shared Organizer

The SharedOrganizer component requires a RDMBS instance configured in order to run. To do so create a fresh MySQL database instance called "AM_SO". Optionally create a new database user ("amigo" for example) and grant it full control over the "AM_SO" instance. Finally, from the administrator or the "amigo" user, populate the instance by executing the /res/AM_SO.sql script found under [amigo] /WP7/Feeling@/SharedOrganizer/trunk/. See also the DBPool configuration above.

Also required is the WP4 UMPS with at least one configured user. Please follow the WP4 UMPS User's or Developer's Guide to create new user profiles. When done, from the UMPS GUI add the following profile setting to each user:

Preferences:OtherPrefs:SharedOrganizer:ServiceNumber with a value of 0 (zero).

Optionally if the value is set to be greater than zero (negative numbers are not allowed), then the same number of profile settings must be created as:

Preferences:OtherPrefs:SharedOrganizer:ServiceX with a string value representing a valid SharedOrganizer Web Service URL and where X is a number starting from 0 and ending to the "ServiceNumber" value minus 1 (one).

Finally the Shared Organizer GUI requires some file copying. Locate the /Server/lps-4.X.X/my-apps/ path from the OpenLaszlo Server installation folder. Create an "organizer" folder and copy inside all the files from:

[amigo] / WP7 / Feeling@ / SharedOrganizer.GUI / trunk

Locate the *AMSOActivityBrokerService.lzx* file under the *AMSOLibrary / AMSOremote* path. Open the file with a text editor and find the following entry:

*<xmlrpc name="ActivityBrokerService" service="**http://idg1044.itd.private:8080**/organizerbroker/xmlrpc">*

Change the base URL portion of the service attribute to match the hostname or IP address of the PC which is running the Shared Organizer bundle and the port which the same OSGi *HTTP Service* is bound to.

Start the OpenLaszlo server and access the mock-up from your web browser by pointing to:

http://yourhost.com:8080/lps-4.X.X/my-apps/organizer/SharedOrganizer.lzx

Take note of the URL and open the "bundle.properties" file under the Oscar /lib folder. Add a new entry with the

it.italdesign.amigo.wp7.feeling.shared.organizer.url

key and append the URL value that has been kept aside.

If the OpenLaszlo instance resides on the same machine as the Oscar server, port conflicts might arise (both softwares might use port 8080 as their default web port). OpenLaszlo embedded Tomcat instance can be tackled by editing the file:

[OpenLaszlo install folder]/Server/tomcat-5.X.XX/conf/LPS/localhost/lps.xml

3.9.4.8 Sketch Presentation

The Sketch Presentation GUI requires some file copying too. Locate the /Server/lps-4.X.X/my-apps/ path from the OpenLaszlo Server installation folder. Create a "sketches" folder and copy inside all the files from:

[amigo] / WP7 / Feeling@ / SketchPresentation.GUI / trunk

Locate the *SKPRGestureService.lzx* file under the *SKPRLibrary* path. Open the file with a text editor and find the following entry:

*<xmlrpc name="GestureService" service="**http://idg1044.itd.private:8080**/gesture/xmlrpc">*

Change the base URL portion of the service attribute to match the hostname or IP address of the PC which is running the Gesture Service Manager bundle and the port which the same OSGi *HTTP Service* is bound to.

Similarily, locate the *SKPRVoiceService.lzx* file under the *SKPRLibrary* path. Open the file with a text editor and find the following entry:

*<soap name="VoiceService" wsdl="**http://172.16.10.34:7776**/VoiceService/VoiceService.asmx?WSDL">*

Change the base URL portion of the wsdl attribute to match the hostname or IP address of the PC which is running the Voice Service Manager and the port which the relative web service is bound to.

Finally edit the *settings.xml* file found in the root folder. Locate the two following lines:

<url value="http://127.0.0.1:7070/lps-4.0.6/my-apps/sketches/SKPRData/brera/" />

<remoteurl value="http://127.0.0.1:7070/lps-4.0.6/my-apps/sketches/SKPRData/nasa"/>

The values are the initial URLs where the application will look for metadata information. The Sketch Presentation GUI will retrieve and open a file named *metadata.xml*, which should be located in each of the specified URLs. The XML format for this file is relatively simple and of the following type:

```
<metadata>
  <image>
        <file>1.jpg</file>
   </image>
    <image>
        <file>2.jpg</file>
   </image>
</metadata>
```

Where each *<file>* element contains a reference to a relatively located image file. Supported image types include JPEG, GIF and PNG (the ones supported by OpenLaszlo itself).

Start the OpenLaszlo server and access the mock-up from your web browser by pointing to:

http://yourhost.com:8080/lps-4.X.X/my-apps/sketches/SketchPresentation.lzx

Take note of the URL and open the “bundle.properties” file under the Oscar /lib folder. Add a new entry with the

it.italdesign.amigo.wp7.feeling.sketch.presentation.url

key and append the URL value that has been kept aside.

3.9.4.9 User Notification Messenger

The User Notification Messenger requires the WP6 Home Agenda software to be installed and configured according to the relative User Guide. In particular, in order for the User Notification Messenger to run correctly, a Java class which defines user’s schedules should be executed each time the Home Agenda application starts in order to have all information loaded.

Locate the *Loader.java* file under the project root folder. It contains a sample configuration related to user scheduled time. To represent such information, the VFREEBUSY calendar definition is used. Refer to the http://www.faqs.org/rfcs/rfc2445.html specification for further details. The following properties are used to define the schedule:

`Organizer`: the subject of the defined schedule.

`Contact`: the user which should be notified if the schedule is not respected.

`DtStart` and `DtEnd`: the time frame of the schedule.

`Duration`: the total scheduled busy time in hours.

`Activity`: the type and location of the scheduled period.

The file should be placed where your Home Agenda Client sources are, in the *es.tid.HomeAgendaClient* package. Compile the class and run it each time the Home Agenda bundle starts by passing the Home Agenda Context Source web service URL as an argument (or set it up in the class source itself). Refer to the Home Agenda documentation for further explanations on the subject.

The User Notification Messenger also requires the WP4 UMPS software to have a specific setting for each user email address. The required value should be placed in the *PersonalDetails:EMail* key (to be created if nonexistent). Please refer to the UMPS documentation on how to define and add values to settings.

3.9.4.10 Conference Manager

As a spin-off of France Telecom's Ambience Sharing application, please refer to the relative documentation for configuration procedures.

The Conference Manager requires France Telecom's eConf-based AVT software. Refer to the software's documentation for installation and configuration details. Remember to setup SIP server connection appropriately and assign the correct user credentials created previously as specified in the Pre-requisites paragraph of this chapter.

If hardware support is planned by using Tandberg video stations, be sure these are up to date in regards of the firmware version. Follow the vendor's documentation for installation and configuration procedures. Configure SIP connections using the user credentials created previously as specified in the Pre-requisites paragraph of this chapter. Be sure to use manual for the server discovery property, automatic identification for the server type property and preferably UDP transport.

If using Tandberg support, the Conference Manager bundle needs to authenticate before it can operate with the remote unit (if authentication is enabled) via web interface. The authentication credentials can be provided by adding two entries in the "bundle.properties" file under the *Oscar / lib* folder. These are of following type:

java.net.authenticator.XXX.username=<your username>

java.net.authenticator.XXX.password=<your password>

where the XXX placeholder can be any of Tandberg's web interface URL, hostname or IP address.

3.9.4.11 Browser Service

The browser service only requires a path to the preferred web browser executable. The application will first look in the "bundle.properties" file under the *Oscar / lib* folder for an entry named *it.italdesign.amigo.wp7.browser.cmd* and, if not found, it will prompt to manually insert the path each time it starts.

3.10 Personal Amigo Device

3.10.1 System requirements

See hardware and software requirements in section 2.10.

3.10.2 Download

All files, software, and bundles specific to the PAD can be found in the gforge repository at the following locations:

[amigo] / WP7 / PersonalAmigoDevice

[amigo] / ius / context_mgmt / Bluetooth / trunk

Additional components that are needed: Media Manager related bundles and components (see deliverable D6.4), as well as the Security Service from WP3 (see deliverable D3.5).

3.10.3 Install

Two homes, in the form of two local networks, have to be installed and configured. The routers/gateways of each network have to be connected to the internet, or to each other (back to back).

One network will be the home domain of the PAD; one will act as the visited domain (the guest home for the PAD). Both homes have to have at least one PC for running the necessary bundles that enable the PAD technical scenario as discussed in section 2.10. The home domain has to run at least the IGD service (which offers a web service interface to an UPnP controllable router) and the PAD Service bundle. The visited domain has to run at least the IGD service, the Awareness Globe Listener, the Awareness Globe Context Source, and the supporting bundles from the CMS task (such as the context broker, and the Context Source manager for OSGi). The PC running the necessary bundles in the visited home should also have Bluetooth connectivity with a Microsoft Bluetooth stack, either built-in or with the use of a Bluetooth dongle. This is needed for contacting the PAD using Bluetooth. Make sure that the Oscar installation on this PC has Bluecove bundle installed and intelbth.dll (Bluecove native dll) installed in the System32 folder of windows. The bluecove bundle and intelbth.dll can be found in gforge at [amigo]/ius/context_mgmt/Bluetooth/trunk. Both homes should also run the amigo security bundle. Detailed instructions for setting up the security service can be found in gforge at [amigo]/deployframework/osgi/trunk/amigo_security/docs.

An RFID reader should be installed to provide user identification. The Awareness Globe Context Source includes a simulator of an RFID reader, making it possible to run the scenario without the use of an actual RFID reader (Awareness Globe or otherwise). The simulator offers a User Interface that can be used to configure the (simulated) tag data, and to simulate a swipe of the RFID tag by pressing a button.

The PAD itself needs the IBM J9 virtual machine installed, as well as Oscar. The additional dll and bundles that need to be installed are available on gforge at [amigo]/WP7/PersonalAmigoDevice/PAD. The PAD should also run the jre13 version of the amigo security bundle (instructions can again be found in gforge).

If multimedia is to be exported by the PAD, then the media management components from WP6 have to be installed as well. Please refer to D6.4 for installation and usage instructions for this. Make sure that for the ContentDiscovery component, the PAD related part is enabled by setting the property padactive=true in the contextdiscover.properties configuration file.

There are also some example clients and services in the gforge PAD folder which can be installed to illustrate and test the PAD. The LightBulbClient and service show how a light bulb can be controlled from a visited domain. The PAD Test Client and service is a minimalistic example which writes some input to the console.

3.10.4 Configure

The PAD has to be configured with the URL of the PAD service at home. This can be done by setting the nl.telin.amigo.pad.homeurl property in the bundle.properties file of Oscar running at the PAD.

The credentials for the visited home WLAN have to be configured in the bundle.properties file on the PC running the Awareness Globe listener bundle by setting the globeclient.ssid, and globeclient.wepkey properties. The default ssid and key used when none are specified in the properties are ‘amigo’ and ‘verysecret’ respectively, so alternatively the wireless LAN can be configured to those settings.

The wireless security in the visited home has to be set to WPA Personal; the type of wireless connectivity supported by the PAD.

Start up all the needed bundles in both homes, as well as on the PAD itself. In the 'home' network check that the PAD Service logs some messages to the console about the external IP address, to check that the control of the home router is working properly. To check whether the export functionality itself will work, open a browser on the PC running the 'home' and point it to the following location:

http://localhost:8080/Amigo/PAD?cmd=open

This will test the detection of exportable services and the NAT mapping functionality. As a result a single string should be returned, containing a list of stringified references to the exported services. Check that that urls contained in the string all point to the external IP address of the home network, since that is the point where the services will be mapped to. Below is an example of a string that could be returned by the PAD service:

```
[soap;http://82.72.92.208:9090/ksoap2/broker;null;Scope=urn:amigo,oid=broker
,ServiceType=ContextBroker],[soap;http://82.72.92.208:9090/ksoap2/IGDService
;null;Scope=urn:amigo,oid=IGDService,ServiceType=IGDService]
```

This example shows the context broker service and the IGD service being mapped to the external IP address of the home.

Once you are satisfied this is working, the whole scenario can be executed by running the RFID reader (simulator), fill in the Bluetooth address of the PAD and press the button to start the scenario.

3.11 SAInt – Seamless Audio Interface

3.11.1 System requirements

System requirements are closely linked with the amount of microphones and loudspeakers used. We recommend:

- 3.0Ghz CPU (multi core preferred)
- 1GB memory
- 50 Mb hard disc space

3.11.2 Download

Software packages and installation routines are available on our webpage (http://nt.uni-paderborn.de) for project partners.

3.11.3 Install

Installation routines for the Spark software and the OSCAR bundles are available on our webpage. Execute the shell-script "install.sh" and follow the instructions.

3.11.4 Configure

The system must be configured according to the placement of microphones and loudspeakers. This can be done via the configuration files. Examples for configuration files are contained in the software package.

4 Components Architecture

In this chapter, we present interactions between components of the WP7 demonstrator. We describe interfaces of components that are used by other elements of the prototype or that can be used by people outside the Amigo consortium. We also explicit how WP7 components leverage Amigo middleware to support communication and activity sharing between remote people.

4.1 Scheduler

Since WP7 applications share the demo environment a scheduler is needed for coordination. WP7 applications must negotiate with that service the permission to start and must notify it when execution starts and ends.

The Scheduler service implements the following interface:

```
Interface IApplicationManager
      void register( String appAmigoRef, String applicationName )
      void unregister( String appAmigoRef )
      boolean requestExecutionClearance( String appAmigoRef, Vector<String>
servicesAmigoRefs )
      void applicationStarted( String appAmigoRef, Vector<String>
servicesAmigoRefs )
      void applicationStopped( String appAmigoRef )
```

The `un/register` methods must be called by applications (application composition services) when they are deployed/halted. The `AmigoRef` parameter is the Amigo reference to the service that controls the composition and/or execution of the application.

The `requestExecutionClearance` method must be called by any application that wants to start. The `appAmigoRef` is a reference to the application service and `servicesAmigoRefs` is a vector containing Amigo references to all services required by this application. The scheduler returns `true` if there are no conflicts with other applications, or `false` if the calling application should not start.

If `requestExecutionClearance` returned `true`, then the application can start execution. Once the application is successfully started it must call the `applicationStarted` method. Here the vector contains references to services actually used by the application.

Once an application stops and frees all used services it must call the `applicationStopped` method.

Applications that register to the Scheduler must implement the `IStateControl` interface:

```
Interface IStateControl
      void start()
      void stop()
```

These two methods can be called by the Scheduler at any time to start or stop an application.

4.2 Resource Manager

Each shared interaction resource of the WP7 demonstrator must be coupled to an Amigo service managing the access to this resource, e.g. a display screen that can be used by several GUIs must be bound to a display resource manager. All services that want to use a shared resource must register to the resource manager and should negotiate with it usage of the resource. The resource manager implements a context source (CS) that can be queried for registered services and for the status of the resource (e.g. resource is in use by some service, is available for a service).

Resource managers have at least two scopes: "urn:amigo" and "urn:hostname", where hostname is the hostname of the PC where the resource manager is running, and to which the interaction resource is bound. A service that requires a particular resource such as a display can use the "hostname" scope of displayManager to look it up while limiting the discovery of the service to the local host.

Resource managers implement the "IResourceManager" interface:

```
Interface IResourceManager
      boolean register( String AmigoRef, String serviceType )
      void unregister( String AmigoRef )
      boolean claimResource( String AmigoRef )
      void serviceStarted( String AmigoRef )
      void serviceStopped( String AmigoRef )
      void informServiceAvailability( String amigoRef, boolean avail )
```

When a service calls the `register` method, the manager stores both service type and its reference in a list of registered services. The list of registered services is exposed by the manager's CS. The return value of this method indicates the availability of the resource at register time.

The `claimResource` method should be called by a service that wants to start executing and that requires the resource. It returns `true` if there are no conflicts with other services running on the resource, or `false` if there are conflicts and the resource cannot be claimed. The resource remains reserved for the claiming service for 3 seconds, i.e. during 3 seconds subsequent `claimResource` calls will be evaluated as if the first calling service would already use the resource.

A service that starts execution and uses the resource must call the `serviceStarted` method. This method can be called regardless of the return value of the `claimResource` method, i.e. a service can start execution whether or not it was allowed by the resource manager.

The `serviceStopped` method must be called by services that stop execution and free the resource.

The `informServiceAvailability` method enables the service to inform the ResourceManager about its availability. This information is exposed by the manager's CS.

Services that register to a resource manager must implement the `IResourceDependentService` interface:

```
Interface IResourceDependentService
      void informResourceAvailability( String resourceRef, boolean avail )
      boolean isStillClaimed ( resourceRef )
```

The `informResourceAvailability` method is called by resource managers each time the availability of the resource changes. If on an exclusive-use resource one of registered services starts execution (calls `serviceStarted`) the resource manager will call `informResourceAvailability( resourceRef, false )` on remaining services. Services must take in account this information when evaluating their own availability. If there is a change in their resulting availability, they have to notify their resource managers using the `informServiceAvailability` method. The information on availability of services (not that of the resource!) registered on the resource is exposed by the CS of the resource manager.

The `isStillClaimed` method might be periodically called by the manager to verify the status of running services. This method can be used to cope with services that failed to call `serviceStopped`.

The ResourceManagers service subscribes to WP4::LMS to get user position and to calculate relative distance to the user. Based on its location and on the location of the user, the resource manager calculates the distance from the user and maps it into one of five distance-dependent interaction zones:

- `outofRange` corresponds to a distance from which the interaction is impossible
- `ambientRange` is a zone within which the user can probably notice the interaction resource, but is to far to interact with it
- `notificationRange` is a zone where the user can interact in a subtle way, but is to far to perform full range of explicit interactions
- `interactionRange` is a distance at which the user can fully interact with the resource.
- `unknownRange` used when the resource is unable to determine the distance

Services (e.g. Palantir) can subscribe for changes in user relative distance. The resource manager's CS provides information such as: User1 is in ambientRange

Note that resource managers could also calculate the relative distance to mobile interaction resources (e.g. AG2). This could be used for located interaction, e.g. transferring a GUI from a display screen to a PDA display only when the PDA is at close proximity to the display.

All information collected by the resource manager is exposed as a context source (CS). The CS provides the following information modeled by the ontology shown in Figure 18:

- user-relative distance: "User1 is in interactionRange from ResourceA "
- list of hosted services: "ResourceA hosts Service1, Service2 …"
- types of hosted services: "Service1 hasType PalantirGUI"
- availability of hosted service: "Service1 isAvailable true"

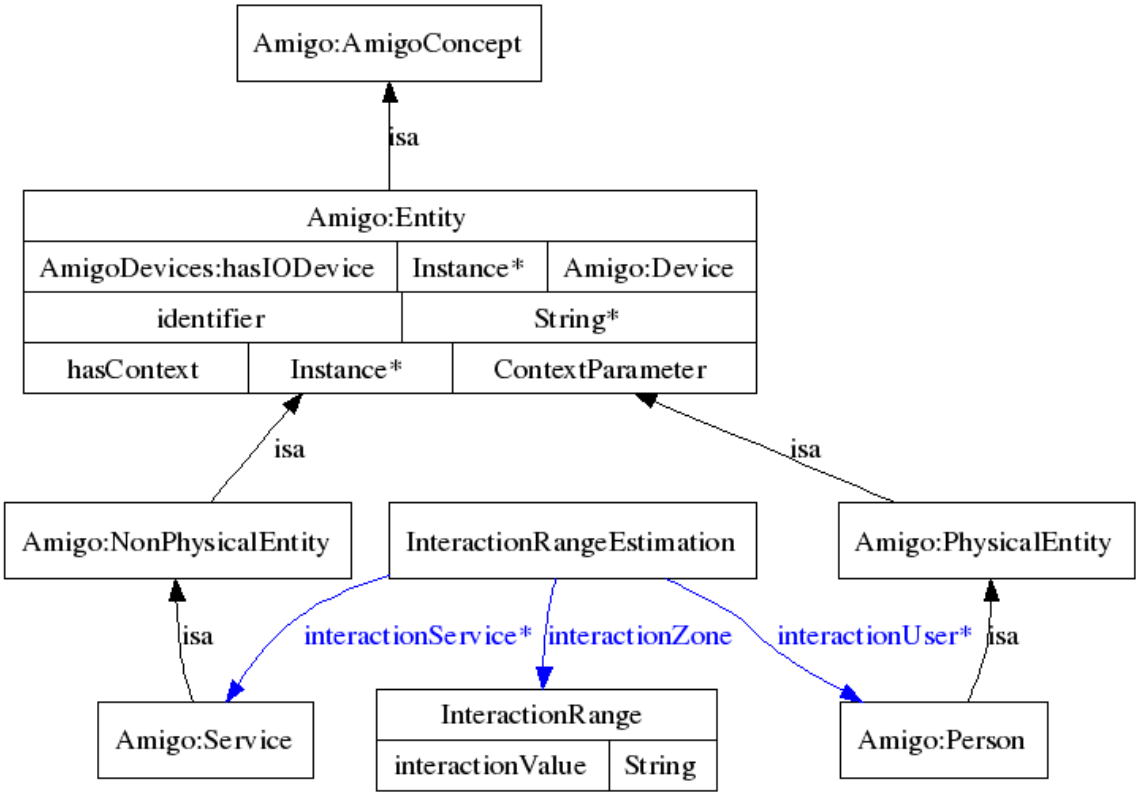

Figure 18 Concepts used by the ResourceManager's CS

4.3 Ambience Sharing

The Ambience Sharing application is composed of four types of interacting software modules that together support context-sensitive audiovisual communication:

- AmSharingApp service implementing communication initialization and location-aware audio-video follow-me function
- AVT modules supporting video streaming/decoding and context-aware video stream adaptation
- AmigoVisioServices acting as Amigo proxies for AVT modules
- SAInt audio system, implementing hands free audio communication and seamless audio stream redirection

In this section, we describe interfaces exported as web services by these components, as well as describe interactions that happen between these components on various events triggered by the users of the system.

Component Interface

The Ambience Sharing application composer service (AmSharingApp) exports only `PresenceStatusEventListener` interface (see section 4.4) needed by the Palantir to inform the AmSharingApp about user status, and the `IStateControl` interface (see section 4.1) needed by the Scheduler service to coordinate execution of WP7 applications.

The AmigoVisioService exports the `AmigoVisioService` interface that allows the AmSharingApp service recovering data about the AVT module for which the given AmigoVisioService acts as a proxy:

```
Interface AmigoVisioService
      String getSipId()
      String getAmigoReference()
```

Mechanisms of interaction

The components of the Ambience Sharing application interact with each other and with other Amigo services in two situations; when the communication is triggered, and at run time when the user changes location.

Ambience sharing is integrated with the Palantir system that offers loosely coupled session initialization interface to the application. If a user touches a picture of a buddy, the selected person's Palantir is alerted. Also the Ambience Sharing clients of the two parties are notified. This notification is a trigger to start the audio-visual communication. We illustrate these interactions in more detail in section 4.4.

Once a communication is started, the AmSharingApp redirects the audio and video streams to available devices that are closest to the user. The redirection is done based on location data managed by the WP4::CMS. The following diagram shows interactions between components of the WP7 prototype when user moves from device1 to device2 while communicating via Ambience Sharing.

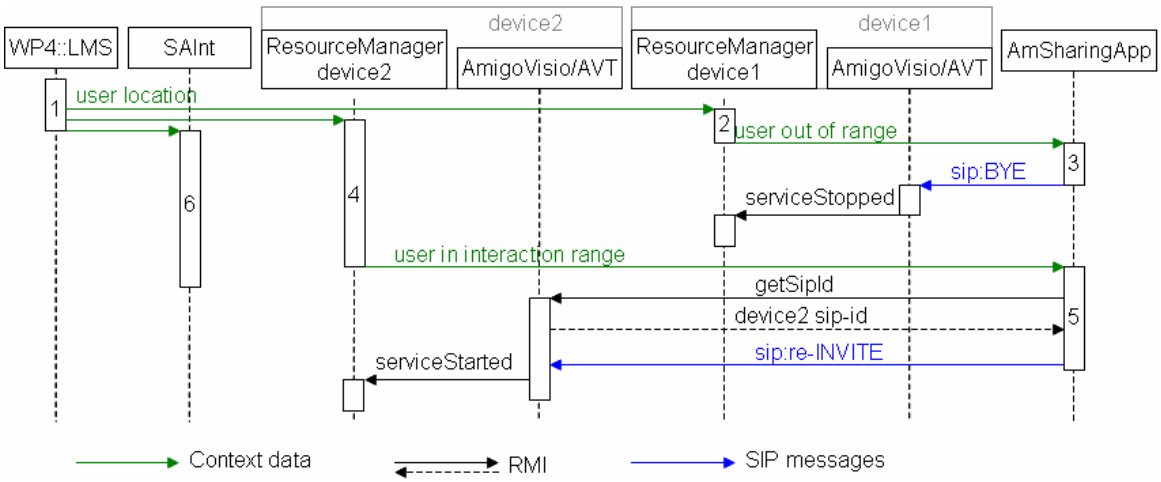

Figure 19 Changing AV communication devices in the Ambience Sharing application.

1. The WP4::LMS (Location Management Service) provides user location through the WP4::CMS (Context Management System) to subscribed context consumers.
2. The user moves away from device1, the ResourceManager running on this device interprets the context data from LMS and notifies the CMS that the user is out of interaction range for this device.
3. AmSharingApp service is subscribed for user relative location data to the CMS, thus it receives the notification issued by ResourceManager of device1 in step 2. The AmSharingApp service sends a "BYE" message to the AVT running on device1, the visual communication is now stopped on this device. The AmigoVisioService is also

notified and calls the `serviceStopped` method of the ResourceManager to free it for other services.

4. The user approaches device2. The LMS notifies user's position to the resource manager of device2, which notifies in turn the AmSharingApp that the user is within the interaction range of the AmigoVisioService running on the device and that this service is available.
5. The AmSharingApp sends a resumes the video streaming between the remote party and the AVT running on device2 by sending SIP "re-INVITE" messages. Finally, the AmigoVisioService notifies the resource manager of device2 that is has started, and that the resource is no longer available for other services.
6. The SAInt service manages the audio redirection independently. When Ambience Sharing application is started, the AmSharingApp calls `connect(String user1, String user2)` method of SAInt and the audio system subscribes to CMS for location data for the specified user and redirects the audio stream to the appropriate audio devices.

4.4 Palantir presence management system

The Palantir service is used by all applications in WP7 demonstrator. In this section, we present how the Palantir service interacts with the Amigo environment to evaluate the presence and awareness and how this information is transmitted to applications. Also we show how the Palantir is used to trigger other applications.

Component Interface

Palanir evaluates user status separately for each registered application. Each application must define its own presence model as an XML document listing services required for each level of user's communication ability (i.e. emitting, receiving and both way dialog). An application registers to the Palantir to receive user status changes by submitting its presence model. To perform the registration the application must use the `PresenceModelLoader` interface:

```
Interface PresenceModelLoader
      void loadPresenceModel(String url, String application)
      void unloadPresenceModel(String url)
```

Below is an example presence model for the Ambience Sharing application:

```
<application name="AmSharingApp" icon-uri="http://AmbienceSharing/icon.jpg">
 <presence status="3"> <!-- emission-reception -->
    <device type="AmShGUI" rating="1"/> <!-- minimal set-->
    <device type="AVTstreamer" rating="1"/>
    <device type="SAInt" rating="1"/>
 </presence> </application>
```

Alternatively, services can subscribe for user status changes by calling the `subscribeForEvents` method of the `PresenceStatusInteractionService` interface exported by the Palantir:

```
Interface PresenceStatusInteractionService
      void changePresenceStatus(String newStatus)
```

```
      void informOfParticularPresenceStatus(String[] userIds, String status)
      String getLocalUserDigest()
      void subscribeForEvents(String listener)
      void unsubscribeForEvents(String listener)
```

Services subscribing with this method are notified about all changes in user status, and not only about changes that are relative to a particular presence model as it is the case when subscribing with the `PresenceModelLoader` interface. The `subscribeForEvents` method is used by the PalantirGUI service that visualizes presence status of user's buddies.

The `PresenceStatusInteractionService` interface can be also used to modify the status of the local user with the `changePresenceStatus` method. The new status is send by the Palantir service through an XMPP server to the Palantir services of user's buddies.

Finally, this interface allows also informing particular users about a change of the local user's status with the `informOfParticularPresenceStatus` method. This method is used by the PalantirGUI service to inform remote users that one wants to start communicating when the user clicks a picture of a remote buddy.

To allow the Palantir notifying services about user status changes, services that submitted a presence model or subscribed using the `subscribeForEvents` method, must implement the `PresenceStatusEventListener` interface:

```
Interface PresenceStatusEventListener
      void presenceStatusProposed(String newStatus)
      void presenceStatusChanged(String newStatus)
      void particularPresenceStatusInformed(String[] userIds, String status)
```

The `presenceStatusProposed` method is called by the Palantir when new estimate of user status is made based on context information provided through WP4::CMS. The proposed status can be shown by visualization services such as the PalantirGUI for validation by the user. If the new estimated status is not canceled by the user within a fixed time, the Palantir considers the new value as valid and send the information to user's contact.

The Palntir calls the `presenceStatusChanged` method to inform services about a change of a user's presence status. Services that loaded a presence model receive status updates that are relative only to the presence model they registered to the Palantir, e.g.:

```
<presence-status user-id="User1">
   <application url="http://AmSharingApp/presencemodel.xml" presence="3"
     awareness="99"/>
</presence-status>
```

Services that registered to Palantir without loading a presence model (i.e. using the `subscribeForEvents` method), receive updates concerning user status changes with respect to all registered presence models, e.g.:

```
<presence-status user-id="User1">
   <application url="http://application1/presencemodel.xml" presence="2"
     awareness="35"/>
   <application url="http://app2/presencemodel.xml"   presence="2"
     awareness="18"/>
   <application url="http://AmSharingApp/presencemodel.xml" presence="3"
     awareness="99"/>
```

</presence-status>

Using the `particularPresenceStatusInformed` method, the Palantir can also inform services that submitted a presence model about particular status levels, such the maximal or minimal awareness value. If the awareness is maximal (i.e. the user wants to use an application), the presence status message contains also a list of services that were used to obtain this status level. Note that the listed services match the services requested by the application in the presence model example above:

```
<presence-status user-id="User1">
   <application url="http://AmSharingApp/presencemodel.xml" presence="3"
     awareness="100">
      <device type="AmShGUI" AmigoRef="instanceURL"/>
      <device type="AVTstreamer" AmigoRef="amigoRef2"/>
      <device type="SAInt" AmigoRef="amigoRef3"/>
   </application>
</presence-status>
```

Mechanisms of interaction

The Palantir presence management service is central to the WP7 demonstrator. It provides applications with user status data. In this section we show how the Palantir estimates user status based on context and how this data is used by other WP7 applications.

In order to estimate user presence, the Palantir service subscribes for the following context data:

1. User location, e.g. user1 enters roomA
2. User relative location: user1 is in interactionRange of resourceA
3. Service location: serviceA is in roomA at X, Y (this information can be queried from the ResourceManager's CS)
4. Service availability (also provided by the ResourceManager's CS)

User ability to communicate is evaluated each time the user moves to a different area, e.g. to a room. On such event, the Palantir performs a location-aware service discovery to obtain a list of services available in the area. This list is then compared with the list of services required by each application and ability level is set accordingly. Figure 20 shows interactions between WP7 services that lead to re-evaluation of user status for the Ambience Sharing application when the user enters the environment.

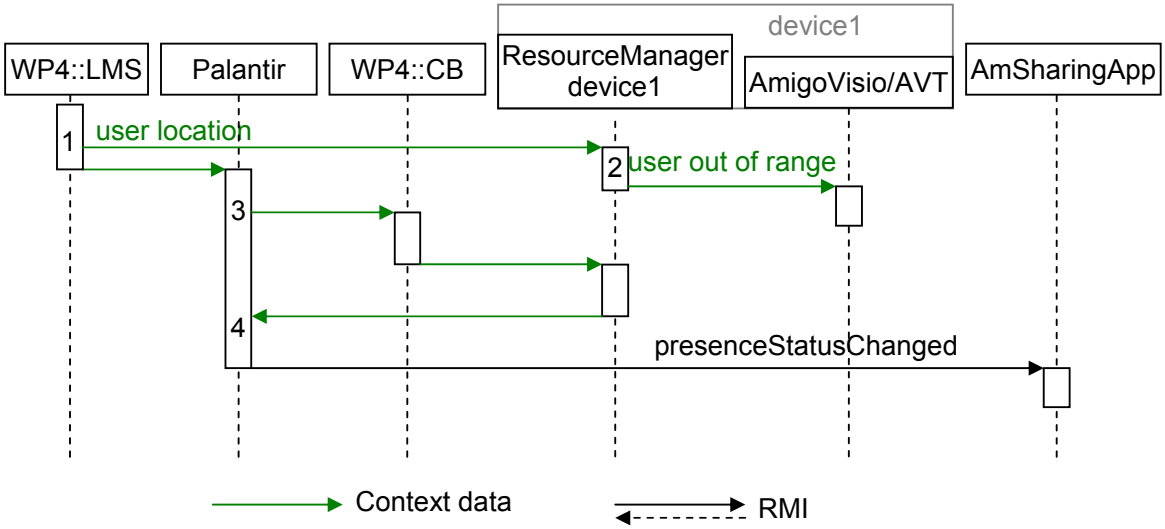

Figure 20 Estimation of user presence status for the Ambience Sharing application when user enters a room

1. The Location Management Service notifies subscribed context consumers (i.e. the Palantir and the resource manager on device1) that the user entered the environment.
2. The ResourceManager calculates user's relative distance to device1 and notifies registered services that user is too far for interaction.
3. The Palantir subscribes through the Context Broker (WP4::CB) to context sources providing information on services in the given location (e.g. in a room). The CB redirects the Palantir query to the ResourceManager's context source, which replies with a list of services that are registered on the resource.
4. The Palantir compares the received list of services with the list of services required by the AmbienceSharing application specified in its presence model. The AmigoVisioService is present on both lists, thus the Palantir considers that the user is now able to communicate through the AmbienceSharing application. The Palantir notifies the AmSharingApp service about the change in user status. This information is also send through an XMPP server to Palantirs of remote buddies. Note that conventional applications also can obtain user status produced by the Palantir from the XMPP server.

While the availability can be estimated automatically based on user and services location, the awareness is meant to be explicitly input by the user, being in most cases impossible to infer automatically from context data. Nevertheless, the Palantir will modulate the awareness level when the user approaches or moves away from a device hosting services required by an application. For instance, if a user approaches device1 in the example above, the resource manager of the device will notify the Palantir that the user entered interaction zone. This information will increase user awareness level. The new value must be still validated by the user so the Palantir will call the `presenceStatusProposed` method on registered services, before calling the `presenceStatusChanged` method (see Figure 21).

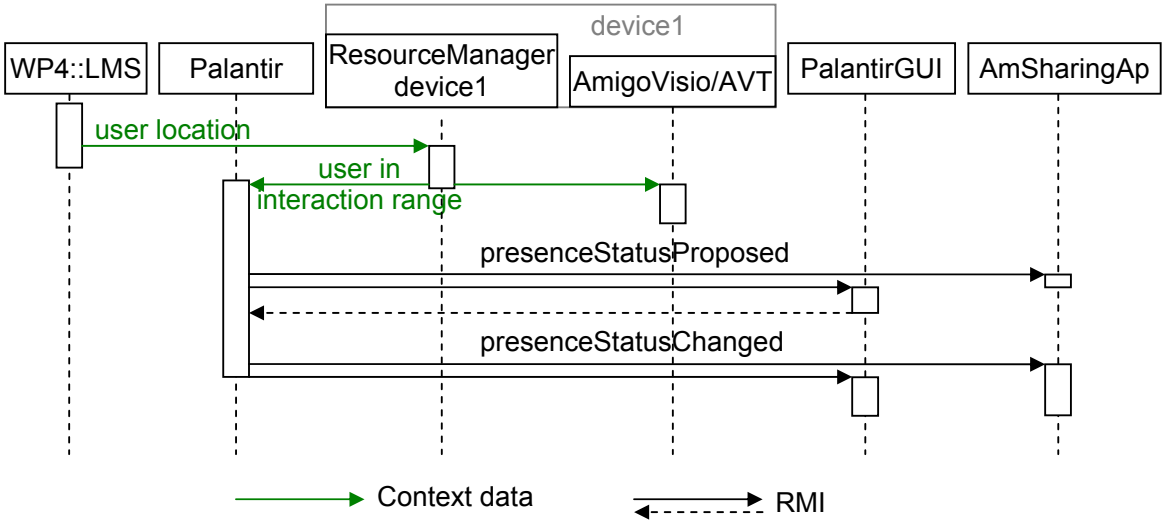

Figure 21 User status evaluation on relative location event

The Palantir together with a user status visualization service, such as the PalantirGUI or the Awareness Globe 2, can be used to trigger other applications. The following diagram illustrates what happens when a user starts from a PalantirGUI a communication with the Ambience Sharing application.

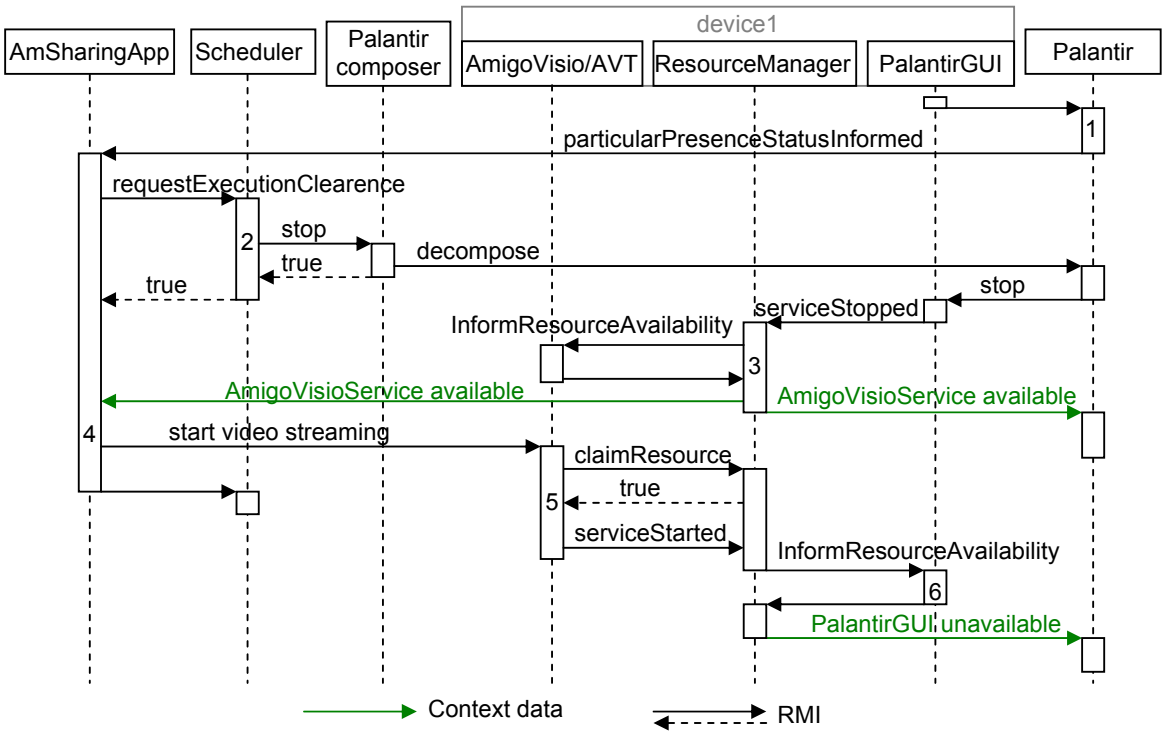

Figure 22 Ambience Sharing application started by the user from the PalantirGUI service

1. The user clicked on a picture of a remote buddy. The Palantir service notifies the AmSharingApp service and the Palantir of the remote buddy that user awareness of the local user is maximal, i.e. that the user wants to start the Ambience Sharing application with the remote buddy.

2. The AmSharingApp requests execution authorization from the Scheduler service. The Scheduler, first stops the Palantir application, and then grants the authorization.
3. The Palantir service disconnects from the PalantirGUI service, which informs the ResourceManager that it is now stopped. The manager informs registered services (i.e. the AmigoVisioService) that the resource is available. In response, the AmigoVisioService evaluates its availability and informs the ResourceManager that it can be used. This information is published by the resource manager as context data.
4. Since the AmSharingApp service knows know that there is an AmigoVisioService available in the environment, it can initialize the communication.
5. The AmigoVisioService requests the authorization to use the resource. Since the resource is available the ResourceManager allows the AmigoVisioService to start.
6. As soon as the AmigoVisioService starts, it notifies the ResourceManager, which in turn notifies registered services that the resource is now occupied. In response, the PalantirGUI informs the manager that since the resource is busy, the PalantirGUI service is unavailable. This information is published by the resource manager as context data.

Note that the Scheduler can and the resource managers can implement application or service-dependent policies. For instance, the Scheduler may refuse execution authorization if the running application more important than the requesting one. Similarly, in case of concurrent access to interaction resources, the manager can arbitrate priorities between services.

Note that the interaction mechanism presented above apply also for other WP7 applications.

4.5 Activity Sharing

While Activity Sharing application suite is developed using conventional technologies and can be deployed outside of an Amigo enabled system, it is integrated with other WP7 applications. Activity Sharing is integrated in the WP7 demo scenario, but also has links to, and relations with Amigo's underlying middleware framework, as well as integrated SW parts of partners in the project.

- **3rd party integration**: Philips' EasyLogic UI is extended with a so-called "redirect" functionality, to integrate and launch 3rd party applications fast and easy. Philips' Poker game, and FT Palantir are integrated in such way that they are found as Amigo service, using CMS. Accordingly, different links to Amigo middleware applications are created, and shown within EasyLogic interface style.
    - **Poker Game**: Integration of Poker game (PHI), which connects to several underlying Amigo services (e.g. UMPS) – the Poker game can be launched from the EasyLogic menu or from the AG2.
    - **Palantir**: Launch Palantir application.
    - **AG2**: Trigger Activity Sharing from within the AG2 interface.
- **Presence status**: Changes in presence status is shared amongst all Amigo applications, by making use of the same shared XMPP server. It does not matter that Activity Sharing uses a shared SQL server for both SIP and XMPP, instead of standard (internal) SIP/XMPP user/roster management, since the presence status update works with both situations. Presence status changes that are triggered by e.g. Palantir will affect availability indications within the EasyLogic interface, showing users either online/offline/busy.
- **eConf video streaming**: Integrated FT's AV encoding and streaming software.

4.6 Awareness Globe

The Awareness Globe 2 is developed in conjunction with the 'Activity sharing' applications described in the previous section. It is integrated with the underlying middleware framework.

- **CMS** is used to determine presence and activity of the user's contacts, and shown as a graphic attached to the contacts image in the Awareness Globe 2 main UI screen.
- Next to this, WP7 applications are able to show as shared activities in the Activities bar on top of the AG2 main UI screen via CMS
- **UMPS** is used to save preferences and statistics from contacts when launching a poker game from the AG2

4.7 Social Radio

Social Radio is a novel approach for mediating awareness in small intimate groups. Instead of traditional communication media, music is used to inform users about the presence and mood of multiple remote peers. The system consists of several smart artifacts and an underlying multi-user communication infrastructure. The artifacts are controlled via a tangible user interface. The Social Radio is integrated with the following Amigo components:

- WP3::OSGi development and deployment framework
- WP7::Palantir service is used for propagating presences information about users. The detection of user's ability to communicate turns the Social Radio application on. The decision to play the music, a remote user is listening to, is made according to the delivered awareness level from the Palantir service. The Social Radio Demonstrator plays music, if the awareness level delivered by the Palantir is greater than 60.

4.8 Feeling@

The Feeling@ application builds up from exposing simple context information to publishing complex activity events, using major features provided by Work Packages 3 and 4 and leveraging WP7 architectural concepts.

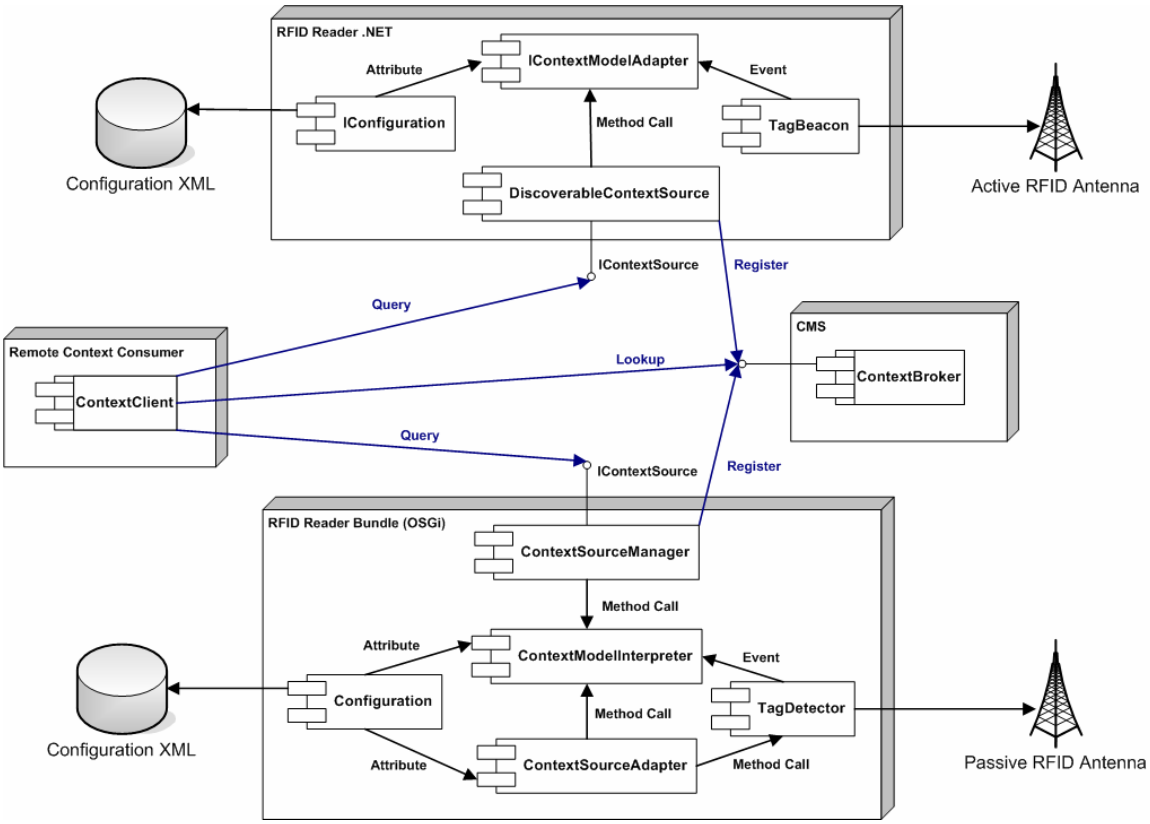

Figure 23 The RFID Reader Architecture

Figure 23 shows the two RFID Reader implementations for active and passive RFID technologies. Both share a very similar architecture. Tag recognition data is polled frequently from the RFID devices, interpreted and represented as a model. Clients or subscriptions query the model and receive updated context information. Configuration in both cases is based on XML files.

The picture also represents WP4 CMS interactions. Both components at startup register themselves to the Context Broker. Context Clients lookup for specific context providers and receive references to the Context Sources they can query.

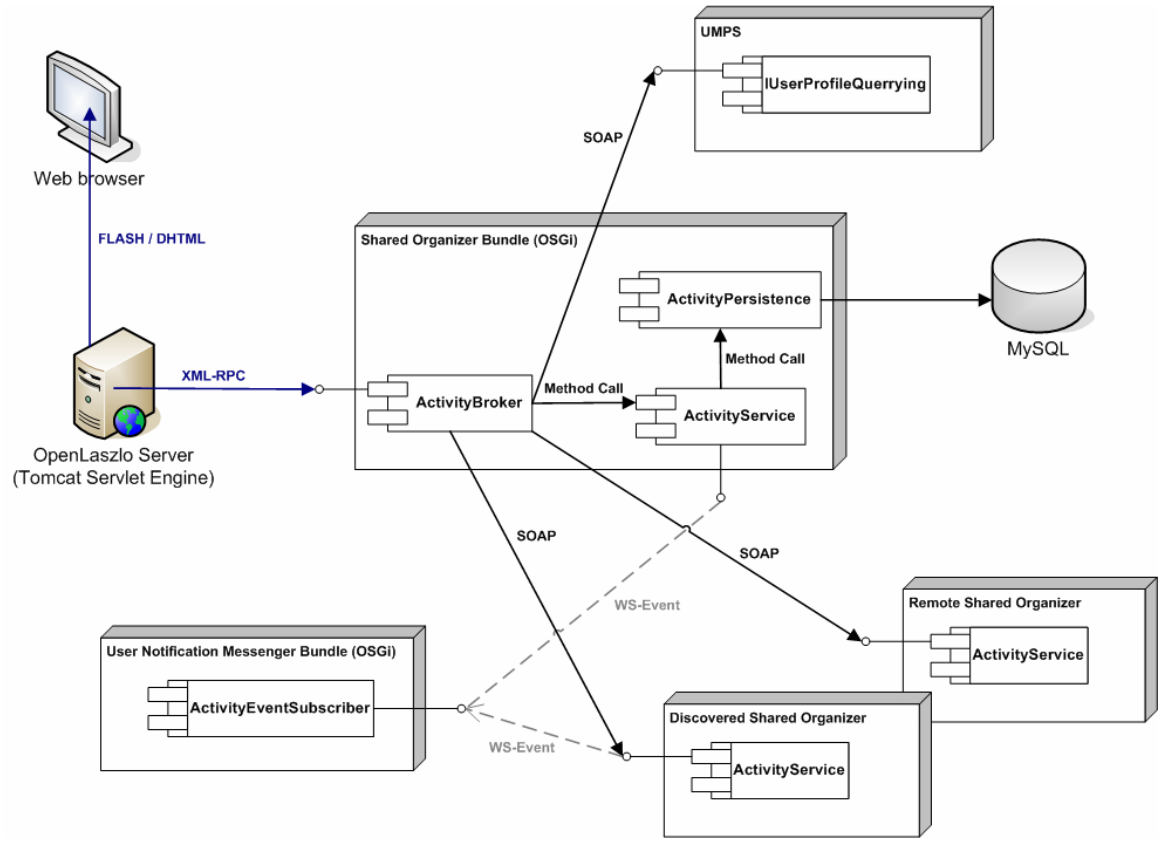

Figure 24 The Shared Organizer Architecture

The diagram in Figure 24 depicts the Shared Organizer back-end distributed architecture. Based on SOA principles, the module consumes multiple remote web services of the same and different type, leveraging WS-Discovery features. At the same time each Shared Organizer instance publishes events, following WS-Eventing standards, for any web component to consume. In Feeling@ application the Shared Organizer generated events are consumed by the User Notification Messenger. Internally, the component handles data persistence locally, just as any other typical application, though information is shared among other instances and aggregated from sparse sources at the same time. The WP4 UMPS module is used to retrieve user profile information.

The Shared Organizer front-end module is contained and handled by the OpenLaszlo engine on a possibly different host. Communications with the back-end occur by the means of XML-RPC technology. Clients are standard web browsers, offering a rich featured user interface with both DHTML and Flash outputs.

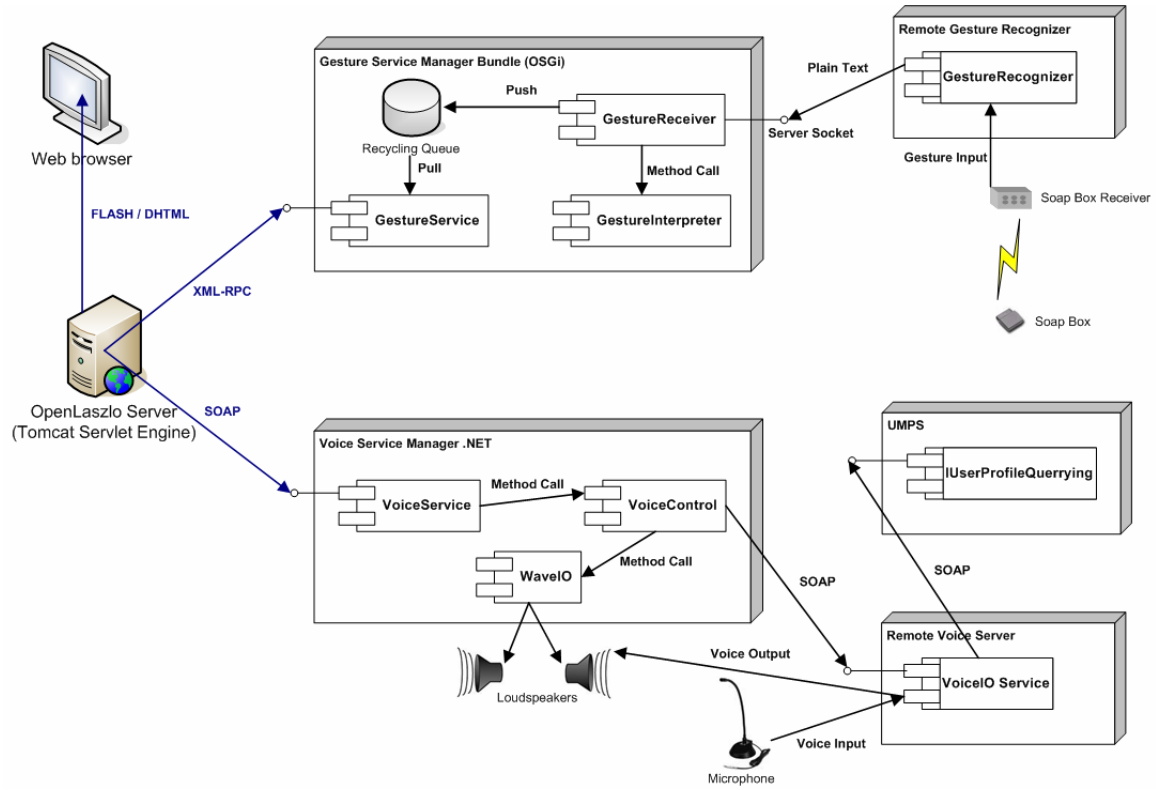

Figure 25 The Sketch Presentation Architecture

Figure 25 represents the Sketch Presentation subcomponents architecture. WP4 Voice and Gesture services have been proxied by two components (the managers) that simplify and uniform front-end access.

The Gesture Service Manager receives real time gesture commands which are stored in a bounded and recycling queue. As soon as the Sketch Presentation GUI polls for new commands, these are pulled out of the queue and serviced back. Gesture commands are received from a simple socket server in plain text or MMIL, with or without SOAP headers.

The Voice Service Manager delegates voice recognition totally to the Voice IO Service which is in charge of receiving microphone input and generating voice notifications to the users (command input request, no audio heard, command not recognized). Recognized commands are sent back to the caller application. Optionally, it can also request speech generation on demand and play it on local loudspeakers.

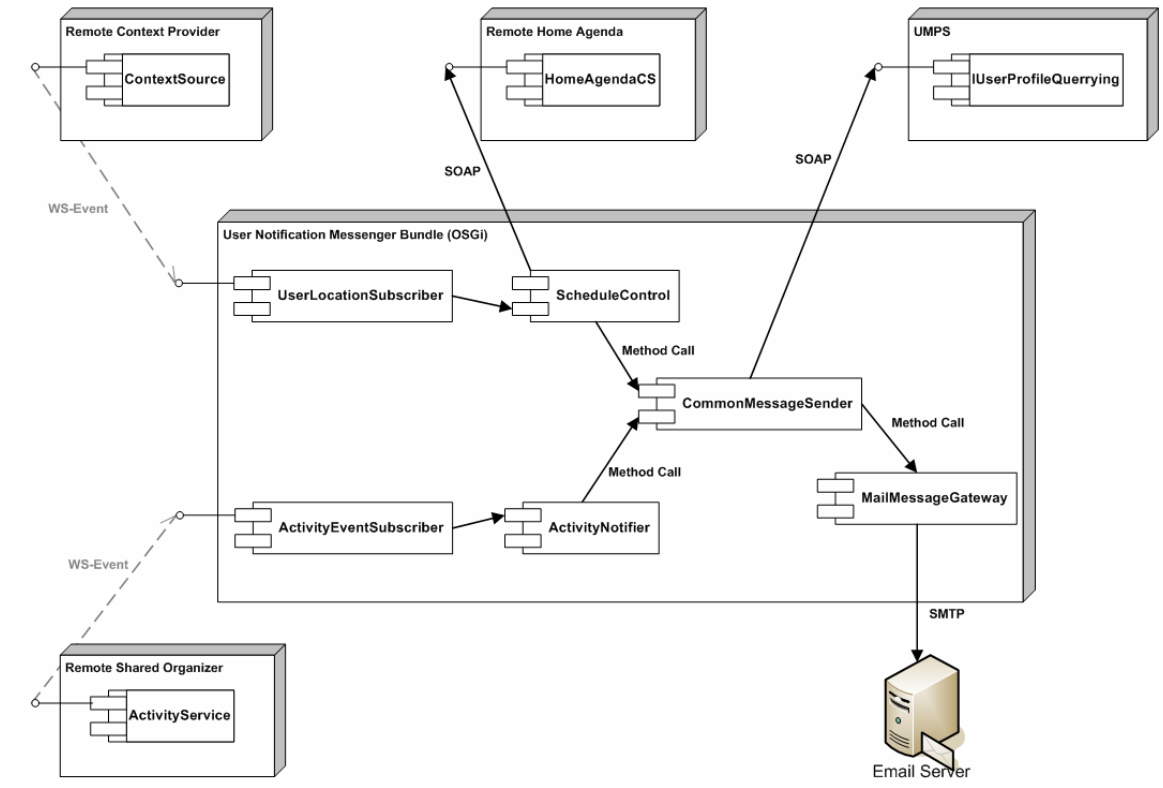

Figure 26 The User Notification Messenger Architecture

Figure 26 shows the User Notification Messenger interactions and internal architecture. This module subscribes and reacts only to web service events (WS-Eventing) thus being characterized as a passive actor of the Feeling@ application. Each external source publishes events which are captured by subscriber components and relayed to each relative processor. Each control component then decides, based on eventually other external sources, to notify a specific user. A centralized component handles the messages that should be sent by delegating to specific technology gateways, as the Mail Message Gateway, the actual transmission. The User Notification Messenger interacts with context source providers for location data, Feeling@ Shared Organizer instances for activity events, WP6 Home Agenda components for schedule information and WP4 UMPS for user profile settings.

4.9 Personal Amigo Device

The Personal Amigo Device (PAD) offers users a simple and transparent way to access home services in a visited environment. It facilitates seamless creation of trust between a user's own home and a visited intelligent environment (i.e. home).

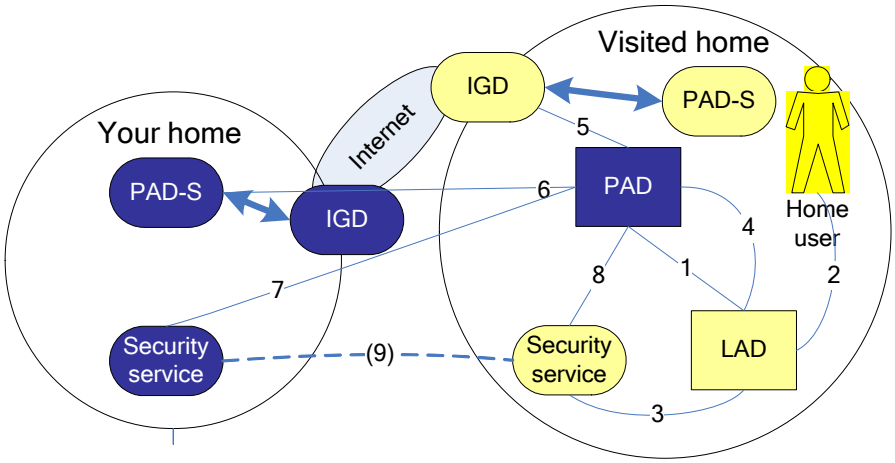

Figure 27: Steps in trust establishment between two homes.

To dynamically establish the trust between the two homes, the PAD acts as a mediator between the (WP3) security services of the two different homes. **Error! Reference source not found.** Figure 27 shows the different steps and interactions between the different components that play a role in the PAD scenario, these are: Personal Amigo Device (PAD), PAD-Service (PAD-S), Internet Gateway Device (IDG, a home router), Local Authentication Device (LAD, an RFID reader like the Awareness Globe v1). These components work together to enable the sharing of services between homes. In order to accomplish that several steps need to be taken, shown below. The numbers in this list correspond to the numbers shown in the component interaction diagram of Figure 27 and in the sequence diagram of Figure 28.

1. Swipe an RFID tag, containing the PAD Bluetooth address, over the LAD.
2. (optional) Ask the home owner permission.
3. Get a token from the security service, to be used by the PAD.
4. Communicate token and WLAN information to the PAD, using the Bluetooth (address was provided in step 1).
5. PAD asks the IGD for the external address and opens up some needed ports.
6. PAD asks PAD-S in its own home for a list of services to be exported, the PAD-S also opens up additional ports for these services.
7. Get a (join) token from the PAD's 'own' security service.
8. The PAD asks the security services to join/associate.
9. The security services exchange information, from this point onward they will work together.

After these steps are taken, the home services are advertised in the visited domain using the WP3 Amigo deployment middleware.

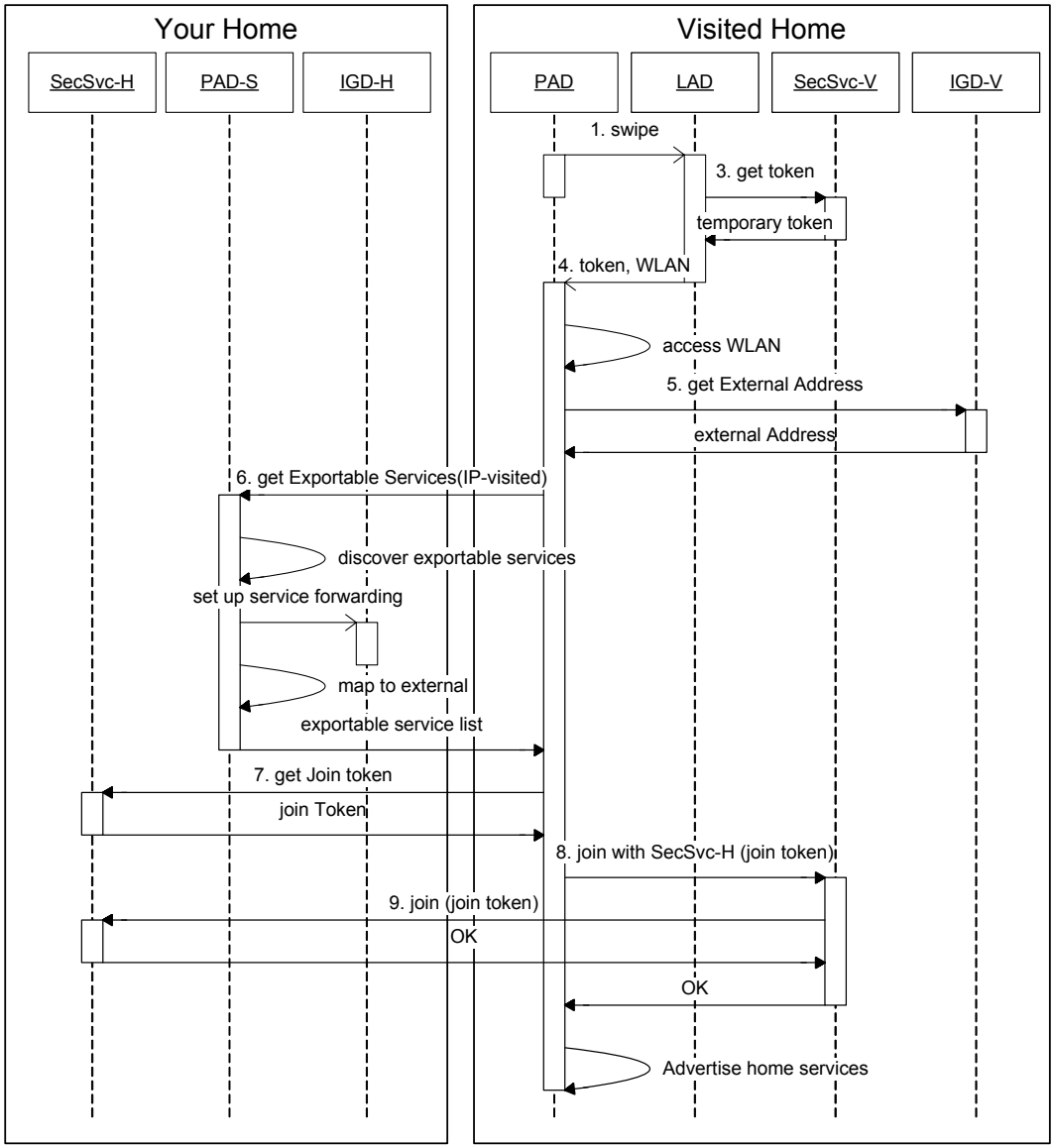

Figure 28: PAD trust establishment sequence diagram.

Since in a home situation typically routers are used with NAT functionality, exporting of services also entails mapping the internal service reference to the external IP address of the router (IGD device). Performing this mapping is part of step 6 of the scenario above.

After the above scenario is finished, the homes are ready for service usage across the home domains. Figure 29 shows how the security services in the homes will cooperate to issue tickets for this service usage across domain boundaries. The numbered steps in the figure correspond to the bracketed numbers in the textual description below.

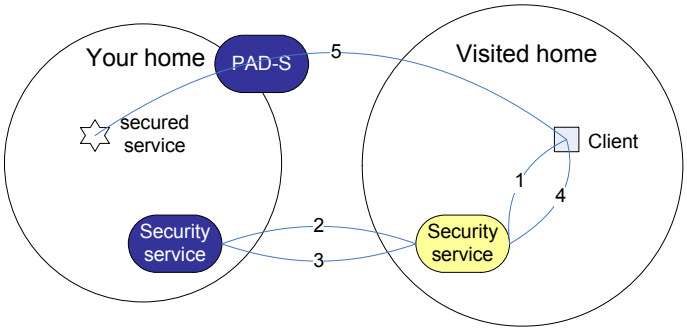

Figure 29: Federated client authorization.

To a service from the home of the PAD, a client at the visited location requests a ticket from the security service of the visited domain (1). After authenticating the requestor, the ticket request is forwarded to the security service in the PAD's home (2), which either denies the access or grants a ticket. This ticket is sent back to the requesting security service (3). From there it is passed to the requestor (4), which can now use the ticket to access the service (5). The ticket request sent in interaction 2 is clearly visible as coming from a remote home and thus access limitations can easily be configured in the security service that creates the tickets for its local services. The control therefore remains with the security service of the PAD's home.

Although the Personal Amigo Device (PAD) is not an integrated application in itself, it does have the ability to connect two homes; not only for general Amigo services but also for the Content Management related components ('Media Manager Core') from WP6. In order to be able to export content between homes some adaptations were needed, since the base WP6 components and services use UPnP for content related functions, in accordance with DLNA. The Media Manager Core provides the User Interface to the media related functions, as shown in Figure 30. For a full discussion of the media management related functionality please see D6.3 and D6.4.

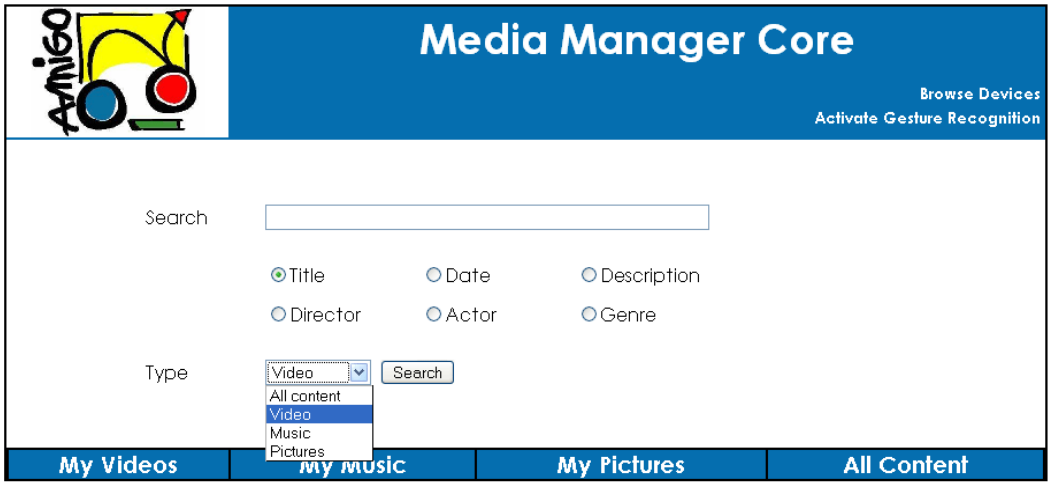

Figure 30: Media Manager Core (MMC) User Interface.

The components that play a role are shown in Figure 31, from which can be seen that the Amigo Content Discovery component plays a pivotal role in Amigo content management. It

discovers every Digital Media Server (DMS) present in a home and compiles an aggregated listing of all their content as well as related metadata.

To be able to share content rather than services between homes, this Content Discovery component was augmented with an Amigo Service interface, which provides access to the description of all the content that is currently known by the Content Discovery component. In Figure 31 this additional interface is shown in the left home as the white ACD' interface.

Another addition that was made to the Content Discovery component was the ability to process information from such an interface. By letting Content Discovery listen to the appearance of these types of interfaces, using passive discovery provided by the Amigo middleware, new content from other homes will be imported as soon as the PAD exports an ACD' interface in a visited home.

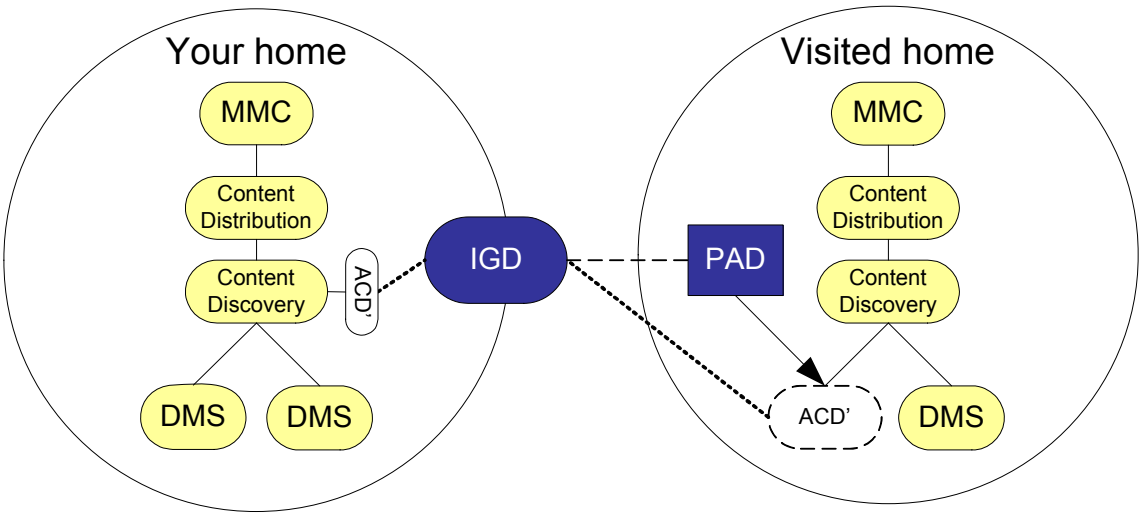

Figure 31: Exporting content between homes.

Figure 31 shows a situation with such an interface exported by the PAD in a visited home, denoted by the dashed ACD' in the right home.

Since media references in UPnP typically use IP addresses, which in a home situation are usually private range IP addresses, these have to be mapped to the external address of the Internet Gateway Device (IGD) in the same way as is done for exported services.

Note that the changes made to the media management components are limited to the Amigo Content Discovery Component (adding an interface, IGD mapping functionality, and the ability to import from the new interface), and that these changes are completely transparent to the rest of the media management components and functionality.

4.10 SAInt – Seamless Audio Interface

The SAInt (Seamless Audio Interface) is software for real-time hands-free audio communication. SAInt is build of two main components and an additional graphical representation for debugging, testing and controlling purposes. Figure 32 shows interactions between SAInt components and other elements of the WP7 demonstrator.

The audio processing part is implemented in the C++ based software SPARK (Speech Processing and Recognition Toolkit). This audio software deploys an inter-process

communication (IPC) interface to the Oscar Bundle SAInt. SAInt is responsible for retrieving location information and publishing web-services for controlling communication sessions as well as notifying clients about context information. Therefore SAInt has the standard IContextSource and IContextSubscription interfaces and can be discovered through the Amigo Context Broker Service.

The SAInt GUI is an example context client for the SAInt, which uses the Context Broker for finding SAInts and registering to one. Once registered, the retrieved context information is displayed in a window. Also the SAInt GUI can be used to invoke the SAInt web-services to start and control communications. This GUI can be used as a reference implementation for new applications.

External connections between different domains are realized through the ambient communication server (ACS). All tasks related to session handling are performed by the SPARK-engine. This reduces the complexity of the SAInt web-services for session control to a minimum. Spark deploys standard values for each new connection concerning the privacy level and the timeout limit, which are predefined in the configuration files. If the standard privacy level is set to private and more than one person is in the room during start-up of a connection, the connection privacy level is automatically reset to public.

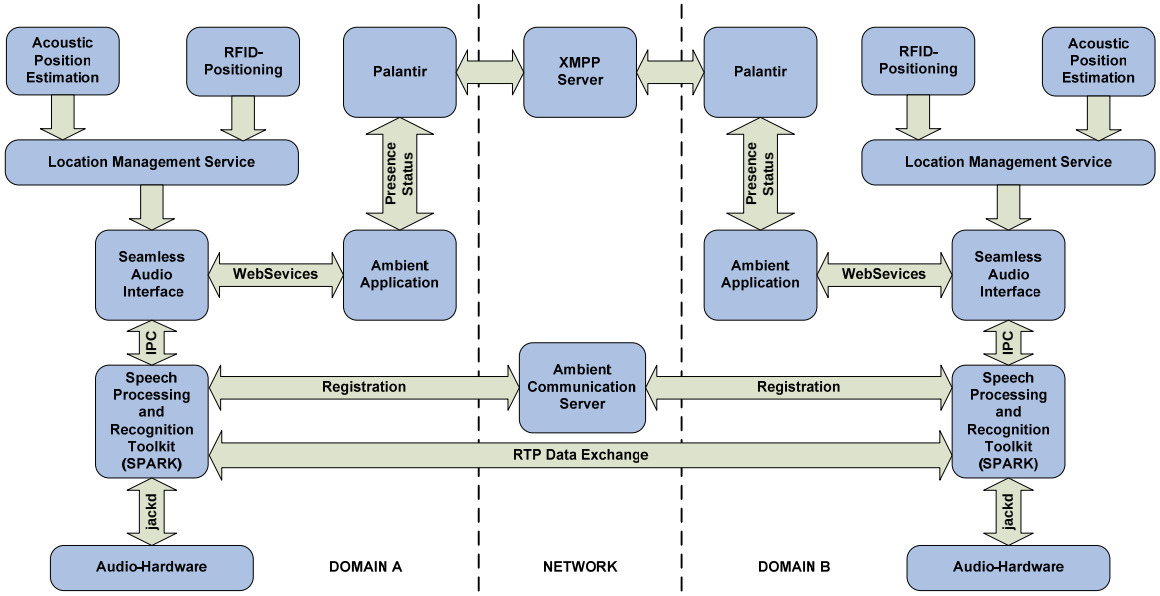

Figure 32 SAInt Architecture

4.10.1 Ambient Communication Server (ACS)

The Amigo scenarios describe ambient communication setups between distant houses. This requires an audio real-time streaming solution, preferably offering a high audio quality. Additionally, a solution has to cope with firewalls and network address translation (NAT) related problems, originating from Routers and DSL-modems.

The implemented server application, the “Ambient Communication Server (ACS)”, meets the requirements and is able to cope with the mentioned network related problems. The server is based on the User Datagram Protocol (UDP) and can be deployed locally for intranet communication or exposed on the internet. For a communication between 2 SAInts, a common ACS is required. All SAInts can see the registered Users at the ACS and can invite others to

join a communication. Moreover, the ACS can be used for handovers between different SAInts.

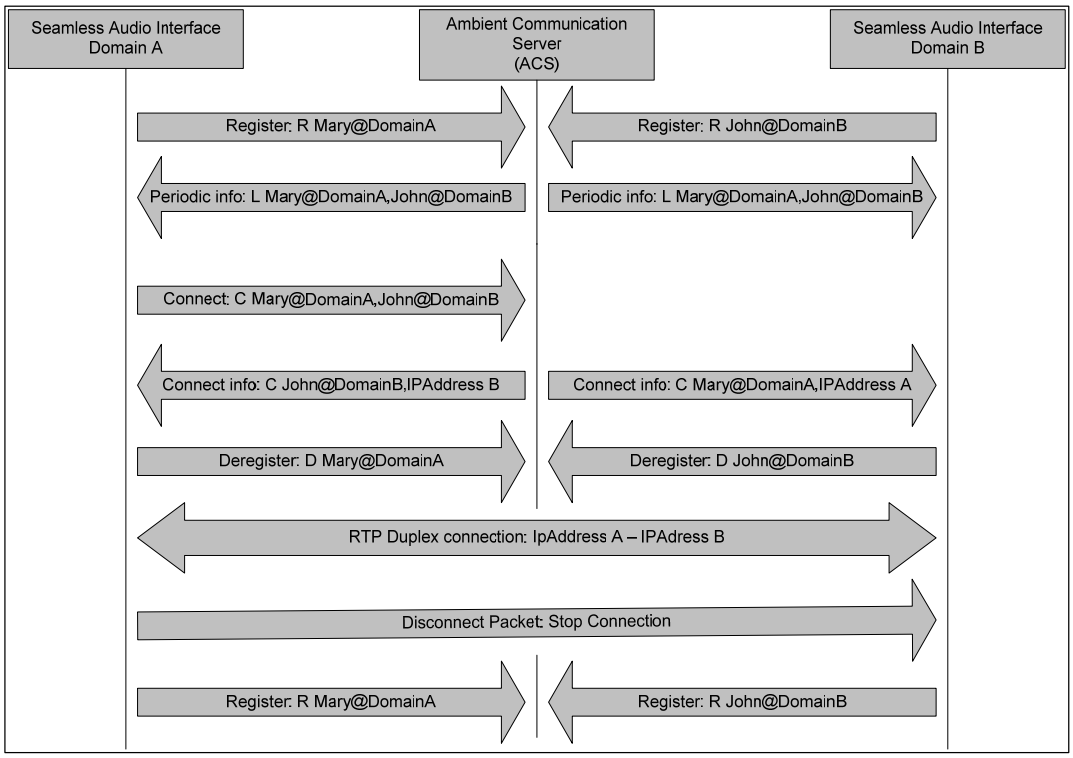

Figure 33: Session initialization by the ACS

Figure 33 shows the standard procedure for setting up sessions. First, the clients (here John and Marry) have to register at the server with their names. From the registration package the IP-Address of each user is refined. This address will be used for the RTP (real-time transport protocol) connection and allows crossing NAT's and firewalls. Periodically the server sends a list of all registered users to the clients. In the given example session initialization, Marry decides to invite John to an ambient communication. Therefore, the ACS is informed to start a connection between the two clients. The ACS sends the name of the communication participant and its IP-address to the other client and an RTP session can start. A session can end by sending an explicit "stop" package or by an implicit timeout. Afterwards the clients re-register at the ACS to be available for further connections.

4.10.2 SAInt interface

The SAInt JAVA interface implements web-services for establishing, terminating and controlling communications. As such it handles the inter-process communication with the SPARK-engine and the context information exchange with appropriate Oscar bundles. This section focuses on the SAInt as both, a context client and a context source.

As a context client, SAInt gathers and combines the context information from the LMS and the local Spark Engine. The resulting context information is posed onto the network by the SAInt JAVA interface.

The following sections describe:

- How to discover SAInt as a context source;
- The ontology model SAInt uses to store context information;

- How to query this ontology model;
- The usage of SAInt's web-methods to start, control and terminate a communication session.

Context Needs Formulation – How to Discover SAInt

The RDF capability of the SAInt as a context source is as follows:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns="http://amigo.gforge.inria.fr/owl/ContextTransport.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://amigo.gforge.inria.fr/owl/ContextTransport.owl#">
  <ContextSourceRegistration>
    <timeliness>current</timeliness>
    <contextType>
      SeamlessAudioInterface
    </contextType>
  </ContextSourceRegistration>
</rdf:RDF>
```

SAInt Context Information

The SAInt context source takes the location information of the LMS and adds Saint specific context to it. SAInt specific context information covers:

- rooms equipped with audio hardware
- registered and available users + rooms
- current audio connections + connection properties

A sample representation of the metadata according to the amigo ontology is as follows:

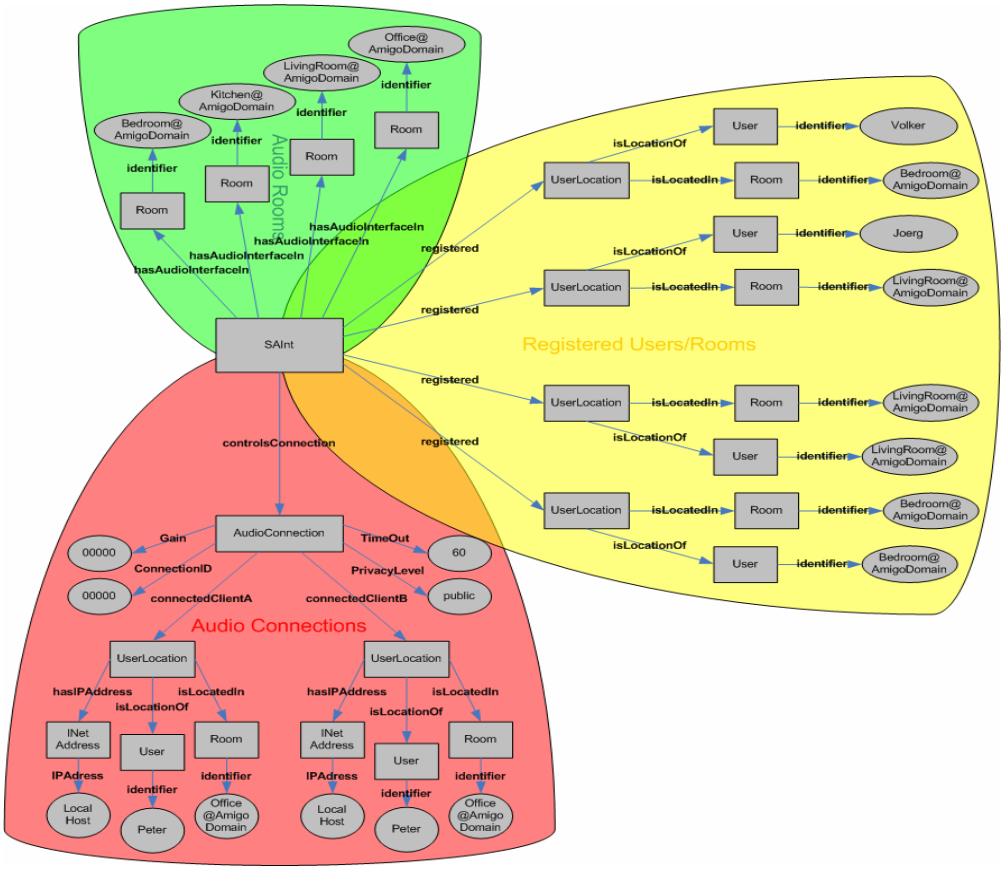

Figure 34: A sample ontology model

The 3 different colours indicate the 3 types of context information the SAInt is providing:

- The green part contains all rooms suitable for audio communication – these rooms have to be equipped with appropriate audio hardware.
- The yellow part contains the "dynamic" LMS/SPARK information about users and the rooms they are in plus the "static" information about the rooms equipped with audio hardware – excluding those already involved in audio communication. For unconstrained room-to-room or room-to-person/person-to-room communication, rooms are treaded as "special" users.
- The red part gives information about all current connections the SAInt is hosting.
  A connection is established between two user locations. The user's location (the room he/she is in) is subject to change. If a change occurs during an ongoing connection, the audio connection will automatically be redirected to the new user location, if the new room is equipped with suitable audio hardware.

  Each connection is characterized by:

  - ConnectionID: the connection ID (00000 – 99999) is unique to each connection and is used to control and terminate the connection;
  - TimeOut: the time-out interval in seconds (0s, 30s, …, 300s, ∞) after which the connection terminates itself under special circumstances;
  - Gain: the current volume in dB;
  - PrivacyLevel: the level of privacy the current connection provides (0 – 10);

- connectedClientA and connectedClientB: the communication partners described by a user location with attributes room, user and INetAddress;

In the example (see Figure 34), the communication is an internal one – the two clients are in the same house (and are registered at the same SAInt). For internal connections, the INetAddress is set to "LocalHost" and the room names are visible. For external connections, the field INetAddress is set to the IP-address the "external" SAInt is registered with at the ACS. In those cases, the room name of the external communication partner is set to "ACS", since the location shouldn't be visible to other users outside ones home environment.

How to query/subscribe to context information

According to the above model, three separate queries are used to get the current information.

The queries ask for:

- **Rooms** equipped with suitable audio hardware; the appropriate SPARQL query is:

```
PREFIX amigo: http://amigo.gforge.inria.fr/owl/AmigoICCS.owl#
PREFIX context: http://amigo.gforge.inria.fr/owl/ContextTransport.owl#
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
SELECT ?audioRoom WHERE {
?sa rdf:type amigo:SAInt .
?sa context:hasAudioInterfaceIn ?ar .
        ?ar context:identifier ?audioRoom .
}
```

- **Users + Rooms** registered at SAInt (excluding users in rooms without audio hardware and those currently involved in an audio connection); the appropriate SPARQL query is:

```
PREFIX amigo: http://amigo.gforge.inria.fr/owl/AmigoICCS.owl#
PREFIX context: http://amigo.gforge.inria.fr/owl/ContextTransport.owl#
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
SELECT ?registeredRoom ?registeredUser WHERE {
        ?sa rdf:type amigo:SAInt .
        ?sa context:registered ?ul .
        ?ul context:isLocationOf ?u .
        ?u context:identifier ?registeredUser.
        ?ul context:isLocatedIn ?r .
        ?r context:identifier ?registeredRoom.
}
```

- **Connections** + **Connection Properties**; the appropriate SPARQL query is:

```
PREFIX amigo: http://amigo.gforge.inria.fr/owl/AmigoICCS.owl#
PREFIX context: http://amigo.gforge.inria.fr/owl/ContextTransport.owl#
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
SELECT ?connectedRoomA ?connectedUserA ?connectedIPA ?connectedRoomB
?connectedUserB?connectedIPB ?connectionID ?privacyLevel ?timeOut ?gain
WHERE {
        ?sa rdf:type amigo:SAInt .
        ?sa context:controlsConnection ?ac .
        ?ac context:ConnectionID ?connectionID .
        ?ac context:PrivacyLevel ?privacyLevel .
```

```
            ?ac context:TimeOut ?timeOut .
            ?ac context:Gain ?gain .
            ?ac context:connectedClientA ?ulA .
            ?ulA context:isLocatedIn ?rA .
            ?rA context:identifier ?connectedRoomA .
            ?ulA context:isLocationOf ?uA .
            ?uA context:identifier ?connectedUserA.
            ?ulA context:hasIPAddress ?ipA .
            ?ipA context:IPadress ?connectedIPA .
            ?ac context:connectedClientB ?ulB .
            ?ulB context:isLocatedIn ?rB .
            ?rB context:identifier ?connectedRoomB .
            ?ulB context:isLocationOf ?uB .
            ?uB context:identifier ?connectedUserB .
            ?ulB context:hasIPAddress ?ipB .
            ?ipB context:IPadress ?connectedIPB .
}
```

A context client interested in all the information has to subscribe to the SAInt three times - each time with a different query. The required three web-services subscribe(), unsubscribe() and query() are defined in the standard IContextSource and IContextSubscription interfaces all context sources provide.

Additionally, the IAudio interface defines the 4 web-services provided by SAInt:

- **connect(string clientA, string clientB) –** The *connect()*-method takes two string arguments, the two clients (either two persons, two rooms or one room and one person) to connect, as input. The command to connect those two clients is passed to SAInt's underlying SPARK-engine via inter-process communication.
  If the arguments are valid, an audio connection will be established. Otherwise the connection request will be rejected.

  On success, a new "AudioConnection" with default properties defined in the SPARK configuration (e.g. Gain=0dB, TimeOut=60s, PrivayLevel=10) and a unique connection ID will be added to the ontology model. Moreover, the according clients will be removed from the list of registered users.

- **configureConnection(string connectionID, string configuration) –** The *configureConnection()*-method takes two string arguments as its inputs – the ID of the connection to be configured and a configuration string.
  The configuration string can be either "Timeout,xx", where "xx" specifies the chosen timeout interval, "Privacy,yy", where "yy" specifies the chosen privacy level, "Gain,zz", with "zz" as the desired volume of the current connection or a combination of the preceding three parameters. The command will be passed to the underlying SPARK-engine. On success, the properties for the specified connection in the ontology model will change and all clients will be notified of this change.

- **disconnect(string connectionID) –** The *disconnect()*-method takes a string argument, the connection ID of the connection to be terminated, as input. The command will be passed to the underlying SPARK-engine, which established the connection previously. The specified connection will be terminated, the according entry in the ontology model will be removed and the two clients will be put back to the list of "registered users".

- **disconnectClients(string ClientA, string ClientB) –** The *disconnectClients()*-method takes two strings as input arguments – the names of the users to be disconnected.

Internally SAInt looks-up the connection ID of the communication between the two persons and uses the same mechanism as in the *disconnect()*-method.

Calling the web-methods triggers an appropriate inter-process communication with the SPARK-engine, which processes the commands and updates the context information afterwards.

4.10.3 Using SAInt with the SAIntGUI and the AmigoRFID

A large software project as SAInt requires testing facilities. For this purpose, a simulated RFID reader (AmigoRFID) and a graphical user interface (SAIntGUI) have been implemented as Oscar bundles.

The **AmigoRFID** bundle can be used to simulate the context information an RFID reader would provide. This includes adding new persons to the system and switching rooms.

The context information will be gathered by the Location Management System (LMS), which will notify SAInt of any context changes. The changes will be passed to the underlying SPARK-engine (via IPC) which will create new context information based on the available ACS information and current connections. The new context information will then be returned to SAInt, which will notify its subscribers.

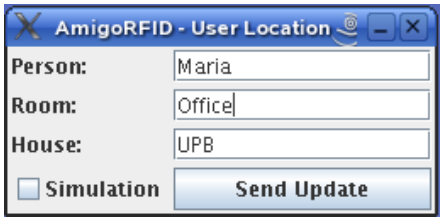

Figure 35: AmigoRFID – a RFID reader context source

The **SAIntGUI** bundle implements a context source that has been designed to use all functionality SAInt provides. As such, the RDF capability (RDF needs) of the SAIntGUI matches those of SAInt. On start-up the SAIntGUI has not subscribed to any context source, but all available context source will be listed at the bottom of the GUI in the <-- *Select a SAInt to subscribe to!* --> field. Subcribing to a SAInt is done by selecting the according entry of that list.

The SAIntGUI will subscribe to the chosen SAInt with all the three queries mentioned in the "**How to query/subscribe to context information**" section. Once subscribed, the SAIntGUI invokes the *query()*-method of the SAInt to get all current context information – this is the initial query. Afterwards the SAIntGUI should look something like that:

All registered users (and rooms they are in) currently not involved in any communication are listed in the top two fields labelled "Client A" and "Client B". The rooms equipped with audio hardware are listed at the bottom of these fields, too.

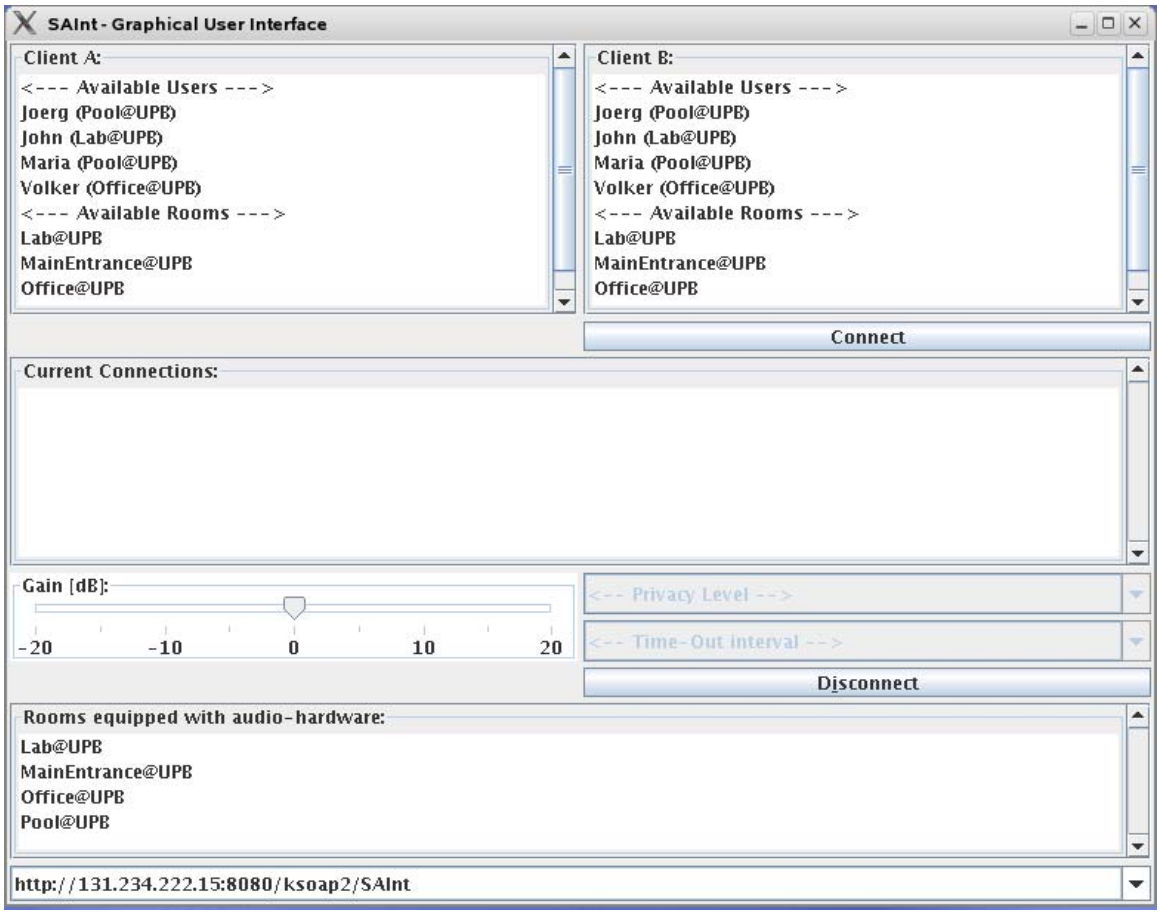

Figure 36: SAIntGUI subscribed to a SAInt

A connection can be established by selecting one client from the “Client A”-field, one from the “Client B”-field and clicking the “Connect”-button, which will invoke the *connect()*-method of SAInt with the chosen clients as arguments.

Connections can be established between rooms, persons or between a person and a room (baby phone functionality). If a connection is established, the according elements will disappear at the client lists and show up in the “Current Connections”-field.

The established connection in Figure 37 is an “internal” one. The persons are in the same domain (here UPB – University of Paderborn) and registered at the same SAInt – the IP-addresses are set to “LocalHost”. For an external communication, the external communication partner’s IP-address will be displayed.

The connection properties of the example are: a timeout of 60s (displayed as 00060), a privacy level that is set to public (PrivacyLevel=0) and the volume set to 0db (displayed as 00000).

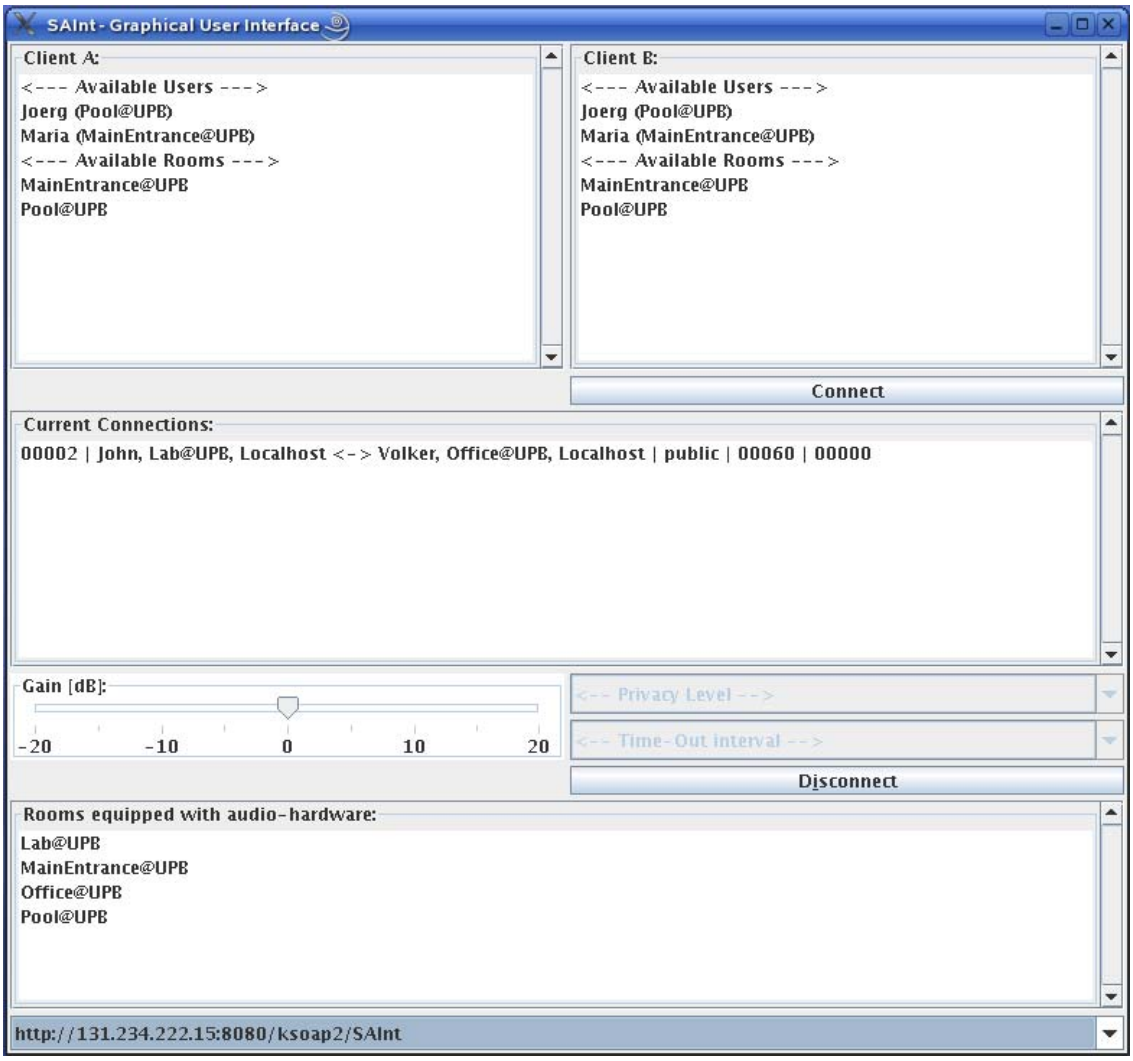

Figure 37: SAIntGUI with an established connection

To change one of the attributes for a connection, the according connection has to be selected from the list. Once selected, the controls *Gain [db]*, *<--PrivacyLevel-->* and *<--Time-Out interval-->* will be available and the according properties can be set/adjusted. Utilizing one of the controls will invoke SAInt's *configureConnection()*-method with either "Timeout,xx", "Privacy,yy" or "Gain,zz" as argument (where "xx", "yy"and "zz" are the chosen values).

A connection can be terminated by selecting the according entry in the connection list and clicking the "Disconnect" button, which will invoke SAInt's *disconnect()*-method with the connection ID as argument. On successful disconnection the connection entry will disappear and the persons and rooms will be put back on the list of available clients.

5 User guide

In this chapter, we adopt end-user perspective and present user guides for WP7 applications. The guide is structured around a walkthrough scenario for an Amigo-enable environment. But the narrative scenario is extended with detailed description of available end-user interfaces and implemented functions. The scenario presented below doesn't show exhaustively use cases enabled by the demonstrator. It is rather an example structuring and facilitating the description of WP7 applications from the user point of view.

Though the user guide is based on a continuous story, this chapter is divided into sections detailing the usage of each application. To simplify the writing of the scenario, we use the following names for the users:

- Maria, the owner of the "local" Amigo environment
- Arthur, Maria's elderly father
- Roberto, Maria's best friend living on the other side of the city
- John, a friend who visits Maria
- Anne, an old friend with whom Maria wants to keep in touch

5.1 PalantirGUI

Maria comes home from work. She is identified by the Amigo system and localized in the entrance hall. The Palantir presence management application was running in the background, and on user detection initializes a PalanirGUI service on the touch-sensitive photo frame in the hall. The PalantirGUI can appear on other displays if the user approaches them.

The PalantirGUI shows the presence status of Maria's friends in a form of a patchwork of pictures. Buddies that are unavailable have washed-out pictures, whereas those who can communicate are presented in color. The size of the buddies' photos corresponds to theirs willingness to communicate.

Maria can control the awareness/presence information the Palantir exposes to her buddies by opening a feed-back panel. The panel is opened by clicking on the background image. It shows the current presence/awareness status of the user as computed by the Palantir, the status exposed to buddies can be modified explicitly by moving two sliders on the sides of the user's picture.

The GUI allows the user to notify a buddy that she/he wants to communicate with her/him. To do so, the user first selects an application (Ambience Sharing, Feeling@, Activity Sharing, etc.) and then clicks on the buddy's picture. On the buddy's PalantirGUI, the image of the caller vibrates to indicate the call. To accept the communication, the called person should click on the vibrating photo of the caller.

Maria decides to see how is her elderly father Arthur. Maria touches the photo of her father on the picture frame. The Palantir in Arthur's home is set to automatically reply when Maria wants to talk to her father, so the audio visual communication is started.

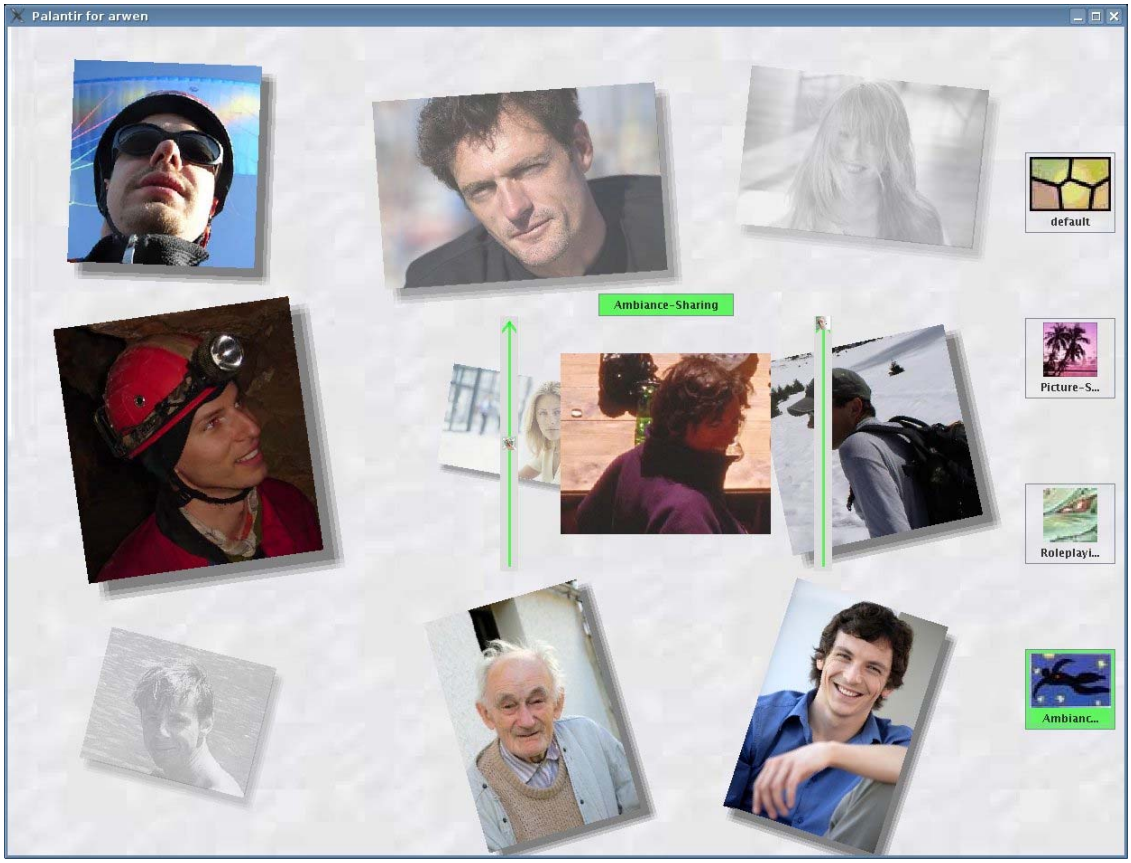

Figure 38: Snapshot of the GUI, while the user is controlling her awareness.

5.2 Ambience Sharing

Maria started the Ambience Sharing application from the PalantirGUI running on a picture frame in the entrance hall. The picture frame shows now a live video stream of her father home. Arthur, the father, is sitting in a chair, but to preserve Arthur's privacy the Ambience Sharing application shows only a ghost image. Maria sees her father as a semitransparent silhouette.

In the meanwhile Maria took off the coat. When she goes to the living room, the live audio and video connection follows her. Whenever there is a display in Maria's proximity the video of Arthur's home is directed to it. At all time, Maria can freely talk and listen to Arthur thanks to the multi-speaker and multi-microphone audio system of her Amigo-enable home.

In the living room, Maria comes closer to the TV set while talking to Arthur. His father stands up and approaches the display on which he can see Maria. As he comes closer his image becomes opaque (similarly to Figure 39).

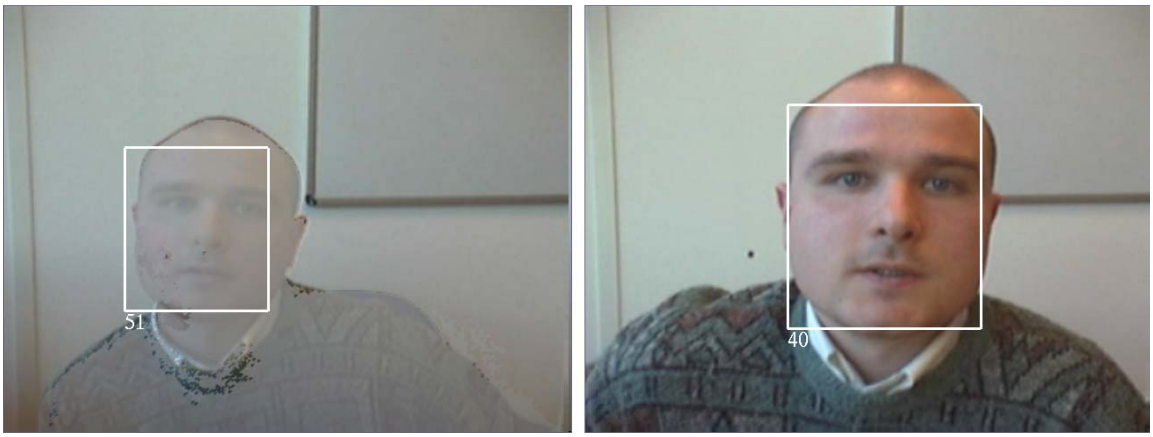

Figure 39 Image of the interlocutor becomes opaque when engaged in a face-to-face communication. Note that the bounding box is displayed only for debug purposes.

Ambience Sharing application also protects the privacy of its users in case if a third person interrupts the communication. In this case, the image of the remote site is hidden until the newcomer appears together with the user in the camera's filed of view.

The Ambience Sharing application is designed to support a quasi-permanent communication link between remote people. Instead of being switched on and off, it adapts the amount of transmitted information to the current context. However, user studies performed in WP1 [Amigo-D1.1, Amigo-D1.2] indicate that users must remain in control and must be able to turn off the system. The Ambience Sharing application can be stopped at any time by pressing the exit button displayed in the upper right corner. When run on the TV, the exit button is mapped to the red button on the TV remote control. Alternatively, the communication can be stopped by starting another application, either from the PalantirGUI, or from the Awareness Globe device (described in section 5.4).

After a chat with her father, Maria stops the Ambience Sharing application by pressing a button on the remote control of the TV. The default TV menu appears.

5.3 Activity Sharing

Maria has recently been on holiday in Spain, and would like to browse her pictures immediately on the TV. Although she is at home alone, she wants to share this experience with her best friend Roberto, who also happens to be at home, at the other side of the city. Maria turns on the TV to check whether Roberto is available.

The TV's top-level menu has an item named “Activity sharing”. Maria enters that menu item and sees an overview of her family and friends, including their availability. Maria selects Roberto (who fortunately also happens to be at home, near his connected TV) to contact for the “Photo sharing” application. In the meantime Maria has connected her camera, which holds the holiday pictures, into the TV. Now Maria and Roberto can select pictures that they would like to watch together; besides, they can see and hear each other through a direct connection to discuss the holiday and Maria's pictures. Roberto looks up some of these photos from his own earlier trip to Spain, so they can compare them. Roberto likes Maria's new pictures (they are better than his own) and copies some of them to his own local system (after Maria's consent).

After they finished the photo slides, Maria and Roberto decide to play a quick game before they go to other activities; 4-in-a-row. Maria wins the game. Roberto looks really sad into the camera and yells: “Good night!” before he turns off the TV.

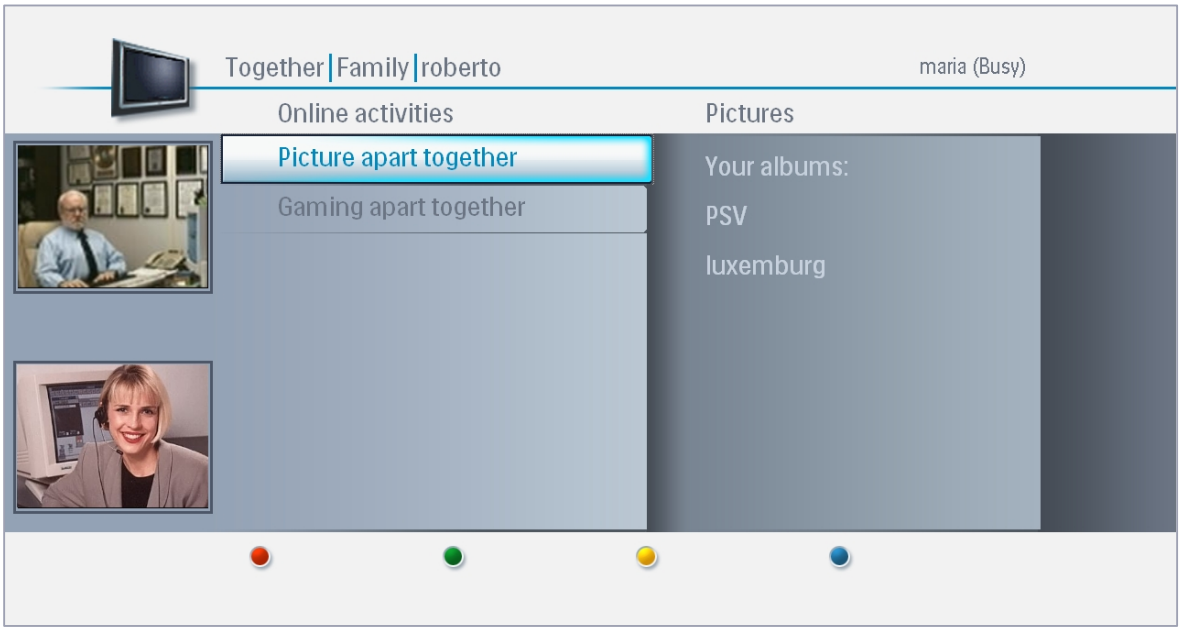

Figure 40: Screenshot of TV interface (video call active, selection of shared activity)

For the two shared activities applications, shared photo browsing and shared gaming, the functionality and the major elements of the associated GUIs are elaborated below. Both of them are designed for interaction with a standard remote control on a TV screen.

<u>Shared photo browser (Picture Apart Together)</u>

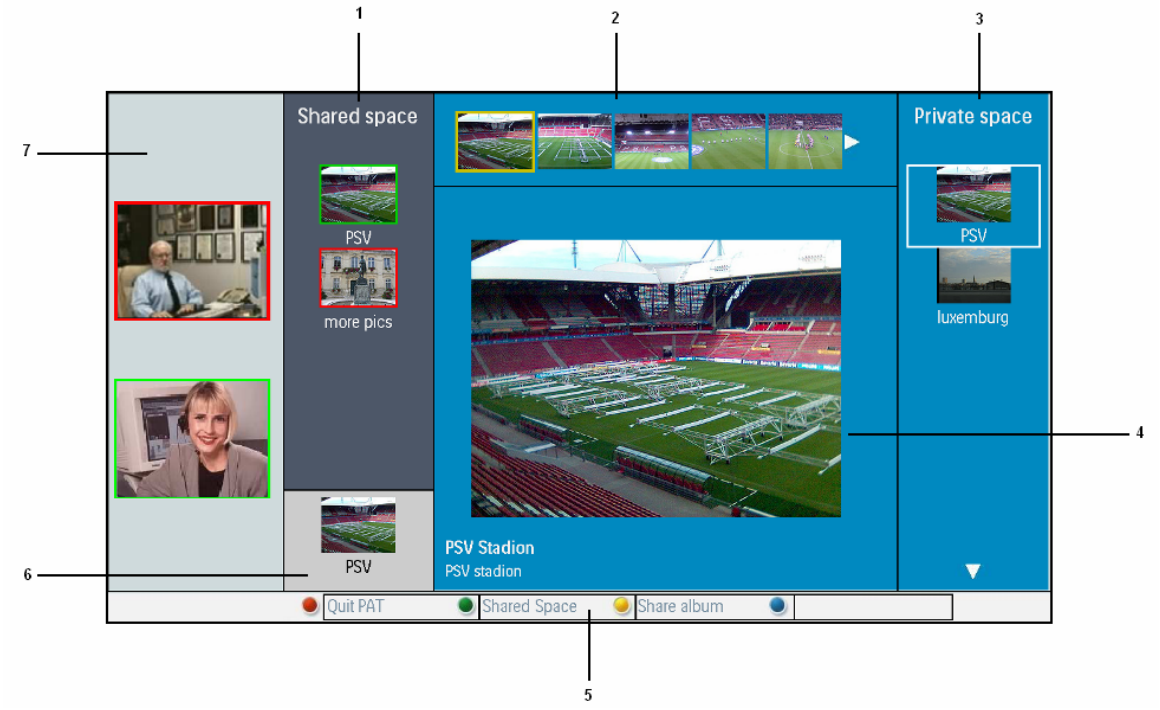

Figure 41: GUI for shared photo browser

1. Shared space: The shared space is almost identical for both users. The only difference is the border color of the albums' thumbnails. The album border of the local user is green, whereas the border is red for the remote user album. The shared space can display up to 3 albums at a time. The green and red album borders correspond to the colored borders of the video call images (in the left column).
2. Thumbnails: Displaying a scrollable view of all photos in the selected album.
3. Private space: The private space shows albums owned by the local user. These albums are not visible for the other party (unless they are shared).
4. View: The view shows an image of 500 x 375 pixels and a description of the selected picture. This part of the screen is identical for both users while sharing pictures.
5. Colored keys: The colored keys have context sensitive functions; only buttons with a label have a function at that time. The red-button is used to stop the shared photo browsing “Quit PAT”. The green and yellow-buttons are “Shared space” & “Share album” when the user is watching the private space, and “Private space” & “Remove album” when the user is watching the shared space.
6. Show Position: This thumbnail shows the image that the other party is watching concurrently. This is useful when the other party is in the shared space, while the local user is in the private space.
7. Video call: The video call that was initialized when the connection was established is visible on the left side of the GUI.

<u>Shared lean-back game: 4-in-a-row game</u>

The 4-in-a-row interface is shown in Figure 42. The GUI presents the current status of the game for both users. On the left side, video call is integrated like in shared picture browsing. The border colors of the video call images (red and yellow) match the colors used in the game.

The user can only use the dish (left-right buttons) on the remote control if it is his turn. If the other party has its turn, arrow buttons (left-right) for the local user are ignored. The color buttons on the remote can be used for functions like starting a new game (without finishing the old one), and finishing the shared gaming activity.

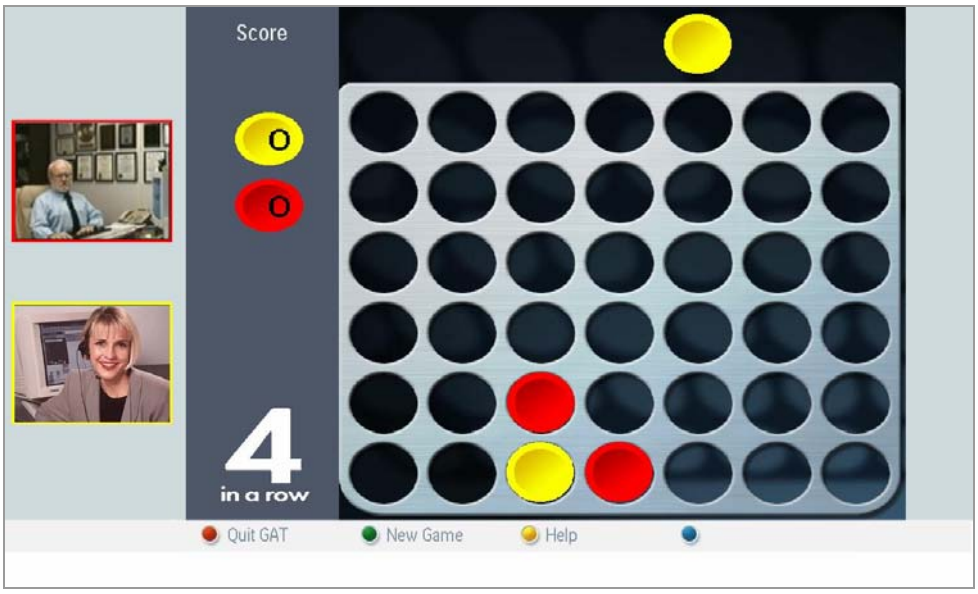

Figure 42: GUI for the 4-in-a-row shared game

Add user:

The Activity Sharing application allows the registration of new users all done by means of the colored keys on the RC. To create a new account on the TV the user enters his nickname, password, and email address, after which a 'friend code' is automatically generated and returned on screen (see Figure 43, Figure 44, Figure 45). At this point the user is centrally registered, but none of user's contacts knows about it. To communicate using the Activity Sharing suite, users must exchange their friend codes.

Figure 43: Activity Sharing: create new user

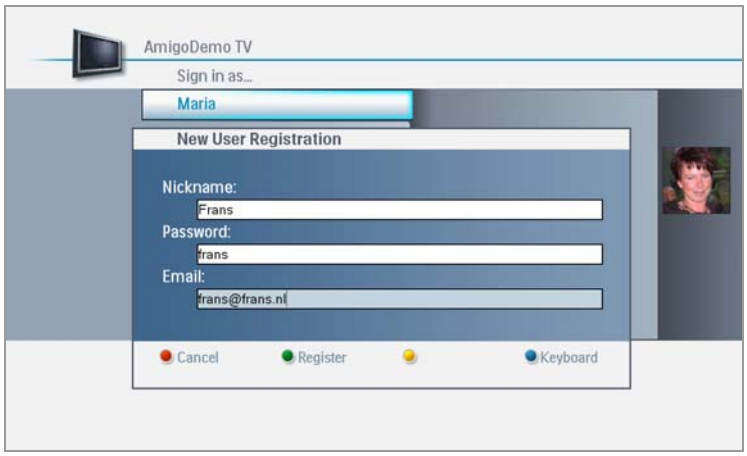

Figure 44: Activity Sharing: enter personal data of new user Frans

Figure 45: Activity Sharing: returned Friend code

Once users exchange their friend codes, they can add each other to their list of friends (see Figure 46, Figure 47). The system requests a friend code, which is entered with the RC. The next and final step in this procedure is to select the community in which the new friend Maria/Frans will be placed (Friends/Family/Others). Maria and Frans now are friends and can do any activities together.

Figure 46: Activity Sharing: add New friend

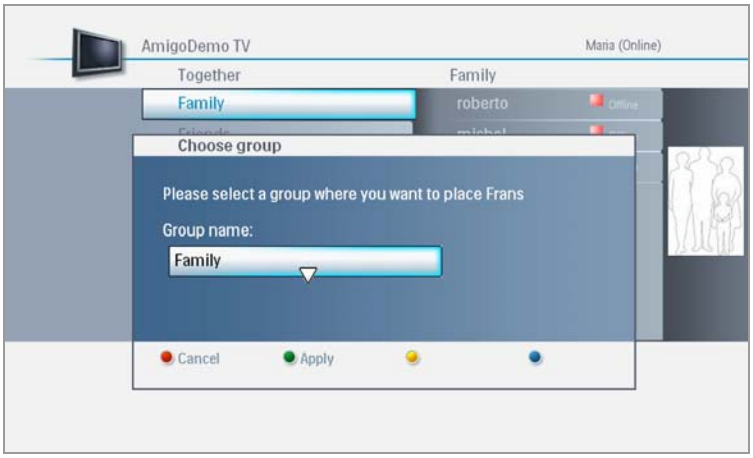

Figure 47: Activity Sharing: select community

Add images:

The Activity Sharing application allows adding images from digital photo cameras and memory cards. When a user connects a camera to the TV, the camera is automatically detected, and images are available for upload. The user is requested whether or not she/he would like to upload the pictures. The pictures can be uploaded to a new or an existing folder. Once the images are uploaded, they are available for sharing with any of the community members.

Figure 48: Activity Sharing: USB key found

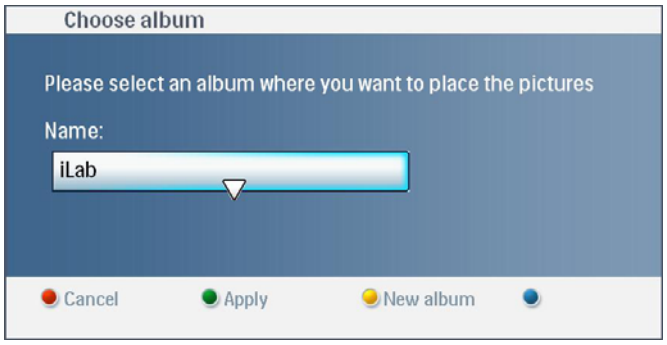

Figure 49 : Activity Sharing, picture management

Launch Poker/Palantir:

From the EasyLogic main menu the user can select a menu item named Home Content. Entering this menu automatically triggers CMS to search for Amigo devices that are registered and lists them in the EasyLogic interface. The submenu that appears shows the registered services, Poker Game and Palantir. After selecting one of both services, it is launched automatically in full-screen mode. Please refer to the User Guides of those applications for detailed functionality.

Figure 50: Screenshot of TV interface, including Home Content

5.4 Awareness Globe

In her house, Maria can pick up the Awareness Globe device (running on a touch sensitive photo frame) and see her contacts' status. The Awareness Globe can be used to start shared activities, such as those described in section 5.3.

To initialize the Awareness Globe, the user must pick up the photo frame, which by default shows photos from family album, and touch the screen.

Figure 51: Standard photo frame view

After this, the user is presented with the log-in screen. The log-in screen shows all registered users for this device and service in the center of the screen. The message bar at the bottom can show messages for the users

Choose yourself as the current user of this device on the log-in screen, by this action the current user makes him/herself available in the shared activities environment of the AG2.

Figure 52: AG2 login screen

The main screen (Figure 53) shows user status, activities and location of the current user and his/her contacts in his 'Circle of friends'. This is the main screen of the Awareness Globe 2 UI.

On top of the screen the Activity bar shows all shared activities the current user can start up with his contacts. These shared activities are placed in 4 categories:

5. Games (play games together)
6. Media (share media, e.g. picture sharing)
7. Communication (e.g. chat)
8. Services (all other services that are related to activity sharing)

The center of the screen shows the 'Circle of friends', on which the current user is selected (shown up front at the bottom). Several contacts can have a certain status indicator attached to their image. The message bar shows messages if available or necessary.

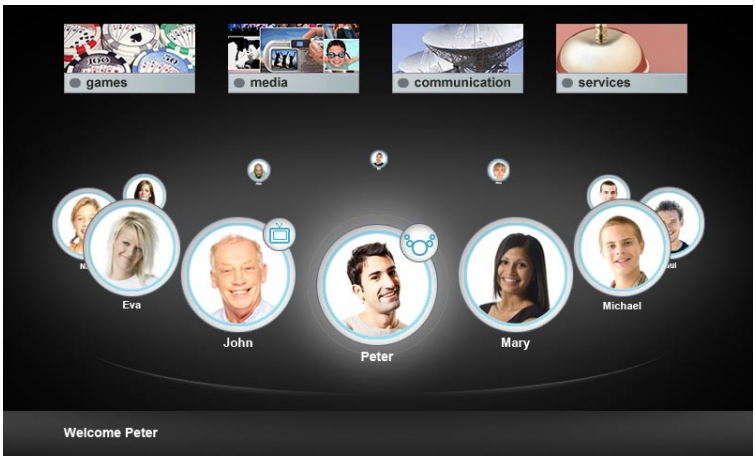

Figure 53: AG2 main screen

The current user can browse his/her 'circle of friends' via touch interaction, swiping from left to right. As well, the current user can directly press on a contact.

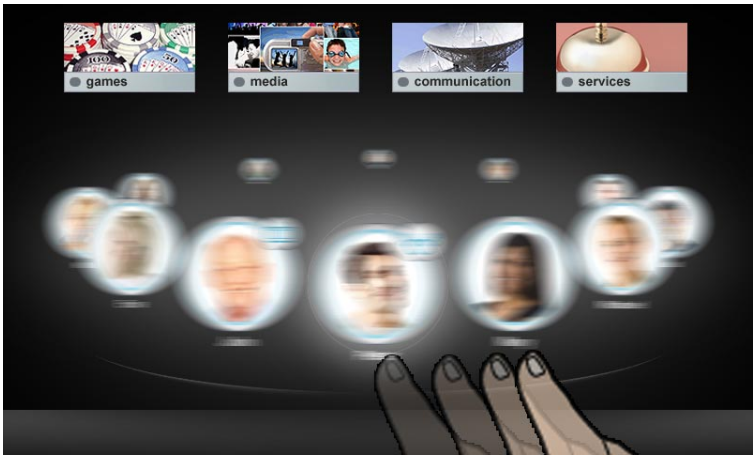

Figure 54: Browsing the AG2

The current user can select contact per contact, while seeing this contacts user status and if available and allowed to share, the activity and/or location of this contact. As well, the current user can see which shared activities are available per contact (filter per contact). Those categories not available are shown dimmed in the Activity bar and cannot be selected.

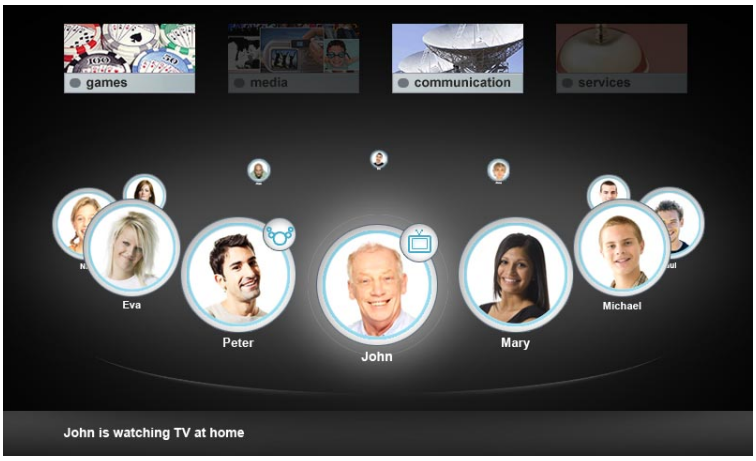

Figure 55: AG2 - Filter per contact, where only available Activity groups are shown highlighted.

This contact can be invited to a shared activity, by selecting a shared activity from the Activity bar on top of the screen. After an Activity is chosen, the selected contact is invited to this shared activity; this is the first interaction path to start a shared activity. The following interaction is as further described in the second interaction path to start a shared activity.

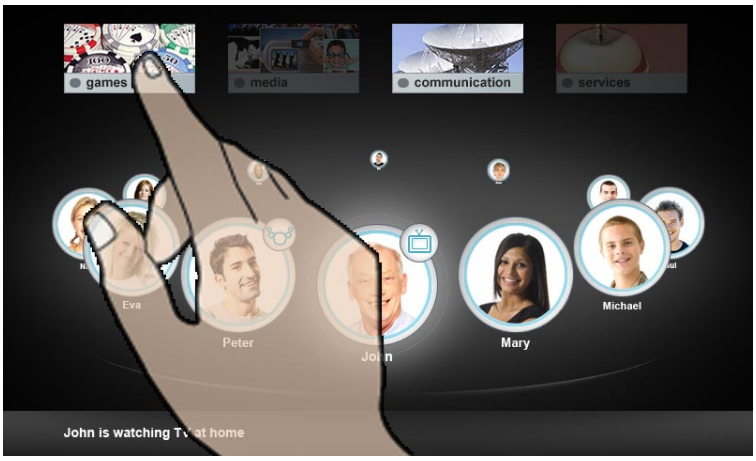

Figure 56: Selecting an activity group to invite a selected contact to

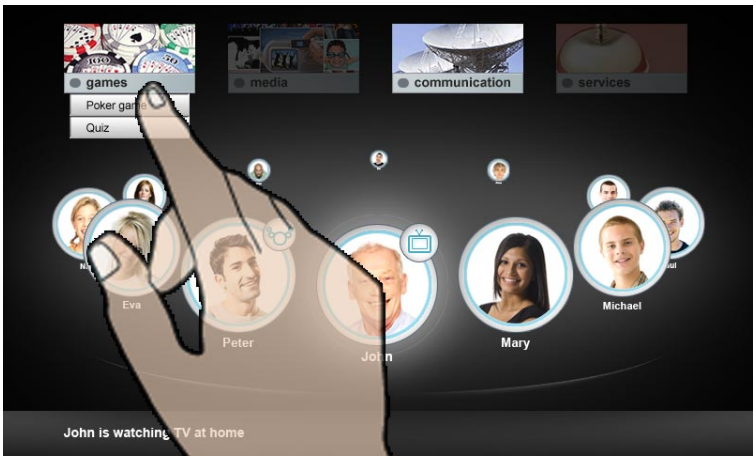

Figure 57: Select a specific activity from the drop-down box

Besides filtering per contact, the current user can also select him/her and browse through his/her shared activities groups. Per activity group the available users are shown (filter per activity, which is the second interaction path to start a shared activity).

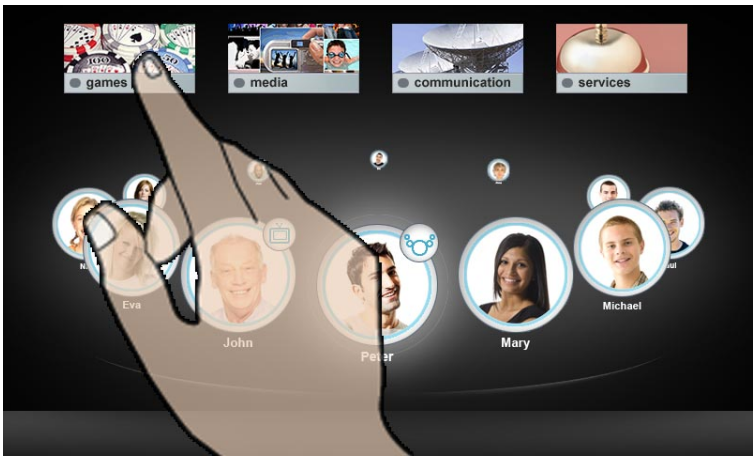

Figure 58: Selecting current users' activities

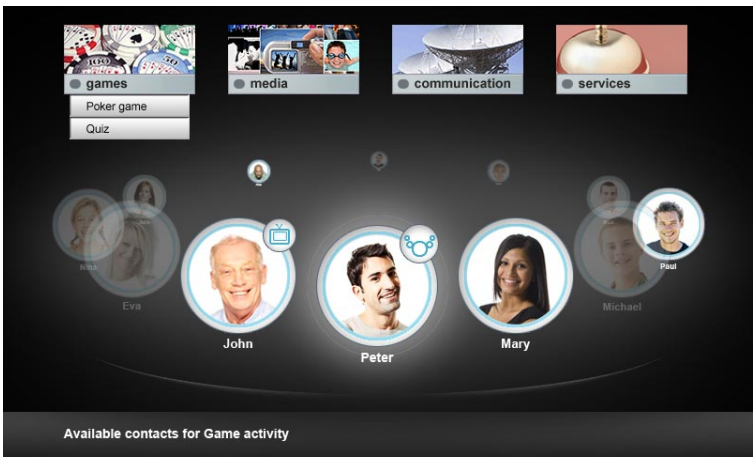

Figure 59: Filter per activity

These contacts can be invited to a shared activity chosen from the drop-down menu. First the user has to select a specific activity (e.g. Poker game). This takes the user to the Poker game launch area, where he can invite players and launch the game from. (Main interaction path 2)

In the ‘Circle of friends, those people available for Poker are shown with a status icon (top-right)

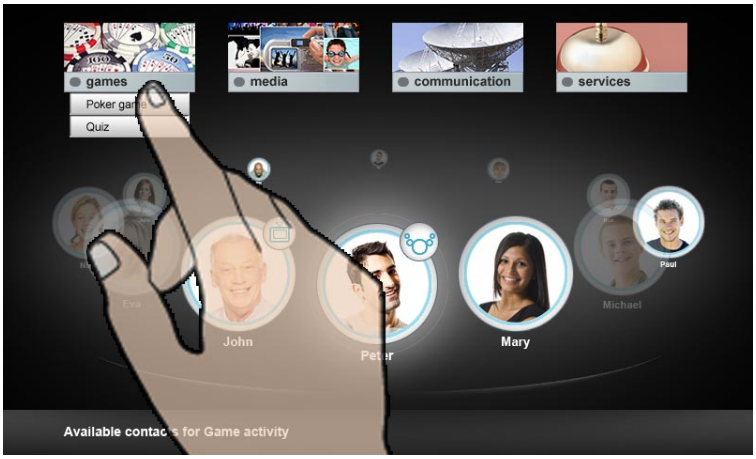

Figure 60: Select a specific activity from the Game group

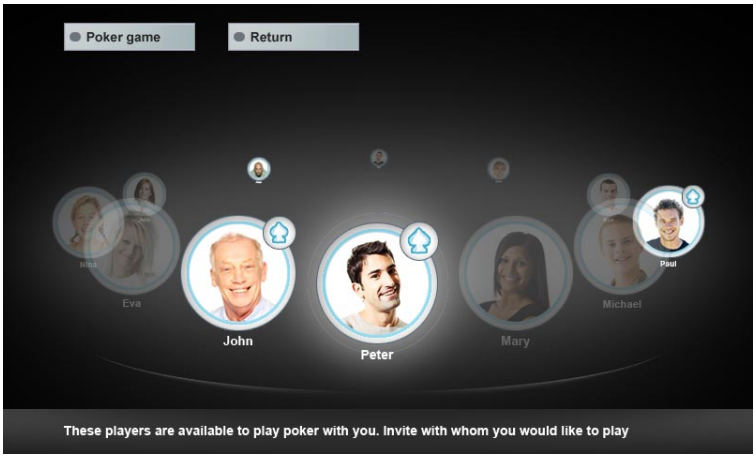

Figure 61: Launch screen of a shared activity

On this ‘launch’ screen, the current user can select a contact, who will receive an invite.

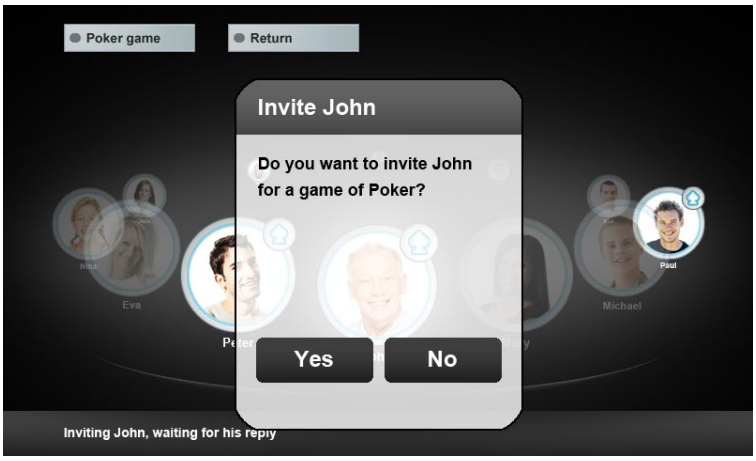

Figure 62: Invite contacts to play a game of Poker

After the contact accepts this invite he/she is taken to the poker table. The current user is able to launch the game by swiping vertically on the Photo frame towards the TV. This will take the current user to the poker table on the TV.

Figure 63 The Poker Game interface

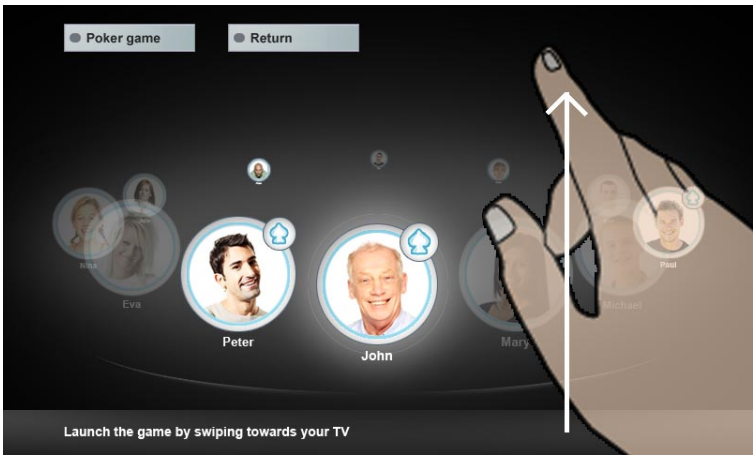

Figure 64: Swiping

Finally, the current user is able to invite other contacts like shown before. If this is done the current user has to start the game by pressing a button on the Photo frame.

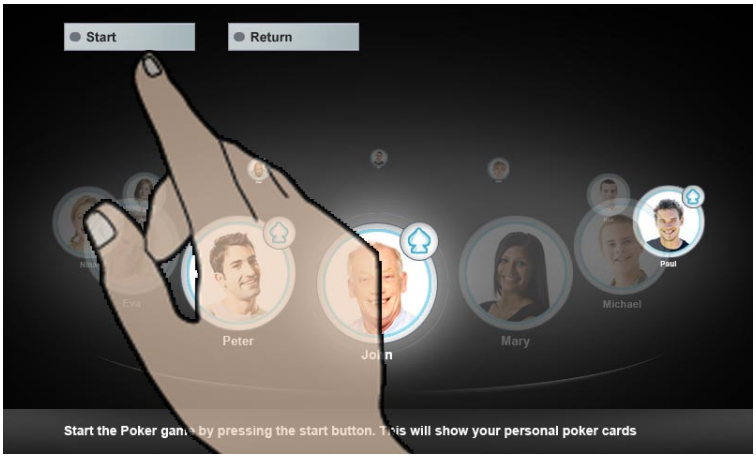

Figure 65: Start the game

After this, the Photo frame moves out of Awareness Globe mode and shows his personal cards (the GUI of the selected application).

5.5 Personal Amigo Device

Maria finished showing her holiday pictures to Roberto just before the doorbell ring. It is John, a fired of Maria who came visiting her. They have a chat about her holidays and about Spanish music. John has some rare flamenco records at home, he propose Maria to listen to them.

John identifies himself to Maria's Amigo system by swiping his personal RFID tag over the home reader. John wears the Personal Amigo Device (PAD) that now establishes a secured link between his and Maria's homes. The PAD also exports John's media to the Media Manager running in Maria's home, so he can play John's music files.

The Personal Amigo Device is not an end-user application, but rather an enabler for sharing intelligent (web) services between homes. The way this is achieved is explained in sections 2.10, 3.10, and 4.9. When all applicable middleware is in place, all the user has to do is swipe the PAD across the RFID reader to initiate the association between the homes. For using the exported services please refer to the appropriate documentation of that service itself. For the Media Manager Core please refer to D6.3 and D6.4.

5.6 Social Radio

While chatting with John, Maria noticed that one of the Social Radio (SR) artifacts lying on the table in the living room lit. She understood that her friend Anne came home. Once john left, Maria decides to put the SR artifact in the "bridge" position, so she could here the music Anne is listening to. Maria continues her home activities while keeping a subtle link providing her with clues about Anne's presence and mood.

5.7 Feeling@

Maria receives a notification message from her colleagues from work. They finished a sketch of a new car model and they ask Maria her opinion. Since the car project is urgent, Maria goes to the study, where she starts the Sketch Presentation application from the PalanirGUI service.

She can now see the various versions of the car design proposed by her colleagues. Video conference with the office is also started. She discusses with them their work and interacts with the application using voice commands and hand gestures.

While the message received from work was send by Maria's colleagues intentionally, the Feeling@ application is able to send alerts automatically when unexpected events happen at home. For instance, Maria can be notified of an unexpected homecoming of her son while she is at work.

The Feeling@ application is composed of two front end user interfaces, the Shared Organizer, which enables users to create, modify and share personal or working activities, and the Sketch Presentation, which allows local and remote picture viewing and comparison.

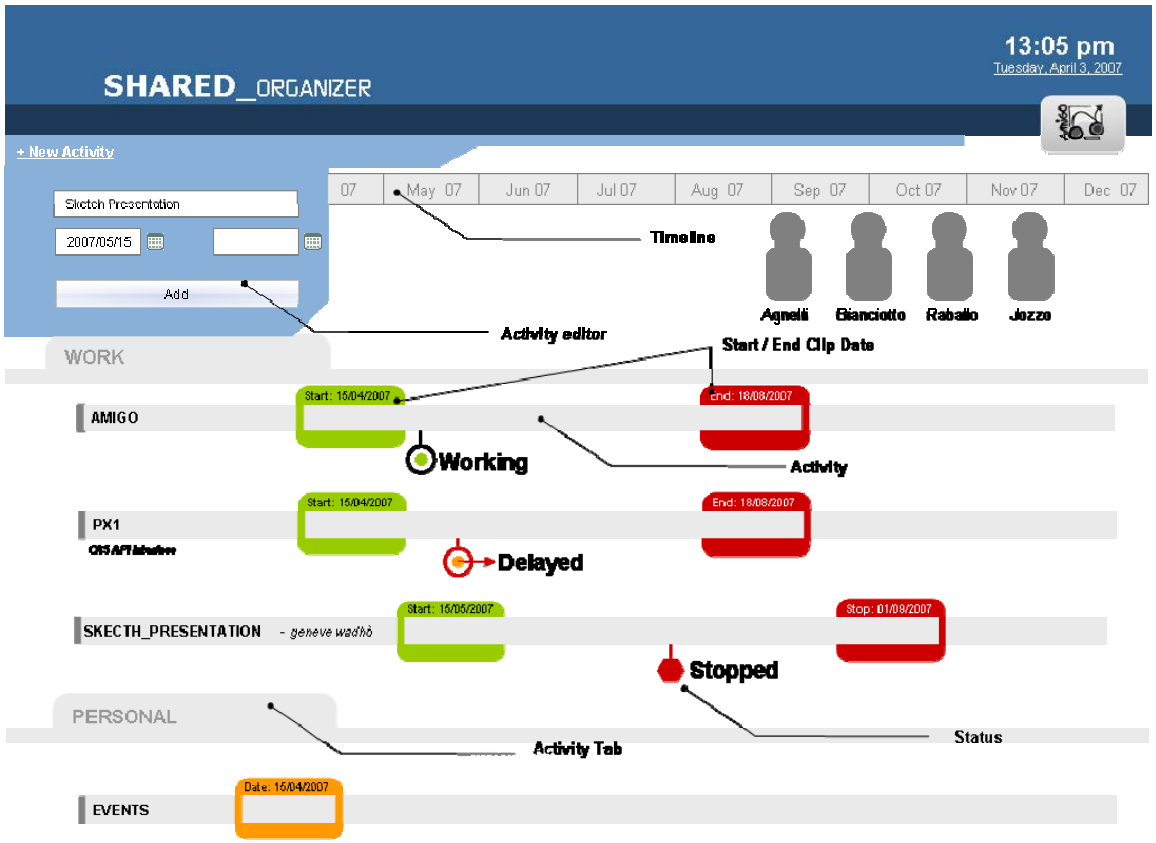

Figure 66 Shared Organizer GUI

The Shared Organizer graphical user interface is fairly simple. Under the timeline bar, active users are listed. Two tabs, “Work” and “Personal” contain activity information and can be opened or collapsed one at a time or simultaneously with simple mouse clicks on their titles. In the upper left corner a link opens an overlay panel which enables new activity creation. Existing activities are represented as bars with start and end data clips along with their current status. By clicking on the activity bar an overlay panel appears which enables activity editing.

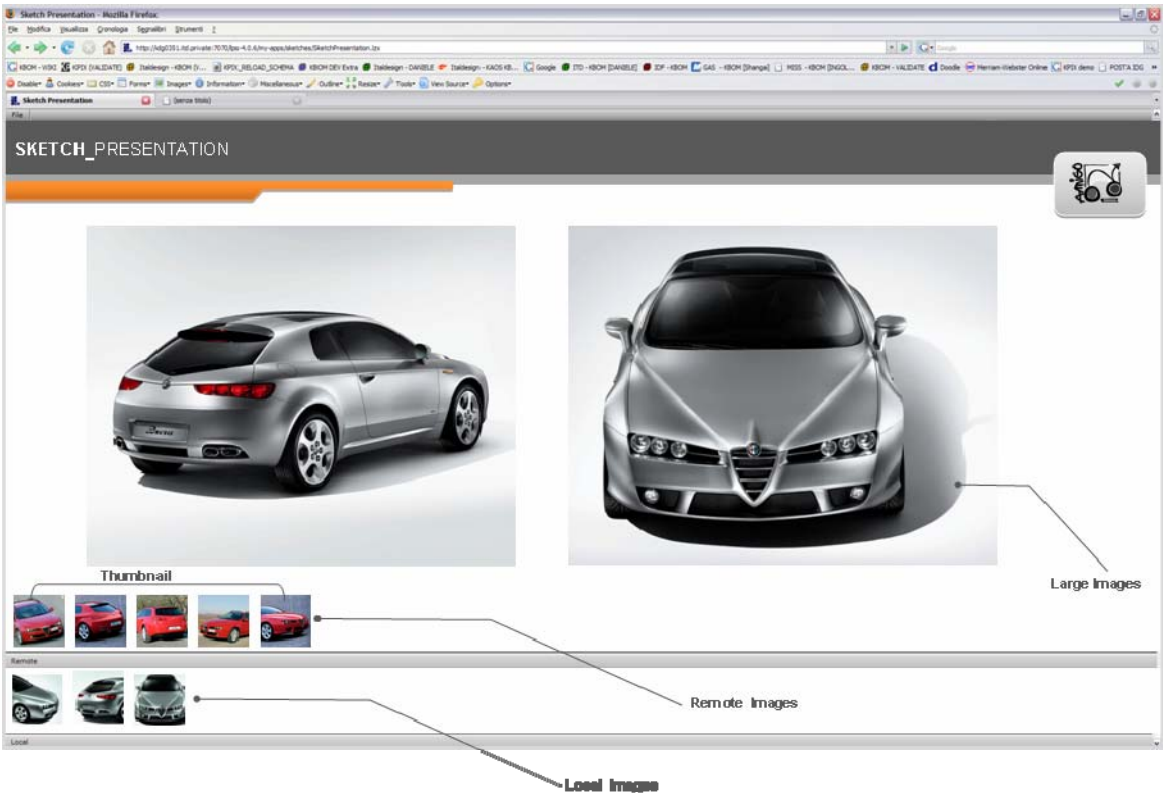

Figure 67 Sketch Presentation GUI

The Sketch Presentation user interface is even simpler. Two tabs at the bottom of the screen contain image thumbnails. The tabs are collapsible individually or simultaneously. The user can select each single thumbnail, shift forward or backwards (left and right) between them or up and down between tabs. Selected thumbnails can be instructed to appear in the upper screen area at bigger dimensions. Images can be shown enlarged on the left, right or center of the upper screen.

Sketch Presentation allows the following gesture or voice commands:

- **Next:** shifts focus on the next right image on the active thumbnail bar;
- **Right:** same as Next;
- **Previous:** shifts focus on the previous left image on the active thumbnail bar
- **Left:** same as Previous;
- **Show Left:** displays the currently selected thumbnail on the left side of the screen hiding any previously positioned image;
- **Show Right:** displays the currently selected thumbnail on the right side of the screen hiding any previously positioned image;
- **Fullscreen:** displays the currently selected thumbnail on the center of the screen hiding any left or right images;
- **Select Local:** moves focus to the lower thumbnail bar;
- **Down:** same as Select Local;
- **Select Remote:** moves focus to the upper thumbnail bar;

- **Up:** same as Select Remote;
- **Open:** opens the currently selected thumbnail bar showing the contents;
- **Close:** closes the currently selected thumbnail bar hiding the contents.

6 Conclusions

This document provides a summary of the results of the implementation of WP7 applications for the extended home domain. It describes the applications building blocks, their interactions, provides installation and deployment instructions, and end-user guides.

WP7 applications extend in various ways the home environment for both *interpersonal communication* and *shared activities*. We produced a whole palette of prototypes having different levels of complexity, and exploiting Amigo software more or less deeply. The Activity Sharing application suite is implemented using conventional technologies, with a short-term exploitation oriented approach. This application focuses on supporting activity sharing using conventional television sets. It can also be considered as an illustration of current trends in the TV-set market and of how easy it is to capitalize on data provided by Amigo services deployed in the environment, e.g. user presence data produced by the Palantir service. Contrary to Activity Sharing, applications such as the Feeing@ or the Ambience Sharing, were built almost exclusively on top of the Amigo middleware and Amigo services. They were designed to be distributed and integrated with the environment, and to adapt and follow the activities of users. These applications are more than just a showcase for middleware technology: they point the way towards the future evolution of communication, when it becomes much more than just a time-metered commodity or the access to pre-packaged synchronous content.

The WP7 applications vary also in their approach to interaction with users. The Activity Sharing, Awareness Globe and Palantir applications support mostly conventional explicit interactions through a graphical user interface, where the user must perform an intentional action oriented towards the system to achieve an explicitly desired goal. The Board Game and Feeling@ applications offer additionally the possibility to interact using other modalities, i.e. voice commands and gestures. Finally, the Social Radio and Ambience Sharing applications show examples of enaction and implicit interaction. In fact, these two applications focus on mediating as transparently as possible social interactions between humans, instead of on enriching interactions between humans and communication systems

WP7 produced not only application prototypes, but also services and service oriented architecture solutions that can be reused and further improved by other projects or developers. The Personal Amigo Device is a complete software and hardware solution build on top of the WP3::Security services and the WP4::Context Management System for seamlessly and securely accessing services and context data between different Amigo domains. The Palantir service together with the ResourceManager and Scheduler services provide a SOA solution enabling presence and location awareness for applications. This solution is used by the Ambience Sharing application to show dynamic location-aware service composition in an audio-video "follow-me" function. Also the XMPP-based content sharing service conceived for the Activity Sharing application, and later picked-up and further developed by WP4 task 4.8, provides a convenient data exchange framework for distributed applications.

WP7 applications extensively use the development and deployment frameworks developed in WP3, as well as the WP4 Context Management System. The collaboration with the two middleware work packages was challenging because of the overlapping schedule. Especially at the beginning of WP7, the software provided by WP3 and WP4 was still under development, and documentation was not always complete. On the other hand, this situation allowed us to influence the work plan of the middleware WPs and to have the necessary support directly from the creators when integrating their software. Apart the components mentioned above, WP7 application use WP3::semantic service descriptions, WP3::Vantage Point for setting up the environment and for test purposes, WP4::UMPS for user preferences retrieval, WP4::ANS for interfacing the CMS, and the WP4::UIS for voice command and gesture interfaces.

Obviously there are Amigo middleware components that we do not demonstrate, e.g. WP3::Domotic services, WP3::Accounting and billing, WP3::Semantic Service Composition, but these are either demonstrated by WP5 (Domotic services) or WP6 (Accounting and billing) or did not meet the requirements of our demonstrators. For instance, the WP3::Semantic Service Discovery and composition system could not be integrated with the demonstrators because it does not include service location as a criterion for selecting a service. Nevertheless, WP7 applications demonstrate a fair number of Amigo components in a variety of usage contexts.

In summary, WP7 demonstrates an important number of application prototypes that can be either shown independently or in a common environment. The applications illustrate a variety of middleware use cases and re-usable services for software developers, as well as of functions and interactions for the end-user. The experience and the results of the work package improved our understanding of service-based systems and provide a solid base for further investigations and developments within the work package partners companies.

Acronyms

ANS	Awareness and Notification Service
AG2	Awareness Globe 2
IUS	Intelligent User Service
MW	Middleware
J2EE	Java 2 Platform Enterprise Edition
J2SE	Java 2 Platform Standard Edition
RFID	Radio Frequency Identification
CD	Content Distribution
CMS	Context Management Service
BG	Board Game
GUI	Graphical User Interface
IUS	Intelligent User Service
MMC	Multimedia Manager Core
MW	Middleware
SIP	Session Initialisation Protocol
XMPP	eXtensible Messaging and Presence Protocol
UMPS	User Modelling & Profiling Service

References

[Amigo-D1.1] Amigo Consortium. Deliverable D1.1: “Amigo User Research Results”. January 2005.

[Amigo-D1.2] Amigo Consortium. Deliverable D1.2: “Report on User Requirements”. April 2005.

[Amigo-D7.1] Amigo Consortium. Deliverable D7.1: “Functional specification and test Plans”. March 2006.

[Amigo-D7.2] Amigo Consortium. Deliverable D7.2: “Extended Home Application Specification”. August 2006.

[Amigo-D7.3] Amigo Consortium. Deliverable D7.3: “Implementation of Building Blocks Functionalities”. April 2007.

[amigo-OS] Amigo Open Source Repository. http://amigo.gforge.inria.fr/home/index.html

[amigo] /svn+ssh://login@scm.gforge.inria.fr/svn/amigo